



↑ Shift

BIG DATA COMPUTING

Prof. Rajiv Misra
Computer Science
and Engineering
IIT Patna



INDEX

S. No	Topic	Page No.
	Week 1	
1	Introduction to Big Data	1
2	Big Data Enabling Technologies	31
3	Hadoop Stack for Big Data	51
	Week 2	
4	Hadoop Distributed File System (HDFS)	79
5	Hadoop MapReduce 1.0	107
6	Hadoop MapReduce 2.0 (Part-I)	113
7	Hadoop MapReduce 2.0 (Part-II)	129
8	MapReduce Examples	139
	Week 3	
9	Parallel Programming with Spark	163
10	Introduction to Spark	201
11	Spark Built-in Libraries	230
12	Design of Key-Value Stores	239
	Week 4	
13	Data Placement Strategies	256
14	CAP Theorem	269
15	Consistency Solutions	283
16	Design of Zookeeper	293
17	CQL (Cassandra Query Language)	339
	Week 5	
18	Design of HBase	360
19	Spark Streaming and Sliding Window Analytics (Part-I)	381
20	Spark Streaming and Sliding Window Analytics (Part-II)	405
21	Sliding Window Analytics	430
22	Introduction to Kafka	447
	Week 6	
23	Big Data Machine Learning (Part-I)	473
24	Big Data Machine Learning (Part-II)	495
25	Machine Learning Algorithm K-means using Map Reduce for Big Data Analytics	563
26	Parallel K-means using Map Reduce on Big Data Cluster Analysis	586

Week 7

27	Decision Trees for Big Data Analytics	596
28	Big Data Predictive Analytics (Part-I)	635
29	Big Data Predictive Analytics (Part-II)	658

Week 8

30	Parameter Servers	695
31	PageRank Algorithm in Big Data	716
32	Spark GraphX & Graph Analytics (Part-I)	732
33	Spark GraphX & Graph Analytics (Part-II)	761
34	Case Study: Flight Data Analysis using Spark GraphX	795

Lecture – 1

Introduction to Big Data

Introduction to Big Data

Refer slide time: (0:16)

Preface

Content of this Lecture:

- In this lecture, we will discuss a brief introduction to Big Data: Why Big Data, Where did it come from?, Challenges and applications of Big Data, Characteristics of Big Data i.e. Volume, Velocity, Variety and more V's.



Preface, content of this lecture, in this lecture, we will discuss a brief introduction to the Big Data, why the big data, where the big data comes from? The challenges and the applications of Big Data, the characteristics of the big data, that is in terms of volume, velocity, variety and many more V's. We are going to see in this part of the lecture.

Refer slide time: (0:43)

What's Big Data?

- **Big data** is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications.



- The challenges include **capture, curation, storage, search, sharing, transfer, analysis, and visualization**.
- The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to **"spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions."**

What, what is a big data? So big data is a term for a collection of data sets, so large and complex that it becomes often difficult to process using traditional data processing applications. Now, to see this particular picture here, a consultant is saying, that there are three continents byte of data, are being created every day in an organization. So, it comes from everywhere it knows all and according to the book of Wikipedia, its name is the big data. So, this is a simple way of explaining a big data using, this particular picture which represents only one aspect that is called volume or a size, which is very big, we will see more such challenges in terms of big data in this particular lecture. So, such a huge volume of particular data, poses various challenges, which includes how to capture such a big amount of data how do you cure it? How do you store such big amount of data? How can you search? And how do you share this information? And how to perform the transfer of this huge volume of data? Also we will see, other further challenges like doing the analysis, analytics and its visualization, which is often Lee they are useful too for many applications, where this big data is going to be used. Now, the trend to this larger size of data sets, is due to this additional information, which are derived from the analysis of a single set of large related data as compared to the smaller sets, with the same total size of the data, this large size will allow the correlations to be found to various opportunities to exploit in terms of spot business trends, to determine the quality of research, to prevent the diseases, to link the legal citations, to come back the crimes and determine the real time roadway traffic conditions. So, given this particular a big data or a large size of data, and it will pose various opportunities and challenges and new, kind of applications, and also social service and is possible, that we are going to see, that is why the big data is going to become popular.

Refer slide time: (3:52)

Facts and Figures

- **Walmart** handles 1 million customer transactions/hour.
- **Facebook** handles 40 billion photos from its user base!
- **Facebook** inserts 500 terabytes of new data every day.
- **Facebook** stores, accesses, and analyzes 30+ Petabytes of user generated data.
- **A flight generates** 240 terabytes of flight data in 6-8 hours of flight.
- **More than 5 billion people** are calling, texting, tweeting and browsing on mobile phones worldwide.
- **Decoding the human genome** originally took 10 years to process; now it can be achieved in one week.
- **The largest AT&T database** boasts titles including the largest volume of data in one unique database (312 terabytes) and the second largest number of rows in a unique database (1.9 trillion), which comprises AT&T's extensive calling records.

Some of the facts and figures, of this particular size of the data or a big data. So, we are going to see into it for example Walmart is a company, which handles 1 million customer transactions per hour. So ,just see that here it deals with the volume or the rate, in which these transactions or the customers are handled, the second such company is which is called a Facebook, which handles 40 million, 40 billion photos from its user base. So, when you say photos that mean now here, the data is in different form and also of large size. So, to dimension of the complexity is being added now, Facebook here the inserts 500 terabytes of new data every day. So, this basically becomes the volume challenge, Facebook stores, accesses, analyzes, 30+ petabytes of user-generated data, every day. Now, similarly a flight generates 240 terabytes of flight data in 6 to 8 hours of flight to make the customer safe, flight and also to basically ensure the, the comforts, during the flight journey. So, that is why this particular flight generates and uses this information for the analysis and providing the solutions, similarly more than 5 billion people, are calling, texting, tweeting, browsing, on their mobile phones, worldwide. so ,here the people are involved in generating the big data. Another thing is about, the decoding the human genome. so, originally it took 10 years to process it. Now, it can be achieved in one week .that means the computations of a big data is now becoming possible, to be completed within their time now, another company which is called AT&T databases, which boots, the titles, including the largest volumes of data, in one database that is 312 terabytes and the second largest number of rows, in a unique database that is one point nine trillion, which comprises AT&T's extensive calling records. So, this particular company is this example, which uses the large size databases ,that is the data which are at the store and then it performs, it has to perform the computations on this large sized data set, to gain the insight and basically drive the business of that company.

Refer slide time :(6:51)

An Insight

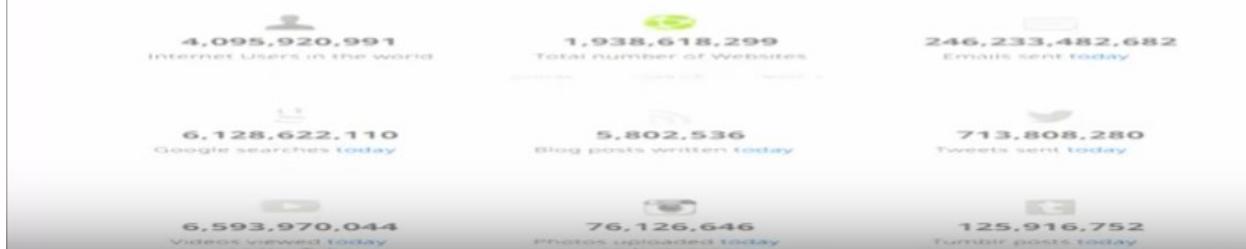
- **Byte:** One grain of rice
- **KB(3):** One cup of rice:
- **MB (6):** 8 bags of rice: Desktop
- **GB (9):** 3 Semi trucks of rice:
- **TB (12):** 2 container ships of rice Internet
- **PB (15):** Blankets $\frac{1}{2}$ of Jaipur
- **Exabyte (18):** Blankets West coast Big Data
Or $\frac{1}{4}$ th of India
- **Zettabyte (21):** Fills Pacific Ocean Future
- **Yottabyte(24):** An earth-sized rice bowl
- **Brontobyte (27):** Astronomical size

Now, let us see the volume, with an insight that one, if we consider that the byte is a one grain of rice and a kilobyte, that is 10^3 , is a one cup of rice, then megabyte which is 10^6 becomes 8 bags of rice and we can see the gigabyte 10^9 , which is nothing but, we can extend it and we can understand that 3 semi-trucks of rice is 1 gigabyte. So, 2 container ships of rice will become 1 terabyte that is 10^{12} , and it represents the, the amount of information which flows on the internet ,petabytes which is 10^{15} is the blankets half of our city Jaipur, except ID which is 10^{18} that size is called a big data and here it comprises of or we can visualize the size as $\frac{1}{4}$ of this blanket, which are there in the country, and zettabyte which is 10^{21} and this basically will fill the Pacific Ocean and that amount of data, which is called a zettabyte, is a future volume of the big data, similarly keep on extending zettabyte becomes, zettabyte that is 10^{24} which becomes ,an earth-sized a rice bowl. And beyond that it's a brown to bide that is 10^{27} that becomes an astronomical size of that particular data. So, we are going and moving towards this kind of huge volume of data, which is of astronomical size, how to handle this kind of data is called a big data computation. And we are going to see these particular entry cases in this part of the course.

Refer slide time: (8:35)

What's making so much data?

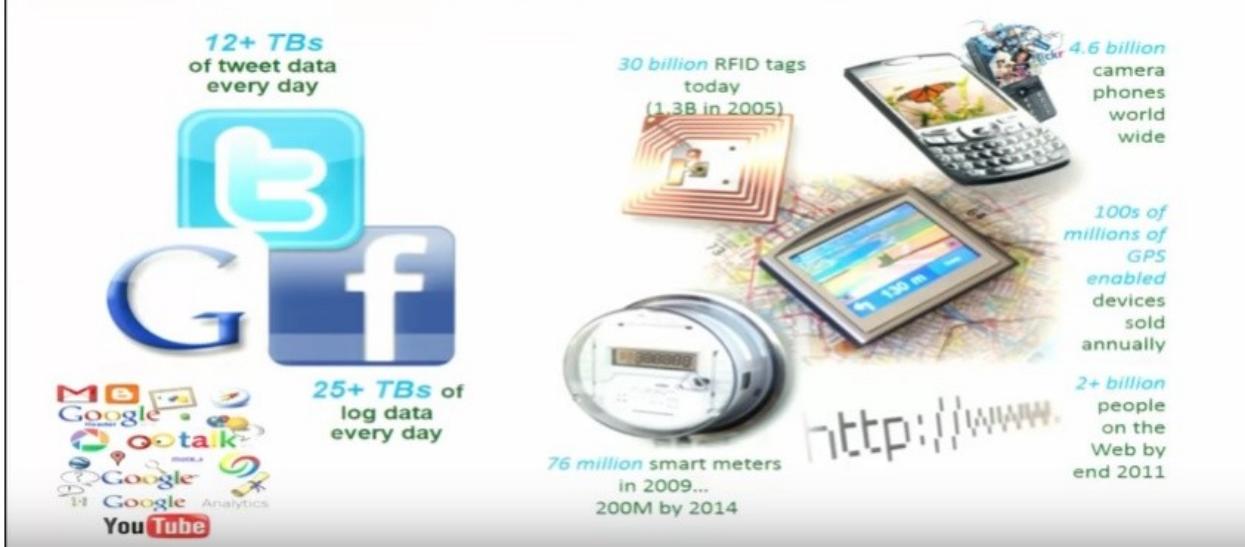
- Sources: People, machine, organization: Ubiquitous computing
- More people carrying data-generating devices (Mobile phones with facebook, GPS, Cameras, etc.)
- **Data on the Internet:**
- Internet live stats



Now, what's make so much of data? Now, here we consider there are three different sources, which make or which contributes to this so much of data? the first one is called people, for example you might have seen the Facebook or a people carrying the mobile phone, all the time they are generating the data either in the form of a text, in a Facebook or a GPS ,when mobile is being carried or basically cameras taking pictures so, these kind of data which is being generated, by the people, another type of data source, which generates a huge volume of data is using sensors. So, sensors are normally deployed in smart city organizations or in the industries or in many places they keep on generating the time series data. And the third type of data, third source is called organizations. So, organizations normally do the transactions, of other services, which transactions also and the customers transactions all these will become the source of data out of this organization and this is all together will form a ubiquitous computing. So, the data on the internet basically sometimes requires, to basically do the analysis over the live statistics, that we are going to see here, in this part of the lecture.

Refer slide time: (10:07)

Source of Data Generation



So, as I told you there are three different sources, one is the user. So, users which are basically using, the services or Facebook, Twitter, Google, you see that they are generating lot of data, and that basically is one of the sources of big data. The second kind of data source, which you see over here, is that the devices are, basically high devices with a sensor they are generating lot of data. For example, smart meters are generating data and RFID tags, in the objects, they are generating data and this camera which is there and sensors, which are there inside mobile phone, they are generating data. And also all the devices nowadays, equal with the devices which are called IoT devices and the sensors, they are continuously generating the data. And also two plus billion people on the web and this particular size of the web also are contributing enormous amount of data. So, there are all kind of sources which are now, generating the data and this becomes a big size of data.

Refer slide time: (11:25)

An Example of Big Data at Work

Crowdsourcing



Another example of a big data at work is, our using crowdsourcing. So, using Crowdsourcing this particular data, will be taken up or captured and perform the computation on it to basically gain the insight of the traffic congestion on the roads are doing the, the sensing ,where let us say an ambulance is moving and it requires a green path, and also it will perform or it will give a route, on the map and the routes are computed using this particular situation, which is dynamically changing at a particular time.

Refer slide time: (12:11)

Where is the problem?

- Traditional RDBMS queries isn't sufficient to get useful information out of the huge volume of data
- To search it with traditional tools to find out if a particular topic was trending would take so long that the result would be meaningless by the time it was computed.
- Big Data come up with a solution to store this data in novel ways in order to make it more accessible, and also to come up with methods of performing analysis on it.

So, where is the problem in this particular entire landscape of a big data. Now ,we see that in traditional RDBMS these queries is not enough sufficient to gain useful information out of the huge volume of data

that means traditional RDBMS queries are insufficient to handle this kind of big data and to gain the insight. To search it with traditional tools to find out if a particular topic was trending or take so long that the result would be meaningless by the time that means it requires a real-time computation or retrieval of that particular data, which is required at a particular point of time. And the traditional RDBMS operations that are that retrieval are slow and it is not useful for many of the applications. So, we will see in this particular part of the course, the remedy or the solutions which big data highs, regarding storing this information and providing the retrieval at a much faster speed and also newer methods which can perform the analysis on this kind of big data, at lightning speed.

Refer slide time: (13:34)

Challenges

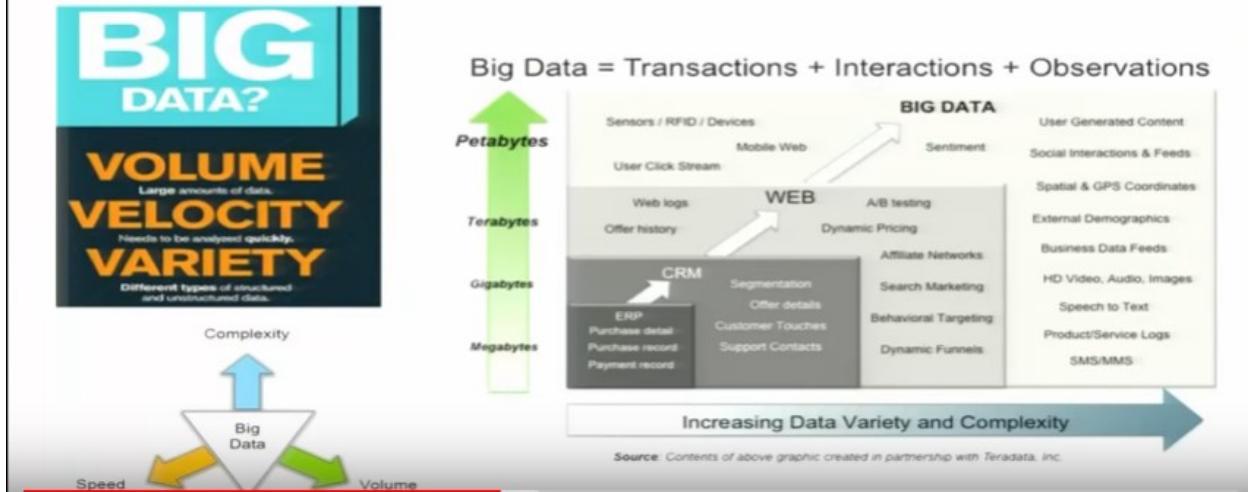
- Capturing
- Storing
- Searching
- Sharing
- Analysing
- Visualization

So, the challenges are summarized what Here, which are basically again around using the big data for different application is about capturing, storing, searching, sharing, analyzing, and visualizing.

Refer slide time: (13:49)

IBM considers Big Data (3V's):

- The 3V's: Volume, Velocity and Variety.



So, IBM and Gartner together, they consider the big data has three different 3Vs so the characteristic of a big data is given as these different 3Vs. So here, gotten or explained or IBM also considered three, most important 3V's so, first three V's which characterize the big data are volume, velocity, variety. What are these? And how it is going to signify this particular data as the big data? So, the characteristics of these particular features, that is the volume, velocity and variety will characterize the data as the big data .and we will see here, that when we say that it is the volume that is the size, that is beyond petabytes ,which will, which we have already seen, if that size is there then basically it enters into the big data domain and, and similarly if let us say the variety means, the data is not only the text data but it is in the form of the images, videos, 3d objects and so on, then basically there is a huge not only the size but also the data variety it is another dimension of the complexity. So, and another dimension which is called basically the velocity, that is the rate at which these data which is being generated, has to be tapped and a processed so, this becomes the speed or the velocity .

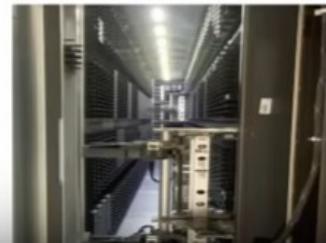
So ,the three means together, they basically will characterize the data as the big data. So, big data will be in the form of transactions, in the form of interactions or in the form of observations or together generating the large size of data that is data sets which need to be analyzed. So, here we see that let us say that Big Data scenario using this big data they are doing sentiment analysis to understand more insight about the entire centric, customer centric businesses that is the sentiments, when it comes there are sentiments two types of sentiment ,individual sentiments, that means if the business is targeted to basically for a particular individual or basically the entire customer base it is not individual but it is the entire population and the businesses are trying to understand the sentiments and plan the new businesses, which are basically possible. So, and similarly sensors RFD, RFID and different devices, they also generate lot of different data, similarly user click streams also generate lot of data, these particular data will perform the mobile web and will be analyzed in the real time, and to gain various insight for example, to understand the traffic condition situation in, in a smart city or, or to basically deal with the disasters. For example, if there is a fire at one place or is being triggered, triggering the fire so, it has to be controlled that is called disasters. So, all these that means the big data the mood inside has to be gained

and it is better serving the community or basically the masses. So, that is why the big data is here, and it is becoming popular day by day.

Refer slide time: (17:34)

Volume (Scale)

- **Volume:** Enterprises are awash with ever-growing data of all types, easily amassing terabytes even Petabytes of information.
 - Turn 12 terabytes of Tweets created each day into improved product sentiment analysis
 - Convert 350 billion annual meter readings to better predict power consumption



Now, let us see in more detail of these characteristics. So, the first characteristic of a big data is called volume and this is nothing but called scale. So, the enterprises are basically our growing and generating the data of all types and the size typically goes beyond terabytes then we'll be categorized as a huge volume and that basically is require different technology, which is called big data and for the computations. for example, if there are 12 terabytes of tweets, which are created every day, and which need to be analyzed, to for the sentiment analysis .so, this sometimes is a big data problem, similarly 350 billion annual meter readings in a smart meter, to for the analysis, to predict the power consumption again becomes a big data problem, where the volume is involved to be undersold using big data problem.

Refer slide time: (18:45)

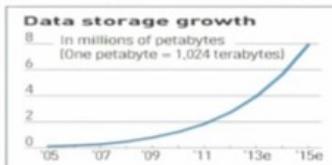
Volume (Scale)

- **Data Volume**
 - 44x increase from 2009-2020
 - From 0.8 zettabytes to 35zb
- Data volume is increasing exponentially

terabytes petabytes exabytes zettabytes

the amount of data stored by the average company today

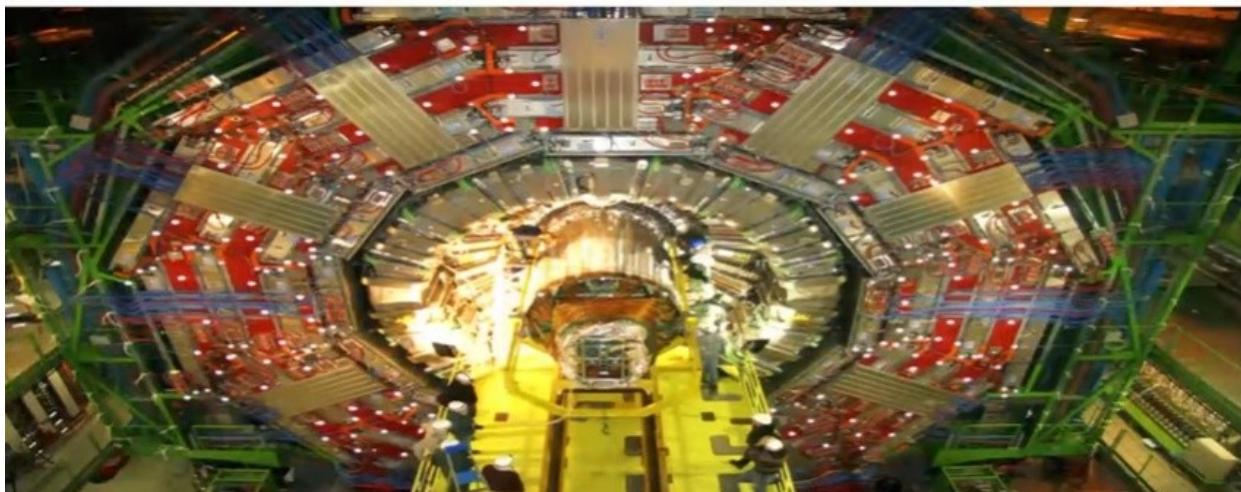
The Digital Universe 2009-2020



And Applications so, here we see that 44 X increase, in the volume of that particular big data is increasing from 2009-2020 and also that is from 0.8 zettabytes to 35 zettabytes. So, data volume is increasing exponentially and it requires a Big Data Platform to be used in this particular considerations.

Refer slide time: (19:16)

Example 1: CERN's Large Hydron Collider(LHC)



CERN's Large Hydron Collider (LHC) generates 15 PB a year

Another example, of generating the huge volume of data is the CERN's Large Hydron Collider, which generates 15 petabytes of data a year.

Refer slide time: (19:27)

Example 2: The Earthscope

- **The Earthscope** is the world's largest science project. Designed to track North America's geological evolution, this observatory records data over 3.8 million square miles, amassing 67 terabytes of data. It analyzes seismic slips in the San Andreas fault, sure, but also the plume of magma underneath Yellowstone and much, much more.

(http://www.msnbc.msn.com/id/44363598/ns/technology_and_science-future_of_technology/#.TmetOdQ--uI)



Another, source of generating big data is the earthscope, and here 67 terabytes of data is being generated and is being analyzed

Refer slide time: (19:43)

Velocity (Speed)

- **Velocity:** Sometimes 2 minutes is too late. For time-sensitive processes such as catching fraud, big data must be used as it streams into your enterprise in order to maximize its value.
 - Scrutinize 5 million trade events created each day to identify potential fraud
 - Analyze 500 million daily call detail records in real-time to predict customer churn faster



Now, the next characteristics in this sequence is called velocity, that is called a speed, sometimes 2 minutes is too late. So, basically that shows, that reflects, that here the time is a factor and everything has to be done within a particular time bound and this particular aspect of computation is called the velocity that means the data which is generated in real time it has to be analyzed and understood about various purposes. For example, if you want to catch a fraud, for an online transaction, then the entire transaction has to be analyzed and being detected whether it is a fraudulent transaction or it is a normal transaction, at that speed. So, big data must be using, the stream data in this particular scenario. so ,velocity is the stream of that particular data which is basically flowing, out of that the applications which need to be analyzed and being used in applications, similarly we have to scrutinize five million trade events which are created every day to identify the potential fraud ,similarly to analyze 500 million daily call details in a real-time to protect the customer churn at a faster or not. So, these kind of applications are now, driving the different companies or the organizations and to retain the customer and also to run the businesses in future. so ,obviously this aspect of a big data is also very much needed and volume, and velocity together, is creating the challenges in this big data computation.

Refer slide time: (21:39)

Examples: Velocity (Speed)

- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions → missing opportunities



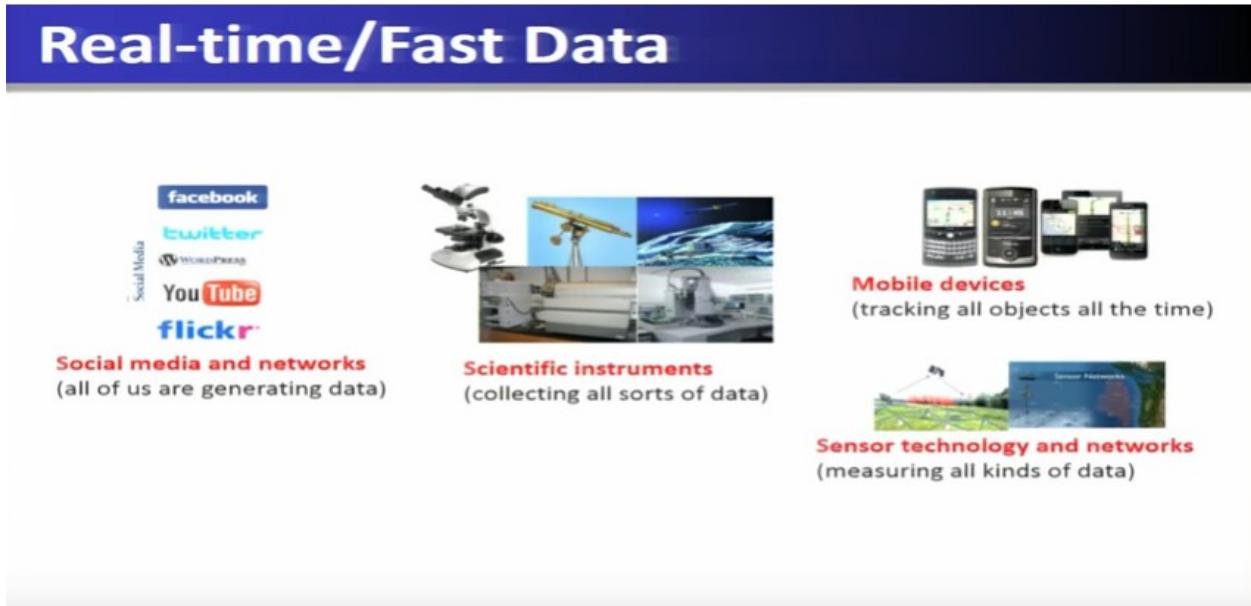
• Examples

- **E-Promotions:** Based on your current location, your purchase history, what you like → send promotions right now for store next to you
- **Healthcare monitoring:** sensors monitoring your activities and body → any abnormal measurements require immediate reaction

So, this we have already understood that Data, indicate in the velocity, means that I generated at a very fast pace and which need to be processed at that speed. Now, another use of this velocity is about online data analytics and here, if you miss or if you do this analysis, rate that means you will be missing the opportunity because, these operations are doing in real time and real time in some deadlines are there after that ,that decision is of no use. So, late decisions will employ the missing opportunities and this means that the velocity is in effect, at for that application. So, the examples of such cases are, like E-promotions and healthcare monitoring, where the sensors are monitoring, your activities and the body and being alerting you for any abnormal measurements, which requires an immediate attention or a reaction.

Refer slide time: (22:48)

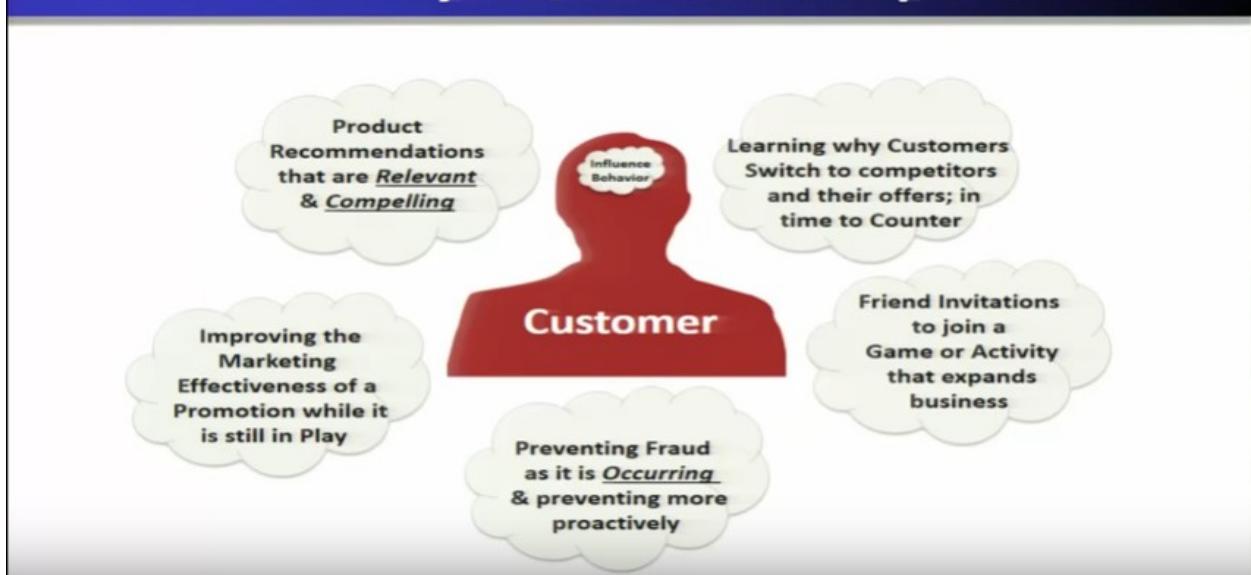
Real-time/Fast Data



So, this will give the real-time or basically the first data and nowadays, the social media and the networks are contributing in this particular dimension and has to be not only captured but, need to be computed in real time all this data similarly for the scientific instruments mobile devices, sensor technology and networks .

Refer slide time: (23:20)

Real-Time Analytics/Decision Requirement



So, this will also be very much requiring this kind of dimension that is the real-time, analysis or the decision-making. So, in most of the businesses, where customer centric decisions are to be taken, that means to give the product recommendations and to learn why the customers are basically making or turn out of that business or how the friend invitations are being sent to join together, which will basically be in the form of gaining, more businesses similarly, how to prevent the fraud? And, how to improve the marketing? It is all customers centric to understand the behavior or sentiment of a customer and do the real-time analytics, and this will be a very good way of running the business and every business has to basically, be a customer centric to drive it further. So, real-time analytics is very much required and the decisions are being used.

Refer slide time: (24:33)

Variety (Complexity)

- **Variety:** Big data is any type of data –
 - Structured Data (example: tabular data)
 - Unstructured –text, sensor data, audio, video
 - Semi Structured : web data, log files



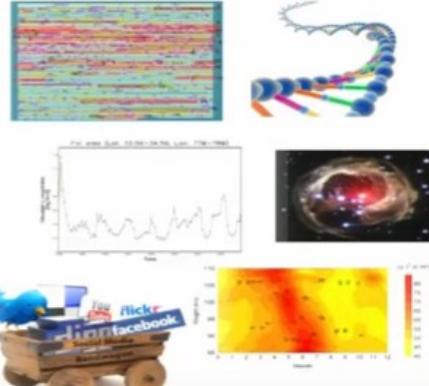
Next dimension is called a variety and which X to another level of a comp, which is called complexity. So, variety means the data, big data is not of one form but, of several type of forms of big data, comprises all for example, the structured data when it calls, when it basically is perceived is, the data which is stored in a form of a tables. Unstructured data which cannot be stored completely in the table or form, which is called unstructured data, which is basically the text sensor data audio and video, there are different type of data, which cannot be termed as the structured data that is lot of variety is there in the data becomes unstructured. Semi-structured is for example XML. So, web data, which is captured in the form of XML, forms a semi structured data, all these different variety, structured, unstructured and semi-structured data, will basically deals with a complexity to the big data, which is called the variety.

Refer slide time: (22:40)

Examples: Variety (Complexity)

- Relational Data (Tables/Transaction/Legacy Data)
- Text Data (Web)
- Semi-structured Data (XML)
- Graph Data
 - Social Network, Semantic Web (RDF), ...
- Streaming Data
 - You can only scan the data once
- A single application can be generating/collecting many types of data
- Big Public Data (online, weather, finance, etc)

To extract knowledge → all these types of data need to linked together



Examples of this variety dimension into the big data is ,the data which is coming out, of the real time data, out of the transactions tables and legacy data, then the text data which is on the web and semi structured data that is the XML data ,which is being captured out of the web, graph data which is nothing but, a social network data Semantic Web, streaming data you can scan the data once and the public big, public data which is available as online or a weather data or a finance data and so on together, these different variety of data will add to different complexity in Big Data computation, but very much needed for decision making into an organization. So, extract the knowledge, out of these varieties of data means that, all these type of data need to be linked together or correlated together and gain the meaningful insight, out of these correlated events or activities.

Refer slide time: (26:53)

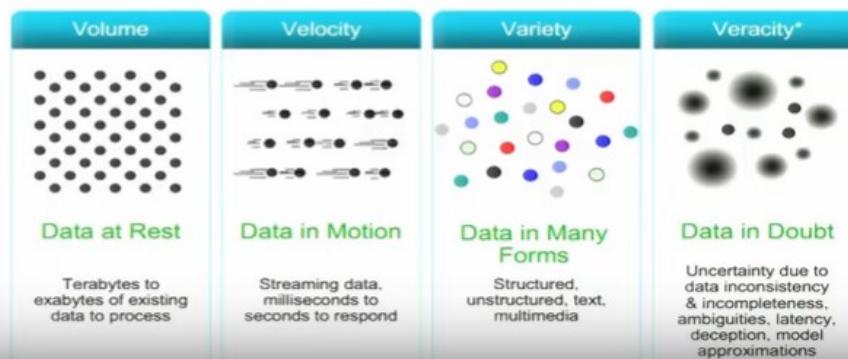
The 3 Big V's (+1)

- **Big 3V's**

- Volume
- Velocity
- Variety

- **Plus 1**

- Value



So, therefore we can summarize here ,the volume that is the data, at rest that means the terabytes or to the exabytes of the existing data need to be processed and this becomes the one V that is three V's of a Big Data, one of the three V's of the Big Data. the second one is called the volume or the velocity, velocity means the data which is there in the motion and this particular data is called a streaming data and this basically varies from milliseconds to the to the second to respond and this rate if it is the constraint and it becomes the velocity that data is called first data. Third type of data which is called, third type of characteristics which is called? the variety ,that means data is in many forms, that is structure, unstructured and semi-structured, that is the data is in the form of text multimedia and so on this becomes the variety of data. The fourth one, that means out of three, one more if we take this is called velocity. So, Velocity means, the data which is in Doubt that means the data which contains the uncertainties and this X to inconsistency, in completeness, latency, deception and this has to be curated before what is going to be used in the data. So, this veracity is basically that kind of errors, noise and uncertainties which are present in the data need to be handled.

Refer slide time: (28:30)

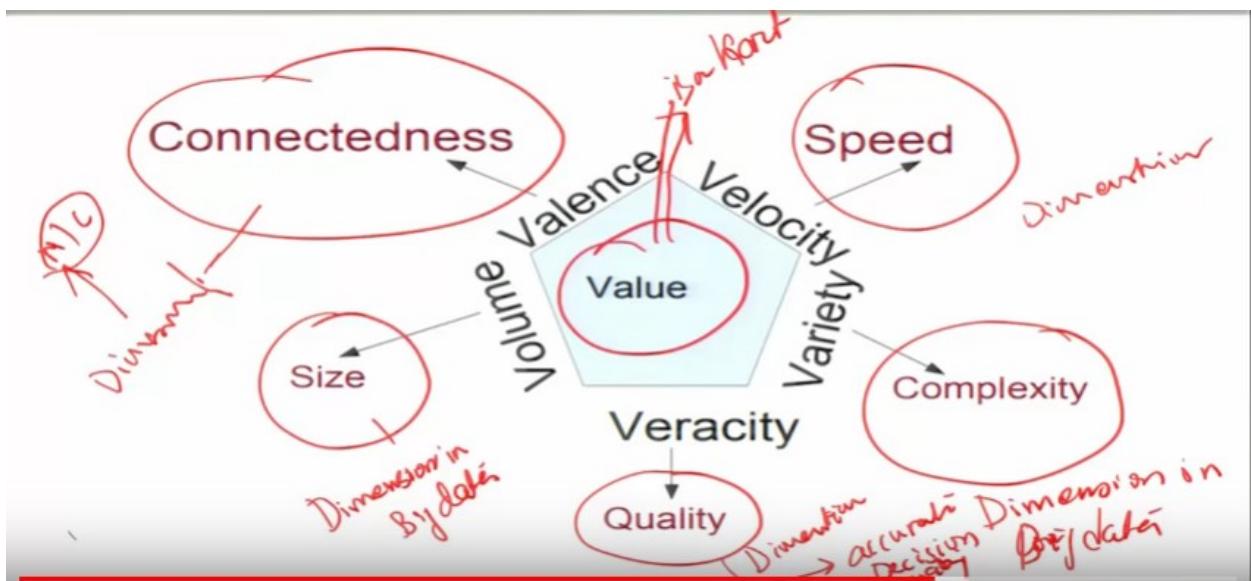
The 3 Big V's (+1) (+ N more)

- Plus many more

- Veracity
- Validity
- Variability
- Viscosity & Volatility
- Viability,
Venue,
Vocabulary, Vagueness,
- ...

And there are many more V's and such as validity and means so, so the time of that particular data, will indicate the validity, variability and viscosity, volatility ,viability, when you vocabulary, vagueness, all these as, all these will add more V. So, it's not three V's, in big data but plus n more V's are there .

Refer slide time: (29:04)



Now, let us summarize the most important V's and which we will be discussing in this part of the course, are as follows. So, that means the big data, first important characteristic of a big data is the volume which will add the complexity in the terms of dimension in the terms of size, the second one is called variety, which will add. So, the first one which is going to add the dimension in the big data is called the volume is going to add this dimension which is called a size in a big data. The second dimension is called the complexity; this is also a dimension in the big data, in the terms of variety. So, what I do will keep on adding the complexity and so, variety is another dimension. So, this complexity is coming out of due to the different variety of data. Third dimension is given in the terms of the speed if it is required and this is called the velocity. So, velocity that is in the terms of speed, will add another dimension and this particular dimension will add more complexity, in a big data computation. Finally the valence, valence means it is the autumn, which is taken out from the chemistry means, the more connected the data is higher violence it is. So, connectivity will go to add one more dimension to the big data. So, why this connectedness is important? Because, if you design the machine learning algorithm and if the data is less connected, than machine learning algorithm will work fine but, if the data is more connected ,then those machine learning algorithm has to be taken into a new way or revisit and a new machine learning algorithm is sometimes required.

So, it basically depends upon these different characteristics of a Big Data, and how the analysis is to be done that is the techniques, need to be means revised again, that is why these complexities are so important in the processing of this big data. finally another porosity, as I told you that lot of noise is there so, with a lot of noise and incompleteness, inconsistencies, remains into the data, and this particular data if it is analyzed obviously the quality of the decisions will go down .so, this is the dimension, which needs that the data has to be cured, I had a quality data is required, so that the decisions also will be more accurate for, accurate decision making. So, this these are basically characteristics will add different dimension of complexity in computation or Anneli or analyzing the data so, hence the big data analytics has to deal with these complexities and we are going to see all these aspects in this part of the course and finally these at the heart of all these dimensions you see that, this is at the heart, meaning to say that finally using this particular different characteristics and their dimensions finally you have to gain some value for extract some value out of that particular big data and which is going to be useful for an application. So, this value it has to give a value otherwise why? This data big data is becoming so, important that we are going to see. So, value is going to be made finally and this will be used in various applications.

Refer slide time: (33:30)

Value

- Integrating Data
 - Reducing data complexity
 - Increase data availability
 - Unify your data systems
 - All 3 above will lead to increased data collaboration
-> add value to your big data



So, value is derived out of integrating these different dimension or characteristics of that particular data. For example, sometimes you can reduce the data complexity, increase the data availability, unify your data streams and all these above will lead to the increased data collaboration and also will add the value to your big data. So, value adding the value, are extracting the value out of the big data for a different application is going to be at the heart or at the center.

Refer slide time: (34:01)

Veracity

- **Veracity** refers to the biases ,noise and abnormality in data, trustworthiness of data.
- 1 in 3 business leaders don't trust the information they use to make decisions.
 - How can you act upon information if you don't trust it?
 - Establishing trust in big data presents a huge challenge as the variety and number of sources grows.



Now, we will we have briefly discussed let us see more detail about the characteristic, which is called the veracity. So, veracity refers to the biases or the noise or the abnormalities, which resides into the data and basically sometimes the doubt on the trustworthiness of the data. So, for example, one in three business leaders do not trust the information that they used to make the decision and for example, if you let us say

age is asked by a particular person and if that particular person is giving a wrong age, and so, basically this goes in the terms of noise or sometimes people don't specify their age and sometimes if the age is going to be important in making decisions. in a particular business and then basically this particular aspect is going to be touched upon as veracity. So, how can you act upon the information if you don't trust on it? For example if somebody gives wrong age information and if you are acting on it then the decisions are not going to be accurate. So, veracity is going to be an important factor and it will affect the decisions and so therefore, the quality of data veracity will ensure that way, so, establishing the trust in a big data presents a huge challenge as the variety and the number of these sources grows.

Refer slide time: (35:38)

Valence

- **Valence** refers to the connectedness of big data.
- Such as in the form of graph networks



Another, characteristic is called valence or often refers to the connectedness of the big data. such as, in we see that the, the graphs of forms of a graph of the network, that means of the graph is dense sparse. So, there are different analysis in, in the algorithm which are to be applied in these different dynamic situations, varies and the valence, is going to be useful in that aspect.

Refer slide time: (36:13)

Validity

Accuracy and correctness of the data relative to a particular use

- Example: **Gauging storm intensity**

satellite imagery vs social media posts



- prediction quality vs human impact

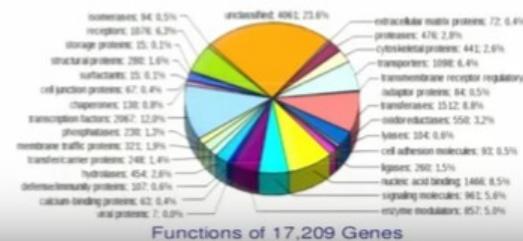
The next we is called validity, that is the accuracy and correctness of the data, relative to a particular use. So, it depends upon a particular use case this validity that is accuracy and correctness of the data is going to be useful. For example, in a satellite imaginary for predicting the quality versus social media post, where the human impact is going to be important part.

Refer slide time: (36:43)

Variability

How the meaning of the data changes over time

- Language evolution
- Data availability
- Sampling processes
- Changes in characteristics of the data source



We will see, more such use cases or examples of these characteristics of a big data, another characteristic is called variability that is, how the meaning of a data changes over time?

Refer slide time: (36:58)

Viscosity & Volatility

- Both related to velocity
- **Viscosity:** *data velocity relative to timescale of event being studied*
- **Volatility:** *rate of data loss and stable lifetime of data*
 - Scientific data often has practically unlimited lifespan, but social / business data may evaporate in finite time

And, furthermore the characteristics are viscosity and volatility both, related to the velocity, viscosity is the data velocity relative to the time scale of event being studied and volatility means the rate of data loss and stable lifetime of the data.

Refer slide time: (37:16)

More V's

- **Viability**
 - Which data has meaningful relations to questions of interest?
- **Venue**
 - Where does the data live and how do you get it?
- **Vocabulary**
 - Metadata describing structure, content, & provenance
 - Schemas, semantics, ontologies, taxonomies, vocabularies
- **Vagueness**
 - Confusion about what “Big Data” means

There is more V's and for example, vocabulary means, metadata describing the structure and vagueness is the confusion about what big data means at that particular application, for that application.

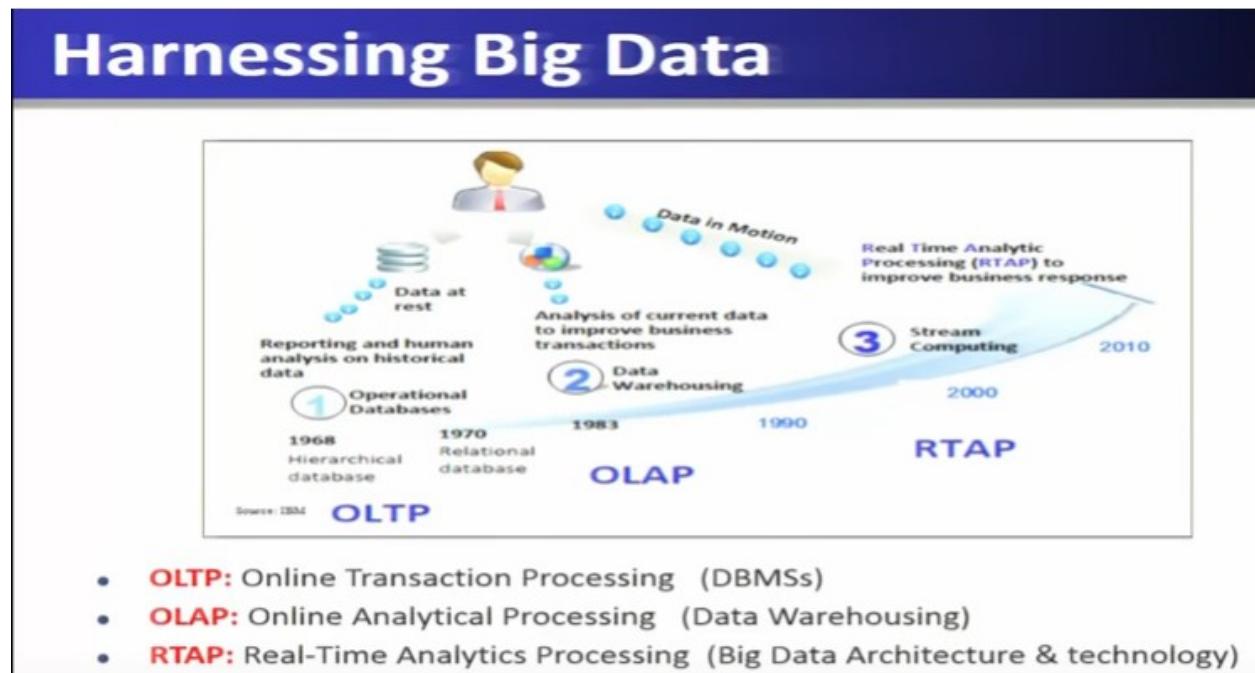
Refer slide time: (37:32)

Dealing with Volume

- Distill big data down to small information
- Parallel and automated analysis
- Automation requires standardization
- Standardize by reducing Variety:
 - Format
 - Standards
 - Structure

So, now coming, how are we going to address? these characteristics and the complexities around ,these different characteristics, which are there in the big data. So ,if the volume is big then, we require to develop the method ,which can be computed in parallel the data and also perform this particular big data to gain the summary of that information and how this particular data is to be handled? That is, what he will be the format, standard, structure? And this all will be taught in the terms of dealing with this volume.

Refer slide time: (38:24)



Similarly if we see, about the harnessing of a big data, one way means earlier, the traditional approach was using the operational databases every company was having the databases where the name of a customer and all the details were stored. so ,that is how the relational database becomes very powerful in and the means the and has developed lot of classical techniques to handle and that is called OLTP, the next stage is again has been passed out, which is called OLAP. and this deals with the data warehouses that comes out of different databases it pulls the relevant information and forms the data warehouse for making the decisions finally nowadays, it is our tap and here the data which is in the form of a stream, that is data is in motion and it has to be called stream computation has to be applied on it to extract the meaningful insight and this RTAP is called, real-time analytic processing to improve the business, response, and this is the latest trend and in the big data, we will see about the stream computation. So, OLTP means online transaction processing, which is related to the date DBMS and OLAP, stands for online analytical processing, which deals with the data warehousing and RTAP which is called real-time analytics, processing which handles the big data architecture and the technology.

Refer slide time: (40:11)

The Model Has Changed...

- **The Model of Generating/Consuming Data has Changed**

Old Model: Few companies are generating data, all others are consuming data



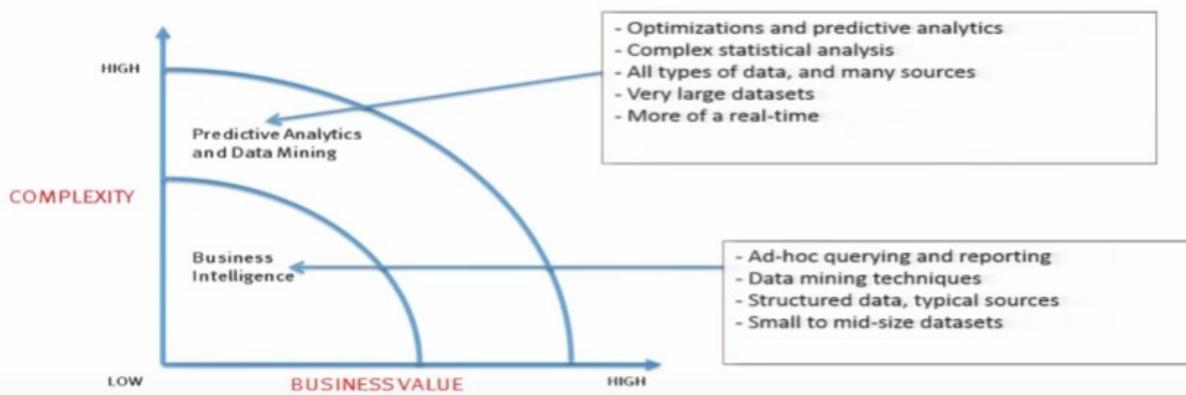
New Model: all of us are generating data, and all of us are consuming data



So, we see that the model of the competition is quite changing, that means the earlier model, if we see was based on the a DBMS, OLTP and OLAP, this was the old model and new model is based on the real-time data, that means all of us are generating the data and all of us are consuming the data is not only the companies which are generating the data and they are consuming it.

Refer slide time: (40:42)

What's driving Big Data

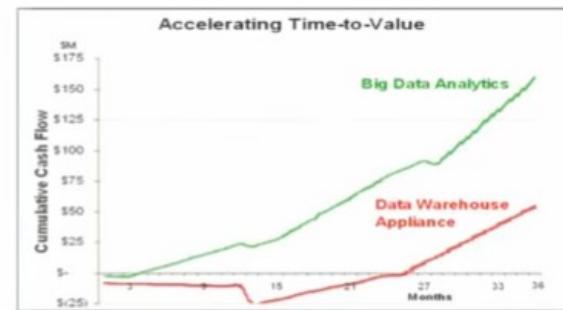


So, the new model, is required and to be integrated into the different business, decision-making and the solutions and therefore, if we see this particular picture which will, which is? What is the driving the big data? further development and its research and its use case is shown here in this particular picture that means earlier it was a business intelligence, we are the value of smotret and the complexity also was moderate but, nowadays it comes predictive analytics and data mining here, the optimizations and predictive analytics are not easy and requires a computation of the big data. And, and also has to be done in the real time. So, all the complexities are now there and the analytics becomes now, called predictive analytics, and earlier analytics, in the business analytics, business analytic, intelligence uses the, the prescriptive and descriptive analytics, but nowadays, predictive analytics is very much of use which needs a real-time or a stream computation processing of the large data sets.

Refer slide time: (41:57)

Big Data Analytics

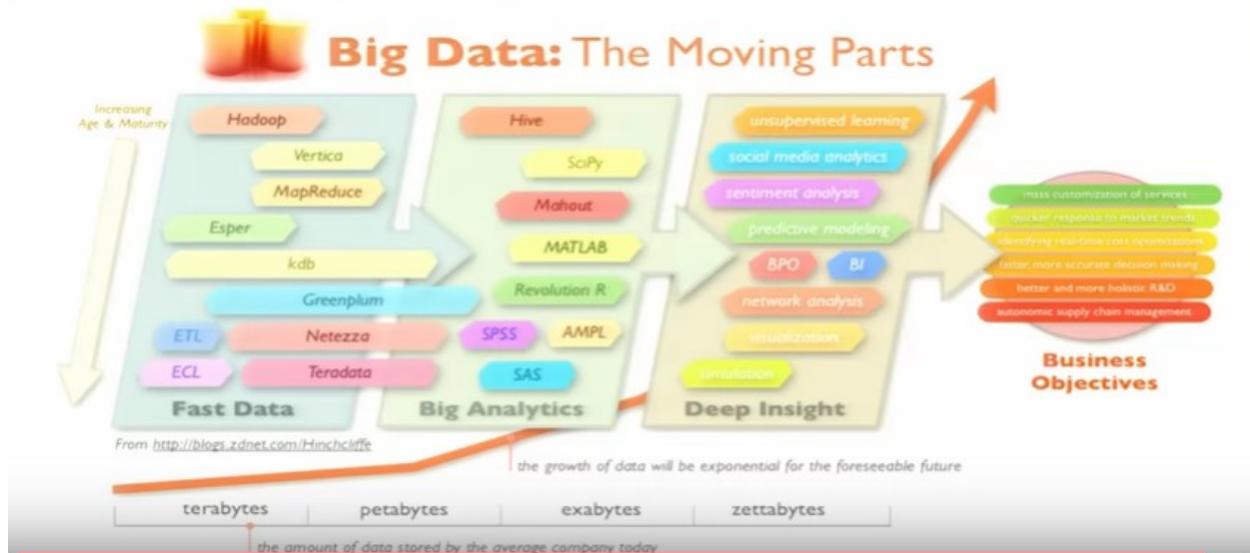
- Big data is more real-time in nature than traditional Dataware house (DW) applications
- Traditional DW architectures (e.g. Exadata, Teradata) are not well-suited for big data apps
- Shared nothing, massively parallel processing, scale out architectures are well-suited for big data apps



So, big data analytics is driving, these different businesses and requires the insights out of the big data computations, that we are going to cover up in this part the course.

Refer slide time: (42:17)

Big Data Technology



So, as far as if you see the big data, which is moving? So, big data is first we see that it is the first data and ETL and all these things, which we have already seen, and then comes the big analytics big data analytics, which are different tools? Which are available? Which is based on the Big Data technology? And nowadays to gain deeper insight, different machine learning and predictive analytics are applied on the big data that we are going to see. So, these kind of I mean now, now those techniques, for analysis, analytics, requires, computing in terms of terabytes, petabytes, exabytes and zettabyte that is the huge size of volume.

Refer slide time: (43:02)

Conclusion

- In this lecture, we have **defined Big Data** and discussed the challenges and applications of Big Data.
- We have also described **characteristics of Big Data** i.e. Volume, Velocity, Variety and more V's, Big Data Analytics, Big Data Landscape and Big Data Technology.

So, conclusion in this lecture, we have defined Big Data and discuss the challenges and various applications of big data, we have also described in more detail about, the characteristics of big, big data and the three most important characteristics of a Big Data, that three V's we have covered in quite detail

that is the volume, velocity and variety. Furthermore we have also seen other V's, which are evolving around that big data as the, the time progresses and matures this particular big data area furthermore. So, big data analytics we have also seen a little bit about that and also, about the big data landscape and various terminologies and technologies we have already just touched upon. Thank you.

Lecture 2

Big Data Enabling Technologies

Big Data Enabling Technologies

Refer Slide Time :(0:17)

Preface

Content of this Lecture:

- In this lecture, we will discuss a brief introduction to Big Data Enabling Technologies.



Preface Content of this lecture. In this lecture, we will discuss a brief introduction to big data enabling technologies. Here different icons are shown, which are different components, which we are going to discuss,

Refer Slide Time :(0:33)

Introduction

- Big Data is used for a collection of data sets so large and complex that it is difficult to process using traditional tools.
- A recent survey says that 80% of the data created in the world are unstructured.
- One challenge is how we can store and process this big amount of data. In this lecture, we will discuss the top technologies used to store and analyse Big Data.

today. Introduction; Big data is used, for a collection of data sets. So large and complex, that it is difficult to process using traditional tools. A recent survey, says that 80% of the data created, in the world are, unstructured. Hence traditional tools, will not be able to handle, such a big data motion. One challenge is, how can we stored and process this big data? In this lecture, we will discuss the technologies and the enabling framework to process the big data.

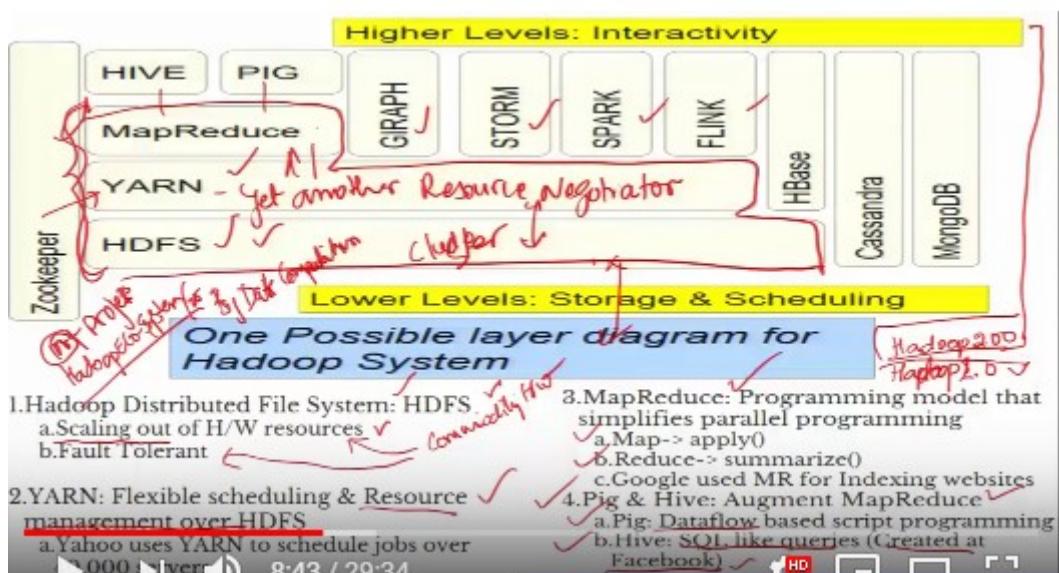
Refer Slide Time :(1:14)

Apache Hadoop

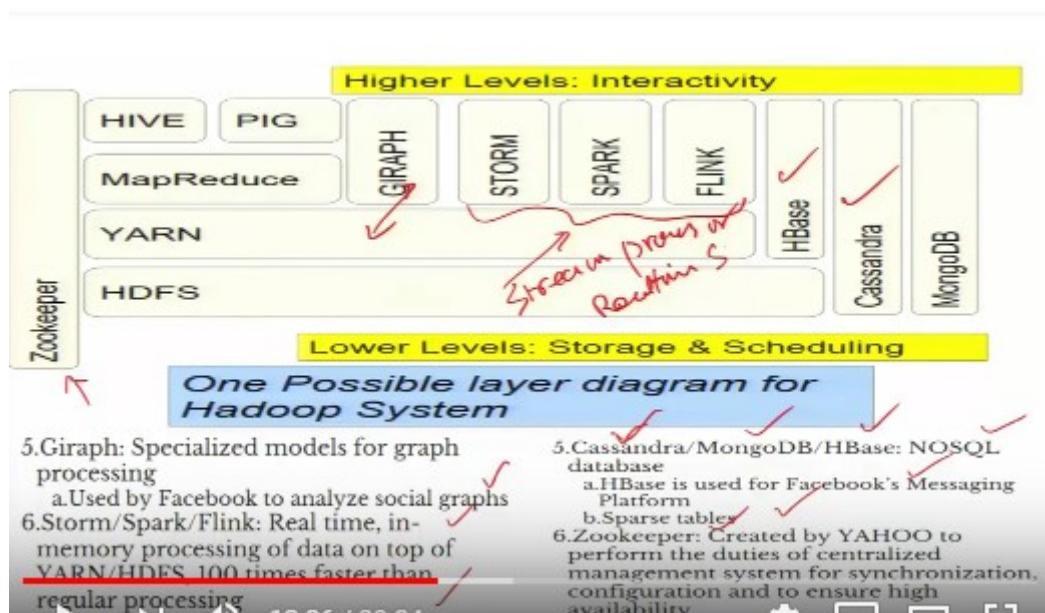
- Apache Hadoop is an open source software framework for big data.
It has two basic parts:
 - Hadoop Distributed File System (HDFS) is the storage system of Hadoop which splits big data and distribute across many nodes in a cluster.
 - a. Scaling out of H/W resources
 - b. Fault Tolerant
 - MapReduce: Programming model that simplifies parallel programming.
 - a. Map-> apply ()
 - b. Reduce-> summarize ()
 - c. Google used MapReduce for Indexing websites.

Apache Hadoop is the tool which is going to be used for, the big data computation. Apache Hadoop is an open source, software framework, for a big data. And, it has two parts, two basic parts. The first one is called, HDFS, Hadoop Distributed File System, the other is called, ‘Programming Model’, which is called a, ‘Map Reduce’.

Refer Slide Time :(1:39)



Refer Slide Time :(8:45)



Now, you will be looking up the Hadoop Ecosystem for big data computation. So, this is the Hadoop Ecosystem, for big data computation, which is summarised as follows. So, as I told you that Hadoop has two main components, that was, the old version of Hadoop, which is called a, 'Hadoop, Version 1, 1.0'. Now there was some disadvantage of Hadoop, Version 1.0. So, newer version, which is called a, 'Hadoop 2.0', came into effect. So this particular picture is, about Hadoop 2.0 version, which includes, besides HDFS, and Map Reduce another version of, YARN is being added. So, YARN is, a Resource Manager, are, for Hadoop. This is yet another Resource negotiator. This is also called, 'Resource Manager'. For Hadoop which runs over HDFS. Now HDFS, is are distributed file system which is run over the cluster. So, below this HDFS, will have, the cluster infrastructure and over that, Hadoop distributed file system runs, which manages, all the notes, and their corresponding memories, using HDFS. And the Resources which are, required by the application, is being allocated using the Resources Manager, which is called a YARN. And the programming framework for this, big data computation, is a Map Reduce. So, in Hadoop 2.0, there are three different mean components, Map Reduce, YARN and HDFS which runs over the cluster, now with this introduction of a YARN.

So, there are some applications, at as, Giraph, Storm, Spark, Flink, they are not going to use, Map Reduce directly, they run over YARN and HDFS. Similarly, the applications which will simplify the use of, Map Reduce further on, called, Hive and Pig they are on over, Map Reduce. Earlier in Map Reduce, in Hadoop version 1.0, all the applications were running over Map Reduce. Now there is a choice that means Map Reduce, or a non-Map Reduce applications, can run with the help of, YARN HDFS. So, now as if, HDFS, is it Hadoop 2.0, it is possible now, to have more flexible. So, there are various projects, which we are going to discuss, with the notion, with the introduction, of YARN in Hadoop 2.0, lot of new projects, are there. More than hundred projects are available in Hadoop eco system. And they're all open source that is free. These projects we will discuss some of them, which is useful, for big data computation. Let us start one by one, that is from HDFS. So HDFS stands for Hadoop Distributed File System. So, this particular file system runs over the cluster. So it basically, is based on scale out, of hardware Resources, that is the notes can add, can be added. Any number of notes can be added. And this particular style, is called, 'Scaling Out'. So it is not always necessary, that very powerful systems are required. So, the Resources which are required, here is the form of cluster, they are called, 'Commodity Hardware'. So these commodity hardwares are prone to failure, so this HDFS, provides lot of provisioning, for fault tolerance that we will discuss, when we go, in more detail, of HDFS. So, HDFS is a file system, over the cluster, and this is called a, 'Hadoop cluster'. Over HDFS, the Resources are being provided reducing Resource Manager, for the different application and this is called a, 'YARN', YARN runs over HDFS. So, the full form YARN is yet another Resource Manager. It is flexible scheduling and Resource management, over HDFS. Now on, over YARN, there is programming model for big data computations and this will simplify the, parallel programming notion, using to function, this is called a, 'Map and Reduce'. So, Google earlier uses,

Map Reduce for indexing and they are so powerful that, almost all big data applications, can be programmed using this, for align Map Reduce that we will see, in this part of the course, in more details. Now, over Map Reduce, there are two tools, which are available, one is called, ‘Pig and Hive’. So, Hive, is created at Facebook. There are all SQL like queries, which runs over, Map Reduce. So, complex programming of Map Reduce can be avoided, using Hive so, Hive will simplify, the programming, over Map Reduce, for big data computation. So, Hive provides, SQL live queries, so it simplifies the entire programming notion and it was developed or it was created at the Facebook. Now Facebook also uses another tool, which is called the, ‘Pig’. Which is, the scripting based or a dataflow based, script programming, runs over the Map Reduce. So, both of this Pig and Hive will augment, the Map Reduce. So, the complex programming of Map Reduce can be simplified by, using these tools, Pig and Hive.

Now, has you know that, within non-Map Reduce application, such as Giraph, Giraph is a graph processing tool, which is, being used by the Facebook, to analyse the social network’s graph that was made simplified, when it was made out of Map Reduce. So, it uses YARN and HDFS and this is non-Map Reduce application, for, computation or computing large graphs, of the social network. So, Giraph is the tool which is now, runs over, YARN HDFS, and this is used, the big graphs computations that we will see, later on, this part of the course. The next tool which is, there is called, ‘Storm, Spark and Flink’. They are going to deal with the real time, in memory processing, of the data or YARN and HDFS. So the fast data are Streaming data applications, either can we do using; a Storm, Spark and Flink and they basically, are in memory computation, which are faster than regular computation. So, Stream processing, or a real time or the real time, Streaming applications are done using, Star, Spark and Flink or YARN and HDFS. Now, then, we are going see, the storage part, the big data, how it can be stored. Now most of these big data is stored in the form of a key value pair and they are also, known as, No Sql Data Store. So, this No Sql Data Store can be supported by, the data base like, Cassandra, MongoDB and HBase. So, these particular HBase, is over HDFS, Cassandra, MongoDB and they’re also, using this kind of for, for, for storing the big data applications. So, HBase was, initially using, used by the Face book for its messaging platform, later on, Face book also used, developed Cassandra, for the key value store purposes. Now, as far as these diffident components, of Hadoop ecosystem and they are also called as, ‘Open Source Projects’, which is used for, the big data computation. Now, another part is, called the, ‘Zookeeper’. Why? Because the names are of, some animals, so the Zookeeper is coordination service, which is useful for all these different projects. So Zookeeper is created, at, by the Yahoo to form, the centralized, management, for synchronization, configuration, to ensure the high availability, that well be see. So this are diffident projects, of this, open source framework, for big data computation and this is called the Hadoop ecosystem. And, we will be, discussing and using in this part course, in upgrade detail of it.

Refer slide time :(12:07)

- **MapReduce** is a programming model and an associated implementation for **processing and generating large data sets**.
- Users specify a **map** function that processes a key/value pair to generate a set of intermediate key/value pairs, and a **reduce** function that merges all intermediate values associated with the same intermediate key



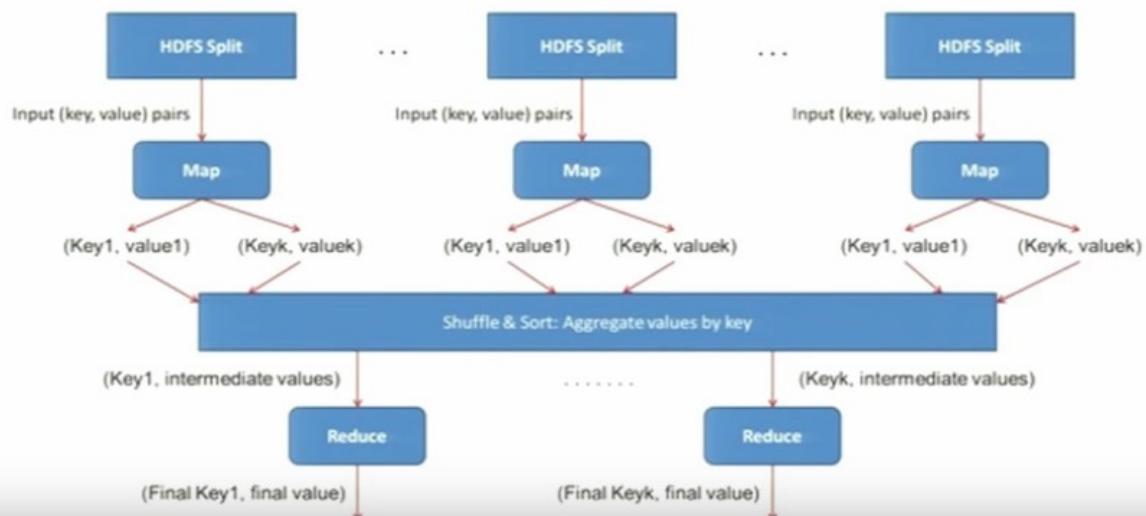
Big Data Computing

Big Data Enabling Technologies

So, now let us see about the map reduce, which is programming model for, big data component. So it is able to process, the large size data sets, using two functions, which are called, ‘Map and reduce’.

Refer slide time :(12:24)

Map Reduce

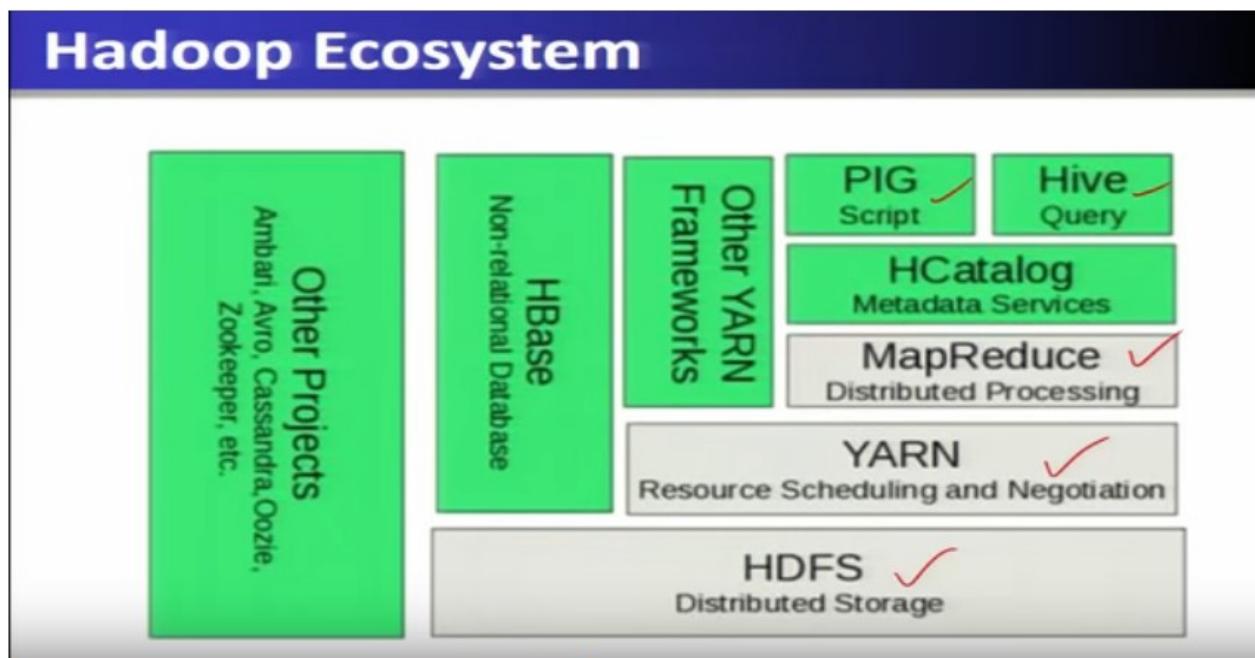


Big Data Computing

Big Data Enabling Technologies

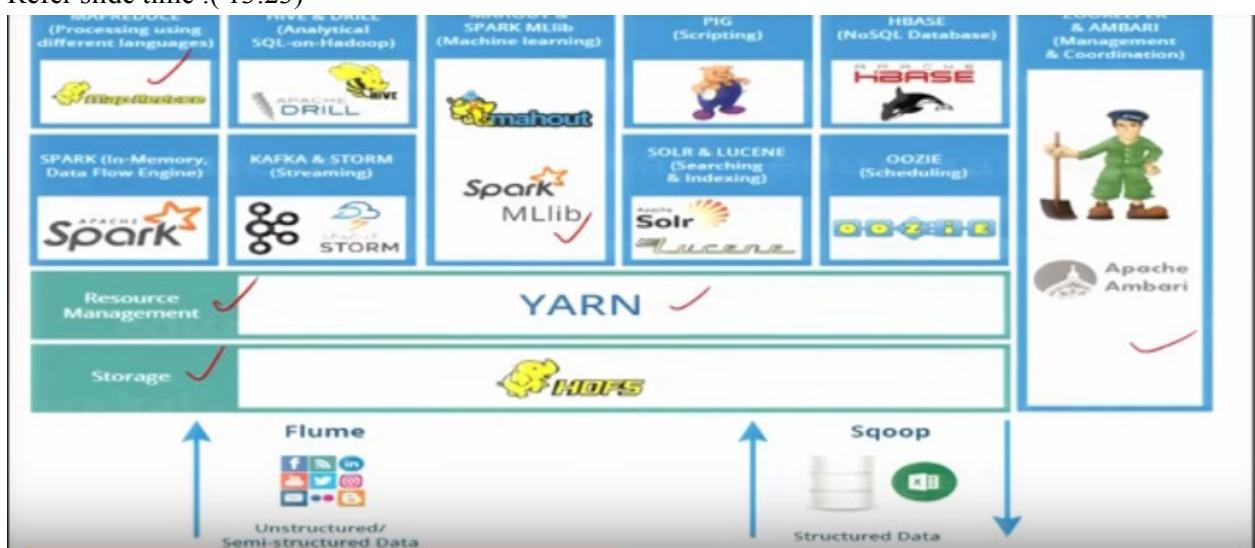
Which user can specify. And, this runs over, HDFS, in version 1.0 and in version 2.0, using YARN, it can negotiate for the, Resources, from HDFS, and run its application.

Refer slide time :(12:41)



So, this Hadoop ecosystem, we can again see, in this picture, in a great detail. So, HDFS will provide and distributed storage. YARN will provide Resource Manager for Hadoop and for distributed processing, that is a programming for big data, is done through the, the map reduce. And Pig and Hive, is basically, these tools which will simplify the programming of the map reduce. And the various others, projects which we have seen, which basically use, HDFS, yarn or map reduce.

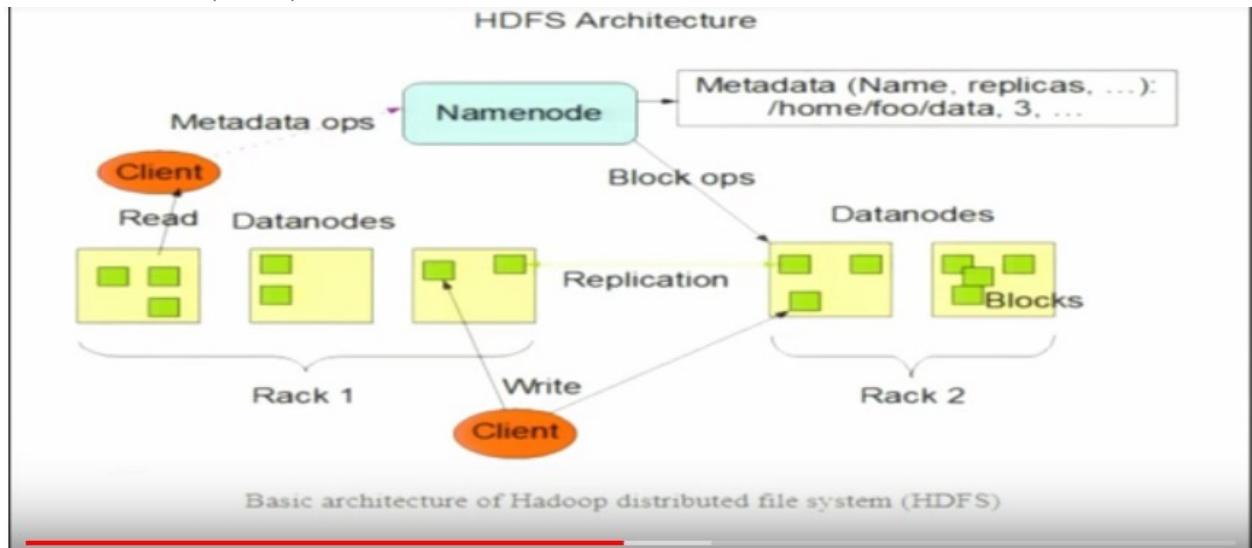
Refer slide time :(13:23)



So this particular, entire Hadoop ecosystem is shown in this, simplified picture, that, HDFS provides storage for big data. The Resources like, CPU, Memory and other resources, are being allocated, using on managed by the yarn, for different applications like, map reduce. Similarly, they are different, other projects, such as, in memory computation that's called, a 'Purchase Spark'. And machine learning over this Spark, called park MLlib and this particular project, which is called,

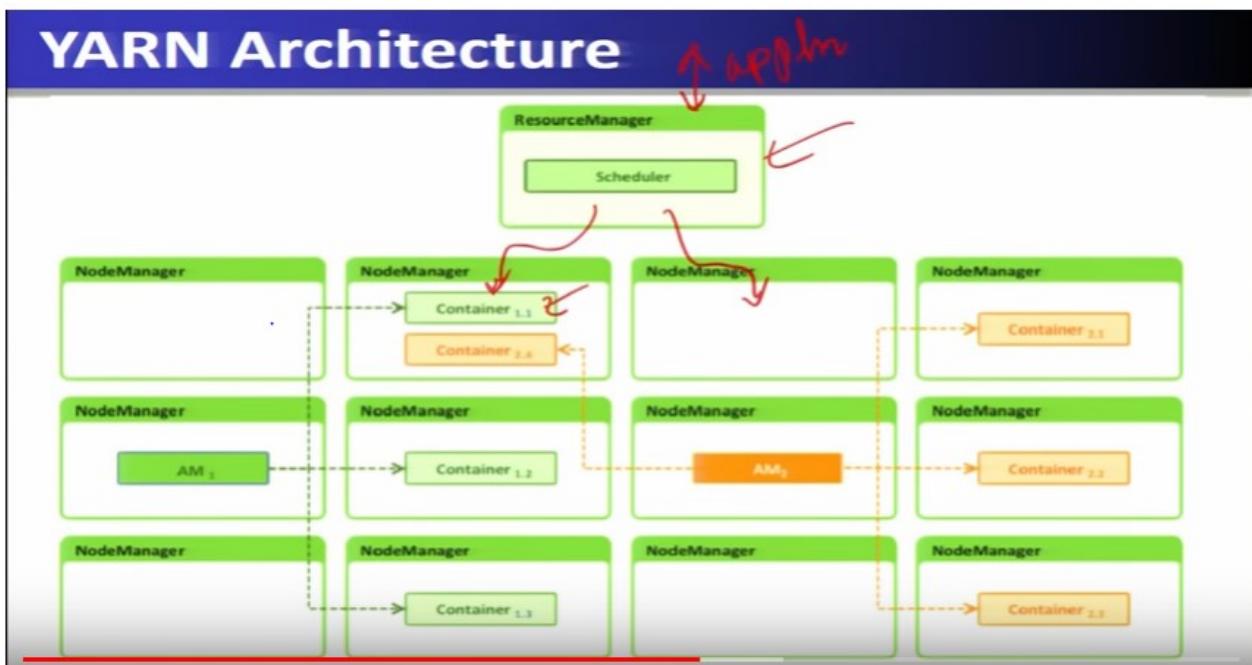
'Streaming applications', over yarn and HDFS, that is called, 'Kafka and Storm.' And the Zookeeper is a centralized service. So, this particular, entire different project will form the Hadoop Ecosystem.

Refer slide time :(12:32)



And more than hundred such projects, are available free, that's in open source projects, which is use for big data computation.

Refer slide time :(14:40)



That, we will see more details, so, this particular a yarn, Resource Manager, which is yet another Resource negotiator, which is a Resource Manager, for the Hadoop ecosystem, for Hadoop system, is responsible for allocating, the Resources, to the various applications, running in a Hadoop cluster and scheduling task, to be executed, on different cluster Nodes. And this open source Hadoop distributed, framework. It has two parts. One is called, 'Resource manager', and this is Resource Manager, it has scheduler within it. So, this particular, here the applications can interact with this Resource Manager

and can demand for different Resources, through YARN. This particular Resource Manager, in turn, knows or allocates the Resources, which are managed, at the Node level, by the Node Manager. And, the Resources are nothing but, the container, in the form of a container. So, whenever Resources is, means required, by the application, it has go through, the Resource Manager, Resource Manager knows and allocates, the Resources to the application, with the help of Node Manager and the Node Manager, Resources, allocates the Resources, in the form the container. So container is the basic resources, which are allocated, to different applications, which are managed internally, by every Node. So every Node is having a node Manager and these particular resources are allocated and reclaim, with help of, a centralized Resource Manager in YARN.

Refer slide time :(16:26)

Hive

- **Hive** is a distributed data management for Hadoop.
- It supports SQL-like query option HiveSQL (HSQL) to access big data.
- It can be primarily used for Data mining purpose.
- It runs on top of Hadoop.



And Hive, is a distributed data management, for Hadoop, and it supports, SQL like query, option that is called, ‘Hive SQL’, to access big data and it is primarily used, for data mining purpose.

Refer slide time :(16:26)

Apache Spark

- **Apache Spark** is a big data analytics framework that was originally developed at the University of California, Berkeley's AMPLab, in 2012. Since then, it has gained a lot of attraction both in academia and in industry.
- Apache Spark is a lightning-fast cluster computing technology, designed for fast computation.
- Apache Spark is a lightning-fast cluster computing technology, designed for fast computation



Now we will see the Apache Spark. Apache Spark project, runs over, HDFS and this is a big data analytics frame work, which in memory computation. So that the, lightning fast cluster computation, is being performed. So several applications like, Stream processing, Machine learning and Large Scale Graph Processing, are already implemented, over this Spark.

Refer slide time :(17:20)

ZooKeeper

- **ZooKeeper** is a highly reliable distributed coordination kernel, which can be used for distributed locking, configuration management, leadership election, work queues,....
- Zookeeper is a replicated service that holds the metadata of distributed applications.
- **Key attributed of such data**
 - Small size
 - Performance sensitive
 - Dynamic
 - Critical
- **In very simple words**, it is a central store of key-value using which distributed systems can coordinate. Since it needs to be able to handle the load, Zookeeper itself runs on many machines.



18:15 / 29:34

Big Data Computing

Big Data Enabling Technologies



Core. And we will see that these, different projects which are more than hundred plus, to manage them, there is a centralized service, which called, ‘The Zookeeper’. So Zookeeper is a highly reliable, distributed, coordination service, and which can be used for, locking, configuration management, either election, and so on. So Zookeeper is a replicated service and it has the attributes, key attributes, such as, Small size, Performance Sensitive, Critical. And it is a part of the Apache Zookeeper, Open source project. And in simple words, it a central store for, the, the key value pair, using distributed system, which can coordinate, since it needs to be, able to handle the load. So the Zookeeper runs itself, on many machines.

Refer slide time :(18:17)

NoSQL

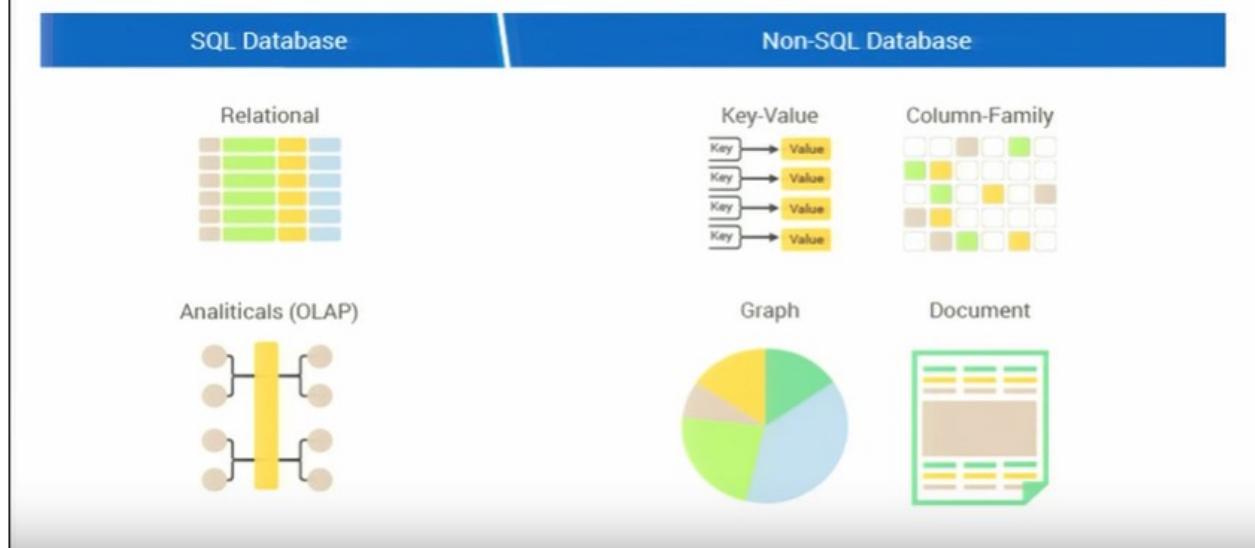
- While the traditional SQL can be effectively used to handle large amount of structured data, we need **NoSQL (Not Only SQL) to handle unstructured data.**
- NoSQL databases store unstructured data with no particular schema
- Each row can have its own set of column values. NoSQL gives better performance in storing massive amount of data.

**Not
Only
SQL**

So, that is the internal working of the Zookeeper, but the primarily Zookeeper, is used to, make, make the coordination and configuration service of all, the projects, which are running, which are executing the big data computation, using different projects. So, that is why, this, the name Zookeeper is being made. Now NoSQL, we see that, that traditional SQL, can be effectively used to handle the large amount of, structured data. But here in the big data, most of the information is, unstructured form of the data, so basically, NoSQL that is, is required to handle that information, because, traditional SQL required, by the, the data to be, in the structured, data format. So NoSQL data base is, stored unstructured data also, however, it is not, enforced to follow a particular, fixed schema structure and schema keeps on, changing, dynamically. So, each row can have its own set of column values. NoSQL gives a better performance, in storing the massive amount of data compared to the SQL, structure.

Refer slide time :(19:39)

NoSQL



So, here you can see that, NoSQL database is primarily a key value store. It is also called a, ‘Column Family’, and it is different and the relational database management system, which is in the form of tables and this is a Column family. So, column wise, the data is stored, in the form of a key value, pairs. So, we will see, more detail, about NoSQL databases. How they are, able to store the big data, in further, slides?

Refer slide time :(20:15)

Cassandra

- **Apache Cassandra** is highly scalable, distributed and high-performance NoSQL database. Cassandra is designed to handle a huge amount of data.
- Cassandra handles the huge amount of data with its distributed architecture.
- Data is placed on different machines with more than one replication factor that provides high availability and no single point of failure.



cassandra

Now, another data base, which supports, this particular data model, that is, NoSQL data model, is called, ‘Cassandra’. So Apache Cassandra is highly scalable, distributed and high-performance, NoSQL database. So Cassandra is designed, to handle the huge amount of information and the

Cassandra handles this huge data, with its distributed architecture, that we will discuss, in this part of the course.

Refer slide time :(20:47)

Cassandra

In the image above, circles are Cassandra nodes and lines between the circles show distributed architecture, while the client is sending data to the node

These particular notes, shown over here, that is, the notes are, I mean, they presented, in the form of a, in the ring. So, all the notes are you see that, they are, placed around circle, that is called a, 'Ring'. And, this particular ring, is the architecture, how, this particular notes are, basically the data centre notes are, organized, virtually in the Cassandra architecture.

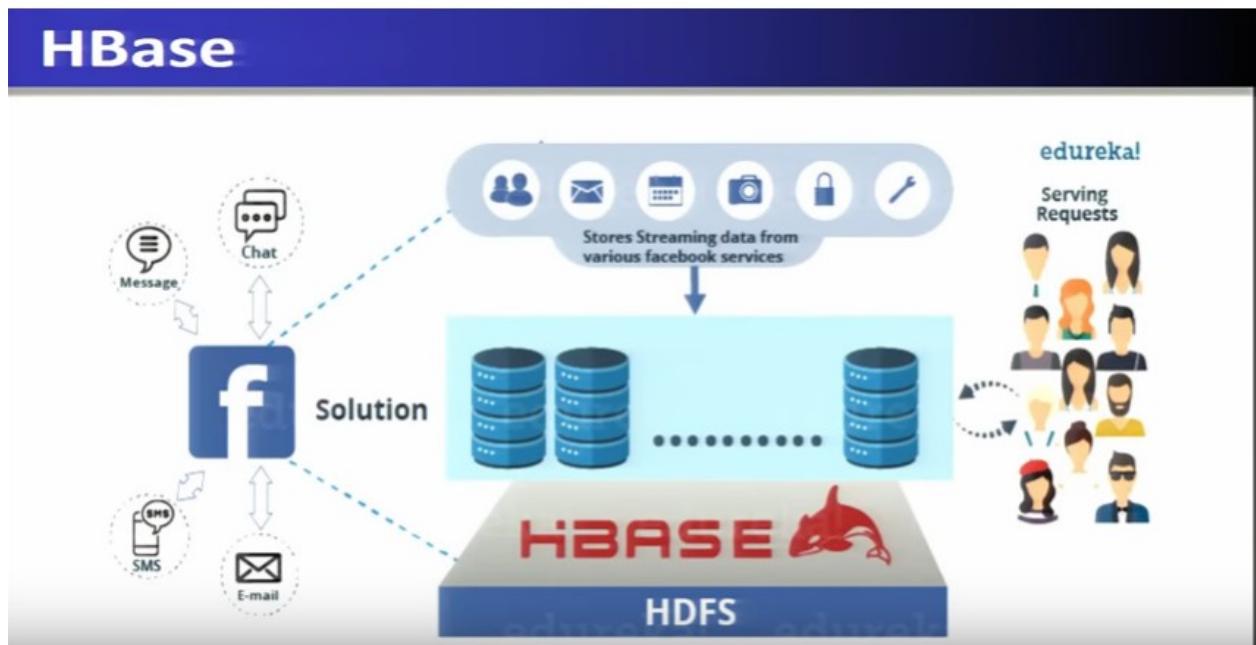
Refer slide time :(21:18)

HBase

- **HBase** is an open source, distributed database, developed by Apache Software foundation.
- Initially, it was Google Big Table, afterwards it was renamed as HBase and is primarily written in Java.
- HBase can store massive amounts of data from terabytes to petabytes.

We will see these things, in more detail. So, HBase is an open source, distributed databases, which is developed by, Apache Software foundation. Initially, it was the Google Big Table and afterwards, it was re-named as, HBase and is primarily, written with high level program language. HBase can store massive amount of data from, terabytes to petabytes.

Refer slide time :(21:44)



So this is, an HBase which runs over HDFS. And

Refer slide time :(21:52)

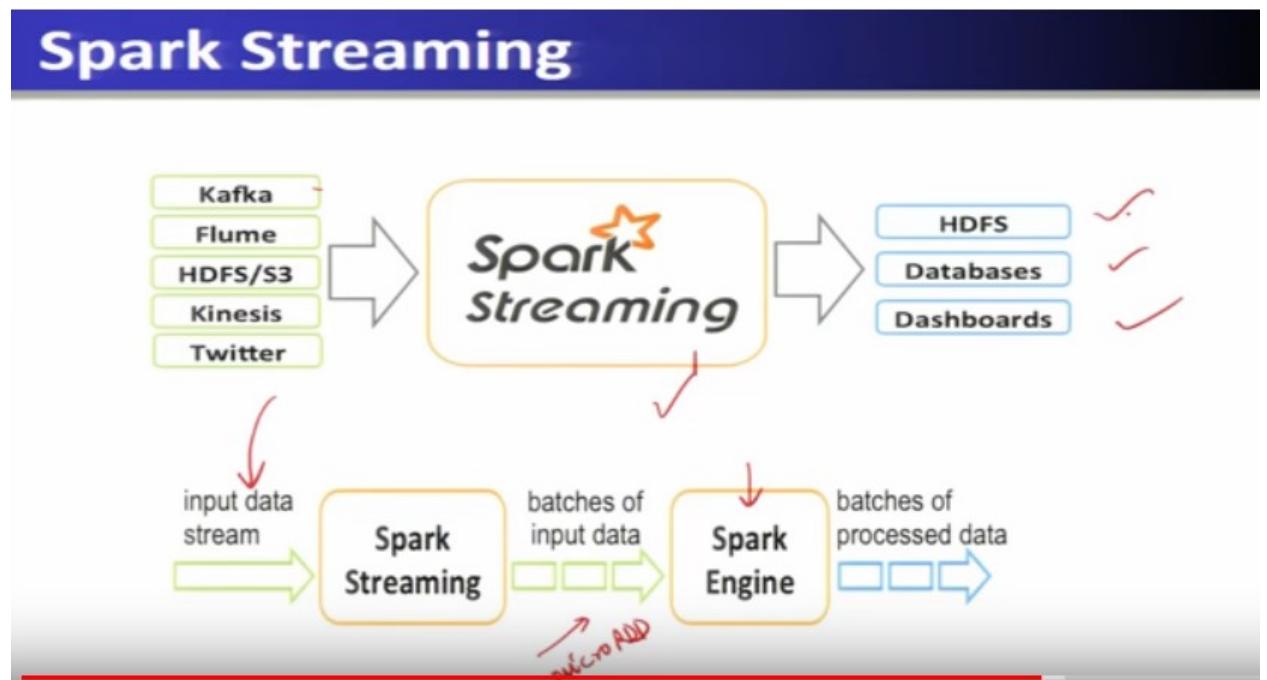
Spark Streaming

- **Spark Streaming** is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams.
- Streaming data input from HDFS, Kafka, Flume, TCP sockets, Kinesis, etc.
- Spark ML (Machine Learning) functions and GraphX graph processing algorithms are fully applicable to streaming data .



now, another project is called, ‘Spark Streaming’. So, Spark Streaming is used for real time stream processing applications and this also is called a, Fast Data Computations. So, stream processing, Spark Streaming, is an extension of the Spark core, that enables scalable, throughput, high fault tolerance, stream processing, of live, data streams. Streaming data, input from HDFS, Kafka, Flume, TCP, etc., is taken up, in this particular system, for computation and is Spark MLlib functions and Graph, GraphX, are fully compatible to the streaming data for computation.

Refer slide time :(22:44)



This particular diagram explains the entire computation of the stream processing. So at the core you can see there are is a Spark Streaming and the data. Which is, in the form of the stream, enters, into the Spark Streaming system, using, the data will captured, either using Kafka, Flume, HDFS, Kinesis and Twitter. And Spark Streaming, will, do the computation, on the streams of, which enters into the Spark core. And after the computation, the data will be, output will be stored, in HDFS or in databases or basically the visual analysis, can be shown in the dashboard. So, for example, if whenever, stream, for example, here, the Twitter Stream, or Kinesis or HDFS or Flume, Kafka, is right in, input in the form of a Stream, through the Spark Streaming, it will be divided into the form of micro batches, this is micro RDD, that we will explain later on. And these micro batches, will be processed, into the Spark engine. And, and then, this processed values are being, generated output, either in the form of, either will be stored in HDFS, databases or dashboard. That we have, that we will see, in more detail later on.

Refer slide time :(24:13)

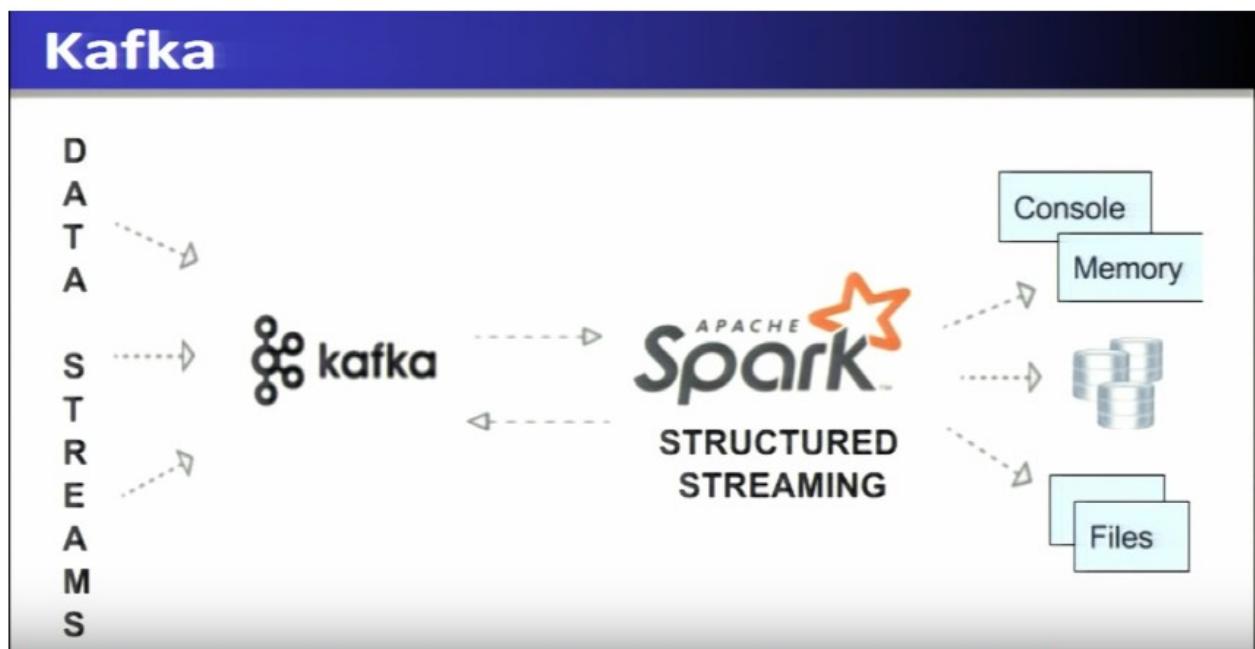
Kafka, Streaming Ecosystem

- **Apache Kafka** is an open-source stream-processing software platform developed by the Apache Software Foundation written in Scala and Java.
- Apache Kafka is an open source distributed streaming platform capable of handling trillions of events a day, Kafka is based on an abstraction of a distributed commit log



So, the data capturing method for, Streaming is, one of such method is called, ‘Apache Kafka’, is an open source, distributed stream processing, software framework and this we will discuss later on.

Refer slide time :(24:31)



So, through Kafka data streams can be, submitted to the Apache Spark, for doing the computations. So this will form a pipeline. So, we will see, how this stream processing, data pipeline, we can form, using these different projects, which are open source frame work available, Apache Open Source framework.

Refer slide time :(25:03)

Spark MLlib

- **Spark MLlib** is a distributed machine-learning framework on top of Spark Core.
- MLlib is Spark's scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction.



The next project is called, ‘Spark MLlib’. Spark MLlib is a distributed machine learning framework, on top of Spark core. So, MLlib is the Spark’s scalable, machine learning library, which consists of common, machine learning algorithm and utilities. Such as, the Classification algorithm, Regression algorithm, Clustering, Collaborative, filtering, and Dimensionality reduction and all the algorithms, which are there in machine learning, they are implemented in this particular frame work. And that is why they are called, ‘Scalable machine learning’ and it is there available in the form of libraries that we will see.

Refer slide time :(25:33)

Spark MLlib Component

Algorithms

- Classification
- Regression
- Clustering
- Collaborative Filtering

Pipeline

- Constructing
- Evaluating
- Tuning
- Persistence

Featurization

- Extraction
- Transformation

Utilities

- Linear algebra
- Statistics

So, the algorithms which are available are shown over here. And we will see in more detail, how we will we are going to use them, in this big data, I mean, Ecosystem.

Refer slide time :(25:49)

Spark GraphX

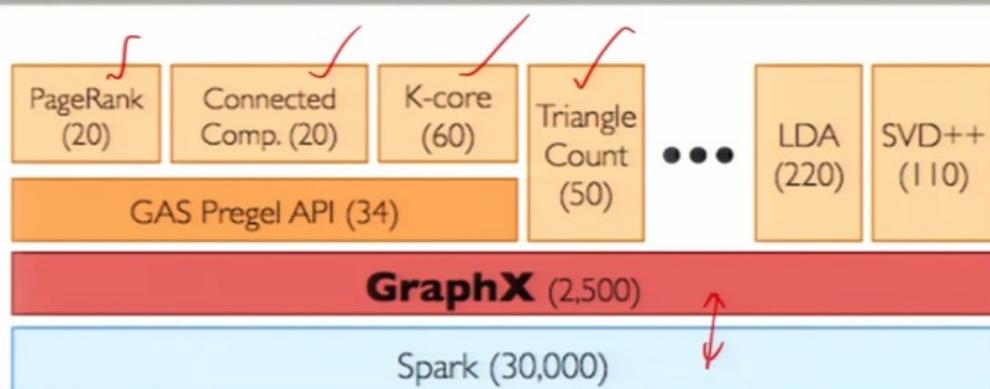
- **GraphX** is a new component in Spark for graphs and graph-parallel computation. At a high level, GraphX extends the Spark RDD by introducing a new graph abstraction.
- GraphX reuses Spark RDD concept, simplifies graph analytics tasks, provides the ability to make operations on a directed multigraph with properties attached to each vertex and edge.



So, Spark GraphX is another Hadoop open source, Apache project and this is the component, which is build over the, core Spark, for computation of a large scale graphs. That is parallel computation of a graph is done using GraphX. So, GraphX extends the Spark RDD's, by introducing the new graph abstraction. And GraphX reuses by Spark RDD concepts that we will see litter on and simplifies than on top of it, using the different graph analytics, which basically are graph algorithms, which will be applied on this frame work.

Refer slide time :(26:42)

Spark GraphX



GraphX is a thin layer on top of the Spark general-purpose dataflow framework (lines of code).

So, GraphX if you see in more detail and this over Spark core and the algorithms which are available, for, for doing the graph analytics, are Page Rank algorithm, Connected Component algorithm, and the K – core and the Triangle Count, all these algorithms are already implemented and made available as a form in, in the GraphX. So, GraphX is thin layer on top of the Spark general purpose data flow and that we will see in more detail and this is being heavily used for, various analytics, that is, ‘The Graph Analytics’.

Refer slide time :(27:25)

Conclusion

- In this lecture, we given a brief overview of following Big Data Enabling Technologies:
 - Apache Hadoop
 - Hadoop Ecosystem
 - HDFS Architecture
 - YARN
 - NoSQL
 - Hive
 - Map Reduce
 - Apache Spark
 - Zookeeper
 - Cassandra
 - Hbase
 - Spark Streaming
 - Kafka
 - Spark MLlib
 - GraphX

In conclusion, we have briefly, overviewed the big data enabling technologies, in short, we have discussed about the, Hadoop Ecosystem, which comprise of, the, the HDFS, File system, YARN, Map Reduce, on then, all map reduce we have seen, Hive and Pig. And then, various NoSQL data databases, we have seen, the, the Cassandra, HBase and then we have seen, using this HDFS, then we have seen in this, framework, Apache Spark and or Spark we have also seen, the, the frame work, for the Streaming computation, that is Spark Streaming and graph computation, that is called GraphX and for machine learning, we have seen MLlib. Also we have seen our Kafka, for, getting the, means, streams, into the Spark Streaming system, for computation. So, that is all and in the next videos, we will be going in, more details, of these different, Apache Spark project, for big data computation and we will see how, we will be using them for different applications, these projects.

Thank you

Lecture 3

Hadoop Stack for Big Data

A Hadoop stack for Big Data preface

Refer slide time :(0:17)

What is Hadoop ?

Today's leading technology is hadoop which is an open source framework for reliable scalable distributed computing for big data analytics.

- Apache Hadoop is an open source software framework for storage and large scale processing of the data-sets on clusters of commodity hardware.

100 & 1000s of nodes

*- Storage
- Large Scale Processing
- Hadoop
- Cluster*

Content of this lecture, in this lecture we will provide insight into Hadoop technologies, its opportunities and challenges for Big Data computing. We will also look into the Hadoop stack and the various applications and technology associated with it for Big Data solutions; let us trace back the beginning of the Hadoop. So, today the leading Big Data technology is Hadoop, which is an open source, software framework, for the reliable, scalable distributed computing, for big data analytics. So, this particular Apache Hadoop, open source software framework is free and is used for storage and, large scale processing of a big datasets, on cluster of commodity hardware. So, we assume here in this case there exists a, a cluster of commodity machines which is nothing but in the form of the racks, and several such racks, will form a cluster of nodes. So, this particular Hadoop will run on this cluster machine, and provides, the solution for storage. So, storage will be available on these nodes ,which are hundreds and thousands of the nodes, which Hadoop will use as a storage.

Similarly the computation on these particular data sets, which are stored on this particular cluster, will require the large-scale computation. Wherever the data is stored. So, the data so this Hadoop, will provide a framework for storage and the large-scale processing, large-scale processing, by this we mean that the data set is too large and it cannot fit in the existing, conventional systems, maybe whatever is the size of the conventional system, one system cannot basically accommodate the entire data set. So, for a large size data set it requires hundreds and thousands, of such machines to store the data. Now when a computation is to be performed on these particular data sets, then it is not possible to bring all the data at one place for the computation rather, computation has to be moved wherever data is stored hence, it is called a large-scale processing or a computation of that particular data set, and that is possible with the help of clusters with of commodity hardware, which comprises of hundreds and thousands of machines. So, in this particular discussion we will first see the, the functionality of Hadoop, we will see its beginning also and

all these things we are going to discuss. So, again in a nutshell we have to before we proceed ,we have to understand that the today's leading technology, leading big data technology is Hadoop , which is, which is an open source framework for reliable, scalable, distributed computing, for the big data analytics, we will explain in this particular lecture. What is we mean by the reliable? What we mean by the scalable? And this particular entire set up of computation on a cluster is called distributed computing, and the large-scale data sets are, being computed and being performed the analysis which is called a big data analytics, which is useful for driving various applications all these things is uncovered, under the Hadoop that is why today's leading Big Data technology is Hadoop. So, with this simple explanation, now we will see into the Hadoop. What this Hadoop? Means, what are the different application it provides and which we are going to use for computing the big data.

Refer slide time :(06:26)

Hadoop Beginnings

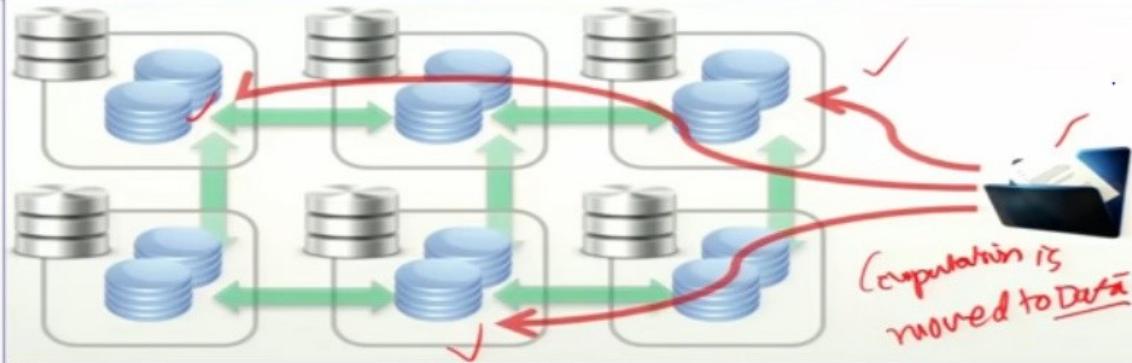
- Hadoop was created by Doug Cutting and Mike Cafarella in 2005
- It was originally developed to support distribution of the Nutch Search Engine Project.
- Doug, who was working at Yahoo at the time, who is now actually a chief architect at Cloudera, has named this project after his son's toy elephant, Hadoop.



So, Hadoop was created by, the Cutting and Mike Cafarella in 2005, that is long back. It was originally developed to support the distribution of the nudge search engine project. Doug Cutting, who was working at yahoo at that time, who is now chief, Chief architect at Cloudera, named his project after his sons toy which is an elephant and named as Hadoop. So, Hadoop is the name given to this particular project out of this context. So, this particular icon, which is their representing the Hadoop, is that is why, which Doug Cutting has named this particular project Hadoop.

Refer slide time: (07:33)

Moving Computation to Data



- Hadoop started out as a simple batch processing framework.
- The idea behind Hadoop is that instead of moving data to computation, we move computation to data.

Now, let us go in more details of this particular leading, Big Data technology which is Hadoop. Now, here we see that this big data is too large; to be fit in to a particular system hence it is being stored in the cluster systems. Now, if it is stored in hundreds and thousands, of the nodes then the computation also has to be moved, wherever the data is stored see these particular arrows. So, this shows that the computation, if you want to perform the configuration on this large-scale data sets. So computation has to move wherever the data is stored. And after doing the computation the results will be collected back to the same environment. So, hence the Hadoop is started out as a simple batch processing framework. Why because data is to be store, and that computation can be performed at any point of time, and this is called a batch processing framework. So, Hadoop initially was designed as the simple batch processing framework, and the idea behind the Hadoop is that instead of moving data, to the computation, here we have shown you in this particular picture, the computation is moving, move to the data where our data is stored, and the computation is performing hence the title says that moving. The computation to the data is one of the main idea behind the success of the Hadoop which is the leading technology for big data.

Refer slide time :(09:24)

Scalability

- Scalability's at it's core of a Hadoop system.
- We have cheap computing storage.
- We can distribute and scale across very easily in a very cost effective manner.

Scalability -
→ thousands of commodity HDFS
Scale out -

The second aspect is called scalability, by scalability we mean that, that whatever is the size and this particular size, of particular data, can be processed stored, and being computed that is called scalability. This is achieved using hundreds and thousands of, of machines, which is called a commodity, hardware machines, why commodity hardware they are not very specialized or super specialized computers they are normal computing machines hence it is called commodity hardware, and this ensures our scalability. So, as this particular this is called scale out, scale out technology says that, if you keep on adding more resources, more hardware it will provide, more resources to accommodate this kind of and this is called scalability. So, again to repeat that the scalability is at the core of the design of Hadoop system, and it is basically achieved out of hundreds and thousands of commodity hardware, which is a cheap computing device for storage and for computations. So, hence a large data set, is can be distributed across these hundreds and thousands of the machine, and we can add these machines more in number to scale without any modifications, and we can get more improved performance if more number of such machines, we keep on adding to accommodate the, the scale or the size of the data ion also to form the computation, at higher-speed in a very cost-effective manner hence it is called a scalability.

Refer slide time :(11:27)

Reliability

- Hardware Failures Handles Automatically!

failure is norm!



- If we think about an individual machine or rack of machines, or a large cluster or super computer, they all fail at some point of time or some of their components will fail. These failures are so common that we have to account for them ahead of the time.
- And all of these are actually handled within the Hadoop framework system. So the Apache's Hadoop MapReduce and HDFS components were originally derived from the Google's MapReduce and Google's file system. Another very interesting thing that Hadoop brings is a new approach to data.

Now, another important thing is so, this is called reliability. So, when we see that we are using the hundreds and thousands of commodity machines, commodity hardware which are not very sophisticated, hence they are prone to the failures. So, failure is a norm, and this is the basically prime, design, aspect which is called reliability that is the fault tolerance is one of the basic design paradigms for in the design of the hadoop system that we will see. So, if we see that an individual machine, are basically the rack of machines, or a large cluster or the, the big supercomputer they can fail at any point of time or some of their components can also fail. So, failure is very common, and that, that have to be accounted for this kind of failure well ahead in time, without any disruption in the application computation. So, and all of these are actually handled within the Hadoop framework. So, Apaches Hadoop Map Reduce and HDFS component were originally derived from Google's, Map Reduce, and Google's, Google file system. So, there the design of the Google file system, and Google Map Reduce already has attended or handled this reliability or a fault tolerant as one of the basic design issue. So, we will see that how this is achieved here in the hadoop system about, reliability of these failures to tolerate with that.

Refer slide time :(13:19)

New Approach to Data: Keep all data



- A new approach is, we can keep all the data that we have, and we can take that data and analyze it in new interesting ways. We can do something that's called schema and read style.
- And we can actually allow new analysis. We can bring more data into simple algorithms, which has shown that with more granularity, you can actually achieve often better results in taking a small amount of data and then some really complex analytics on it.

Now, here doing all this new approach to the data keeping all the data always. So, we see that in the new approach in Hadoop, we are going to keep all the data we have, and we can take the data and do the analysis or in many interesting ways. So, we are not constrained about fitting the data into the schema and then storing it into that form rather all the data will be kept as it is and whenever it is read it will be fitted into the schema and being available for the analysis. So, that means that keeping all the data that means the data will be while reading, while reading the data, it will be fit into the reading into the schema. So, this will simplify the analysis, and at the time of storing, we have to just keep all the data without any constraint of fitting it into the schema, and this will bring into the more data into the simple algorithms, and the analysis becomes, easier in here in this case. So, while we will discuss the analytics, part complex analytics, when we perform on the big data then, we will see that this aspect of this particular design that is reading into the schema becomes, quite easier to design the complex analytics on it.

Refer slide time :(15:00)

Apache Hadoop Framework & its Basic Modules

~~Summary Design Issues in Hadoop~~

- ↳ Reliability
- ↳ Scalable
- ↳ Keeping all data ready on schema
- ↳ moving computation to the data

So, with this different design issue, which we have discussed, let us summarize them one the issue which was called reliability. The second design issues, that we have just covered let us summarize, it first issue was the reliability, scalability, and keeping, all the data. So that while reading on the schema, and this was the different and moving data to the computation, moving computation, to the data. So, these are the four different main design issues, which basically have made the leading technology for big data computation, which has made this Hadoop as the leading technology for big data computation, Apache Hadoop framework and its basic modules.

Refer slide time :(16:34)

Apache Framework Basic Modules

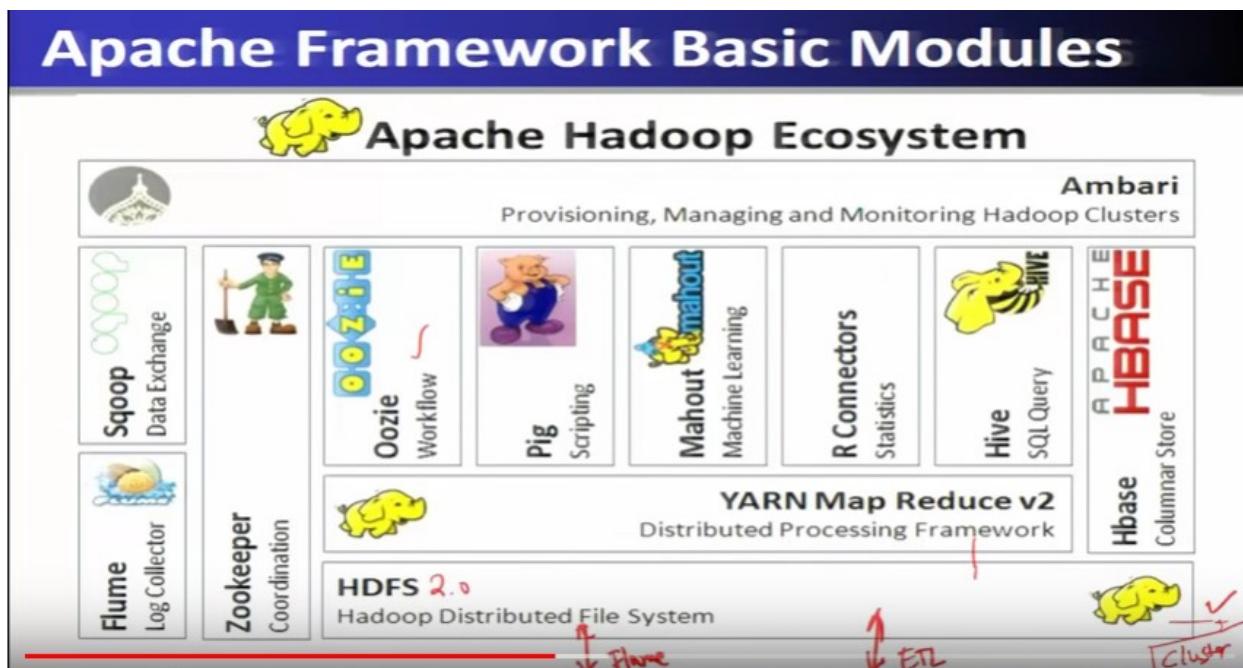
- **Hadoop Common:** It contains libraries and utilities needed by other Hadoop modules.
- **Hadoop Distributed File System (HDFS):** It is a distributed file system that stores data on a commodity machine. Providing very high aggregate bandwidth across the entire cluster.
- **Hadoop YARN:** *Resource Manager Scheduler ✓* It is a resource management platform responsible for managing compute resources in the cluster and using them in order to schedule users and applications.
- **Hadoop MapReduce:** It is a programming model that scales data across a lot of different processes.

So, Apache Hadoop has four different basic modules, the first one is called Hadoop Common, it contains the libraries and utility is needed by other Hadoop model, modules. The second module is called Hadoop

distributed file system that is HDFS, it is a distributed file system that stores the data on the commodity machines. So, this file system will insure how this particular data or a big data is to be distributed, and stored or across different hundreds and thousands of the node, and keep track of all these data whether they are available, or on the node whether the nodes are alive containing those data or they are not alive in all the cases this particular, Hadoop distributed file system will do the management, and user will be given this kind of service. So, this Hadoop will provide a very high, aggregate bandwidth across the entire cluster, So this storage and the retrieval of the huge amount of data over the commodity, over the cluster, will become is possible here using Hadoop to provide this axis at a very high educated bandwidth so, that it is, it's performance also we are going to cover in this part of the course. Now, the next module which is another basic module of Apache Hadoop is called YARN. So, YARN is the resource manager or it does the resource management for managing the computer resources in the cluster and using them in order to schedule the user ion application.

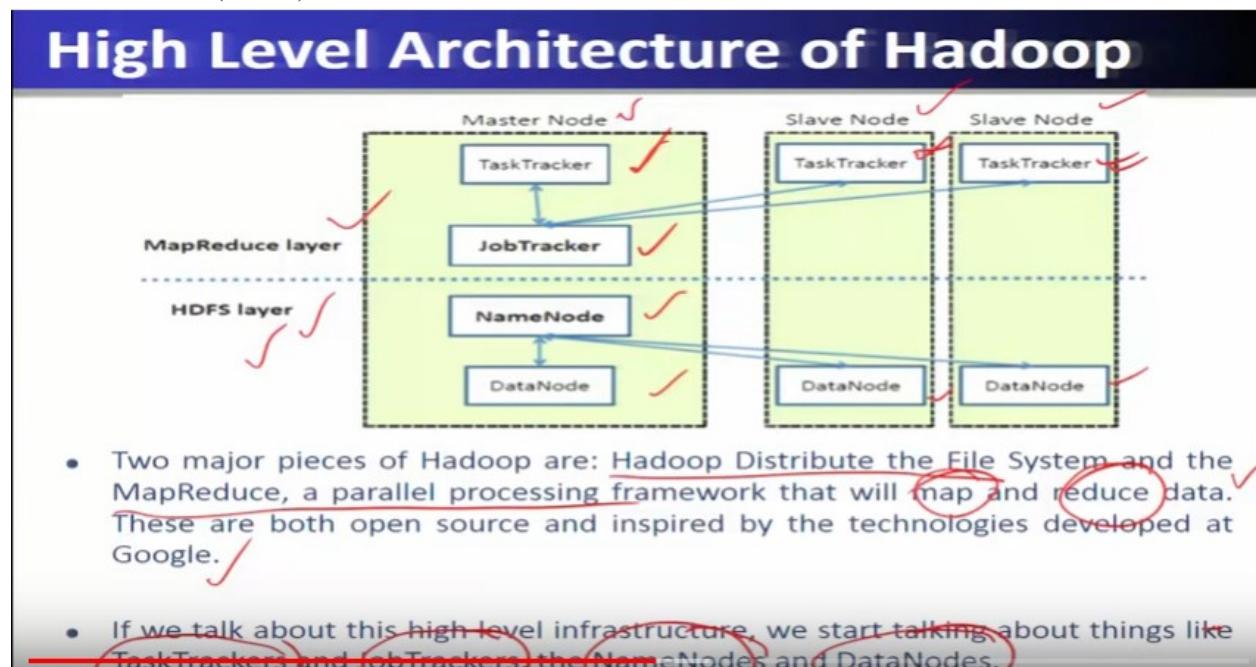
So, it is the resource manager and the she jeweler, by saying that resource manager and the she doula means the resources which are available on the cluster it will keep track of it and whenever is required by the different application, it will do the she ruling that means it will allocate and schedule for different resources, for different users and application that is called yarn in the map in the hadoop. And the fourth basic framework of Hadoop is called Map Reduce this is the programming paradigm and which will ensure the computations will reach wherever the data is using this particular program that is called Map Reduce. So, Map Reduce is the basic programming model, using which all the programs, application programs can be, can be designed to run in this entire cluster system. So, this particular programming model ensures that scalability is ensured while it is achieving the performance, while computing the big data that we will see in more detail about all these four different modules of Apache Hadoop.

Refer slide time :(20:01)



So, Apache Hadoop basic modules, we can see here and in this particular slide, which basically is about the Apache Hadoop ecosystem. Let us see at the end or at the bottom most, part of this particular ecosystem, here is an HDFS, and below this the data will be pulled either through the flume or through the databases, that is the ETL, which will provide the data, into this particular system which will be stored using Hadoop file system on the cluster. So, this HDFS, Hadoop file system, Hadoop distributed file system will run on the cluster. So, the hardware is the cluster on which the Hadoop distributed file system would run yarn, and will be running over top of HDFS. So, this HDFS is 2.0 versions. So, YARN and Map Reduce version 2 they will run on top of this HDFS, and using YARN and Map Reduce there are different other applications, of Hadoop system which will run that is called Suzie Pig, Mahout and hi HBase, Sqoop, Flume, Zookeeper. Now, let us see in more detail about all these different applications, which are available in the Hadoop ecosystem and we will summarize, them they are used and they are going to be useful for the Big Data computation.

Refer slide time :(21:58)



Before going, there let us see the high-level architecture of the hadoop. So, as you know that the Hadoop runs on the cluster system, and cluster system comprises of hundreds and thousands of the nodes, and each node, an they will be running different applications, or the modules of Hadoop system and we call them one such models called the master, the other nodes are called the slave nodes. So, as far as Map Reduce HDFS, layer is concerned so this particular master will contain, master node contains the module of HDFS layer, which is called a name node and all other nodes will contain another module of HDFS layer which is called a data node. So, all the data nodes which are running on different nodes will communicate with a single name node of HDFS layer, similarly these nodes are also being used for the, the Map Reduce layer. So, the Map Reduce layer will contain the task tracker this particular job, of task tracker will run on those nodes similarly there will be for every node there will be a task tracker and whenever the jobs are running they will basically form the task job tracker. So, task trackers will be running on all the nodes and there will be a single job tracker. So, for every application this job tracker

will be created and the task of that job or the application will be actually running using the tasks using the task tracker. So, this particular combination of these two different services run by the Map Reduce and HDFS, will run on will be launched on the same set of the nodes. So, hence two pieces of Hadoop are, the two major components of the Hadoop are HDFS, file system and the Map Reduce parallel processing framework and that will map and reduce the data. Means that the data which is stored by HDFS will be performed for computation, that is called parallel processing framework given by the map and reduce. So, the programs which are called map will reach the computation wherever the data is stored called a map, and after performing those computation the aggregate values are being collected that is called reduce of that particular on that data item and these technologies were initially conceived at the Google, and which was open source by the Yahoo, that is an Apache Hadoop, which is basically available on free for big data computations. Here, in this particular high-level architecture we have referred the names like task tracker, job tracker, name nodes on data nodes all these different components which are used here in this high-level infrastructure, for describing the architecture of Hadoop is going to be described in more detail in further slides.

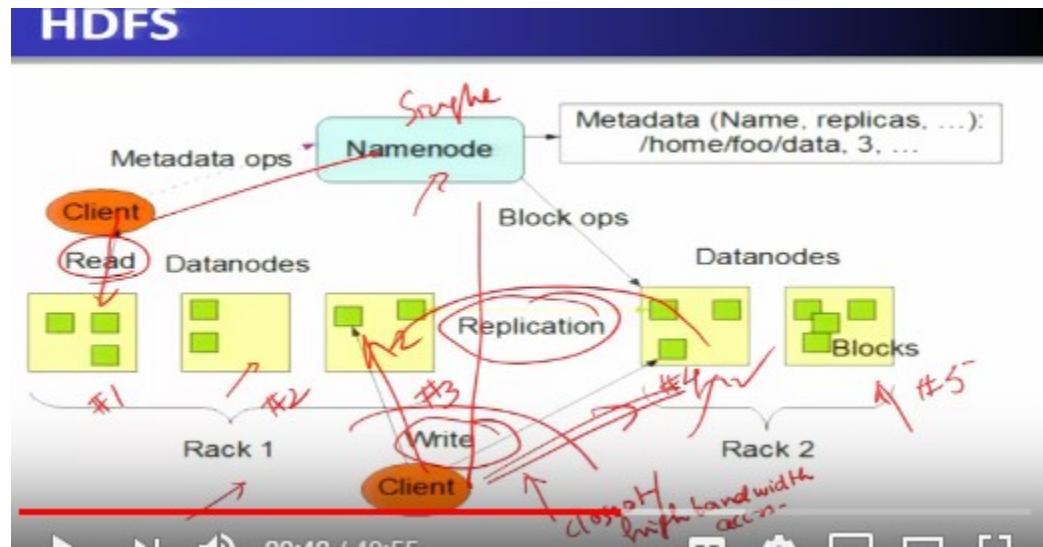
Refer Slide Time :(25:56)

HDFS: Hadoop distributed file system

- Distributed, scalable, and portable file-system written in Java for the Hadoop framework.
- Each node in Hadoop instance typically has a single name node, and a cluster of data nodes that formed this HDFS cluster.
- Each HDFS stores large files, typically in ranges of gigabytes to terabytes, and now petabytes, across multiple machines. And it can achieve reliability by replicating across multiple hosts, and therefore does not require any range storage on hosts.

So, as GFS stands, for Hadoop distributed file system which is a scalable, reliable, distributed computing or storage platform and this particular platform is, written in the Java, now the node in Hadoop instance has a single name node and also the different data nodes, together will form a HDFS cluster. So, that means, that in HDFS cluster there is, one node which is called as our name node there is a single name node, and there are multiple data nodes which basically, will be running this HDFS cluster. Now each HDFS will store the large files typically, in the range of kilobytes to, terabytes and, now the petabytes, across multiple machines, and it can achieve the reliability by, replicating across multiple hosts, and therefore does not, require any range storage on the hosts.

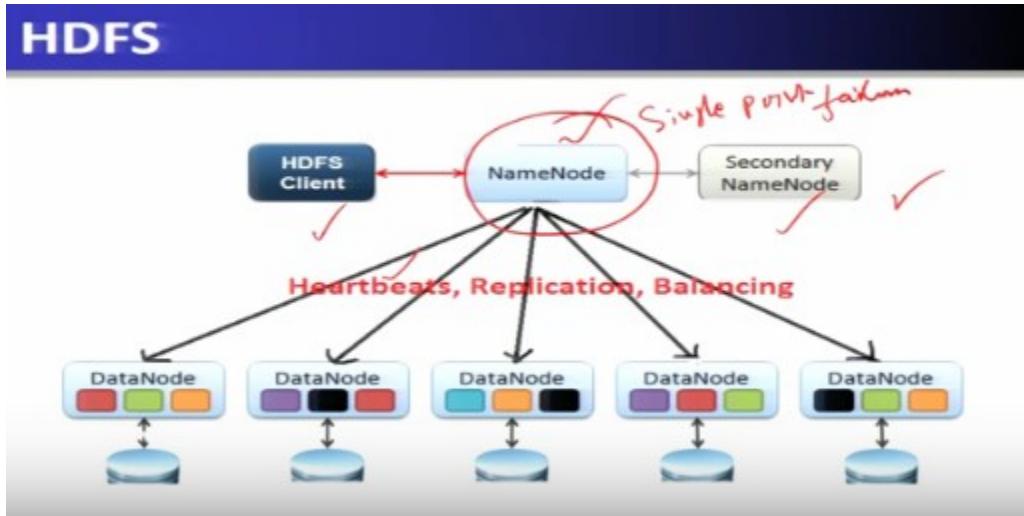
Refer Slide Time :(27:23)



So, let us see the more details of these name nodes, and data nodes in this particular diagram you see that there is, a single name node there are multiple data nodes which are running on different machines. let us assume that this rack has machine one, machine two, iron machine three, different nodes similarly, on another rack it has another machine that is, machine number four, and machine number five, these different one, two, three, four, these machines are running the data nodes and there is one of them will be running a single name node on the same machine let us assume that, this particular HDFS this is, the architecture of HDFS when it runs on the on, on the cluster. Which comprises of several racks? So, whenever a client wants to do, the operation which is called the write operation, and the read operation, they have to go through this, particular name node, and name node, will then guide them where the data is stored actually, and the closest of them for example data is stored at the node 3 and as, for as node 4 so, it will prefer the closest one this is called closest. To provide the high bandwidth access, to the data similarly and it will write on this particular the closest one and then all the replicas, for example replication of this, particular data at more than one nodes, will be done in the terms of blocks, that is called replication these details we are going to explain in the further slides, in more detail similarly there is another operation which is called a read so when a client wants to read a particular data which is stored in HDFS it will know, through the name node where those data is stored and then the closest one or the nearest one replicas will serve as, the read for that client.

Refer Slide Time :(29:46)

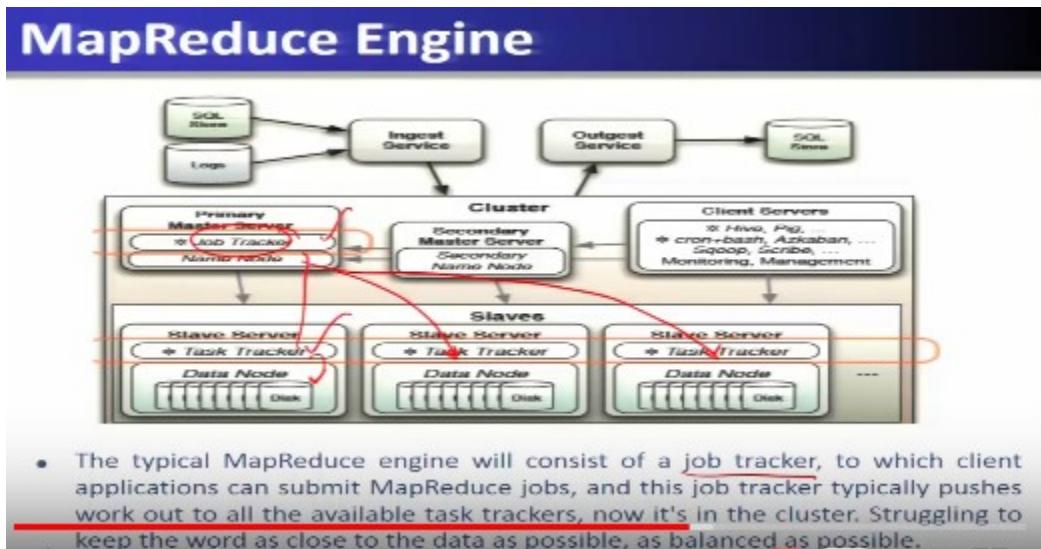
HDFS



Now as far as the name note is concerned there is a single name node it will keep track of the data which are now stored on the data nodes, whether they are alive or they are dead so hence these heartbeats are continuously exchanged between the name node and, the data node to know, the position and the situation of the data which is being managed by or restored by, the data nodes and whenever they are down that means a fault tolerance has to take place then it will be removed out of this, name node name node will not give the reference to all other clients, requests for that particular date are not referencing and when ever again it will come up then again it will be modified or it will be made consistent, synchronized with all the previous updates, for the replicas and then only it will be made available to the further client requests. Now there is a single name node which is a prone to the failure, single point, failure to avoid this kind of situation a secondary name node passively, try to keep a most recent backups so, if the name node downs, then secondary name node can become the primary name node and can continue to operate in this, environment that is Hadoop distributed file system.

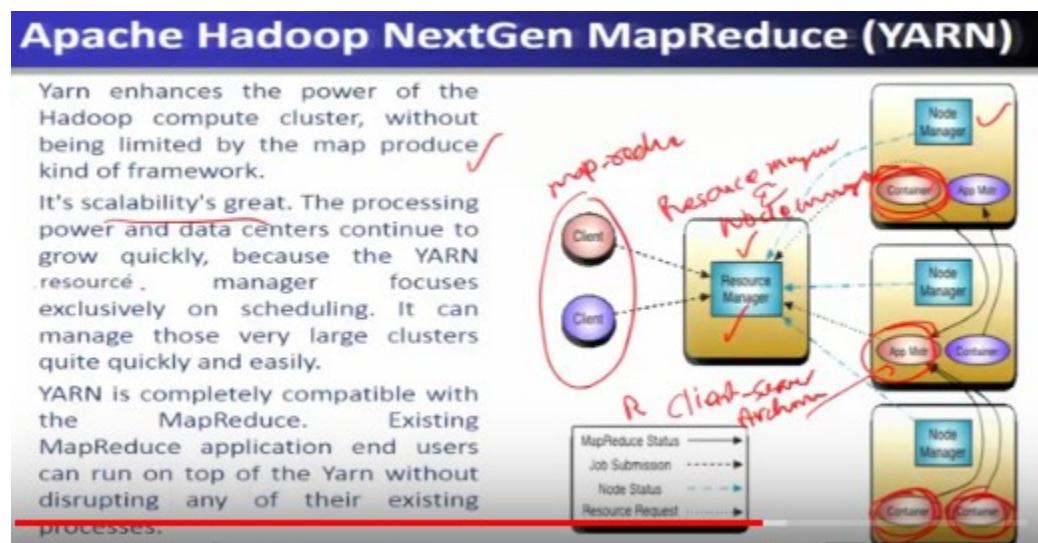
Refer Slide Time :(31:19)

MapReduce Engine



Operations, now the compute engine which is called the Map Reduce engine will perform the parallel computation or this, architecture which is provided by HDFS so, if you see here that HDFS was having name node and the data nodes, the similar kind of notion ,Map Reduce also, will provide so, Map Reduce will have the job tracker, and task tracker, so job tracker, will now keep on, tracking or keep on communicating with the help of task trackers, and the task trackers, will actually use the, the data nodes, and perform the contradictions, that is the map operation wherever the data is available that is done by, the task tracker, and after the computation the reduced function will be performed and hence there is, a combination of a job tracker and, a task tracker so, again, let us summarize the typical, Map Reduce engine will consist of a job tracker, to which the client applications can submit the Map Reduce jobs, and this job tracker typically pushes, the workouts to all other available task trackers so, as I told you that this job tracker will push out to all other task trackers, now it's the cluster, now which is there in the cluster? struggling to keep the words as close, as close to the data as possible and it will be done, through the balancing so, all these optimizations are basically, online managed by, these set of tasks job tracker and, the task tracker combination in the Map Reduce engine.

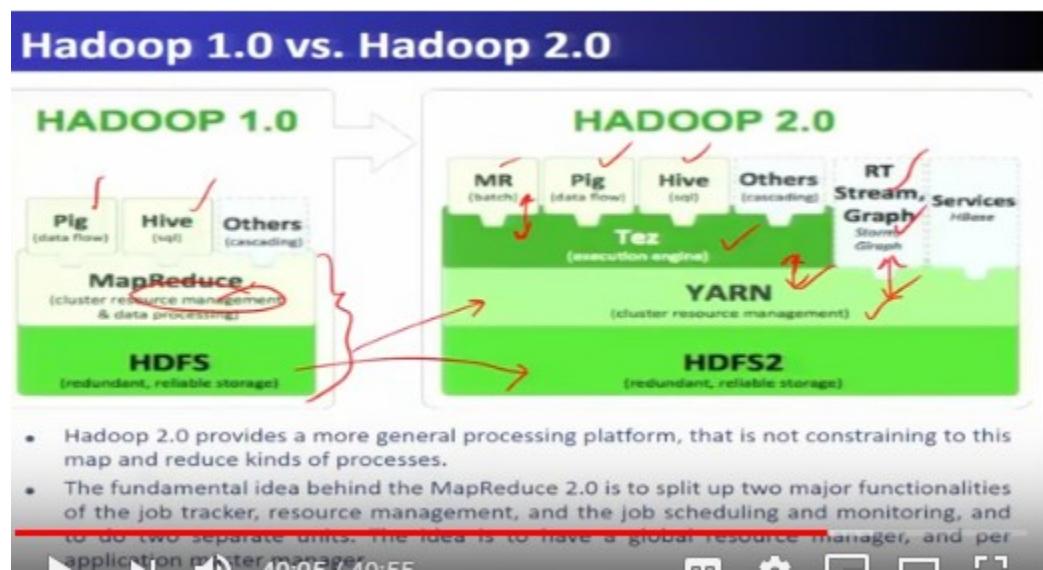
Refer Slide Time :(33:22)



Now the next important component of the Hadoop is called YARN, now YARN NextGen YARN, is there to provide two important services, one is called resource manager which is, is which is, at the core at the heart, the other one is called scheduling how? this, particular different resources are scheduled to the Map Reduce jobs, and this task, this is the task, of the YARN so, YARN enhances, the power of the Hadoop compute cluster without being limited by, the map produced Map Reduce kind of framework now its scalability, is great in the sense that the processing power and the data center continue to grow quickly because, YARN Resource Manager focuses exclusively, on scheduling it can manage those very large clusters quickly, and easily, so YARN is completely compatible with Map Reduce and existing Map Reduce application and user can run on top of YARN without any disrupting of their existing processes, let us see through this particular figure about the components ,of the YARN so, YARN has two components here shown as the resource manager and, the node manager so, there is a single resource

manager so, whenever the client submits their job let us say it's a Map Reduce job which requires, the resources are to be allocated hence this is acting as the resource manager. So, the resource manager after receiving the request, from the client it will now, contact to the node manager and, the node manager in turn will ,will assign the resources as far as, the application is concerned every application has, application master and whatever resources is being allocated by, the node manager through the resource manager these, particular are assigned to the application master and which is known, as the containers here they, are shown as in the pink color so, this particular application master will know where? what are the different containers, are allocated by, the node by the node manager and this, particular application master will independently run so ,these application master container they are, are the resource blocks, which are allocated by ,the resource manager with the help of node manager so, note when a resource manager and, the node manager, is the client-server architecture of the YARN resource manager and, the node manager is, client-server architecture so, YARN is work in the client-server architecture where the node manager will take the request from the client, and with the help of several node managers it is trying to allocate the resources and do the Scheduling of these activities, we are going to discuss this in more details.

Refer Slide Time :(36:58)



Now, we will see that, the transition from Hadoop 1.0 to, Hadoop 2.0 so, Hadoop 2.0 has, become very much flexible so, newer applications are also able to execute or under Hadoop 2.0 which was not possible using these drawbacks, of Hadoop 1.0 let us, see these details why nowadays, we are using Hadoop 2.0 and, discarded to use the original Hadoop 1.0 so, Hadoop 1.0 has, two components only, that is HDFS and Map Reduce so, that means the resource management and scheduling also was the part of the, the programming paradigm that is called Map Reduce with this, release of this responsibility a YARN is being included so, that is source management and scheduling which is done by, the YARN so, YARN is, a cluster resource manager, and HDFS therefore has some more capabilities, added hence its name is HDFS version 2, and which supports to run the yarn on top of it and about that YARN there is execution, engine which is called at age? And in some of the distributions, the stage is used and about age this Map

Reduce programs, will run as the batch processing of the big data computations, there are other applications such as Pig, Hive which used to run earlier in the version Map Reduce version 1, about the Map Reduce nowadays, this big and high are now running using the Map Reduce and this version 2, which is shown over here big and high that is running continuously, similarly, other applications such as, the stream processing ,or graph processing, and machine learning, all they run and storm and 0 up they run above the YARN system without the Map Reduce. So, Map Reduce 2.0 provides, a more germinal, processing framework that is not constraining to this map and reduced kind of processes, so the fundamental idea behind Map Reduce 2.0 is, to split two major functionality, of a job tracker resource management and the job scheduling and monitoring to two separate units, hence some of the responsibilities, that is the resource management responsibilities, of a Hadoop of a Map Reduce point one is now, broken up as Map Reduce version 2, YARN. Hence this becomes, more flexible and for paving a way to a newer application that we will see which are possible under Hadoop 2.0, version.

Refer Slide Time :(40:08)

What is Yarn ?

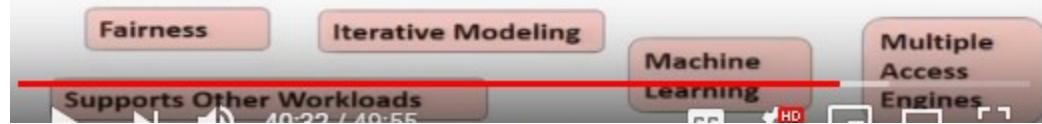
- Yarn enhances the power of the Hadoop compute cluster, without being limited by the map produce kind of framework.
- It's scalability's great. The processing power and data centers continue to grow quickly, because the YARN research manager focuses exclusively on scheduling. It can manage those very large clusters quite quickly and easily.
- YARN is completely compatible with the MapReduce. Existing MapReduce application end users can run on top of the Yarn without disrupting any of their existing processes.
- It does have a Improved cluster utilization as well. The resource manager is a pure schedule or they just optimize this cluster utilization according to the criteria such as capacity, guarantees, fairness, how to be fair, maybe different SLA's or service level agreements.

Scalability MapReduce Compatibility Improved cluster utilization

Refer Slide Time :(40:33)

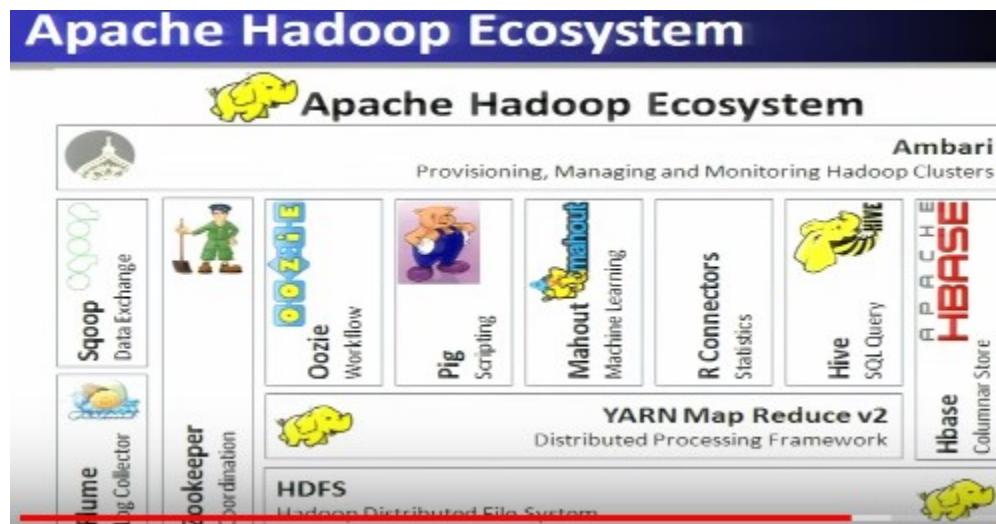
What is Yarn ?

- It supports other work flows other than just map reduce.
- Now we can bring in additional programming models, such as graph process or iterative modeling, and now it's possible to process the data in your base. This is especially useful when we talk about machine learning applications.
- Yarn allows multiple access engines, either open source or proprietary, to use Hadoop as a common standard for either batch or interactive processing, and even real time engines that can simultaneous acts as a lot of different data, so you can put streaming kind of applications on top of YARN inside a Hadoop architecture, and seamlessly work and communicate between these environments.



So, what is the YARN? So YARN is a resource manager? that is the full form of YARN is yet another resource negotiator so, YARN enhances, the power of Hadoop compute cluster without being limited by, the map and reduce so, map all these things, we have already covered.

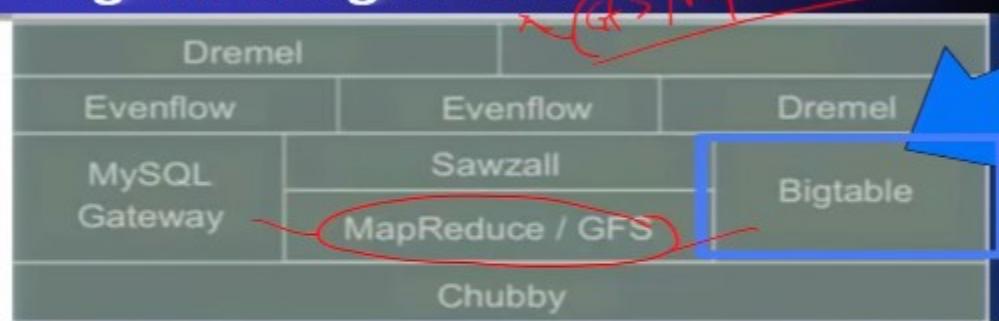
Refer Slide Time :(40:34)



Now, let us go and see what do, we mean by the Hadoop Zoo. Now in this Hadoop ecosystem we see there are lot of icons, and most of these icons, that is Hadoop is representing try a different similarly, there are some toy Pig and all these different animals, which are becoming an icon to these applications are require a coordination service hence the word Zookeeper, is being used as the coordination service. So we will see all these more applications, in more detail one by one.

Refer Slide Time:(41:12)

Original Google Stack

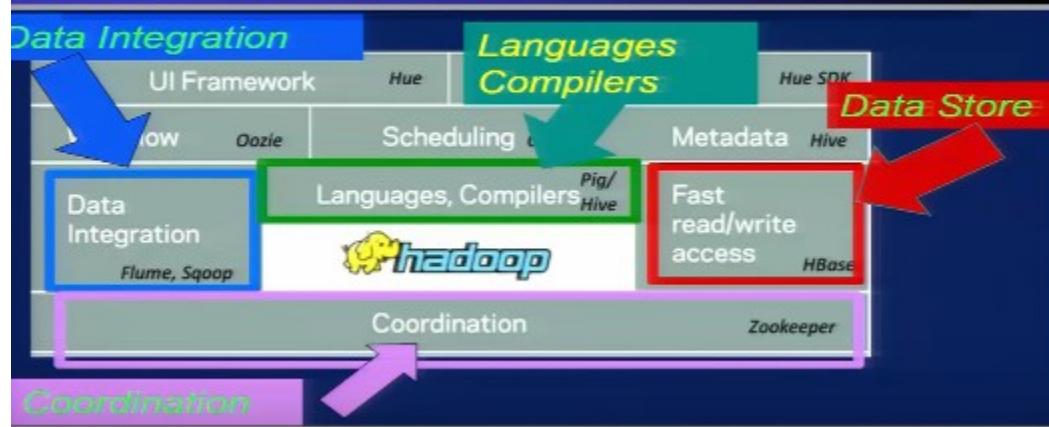


- Had their original MapReduce, and they were storing and processing large amounts of data.
- Like to be able to access that data and access it in a SQL like language. So they built the SQL gateway to adjust the data into the MapReduce cluster and be able to query some of that data as well.

Now before that we will see the distributions so, the distributions, of Hadoop are this big data is called the stack so, initially it was the, the Google GFS, Google file system and the Map Reduce hence it was called a Google stack initially, so Google stack was using initially, the Google file system and the Map Reduce and it was also using the database which is called a big table, and MySQL gateway and, and so, on so this, was called a Google stack.

Refer Slide Time :(42:00)

Cloudera's Version of the Stack



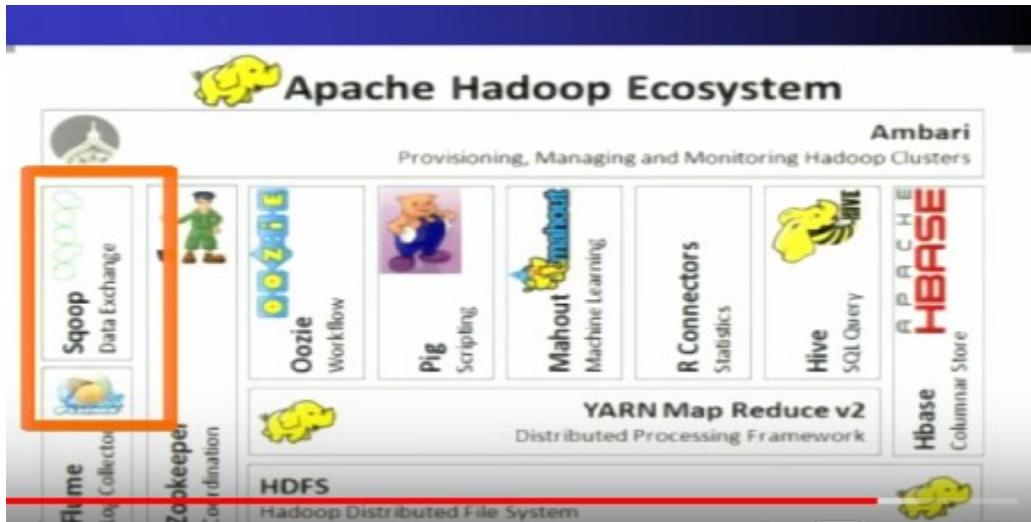
So, the now as, far as the next distribution, of this Hadoop is another distribution is from the Face book and under this particular distribution we see that there will be there is a Hadoop and then zookeeper, HBase has all these different applications, together will give another stack Yahoo's, version of this distribution is called Yahoo stack, which is nothing but a Hadoop stack so Hadoop stack also contains Hadoop and this HBase and all these components, which are shown over here. Link Dell, link dell,

distribution is also called its version of stack. The most important version most important distribution is called the Cloudera's distribution.

Refer Slide Time :(42:55)

Hadoop Ecosystem Major Components

Refer Slide Time :(42:58)



Now let us see the Hadoop ecosystem and its major components let us start with a with a Sqoop.

Refer Slide Time :(43:02)

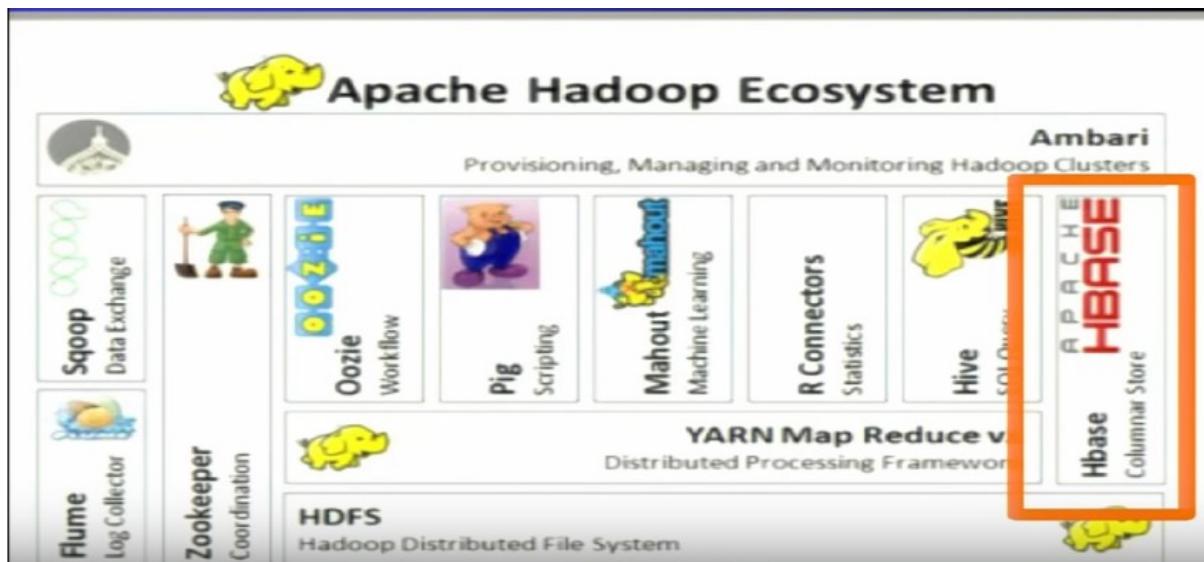
Apache Sqoop

- Tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases



so, you see that a Sqoop is application of the Hadoop ecosystem so patch a sqoop is full form is basically the SQL on Hadoop so, you see that the SQL is basically the database and this entire database is now pulled into the Hadoop system hence it is called Sqoop that is SQL on the Hadoop it is so it is the tool and it is the application for efficiently transporting bulk data between the Apache Hadoop and the SQL data store.

Refer Slide Time :(43:50)



Now, the next application is we are going to touch upon is Apache HBase.

Refer Slide Time :(44:00)

HBASE

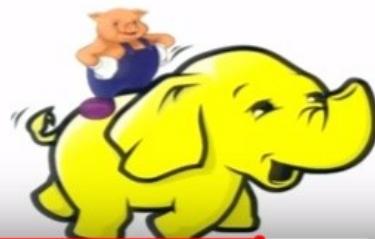
- Hbase is a key component of the Hadoop stack, as its design caters to applications that require really fast random access to significant data set.
- Column-oriented database management system
- Key-value store
- Based on Google Big Table
- Can hold extremely large data
- Dynamic data model
- Not a Relational DBMS

So, HBASE is a key component of Hadoop stack and it and its design data to the application that require really fast random access to the significant data set. So, it is HBase is nothing but a column oriented, distributed, database management system, which is based on key value store, the design of HBase is based on the original Google's Big Table and it can hold extremely large data, data set for storage and writable purposes so, it is a, it is now based on the dynamic data model and it is not a relational DBMS, hence its it is are NoSQL, data model the next application is called the pig.

Refer Slide Time :(44:55)

PIG

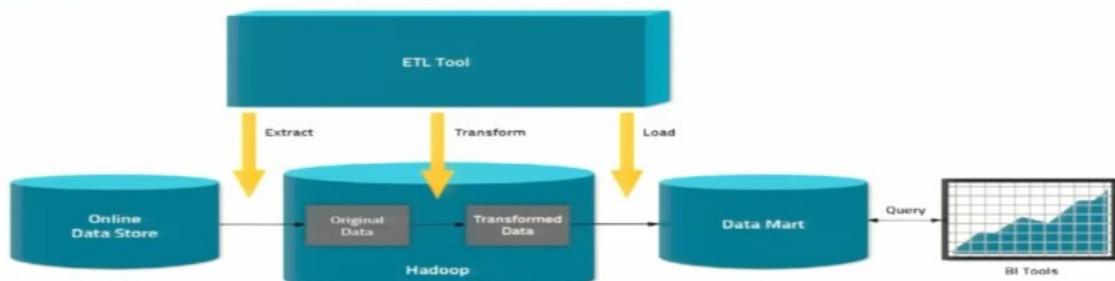
- High level programming on top of Hadoop MapReduce
- The language: Pig Latin ✓
- Data analysis problems as data flows
- Originally developed at Yahoo 2006



which is a scripting language on top of Hadoop Map Reduce. So instead of going to the complication of a complex Map Reduce application program, rather simple view of this scripting language is being provided and that language is called a Pig Latin, and this is useful for the data analysis and as the data flow. So, it is based on data, data flow model and it was originally developed at Yahoo in 2006.

Refer Slide Time :(45:34)

PIG for ETL



- A good example of PIG applications is ETL transaction model that describes how a process will extract data from a source, transporting according to the rules set that we specify, and then load it into a data store.
- PIG can ingest data from files, streams, or any other sources using the UDF: a user-defined functions that we can write ourselves.
- When it has all the data it can perform, select, iterate and do kinds of transformations.

And Pig is used for ETL and here, you can see that the, the traditional ETL technologies, ETL stands for extract transform and load. So, out of the different databases it will store and this Pig is used for doing the analysis.

Refer Slide Time :(46:05)

Apache Hive

- Data warehouse software facilitates querying and managing large datasets residing in distributed storage
- SQL-like language!
- Facilitates querying and managing large datasets in HDFS
- Mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL

The next application is hive, which is an SQL query. So, using SQL query or the Map Reduce, this hive will basically perform the, the storage system and the, the analysis in a much easier manner.

Refer Slide Time :(46:26)

Oozie



- Workflow scheduler system to manage Apache Hadoop jobs
- Oozie Coordinator jobs!
- Supports MapReduce, Pig, Apache Hive, and Sqoop, etc.

Another application is called Oozie. And here, the workflow scheduler system is to manage the Hadoop jobs using Oozie. So, you see is another coordinator of jobs and it supports Map Reduce Pig, Hive and is Sqoop.

Refer Slide Time :(46:39)

Zookeeper

- Provides operational services for a Hadoop cluster group services
- Centralized service for: maintaining configuration information naming services
- Providing distributed synchronization and providing group services

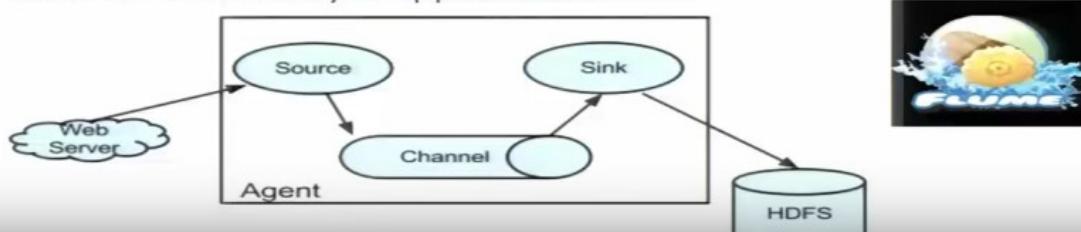


Another coordination service is called a Zookeeper, which provides the coordination service and it will give you a centralized service, for maintaining the configuration and the naming service, it provides the distributed synchronization and the group services.

Refer Slide Time :(47:04)

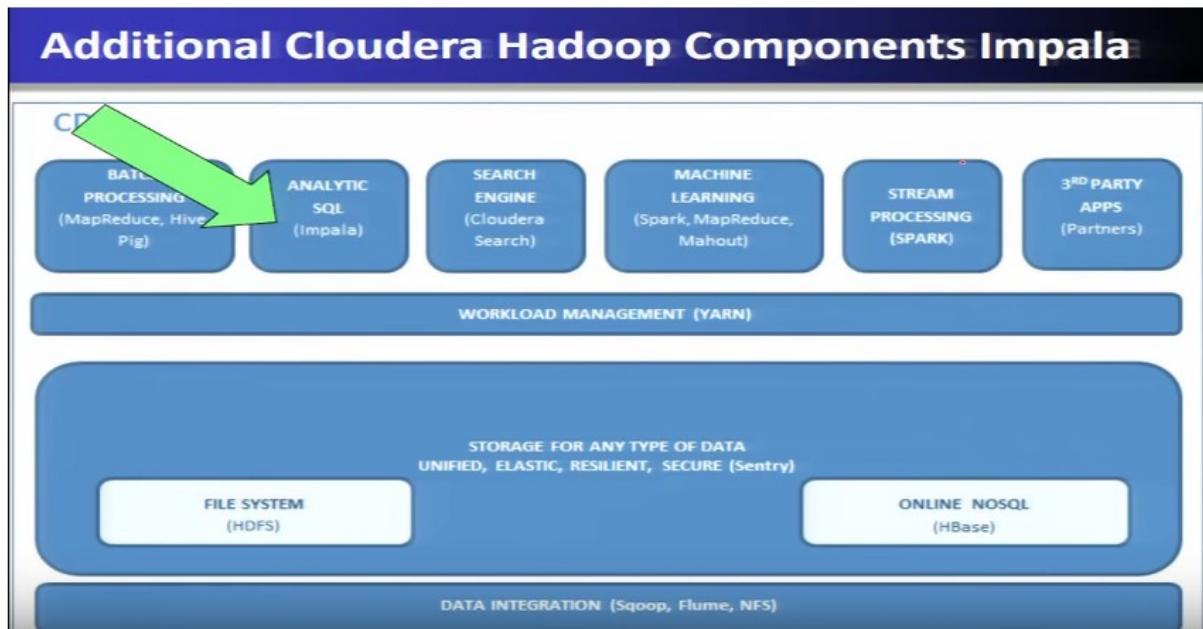
Flume

- Distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data — data ingestion
- It has a simple and very flexible architecture based on streaming data flows. It's quite robust and fail tolerant, and it's really tunable to enhance the reliability mechanisms, fail over, recovery, and all the other mechanisms that keep the cluster safe and reliable.
- It uses simple extensible data model that allows us to apply all kinds of online analytic applications.



Finally another application is called a Flume, which is a distributed reliable available service, for efficiently collecting aggregating moving, a large amount of data into the of the locks into the HDFS system hence, it is used for data injection, please use for data ingestion that is the flume system.

Refer Slide Time :(47:34)



Now, another component is called the Impala and that is nothing but an analytics engine, which is SQL based engine.

Refer Slide Time :(47:48)

Impala

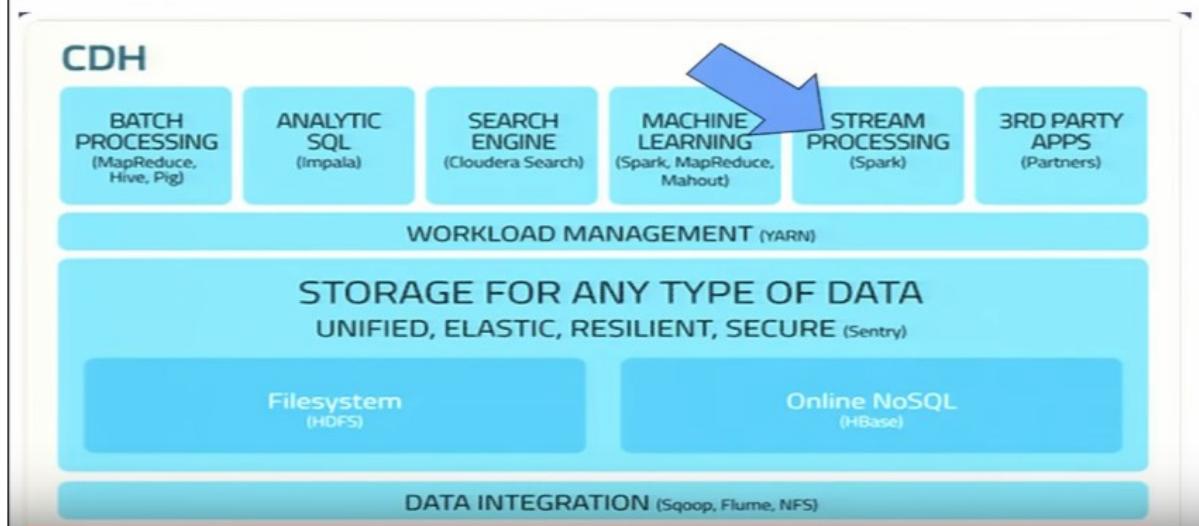
- Cloudera, Impala was designed specifically at Cloudera, and it's a query engine that runs on top of the Apache Hadoop. The project was officially announced at the end of 2012, and became a publicly available, open source distribution.
- Impala brings scalable parallel database technology to Hadoop and allows users to submit low latencies queries to the data that's stored within the HDFS or the Hbase without acquiring a ton of data movement and manipulation.
- Impala is integrated with Hadoop, and it works within the same power system, within the same format metadata, all the security and reliability resources and management workflows.
- It brings that scalable parallel database technology on top of the Hadoop. It actually allows us to submit SQL like queries at much faster speeds with a lot less latency.



So, its query engine runs on top of Apache Hadoop. So, Impala brings a scalable parallel database technology to the Hadoop and allows user to submit low latency queries within a particular system.

Refer Slide Time :(48:06)

Additional Cloudera Hadoop Components Spark The New Paradigm



Now, another component which Hadoop supports is the spark.

Refer Slide Time :(48:14)

Spark

- Apache Spark™ is a fast and general engine for large-scale data processing
- Spark is a scalable data analytics platform that incorporates primitives for in-memory computing and therefore, is allowing to exercise some different performance advantages over traditional Hadoop's cluster storage system approach. And it's implemented and supports something called Scala language, and provides unique environment for data processing.
- Spark is really great for more complex kinds of analytics, and it's great at supporting machine learning libraries.
- It is yet again another open source computing frame work and it was originally developed at MP labs at the University of California Berkeley and it was later donated to the Apache software foundation where it remains today as well.

which is a spark, is a fast general-purpose engine for a large-scale data processing. So, spark is a scalable data analytics platform and it supports the in-memory computation, I enhance its performance is much better why because it supports in-memory computation. So, if Spark supports complex kind of analytics which is called a big data analytics and hence it is of great interest in today's the big data computation spark engine.

Refer Slide Time :(48:44)

Spark Benefits

- In contrast to Hadoop's two stage disk based MapReduce paradigm Multi-stage in-memory primitives provides performance up to 100 times faster for certain applications.
- Allows user programs to load data into a cluster's memory and query it repeatedly
- Spark is really well suited for these machine learning kinds of applications that often times have iterative sorting in memory kinds of computation.
- Spark requires a cluster management and a distributed storage system. So for the cluster management, Spark supports standalone native Spark clusters, or you can actually run Spark on top of a Hadoop yarn, or via patching mesas.
- For distributed storage, Spark can interface with any of the variety of storage systems, including the HDFS, Amazon S3.

So, the spark engine, Spark has the performance 100 times faster, than for applications if spark is used.

Refer Slide Time :(49:07)

Conclusion

- In this lecture, we have discussed the specific components and basic processes of the Hadoop architecture, software stack, and execution environment.

So, conclusion in this lecture, we have discussed the components and the applications of Hadoop ecosystem and also we have covered about its execution environment.

Thank you.

Lecture 04

Hadoop Distributed File System (HDFS)

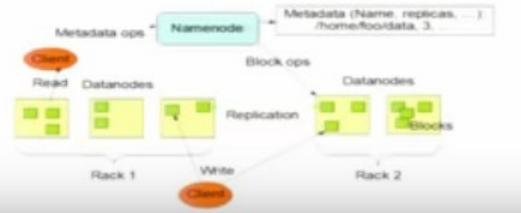
Hadoop distributed file system, HDFS.

Refer Slide Time :(0: 18)

Preface

Content of this Lecture:

- In this lecture, we will discuss design goals of HDFS, the read/write process to HDFS, the main configuration tuning parameters to control HDFS performance and robustness.



Preface content of this lecture; in this lecture, we will cover the goals of HDFS, read/write process in HDFS, configurations tuning parameters to control HDFS performance and robustness.

Refer Slide Time :(0: 37)

Introduction

- Hadoop provides a distributed file system and a framework for the analysis and transformation of very large data sets using the MapReduce paradigm.
- An important characteristic of Hadoop is the partitioning of data and computation across many (thousands) of hosts, and executing application computations in parallel close to their data.
- A Hadoop cluster scales computation capacity, storage capacity and IO bandwidth by simply adding commodity servers. Hadoop clusters at Yahoo! span 25,000 servers, and store 25 petabytes of application data, with the largest cluster being 3500 servers. One hundred other organizations worldwide report using Hadoop.

Introduction, Hadoop provides distributed file system, as a framework for the analysis and performance of very large datasets computations using Map Reduce paradigm. Important characteristics of Hadoop is the partitioning of data and computation across hundreds and thousands of the nodes of a cluster and executing these computations, in parallel, close to their data, Hadoop cluster scales computations capacity, storage capacity and i/o bandwidth by simply adding commodity servers, Hadoop clusters at Yahoo! spans 25,000, servers and store 25 petabytes of application data, with the largest cluster being 3,500, servers 100 other organization worldwide report using Hadoop.

Refer Slide Time :(1: 52)

Introduction

- Hadoop is an **Apache project**; all components are available via the Apache open source license.
- **Yahoo!** has developed and contributed to 80% of the core of Hadoop (**HDFS and MapReduce**).
- **HBase** was originally developed at **Powerset**, now a **department at Microsoft**.
- **Hive** was originated and developed at **Facebook**.
- **Pig, ZooKeeper, and Chukwa** were originated and developed at **Yahoo!**
- **Avro** was originated at **Yahoo!** and is being co-developed with **Cloudera**.

With this introduction let us, see the picture of our overall viewpoint of Hadoop. So, Hadoop is an Apache project; all the components are available via Apache open source license. And Yahoo! has developed and contributed 80% of the core that is, core of Hadoop that is, HDFS and Map Reduce. HBase was originally developed at Powerset, now Microsoft. Hive was originated and developed Facebook. Pig, Zookeeper, Chukwa were originated and developed at Yahoo!

Refer Slide Time :(2: 38)

Hadoop Project Components

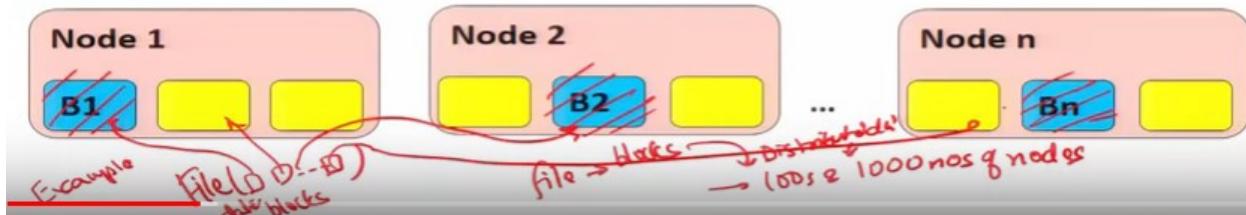
HDFS	Distributed file system
MapReduce	Distributed computation framework
HBase	Column-oriented table service
Pig	Dataflow language and parallel execution framework
Hive	Data warehouse infrastructure
ZooKeeper	Distributed coordination service
Chukwa	System for collecting management data
Avro	Data serialization system

So, some of the Yahoo! Project components and the first and foremost is the HDFS: that is Hadoop distributed file system and the other important component of Hadoop project is the Map Reduce that is distributed parallel computation framework.

Refer Slide Time :(2: 57)

HDFS Design Concepts

- **Scalable distributed filesystem:** So essentially, as you add disks you get scalable performance. And as you add more, you're adding a lot of disks, and that scales out the performance.
- **Distributed data on local disks on several nodes.**
- **Low cost commodity hardware:** A lot of performance out of it because you're aggregating performance.



Now, let us see the Hadoop distributed file system: some of the design concepts and then we will go in more detail of HDFS design. Now, the first important thing is about a scalable, distributed file system: that means, we can add more disk and we can get a scalable performance that is one of the major design, concepts of HDFS, in realizing the scalable distributed file system. Now, as you add more you are adding a lot of disk and, this will automatically scales out the performance, as far as the design, goal of HDFS is concerned. Why this is required is because, if the data, set is big and very large, cannot fit in, to one computer system. So, hundreds and thousands of computer systems are used, to store that file. So, hence the data, of a file is now, divided into the form of a blocks and distributed, onto this large-scale infrastructure. So, that means a distributed data, on the local disk is now, stored on several nodes, this particular method, will insure the low cost commodity hardware usage, to store the amount of information, by distributing it across all these multiple nodes, which comprises of low-cost commodity hardware. The drawback is that, some of the nodes may get failure: that we will see that, it is also included as per the design, goal of HDFS and the low cost commodity hardware, is to be used in this particular manner, a lot of performance out of it, is achieved because, we are aggregating the performance, of hundreds and thousands of such commodity low cost hardware. So, in this particular diagram we see that, we assume and number of nodes let us say node 1, node 2, on so on, node n these nodes are, in the range of hundreds and thousands of the nodes and if a file is given, file is broken into the, to the blocks and these blocks are now, having a data, which will be distributed, on this particular kind of setup. So, in this particular example, we can see that, a file is there and that file is, divided into the blocks, file data is block, in divided into the data, blocks and each block, is stored across different nodes.

So, we can see here, all the blue colored nodes, blue color blocks of these nodes are storing a file data. So, hence the file data is now distributed, onto this local disk in HDFS.

Refer Slide Time :(7: 02)

HDFS Design Goals

- **Hundreds/Thousands of nodes and disks:**
 - It means there's a higher probability of hardware failure. So the design needs to handle node/disk failures.
- **Portability across heterogeneous hardware/software:**
 - Implementation across lots of different kinds of hardware and software.
- **Handle large data sets:**
 - Need to handle terabytes to petabytes.
- **Enable processing with high throughput**

So, hundreds and thousands of the nodes are available and their disk is being used for storing. Now, these comprises of the commodity hardware, so they are prone to the hardware failure and as I told that, that this design. So, they are prone to the hardware failure, so the design needs to handle, the node failures. In this particular case, so HDFS design goal, is to handle, the node failures also. So, another aspect is about the portability across heterogeneous Hardware, why because? There are hundreds and thousands of community hardware machines, they may be having different operating system and the software running, so hence, this heterogeneity also requires, the portability support, in this particular case. That is also one of the HDFS design goals, another important design goal of HDFS is to handle, the large data sets. So, the data sets so the file size, ranging from terabyte to the pet bytes that is huge, file or huge dataset is also now, being able to stored here in HDFS file system so it provides a support of, the handling the large datasets also enable the processing with the high throughput. So that means how this is all ensured the processing with the high throughput? That we will see and it has kept as one of the important design goals of HDFS.

Refer Slide Time :(8: 44)

Techniques to meet HDFS design goals

- **Simplified coherency model:**

- The idea is to write once and then read many times. And that simplifies the number of operations required to commit the write.

- **Data replication:**

- Helps to handle hardware failures.
- Try to spread the data, same piece of data on different nodes.

- Replication

- **Move computation close to the data:**

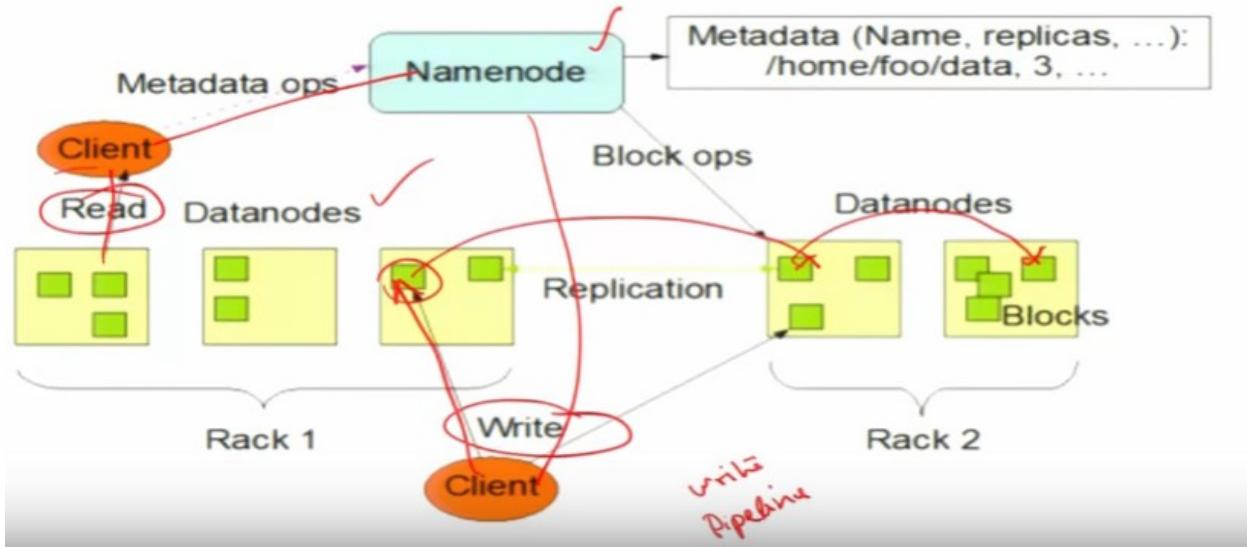
- So you're not moving data around. That improves your performance and throughput.

- **Relax POSIX requirements to increase the throughput.**

Now let us, understand the techniques to meet these design goals, the first of this technique is called a, 'Simplified coherency' model that is, which is nothing but, right once and read many times, this will simplify the number of operations, hence since we are going to write once and read many times. So, most of the design will be based on that coherent model. Another technique which will meet these design goals, which we have seen in the previous slide, is about the data application. Now, since there is a possibility of hardware failures or a failing of these nodes, which are of commodity Hardware therefore, the data blocks, which are store in this particular HDFS file system, is replicated at more than one nodes and hence the data application, is the technique, which will be there to handle the hardware failures. So, by data application means the data blocks, will be spirited, across different nodes and add more than one times, these replications are the same piece of data, is now available on different nodes using replication. So, it is not that only one copy of a data is stored but, the replication factor or replication we will tell how many, different same piece of data, is stored on how many nodes? So, even if that node is fail or Iraq is failed, even then there is a possibility that, the data is available and it will overcome from such failures that is done through the hardware data application. Another important technique which basically ensures, the high performance throughput, is that we moved, to the computation rather computation will be moving close to wherever the data is there hence, we are not moving the data around and this will improve the performance and the throughput of the system. Another technique which is used to meet the HDFS design goal is that, we will relax some of the POSIX requirements, we increase the throughput for example, when I write by the client, then the write operation, will keep on doing the cache, at the clients end. So, this basically, is the relaxation of the POSIX and this will increase the throughput that we will see later on in this particular part of the discussion.

Refer Slide Time :(11: 37)

Basic architecture of HDFS



So, this is the basic architecture of HDFS, which comprises of a single name node and multiple data nodes and which supports the two operations which are write and read, by the clients, so and whenever this, particular client want to do a read operation, a write operation then it has to contact to the name node, try to find out the data, nodes where these particular blocks of a file, need to be written and out of them one, the client will write to the, closest of these data blocks and that particular data block in turn will replicate, through the other data nodes and that particular data node in turn will replicate to the other data node, which is the, which is in the same rack. So, this way, the entire operation: that means this writing on a one block and then replicating at other data nodes, will happen at the same point of time hence, the right is done in, a pipeline mode. This will increase the throughput, of this right operation, similarly whenever a read operation, is required by the client, then this client will contact to the name node, try to find out the blocks or data nodes, where that blocks are stored and out of them, the client will prefer to read, from the data block which is very close to that particular client. In this way the throughput is increased so, therefore a name node and data node together, will which constitutes the architecture of HDFS, is able to support this large-scale data storage and also, ensure the computations at that, place with a high throughput.

Refer Slide Time :(13: 50)

HDFS Architecture: Key Components

- **Single NameNode:** A master server that manages the file system namespace and basically regulates access to these files from clients, and it also keeps track of where the data is on the DataNodes and where the blocks are distributed essentially.
- metadata -
- **Multiple DataNodes:** Typically one per node in a cluster. So you're basically using storage which is local.
- **Basic Functions:**
 - Manage the storage on the DataNode.
 - Read and write requests on the clients
 - Block creation, deletion, and replication is all based on instructions from the NameNode.

So, let us see the HDFS architecture, again and describe its key components. So, a single name node that we have seen, is nothing but, a master server that manages the file system name, namespace and basically regulates, access to these files from the client and also, keep track of where the data is on the data node and where the blocks, are distributed essentially. So, single name node will store the Metadata that means the information about the data, where it is, stored on the data nodes, is maintained by the name space. Now, as far as the data which is actually stored on the data multiple data nodes. So, that means one typically, one per node, in a cluster, is there that is maintained by the name node information and which is used to store the, the data locally. So, the basic functions here, in this key components are that of HDFS is to manage, the storage on the data nodes, where the actual data, is managed or is told and read and write requests, are being initiated by the client, into the HDFS support in the HDFS architecture, similarly for the block creation, deletion and the replication is all based on the instructions, from the name node. So, name node is, basically managing the entire operations of the data, placement data axis, in terms of block creation, deletion and replication.

Refer Slide Time :(15: 44)

Original HDFS Design

- Single NameNode
- Multiple DataNodes
 - Manage storage- blocks of data
 - Serving read/write requests from clients
 - Block creation, deletion, replication

So, we have seen that, in the original HDFS design there is single name node and a multiple data nodes and these data nodes, will manage the storage that is nothing but a blocks of data and these data nodes are serving, for the read and write request from the initiated by the client and also these data nodes, will perform the block creation, deletion and replication.

Refer Slide Time :(16: 07)

HDFS in Hadoop 2.0

- **HDFS Federation:** Basically what we are doing is trying to have multiple data nodes, and multiple name nodes. So that we can increase the name space data. So, if you recall from the first design you have essentially a single node handling all the namespace responsibilities. And you can imagine as you start having thousands of nodes that they'll not scale, and if you have billions of files, you will have scalability issues. So to address that, the federation aspect was brought in. That also brings performance improvements.
- **Benefits:**
 - Increase namespace scalability ✓
 - Performance ✓
 - Isolation ✓

Now let us see, what is new in Hadoop version 2.0? So, HDFS, in aversion Hadoop 2.0 or HDFS 2.0, uses the HDFS Federation that means that, it is not a single namespace but it is a Federation, as that is called, ‘HDFS’ name node Federation. So, so this particular Federation will now, have multiple data nodes and multiple name nodes, are there and this will increase, the reliability of the name node, in this case of Federation. So, it is not one name node, but it is, n number of n in name nodes and this particular method is called the, ‘HDFS Federation’. The benefits is to increase the namespace scalability, earlier there was one name is space now it has, a Federation of name a space so, obviously the scalability is

increased and also, the performance is increased and also, the isolation, performance is increased, why because? Now, the nearest namespace is used to serve, the clients requests and isolation means if let us say a high, requirement high a large amount of resource requirement is there for a particular application, it is not going to affect, the entire a single namespace, why because? It is a Federation of name is space other, applications is not going to be affected by a very high, requirement for a particular application that is called, 'Isolation'.

Refer Slide Time :(18: 00)

HDFS in Hadoop 2

- How its done
 - Multiple Namenode servers ✓
 - Multiple namespaces ✓
 - Data is now stored in Block pools
-
- So there is a pool associated with each namenode or namespace.
 - And these pools are essentially spread out over all the data nodes.

Now, in HDFS version two, are in Hadoop version two how, this all is done let us go and discuss. Now, here as we have mentioned that, instead of one name node here now we have, multiple name mode servers and they are managing, the namespace hence they becomes a multiple namespace and the data, is now stored in the form of a block pools. So, now this block pool is also, going to be managed across these data nodes, on the nodes of the cluster machines. So, it is not only one node, but several nodes are involved and they will be storing the block pools. So, there is a pool associated with each name node or a namespace and these pools are essentially, spread out over all the data nodes.

Refer Slide Time :(19: 01)

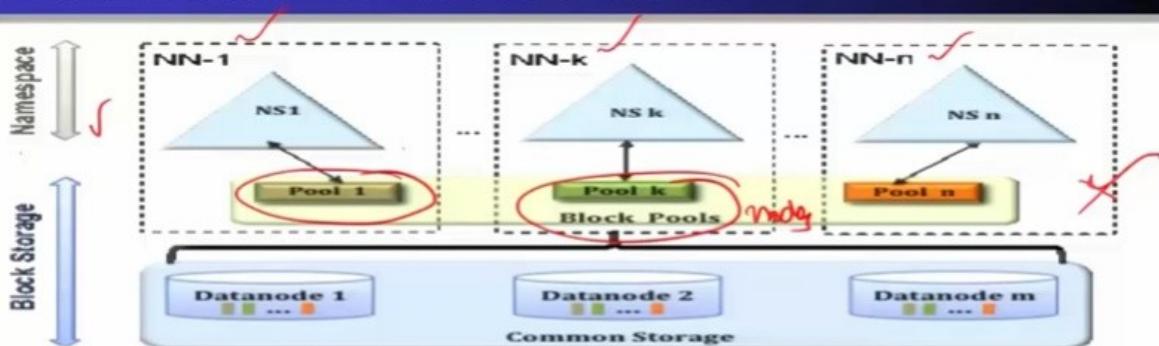
HDFS in Hadoop 2

- High Availability-
Redundant NameNodes
- Heterogeneous Storage
and Archival Storage
 - ARCHIVE, DISK, SSD, RAM_DISK

That we will see, in the further picture.

Refer Slide Time :(19: 04)

Federation: Block Pools ✓



- So, if you remember the original design you have one name space and a bunch of data nodes. So, the structure looks similar.
- You have a bunch of NameNodes, instead of one NameNode. And each of those NameNodes is essentially right into these pools, but the pools are spread out over the data nodes just like before. This is where the data is spread out. You can gloss over the different data nodes. So, the block pool is essentially the main thing that's different.

So, here you can see in this particular diagram that the namespace it has multiple namespace, namespace one, namespace 2 and so on up to namespace n. Let us assume that it has, multiple namespaces and each name space, is having a block pool. So, these are called, 'Block Pools' and these block pools are stored on the nodes. So, they are on different nodes, just like a cluster machine so, each block pool is stored on a

different node. So, different nodes are there and this is going to manage, the multiple namespace and this is called the, 'Federation' of the block pools. So, hence now it is not a single point of failure, even if one or more name node, namespace or a name node fails, it is not going to affect, anything and it also increases the, the performance reliability and throughput and also, performs also provides you the isolation. So, if you remember the original design you have only one namespace and a bunch of data nodes, so the structure looks, like similar but, internally it is managed as the Federation. So, you have a bunch of named nodes now, instead of one named node and each of these named nodes is essentially write to these pools. But, the pools are spread out over the data nodes just like before, this is where the data is spread out and you can grass over the different data nodes. So, the block pool is essentially the main thing that's different in Hadoop, version or HDFS version two.

Refer Slide Time :(20:45)

HDFS Performance Measures

- Determine the number of blocks for a given file size,
- Key HDFS and system components that are affected by the block size. – $64\text{ MB to }128\text{ MB}$
- An impact of using a lot of small files on HDFS and system

↑ Replication - 3
↑ → ↑
↑

So, HDFS performance measures, if we see that, here we see that, determine the number of blocks, for a given file, for a given file size. And the, the key HDFS and the system components are affected by the block size and impact of ,using a lot of small files on, HDFS and HDFS system. So, these are some of the performance, measures that we are going to, tune the parameters and measure the performance. Let us summarize it again, all these and different, tuning parameter for performance. And so, basically the number of, how many number of blocks for a given file size? And this is required, to be known, in so basically, we will see there is a performance measure or, or basically there is a tradeoff, in the number of blocks to be replicated. The another key component is, about the size of the block. So, here the block size, which varies from 64 MB to 128 MB. So, if the block size is 64 then, what is the performance and if we increase the block size, then what will be the performance, similarly the number of blocks, that means, how many this is the replication, if the replication factor is three that means, every block is replicated on three different nodes, for a given file size. So, if the replication is 1, then obviously we are, saving lot of space. But, performance we are going to sacrifice, so there is a trade-off between this. And another important parameter, for HDFS performance is about, the number of small files, which are there on, the

HDFS. So, if there are lot of small files, which are there in HDFS, the performance goes down, we will see how and how this particular problem is to be overcome, in the further slides.

Refer Slide Time :(23:08)

Recall: HDFS Architecture

- Distributed data on local disks on several nodes



So, let us see that, recall again the HDFS architecture, where the data is distributed, on the local disk, on several nodes. So here, in this particular picture, we have shown, several nodes where data is divided and that is called, 'Distributed data on different local disks', it is stored.

Refer Slide Time :(23:31)

HDFS Block Size

- Default block size is 64 megabytes.
- Good for large files!
- So a 10GB file will be broken into: $10 \times 1024 / 64 = 160$ blocks



Now, the data is stored in the terms of block and the block size is going to matter much, in the performance, the default block size 64MB (megabytes). And this is good for a large file. So, if there is a file size of 10 GB, then this particular file is broken into, 160 blocks of size 64 megabytes.

$$10 \times \left(\frac{1024}{64} \right) = 160$$

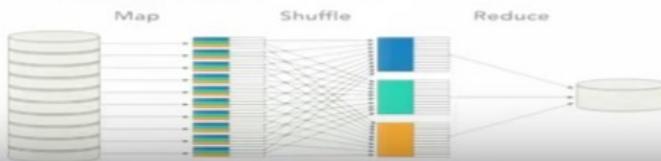
So, 160 blocks we have to just store, in a distributed manner, on several nodes and therefore, this particular block size is going to matter much. So, if we increase the number of or the size of the block, then obviously, it will be less than 160 blocks. So, if the number of blocks are more hence, the parallel operations is more possible and that we are going to see about, what is the, what is the effect of keeping the small block size, of 64 or a more than 64.

Refer Slide Time :(24:34)

Importance of No. of Blocks in a file

NameNode memory usage: Every block that you create basically every file could be a lot of blocks as we saw in the previous case, 160 blocks. And if you have millions of files that's millions of objects essentially. And for each object, it uses a bit of memory on the NameNode, so that is a direct effect of the number of blocks. But if you have replication, then you have 3 times the number of blocks.

Number of map tasks: Number of maps typically depends on the number of blocks being processed.



So, the importance of the number of blocks in a file. So, if the number of blocks are more, than the amount of memory which is used in the name node, will be more in that case. So, for example, here every block that, you create basically every file, could be a lot of blocks we saw in the previous case, 160 blocks. and if you have a million of files and this, that millions of objects, essentially is required to basically store that, amount of space in the name node to manage it and it becomes, several times bigger. If let us say, the number of blocks are more. And the files are more. So, we will see this kind of importance, of the number of blocks, so it is going to affect, into the size, into the name node. And it measures, how much memory is going to be used, in the name node to manage that, many number of blocks of a file. Now, the number of map tasks, also is going to matter much, for example, if the file is divided into 160 blocks, so at least 160 different, map functions are required to be executed, to cover

entire data set operations or computations. So hence, if the number of blocks are more, not only it is going to take more space, in the name node, but also more number of map, functions also required to be executed.

Refer Slide Time :(26:16)

Large No. of small files: Impact on Name node

- **Memory usage:** Typically, the usage is around 150 bytes per object. Now, if you have a billion objects, that's going to be like 300GB of memory.
- **Network load:** Number of checks with datanodes proportional to number of blocks

Hence, there has to be a trade-off. Similarly, if there is a large number of a small files: this will impact on the name node, why because a lot of memory is required, to store the information of this number of small files. Hence, the network load is going to increase, in this particular case.

Refer Slide Time :(26:37)

Large No. of small files: Performance Impact

- **Number of map tasks:** Suppose we have 10GB of data to process and you have them all in lots of 32k file sizes? Then we will end up with 327680 map tasks.
- Huge list of tasks that are queued.
- The other impact of this is the map tasks, each time they spin up and spin down, there's a latency involved with that because you are starting up Java processes and stopping them.
- Inefficient disk I/O with small sizes

So, if there is a large number of a small file: there is a performance issue or a problem of a performance. So, suppose you have 10GB of data, to process and you have all in a lots of 32KB of a file size? Then we are end up with, so many number of map tasks. So, huge list of tasks are now queued up and the performance will go down, why because, when they spin up and spin down ,there is a latency involved and because, you are starting up the Java and stopping them and also, it is inefficient due to the disk i/o, with the small sizes.

Refer Slide Time :(27:19)

HDFS optimized for large files

- Lots of small files is bad!
- **Solution:**
 - Merge/Concatenate files ✓
 - Sequence files ✓
 - HBase, HIVE configuration ✓
 - CombineFileInputFormat ✓

So, HDFS is therefore optimized for a large file size. So, lot of small files is bad and the solution, to this particular problem is to merge and concatenate different files are, there is a operation which is called,

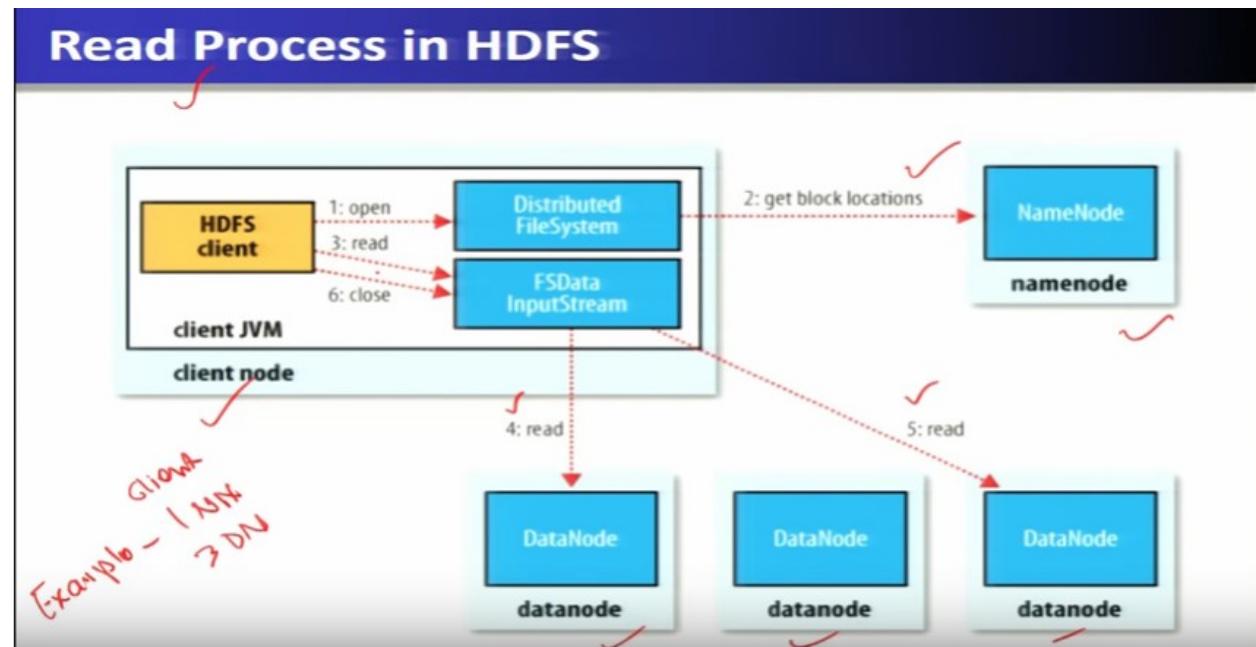
'Sequence Files', several files are Mouse together, in a sequence and that is called a, 'Sequence File'. And which is treated as one big file instead of keeping, many number of small files, another solution, to the small number of lots of small number of files is, using the HBase and hive configuration, for this particular large number of small files. They will be used to optimize this particular issue. And also there is also, another solution is to combine the input file, input format, file input format.

Refer Slide Time :(28:20)

Read/Write Processes in HDFS

Now, let us see, in more detail about read and write processes, which is there in HDFS, how it is being supported.

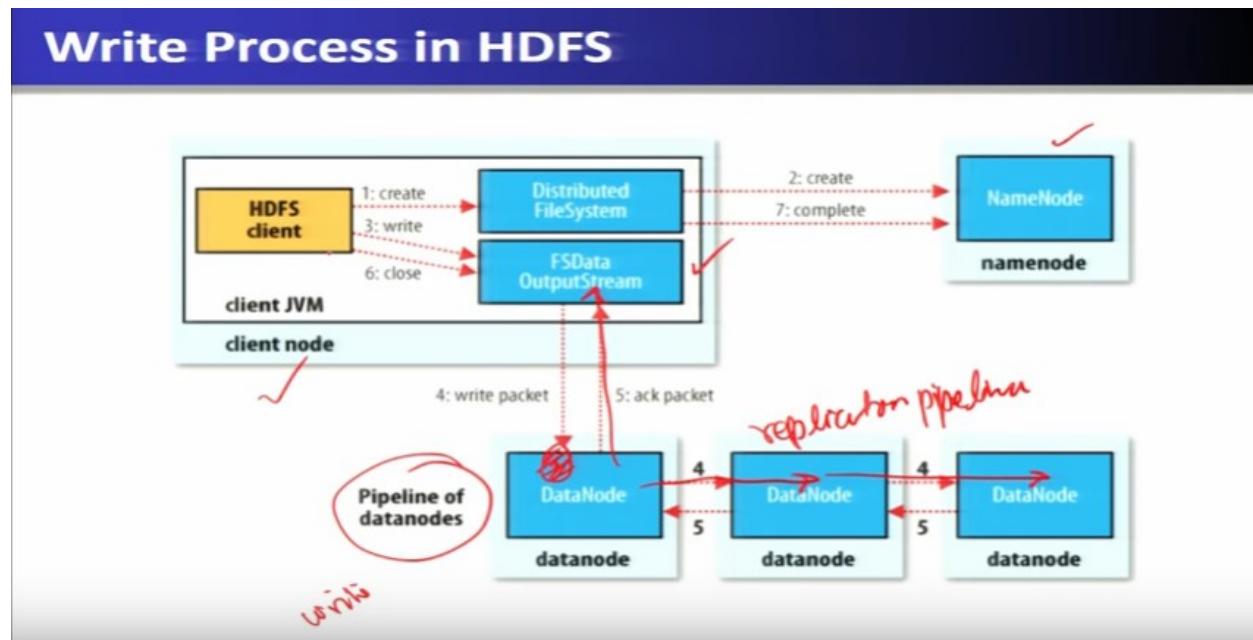
Refer Slide Time :(28:26)



Now, the read process in HDFS, if we see that, first of all we have to identify that there is a name node. And this is the client. And there are several data nodes, in this example, we are having one name node and

there are three data nodes and there is a client, which will perform the read operation. So, the HDFS client, will then request to read a particular file, this is the read operation and this particular request will go to the name node, to know the, the blocks where that, read operation is to be executed and the data is to be given back to the to the client. So, it will send the request to the name node and then, name node will ,give back this information, back to this particular client end of HDFS and from there, it will have two options, whether to read from the block number four or to read from the block number five. Then it will try to read the, the one which is the closest one and this particular data is given back to the client.

Refer Slide Time :(29:56)



Now, then let us see the write operation, which is initiated again by the client. So, whenever there is a client wants to do a write operation and this particular write operation is now, going to be requesting, the name node to find out the, the data node which can, be used to store, the, the clients data. And after getting this information back, this particular right operation is being performed on this particular data node, which is, which is the closest one and that, particular data node has to, do this replication. If let us say, replication factor is 3 then, it will do this in the form of a pipeline. So, the client will write down or will write the data, on a particular data node and that data node, in turn will carry out, the pipeline for the replication, this is called a, 'Replication Pipeline'. And once the replication is over, then it will send the acknowledgment and the right operation is completed, in this particular process.

Refer Slide Time :(31:18)

HDFS Tuning Parameters

Now, let us see in more detail about, HDFS tuning parameters.

Refer Slide Time :(31:24)

Overview

- Tuning parameters
- Specifically DFS Block size
- NameNode, DataNode system/dfs parameters.

So, HDFS tuning parameters, we are going to see, especially the DFS block size, from that viewpoint and also we will see the name node, data node and all these different tuning parameters.

Refer Slide Time :(31:37)

HDFS XML configuration files

- Tuning environment typically in HDFS XML configuration files, for example, in the hdfs-site.xml.
- This is more for system administrators of Hadoop clusters, but it's good to know what changes affect impact the performance, and especially if you're trying things out on your own there some important parameters to keep in mind.
- Commercial vendors have GUI based management console

Now, as far as tuning parameter is concerned in HDFS, there is a file XML configuration file. And for example, hdfs-site.xml file is there, where this particular environment or configuration parameter can be set. In some of the cases like, Cloudera supports automatic GUI for these configurations or tuning parameters of HDFS that is, through the management console.

Refer Slide Time :(32:13)

HDFS Block Size

- Recall: impacts how much NameNode memory is used, number of map tasks that are showing up, and also have impacts on performance.
- Default 64 megabytes: Typically bumped up to 128 megabytes and can be changed based on workloads.
- The parameter that changes dfs.blocksize or dfs.block.size.

Let us see, what are the, which are most important, which need to be decided for performance from, performance perspective. Now here, HDFS block size recall that, impacts how much name node memory is used, the number of map tasks that are showing up and also have the impacts on the performance. So, the by default the block size is 64 megabytes. And typically, it can go up to 128 megabytes and it can be changed based on the workloads. So, if let us say that, we want to have a better performance and the size,

file size is too big, too large, then obviously more than 64 megabytes is good enough, so that so, so the parameter that this, make this particular changes is known as, DFS block size or dfs.block.size, where we have to mention about, the, the, the, the block size, by default it is 64 but we can increase up to, 128 megabytes. So if the, if the block size is more obviously, the number of blocks, will be, will be less and if it is less than, the amount of space which is required, to store in the namespace memory, will be less and also, if it is less and also the number of map tasks, which will be required to execute also, will be less. And so, basically there is a trade-off, where the performance is required, so we have to set, this block size accordingly and application.

Refer Slide Time :(34:00)

HDFS Replication

- Default replication is 3.
- Parameter: dfs.replication ✓
- Tradeoffs:
 - Lower it to reduce replication cost
 - Less robust ✓
 - Higher replication can make data local to more workers
 - Lower replication → More space

So, another parameter is called the, ‘HDFS’, application by default that application is 3 and this parameter is set in a dfs.replication, configuration file. And there is a trade-off, that means, if we lower, it to reduce the replication cost, that means, if the replication factor is not 3, if it is less, than the replication cost will be less. But, the trade-off is that, it will be less robust, robust in the sense, if some of the nodes are filled and there is only one replication, there is no replicas available of that node, so that particular data will not be available. So hence, it will be less robust. And also, the it will lose the performance, for example, if it is replicated then, it will be able to serve that, particular data block, from the closest possible, data block to the client. So, higher application can make data local to the more workers, lower replication means, and more space.

Refer Slide Time :(35:08)

Lot of other parameters

- Various tunables for datanode, namenode.
- **Examples:**
 - Dfs.datanode.handler.count (10): Sets the number of server threads on each datanode
 - Dfs.namenode.fs-limits.max-blocks-per-file: Maximum number of blocks per file.
- **Full List:**
 - <http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml>

Lot of other parameters are there, which can be set. But, these two parameters which we have covered. And that is block size and application factor are the two most important, tunable parameters. So, the other such parameters are available, for example, DFS data node dot handler count is 10, that means, the number of the server threats, on each data nodes that is, maximum up to 10 and this is going to be a factor of this performance, of that data node operations. Similarly, there is another parameter which is called, ‘Name Node’. Offense limits that is the maximum blocks per file, that is maximum number of blocks per file is also set, as per, this one.

Refer Slide Time :(36:01)

Common Failures

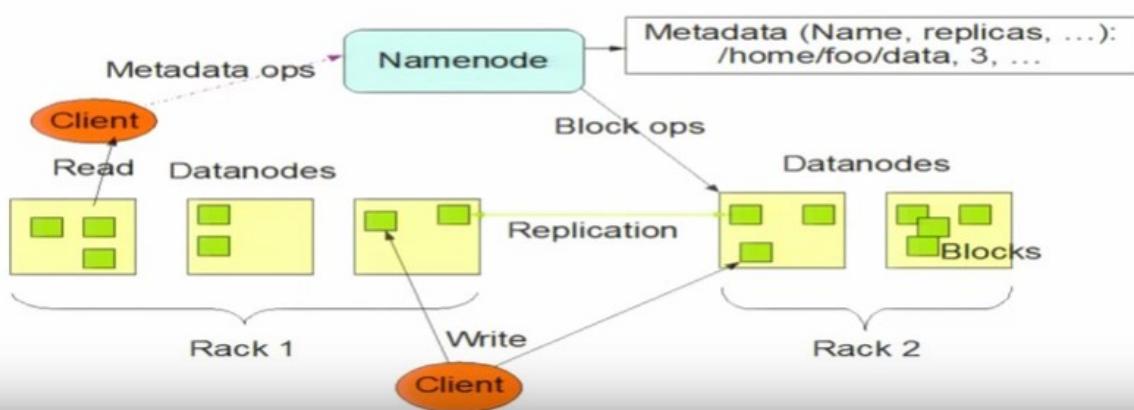
- **DataNode Failures:** Server can fail, disk can crash, data corruption.
- **Network Failures:** Sometimes there's data corruption because of network issues or disk issue. So, all of that could lead to a failure in the DataNode aspect of HDFS. You could have network failures. So, you could have a network go down between a particular and the name node that can affect a lot of data nodes at the same time.
- **NameNode Failures:** Could have name node failures, disk failure on the name node itself or the name node itself could corrupt this process.

So, let us see the, HDFS performance and its robustness. So, the common failures is a data node failure and the server can fail, disk and crash and the data also can become corrupt, in that case, the, the replicas is will be able to overcome, from this particular failures. another failure is called, ‘Network Failure’, sometimes there is a corruption of network or a disk issues. So, it could lead to the failure, of the data node in HDFS. So, when a network go down, then If, if let us say, it is replicas, replicas are across the, the rack then, it can be able to serve, from the other place. Similarly, name node if it is filled and it could be named node failure, disk failure, on the name node, on the name itself, it could corrupt this particular process. So, the Federation is there to, overcome from this name node failures.

Refer Slide Time :(37:09)

HDFS Robustness

- NameNode receives heartbeat and block reports from DataNodes



So, HDFS robustness, we have so far discussed. And so therefore the replication, on the data node is done. So, that it is a lack fault tolerant, that means, the replicas are across racks, so that if the one rack is down, it will be able to serve, from the other end. So, Name node receives the heartbeats and block the report from, this one data nodes, so all these is measured and wherever there is data note down, this information is captured or understood and the name node and that particular node is now, not being used by for the client, for the requests.

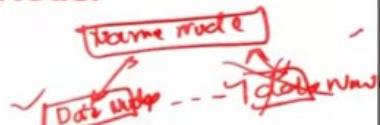
Refer Slide Time :(37:59)

Mitigation of common failures

- Periodic heartbeat: from DataNode to NameNode.

- DataNodes without recent heartbeat:

- Mark the data. And any new I/O that comes up is not going to be sent to that data node. Also remember that NameNode has information on all the replication information for the files on the file system. So, if it knows that a datanode fails which blocks will follow that replication factor.
- Now this replication factor is set for the entire system and also you could set it for particular file when you're writing the file. Either way, the NameNode knows which blocks fall below replication factor. And it will restart the process to re-replicate.



Now, the mitigation of common failures. So, periodic heartbeats from, data node to the name node is done that, we have seen and data nodes, without recent heartbeats is being marked. So mark the data and the new input, output, a new I/O that comes up is, not going to be sent to that node. Data node also remembers that, a name node has the information on all, the replication information, for the files, on the file system. So, if it knows that data, node fails which blocks, will follow that particular application factor. Now, this replication factor is set, for the entire system, so you could, set it for a particular file, when you are writing to a file either way, the name node knows, which block falls below the replication factor and it will restart the process to replicate, to read replicate. So, therefore let us see, this particular diagram that, this is the name node and it keeps on checking, the data nodes and several data nodes. So, these data nodes keeps on sending their, heartbeats at a periodic interval and by that, particular heartbeats, the name node knows that, these particular data nodes are active. If the heartbeats is not, received at the name node, name node, now, understand that this is down and if it is down then, this particular application factor is basically is reduced, for that, particular replicas stored on that data node. So, the name node knows which of that block falls, below the replication factor. And it will restart the process to re replicate. So, that number of, so that replication factor is maintained at all points of time. And that particular data, data node, which is down, which is detected as down, will not be used for, further usage. So, the migration of common failure is, handled by the name node, in this particular manner, with the help of the periodic heartbeats.

Refer Slide Time :(40:28)

Mitigation of common failures

- Checksum computed on file creation.
- Checksums stored in HDFS namespace.
- Used to check retrieved data.
- Re-read from alternate replica

Mitigation of other common failures, such as, checksum computed on a file, we show that, the data or a block is corrupted or a checksum stored, in the HDFS namespace, also tells that it is failed. And used to check the retrieve data and reread the alternative, alternate replicas. So, that means that, whenever there is using checksum, if it is directed that, the data is replica is not or the data, which is accessed is, having an error using, some failure then, alternative replica is consulted up or is being accessed and then, it will be also made the, proper corrections, wherever there is a failure.

Refer Slide Time :(41:30)

Mitigation of common failures

- Multiple copies of central meta data structures.
- Failover to standby NameNode- manual by default.

So, multiple copies of Central meta structure is being maintained to handle with these common failures. And failover, to standby the name node is there and normally it is manually done, by default.

Refer Slide Time :(41:47)

Performance

- Changing blocksize and replication factor can improve performance.
- **Example: Distributed copy**
- Hadoop distcp allows parallel transfer of files.

Now, let us see, the performance issue that, if we change the blocksize and the replica factor, replication factor, how is it going to improve the performance. Let us take an example, of a distributed copy operation, hadoop supports a distributed copy that, allows the parallel transfer of the files.

Refer Slide Time :(42:08)

Replication trade off with respect to robustness

- One performance tradeoff is, actually when you go out to do some of the map reduce jobs, having replicas gives additional locality possibilities, but the big trade off is the robustness. In this case, we said no replicas. Might lose a node or a local disk: can't recover because there is no replication. —
- Similarly, with data corruption, if you get a checksum that's bad, now you can't recover because you don't have a replica.
- Other parameters changes can have similar effects.

Now, there is a trade-off between the replication, trade-off with the respect to the, to the robustness. Before we start, the idea is that, if we reduce the replication factor, then it is going to affect to the robustness. For example, if let us say, it is not replicated, to the other data nodes. And if that data nodes,

containing that data or a block fields, then it is not available at other end. Hence, it is going to affect the robustness. So, replication is so important, that we are going to discuss. So, one performance trade-off is actually, when you go out, to do some of the Map Reduce jobs, having replicas gives additional locality possibility. But, the big trade-off is the robustness, in this case we said, no replica, might lose a node or, a local discount recover because, there is no replica. Hence, if replication factor is, is not immutable, so if the replication so, no replica is available, if no replica is available, then obviously it is lead to a failure. And hence, there is no hence, it is not robust. Similarly, with the data corruption and if we get, the checks that is bad and we cannot recover, why because, we do not have any replicas and other parameters, changes have similar effects. So, basically there is a trade-off between, the replica and the robustness.

Refer Slide Time :(43:51)

Conclusion

- In this lecture, we have discussed design goals of HDFS, the read/write process to HDFS, the main configuration tuning parameters to control HDFS performance and robustness.

So in conclusion, in this lecture, we have discussed the HDFS. HDFS version two and operation that is read and write which is supported in HDFS, we have also seen, the main configuration, we have also seen the performance, parameters and the tuning parameters, with respect to the block size and the replication factor, to ensure about the HDFS performance and its robustness trade-off. Thank you.

Lecture 05

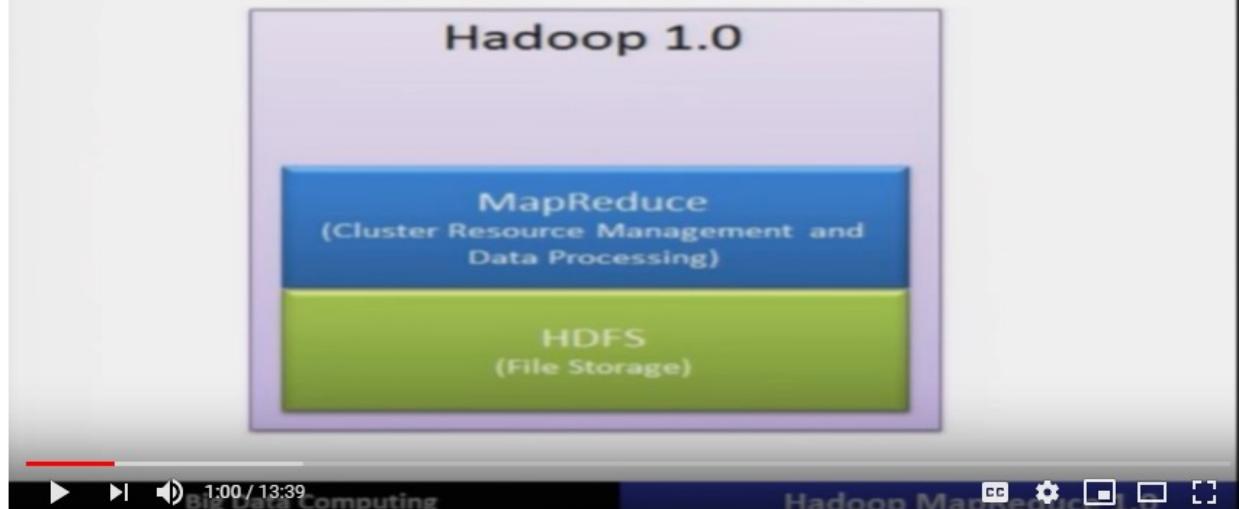
Hadoop MapReduce 1.0

Hadoop MapReduce 1.0 version

Refer slide time :(0:18)

What is Map Reduce

- MapReduce is the execution engine of Hadoop.



So, MapReduce is an execution engine of Hadoop and we are going to briefly describe Hadoop 1.0, its components, that is MapReduce which runs on HDFS. So, MapReduce is the programming paradigm, of the Hadoop system, for big data computing and it also performs, in MapReduce version 1.0 the resource management and the data processing, aspects. Also which runs over HDFS 1.0. So, we are going to cover this Hadoop or a MapReduce 1.0 version, in a brief.

Refer slide time :(01:04)

Map Reduce Components

- The Job Tracker
- Task Tracker

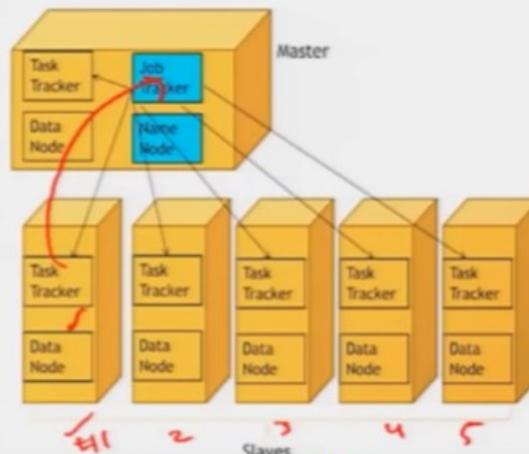


So, MapReduce has two different major components, one is called the,' Job Tracker,' the other one is called the,' Task Tracker'.

Refer slide time :(01:15)

The Task Tracker

- Task tracker is the MapReduce component on the slave machine as there are multiple slave machines.
- Many task trackers are available in a cluster its duty is to perform computation given by job tracker on the data available on the slave machine.
- The task tracker will communicate the progress and report the results to the job tracker.
- The master node contains the job tracker and name node whereas all slaves contain the task tracker and data node.

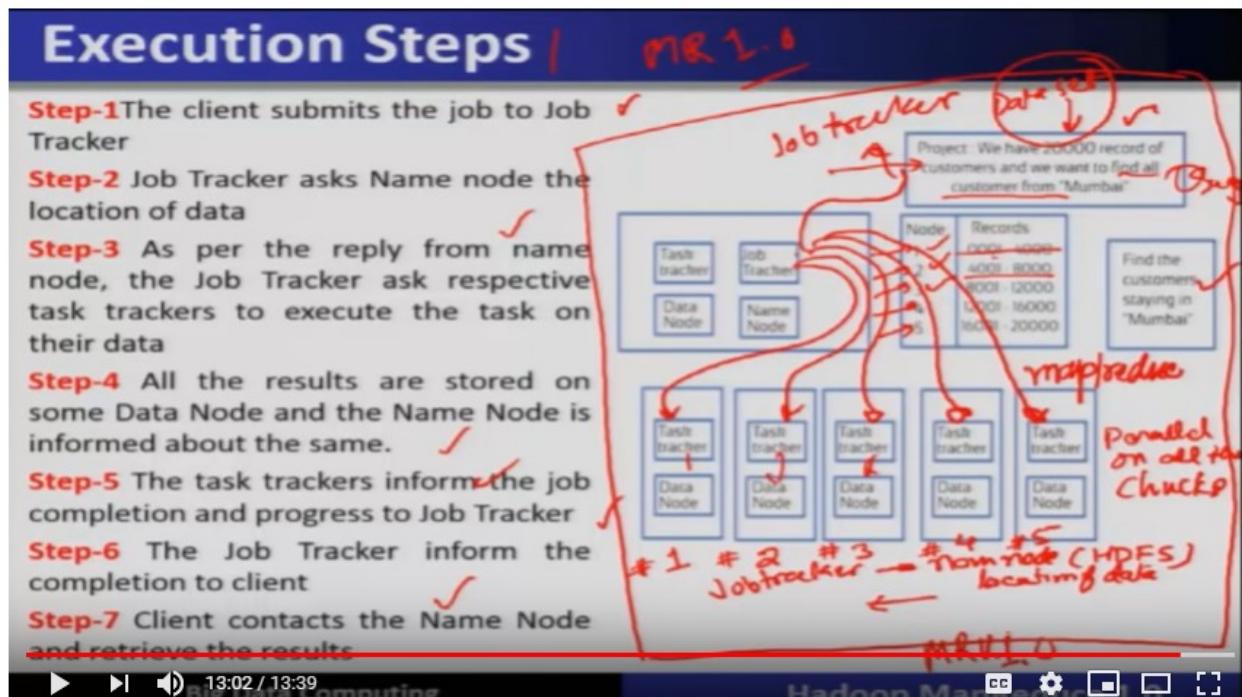


And the scenario of the job tracker is, you can see here, in this particular diagram. So, this is the job tracker, which is a part of MapReduce version one. Now this particular job tracker runs, on the master or this is the master node. So, this master node and this name node, is a part of HDFS, HDFS 1.0 version. So, both of them may resides on the same machine or may not be in the same machine, but for the sake of simplicity we assume that the name node and the job tracker resides, on the same node which is called a 'Master', over here in this particular scenario, since it is having a master and several slaves. So, hence this is basically a client-server architecture, which is followed in MapReduce, version 1.0 and also in the HDFS version 1.0. So, in this particular diagram, we can see here, the job tracker resides on a particular node which is basically the, the master node. And on the same master node, let us assume that as DFS name node is there. So, we are not going to refer in this part of the discussion why because, we are focusing only on the MapReduce.

So, hence we are not going to discuss this name node part, of HDFS. Now another component, is called the, 'Task Tracker', the task record may resides, on the same node also resides on other different slave nodes. So, the job tracker and task tracker they, they run in the form of a client-server model. So, job tracker is basically, running as a must is a server and the task records is run as its client. So, it's a client-server model let us understand, more into the functionality, of job tracker. So, job tracker as, I have already mentioned, is hosted inside the master and receives the job execution requests, from the client. So, the so, the client or the application, which basically is nothing but a MapReduce, program when it is submitted by the client, then the job tracker has to deal with that particular program execution. So, its main duty is to break, down the receive, its main duties are to break down, the received job, that is a that is the big data computation, specified in the form of MapReduce jobs. And this particular MapReduce, is divided into the smaller chunks and that is the small parts and these small parts that is called, 'Chunks', are allocated with the map function and map and reduced function and this particular partial, contradictions are happening at this particular slave nodes with the help of the, task tracker. And this is

the, the entire unit of execution, of this particular job. So, let us see the more detail of the task tracker. So, the task tracker is the MapReduce component, on the slave machine, as there are multiple slave machines, as we have shown here five of them. So, many task records are available, in the cluster its duties to perform the computations, which are assigned by the by that the job tracker, on the data which is available on the slave machines. So, the task tracker will communicate, the progress and report the result back to the job tracker, the master node contains the job tracker and the name node whereas all the slave nodes contain the, the task tracker and data nodes. So, in this particular way, the job tracker keeps the track, of the map and reduce jobs, which are being allocated at different nodes and which are executing on, the data set which are assigned which are allocated, to these particular nodes, where the data is there the, the map and reduced function or the configuration will be performed. So, it is a computation engine. So, MapReduce is a computation engine, in version 1.0. so, not only it allocates, the MapReduce jobs to different slave nodes, where the data also resides, in the form of a chunks and it will then connect and so, basically not only it assigns but also it tracks, keep track ,of the progress and the resources, which is being allocated. Okay? Okay? Discovery hmm physically in order the execution steps.

Refer slide time :(08:04)



So, we are going to now trace all the execution steps, for the life cycle of a MapReduce job, till the application is submitted, by the client and to the to the MapReduce and it finishes and we are going to trace the, the execution cycle or the execution steps in the MapReduce, version 1.0. so, the first step is, the client submits the job, to the job tracker for example here, we have to find out we have to 20,000 records, of the customer and we want to find out all the customers from Mumbai. So, that means the query is basically to be executed, on these data set and this particular operation to find all the customer, this is the query, which is to be performed, using the MapReduce program which is being submitted. So, this particular request is being submitted to the job tracker and job tracker will ask the name note, about the location of this particular data set. So, the job tracker will consult the name node, which is a part of HDFS, to find out the location, of the data where it is being installed. So, now I say that this 20,000 records are divided like this, there are five different nodes, which stores all of them this is node number one, two, three, four, and five. So, the records first four thousand records stored are installed on this particular node number one and the next 4,000 is stored on the node number two and then next four thousand node number three and furthermore node number 4 and 5 respectively stores the remaining twenty thousand records.

Now this is called the chunks or the splits. The entire, two thousand twenty thousand record is splitted and stored on four different, five different nodes and this information will be given from, this name node back to the job tracker now the job tracker as per the reply by the name node the job tracker ask the respective task tracker to execute the tasks on their data. So, therefore, this particular job tracker, now assigns the, the map function or map and reduce function map function, on the stash tracker to execute, on that data chunk. So, with this particular direction the task tracker will perform this execution, at all places there in parallel. So, this particular execution of MapReduce program is done in parallel, on all the chunks. So, after the computation, of a MapReduce so, all the results are stored on the same data node, so whatever is the result it will be stored on the same data node and the name node is informed, about these particular results. So, the task tracker informs the job tracker, about the completion and the progress of this jobs assigned to those job tracker now the job tracker informs this particular completion, to that particular client and the client contacts the name node and retrieve the result back. So, after completing it this particular job tracker will inform, to the client about the completion of this particular job, that means now the result of the query is now ready, which the client will be able to access, with the help of name node and gets back the result. So, this is the entire execution engine, of which is, there in the form of, a MapReduce version 1.0, that we have briefly explained. Thank you.

Noc19-cs33

Lec 06-Hadoop

MapReduce 2.0

(Part-I)

Hadoop, maps reduce 2.0 versions.

Refer Slide Time :(0:17)

Preface

Content of this Lecture:

- In this lecture, we will discuss the '**MapReduce paradigm'** and its internal working and implementation overview.
- We will also see many examples and different applications of MapReduce being used, and look into how the '**scheduling and fault tolerance**' works inside MapReduce.

Content of this lecture in this lecture; we will discuss MapReduce paradigm and also what is new in MapReduce version 2? We will also look into the internal working and the implementation or we will also see many examples how using MapReduce, we can design different applications and also look into how the scheduling and the fault tolerance is now being supported inside the MapReduce implementation of version 2.

Refer Slide Time :(0: 49)

Introduction

- **MapReduce** is a programming model and an associated implementation for **processing and generating large data sets**.
- Users specify a **map** function that processes a key/value pair to generate a set of intermediate key/value pairs, and a **reduce** function that merges all intermediate values associated with the same intermediate key.
- Many real world tasks are expressible in this model.

Introduction of MapReduce; we have to say that MapReduce is a programming model and associated implementation for processing and generating the large data set. Now in a version 2.0, MapReduce per version 2.0, now we have separated out the programming model and as far as the resource management and scheduling, is done using YARN, which is another component of Hadoop 2.0. So, in Hadoop 2.0 if we see this kind of a stack. So it becomes three different layers. So, the first layer is HDFS 2.0, then, this is Hadoop 2.0 and then comes the YARN and here the MapReduce version 2.0. So, now YARN will, do the functionalities of resource management and scheduling which was a part of MapReduce 1.0 version. So, this will make the simplification, into the MapReduce which is only now, the programming model it will focus on the programming aspect and rest of this particular part that is the resource management and the she ruling will be performed by YARN. So, with this, we can design many new applications on this particular HDFS and YARN, bypassing the MapReduce also. So, we will see, the Hadoop stack or different distributions, which either uses HDFS YARN or HDFS, YARN, MapReduce or top of it lot of applications, are available for big data computations. Now, in this particular framework that is MapReduce version 2 which is, now purely a programming model the users, can specify the map function, that processes a key value pair, to generate a set of intermediate key value pairs, meaning to say that the map function it is accepting the input, in the form of a key value pair. Which is quite trivial why because, any data set we can represent in the form of a key value pair? So, the record is nothing but, a key value pair. So, this will become the input to the map function and output is also, the specified in the key value pair. So, what is the key and value pair? That we will discuss, in more detail in the for the slides but, let us general let us understand let us, us at this point of time that any, set of records or a big data set we can represent easily in the form of a key value pair. It's not a big thing that we are going to discuss. Then the second part is called the, 'Reduce Function'. So, reduced function merges, all the intermediate values associated with the same intermediate key. So, that means the internet values or the output which is generated by the map, will be the input to the, to the reduced function that is the output of map function that is nothing but, in the form of a key value pair, that is in the intermediate form of the result and then it will, now do the combination it will combine, it and give the final output. It will combine and combine by

the key and give the results out. So, the map functions and reduce together, will perform in parallel all the computations which are desired to process the large data set. And so, many real-world applications or the task we can express using this particular model that is using MapReduce jobs. So, this particular model allows, to execute the application in, in parallel at all the machines which are basically running that the, the task that is, the slave machines.

Refer Slide Time :(6: 05)

Contd...

- Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines.
— Parallel execution is automatically done in PR
- The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. — APP, input Dataset (large & Storaged (Cluster))
— 100s of thousands of nodes
- This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.
— MR
- A typical MapReduce computation processes many terabytes of data on thousands of machines. Hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

So, the programs, are written in this style are automatically, are executed in parallel on a large cluster of commodity machines. So, the parallel programming paradigm is automatically being taken care and the users or the programmers, are not going to be bothered about, how the parallelization and how the synchronization, is to be managed. So, the so, parallel execution is automatically done in, the MapReduce framework. Now as far, as the run time is concerned, run time take care of the details of partitioning the data scheduling the programs like equation across the set of machines, handling machine failures, managing the required inter machine communication and this, allows the programmer, without any experience with the parallel and distributed computing to easily utilize the resources of the large distributed system. So, let us explain, this entire notion of a runtime operations which this particular MapReduce allows or provides to the execution environment, to the for the execution of these programs on a large-scale commodity machines. So, as far as MapReduce is concerned now, starting with the data set, the input data set, we assume to be a very large and stored on the clusters, that means may be stored on hundreds and thousands of nodes one data set let us assume that it is stored. So, this will be stored with the help of the partitioning. So, the data set will be partitioned and stored on these nodes. So, let us say that if 100 nodes, are available the entire data set is partitioned into 100 different chunks and stored, in this way, for computation in this scenario. Now, the next task is, to schedule the program's execution, across these, let us say 100 different machines where the entire data set is stored. So, then, it launches the, the programs in the form of a map and reduce, we will execute, in parallel, at all places ,that is let us say

if hundred different nodes are involved, all hundred different chunks, will experience the execution in parallel. So, this is also, to be done automatically by, the MapReduce using, the job tracker and a task tracker, that we have already seen. And in the newer version that is 2.0, this particular scheduling and resource allocation, this is to be done by the YARN, once this particular request comes from, this MapReduce execution, for that particular job, of that data set. Now, there is a issues like, these nodes, are comprises of commodity, Hardware that we have seen in the previous lecture of HDFS. So the nodes may fail, in case of node failures, automatically ,it will, be taken care by doing the scheduling of this alternative replicas and the execution still carries on without any effect on the execution. so it requires, also to manage the inter machine communications, for example ,the intermediate results, when it is available ,then, they are sometimes required to communicate with each other ,some values, some results, this is called, 'shuffle' and 'combined'. So, shuffle and combine requires, intermission communication, that we will see, in more detail in the further slides. Now, this will allow the programmers without any experience, with the parallel and distributed systems, to use it. And this will simplify, the entire programming, paradigm and the programmers has to only write down a simple MapReduce programs. so we will see how the programmers will be able to write the MapReduce program for different application and the execution will be automatically, taken care by, this framework. So a typical, MapReduce computation may vary from terabytes on thousands of the machines. So that we have, already seen ,hundreds of MapReduce programs have been implemented and upwards of 1,000 MapReduce jobs are executed on Google's, cluster, every day. So, the companies like Google and other big companies, which works, in the big data computation they uses this technology, which is the MapReduce programs. That means applications are written in the MapReduce that we are going to see them.

Refer Slide Time :(12:02)

Distributed File System

- Chunk Servers**
 - File is split into contiguous chunks
 - Typically each chunk is 16-64MB
 - Each chunk replicated (usually 2x or 3x)
 - Try to keep replicas in different racks
- Master node ✓**
 - Also known as Name Nodes in HDFS
 - Stores metadata ✓
 - Might be replicated
- Client library for file access**
 - Talks to master to find chunk servers
 - Connects directly to chunkservers to access data

Annotations:

- *blocks / chunks*
- *replication 3x - 3 no def*
- *Rack failure tolerance*

now, we will see, these components, which are used in, the MapReduce program execution and they are the chunk servers, where the file is split into the chunks, of 64 megabytes and these chunks are also replicated ,this is also sometimes called as 'blocks'. We call it as chunks, both are synonymous, to be used here in this part of the discussion. Now, another thing is, the these particular chunks are, replicated sometimes, two times the replication all three times, this is called, 'replication factor'. So, it is replicated, that means ,every chunk is not is stored on, more than one, that is more than one nodes ,for example ,when it is the replication factor is three times, so every chunk, is stored on three different nodes. Sometimes, they are try to be stored on a different rack, to have the rack, failure tolerant. So, that means if, the node on one rack or if one rack fields, so it continues to get the other replicas on a different rack, so that is how, different replications are done to support the, the fault tolerance or the failure tolerance. Now the next component of a distributed file system, which is used here in the MapReduce is called, 'Master Node' also known as the name node in HDFS, which stores the metadata, metadata means the information where the exact, data is stored on which data nodes. So, this information is maintained at the master node by and it is also called as a, 'Name Node' in the terminology of HDFS. Now, another component is called, 'Client Library', for the file axis, which talks to the master, to find out the chunk servers, connects directly to the chunk server, to access the data.

Refer Slide Time :(14: 07)

Motivation for Map Reduce (Why)

- **Large-Scale Data Processing**
 - Want to use 1000s of CPUs
 - But don't want hassle of managing things

- **MapReduce Architecture provides**
 - Automatic parallelization & distribution (Parallel on disk)
 - Fault tolerance ✓
 - I/O scheduling ✓ — optimizations/ performance
 - Monitoring & status updates —

Now, we will see about, this MapReduce and its motivation why we are? So, much enthusiastic, about this more produced programming paradigm and which is very much used in the Big Data computation. So, the motivation for MapReduce is, to support the large-scale data processing. So, if you want, to run the program on thousands of CPUs and then this MapReduce is the only paradigm available and all the companies, which are looking for the scale? A large-scale data processing the MapReduce is the only framework, which is basically doing all this task? And another important motivation is that, this particular paradigm makes the programming very simple and does not require the programmer to go and manage

the intricacies of the menu detail, underneath. Now MapReduce architecture also provides automatic parallelization and the distribution, of data and the configuration of it. So, the parallel computation, on distributed system is all abstracted and the programmer is provided a simple MapReduce paradigm and he does not have, to bother on these details automatically, the parallelization and distributed computation, is being performed another important part is the failures. So, the fault tolerance is also supported in the MapReduce architecture and the programmer does not have to bother about, that automatically it has taken care, provided the programmer gives sufficient, configuration information so that this fault tolerance is taken care of in different applications, another thing is input-output scheduling, is automatically done. So, those lots of optimizations are used to reduce the number of I/O, operations and also improve upon the performance of the execution engines. So, that is all automatically done by the MapReduce architecture. Similarly the monitoring of all the data nodes, that is the task, that is the task records, execution and their status updates is all, done using the client-server interaction in the MapReduce architecture, that is also taken care of.

Refer Slide Time :(17: 05)

What is MapReduce?

- Terms are borrowed from Functional Language (e.g., Lisp)
- **Sum of squares**
 - $(\text{map square } '(1 2 3 4))$
 - Output: $(1 4 9 16)$

Intermediate result

Parallel execution

[processes each record sequentially and independently]
- **(reduce + '(1 4 9 16))**
 - $(+ 16 (+ 9 (+ 4 1)))$
 - Output: 30 ✓ *Sum of squares.*

[processes set of all records in batches]
- Let's consider a sample application: Wordcount
 - You are given a **huge** dataset (e.g., Wikipedia dump or all of Shakespeare's works) and asked to list the count for each of the words in each of the documents therein ✓

Now, let us go in more detail of the MapReduce paradigm. So, let us see, what this MapReduce? Is from programmers perspective, this particular map and reduce terms is borrowed from the functional programming language, with let us say that Lisp, is one such programming language, which has this kind of features, that is MapReduce. So, let us see the functionality, which is supported in the functional programming language of MapReduce? And let us say that, we want to; write a program for calculating the sum of squares of a given list. So, let us assume that the list which we are given, is one, two, three, four is a list of numbers and we want to find out, the squares of these numbers. So, there is a function which is called a, 'Square'. And a map says that this particular square function is to be applied on each and every element of the list. So, let us see the output will be:

(map square '(1 2 3 4))

Output: (1 4 9 16)

[processes each record sequentially and independently]

So, all the squares are computed and output is given again in the form of a list. Now, this particular operation is square may be executed in parallel, at all the elements and this result will be given very efficiently ,it is not that sequentially one by one, all the squares are being computed. So, therefore it will, process them independently instead of these records are not to be carried out in sequentially hence, it's a parallel execution, environment of map function ,of a map function that is, performing the square operation on all the elements, which are being provided in the map function. Similarly to find out the sum this was to calculate the, the squares, now another routine which is required to make the summation, of this intermediate result. So, this result is called, 'Intermediate Result'. So, with this input, we have to perform another operation which is called a, 'Reduce'. Reduce will require an addition operation, the addition on this particular list, which is nothing but, an intermediate result which is calculated by the map function and you can see that, the sum is the binary operator. So, it will start doing the summation, 2 elements at a time and once they are calculated. So, the entire output will be given in the form of whatever is required sum of the squares. So, this is the sum of squares.

(reduce + '(1 4 9 16))

(+ 16 (+ 9 (+ 4 1)))

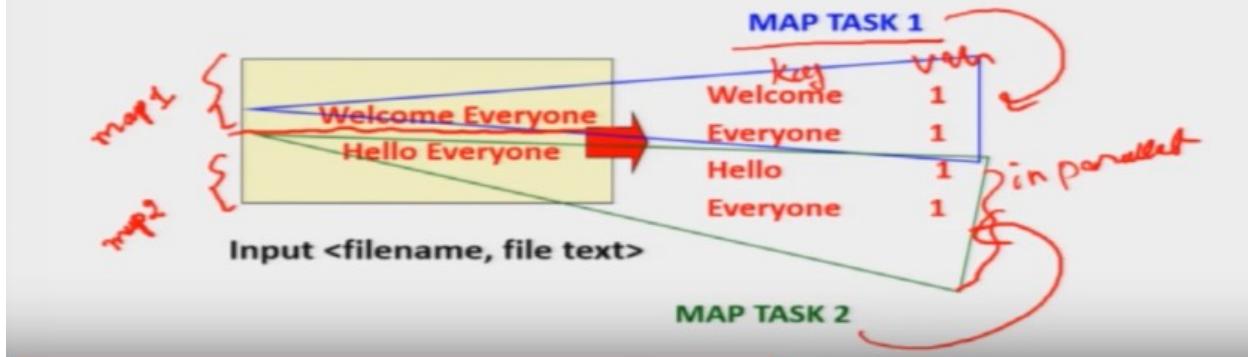
Output: 30

So, given this kind of functionality or a function of a functional programming language, we will see that, how this particular MapReduce? Can be applied here, in the Hadoop scenario. Now, let us see a small application, simple application, here it, is written a sample application of a word count. So, that means, let us say we have a large collection of documents, let us assume it is a Wikipedia dump, of particular Shakespeare's, works it can be any, different important persons related works, we are going to process this huge data set and we are asked to find out, to list the count for each of, the words in this one document and then query will be to find out a particular reference, in the Shakespeare's works about, the characters how many? Which character is more important based on how much referencing? Shakespeare has done for that character.

Refer Slide Time :(21: 31)

Map

- **Parallelly** Process individual records to generate intermediate key/value pairs.

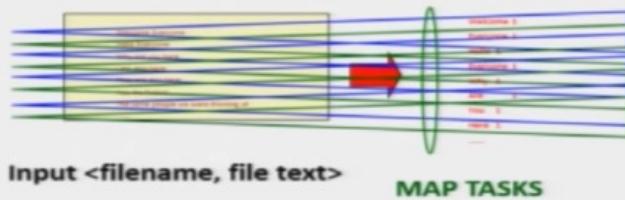


So, let us see how this is all done using MapReduce? So, MapReduce will take huge collection of data set first it will be considered in the form of the key value pairs, for example if it is, the document is let us say welcome everyone, hello everyone. So, here this is the key is about, every word it becomes a key and how many times it is appearing in a particular sentence becomes the value? So, the file is also in a given the input in the form of the filename and the text at the file text so, so this becomes the key, value. So, file name will be the key and these are text, the value and further on when we process using map function, when we apply a map function, on this particular input it, will generate this another intermediate key value. This is called, 'Intermediate Result'. In a form of key values, that means for every word, will become a key, every word will become a key, what will become a key? And the instances, when it is occurring, into a particular line of a, of a file then it becomes the value, for example on in a line number one, welcome, is appearing once. So, it gives one similarly in the line number one everyone is also is available only once. Similarly in the second line when it is being processed. So, hello is the appearing one. So, it will be treated as once and on the second line everyone is appearing again, so this particular key value, which is emitted out of map? Is further used by the reduce function. Now, let us say that, let us see how, in parallel this map function can work, for example these two lines, instead of processed them sequentially let us divide in two, into two different chunks and this chunk, will be given map function one and this another chunk will be given second map function. So, if there are two chunks we are going to process them in parallel. So, this particular task, map task one will execute and map task 2 will also execute, in parallel. So, in parallel if they are executed, so key this key value pair welcome and everyone, will be collected up by the task number 1, similarly the other task that is hello and everyone, will be collected in the map task 2. So, we need to say that, this particular map function allows, the process individual record to generate the intermediate key value pairs, in parallel and that is done automatically.

Refer Slide Time :(24: 46)

Map

- **Parallelly Process a large number** of individual records to generate intermediate key/value pairs.



So, parallel process a large number of individual record, generate the intermediate key value pair, that we have seen if these numbers or the chunks, are increased more than, two then, this is, an example where multiple map functions are simultaneously, at the same time that is in parallel they will, execute and process these records.

Refer Slide Time :(25: 09)

Reduce

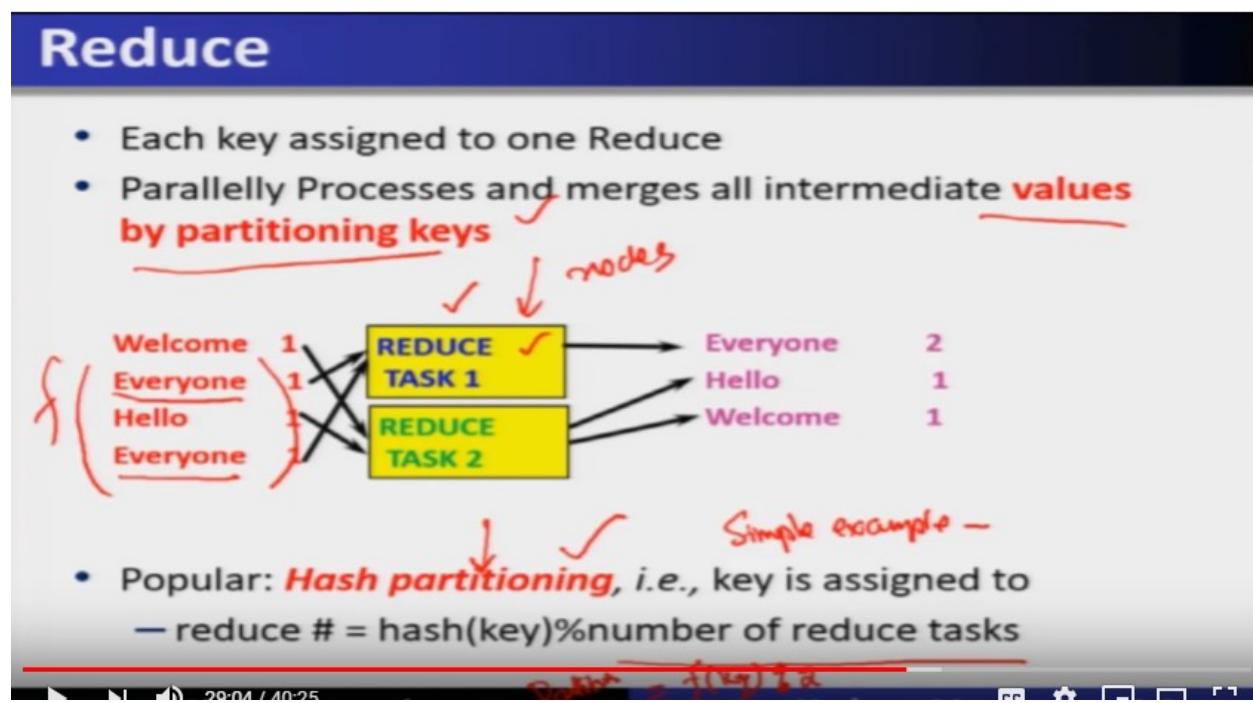
- Reduce processes and merges all intermediate values associated per key (Group by key)
Computing by reduce



Now, we will see the next phase that is called, 'Reduced Operation'. So, after the map the intermediate values, which is being generated, in the form of the key value pair will be used as the input for the

reduced function. So, reduce processes these intermediate values, which is output by the map function and then merges all the intermediate values, associated per key, that means based on the similarity of the key values, they are all combined together that is group by key operation, is being performed and using, in this particular, group by key operation whatever is the reduced function applies, this will be the computation will come, from the reduced function. So, computation on this group by key, will be applied by, the reduce function to get the entire output, for example if this is, the input which is, input to the reduced function and we want to do a word count. So, now what it will do is, it will group by the key when you say group by the key then, everyone is appearing twice. So, with everyone there are two different values or two times, this everyone one will come and this will be added up by the reduced function, similarly hello one is appearing once. So, group by key is, having only this tuple or entity similarly for welcome. So, this particular reduced function, will execute in this particular manner.

Refer Slide Time :(27: 05)



So, each key assigned to one reduce function and here we can say that if there are two different nodes, which runs this reduce task, let us say task 1 and reduce task 2. So, each key is, assigned to one reduce task for example this, particular key will be assigned to the task 1. So, when this second key also second time when everyone comes it should also, assign to that same reduce task, whereas welcome and hello can be assigned to the reduce task 2. Now, parallel process and merges all, the intermediate value by partitioning keys that, we have already told you, but how, these keys are to be partitioned and the simple example is, let us, say that use a, hash partitioning, that means if you have a hashing function f, and when you apply this has have a hashing function, on a particular keyword, it will give a particular value and this value will be mod, how many number of reduced tasks here? Let us, assume that it is, the number of tasks is 2 and modulo, some hash function and on that particular hash function, we have to define a key and this

will generate the, the partitioning. So, partitioning we have shown you through, a simple example, the way partitioning is carried out, that is through the highest partitioning, it can be very complicated also depending upon different application how ,the partitioning is done and how many different worker nodes are reduced nodes are there or reduce task is allocated by the YARN? And based on that this particular partitioning is done and after partitioning then it will merge, the intermediate values and gives, the result back.

Refer Slide Time :(29: 09)

Programming Model

- The computation takes a set of **input key/value pairs**, and produces a set of **output key/value pairs**.
- The user of the MapReduce library expresses the computation as two functions:
 - (i) The Map
 - (ii) The Reduce

Programming model, the computation takes a set of input key value pairs and produces a set of output key value pairs, in this programming model. To do this, the user has to specify using MapReduce library the two functions, the map and reduce function, which we have already explained.

Refer Slide Time :(29: 38)

(i) Map Abstraction

- Map, written by the user, takes an input pair and produces a set of **intermediate key/value pairs**.
- The MapReduce library groups together all intermediate values associated with the same **intermediate key 'I'** and passes them to the **Reduce function**.

Map abstraction is written by the user which will take the input pairs and produces the set of intermediate key value pairs. This function is to be written, based on the applications and the programmer has to, write down as per the application what is to be done in the map function? And the remaining part will take care in the reduced function. So, the reduced library, MapReduce library, groups together all intermediate values associated with the same intermediate key and passes them to the reduced function that was the map abstraction.

Refer Slide Time :(30: 21)

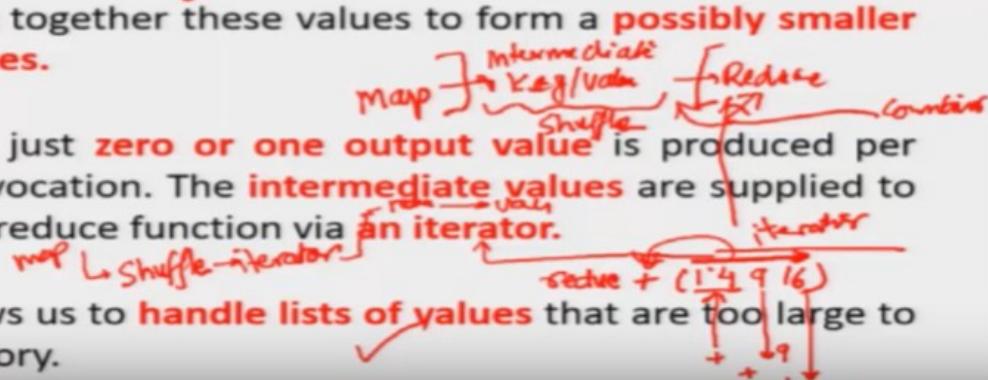
(ii) Reduce Abstraction

- **The Reduce function**, also written by the user, accepts an **intermediate key 'I'** and a set of values for that key.

- It merges together these values to form a **possibly smaller set of values**.

- Typically just **zero or one output value** is produced per Reduce invocation. The **intermediate values** are supplied to the user's reduce function via **an iterator**.

- This allows us to **handle lists of values** that are too large to fit in memory.



In the reduced function, also it is written by the user, which accepts an intermediate key and a set of values for that particular key, it merges together these values, to form a possibly a smaller set of values, typically just 0 or 1 output values produced per reduce invocation, the intermediate values are supplied to the users, reduced function via an iterator. This allows us to; handle the list of values that are too large to fit in the memory. So, that means, if we see the outcome of the map function, it will generate the intermediate key value pair, which will be taken in the reduce phase as the input. So, here, what it does is? This stage which is called a, ‘Shuffle’. It merges together, shuffle and the combiner, here there is a combiner, sometimes a combiner is a part of both the activities map and reduces. So, the combiner. So, first it will shuffle, that is it will arrange, according to the group by keys and this is called a, ‘Shuffle’. And after that it will combine, that is all the same keys, with the values to be combined and is given to the reduce to operate. So, therefore, 0 r1 output values is produced, per reduce invocation the n iterator intermediate values, supplied to the users reduced function, wire and n iterators meaning to say that this particular function, which is called, ‘Iterator’. Can be understood, in more detail like this, for example if you want to do, a summation of the intermediate values which is, basically the sum of squares. Let us say, we want to, do this, particular summation of all these things. So, iterator has, two for example if there are, maybe another example, iterator has to evolve for example if there are yeah, iterator has to evolve or to see the, the elements which are similar, it has to scan and this is called, ‘Iterator’. So, it will scan through all, the elements and perform the operation together and for example one and four, can be treated together. So, iterator here will identify, a binary or two set of elements, first two elements and perform the plus operation and then it will, take the next operation, next element and perform the plus operation and then perform then take the next element 16 and performed on the plus operation to become the overall sum. So, the iterator job is, to scan, through this particular list and provide the elements for the operation which is specified in the reduced function. So, we have seen, that out of the map function the, there is a activity, which is called a, ‘Shuffle’. Which will sort, the elements? So, that all the key value, key values, with the same key group by same key they are to be sorted and given back to the next phase. And in the reduced function, it will use the iterator, to scan, through this particular list of elements, which are already sorted and provided by the shuffle. So, after the map function, lot of internal communication and activities are going on, in the form of shuffle and iterator and then only the reduced function will be, applied and give the values out. So, this allows, these operations allows us, to handle the list of values, that are too large to be fed, in the memory and why because? They are not, to be brought at one place but, they are to be, operated wherever these use our reciting, hence it is, possible that the, the operations are performed, even on a very large data set which cannot fit, into a particular memory.

Refer Slide Time :(35: 54)

Map-Reduce Functions for Word Count

```
map(key, value):
    // key: document name; value: text of document
    for each word w in value:
        emit(w, 1)

reduce(key, values):
    // key: a word; values: an iterator over counts
    result = 0
    for each count v in values:
        result += v
    emit(key, result)
```

Now, we will just see, the structure of the map and reduce function which programmers used to write down and here, we take the example of a word count. Will not go in detail in explanation at this stage, explanation we will see in more detail in the further slides. So, as far as, the map function this is the pseudo code, of a map function it cannot be executed in this form, why because? It has to be programmed as per the specification of that particular language, of MapReduce but, we are going to see the, the pseudo code. So, pseudo code has to specify the map function with the key and value and where key here, in the word count example, is the name of the document and the values are the, the text, of the document which are be retrieved and given in this map function. So, for each word, which is basically getting as the value, in this particular for each word in the value, which is appearing the, map will omit, that particular word and one. Now, the next pseudo code, for reduce takes, the same format as emitted out of the map function. So, that means the world becomes, the key and the value becomes as per the value over here, that format should match then only the output of map can be acceptable that is the, the reduce function. So, here, it is explained that the key, is the word and values is basically the values, when it is grouped by key then, the iterator has to operate, over this set of values. So, let us take this set of values and we perform the reduce operation. So, reduce we will take all these values, out of iterator and does the summation and results, in to that, how many times that key is appearing? That is it. We'll do the summation of all the values but, this has explained .

Refer Slide Time :(38: 23)

Map-Reduce Functions

- **Input:** a set of key/value pairs ✓
- User supplies two functions:
 $\text{map}(k, v) \rightarrow \text{list}(k_1, v_1)$ *intermediate key/va*
 $\text{reduce}(k_1, \text{list}(v_1)) \rightarrow v_2$ ✓
- (k_1, v_1) is an intermediate key/value pair ✓
- **Output** is the set of (k_1, v_2) pairs ✓

The MapReduce paradigm and let us, see some more detail of it and here, we have given the input as a set of key value pairs and the user has to specify, the function map, wherein he has to specify, the what is key and what is value? And after that this map function will emit, intermediate values in the form of a list, which is having a (k_1, v_1) . So, k_1 and v_1 is the intermediate key value pair, and this intermediate key value pair, is given to the reduce by the shuffler. So, once the reducer will get the key and intermediate key and value pair. So, it will get the entire list of values and here, the iterator will operate on it and applying, the reduced function it will emit that particular value. So, the output will be, the (k_1, v_2) values combined. So, this is the abstraction of MapReduce and overall picture how the program, can be written using MapReduce paradigm. Thank you.

Lec 07-
Hadoop MapReduce 2.0
(Part-II)

Let us see, some of the applications, of MapReduce.

Refer Slide Time :(0: 15)

Applications

- Here are a few simple applications of interesting programs that can be easily expressed as **MapReduce computations**.
- **Distributed Grep:** The map function emits a line if it matches a supplied pattern. The reduce function is an identity function that just copies the supplied intermediate data to the output.
- **Count of URL Access Frequency:** The map function processes logs of web page requests and outputs (URL; 1). The reduce function adds together all values for the same URL and emits a (URL; total count) pair.
word count like program (A)(B)(AB)
- **ReverseWeb-Link Graph:** The map function outputs (target; source) pairs for each link to a target URL found in a page named source. The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair: (target: list(source))
map (A) (B) (C) (AC) (ABC)

We are already the programs are available and its, use in the production, environment, by different companies. So, here are the few, simple applications of interesting programs, that can be, that has, already been expressed, as the MapReduce computation. So, the first one is called, 'Distributed Graph'. Here, the map function emits a line, if it matches a supplied pattern, for example, if the document is given and we, we have already given one pattern. So, all the lines in that particular document, where that particular pattern is appearing, will be emitted and reduced function will become an identity function that that just copies, the supplied intermediate values to the output that is called, 'Intermediate Graph' or 'Distributed Graph'. so, the difference between the Graph and this distributed graph is that, here the document can be very big, it cannot fit into one system, memory and therefore a document which is distributed the stored on data nodes, can perform this operation the graph. So, it will filter and extract, only those set of documents where we are interested in that particular pattern, that's called, 'Distributed Graph', has various purposes various applications, the next application is the count of URL, access frequency. Now, to do this, there is a map function, which processes the log of webpage, requests and the output (URL;1). So, that means the map function, what it does is? It inspects the log of web pages and for every URL, it encounters it will emit one as per the map phase. Now, reduce function will combine. So, it will collect the all the URLs, that means group by key and it will do the summation, of how many times that number of ones are there it has to, just do a count. So, it is just like a word count program, an extension of a word

count, which will find out, the URL access frequency that means our URL how many times it is being referred in a particular log file? Now, another example, another application, where it is used this MapReduce program? Is called, Reverse Web Link Graph', for example there are the web pages and web pages are, pointing to each other. Let us say this is a B and C. We want to find out that, for let us say web page see, how many different pages are pointing to it? We are given these, kind of pairs that is a is pointing to C, web page is pointing to web page C and web page be pointing to A and B is pointing to C, out of this we have to, now find out that for a particular web page, how many links are pointing to it? It's called, 'Reverse Web Link Graph'. So, it's called a, 'Reverse Web Link Graph'. So, the map function here, given this as the input, to the map function it will output the target and the source pair. So, for example here the target is C. So, C and the source. So, given AC, in the map it will emit, C and A, for each link in this, particular case similarly for BC, it will emit, C and B similarly for BA, it will emit, A and B, for each link to the target URL found in the, in a page named source. Now, the reduced function then concatenate, the list of all sorts URLs, associated with a given target URL and emits appear, that is called, 'Target and List Source' for example, here C is appearing ,two times and this list will become for C a comma B. So, for C this is, the list is, C comma, A B. So, this is, one and another one is a and B. So, this will be given to the reduced function and reduce will take this, target C and find out this list or the source. So, C, page is being pointed to by A and B, that is being computed here by, the MapReduce and the web page A, is being pointed to by only one, web link that is called A B. So, this way, popularity of the pages can also be calculated if you want to do a sum, you have to just make a count of it. So, it becomes, 2 this becomes 1 and this kind of statistics or this kind of output can be used in computing, the Page Rank.

Refer Slide Time :(6: 31)

Contd...

- **Term-Vector per Host:** A term vector summarizes the most important words that occur in a document or a set of documents as a list of (word; frequency) pairs.
- The map function emits a (hostname; term vector) pair for each input document (where the hostname is extracted from the URL of the document).
- The reduce function is passed all per-document term vectors for a given host. It adds these term vectors together, throwing away infrequent terms, and then emits a final (hostname; term vector) pair

Now, another application is called, ‘Term-Vector per Host’. So, term vector summarizes the most important words that occur in a document or a set of documents as a list of a word and frequency pair. So, the map function omits the hostname and the term vector, appear for each input document, where the hostname is extracted from URL of that document, the reduced function is passed, on the all per document on vector, for a given host it at these term vector together, throwing away the infrequent terms and then omits the final hostname and term vector pair. So, that means, for a given term document we are going to find out the most important or most frequent words and we have to basically omit the hostname and the term vector pair in this particular application.

Refer Slide Time :(7:30)

Contd...

- **Inverted Index:** The map function parses each document, and emits a sequence of (word; document ID) pairs. The reduce function accepts all pairs for a given word, sorts the corresponding document IDs and emits a (word; list(document ID)) pair. The set of all output pairs forms a simple inverted index. It is easy to augment this computation to keep track of word positions.
- **Distributed Sort:** The map function extracts the key from each record, and emits a (key; record) pair. The reduce function emits all pairs unchanged.

Another application is about, the inverted index, which all the search engines are mostly does this, let us see, what this application? Is and how the Map Reduce can be used to program, doing this inverted index? So, here, in this application the map function parses, each document and emits a sequence of world and document ID pair, the reduced function accepts, all the pairs of a given word sorts the corresponding document IDs and emits, the word and list of document ID pair. So, the set of all output pairs forms a simple inverted index it is, easy to augment, this configuration to keep track of the word positions. So, we will, in the later slides you will see, more detail of MapReduce program for this application that is, for inverted index that is, it is possible that if, the set of documents are given, for example search engine does this, Google when we type a keyword it gives you all the list of web pages? Where those documents? Where these keywords are appearing and that is being performed using inverted index? So, every search engine ,often computes this inverted index and that if, the number of documents is huge, the search engine like Google, are being by Microsoft all, they are, they are basically computing the inverted index and whenever the user searches it, it will perform it will check this, inverted index and

gives, that particular outcome. We will see, in more details how, this MapReduce exactly program this, inverted index application, similarly the distributed sort, that is the map function extracts the key, from each record and emits, the key and record pair the reduced function just emits all the pairs unchanged, that means automatically, internally the map function gives, when it emits the key and required, pair it provides in this sorted order and if there is, nothing different happens, into the shuffle phase, then if the output is, given as it is, then this, particular outcome of the map face will be taken up, as and it will be emitted unchanged, by the reduced function and this will perform the distributed sort.

Refer Slide Time :(10: 18)

Applications of MapReduce

(1) Distributed Grep:

- Input: large set of files ✓
- Output: lines that match pattern ✓
- Map – *Emits a line if it matches the supplied pattern*
map(input) emit(line) ✓
- Reduce – *Copies the intermediate data to output*
reduce(line) ✓

We will see, in more detail of distributed sort application, how it is done in the Map Reduce? Applications of MapReduce, we are going in now, little more details, of these applications which we have, some of them which we have summarized. Let us take the example of distributed grab how using MapReduce we can perform this distributed group operation? We assume, that the input is a large collection of files and the output, I want to get is, the lines of a file, that matches that particular pattern, that matches a given pattern. So, the map function, will emit, a line if it, matches the supplied pattern. So, map function will, emit a line. So, here the things are quite simple, why because? Whenever the line is matched, in the map function, line and a pattern. So, it emits only the line and the reducer does not have to do anything it will copy the, all the intermediate data, which is given by the map function and it, will output.

Refer Slide Time :(11: 53)

Applications of MapReduce

(2) Reverse Web-Link Graph:

- **Input:** Web graph: tuples (a, b)
where (page a → page b)
 - **Output:** For each page, list of pages that link to it
 - **Map –** process web log and for each input <source, target>, it outputs <target, source>
 - **Reduce -** emits <target, list(source)>
- web-graph
Example
Source target
emits(target, source)
Ex - emit(A, B)
emit(C, A)
emit(C, B)

So, this will become the distributed graph. Reverse web link graph we have already, seen let us see again, for the sake of completeness and we assume, that the web graph is, given in the form of a tuples, A, B. So, again I am drawing the same picture, let us say it is, A B and C these are the web pages, web pages and let us, say that A is pointing to C and B is pointing to A. So, in this example, the tuples which is given as the input are AC, then BC, then BA. And so, as far as, the map is concerned map function on getting this, as the input. So, that means these are the edges, which are given as the input to the map function and that means the source and the target, source this is, the source and this is, the target. What it does is it emits<target, list(source)>? That means, for example for BA, it will, emit(A, B) and for AC it will, emit(C, A) and for BC it will emit (C, B). Now, after doing this emit, the reduced function will accept this target, target means these, things will be accepted here, in this reduced function. And we will, form the list that is the, the list of sources. So, for C, it will emit (C, (A, B)) this will be the output and for, for A it will emit, the B itself. So, for a page C, A and B they are pointing you can see, in this particular picture, A for C A and B they are pointing to it and for, for A B is pointing to it and this is called, ‘Reverse Web Link’. So, output for each page, the list of pages that link to it that we have already achieved, in this particular application using MapReduce program. So, you can so, the programmers can easily, write down the MapReduce program for different this application we have seen, the reverse web link graph.

Refer Slide Time :(15: 14)

Applications of MapReduce

(3) Count of URL access frequency:

- Input: Log of accessed URLs, e.g., from proxy server
- Output: For each URL, % of total accesses for that URL

- Map – *Process web log and outputs <URL, 1>*
 - Multiple Reducers - *Emits <URL, URL_count>* $\langle \text{url}, \text{url_count} \rangle$
(So far, like Wordcount. But still need %)
 - Chain another MapReduce job after above one
 - Map – *Processes <URL, URL_count> and outputs <1, (<URL, URL_count>) >* $\langle 1, (\text{url}, \text{url_count}) \rangle$
 - 1 Reducer – Does two passes. In first pass, sums up all *URL_count's* to calculate *overall_count*. In second pass calculates %'s
- Emits multiple <URL, URL_count/overall_count>*

Similarly if you want to find out, if you want to count, the URL access frequency, that means the input is in the form of the log file which has accessed URLs and normally, this particular log file we can obtain from the proxy server, which maintains the log of URLs, which are accessed by the different clients. So, out of this, particular log analysis and the output we want is, that for each URL, we want to find out the percentage of total accesses, for the URLs. So, we want to find out and then rank it, later on we have some other applications. So, how to find out for a particular URL how? What is the percentage of accesses for that URL according to that log accesses? So, the map for this particular program will require to get the weblog and output and it will emit (URL, 1). So, for every URL it encounters in, in the weblog, it will emit(URL,1), the map function will do this, output. Now, then to find out this, access frequency, in the percentage it requires multiple, reducers. So, the first reducer, it will emit the URL and the URL count. So, that means, it will, do this, for every URL, it will also, do a count. So, it is just like, what count program? Like what count it will do a URL count? So, out of this , particular URL count, the map function, another map function will, will execute, it will take the (*<URL, URL_count>*) and output as 1, (*<URL, URL_count>*). So, after out after this, output the reducer will now, perform two different passes, in the first pass it will, sum all the URL counts, to calculate the overall count and in the second pass it will, calculate the percentage of that URL and percentage of that URL. So, it will emit the multiple URL values and URL count divided by the overall count. So, in this particular, example we have seen, not only one Map Reduce but, a series of Map Reduce. So, this is, the first Map Reduce function, will emit this, one URL and URL count and which will be taken by another Map Reduce function, which will compute, which will now, compute, which will emit<1, (*<URL, URL_count>*)>. which is it will emit one and whatever values we are getting and then there is, a third Map Reduce which it will now, calculate all the sum and find out the URL. So, it will, emit(*<URL, URL_count>*), divided by overall count. So, they see that, this is a chain of, MapReduce functions, which are required to solve this particular problem. So, earlier examples which we have shown only one MapReduce but, now we have shown that several sequence of MapReduce, are required to solve. So, if there is, such complicated or complex applications are there. So, it is, possible to make a chain of MapReduce job and solve this particular problem.

Refer Slide Time :(19: 44)

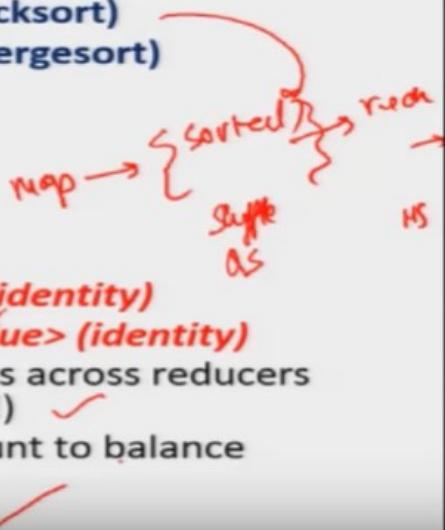
Applications of MapReduce

(4) Map task's output is sorted (e.g., quicksort)

Reduce task's input is sorted (e.g., mergesort)

Sort ✓

- Input: Series of (key, value) pairs ✓
- Output: Sorted <value>s ✓
- Map – $\langle \text{key}, \text{value} \rangle \rightarrow \langle \text{value}, _ \rangle$ (identity) ✓
- Reducer – $\langle \text{key}, \text{value} \rangle \rightarrow \langle \text{key}, \text{value} \rangle$ (identity) ✓
- Partitioning function – partition keys across reducers based on ranges (can't use hashing!) ✓
 - Take data distribution into account to balance reducer tasks ✓



Another application we will see, about the sorting and here, the input is given in the form of a $\langle \text{key}, \text{value} \rangle$ pairs and we want the sorted, values to be output. This particular program as we have, shown you quite simple for example, the input whatever is given to the map function $\langle \text{key}, \text{value} \rangle$, it will output only the value? And the reducer job, also will just output this $\langle \text{key}, \text{value} \rangle$, whatever is there? So, in this particular process, when the map, outputs these values, which are already in the sorted form, normally quick sort is done, here when during the shuffle phase and if the same thing is output, in the in the reduced function. So, it passes on and it uses the mud shot. So, it is a quick sort and the reduced function uses the merge sort. So, quick sort and merge sort together, will sort the applications and we don't have to do much, the partitioning function we have to be careful ,during the sort is that partitioning, partition keys across the reducer, is based on the on the ranges and you cannot use hashing, otherwise it will disturb the sorted order. Okay?

Refer Slide Time :(21: 14)

The YARN Scheduler

- Used underneath Hadoop 2.x + ✓

- YARN = Yet Another Resource Negotiator

(Resource Manager & Scheduler)

- Treats each server as a collection of **containers**

- Container = fixed CPU + fixed memory

- Has 3 main components ✓

- Global Resource Manager (RM) ✓

- Scheduling ✓

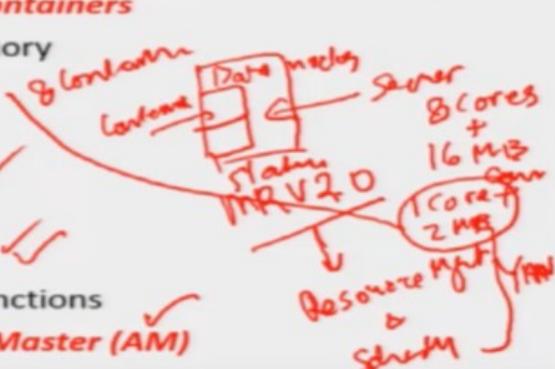
- Per-server Node Manager (NM) ✓✓

- Daemon and server-specific functions

- Per-application (job) Application Master (AM) ✓

- Container negotiation with RM and NMs ✓

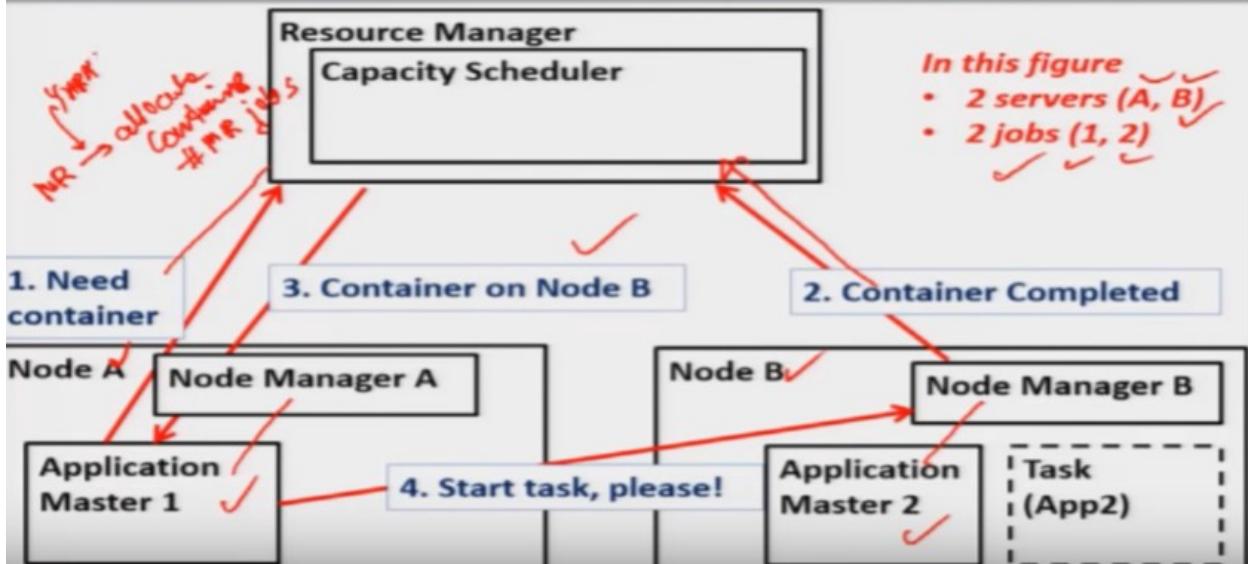
- Detecting task failures of that job ✓



The YARN scheduler. So, for MapReduce, job scheduling and resource management, YARN is used. So, let us see and go through the YARN scheduler in more detail because, in Map Reduce version 2, 2.0 the scheduling resource management and scheduling, is done by the YARN. So, let us see, how the YARN does this scheduling? So, it is used underneath, the Hadoop 2.0 versions and onwards. So, YARN full form is yet, another resource negotiator and its job is the, the resource manager and scheduler. Now, this YARN treats each server as a collection of containers. So, by this means is that so, the data nodes which are called as, 'Slaves'. Is in the form of containers, containers is, a container is having a CPU and a fixed memory. So, for example if let us say, a data node or this machine or a server has let us say, eight cores and has some memory, let us say 16 MB of space. So, the container will contain, a one core and let us say, 2 MB of space this is, one container. So, it will have, an four eight different container in this example, eight containers, in this, configuration. So, it depends upon, how many cores are there in the server? So, that many number of containers and the memory together can form the containers and container is the unit, which is being allocated, by the YARN scheduler, for MapReduce job. So, it has three different components, YARN has three different components, one is called, 'Global Resource Manager' and which performs the overall scheduling and for per, node manager it is called, 'Per Node Manager'. It is called, 'Node Manager' also, daemon and the server functions, it will specify and for per application, that is the job, there is another job application master and this application master will negotiate, with YARN, for getting the container, with there is, the resource manager and the node manager and whenever it detects a failure? For that particular job the application master again contacts the, the YARN component that is the resource manager and a node manager.

Refer Slide Time :(24: 22)

YARN: How a job gets a container



Let us see, how this all flows are in this YARN between the resource manager and node manager? So, let us say that, we have two servers A and B, A and B there are two servers, A and B and we have a two jobs, one and two which are to be allocated. So, how it does is first of all, this particular resource manager is contacted and it knows, about the scheduling or it will schedule these jobs to be allocated? The containers and then it will be shade ruled over there. So, this will, this client will give a request to the to the resource manager, that is a component of a YARN, for she ruling these two jobs and this particular resource manager knows, that there are two servers A and B, with the available container, within it. So, let us assume, that there is one container available, on B and there is, containers available on A also. So, these containers, after allocate after allocation, it is informed to the application master and the application master in turn, contact with the node manager and starts, the execution on these containers and once the application, execution is over, then these containers, then this application master will inform and the containers will be returned back. So, just see that, that means MapReduce jobs, with the help of YARN, MapReduce jobs, with the help of YARN, allocates the container, for that many number of, for the number of MapReduce jobs. So, this is done, through the help of YARN. So, this is explained here, in this particular picture. Thank you.

Lecture 8

MapReduce Examples

MapReduce examples,

Refer slide time :(0:15)

Example: 1 Word Count using MapReduce

```
map(key, value):
// key: document name; value: text of document
for each word w in value:
    emit(w, 1)

reduce(key, values):
// key: a word; values: an iterator over counts
result = 0
for each count v in values:
    result += v
emit(key, result)
```

Annotations in red:

- map(key, value):
- // key: document name; value: text of document
- for each word w in value:
- emit(w, 1)
- reduce(key, values):
- // key: a word; values: an iterator over counts
- result = 0
- for each count v in values:
- result += v
- emit(key, result)

Handwritten annotations:

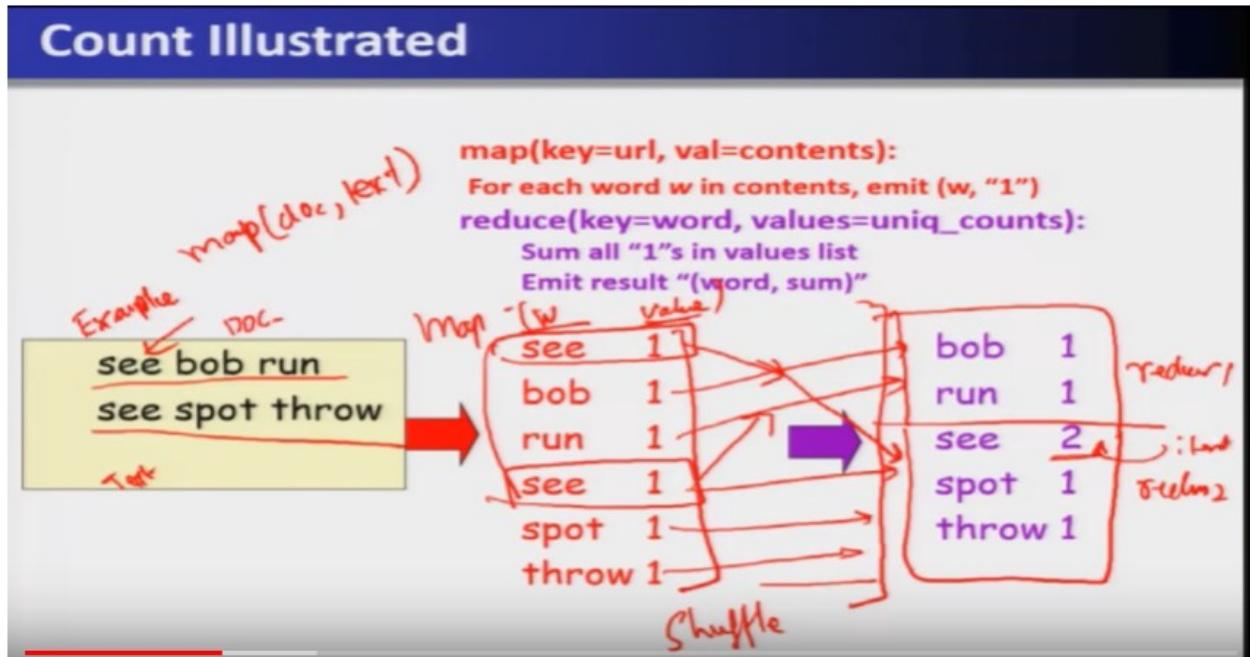
- map emit
- mapemit

Example one, word count using MapReduce? So, here we see the structure of map and reduce program which, will do the word count, in a given document.

```
map(key, value):
// key: document name; value: text of document
for each word w in value:
    emit(w, 1)
```

```
reduce(key, values):
// key: a word; values: an iterator over counts
result = 0
for each count v in values:
    result += v
emit(key, result)
```

Refer slide time :(01:57)



Let us see this illustration through this running example, now in this example we consider a document, let us say this name of a document, docx file and this is the text, of the document now, when this particular name of the document, is given and the text is given. So, these particular text words, as it is appearing it will emit word and 'w'. So, here we see that, 'w' and this is the value 1 and this is the value. So, as these words are appearing out of this particular text, this map function will emit, these w and 1 out of this particular previous program. So, after this emit of these words, it will do a shuffle, in so-called, what it will do the same words it will try to, collect together, it will sort them and collect them, together and then pass on, this is the shuffle phase and then pass on to, the reducer phase. Now it depends upon how many reducers we have let us assume we have one and two different reducers. So, as far as this Bob is concerned, it will be passed on over here this will pass on over here Bob will go to this particular function a run will go to this particular function, to this reducer I spot will go here, here and throw will go here. And this reducer will now, for example in 'see', there are two different, these values when it comes and iterator it will, go through this iterator function and make the summation of it.

Refer slide time :(04:34)

Example 2: Counting words of different lengths

- The map function takes a value and outputs key:value pairs.
- For instance, if we define a map function that takes a string and outputs the length of the word as the key and the word itself as the value then
 - map(steve) would return 5:steve and
 - map(savannah) would return 8:savannah.

Example Doc.
This is my pen
4 2 2 3
4:1 3:2 3:1

This allows us to run the map function against values in parallel and provides a huge advantage.

Now let us see another example, example number two, counting words of different lengths. So, in a given document we have to find out, the documents of a given length for example, ‘this is my pen’. So, here you see that this word is of length four this word is of length two this word or is of length two this word is of length three. So, counting the words of different lengths is, you see this particular word, of length 2 is appearing two times, the word of length four is appearing one times word of length three is appearing one times this we have to give as an output from for a given document file. So, let us see how map and reduced function, is utilized to do this particular job.

Refer slide time :(05:40)

Example 2: Counting words of different lengths

Before we get to the reduce function, the mapreduce framework groups all of the values together by key, so if the map functions output the following **key:value pairs**:

3 : the
3 : and
3 : you
4 : then
4 : what
4 : when
5 : steve
5 : where
8 : savannah
8 : research

map
emit (length, word)
key value
length
word
reduce
(Key, list)

They get grouped as:

3 : [the, and, you] ✓
4 : [then, what, when]
5 : [steve, where]
8 : [savannah, research]

Now to do this counting words of different lengths, we have to design the map and reduce function. So, given this particular document, the map function will emit the key and the value. So, the key comprises of the length, of about and the word itself. So, for example, at emit 'the' this is the word, this will be emitted and also the length, will be emitted so key value pair will be key, will be the length, here length of the word and the word itself will become the value this will be emitted out of the map function. Now after that what the reducer? Will do reducer will accept this format and then it will be group by so, the reducer, will now collect the, the key and corresponding it will outcome the list of the words. So, for example here, you can see the word of length 3 it is 3 different types of 3 different words, which are appearing, is being clubbed together, as a list. So, this will be the output of the reduce function.

Refer slide time :(07:27)

Example 2: Counting words of different lengths

- The reductions can also be done in parallel, again providing a huge advantage. We can then look at these final results and see that there were only two words of length 5 in the corpus, etc...
- **The most common example of mapreduce is for counting the number of times words occur in a corpus.**

So, the reductions can be done in parallel, again providing a huge advantage, we can then look at these final designs and see that there are only two words of length 5 in the corpus and only two words of length 8 and so on. So, this we have seen through an example that using MapReduce program complex program or application can be easily programmed and this is going to be used in a big data applications.

Refer slide time :(08:01)

Example 3: Word Length Histogram

Abridged Declaration of Independence

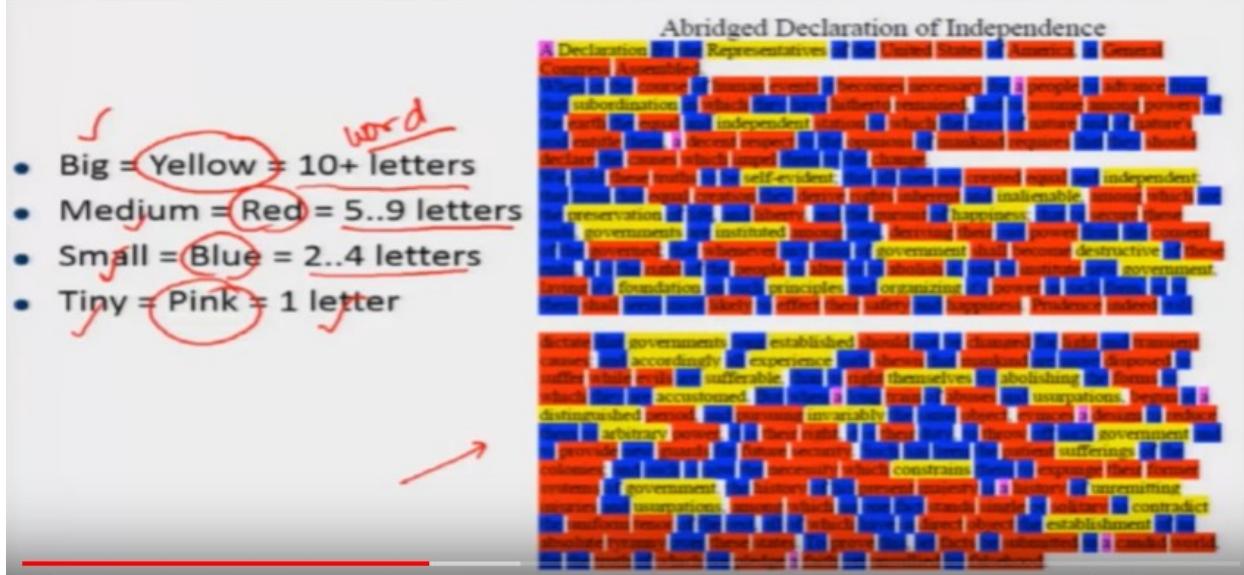
A Declaration By the Representatives of the United States of America, in General Congress Assembled. When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change. We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying its foundation on such principles and organizing its power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

How many “big”, “medium” and “small” words, are used ?

Example number 3; Here we have to find out the word length histogram and if this particular document is given and then we what we have to, find out that how many big medium and small words are appearing in this particular document and this becomes the word length histogram.

Refer slide time :(08:23)

Example 3: Word Length Histogram



For example,

Big = Yellow = 10+ letters

Medium = Red = 5..9 letters

Small = Blue = 2..4 letters

Tiny = Pink = 1 letter

So, given this particular document if we color them according to this word length categorization so, the document will look like, in this manner now we have to find out the word and histogram.

Refer slide time :(09:19)

Example 3: Word Length Histogram

Split the document into chunks and process each chunk on a different computer

Chunk 1

Chunk 2

Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled.

When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.

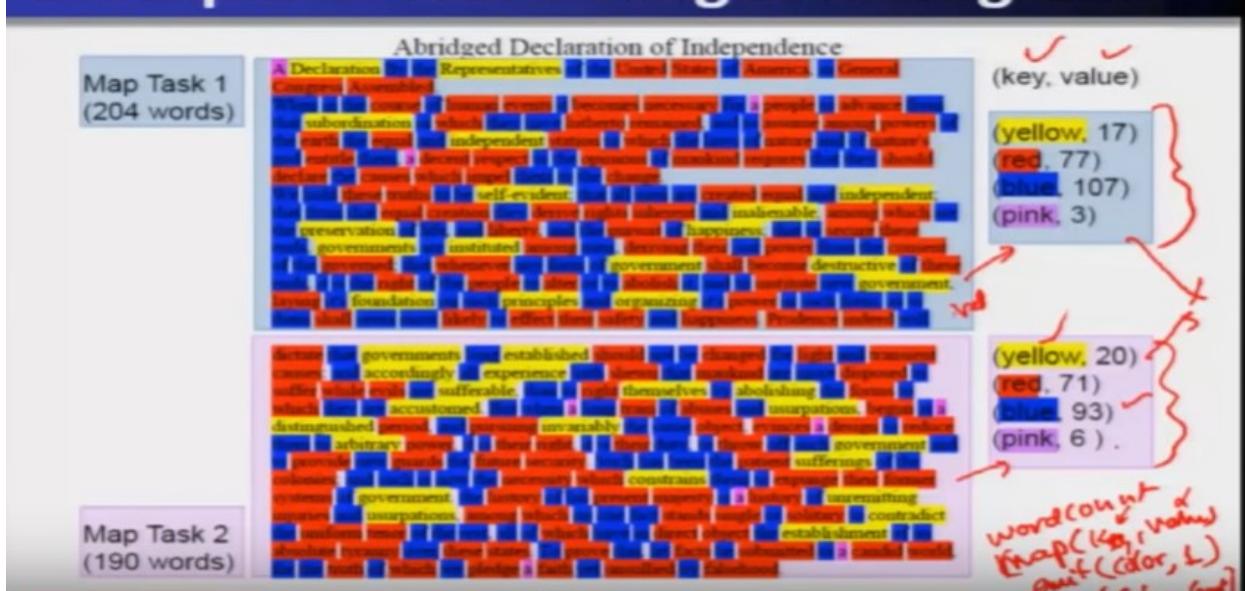
We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying its foundation on such principles and organizing its powers in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will

dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government, the history of his present majority is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

Now to do this you know that in MapReduce program, we can divide this entire document into the chunks let us assume that this is upper, when this particular this portion of a document, is chunk one the other portion the remaining portion is chunk two so we divided into two chunks.

Refer slide time :(09:41)

Example 3: Word Length Histogram



And now we have to write down the map function. So, map function will be based on, this word count application, wherein if a word is given and the map functions will take the key and the value. So, key will be this document and this value will be these words out of this particular, document is written now whenever a word is, there then through this particular length, we have to find out whether it, will emit whether it is the color, of our award and one will be emitted, similarly as far as the reduce function, is concerned reduce for every key it will make a sum of a count, I it will do the aggregation. So, count means the iterator, iterator will make the sum and this will be the output. So, in this the, the chunk one will omit this statistics that is yellow 17 different times it is appearing red 77 blue 107 and pink is 3 similarly the, the map task 2, will emit the yellow as 20 red as 71 and blue as 93. So, these numbers are internally done, that means that every task the reduce function is applied that is why instead of 17 times once it is doing this, now this reduce again will combine them and give the final outcome that yellow is 37 and red is 148 and blue is 200 and this pink is 9.

Refer slide time :(11:56)

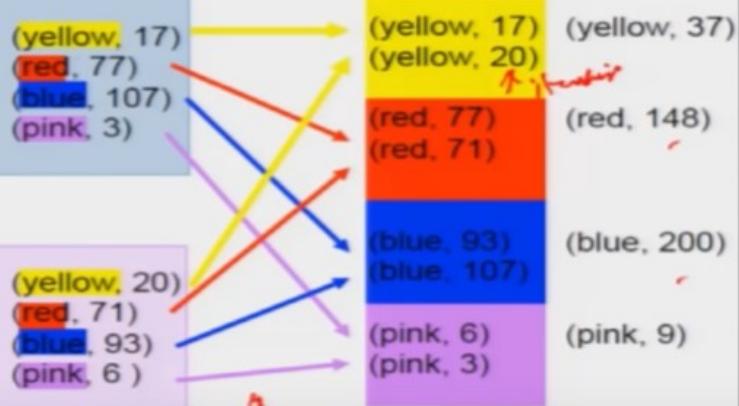
Example 3: Word Length Histogram

Map task 1

A Declaration By the Representatives of the United States of America, as Followed.
When as the cause of human errors it becomes necessary for a people to reduce these
that vulnerabilities as which they have better resources, and to increase many powers of
the earth the equal and independent nations in which the laws of man and of nature's
and to increase the power of the people to reduce these in the case of numerous responses, but they cannot
decide the causes which caused these to change.
We hold these truths to be self-evident: that all men are created equal and independent,
that those that equal countries they derive rights inherent and inalienable, among which are
the pursuit of happiness, and that to secure these rights, that governments are instituted among men, deriving their just power from the consent
of the governed; that whenever any form of government becomes destructive of these ends,
it is the right of the people to alter or to abolish it, and to institute new government,
laying its foundation on such principles, and establishing a new constitution of its own; or to
these shall never cease to effect their safety and happiness. Providence willing will

"Shuffle step"

Reduce tasks



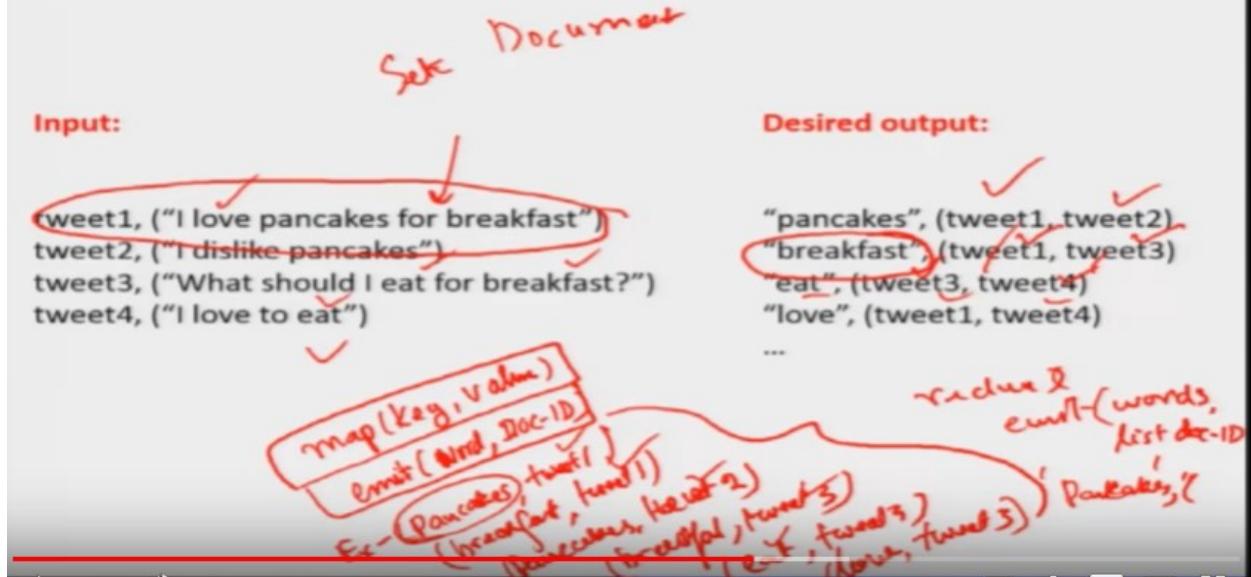
Map task 2

Shows that governments long established should not be changed for light and transient
causes, and accordingly all experience both causes that resulted in some disposed to
offer while evils are suffered, than in right instances by associating the terms in
which they are accomplished. But when a long time of slaves and serfs, begins at a
certain point, and the slaves and serfs are disposed to offer while evils are suffered, than in
which they are accomplished, it is their right, or in their duty, to throw off such government and
to provide new guards for future security. Such has been the patient sufferings of the
colonies, and such is now the necessity which compels them to expunge their former
creations, and to establish a new nation, for the purpose of securing their
independence and tranquillity, having which no one but stands single or separate to combat
the sudden issue of the rest, all of which have in direct object the establishment of an
absolute empire over these states. To prove this, let facts be collected in a candid world,
in the view of which we judge it best for mankind by themselves.

So, if we see these different steps so, this step is called a, ‘Shuffle Step’. And through this different 17 and 20 here there will be an iterator and thus reduce function will long add, after going through this iterator and these values will outcome. So, this example shows the word length histogram which is nothing but an extension of word count program

Refer slide time :(12:28)

Example 4: Build an Inverted Index



Now there is another example for to build an inverted index. So, by mean inverted index is, for example if a document is given or a set of documents is, is given then we want to find out the, the text or a particular word, which is appearing in a particular document and then we will finally collect a world which is appearing in all list of all documents where it is appearing, this is called, 'Inverted Index'. And search engines are now using this concept. So, let us see how this happens so map function will take a key value pair and will emit the, the name of the word that is called, 'value'. What value? And the document ID it will emit. So, for example in this particular case,

Input:

tweet1, ("I love pancakes for breakfast")
 tweet2, ("I dislike pancakes")
 tweet3, ("What should I eat for breakfast?")
 tweet4, ("I love to eat")

Desired output:

"pancakes", (tweet1, tweet2)
 "breakfast", (tweet1, tweet3)

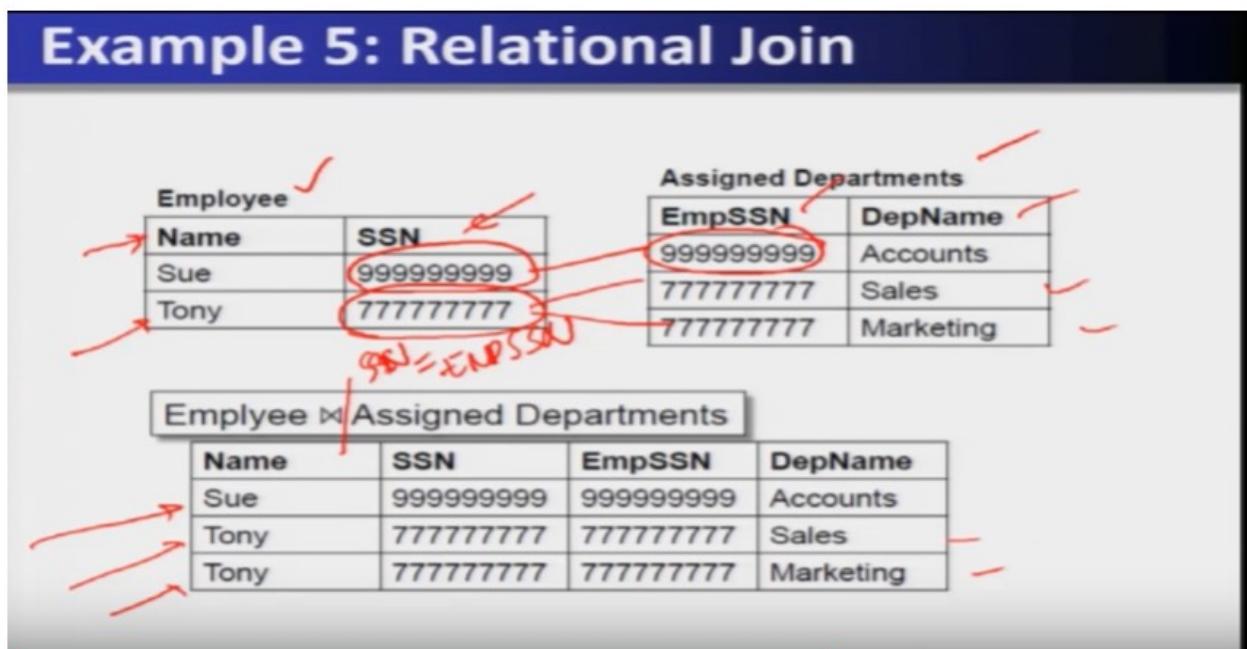
“eat”, (tweet3, tweet4)

“love”, (tweet1, tweet4)

...

So, this will form the inverted index, using simple MapReduce program and whenever a search engine uses, the word breakfast, for searching so, it will give these two documents, to it one and two it three where this breakfast, was being referred in that document.

Refer slide time :(16:48)



Now we will see the operations, how we are going to perform the relational, join operation using MapReduce, by relational join let us understand this example and then we will see, how that is done using MapReduce. Let us say that employee is a table having, the attributes as a name and its SSN number, similarly another document which is called, ‘Assigned Departments’, here we have in SSN and the department name. Now if you want to join on, employee and assigned departments, if you want to join on SSN equal, employ SSN with the accounts. So, if we join them this particular tuple will be generated, what about the other one here this SSN has two matches. So, therefore this particular tuple two different tuples will be generated accordingly wherein this is the sales and marketing will be reflected. Now let us see how this we are going to achieve using MapReduce operation.

Refer slide time :(18:25)

Example 5: Relational Join: Before Map Phase

Employee

Name	SSN
Sue	99999999
Tony	77777777

Assigned Departments

EmpSSN	DepName
99999999	Accounts
77777777	Sales
77777777	Marketing

Key idea: Lump all the tuples together into one dataset

Employee, Sue, 99999999
Employee, Tony, 77777777
Department, 99999999, Accounts
Department, 77777777, Sales
Department, 77777777, Marketing

map side - unary operation
join - binary operation
join Table 1
Key, value Key
What is this for?
map (key, value) reduce
out (ssn, tuple)

Now before going into details we have to understand that map and MapReduce is a unary operation and this join is a binary operation, when it requires two tables, table 1 and table 2. So, how this MapReduce which is a unary operation, will be used to do a relational join and that we have to see that now what we will do is we consider the entries or the tuples of the table as, as a single tuples, as the collection of all the tuples and we will attach this identity of the name of a table also. So, it becomes a key value pair so, key value pair means that, the name of the table will become let us say key and the remaining couple will become the value this way we will list out all the tuples, which are there in different tables. Now if this becomes, a complete data set then we can perform the join operation, easily how the join is happening ? join is happening around a particular key. So, around a SSN number so SSN number will become the key here in this case and we will omit the, the Map Reduce will, will emit the key and the value. So, key will be this SSN number and the value it will emit will be the entire tuple. So, now as far as the reduce is concerned, reduce will now group by this key and, and then after group by key then, it will try to do the iterator and if these table IDs are different, then it will make a joint operation within it.

Refer slide time :(20:50)

Example 5: Relational Join: Map Phase

Employee, Sue, 999999999
Employee, Tony, 777777777
Department, 999999999, Accounts
Department, 777777777, Sales
Department, 777777777, Marketing



Join ✓



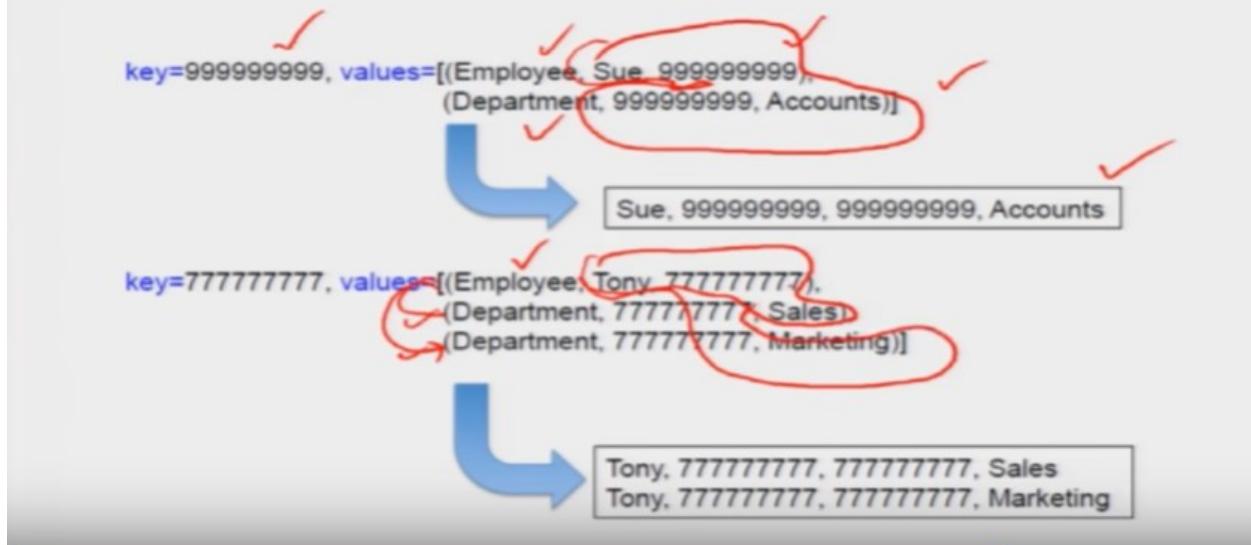
key=999999999, value=(Employee, Sue, 999999999)
key=777777777, value=(Employee, Tony, 777777777)
key=999999999, value=(Department, 999999999, Accounts)
key=777777777, value=(Department, 777777777, Sales)
key=777777777, value=(Department, 777777777, Marketing)

why do we use this as the key?

So, let us see here in this case as I told you that you we have to decide what is the key and what is the value. So, key will be the, the, the SSN number around which we are going to join. So, whatever is the join become, the key and the entire tuple including the name of the table or a name of the document s and all the tuple will be that value. So, this after, after finding out this kind of

Refer slide time :(21:23)

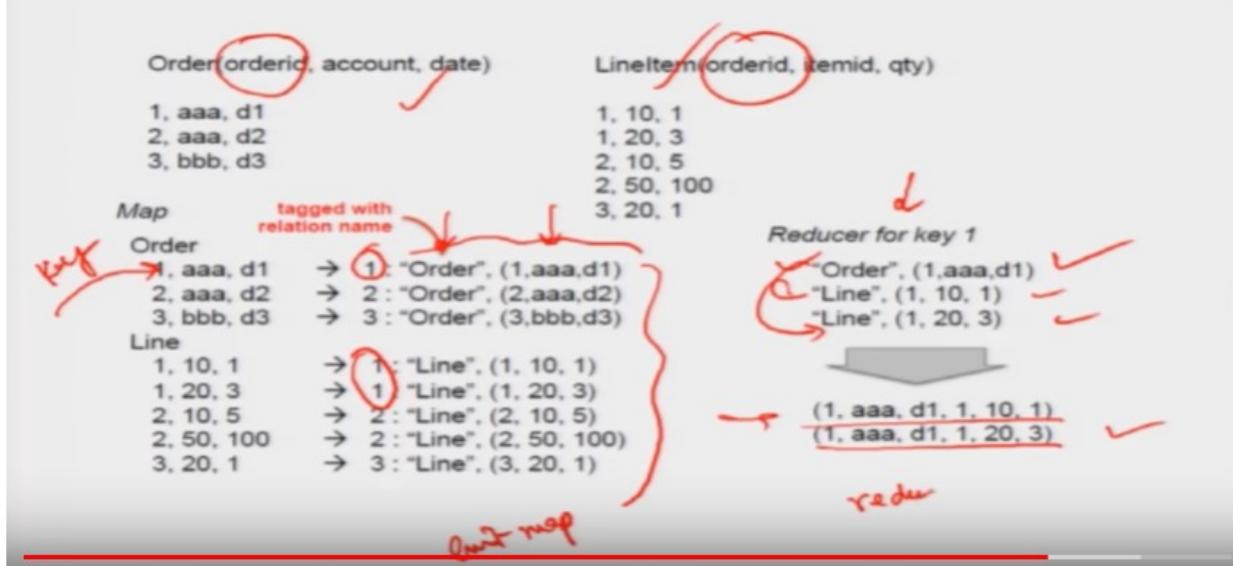
Example 5: Relational Join: Reduce Phase



things, then now we have to do a join operation. So, in the join we will see that when we make a group by key, let us say that two different tuples are appearing and since they are employ their, their tables are different. So, we are going to join them using these different notions and similarly here in the other case, we have a employ and two different Department. So, this employee will be joined so, it will generate two different tuples where in the Tony and Sales, is one tuple Tony and Marketing will be another tuple this way the join operation.

Refer slide time :(22:10)

Example 5: Relational Join in MapReduce, again



Relational join we can perform now if you take another example, let us say that we have two different tables, one is the order table the other is line item table and their tuples are listed over here then with a map function, what we will do is so, we have to join around order ID, if we are going to join around order ID. So, order ID will become the key so, the map will emit the key and the corresponding all the values that is the name of the table and the, the corresponding. So far both the tables, it will generate this, this will emit out of the map function. Now then the reducer will combine, by this one key so, it will group by key for example this order, and for example key number one is appearing three times. So, one with the order the other is with the line. Now then we are going to combine this order with two different lines. So, basically it will generate two different tuples which is shown over here. so, by this example we have shown

Refer slide time :(23:32)

Example 6: Finding Friends

- Facebook has a list of friends (note that friends are a bi-directional thing on Facebook. If I'm your friend, you're mine).
- They also have lots of disk space and they serve hundreds of millions of requests everyday. They've decided to pre-compute calculations when they can to reduce the processing time of requests. **One common processing request is the "You and Joe have 230 friends in common" feature.**
- When you visit someone's profile, you see a list of friends that you have in common. This list doesn't change frequently so it'd be wasteful to recalculate it every time you visited the profile (sure you could use a decent caching strategy, but then we wouldn't be able to continue writing about mapreduce for this problem).
- We're going to use mapreduce so that we can calculate everyone's common friends once a day and store those results. Later on it's just a quick lookup. We've got lots of disk, it's cheap.

that using MapReduce, we can perform the relational join, example number six, for finding the common friends. So, this kind of finding friends is very common in the social networks, like, Facebook. So, Facebook has a list of friends and we want to find out the your list of common friends, which are there in the social network like, our Facebook now this kind of operation of finding common friends is quite common, when you visit someone profile and see the list of friends that you have the common this list does not change frequently and but it has to be calculated, at a regular intervals using Facebook.

Refer slide time :(24:32)

Example 6: Finding Friends

- Assume the friends are stored as **Person->[List of Friends]**, our friends list is then:

- A -> B C D
- B -> A C D E
- C -> A B D E
- D -> A B C E
- E -> B C D

*input
Person -> [List of friend?]*

So, let us see that how using MapReduce we are going to perform this kind of operation. So, assume that the friends are told in this format, for example a person and the list of all the friends is basically given as the input for example person

A → B C D

B → ACDE and so on.

So, the person and arrow followed by, the list of friends is given as the input to this particular program, now then we will see that.

Refer slide time :(25:11)

Example 6: Finding Friends

For map($A \rightarrow B C D$) :

(A B) -> B C D

(A C) -> B C D

(A D) -> B C D

$$\begin{array}{l} AB \rightarrow BCD \\ AC \rightarrow BCD \\ AD \rightarrow BCD \end{array}$$

For map($B \rightarrow A C D E$) : (Note that A comes before B in the key)

(A B) -> A C D E

(B C) -> A C D E

(B D) -> A C D E

(B E) -> A C D E

→ .

How to find out a common friend, is first thing is for a map function, what we will do is? So, if the person A and all the list of common friends is given so, we are going to generate a tuple:

For map($A \rightarrow B C D$) :

(A B) -> B C D

(A C) -> B C D

(A D) -> B C D

For map($B \rightarrow A C D E$) : (Note that A comes before B in the key)

(A B) -> A C D E

(B C) -> A C D E

(B D) -> A C D E

(B E) -> A C D E

Refer slide time :(26:00)

Example 6: Finding Friends

- Before we send these key-value pairs to the reducers, we group them by their keys and get:

(A B) -> (A C D E) (B C D)
✓(A C) -> (A B D E) (B C D)
(A D) -> (A B C E) (B C D)
(B C) -> (A B D E) (A C D E)
(B D) -> (A B C E) (A C D E)
(B E) -> (A C D E) (B C D)
(C D) -> (A B C E) (A B D E)
(C E) -> (A B D E) (B C D)
(D E) -> (A B C E) (B C D)

And finally the next thing has to be done by the reducer. So reducer will find out, reducer will reduce a group by key, for example A B is a key and it has two different, such lists which has been obtained and now similarly, for a key a see two different lists and so. So, once this is obtained so, reducer

(A B) -> (A C D E) (B C D)
(A C) -> (A B D E) (B C D)
(A D) -> (A B C E) (B C D)
(B C) -> (A B D E) (A C D E)
(B D) -> (A B C E) (A C D E)
(B E) -> (A C D E) (B C D)
(C D) -> (A B C E) (A B D E)
(C E) -> (A B D E) (B C D)
(D E) -> (A B C E) (B C D)

Refer slide time :(26:24)

Example 6: Finding Friends

- Each line will be passed as an argument to a reducer.
- The **reduce function will simply intersect the lists of values** and output the same key with the result of the intersection.
 - For example, **reduce((A B) -> (A C D E) (B C D))** will **output (A B) : (C D)**.
 - **and means that friends A and B have C and D as common friends.**

Will find out by taking the intersection, of these lists of values and that becomes stuck. A list of common friends, for example,

reduce((A B) -> (A C D E) (B C D))
will **output (A B) : (C D)**

Refer slide time :(26:49)

Example 6: Finding Friends

- The result after reduction is:
- (A B) -> (C D)
- (A C) -> (B D)
- (A D) -> (B C)
- (B C) -> (A D E)
- (B D) -> (A C E)
- (B E) -> (C D)
- (C D) -> (A B E)
- (C E) -> (B D)
- (D E) -> (B C)

Now when D visits B's profile,
we can quickly look up (B D) and
see that they have three friends
in common, (A C E).

In this particular manner, by finding the intersection and for every set of persons, now in this manner,
we are computing the list of common friends, in parallel and

Refer slide time :(26:04)

Reading

Jeffrey Dean and Sanjay Ghemawat,

"MapReduce: Simplified Data Processing on Large Clusters"

<http://labs.google.com/papers/mapreduce.html>

this particular operation is being performed. Thank you

Lecture 09

Parallel programming with Spark

Parallel programming with the Spark.

Refer slide time :(0:17)

Preface

Content of this Lecture:

- In this lecture, we will discuss:
 - Overview of Spark
 - Fundamentals of Scala & functional programming
 - Spark concepts
 - Spark operations
 - Job execution

Preface: Content of this lecture. In this lecture we will discuss overview of Spark, fundamental of Scala and functional programming, Spark concepts, Spark operations and the Job execution.

Refer slide time :(0:30)

Introduction to Spark

Introduction to Spark.

Refer slide time :(0:32)

What is Spark?

- Fast, expressive cluster computing system compatible with Apache Hadoop
 - Works with any Hadoop-supported storage system (HDFS, S3, SequenceFile, Avro, ...)
- Improves **efficiency** through:
 - In-memory computing primitives
 - General computation graphs
- Improves **usability** through:
 - Rich APIs in Java, Scala, Python
 - Interactive shell

→ Often 2-10× less code

What is this Spark? It is fast expressive cluster computing systems, which is compatible to Apache Hadoop, now this particular Spark system works, with any Hadoop supported storage system such as HDFS, S3, sequential file. And so on which improves, the efficiency through in-memory computational, primitives and general computational graph. So, it also improves the usability through rich collection of API, is in the form of scale a Java Python, it has the interactive cell. So, this all comprises of the Spark, scenario. So, using this in memory computation, it is 100 times faster, compared to the previous generation MapReduce systems and also with the interactive, shell it has reduced often (2-10 x) less code here in this particular system.

Refer slide time :(01:44)

How to Run It

- Local multicore: just a library in your program
- EC2: scripts for launching a Spark cluster
- Private cluster: Mesos, YARN, Standalone Mode

So, how to run it basically using local multi-core system or using with the, with a private cluster using Mesos, YARN and standalone mode

Refer slide time :(01:57)

Scala vs Java APIs

- Spark originally written in Scala, which allows concise function syntax and interactive use
- APIs in Java, Scala and Python
- Interactive shells in Scala and Python

So, Spark originally was written in Scala, which allows concise function that is syntax and interactive use, there are APIs, which are available for Java and Scala and Python now. So, interactive shells are available in a Scala and Python.

Refer slide time :(02:18)

Introduction to Scala & functional programming

Now let us introduce, to the Scala functional programming language.

Refer slide time :(02:27)

About Scala

- **High-level language for the Java VM**
 - Object-oriented + functional programming
- **Statically typed**
 - Comparable in speed to Java
 - But often no need to write types due to type inference
- **Interoperates with Java**
 - Can use any Java class, inherit from it, etc; can also call Scala code from Java

So, it is a high-level language, for the java virtual machine, that is it can compile through the java virtual machine, byte code and it is statically, typed that is and then it is interoperates with the Java.

Refer slide time :(02:46)

Quick Tour

Declaring variables:

```
var x: Int = 7  
var x = 7 // type inferred
```

Java equivalent:

```
int x = 7;  
  
final String y = "hi";
```

Functions:

```
def square(x: Int): Int = x*x  
  
def square(x: Int): Int = {  
    x*x  
}  
  
def announce(text: String) {  
    println(text)  
}
```

Java equivalent:

```
int square(int x) {  
    return x*x;  
}  
  
void announce(String text) {  
    System.out.println(text);  
}
```

Declaring variables:

```
var x: Int = 7
var x = 7 // type inferred
val y = "hi" // read-only
```

Functions:

```
def square(x: Int): Int = x*x
def square(x: Int): Int = {
  x*x
}
def announce(text: String) {
  println(text)
}
```

And so, here we are going to see the quick tour, of functional programming language, that is the Scala. In Scala, the variables are defined using where function, where there are two ways, you can define the variable one is by specifying the type of the variable:

Refer slide time :(05:41)

Quick Tour

Generic types:

```
var arr = new Array[Int](8)  
var lst = List(1, 2, 3)  
// type of lst is List[Int]
```

Java equivalent:

```
int[] arr = new int[8];  
List<Integer> lst =  
    new ArrayList<Integer>();  
lst.add(...)
```

Indexing:

```
arr(5) = 7  
println(lst(5))
```

Java equivalent:

```
arr[5] = 7;  
System.out.println(lst.get(5));
```

is the generic types.

Generic types:

```
var arr = new Array[Int](8)
```

```
var lst = List(1, 2, 3)
```

```
// type of lst is List[Int]
```

Indexing:

```
arr(5) = 7  
println(lst(5))
```

Refer slide time :(06:33)

Quick Tour

Processing collections with functional programming:

```
val list = List(1, 2, 3)
list.foreach(x => println(x))      // prints 1, 2, 3
list.foreach(println)               // same
list.map(x => x + 2)              // => List(3, 4, 5)
list.map(_ + 2)                    // same, with placeholder notation
list.filter(x => x % 2 == 1)       // => List(1, 3)
list.filter(_ % 2 == 1)             // => List(1, 3)
list.reduce((x, y) => x + y)       // => 6
list.reduce(_ + _)                 // => 6
```

All of these leave the list unchanged (List is immutable)

▶ ▶ ⏴ 10:21 / 38:57

Parallel Programming CC SPARK

Now processing collection with the functional programming,

Processing collections with functional programming:

```
val list = List(1, 2, 3)
list.foreach(x => println(x)) // prints 1, 2, 3
list.foreach(println)         // same
list.map(x => x + 2)        // => List(3, 4, 5)
list.map(_ + 2)              // same, with placeholder notation
list.filter(x => x % 2 == 1) // => List(1, 3)
list.filter(_ % 2 == 1)       // => List(1, 3)
list.reduce((x, y) => x + y) // => 6
list.reduce(_ + _)           // => 6
```

Refer slide time :(10:25)

Scala Closure Syntax

```
(x: Int) => x + 2 // full version
x => x + 2 // type inferred
✓ _ + 2 // when each argument is used exactly once
x => { // when body is a block of code
  val numberToAdd = 2
  x + numberToAdd
}

// If closure is too long, can always pass a function
def addTwo(x: Int): Int = x + 2
list.map(addTwo)
```

function

Scala allows defining a "local function" inside another function

▶ ▶ ⏴ 11:49 / 38:57 Big Data Computing Parallel Programming CC BY-NC-SA

Here in this case now let us see the Scala closure syntax more of it.

```
(x: Int) => x + 2 // full version
x => x + 2 // type inferred
_ + 2 // when each argument is used exactly once
x => { // when body is a block of code
  val numberToAdd = 2
  x + numberToAdd
}
// If closure is too long, can always pass a function
def addTwo(x: Int): Int = x + 2

list.map(addTwo)
```

Refer slide time :(11:54)

Other Collection Methods

- Scala collections provide many other functional methods; for example, Google for “Scala Seq”

Method on Seq[T]	Explanation
map(f: T => U): Seq[U]	Pass each element through f
flatMap(f: T => Seq[U]): Seq[U]	One-to-many map
filter(f: T => Boolean): Seq[T]	Keep elements passing f
exists(f: T => Boolean): Boolean	True if one element passes
forall(f: T => Boolean): Boolean	True if all elements pass
reduce(f: (T, T) => T): T	Merge elements using f
groupBy(f: T => K): Map[K, List[T]]	Group elements by f(element)
sortBy(f: T => K): Seq[T]	Sort elements by f(element)
...	



12:34 / 38:57

Big Data Computing

Parallel Programming



Inside another function, there similarly there are other collection, methods which are available in the Scala. So, a Scala collection provides many other functional methods, for example as Google for a Scala sequence we can see on it. So, map function is can be applied and this will pass each element, through a function f and similarly flat map one-to-many map function, it will apply similarly for the filter it will keep the element passing f and exist means a true if one element passes, for all reduce group by and sort by all different methods are available.

Refer slide time :(12:35)

Spark Concepts

Now let us see the Spark concepts.

Refer slide time :(12:40)

Spark Overview

- **Goal: Work with distributed collections as you would with local ones**
- Concept: resilient distributed datasets (RDDs)
 - Immutable collections of objects spread across a cluster
 - Built through parallel transformations (map, filter, etc)
 - Automatically rebuilt on failure
 - Controllable persistence (e.g. caching in RAM)

So, here the goal of Spark is to provide a distributed collection and here the concept of Spark ,is to support, this distributed computation in the form of resilient, distributed data sets and RDDs are immutable collection of objects which are spread across the clusters and these RDDs they are built through a transformation, such as map filter etcetera and these particular, RDDs they are automatically rebuilt, on the failure, that means a lineage is automatically generated and whenever there is a failure it will be reconstructed, similarly it is also controllable, persistence that is the cache.

Refer slide time :(13:29)

Main Primitives

Resilient distributed datasets (RDDs)

- Immutable, partitioned collections of objects
- Transformations (e.g. map, filter, groupBy, join)
 - Lazy operations to build RDDs from other RDDs

Actions (e.g. count, collect, save)

- Return a result or write it to storage

In the rhyme is being done. So, the main primitives is RDDs which are immutable partition collection, of objects various transformations are applied on the RDDs such as, map filter and group by join and these are all lazy operations to build RDDs from, other RDDs and besides transformations, actions also can be defined on our RDD such as count collect, save and it will return the values or it will write it on the under disk.

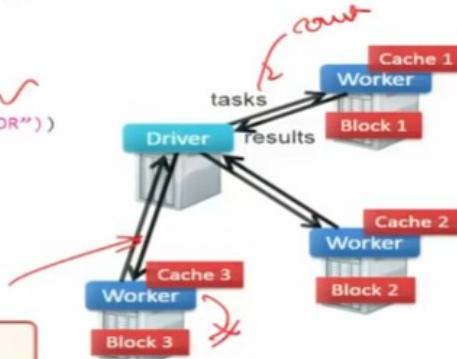
Refer slide time :(13:58)

Example: Mining Console Logs

- Load error messages from a log into memory, then interactively search for patterns

```
✓ lines = spark.textfile("hdfs://...") ✓
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split('\t')[2])
messages.cache()

messages.filter(lambda s: "foo" in s).count()
messages.filter(lambda s: "bar" in s).count()
...
Result: scaled to 1 TB data in 5-7 sec
(vs 170 sec for on-disk data)
```



Now let us see one example of, of mining the console logs, using the Scala program written, in Spark and here, it what it will do it will load the error messages, from a log into the memory and then interactively search for a pattern. So, let us see that, the Spark program now you, you, you know that it runs in a form of a master that is the driver and the worker, system and the first line written is to read this particular log files, that is in the from the HDFS, it will read the text file and this particular lines after reading it, it will become in memory. So, it will generate the, the base RDD after reading the file and then what it does is apply the filter, on the lines which are now red, into the in memory, it will apply the filter operation that we have seen, on the lines and in this filter operation, what it does is it will identify the, the errors which are appearing, in the in the log files and this will perform a transformation, after filtering and they are collected in the form of, another RDD that is errors RDD and using this errors, RDD down we can apply, the map function and which says that using this map function. Now you can split this, s to the wherever the tabs are there and they will be now dividing, these into the different error messages and these error messages are now cached.

So, apply the filter on these error messages that wherever this, foo is available or appearing, in the message and you have to make a count, of it how many such foo messages are appearing, in this error message. So, here the count is the action and before that all were transformations. So, let us see how it will be done. So, wherever these messages are restoring on that, this 'foo' will be counted so, just see that the driver will perform a task, to count. So, task is to count, this particular task will be communicated by the driver to all the workers and they will count and then return the result back to the driver and also will be in the cache, these messages will be in the cache. Now the next time when you want to filter the, the particular string that is called a, 'bar', from, the message then, it will be performed in the cache itself it will not go and look up into the into, the disk it will not be done in the disk, it will be now returned back from the from the cache itself. So, the results will be quickly returned back because so, therefore the full, text search of a Wikipedia can be done, in a less than one second if it is in the cache or, or it is 20 second

if it is done through the disk access. So, therefore if the data is scaled to one terabyte, then it will take five to seven seconds, if it is to be accessed through the cache or it is 170 seconds if it is on disk data

Refer slide time :(17:49)

RDD Fault Tolerance

RDDs track the transformations used to build them (their *lineage*) to recompute lost data

E.g:

```
✓
messages = textFile(...).filter(lambda s: s.contains("ERROR"))
    .map(lambda s: s.split('\t')[2])
✓ ✓
```



▶ ▶ 🔍 18:51 / 38:57 Parallel Programming CC BY-NC-SA

So, in all cases this entire big corpus of terabyte of data can be processed very efficiently and in a very quickly. Now let us see the RDDs fault-tolerance. So, RDDs will track the transformations, used to build them through the lineage to recompute the last data. So, here we can see that, once we specify these filter that is using, filter which contains the error and then it will split, all those messages, which are tab separated and this will be collected in the form of messages. So, let us see that, this particular filtered RDDs, will contain the information and they are now stored in the HDFS file system. So, RDDs keep track of the transformations, used to build them, their lineage to recompute the last data.

Refer slide time :(18:55)

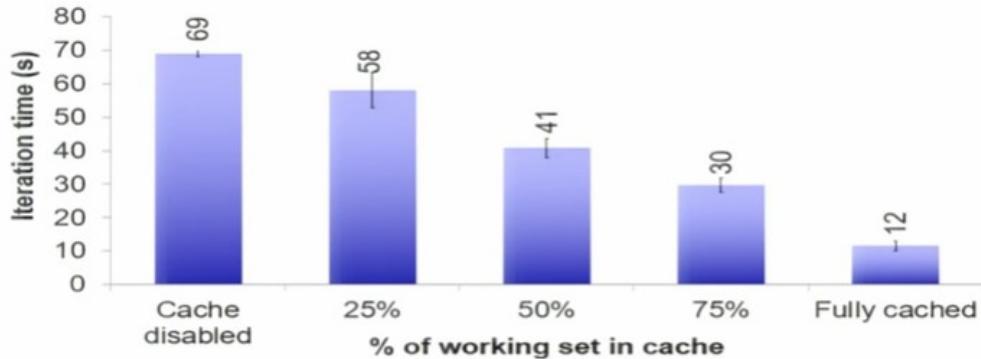
Fault Recovery Test



So, fault recovery, test if we see here is that so, the iteration time and the failure happens, is recovery is quite efficiently and done.

Refer slide time :(19:09)

Behavior with Less RAM



Now behavior with less RAM, can see that if it is fully cached, then it will iteration time will be quite less here in this case.

Refer slide time :(19:18)

First Stop: SparkContext

- Main entry point to Spark functionality
- Created for you in Spark shells as variable sc
- In standalone programs, you'd make your own (see later for details)

Now the question is what language you can use, obviously a scalar will be performing, better one let us see the two of Spark further operations. So, easiest way, to use the Spark is by the interpreter over, the shell and it runs in a local mode with, only one thread by default what control, can be with the master one and cluster. So, the first stop is through, the to the Spark context, is the main entry point to the Spark functionality which is created through the Spark shell.

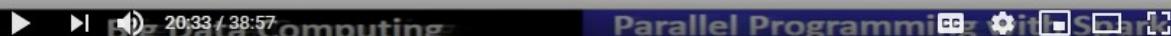
Refer slide time :(19:18)

Creating RDDs

```
# Turn a local collection into an RDD
sc.parallelize([1, 2, 3]) ✓

# Load text file from local FS, HDFS, or S3
sc.textFile("file.txt")
sc.textFile("directory/*.txt")
sc.textFile("hdfs://namenode:9000/path/file")

# Use any existing Hadoop InputFormat
sc.hadoopFile(keyClass, valClass, inputFmt,
conf)
```



And let us see how to create, this entire operation:

```
# Turn a local collection into an RDD
sc.parallelize([1, 2, 3])
# Load text file from local FS, HDFS, or S3
sc.textFile("file.txt")
sc.textFile("directory/*.txt")
sc.textFile("hdfs://namenode:9000/path/file")
# Use any existing Hadoop InputFormat
sc.hadoopFile(keyClass, valClass, inputFmt, conf)
```

Refer slide time :(20:37)

Basic Transformations

```
nums = sc.parallelize([1, 2, 3])
# Pass each element through a function
squares = nums.map(lambda x: x*x) # => {1, 4, 9}
# Keep elements passing a predicate
even = squares.filter(lambda x: x % 2 == 0) # => {4}
# Map each element to zero or more others
nums.flatMap(lambda x: range(0, x)) # => {0, 0, 1, 0, 1, 2}
```

The video player interface shows the following details:

- Progress: 23:41 / 38:57
- Course: Big Data Computing
- Section: Parallel Programming
- Lecture: Spark

A callout box highlights the `Range object (sequence of numbers 0, 1, ..., x-1)`.

The data and creating the RDDs now the basic transformations are shown over here:

```
nums = sc.parallelize([1, 2, 3])
# Pass each element through a function
squares = nums.map(lambda x: x*x) # => {1, 4, 9}
# Keep elements passing a predicate
even = squares.filter(lambda x: x % 2 == 0) # => {4}
# Map each element to zero or more others
nums.flatMap(lambda x: range(0, x)) # => {0, 0, 1, 0, 1, 2}
```

Refer slide time :(23:45)

Basic Actions

```
nums = sc.parallelize([1, 2, 3])
# Retrieve RDD contents as a local collection
nums.collect() # => [1, 2, 3]
# Return first K elements
nums.take(2) # => [1, 2]
# Count number of elements
nums.count() # => 3
# Merge elements with an associative function
nums.reduce(lambda x, y: x + y) # => 6
# Write elements to a text file
nums.saveAsTextFile("hdfs://file.txt")
```

▶ ▶! 25:28 / 38:57

Parallel Programming CC BY SA

And this is the basic transformation and now we are going to see some of the actions,

```
nums = sc.parallelize([1, 2, 3])
# Retrieve RDD contents as a local collection
nums.collect() # => [1, 2, 3]
# Return first K elements
nums.take(2) # => [1, 2]
# Count number of elements
nums.count() # => 3
# Merge elements with an associative function
nums.reduce(lambda x, y: x + y) # => 6
# Write elements to a text file
nums.saveAsTextFile("hdfs://file.txt")
```

Refer slide time :(25:29)

Working with Key-Value Pairs

- Spark's "distributed reduce" transformations act on RDDs of *key-value pairs*
- Python: pair = (a, b)
pair[0] # => a
pair[1] # => b
- Scala: val pair = (a, b)
pair._1 // => a
pair._2 // => b
- Java: Tuple2 pair = new Tuple2(a, b);
// class scala.Tuple2
pair._1 // => a
pair._2 // => b

And now let us see how to work with key value pairs, now is part distributed reduced transformation x, on RDDs of the key value pairs and key value pairs means here,

Python: pair = (a, b)

pair[0] # => a

pair[1] # => b

Scala: val pair = (a, b)

pair._1 // => a

pair._2 // => b

Java: Tuple2 pair = new Tuple2(a, b); // class scala.Tuple2

pair._1 // => a

pair._2 // => b

Refer slide time :(25:59)

Some Key-Value Operations

```
✓ pets = sc.parallelize([("cat", 1), ("dog", 1),
("cat", 2)])
✓ pets.reduceByKey(lambda x, y: x + y)
# => {(cat, 3), (dog, 1)}
✓ pets.groupByKey()
# => {(cat, Seq(1, 2)), (dog, Seq(1))}
✓ pets.sortByKey()
# => {(cat, 1), (cat, 2), (dog, 1)}
```

reduceByKey also automatically implements combiners on the map side

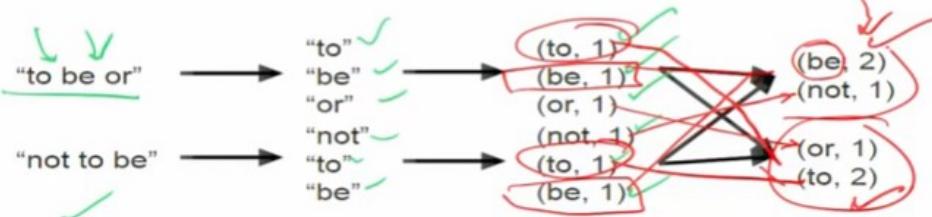
Now let us see some more key value pair operations, for example

```
pets = sc.parallelize([("cat", 1), ("dog", 1), ("cat", 2)])
pets.reduceByKey(lambda x, y: x + y)
# => {(cat, 3), (dog, 1)}
pets.groupByKey()
# => {(cat, Seq(1, 2)), (dog, Seq(1))}
pets.sortByKey()
# => {(cat, 1), (cat, 2), (dog, 1)}
reduceByKey also automatically implements combiners on the map side
```

Refer slide time :(27:31)

Example: Word Count

```
lines = sc.textFile("hamlet.txt")
counts = lines.flatMap(lambda line: line.split(" "))
    .map(lambda word: (word, 1))
    .reduceByKey(lambda x, y: x + y)
```



Now let us see the full use of full execution of, the word count program. So, what count program means

```
lines = sc.textFile("hamlet.txt")
counts = lines.flatMap(lambda line: line.split(" "))
    .map(lambda word: (word, 1))
    .reduceByKey(lambda x, y: x + y)
```

So, these words means map function, will what it will do it will for every word it will generate what comma 1. So, 2 it will generate 2 comma 1 B, B comma 1 or, or 1 not comma 1 and 2 comma 1 and B comma 1. So, after having generated this map function, now it will reduce by the key. So, reduced by the key in the sense it will say that for, a particular key it will do the summation. So, for the same keys for example B, is the key which is appearing in this particular, map output and here also this is the output, they will be collected back and that their values will be aggregated, according to the plus sign. So, 2 will be there similarly, 2 is also appearing two times and 2 will be gathered over here and their values also will be, added up and whereas all others are appearing only once. So, not will be collected over there and R will be connected over there. So, this is the reduced function, which will be applied.

Refer slide time :(29:41)

Other Key-Value Operations

- `val visits = sc.parallelize(List(("index.html", "1.2.3.4"), ("about.html", "3.4.5.6"), ("index.html", "1.3.3.1")))`
- `val pageNames = sc.parallelize(List(("index.html", "Home"), ("about.html", "About")))`
- `visits.join(pageNames)`
 `// ("index.html", ("1.2.3.4", "Home"))`
 `// ("index.html", ("1.3.3.1", "Home"))`
 `// ("about.html", ("3.4.5.6", "About"))`
- `visits.cogroup(pageNames)`
 `// ("index.html", (seq("1.2.3.4", "1.3.3.1"), seq("Home")))`
 `// ("about.html", (seq("3.4.5.6"), seq("About"))))`

There are other key value operations and now we can see about these operations, which basically are mentioned, were here like, like join and cogroup operations.

Refer slide time :(30:01)

Controlling the Level of Parallelism

- All the pair RDD operations take an optional second parameter for number of tasks

```
words.reduceByKey(lambda x, y: x + y, 5)
```

```
words.groupByKey(5)
```

```
visits.join(pageViews, 5)
```

Can also set `spark.default.parallelism` property

And now, we can all the pair's RDDs operations take the optional second parameter for the number of tasks and this we have shown over here.

Refer slide time :(30:13)

Using Local Variables

- External variables you use in a closure will automatically be shipped to the cluster:

```
query = raw_input("Enter a query:")
pages.filter(lambda x:
x.startswith(query)).count()
```

- Some caveats:

- Each task gets a new copy (updates aren't sent back)
- Variable must be Serializable (Java/Scala) or Pickle-able (Python)
- Don't use fields of an outer object (ships all of it!)

And external variables you can use in the closure will automatically be shipped to the to the cluster.

Refer slide time :(30:24)

Closure Mishap Example

```
class MyCoolRddApp {  
    val param = 3.14  
    val log = new Log(...)  
    ...  
  
    def work(rdd: RDD[Int]) {  
        rdd.map(x => x + param)  
        .reduce(...)  
    }  
}
```

So, all these are the implementation issues.

Refer slide time :(30:27)

Other RDD Operations

`sample()`: deterministically sample a subset
`union()`: merge two RDDs
`cartesian()`: cross product
`pipe()`: pass through external program

See Programming Guide for more:
www.spark-project.org/documentation.html

Refer slide time :(30:29)

More Details

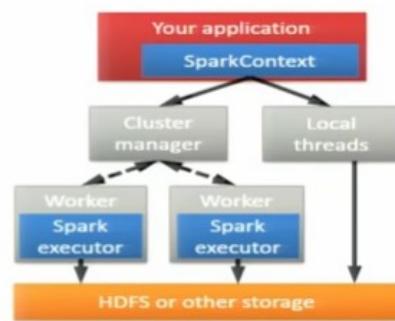
- Spark supports lots of other operations!
- Full programming guide: spark-project.org/documentation

So, for other RDD operations the programming guide is there.

Refer slide time :(30:27)

Software Components

- Spark runs as a library in your program
(one instance per app)
- Runs tasks locally or on a cluster
 - Standalone deploy cluster, Mesos or YARN
- Accesses storage via Hadoop InputFormat API
 - Can use HBase, HDFS, S3, ...

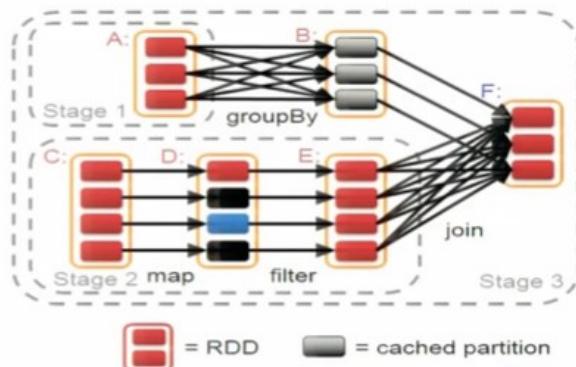


And that can be seen now let us see about, the job execution in his Spark it has various software components and when you say Spark context, will be creating it will be created, in the master job master and then it will create the worker threads, in the form of power context and then executors will execute them.

Refer slide time :(30:56)

Task Scheduler

- Supports general task graphs
- Pipelines functions where possible
- Cache-aware data reuse & locality
- Partitioning-aware to avoid shuffles



Similarly there will be at a scheduler Task, you will support the general tasks graphs, internally and pipelines will be also pipeline functions we are possible, they will be created by the task scheduler and cache aware data reuse and locality and partition aware things are done to avoid the shuffles.

Refer slide time :(31:21)

More Information

- Scala resources:
 - www.artima.com/scalazine/articles/steps.html
(First Steps to Scala)
 - www.artima.com/pins1ed (free book)
- Spark documentation: www.spark-project.org/documentation.html

So, more information about resources, on Scala is available.

Refer slide time :(31:28)

Hadoop Compatibility

- Spark can read/write to any storage system / format that has a plugin for Hadoop!
 - Examples: HDFS, S3, HBase, Cassandra, Avro, SequenceFile
 - Reuses Hadoop's InputFormat and OutputFormat APIs
- APIs like `SparkContext.textFile` support filesystems, while `SparkContext.hadoopRDD` allows passing any Hadoop JobConf to configure an input source

And let the Hadoop is Spark can read and write to any storage system format that has plug into the Hadoop. And API is like a Spark on text file supports, while the Spark context Hadoop RDD allows pass any Hadoop job, to configure the input file.

Refer slide time :(31:45)

Create a SparkContext

```
Scala
import spark.SparkContext
import spark.SparkContext._

val sc = new SparkContext("masterUrl", "name", "sparkHome", Seq("app.jar"))

Java
import spark.api.java.JavaSparkContext;
JavaSparkContext sc = new JavaSparkContext(
    "masterUrl", "name", "sparkHome", new String[] {"app.jar"});

Python
from pyspark import SparkContext
sc = SparkContext("masterUrl", "name", "sparkHome", ["library.py"])
```

And this is to create the Spark context, now Spark context when we say it has different arguments. So, the first one is called, ‘Master URL’, is nothing but to create the URL or a local oblique in local node n. so, next one next thing is the application name and then, Spark will now install the path on the cluster, with the name is Spark home and finally the list of jar files also has to be given in the Spark context automatically it creates a Spark context using the Java jar file.

Refer slide time :(32:25)

Complete App: Scala

```
import spark.SparkContext
import spark.SparkContext._

object WordCount {
  def main(args: Array[String]) {
    val sc = new SparkContext("local", "WordCount", args(0), Seq(args(1)))
    val lines = sc.textFile(args(2))
    lines.flatMap(_.split(" "))
      .map(word => (word, 1))
      .reduceByKey(_ + _)
      .saveAsTextFile(args(3))
  }
}
```

So, now this is the complete word count program, which we have discussed, shown over here as the local and what count is the name of the program arguments are there and this argument number one.

Refer slide time :(32:40)

Complete App: Python

```
import sys
from pyspark import SparkContext

if __name__ == "__main__":
    sc = SparkContext("local", "WordCount", sys.argv[0], None)
    lines = sc.textFile(sys.argv[1])

    lines.flatMap(lambda s: s.split(" ")) \
      .map(lambda word: (word, 1)) \
      .reduceByKey(lambda x, y: x + y) \
      .saveAsTextFile(sys.argv[2])
```

And

Refer slide time :(32:41)

Now let us see an example

Refer slide time :(32:43)

Example: PageRank

Of a PageRank algorithm how we can execute in the

Refer slide time :(32:48)

Why PageRank?

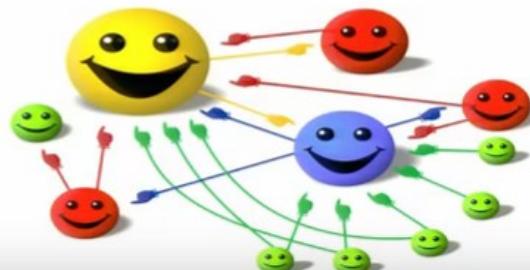
- Good example of a more complex algorithm
 - Multiple stages of map & reduce
- Benefits from Spark's in-memory caching
 - Multiple iterations over the same data

Refer slide time :(32:48)

Spark system.

Basic Idea

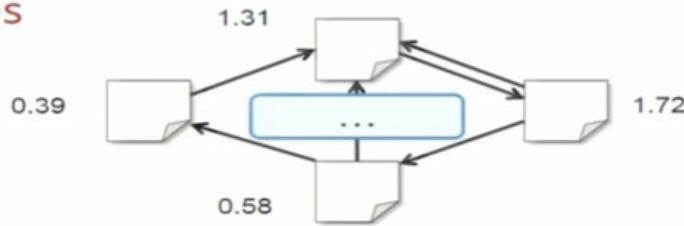
- Give pages ranks (scores) based on links to them
 - Links from many pages → high rank
 - Link from a high-rank page → high rank



Refer slide time :(32:51)

Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $\text{rank}_p / |\text{neighbors}_p|$ to its neighbors
3. Set each page's rank to $0.15 + 0.85 \times \text{contribs}$



So, let us start let us see the algorithm,

1. Start at each page the rank of one.
2. On each iteration, have page p contribute $\text{rank}_p / |\text{neighbors}_p|$ to its neighbors
3. Set each page's rank to $0.15 + 0.85 \times \text{contribs}$

So, here we can see that, here we can see that, this particular node, is basically this particular node has, two links out so, the contribution of one will be divided equally 0.5 and 0.5 similarly this page also has two outgoing lines links so, it will be also having a contributions of 0.5 and this will have only one. So, it will be having the contribution of entire 1 and this page is also having 1. So, it will be having contribution of 1, now let us see that this particular page, we can see that it is now incoming. So, it is incoming with 0.5 so, now we have to calculate according to this $0.15 + 0.85 \times 0.5$. So, that will be the new page rank of this and as far as this particular page, is concerned the incoming is once $0.85 \times 1 + 0.15$. so, the page rank will become 1 in that case so, 1 will not be changed so, this will be the page rank 1 and how about this so, here 1 2 3 different links are coming so, with a with this, this link will be 1 plus this will be 1 and this will be 0.5 and this also will be the this also will be 0.5 so, this become $2 \times 0.85 + 0.15$. And this also I will have the same so, let us see that after doing this particular iterations. So, the PageRank will now be changed, into as we have seen that it will be 1 it was pointed point, five eight point five eight point eight five one point eight five and this particular, iterations will continue and here it will stop ,why because it is not going to change.

Refer slide time :(35:52)

Scala Implementation

```
val links = // RDD of (url, neighbors) pairs →
var ranks = // RDD of (url, rank) pairs ←

for (i <- 1 to ITERATIONS) { ✓
  val contribs = links.join(ranks).flatMap {
    case (url, (links, rank)) =>
      links.map(dest => (dest, rank/links.size))
  }
  ranks = contribs.reduceByKey(_ + _)
    .mapValues(0.15 + 0.85 * _)
}

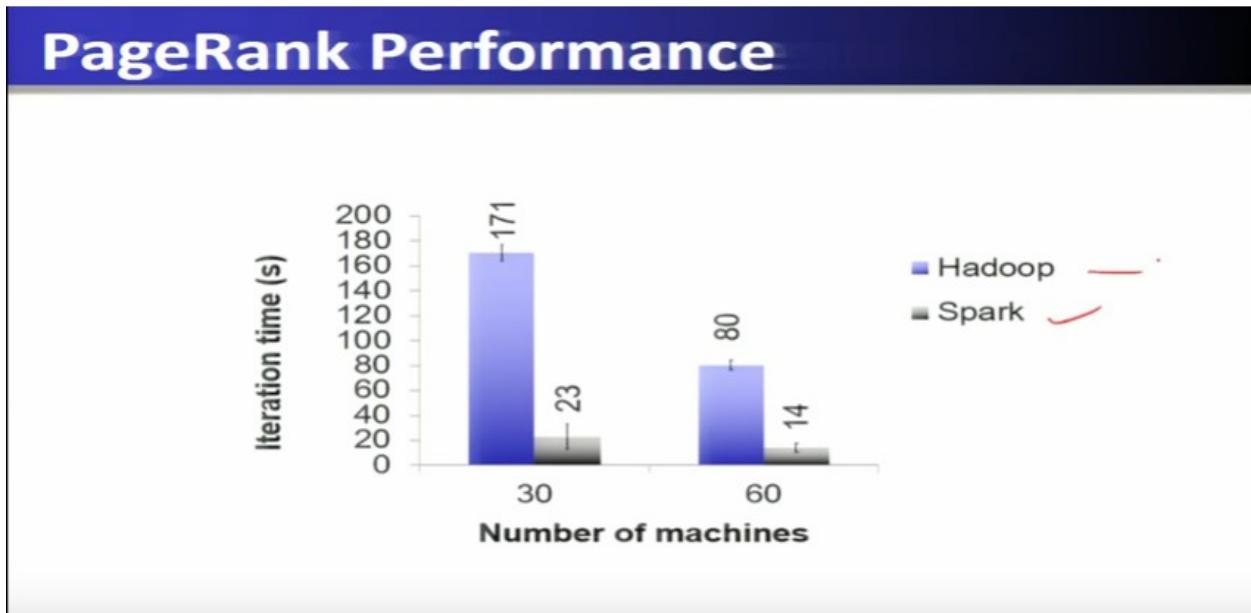
ranks.saveAsTextFile(...)
```

Further let us see how this entire PageRank algorithm can be implemented using scalar.

```
val links = // RDD of (url, neighbors) pairs
var ranks = // RDD of (url, rank) pairs
for (i <- 1 to ITERATIONS) {
  val contribs = links.join(ranks).flatMap {
    case (url, (links, rank)) =>
      links.map(dest => (dest, rank/links.size))
  }
  ranks = contribs.reduceByKey(_ + _)
    .mapValues(0.15 + 0.85 * _)
}

ranks.saveAsTextFile(...)
```

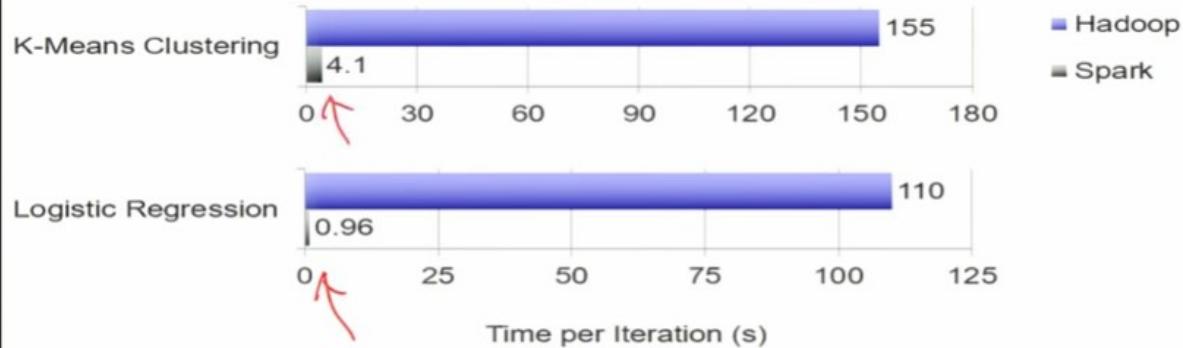
Refer slide time :(37:23)



So, we see that the page rank performance, with the Spark it is very efficient and very fast compared to The Hadoop here, if the number of machines are 16 and the iteration time is very less,

Refer slide time :(37:36)

Other Iterative Algorithms



There are other iterative algorithms which are implemented, in the SPARC such as K-means clustering logistic regression in all the cases you see that, Spark is very very efficient compared to the Hadoop.
Refer slide time :(37:53)

References

- **Parallel Programming With Spark**
Matei Zaharia, UC Berkeley
- **Parallel Programming with Spark**
Qin Liu, The Chinese University of Hong Kong
- <http://spark.apache.org/>
- <https://amplab.cs.berkeley.edu/>

In the iterations so, these are some of the references, for the Spark to be.

Refer slide time :(37:59)

Conclusion

- Spark offers a rich API to make data analytics *fast*: both fast to write and fast to run
- Achieves 100x speedups in real applications
- Growing community with 14 companies contributing
- Details, tutorials, videos: www.spark-project.org



So, conclusion is, Spark offers a rich API is, to make the data analytics fast. Both fast to write and fast to run. It achieves hundred times speed-up, in the real applications, the growing community with 14 companies are contributing to it and details tutorials are available in the website, www.spark-project.org. Thank you.

Lecture -10

Introduction to Spark

Refer slide time: (0:13)

Introduction to Spark



Dr. Rajiv Misra
Dept. of Computer Science & Engg.
Indian Institute of Technology Patna
rajivm@iitp.ac.in

Introduction to Spark.

Refer slide time: (0:15)

Preface

Content of this Lecture:

- In this lecture, we will discuss the '**framework of spark**', Resilient Distributed Datasets (RDDs) and also discuss Spark execution.

Preface: Content of this lecture: This lecture, we will discuss the 'framework of Spark'. Resilient distributed data sets and also, we will discuss the Spark execution.

Refer slide time: (0:30)

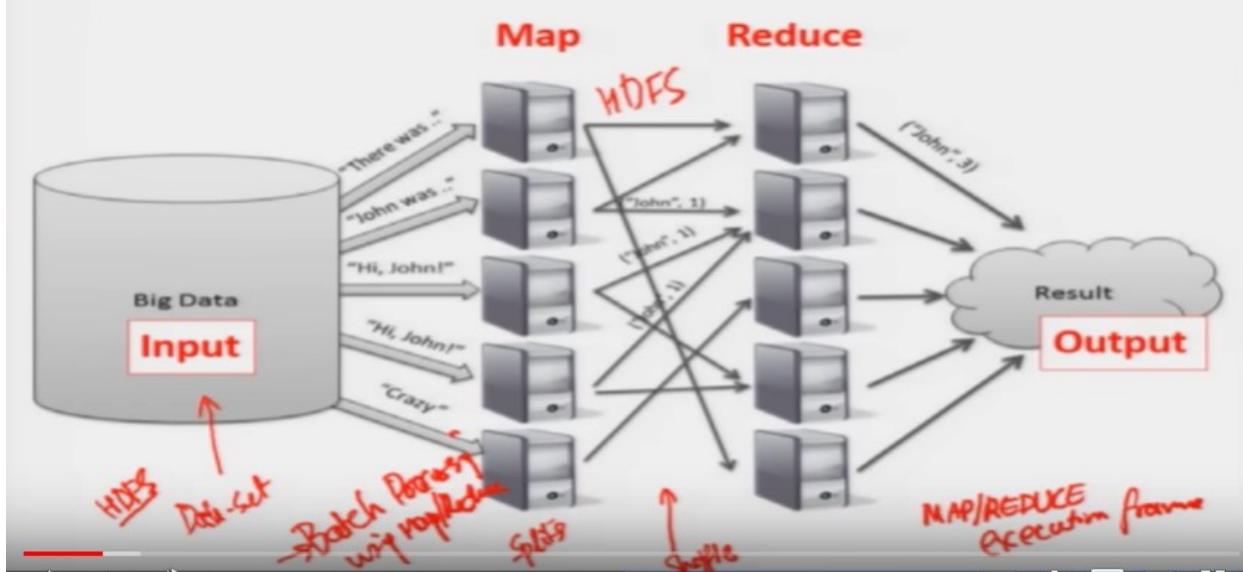
Need of Spark

- **Apache Spark** is a big data analytics framework that was originally developed at the University of California, Berkeley's AMPLab, in 2012. Since then, it has gained a lot of attraction both in academia and in industry.
- It is an another system for big data analytics
- **Isn't MapReduce good enough?**
 - Simplifies batch processing on large commodity clusters

The need of his Spark. Apache Spark is a big data, analytics framework that was originally developed, at University of California, Berkeley, at AMP Lab, in 2012. Since then, it has gained a lot of attraction, both in the academia and in the industry. It is, an another system for big data analytics. Now, before this Spark, the Map Reduce was already in use. Now, why is not the Map Reduce good enough? That we you are going to explore, to understand, the need of the Spark system, before that we have to understand that, Map Reduce simplifies, the batch processing on a large commodity clusters.

Refer slide time: (1:23)

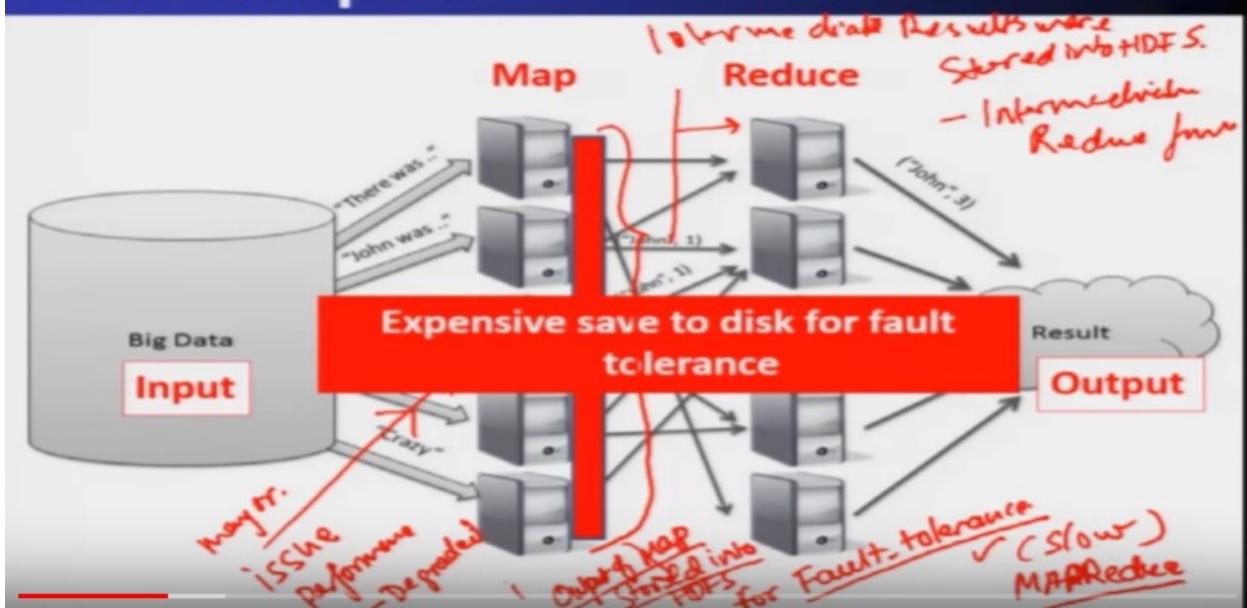
Need of Spark



So, in that process, the data file or a dataset, which was the input? which was input, through the HDFS system, uses, the map function, to be spitted into across different, splits and then the map function was applied on to this particular splits, which in turn will generate, the intermediate values, which is in the form of shuffle and is stored in HDFS and then passed on, to the reduced function giving the output, so this is, the scenario of a Map Reduce, execution framework, as you have seen the data set, is now stored in the HDFS file system. Hence, this particular computation is done for the batch processing system. So we have seen that, this particular batch processing, was done using the Map Reduce system and that was in use earlier, before the development of a Spark.

Refer slide time: (3:04)

Need of Spark



Now let us see, why this particular model, of Map Reduce is not good enough to be there? Now, as you have seen, in the previous slide that, the output of the Map function, will be stored in HDFS and this will ensure the fault tolerance. So the output of, map function, was stored into HDFS, file system and this ensures of the fault tolerance. So in between, if there is a failure, when it is stored in the HDFS file system, still the data is not lost and is completely saved, that is why? It is ensuring the fault tolerance and because of that fault tolerance, the intermediate values or intermediate results, out of the map functions were stored into HDFS file. Now, this particular intermediate results will be now further fed on to the reduced function. So that was, the typical Map Reduce, execution pipeline. Now, what is the issue? The problem is, mentioned over here, that the major issue is the performance. so that means, the performance is degraded and it has become expensive, to save to the disk for this particular activity of fault tolerance, hence this particular Map Reduce, framework becomes quite slow, to some of the applications, which are interactive and required to be processed in the real time, hence this particular framework, is not very useful, for certain applications.

Refer slide time: (5:50)

Need of Spark

- MapReduce can be expensive for some applications e.g.,
 - Iterative ✓ ✓ ✓
 - Interactive ✓ ✓ ✓
- Lacks efficient data sharing ✓ HDFS
- Specialized frameworks did evolve for different programming models
 - Bulk Synchronous Processing (Pregel) ✓ Graph Processing ✓ BSP
 - Iterative MapReduce (Hadoop) ✓

Therefore, let us summarize that, Map Reduce can be expensive, for some applications and for example, interactive and iterative application. They are very expensive and also, this particular framework, that is the Map Reduce framework, lacks efficient data sharing, across the map and reduce phase of operation or iterations. So lacks efficient data sharing in the sense, the data which is, the intermediate form of Map Reduce is, map is stored in the file system, HDFS file system. Hence, it is not shared, in an efficient manner. Hence, this particular sharing the data across map and reduce phase, has to be through the disk and which is a slow operation, not an efficient operation. Hence, this statement which says that, Map Reduce, lacks the efficient data sharing, into the system .so due to these, drawbacks, there were several specialized framework did evolve, for different programming models, such as, so the pregel system was, there for graph processing, using bulk synchronous processing BSP. and then, there is another framework, which is called, 'Iterative Map Reduce' was also made and they allowed, a different framework some a specialized framework, for different applications to be supported.

So the drawback of Map Reduce has, given the way to several frameworks and they were involved a different programming models, so bulk synchronous, parallel bulk synchronous parallel processing framework, is about there is a synchronization barrier, so the processes at different speed the joints at the synchronization barrier and then they will proceed further on, so this is called, 'BSP model'. so this BSP model also, allows a fast processing for the typical graph application, so graph processing is done using BSP, to make it more faster. Because, the Map Reduce does not support the graph processing within it. so grab processing framework requires, that the data which is being, taken up by the neighboring nodes, then basically these nodes ,the neighboring node will collect the data, will gather the data and then perform the computation and then again scatter, these data to the other points. so this particular operation, the communication, computation and then communication, is to be completed as one step, that is the lockstep and hence the bulk synchronous processing, comes into the and effect where this all three operations, all three actions, are to be performed and then only the next step will takes place using bulk synchronous processing. Hence, this kind of paradigm, that is called,' Bulk Synchronous Processing framework' ,is useful for the graph processing, that we will see that, how this particular different programming

paradigm, such as bulk synchronous processing and iterative Map Reduce, for the iterative applications, for example, the iterations of a Map Reduce, is basically you can see, in the machine learning algorithms. so these different frameworks were evolved, out of this Map Reduce drawbacks and they, they existed, over the several period, over the period of time, to fill up this particular gap. Now, seeing all these scenarios and to provide the data sharing and to support the applications such as interactive and iterative applications, there was a need for the SPARK system.

Refer slide time: (10:38)

Solution: Resilient Distributed Datasets (RDDs)

Resilient Distributed Datasets (RDDs)

- Immutable, partitioned collection of records
- Built through coarse grained transformations (map, join ...)
- Can be cached for efficient reuse

Cannot change

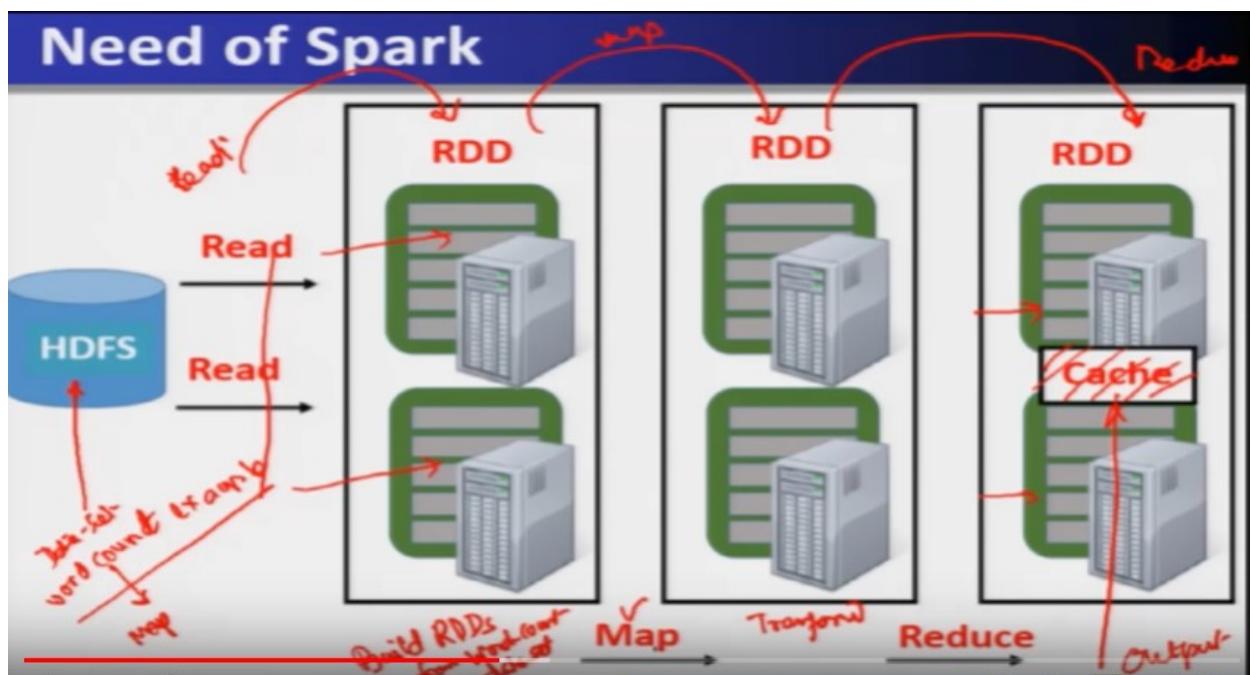
SPARK Abstract data type - RDD

- RDD can be built - Transformations
- RDD immutable, partitioned collection
- RDD can't change Records
- RDD can be cache in iterative / interactive applications

So the solution: which is Spark has given, is in the form of an abstract data type, which is called, 'Resilient Distributed Data Set'. So the SPARK provides, a new way, of supporting the abstract data type, which is called resilient distributed data set or in short it is RDDs. Now, we will see in this part of the discussion, how this RDDs are going to solve ,the drawbacks of the batch processing Map Reduce, into an more efficient data sharing and also going to be supporting, the iterative and interactive applications. Now, this particular RDDs, are resilient distributed data sets, they are immutable, immutable means, we cannot change. It cannot be changed, so that means once an RDD is formed, so it will be an immutable. Now, in this particular way, this immutable resilient distributed data set can be partitioned in a various ways, across different cluster nodes. So partition collection of Records, so partitioning can happen, in s for example, if this is a cluster, of machines which are connected, with each other. So the RDDs can be partitioned and stored, at different places, at different segments. Hence, the immutable partition collection of records is possible and in this particular scenario, that is called, 'RDDs'. Now, another thing is, once an RDD is formed, then it will be formed using, it will be built, RDDs will be built, through a course gain transformations, such as, map, join and so on. Now, these RDDs can be cached for efficient reuse, so that we are going to see that, lot of new operations can be performed on it, so again let us summarize, that the Spark has, given a solution, in the form of an abstract data type ,which is called as a, 'RDD.' and our

RDD can be built, using the transformations and also can be changed, can be changed into another form, that is RDD can become another, RDD by making various transformations, such as map, join and so on. That we will see in due course of action, these RDDs are, are immutable, partition collection of record. Means that, once an RDD is formed, so as an immutable, immutable means, we cannot change, this entire collection of records, can be stored and in a convenient manner onto the, onto the cluster system. Hence, this is an immutable partition collection of the record, these RDDs can be cached, can be cached in memory, for efficient reuse. So as we have seen that, Map Reduce lacks, this data shearing and now using the RDDs, a Spark will provide, the efficient, sharing of data in the form of, all RDDs.

Refer slide time: (15:02)



Now let us see, through an example of a word count. Word count example, to understand the need of Spark. So in the word count application, the data set, is installed through the HDFS file system and is being read, so after reading this particular data, set from the file system, this particular reading operation, will build the RDDs. and which is shown here, in this block number 1. So that means once, the data is read, it will build, the RDDs from the word-count data set. Once these RDDs are built, then they can be stored at different places, so it's an immutable partitioned collection and now various operations we can perform. So we know that, these RDDs, we can perform various transformations and first transformation which we are going to perform on these RDDs, which are, which is called a, 'Map Function'. Map function for, the word count program, is being applied on different RDDs, which are restored. So after applying the map function, this particular output, will be stored in memory and then again, the reduced function will be applied, on these RDDs, which is the output? Or which is the transformations? Which is the transform or RDDs, out of the map function? Again the reduce function will apply. And the data and the result of this reduce, will remain in cache. so that, it can be, used up by different application .so you can see that, this particular transformation, which is changing the RDDs from one form, to another form

that means after reading, from the file, it will become an RDD and after applying the map function, it will change to another RDD and map function and after applying the reduce function. it will change to, another form and the final output, will be remained in the cache memory. Output will remain in the cache, so that, whatever application requires, this output can be used up, so this particular pipeline, which we have shown, is quite, easily, understandable and is convenient to manage and part and to store, in the partition, collection manner, in this cluster computing system.

Refer slide time: (18:11)

Solution: Resilient Distributed Datasets (RDDs)

Resilient Distributed Datasets (RDDs)

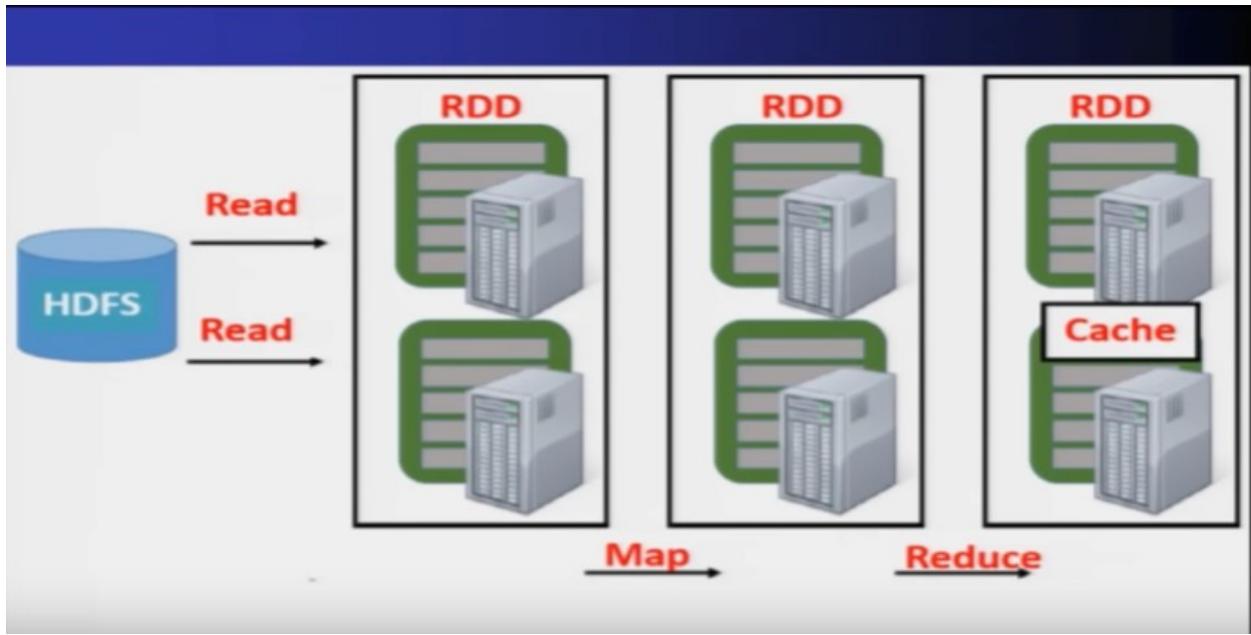
- Immutable, partitioned collection of records
- Built through coarse grained transformations (map, join ...)

Fault Recovery? ↗ *concept used by spark for fault-tolerance is called lineage!*

- Lineage! ↗
- Log the coarse grained operation applied to a partitioned dataset
- Simply recompute the lost partition if failure occurs!
- No cost if no failure ↗

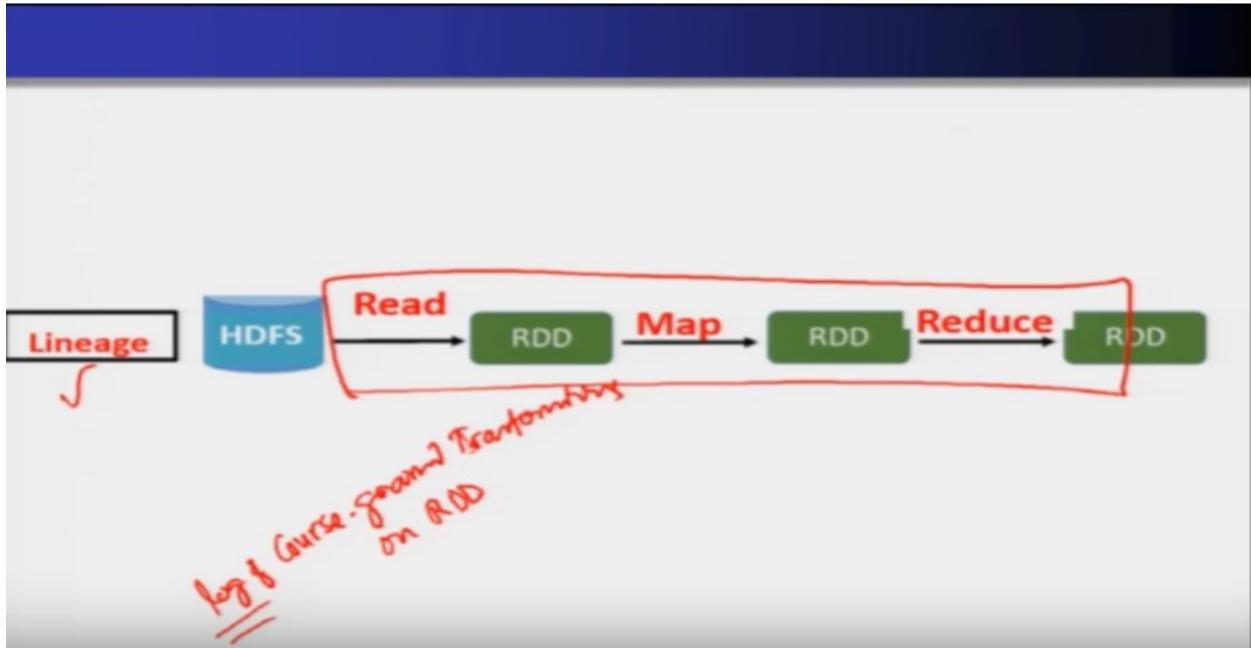
So, we have seen that, this RDDs has simplified this particular task and has also made this operation efficient. Hence, RDDs is an immutable, partition collection of the records. And they are built through the coarse grained transformation that we have seen in the previous example. Now, another question is, since the Map Reduce was storing ,the intermediate results of a map, before it is being used in the reduced function, into an SDFS file system, for ensuring the fault tolerance .now since, by produced since the SPARK is not using, this intermediate results storage, through the HDFS rather, it will be in memory storage, so there will be how this Spark ensures the fault tolerance that we have to understand now, the concept which Spark uses, for fault tolerance is called, 'Lineage'. So this park uses the lineage to achieve the fault tolerance and that we have to understand now, so what Spark does is? It locks, the coarse-grained operations, which are applied, to the partition data set. meaning to say that, all the operation like, reading of a file and that becomes an RDD and then making a transformation on an RDD using map function and then again another transformation, of RDDs using reduced function, join function and so on. All these operations they form, the course gained operations and they are to be logged into a file, of before applying it. So if the data is, so basically ,if the data is, lost or if the, if the system, crashes the node crashes, they simply recomputed, these lost partition and whenever there is a failure, if there is no failure, obviously no extra cost ,is required in this process.

Refer slide time: (20:37)



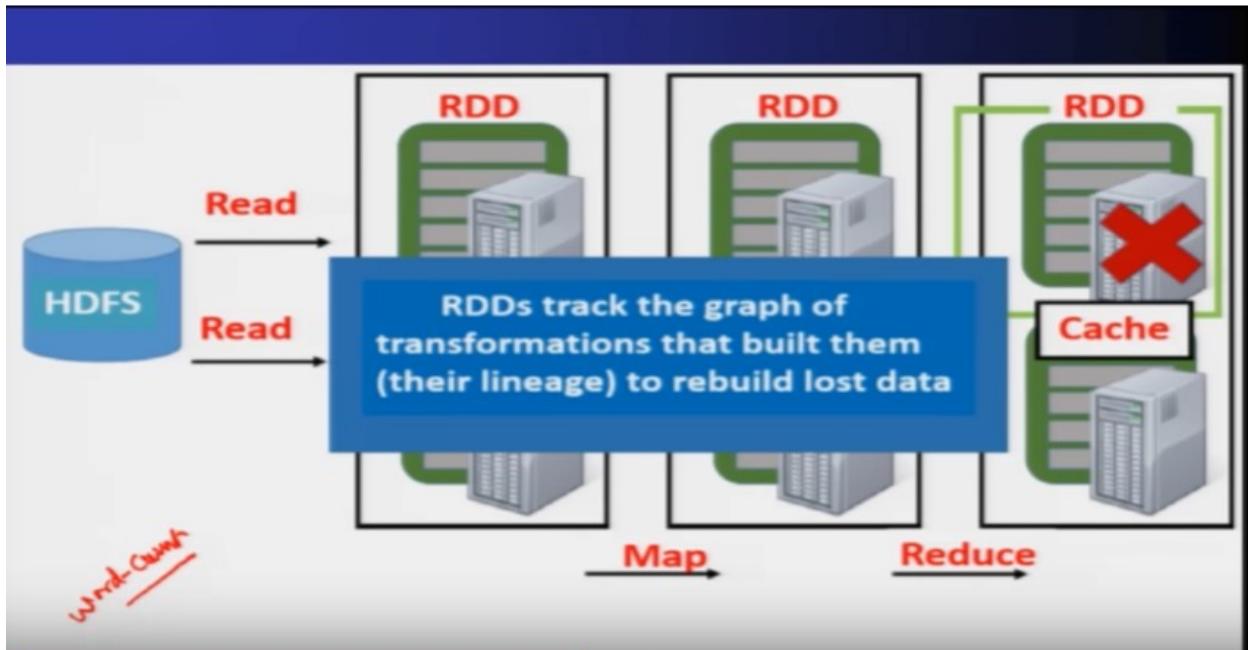
Let us see this, through an example.

Refer slide time: (20:40)



So again, let us explain that, lineage is in the form of the course grained, you said, it is a log of a coarse grained operation. Now this particular, lineage will, keep a record of all these operations, coarse-grained operations, which are applied and that will be kept in a log file. Now we will see, how using this lineage or a log files, the fault tolerance can be achieved.

Refer slide time: (21:24)



Let us see, through this particular example. That let us see, that the word count example, which we have seen in the last slide. Now the, the same word count example, we have to, we have, we have to see that, these RDDs, will keep track of ,oddities will keep track, the graph of transformation, that build them, their lineage to rebuild lost data. So the, so there will be a log of all the coarse grained operation, which are performed and which has, built these RDDs transformations. and this is called, 'Lineage'. let us see, what happens is for example, after reading this particular RDD will be formed after the read operations, on the data set and then, on this particular data, this RDD we have performed, the map operation, RDD transformed RDDs and this transformed RDD again, is now applied with the reduced function, to make this particular RDD and is stored in the cache. Now, consider that if this particular node, which has stored this transformed RDD if it is filled, obviously it has to trace back, to the previous RDD and consult, this lineage, which will tell that, this is an output of the map function, this is an RDD transformed and RDD, this RDD when we apply, the reduced function, it will recreate, the same RDD which is lost. So let us see, what is written? What we have just seen?

Refer slide time: (23:17)

Solution: Resilient Distributed Datasets (RDDs)

Resilient Distributed Datasets (RDDs)

- Immutable, partitioned collection of records
- Built through coarse grained transformations (map, join ...)

Fault Recovery?

- Lineage!
- Log the coarse grained operation applied to a partitioned dataset
- Simply recompute the lost partition if failure occurs!
- No cost if no failure

So we have to simply recompute the last partition, whenever there will be a failure, how we have to trace back and apply the same transformation again, on RDD and we can recomputed, that the, the RDD which is lost in the partition due to the failures. So now, using lineage, concept we have seen that the fault tolerance, is achieved in a Spark system.

Refer slide time: (23:48)

What can you do with Spark?

- **RDD operations**
 - Transformations e.g., filter, join, map, group-by ...
 - Actions e.g., count, print ...
- **Control**
 - **Partitioning:** Spark also gives you control over how you can partition your RDDs.
 - **Persistence:** Allows you to choose whether you want to persist RDD onto disk or not.

Now we will see that, what more we can do here in the Spark? So RDDs transfer, which RDDs provide various operations and all these operations are divided into two different categories, the first category is

called, ‘Transformations’. Which we can apply, as an RDD operation. second is called, ‘Actions’, which we can perform using RDDs, operations so as far as the transformations, which RDD supports is in the form of filter, join, map, group by all these are different transformations, which RDD supports, in the Spark system .Now, another set of, operation which RDD supports is called, ‘Actions’. so actions, in the sense the output of some, some operations, is whenever there then it is called, ‘Action’. For example, count, print and so on. Now, then another thing which, Spark can provide is called, ‘Control Operations’, to the programmer level.

So there, are two interesting control, which is being provided by the Spark, to the programmers. The first is called, ‘partitioning’. So, the Spark gives the control or how you can partition your RDDs, across different cluster systems. and second one is called the, ‘Persistence’. Persistence allows you to choose, whether you want to persist RDDs on to the disk or not. So by default, it is not persisted, but if you allow, if you choose this persistent, RDDs then the, RDDs I have to be stored in HDFS. Hence, the persistent and partitioning both controls are, given to the, to the programmer the user in buys by the Spark system.

Refer slide time: (25:36)

Spark Applications

- i. Twitter spam classification
- ii. EM algorithm for traffic prediction
- iii. K-means clustering
- iv. Alternating Least Squares matrix factorization
- v. In-memory OLAP aggregation on Hive data
- vi. SQL on Spark

There are various other, Spark applications, where Spark can be used first, these applications are such as, Twitter respond classification, algorithm for traffic prediction, k-means clustering algorithms, alternating least square matrix factorization, in memory OLAP aggregation on his, pond hive data and SQL on Spark.

Refer slide time: (26:02)

Reading Material

- Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica

"Spark: Cluster Computing with Working Sets"

- Matei Zaharia, Mosharaf Chowdhury et al.

"Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing"

<https://spark.apache.org/>

These are some of the applications and these are the reference, material for further studies, on the Spark system. That we have, that is available on HTTPS, Spark dot Apache dot org.

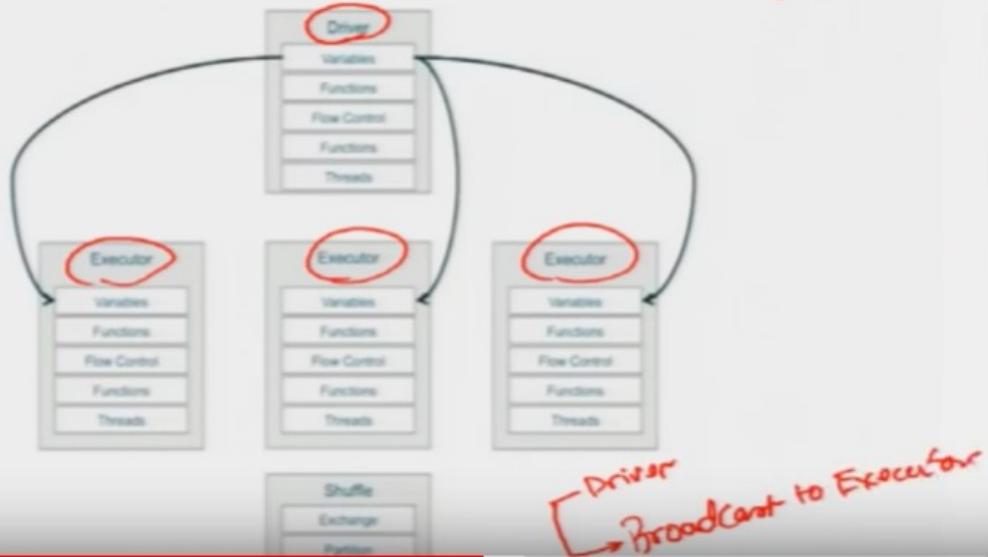
Refer slide time: (26:18)

Spark Execution

Now we will see, about the Spark execution.

Refer slide time: (26:21)

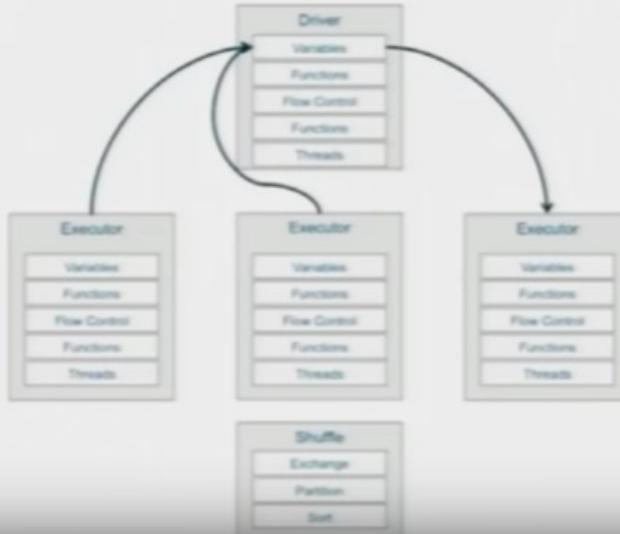
Distributed Programming (Broadcast)



So a Spark execution, is done, in the form of distributed programming, that is, the first operation is called, 'Broadcast'. So there is a, there are three different entities, one is called the, 'Driver Entity', of the Spark, the other entity is called, 'Executors'. Which are there on different nodes, data nodes and then there is a shuffle operation. So let us see the, first operation is called, 'Broadcast'. So the driver will broadcast, these different commands, to the different executors, that is called a, 'Broadcast Operation'. We will broadcast, to the executors.

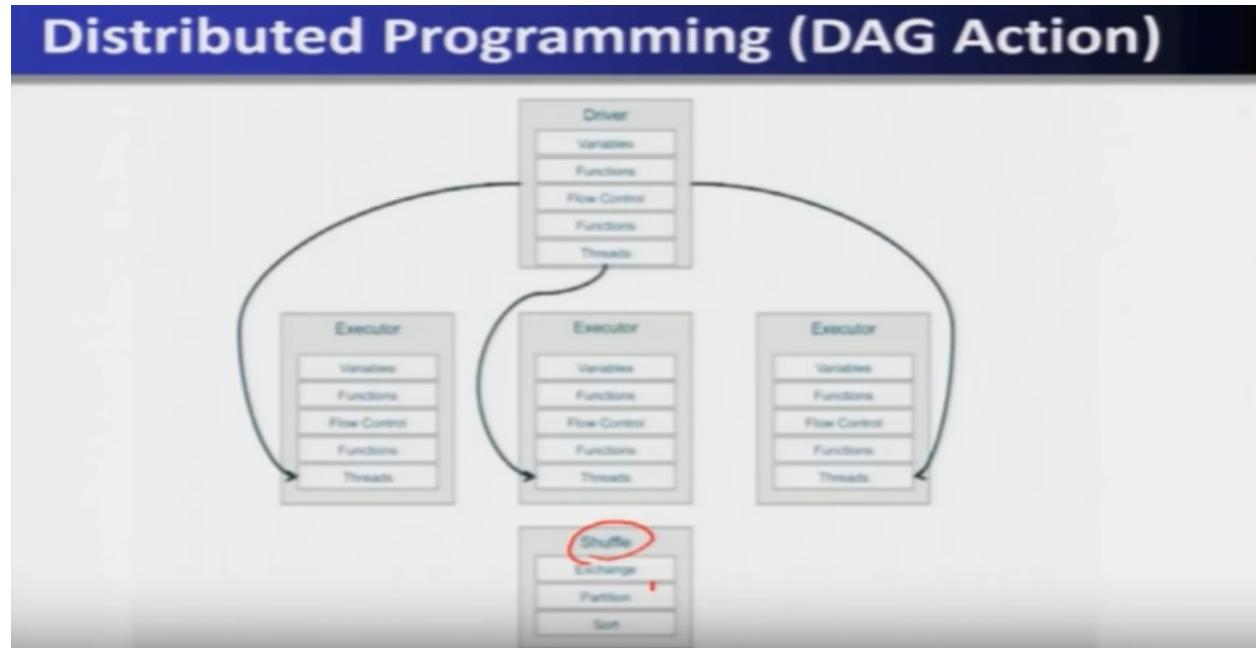
Refer slide time: (27:21)

Distributed Programming (Take)



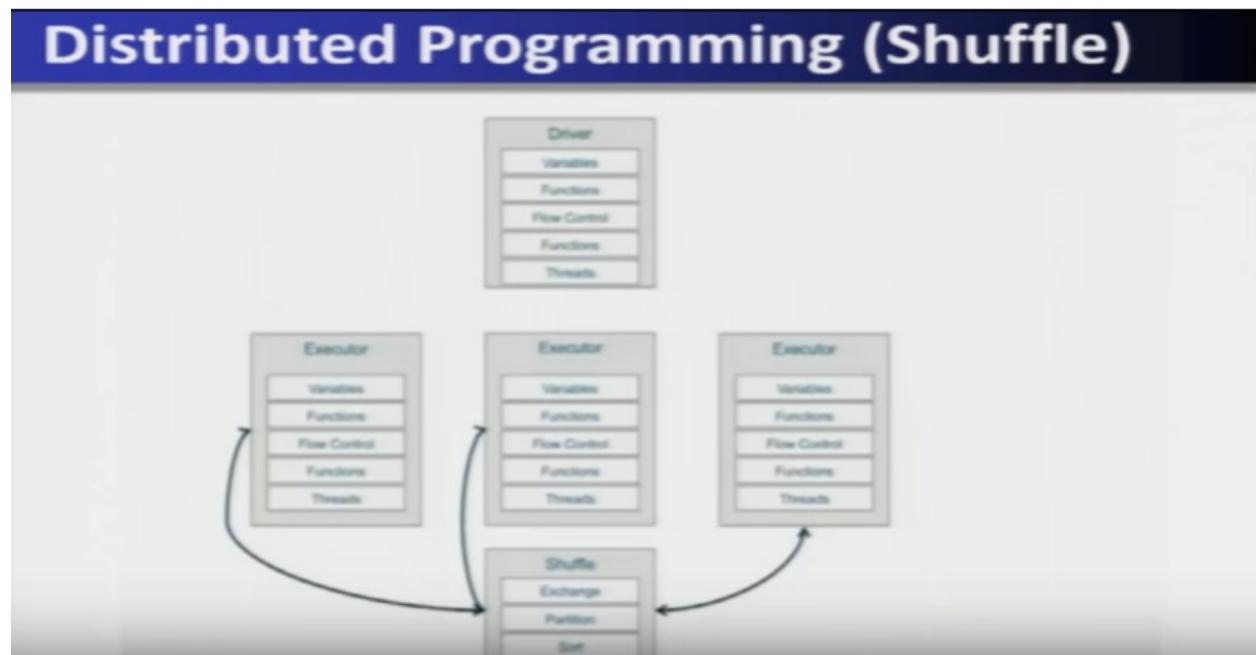
Now, these executors will execute and give the result back to the drivers.

Refer slide time: (27:22)



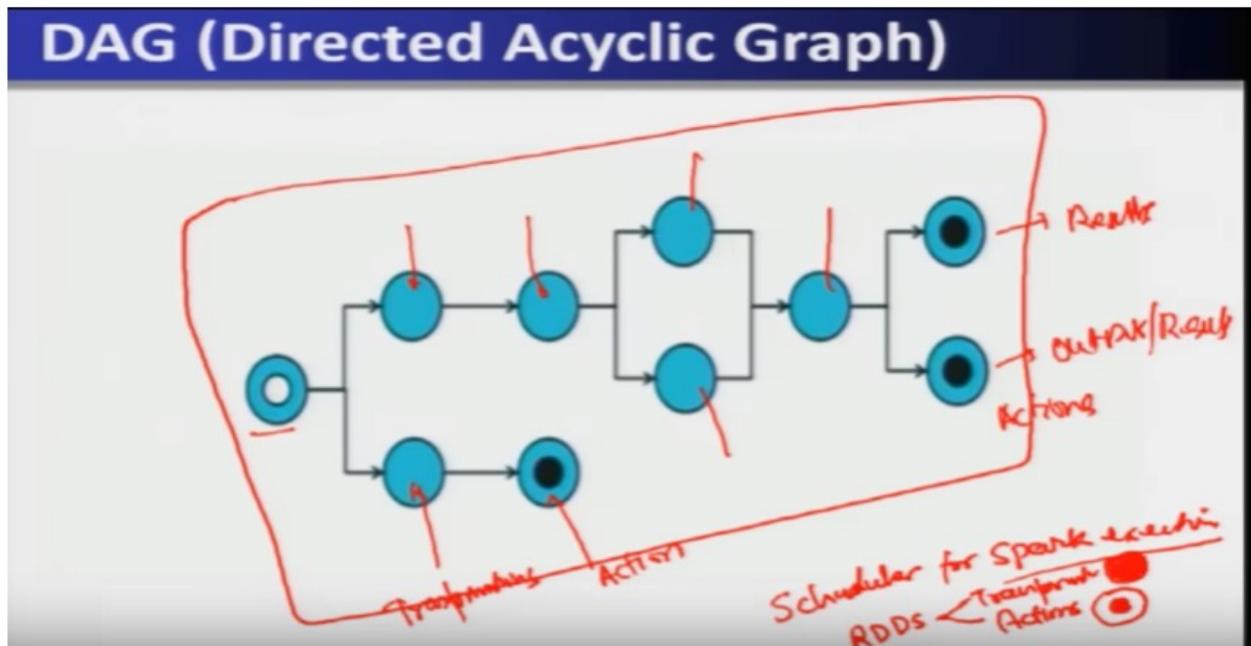
And then again, the driver will give further operations or the functions.

Refer slide time: (27:34)



And these, particular functions are used by the shuffle and again given back to the executors.

Refer slide time: (27:42)



This all will be performed, the entire task operations, will be performed using, the directed acyclic graph. So directed acyclic graph is the Schuler, for the SPARC execution. So SPARC execution, as we have seen that RDDs, can be executed using, two operations. One is called, 'Transformations'. The other one is called, 'Actions'. So, in this particular RDD, we have shown you the actions, in the form of, the dotted circle and transformation in the form of a gray circle. so you can see here, that, this shows that, this is an RDD and from this, this particular RDD, is obtained using the transformations and further, this particular RDD is now giving performing the action part and this is also a transformation, transformation, transformation and these are the actions. So these actions will give the output or the results, of the execution. This complete schedule is, available at the driver and using this particular scheduler the driver in turn will supply, the different actions, different operations on RDDs, in the form of transformations and actions as, it is defined in the directed acyclic graph scheduler.

Refer slide time: (29:52)

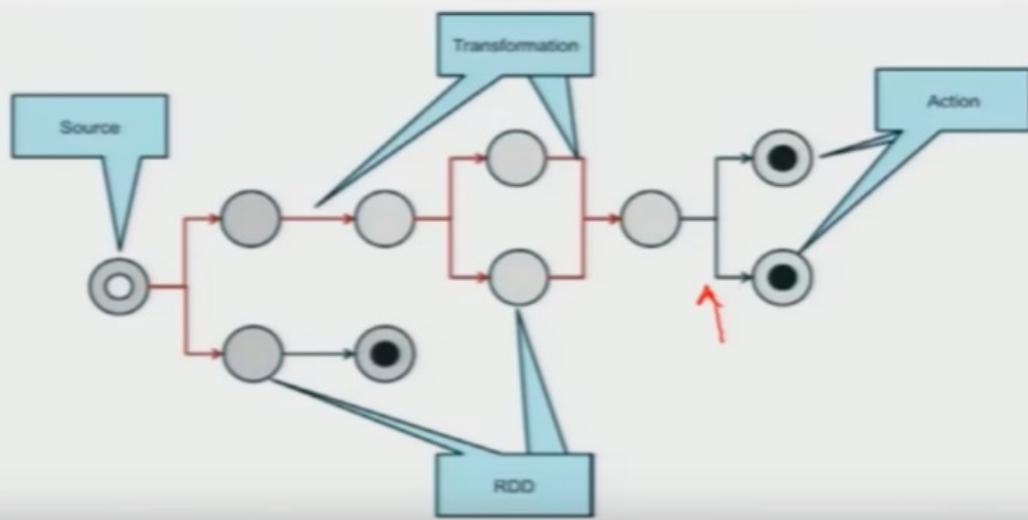
DAG (Directed Acyclic Graph)

- **Action** ✓
 - Count ✓
 - Take ✓
 - Foreach ✓
- **Transformation** ✓
 - Map ✓
 - ReduceByKey ✓
 - GroupByKey ✓
 - JoinByKey ✓

Therefore, this directed acyclic graph or a DAG, will take either the actions or transformations. So actions include the, Count, Take, Foreach and the transformation involves, the map, reduced by key, joined by key and group by key.

Refer slide time: (30:16)

DAG (Directed Acyclic Graph)



So these are all, a diagram I have already explained, that these are all RDDs and these are all transformations, the arrows are transformations, from one RDDs to another RDDs and if this is an, action. On these RDDs, it will be performed on the action.

Refer slide time: (30:44)

Flume Java

1. val conf = new SparkConf().setMaster("local[2]")
2. val sc = new SparkContext(conf) ✓
3. val lines = sc.textFile(path, 2) ✓
4. val words = lines.flatMap(_.split(" ")) ✓
5. val pairs = words.map(word => (word, 1)) ✓
6. val wordCounts = pairs.reduceByKey(_ + _) ✓
7. val localValues = wordCounts.take(100)
8. localValues.foreach(r => println(r)) ✓

— DAG automatically
wakes Executors/splits

So let us see, a simple word count application, which is written in, a Spark, using Flume Java.

1. val conf = new SparkConf().setMaster("local[2]")
2. val sc = new SparkContext(conf)
3. val lines = sc.textFile(path, 2)
4. val words = lines.flatMap(_.split(" "))
5. val pairs = words.map(word => (word, 1))
6. val wordCounts = pairs.reduceByKey(_ + _)
7. val localValues = wordCounts.take(100)
8. localValues.foreach(r => println(r))

Refer slide time: (32:44)

Spark Implementation

Let us, see the SPARK implementation, in more details.

Refer slide time: (32:46)

Spark ideas

- Expressive computing system, not limited to map-reduce model
- Facilitate system memory
 - avoid saving intermediate results to disk
 - cache data for repetitive queries (e.g. for machine learning)
- Compatible with Hadoop

So, Spark ideas are an expressive computing system, which is not limited by, the Map Reduce model. That means, beyond Map Reduce also, the programming can be now done, in the SPARK. Now, this Spark will facilitate the system memory and it will avoid saving that immediate, results to the disk, it will cache for repeated, repetitive queries, that means, the output of the transform actions or the transformation will remain in the cache, so that, iterative applications can, can make use of this fast or efficient data sharing .

Refer slide time: (33:31)

RDD abstraction

- Resilient Distributed Datasets
- Partitioned collection of records
- Spread across the cluster
- Read-only
- Caching dataset in memory
 - different storage levels available
 - fallback to disk possible

The SPARK is also compatible with, with the Hadoop system. RDDs is an abstraction as I told you, it is a resilient distributed datasets, a partition collection of record ,they are spread across the cluster, they are here only and caching data sets, in the possible, in memory and different storage levels are possible.

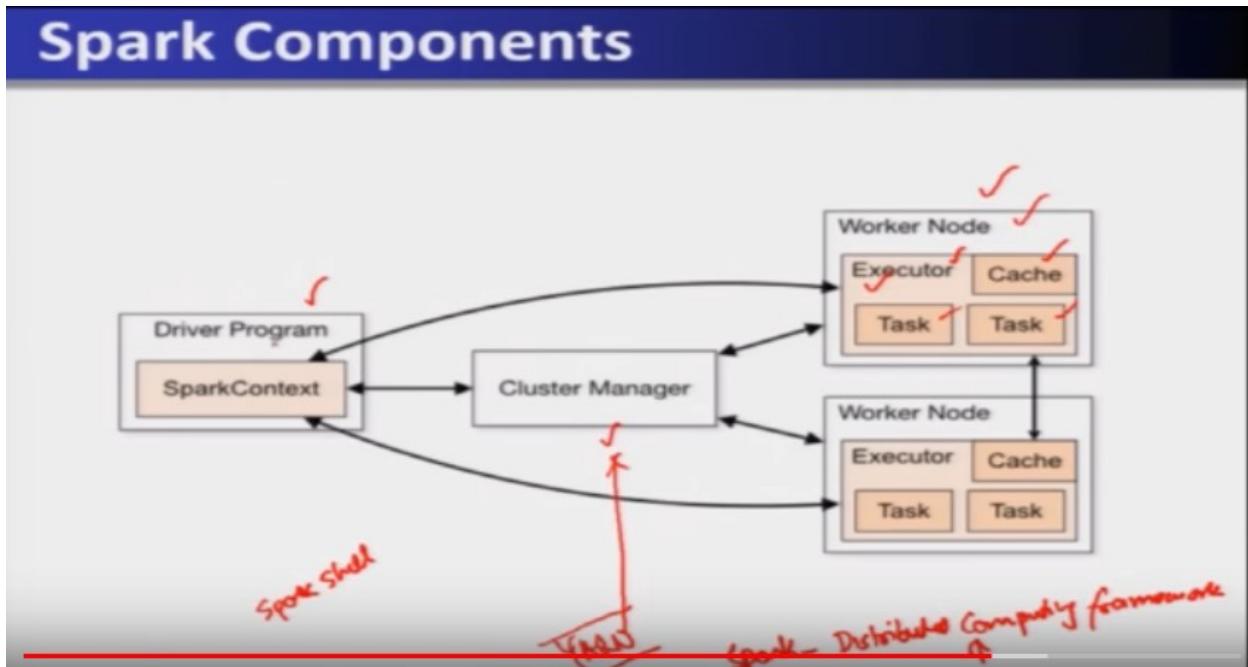
Refer slide time: (33:48)

RDD operations

- *Transformations* to build RDDs through deterministic operations on other RDDs
 - transformations include *map*, *filter*, *join*
 - lazy operation
- *Actions* to return value or export data
 - actions include *count*, *collect*, *save*
 - triggers execution

As I told you that, the transformations and actions, there are two operations, RDD supports and transformations include map filter joint, they are lazy operations and actions, include the count collect sale and they are trigger executions. Spark components, let us go and discuss

Refer slide time: (34:13)

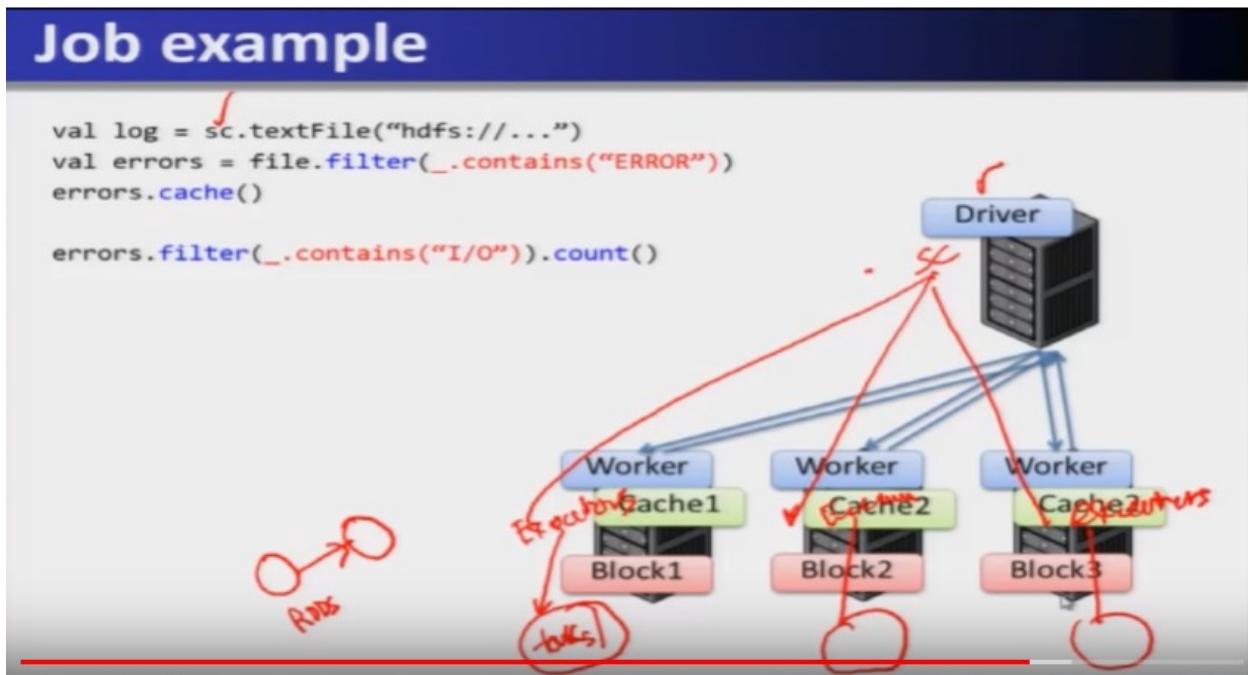


The Spark components, in more details. So, as you know that, is Spark, is a distributed computing framework. So now, we will see here, what are the different components? Which together will form? The computational environment of the Spark execution. Now, we have seen, there is a cluster manager, there is a driver program, there is a worker nodes and within the worker nodes, you have the executors, and within the executors what are the different tasks? And what is the cash? All these different components together will form the distributed computing framework, which will give an efficient, execution, environment, for the Spark program or a Spark applications. So here, so within, a driver program, when whenever a Spark, shell is being prompt, that will be inside the driver program, will create the Spark context.

Now executing the Spark context means that, it will communicate with the worker nodes, within the worker nodes the executors, so a Spark context will, create will, interact or communicate with the executors. And within the executors, the tasks will be executed. so executors within the executable, yes tasks will be executed and executors will be computed or will be executing on the worker nodes, so the driver program, then interacts with the cluster manager and cluster manager intern will interact with these worker nodes .so this all will happen, inside the Spark and Spark will dust the cluster manager. Now, there is an option in the Spark that instead of going through the cluster manager, you can also use the yarn and other, such resource manager and Scheduler. Now, let us see, what do you mean, by the, driver

program, Spark context, server and cluster manager, worker node, executor, then tasks and through, this to understand this.

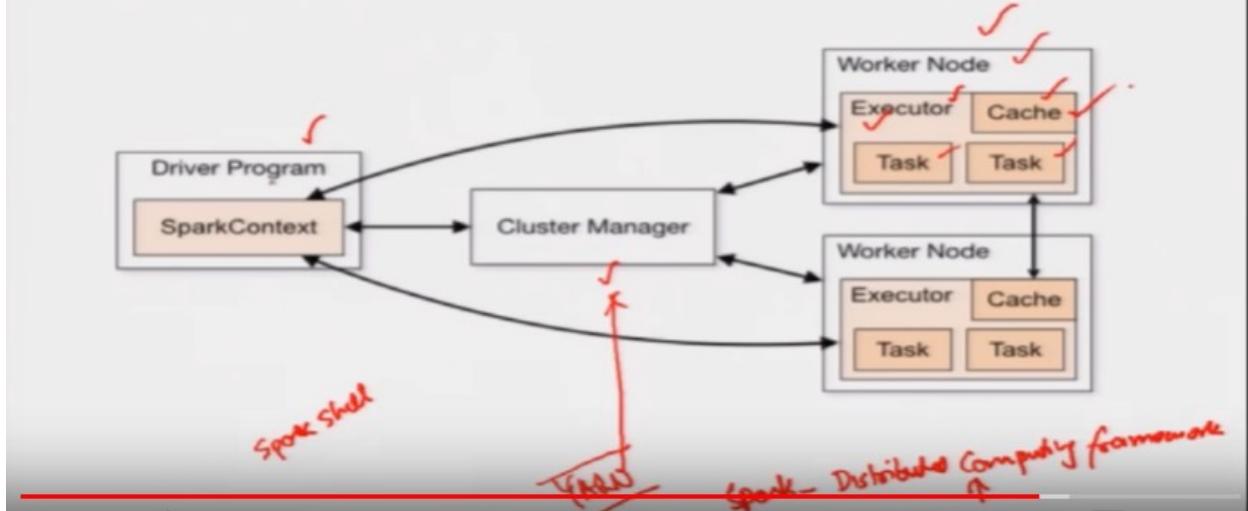
Refer slide time: (36:56)



Let us see, a simple application. Now, we have seen here, the driver program and this driver program will create a Spark context, it will create a Spark context SC. and this in turn will now, communicate with the executors, which are running inside, which are running inside. So, so this particular driver program, intern knows the, the different worker program communication and the Spark context, will now communicate to the executors. And these executors in turn, will communicate or will, will execute the tasks. So these tasks are, nothing but, the various transformations, the RDDs, through the dag, they are being transformed and they are done through the tasks. So different executors various tasks are being created and executing. So this will create the job execution and let us go back and, and see that, these

Refer slide time: (38:18)

Spark Components

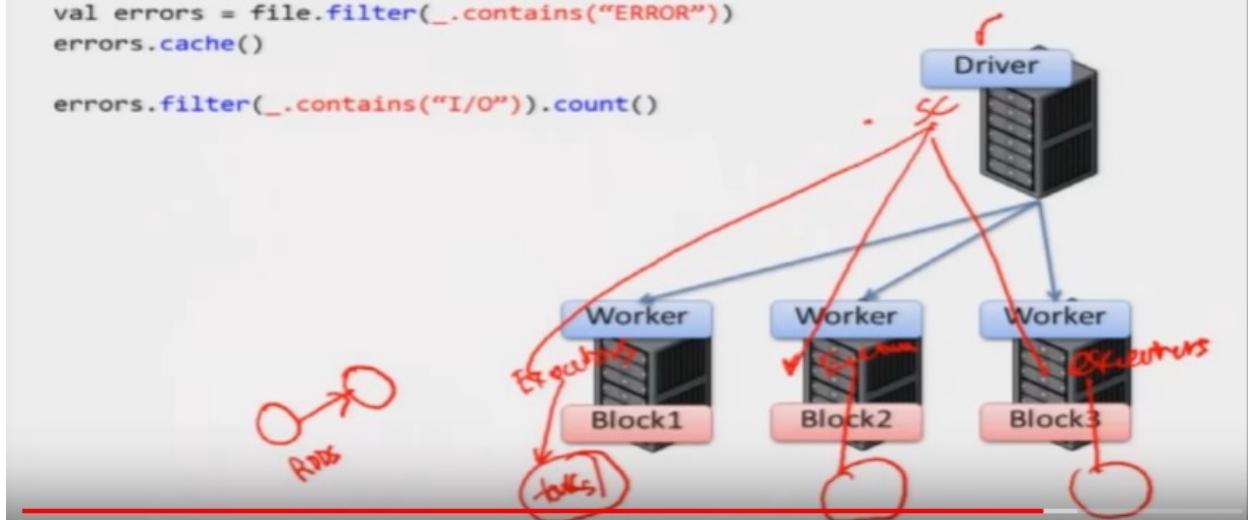


So this particular way, the driver program, will execute, the DAG and server Spark context SC. will be created which in turn will communicate inside the worker node with the executors and these executors in turn will execute various transformations and actions and the result will be remained in the cache.

Refer slide time: (38:46)

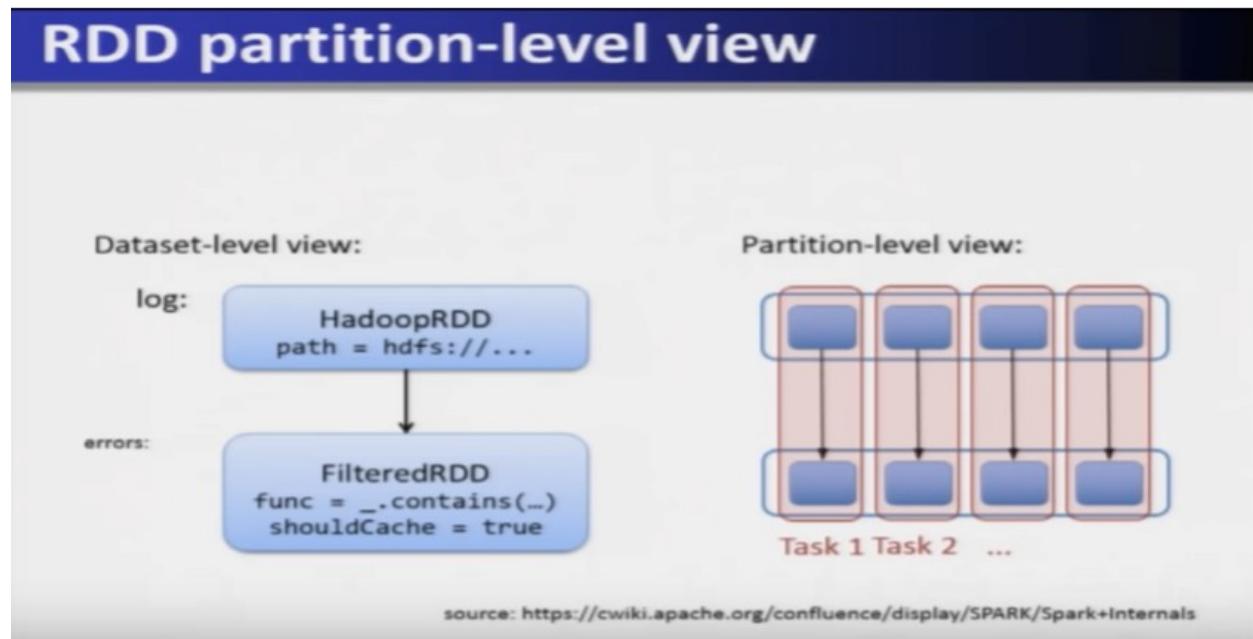
Job example

```
val log = sc.textFile("hdfs://...")  
val errors = file.filter(_.contains("ERROR"))  
errors.cache()  
  
errors.filter(_.contains("I/O")).count()
```



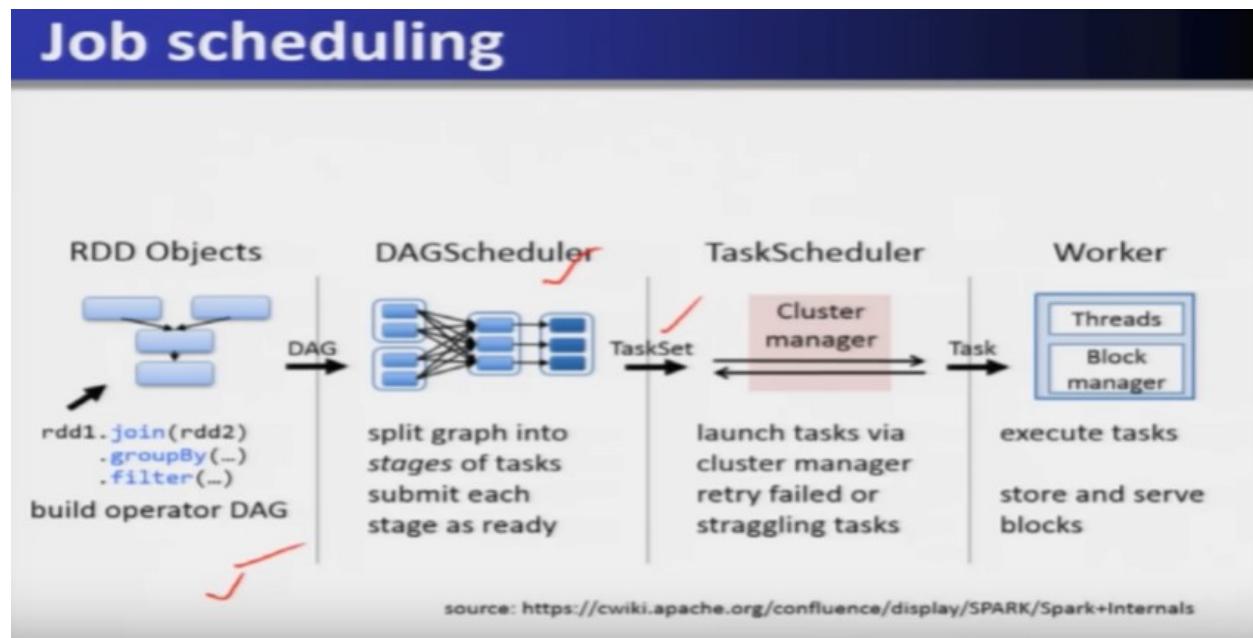
So that was, about the Spark components.

Refer slide time: (38:49)



So we have seen that, in this manner, the operation of Spark is being performed. Now, another view of partition level view, we can see here that, the partitioning so that means, RDDs are partitioned

Refer slide time: (39:09)



And different tasks are being executed. similarly job scheduling, that means, once an RDD, is that means operations are, given automatically it will build, the DAG and DAG will be given to the DAG scheduler and that, DAG scheduler will split the graph into the stages of, task and submit each stage as it is ready. So task set is created and given to the task a scheduler and as far as the cluster manager, is concerned it launches, the tasks, via the cluster manager and retry the field or straggling task. And this task is, given to the worker that we have seen in the previous slide and this particular workers will create the thread and execute them, there.

Refer slide time: (39:59)

Available APIs

- You can write in Java, Scala or Python
- Interactive interpreter: Scala & Python only
- Standalone applications: any
- Performance: Java & Scala are faster thanks to static typing

There are different APIs, which are available and you can write, these APIs is using Java program, scala or a Python. There is also an interactive interpreter: available, access through the scalar and Python. Standby applications are, there are many applications and performance: if we see that Java and C are faster and thanks to the static typing

Refer slide time: (40:26)

Hand on - interpreter

- script

```
http://cern.ch/kacper/spark.txt
```

- run scala spark interpreter

```
$ spark-shell
```

- or python interpreter

```
$ pyspark
```

Now let us see, the hands-on session, how we can perform, using Spark. So as Spark, we can run, as a scalar, so Spark shall, will be created.

Refer

slide

time:

(40:44)

Hand on – build and submission

- download and unpack source code

```
wget http://cern.ch/kacper/GvaWeather.tar.gz; tar -xzf GvaWeather.tar.gz
```

- build definition in

```
GvaWeather/gvaweather.sbt ✓
```

- source code

```
GvaWeather/src/main/scala/GvaWeather.scala
```

- building

```
cd GvaWeather  
sbt package
```

- job submission

```
spark-submit --master local --class GvaWeather  
target/scala-2.10/gva-weather_2.10-1.0.jar ✓
```

And we can download, the ,the, the file, that data set file and then, it can be, built using ,the package and then this particular task or a data file can be submitted ,to the, to the master node of that Spark system. So, so directly Spark, can store it into the system and now it can perform, various operations, on this particular data set, using a Scala program.

Refer slide time: (41:20)

Summary

- Concept not limited to single pass map-reduce
- Avoid sorting intermediate results on disk or HDFS
- Speedup computations when reusing datasets

The handwritten notes compare RDDs and Spark:

- RDDs**:
 - 1) MR/MR-like apps
 - 2) iterative
 - 3) Hadoop
- SPARK - RDDs**:
 - MR/REDUCE - beyond
 - in-memory operations
 - speedup efficient

Now summary, the concept of, of Map Reduce are limited to the single path Map Reduce is basically limiting various other applications and this particular, concept is avoiding, the sorting intermediate results, storing intermediate results on the disk or on HDFS. And also, speed-up computations are required, when reusing the datasets. and all these features are available, as part of ,this part that we have seen using RDDs. So using RDDs, now Spark provides, the not only Map Reduce, operations, beyond Map Reduce, it can also use .second thing is, it can be in memory, operations, not necessarily to be stored, in HDFS in to store the intermediate results. So this way of in-memory computations, will make the speed-up and brings about the efficiency, data sharing across different iterations. So iterative and interactive applications both are, easily supported and Map Reduce and non Map Reduce applications are also supported, by the Spark system. So all this is possible, with the help of RDDs and their operations. so we have seen that now the RDDs are saw a Spark is very much required and all the drawbacks of Map Reduce and Hadoop, is now not there with the SPARK and therefore the Spark now has, now various new applications. For example, the Spark system will, Spark.

Refer slide time: (43:49)

Conclusion

- RDDs (Resilient Distributed Datasets (RDDs) provide a simple and efficient programming model — batch, iterative, streaming
- Generalized to a broad set of applications
- Leverages coarse-grained nature of parallel algorithms for failure recovery



So the Spark is a core, as a core, can be used for building different applications, such as, Spark MLlib, that is the machine learning or the Spark, then Spark streaming ,that is the real-time applications, over the Spark and Spark graphics, the graph computation or the Spark. So, now Spark can use HDFS or may not use HDFS, Spark is independent. Therefore, let us, conclude this discussion, that RDD, is resilient distributed datasets, will provide a simple and efficient programming model, for different supporting various applications, which are the batch and interactive and iterative applications, all are supported using this concept, which is called, 'RDDs'. Now, this is generalized, to a broad set of applications. And it will, leverage the coarse-grained nature of parallel algorithm, for fault recovery. So that is why, this is a hundred times, faster, compared to the, traditional Map Reduce. So, Spark is 100 times faster, that is what is compared, with the performance, by the Spark production clusters. Thank you.

Lecture – 11

Spark Built-in Libraries

Refer slide time: (0:14)

Spark Built-in Libraries



Dr. Rajiv Misra

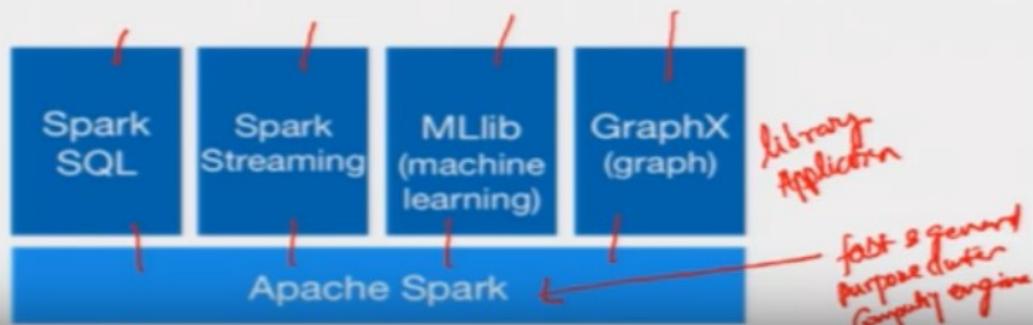
Dept. of Computer Science & Engg.
Indian Institute of Technology Patna
rajivm@iitp.ac.in

The Spark Built-in Libraries.

Refer slide time: (0:16)

Introduction

- Apache Spark is a fast and general-purpose cluster computing system for large scale data processing
- High-level APIs in Java, Scala, Python and R



Now Spark, Apache Spark is a fast and general-purpose cluster computing for large-scale data processing. The Spark supports, high level APIs, in the form with, Java, Scala, Python and R. Now let us see the Spark core, which is the fast and a general purpose cluster computing engine. Now this particular Spark is giving a fast and general-purpose computing, which provides or which has enabled various applications, to be run on top of this Spark core or Spark engine. Such as the various components, on various libraries, which are supported, by the Spark, for different applications, are summarized in this particular diagram. So the first one is called, ‘Spark Sql.’ So that means, Sql like, commands are being provided and by that, the key value store and various programming can be supported, using, Sql like commands. The another one is called as, ‘Spark streaming’ for real-time streaming applications. Here the data will be in the form of micro batches and the streams will be computed in, real-time. Another one is called, ‘Spark MLlib’, that is machine learning libraries, which are provided, over the ‘Spark Core’. Finally the graph processing is done over top of the Spark in the form of library, which is called, ‘Graph X’.

Refer slide time: (2:12)

Standard Library for Big Data

- Big data apps lack “libraries” of common algorithms
- Spark’s generality + support“ for multiple languages make it“ suitable to offer this
- Much of future activity will be in these libraries

The diagram illustrates the 'Standard Library for Big Data'. At the top, four language icons (Python, Scala, Java, R) are aligned horizontally. Below them is a blue rectangular box labeled 'Core' in the center. Inside this 'Core' box are four smaller white boxes: 'SQL' on the left, 'ML' in the middle, 'graph' to its right, and three dots (...) further to the right. At the bottom right corner of the 'Core' box is the 'Spark' logo, which consists of the word 'Spark' in a stylized font with an orange star icon.

So these are the standard libraries, which are used for various big data applications and also supports, various common algorithms, which are used for big data analytics.

Refer slide time: (2:26)

Machine Learning Library (MLlib)

MLlib algorithms:

- (i) **Classification:** logistic regression, linear SVM, "naïve Bayes, classification tree
- (ii) **Regression:** generalized linear models (GLMs), regression tree
- (iii) **Collaborative filtering:** alternating least squares (ALS), non-negative matrix factorization (NMF)
- (iv) **Clustering:** k-means
- (v) **Decomposition:** SVD, PCA
- (vi) **Optimization:** stochastic gradient descent, L-BFGS

So let us see, these libraries, standard libraries, which are part of the Spark core, in more details.

Refer slide time: (2:31)

Machine Learning Library (MLlib)

MLlib algorithms:

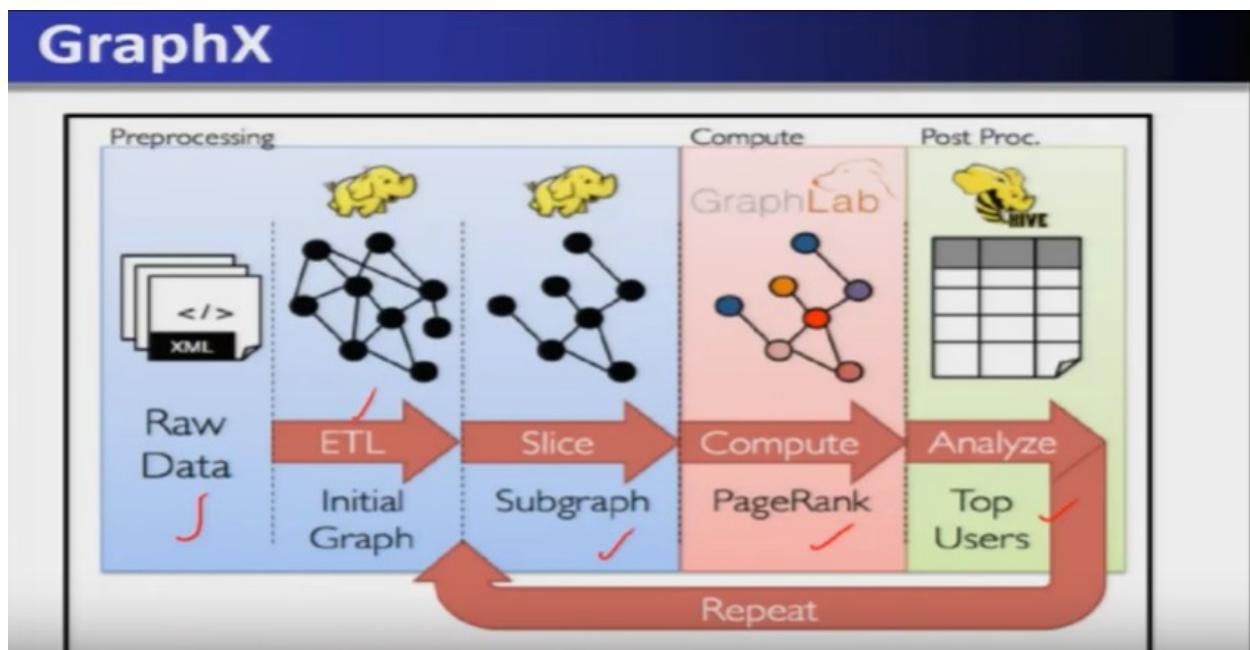
- (i) **Classification:** logistic regression, linear SVM, "naïve Bayes, classification tree
- (ii) **Regression:** generalized linear models (GLMs), regression tree
- (iii) **Collaborative filtering:** alternating least squares (ALS), non-negative matrix factorization (NMF)
- (iv) **Clustering:** k-means
- (v) **Decomposition:** SVD, PCA
- (vi) **Optimization:** stochastic gradient descent, L-BFGS

SPARK MLlib
Scalable Machine learning
Algorithms for
big data
Analytics

For example, the Spark machine learning library, which is called, 'A Spark MLlib Library'. Now this particular, Spark MLlib Library provides a collection of machine learning algorithms, which are provided

here, for doing the big data analytics, that is called, ‘Scalable Machine Learning Algorithms’ for big data analytics. Let us summarize what are, what are the different machine learning, scalable algorithms, are available in the form of MLlib. For the classification application, the algorithms like, logistic regression, linear support vector machine, Naive Bayes and decision trees are available, as part of the MLlib library. For regression application, generalized linear model (GLM) and regression tree is available. Similarly, for collaborative filtering, alternating least squares and non-negative matrix factorization is available, as part of MLlib. For unsupervised clustering or cluster analysis, parallel K means algorithm is available as part of MLlib. Similarly, for decomposition, support, SVD and principal component analysis is available for decomposition. And for the optimization, various libraries are available, will such as, Stochastic Gradient Descent and L-BFGS.

Refer slide time: (4:24)



Let us see the another library, which is supporting the graph applications over the Spark, that is large-scale graph computation over the Spark, which is supported in the form of the graphics. So here the graph, here the raw data, it takes and this figure shows that, that extract, transform and load this particular, ETL, it will through that, it will create the initial graph. It will perform various transformations, operations, on the graph, such as; it will create the sub graph, it will perform the graph algorithms. Which are, such as, page rank and it will do the different analysis. So all these operations are supported in the form of GraphX.

Refer slide time: (5:15)

GraphX

- General graph processing library
- Build graph using RDDs of nodes and edges
- Large library of graph algorithms with composable steps

So GraphX, is a general purpose, graph processing library and it builds the graph using RDD's of the nodes and edges. Large library of graph algorithms are available as part of the GraphX.

Refer slide time: (5:31)

GraphX Algorithms

(i) Collaborative Filtering

Alternating Least Squares
Stochastic Gradient Descent
Tensor Factorization

(ii) Structured Prediction

Loopy Belief Propagation
Max-Product Linear Programs
Gibbs Sampling

(iii) Semi-supervised ML

Graph SSL
CoEM

(iv) Community Detection

Triangle-Counting
K-core Decomposition
K-Truss

(v) Graph Analytics

PageRank
Personalized PageRank
Shortest Path
Graph Coloring

(vi) Classification

Neural Networks

Let us summarize some of the algorithms, which are available, as part of the GraphX library and doing, and for the Graph processing. Collaborative filtering and then structured prediction, then semi-supervised machine learning and then, community detection, graph analytics and classification, using neural networks. So all these graph analytics are available, such as page rank algorithm on graphs, personalized page rank algorithm, shortest path graph coloring. All these algorithms, are available, as part of the graph X library.

Refer slide time: (6:23)

Spark Streaming

- Large scale streaming computation
- Ensure exactly one semantics
- Integrated with Spark → unifies batch, interactive, and streaming computations!

```
graph LR; A[live data stream] --> B[Spark Streaming]; B --> C[batches of X seconds]; C --> D[Spark]; D --> E[processed results]
```

Similarly, for community Detection, triangle counting, K-core Decomposition, K-Truss, all these algorithms are also available for doing the graph analytics. Similarly for SPARK streaming, this particular Spark core provides library for supporting the Spark streaming, Large-scale streaming applications are supported in the form of Spark streaming, Spark standard streaming library system. And it will unify, the integrate with the Spark to unify for batch interactive and streaming computations together.

Refer slide time: (6:57)

Spark SQL

Enables loading & querying structured data in Spark

From Hive:

```
c = HiveContext(sc)
rows = c.sql("select text, year from hivetable")
rows.filter(lambda r: r.year > 2013).collect()
```

From JSON:

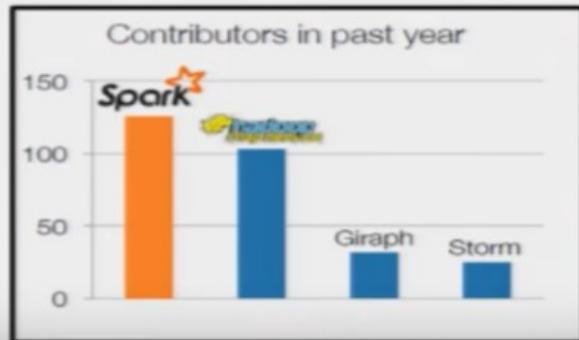
```
c.jsonFile("tweets.json").registerAsTable("tweets")
c.sql("select text, user.name from tweets")
```

Next library which is available with the Spark core is called, ‘Spark SQL’. Now it will enable loading and querying structured data, in the form of the Spark. It is also having the links or APIs with the Hive and with JSON.

Refer slide time: (7:16)

Spark Community

- Most active open source community in big data
- 200+ developers, 50+ companies contributing



So Spark community, now is most open-source, most active open-source community in the Big Data and 200 plus developers and 50 plus companies are contributing. And these icons shows that, the complete presence of a Spark in all the, the production clusters, at most of these companies are shown over here and you can see that. The Spark surpassed the Hadoop Map Reduce in the number of contributions. Thank you.

Lecture 12

Design of Key Value Stores

Refer slide time :(0:23)

Preface

Content of this Lecture:

- In this lecture, we will discuss the design and insight of **Key-value/NoSQL stores** for today's cloud storage systems.
- We will also discuss one of the most popular cloud storage system i.e. **Apache Cassandra** and different consistency solutions.

Preface content of this lecture; in this lecture we will discuss the design and insight of key-value store, which is also known as NoSQL stores, which is used for storage of the big data system. These key-value stores are also provided as the storage system for the big data and also supported as the cloud storage systems. We will also discuss, one popular key value store which is called, ‘Apache Cassandra’, which is one of the most important and widely used, NoSQL stores, used for the big data storage and also I will add the cloud storage systems, will also see the different forms of consistency solutions which are provided by Apache Cassandra in this lecture.

Refer slide time :(01:20)

The Key-value Abstraction

- **(Business) Key → Value**
- **(flipkart.com) item number → information about it**
- **(easemytrip.com) Flight number → information about flight, e.g., availability**
- **(twitter.com) tweet id → information about tweet**
- **(mybank.com) Account number → information about it**

Let us understand the key value abstraction, key value for, any business and important, item or entity is called a ‘Key’, and all the attributes related to that becomes, the value for example in a flipkart.com company, they are the item which they deal as the sales is called the, ‘Key’ and all the information about that item becomes, the value for example how much the quantity of that item what are the reviews of that item? And so, on and so forth similarly for another company easemytrip.com which deals with the flight reservation, there the key will become the flight number and the information about the flight for example the seat availability and so on, will become the value. So, key-value store is defined in this manner, similarly in a twitter.com the tweet ID becomes, the key and the information about the tweet that is the

time, day and the text of the tweet becomes the value of that system. In mybank.com in various bank operations, the key becomes the account number and the information pertaining to that account becomes the value. so, this way the key value abstraction, will provide the storage, of the data in different businesses and different companies, which is required to be stored, in a large amount, that we are going to see how we are going to handle this key value abstraction, through the databases which are available, in this domain of technology.

Refer slide time :(03:20)

The Key-value Abstraction (2)

- It's a **dictionary datastructure**.
 - Insert, lookup, and delete by key
 - Example: hash table, binary tree
- But distributed.
 - Cluster | Datacenter
- Seems familiar? Remember **Distributed Hash tables (DHT) in P2P systems?**
- Key-value stores reuse many techniques from DHTs.
 - for BIG dataStorage
 - for Key-value store

So, key-value store, which we have just discussed, can be visualized as the dictionary data structure, for example, to have this abstraction key value ,abstraction for example the operations which require to be supported in the dictionary data structure such as insert, lookup and delete by key. And this kind of key abstraction can easily be provided by hash table binary tree and so on these data structures. So, in the dictionary area data structures what we have seen that in the form of hash table and binary tree can implement these key value abstraction but the issue, there it is that these dictionary data structure can be supported by a single system in a computer, now the size of the data if it becomes very big, then this becomes a problem that means a single system cannot store the entire data hence, this kind of scenario which is available for providing the key-value abstraction using dictionary data structure like hash table and binary tree is good enough for a small amount of data. So, when it becomes a big data, this key-value store how we are going to handle, this key-value store abstraction, becomes the point of discussion in this lecture. The way to store the large amount of data is not to be handled in a distributed manner that is not one system but a cluster or a data center, will be used to store the big data that is in the form of a key-value store, hence this particular concept of storing the key-value or providing the key-value abstraction

for a big data is not done through the single system but it has to be achieved through a distributed computation system or we can understand it by providing this entire storage in the form of the cluster or in the form of a data center. Now this particular concept seems familiar if we recall the use of distributed hash tables in a peer-to-peer system, now we are going to see that this key-value store may reuse many techniques which we have known in the peer-to-peer systems in form of distributed hash table we will see how this entire technique can be utilized to design this key-value abstraction for the Big Data system.

Refer slide time :(07:17)

Is it a kind of database ?

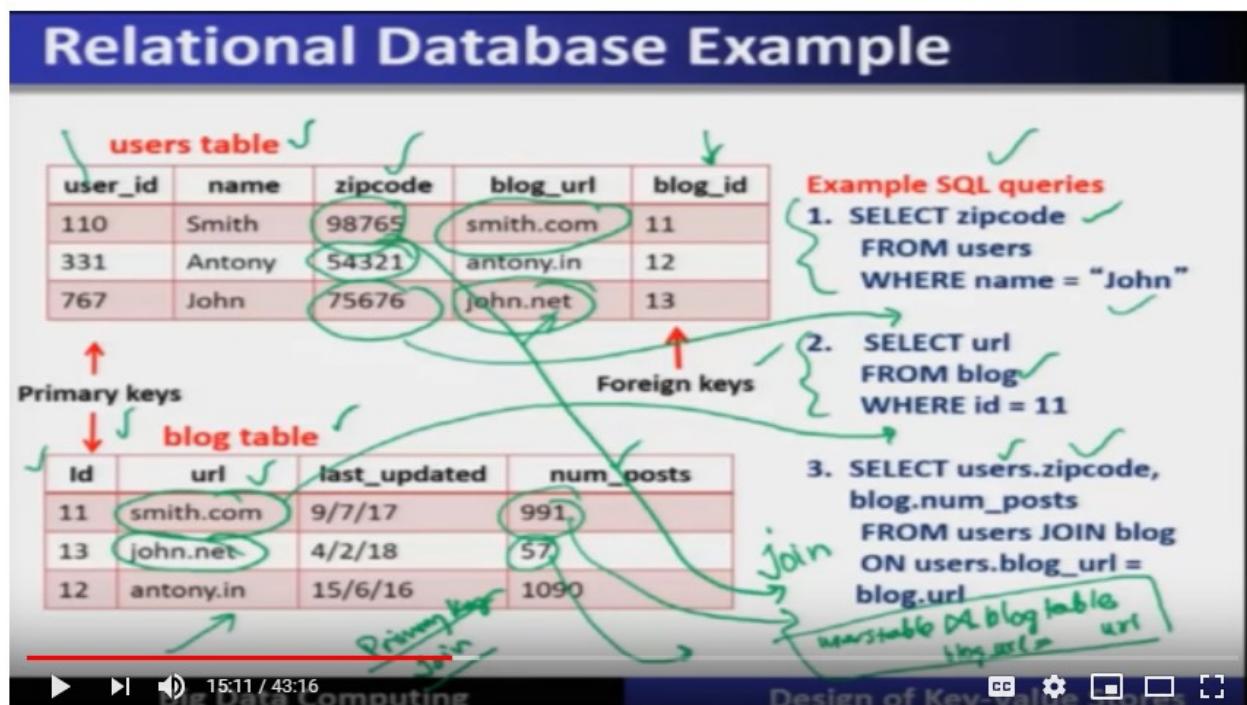
- Yes, kind of
- **Relational Database Management Systems (RDBMSs)** have been around for ages ✓
- **MySQL** is the most popular among them ✓
- Data stored in tables ✓ ✓
- Schema-based, i.e., structured tables ✓
- Each row (data item) in a table has a primary key that is unique within that table ✓
- Queried using **SQL (Structured Query Language)** ✓
- Supports joins ✓

Database
RDBMS
- Schema
- Structured data store
- Primary Key/Un
- SQL ↗
Key-value
store
for Big
data

Is it a kind of database, yes it is a kind of relational database management system, that have been around for ages, the most important among them is MySQL database system which stores the data in the form of tables, which are the schema based structure tables where in the each row that is the data item in the table has a primary key and that is unique within that particular table. And these queries are done using a language which is called, ‘Structured Query Language’ or a SQL which supports the join, therefore we see that the database which are also called, ‘Relational Database Management System’, has the schema and the data has to be organized according to this schema hence it is called ,’Structured Data Store’, and this particular data has this particular system of a database management system in our DBMS highest from primary key, I mean the concept of a joint operates over there and the queries or the data can be queried with the help of a language which is called, ‘SQL’. Now let us see, how we are going to use this concept for our key value store, for the Big Data. So, the question is can we use this concept of, the

existing structured, data that is structure tables which are managed in the form of well-known relational database management system. So, can we use it or we cannot use it both these questions are to be answered in the further slides.

Refer slide time :(09:38)



Before that let us understand RDBMS, the concept of storing the key-value store or the, the data in a structured format and performing the queries and the concept of primary key and a joint operation just we will review it and then we will go ahead. So, in a relational database management system data is organized in the form of a schema and they are called as the ‘Tables’. So, here in this picture, we have in this example we have shown the users table and a blog table, where in the users table this is the primary key, user_ID will become a primary key and all other attributes, name, zip code, blog URL and blog ID becomes other attributes in the schema, similarly the blog table has ID as the primary key and all other, all other attributes like URL last updated and number post are also a part of the schema of the blog table. Now as far as in users table if we go inside the detail of the attribute blog ID, this we call it as foreign key, this blog ID is referring to the IDs which are the primary key, of a blog table hence this becomes, a prime foreign key and now this particular database once it is defined then we can perform the SQL queries, on top of it the SQL queries are such that if you want to find out the zip code from the users,

where the name of the user is the John if you want to find out that zip code this particular query will fetch this particular value over here, similarly if you want to run another query on this particular database let us say select URL from this particular table blog where ID becomes 11. So, this particular URL smith.com will become the result of this query, now furthermore if we require to, to select the users zip code. So, users zip code and, and blog number posts together from the users and the blog file together and where in the users blog URL, is equal to the blog URL on the other table that means this particular query can be satisfied when we join two different tables and that is the users, table when we join with the blog table, on the key as the users on the key users blog, blog URL is equal to the URL of the other table. So, this requires the join operation, this particular requires the join operation to perform this particular key. So, the outcome will be the when the users blog URL and the blog URL are same and the URL in the blog table if they are same for example the Smith both are same and let us see the when Smith, is the same then we want to find out the users zip code the user zip code is this value and blogs number post nine, nine one. So, these values will be given as the output similarly in another case, when this John net and they are same then in that case the number of post is 57 and the, the zip code is sorry, zip code is five seven four seven five six seven. So, this way we have shown that once the relational database is defined and data is stored in the form of the tables that is in the structure schema, the every table defines a primary key, then we can perform the different queries that is SQL based on the primary key are based on the, the operation which are called, 'Join Operation'. So, therefore in our DBMS the primary key and the join operation plays a very important role, in that retrieval storage and retrieval of the data.

Refer slide time :(15:10)

Mismatch with today's workloads

- **Data: Large and unstructured:** Difficult to come out with schemas where the data can fit
- **Lots of random reads and writes:** Coming from millions of clients. *large number of queries read/write*
- **Sometimes write-heavy:** Lot more writes compare to read *write-heavy workloads*
- **Foreign keys rarely needed**
- **Joins infrequent**

Foreign key is rarely used in today's workload

Join because it is frequent

Relational DBMS is good for structured data (Coming to the schema) - Foreign key & Join

Now as far as in today's workload, when we see that, whether this kind of our DBMS ,that is the structure data can be further utilized or not now to answer this question let us understand the today's workload. So, there is a mismatch in today's workload with the existing our DBMS or a structural data system in the following manner, first is that the first mismatch is about the data. So, in today's workload the data is characterized by a very large, volume and also the data is unstructured, meaning to say that when it becomes unstructured, then it cannot fit in any of the schema, which is a predefined. And second thing is the data volume is too large it cannot fit in to several even tables which can be stored on a one computer system. Now another mismatch in today's workload, we can think of in the terms of read and writes, which are in large number of random, read and write operations coming from millions of clients. So, how are you going to handle this large, number of queries that is the read and write operations, third thing is about today's workload, is that which differs from the previous our DBMS systems which were handling the workload, is that these workloads have the right heavy operations. So, most of the time, these workloads requires the right operations, with compared to the, the lesser number of read operation but read and write put together are much larger in the volume compared to the previous our DBMS s hence, this is known as the write heavy workloads.

Four thing is that, in this particular workloads we are not going to handle we are not going to use this foreign key and this foreign key is rarely used. So, foreign key is rarely used in today's, workload, also we see that this joint operation, also is becoming infrequent in today's workload, therefore the foreign key and the joint operation, is not very much used in today's workloads. So, therefore we are going to design or we are going to design a database management system, which is going to handle today's workload which has these following characteristic or the requirements to handle the large volume, of unstructured, data which cannot be specified in schema. Second thing is that it is the right heavy, workloads light of right operations are too many numbers and finally the foreign key and joint operations are not very required in the today's workload. So, let us see with this particular requirement how we are going to design a new database system, which is going to cater to these today's workloads and how our why and we are going to provide how we are going to provide the key value store for the Big Data system.

Refer slide time :(20:07)

Needs of Today's Workloads

- Speed ↗
 - Avoid Single point of Failure (SPoF)
 - Low TCO (Total cost of operation and Total cost of ownership)
 - Fewer system administrators
 - Incremental Scalability ↗
 - Scale out, not scale up ↗
3. lightning fast writes
2. Fault-tolerance/availability

Scalability -
(adding) more nodes
Computer system to scale
linearly the performance
+ storage system
Scale Out



So, therefore in today's workload, what is needed is the speed? In which these write heavy workloads are to be catered. So, that means a lightning-fast, writes has to be supported, in today's workload. Second point is that we have to avoid, in today's workload the single point of failure, that means the data that means we are not going to be affected, by the failure of a node, in this kind of system, that is called, 'Fault-Tolerance'. And availability third important criteria is about low cost of operation and total cost of ownership and fewer system administrators are required to manage, this entire big data system and also it will be handling able to handle the incremental scalability, that means to support the scale out. So, let us understand the scalability aspect scalability by scalability we mean that, as the data volume grows, we keep on adding more system and therefore the capacity of that handling of storage of a big data system automatically increases, without that is called scalability, scalability means we can keep on adding, more nodes or a more computer systems, to scale linearly, the performance, of a storage system, this is called as the, 'Scalability', this technique of adding the computer, without replacing with the new one is called a, 'Scale Out'.

Refer slide time :(22:33)

Scale out, not Scale up

- **Scale up = grow your cluster capacity by replacing with more powerful machines**
 - Traditional approach ✓
 - Not cost-effective, as you're buying above the sweet spot on the price curve
 - And you need to replace machines often ✓ *Costly & scale*
- **Scale out = incrementally grow your cluster capacity by adding more COTS machines (Components Off the Shelf)**
 - Cheaper ✓
 - Over a long duration, phase in a few newer (faster) machines as you phase out a few older machines ✓
 - Used by most companies who run datacenters and clouds today



24:45 / 43:16

Big Data Computing

Design of Key-Value Stores



Let us understand about, what you mean by a scale out? And which is supported, in the new today's workload system, to handle the big data system. Now is scale out means, that it will support to increment, incrementally grow, your cluster capacity by adding more, component of shelf systems, that means this way of, of achieving the scalability becomes, cheaper why because we are not going to replace with the costlier system, we keep on adding more number of systems, that is called, 'Scaling Out', and also over the long duration, we have to face in a new phase, in a new phase a few newer faster machines as you phase out a few older machines. So, that is called a, 'Scale Out'. So, hence this becomes a cheaper way of scaling up the, the entire system. And that is how we are going to build the cluster systems, which is being supported by a scale out technology, this particular is killing out is supported by oil is being used by many companies, which runs the data centers and the cloud today, in contrast to scale out there is a scaling up scaling up, means the traditional computer system, we are going to replace with a high capacity powerful machines, by to increase the, the capacity of the system in terms of memory and the processing capabilities and so on. So, this is a very costly affair, by providing the scalability which is called, 'Scaling up' by replacing with a very powerful machines, that means the old machines, need to be replaced with the newer machines this becomes a costly affair, in compared to the two the scale out.

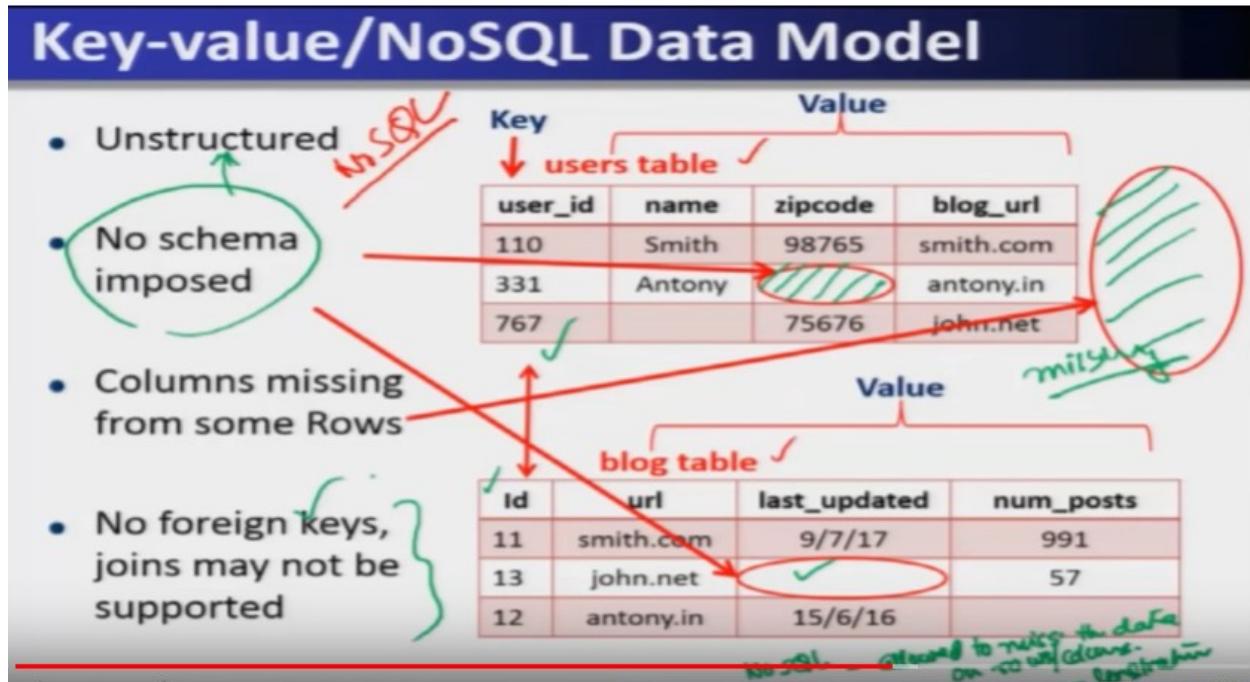
Refer slide time :(24:49)

Key-value/NoSQL Data Model

- NoSQL = “Not Only SQL” ✓
- Necessary API operations: **get(key)** and **put(key, value)** ✓
 - And some extended operations, e.g., “CQL” in Cassandra key-value store
- **Tables** ✓
 - “Column families” in Cassandra, “Table” in HBase, “Collection” in MongoDB
 - Like RDBMS tables, but ...
 - May be unstructured: May not have schemas ✓ *not have schema*
 - Some columns may be missing from some rows
 - Don’t always support joins or have foreign keys
 - Can have index tables, just like RDBMSs

Mechanism of scalability, now we are going to understand this concept of a key value store, of the know sequel data model. So, key value store normally is provided, in today's to handle, the today's workload in the form of NoSQL data model, now let us see understand what do you mean by NoSQL data model, NoSQL stands for not only SQL. So, not only SQL means beyond SQL whatever is limitations, of SQL it will go beyond and it will provide the support, to the today's workload, that is why it is called, ‘NoSQL Data Model’. And which is used to store the large amount of key value store, abstractions, now this new sequel is to data model provides the necessary API operations and the two most important API, is which are provided by NoSQL system are get by key and put by key. So, put by key value, is basically, writing the key-value pair and get means we want to read the key value pair. So, given a key we want to read that key-value pair. So, these operations are provided as in, in the different NoSQL database systems which are available as of today. Now and some extended operations are being provided, in the form of for example are also provided for example the CQL that is called, ‘Cassandra CQL Language’, is being provided by the Cassandra database system, Cassandra provides a NoSQL data model and this is we are going to understand here in this lecture, about this particular data model that is called, ‘Cassandra’, which supports NoSQL for providing the key value store for big data systems. So, in the NoSQL system, the tables are often called the, ‘Column Families’, for example in case and right is the tables are called, ‘Column Families’, and in HBase, it is called , ‘Table’, and in MongoDB it is called collection. Like our DBMS tables and here, it is called, ‘Column Families’. These particular tables may be unstructured, that means they do not have any fixed is schema and some column may be missing and some rows may also be missing hence, they are not called a, ‘Structured Data’. So, there is no schema which can fit this data hence it is called, ‘Unstructured Data’. So and also it doesn't support, the join and the foreign keys and also can have, the index tables like RDBMS. So, these are the some of the features of the key value store which is provided in the form of NoSQL.

Refer slide time :(28: 13)



So, let us understand these, concept which is called, ‘Unstructured’. Where no schema is there and what do you mean by this? And how no foreign keys and joint operations are supported. Let us take the same example of the two tables, the user table and the block table which captures the data in, in this today's workload system or in a NoSQL system. In today's NoSQL system, let us understand by this example and here, we can see that this particular table users table which is called a, ‘Column Family’. And sometimes table in HBase, here we can see that in some of the attributes and even some of the columns are missing, if they are missing that means sometimes it is called a, ‘Null Value’ or there is no schema which is imposed if the entire column is missing. So, therefore this particular model, data model is called, ‘Unstructured Data’. Similarly an entire column can be missing and from some rows and also, you can see the same thing here, these particular entities are missing, some of the entries are missing from the table which is not possible to be there in RDBMS, but in the NoSQL this is allowed and this is not a problem, hence it is called, ‘Unstructured Data’. Why because there is no schema, which can fit this, this type of data. Which is coming in today's workloads another thing is they are. So, primary keys are there, but there is no concept of foreign key, foreign keys are not required and also there is no joins which are supported in NoSQL system.

Refer slide time :(30: 48)

Column-Oriented Storage

NoSQL systems often use column-oriented storage

- RDBMSs store an entire row together (on disk or at a server)
- NoSQL systems typically store a column together (or a group of columns).
 - Entries within a column are indexed and easy to locate, given a key (and vice-versa)
- **Why useful?**
 - Range searches within a column are fast since you don't need to fetch the entire database
 - E.g., Get me all the blog_ids from the blog table that were updated within the past month
 - Search in the last_updated column, fetch corresponding blog_id column
 - Don't need to fetch the other columns

Now, this storage system is called column-oriented storage system. Now, we have to understand this concept in a better in more details. So, NoSQL systems often use column-oriented storage. What do you mean by column oriented stories? That means the SQL system uses to store are used to process the row wise and now, in NoSQL system if it is, storing the entire columns and column wise operations if it is, being performing then it is called, ‘Column-Oriented Storage’. So, NoSQL system often uses the column oriented storage. So, RDBMS store, the entire row together, on the disk or at the server whereas the NoSQL system typically store a column together or a group of columns. Now, entries within the columns are indexed and easy to locate a given a key and vice versa. So, that means we are here, in the column oriented storage we are handling the columns. That is why it is called a, ‘Column Oriented Storage’. In contrast to the RDBMS which uses, the rows together and they are stored together in these forms. Why this is all useful concept is? Because, the range searches within the column or fast since you don't need fetch the entire database. So, that means if your query is targeted to be answered from that column itself, all the entries of the column then the column is required to be pressed and this and it can without fetching the entire database. So, hence the range queries can be easily supported, within the columns. We will see this kind of operations in further detail how, that is all implemented in NoSQL systems. For example get me all the blog IDs, from the block table that were updated within the last month if that is, the query then we have to search in the last updated column fetch the corresponding block ID column and we don't have to fetch all other columns in this case. So, this kind of query which is very common in today's workloads and then that is why the today's NoSQL databases are column oriented storage.

Refer slide time :(33: 37)

Design of Apache Cassandra



Now, let us go and discuss the design of Apache Cassandra, Apache Cassandra before we go ahead let us see there are two, there are two companies, one is called, ‘Google’. And this Google has inspired this NoSQL system, which is now taken by the Face book and then Face book, has developed this Cassandra and made the open source and hence this is called, ‘Apache Cassandra’. So, we will see the development and we will see why it is so, important in today’s scenario and what are the important things about Cassandra, we are going to see in the design, of Apache Cassandra.

Refer slide time :(34: 34)

Cassandra

- A distributed key-value store ✓
- Intended to run in a datacenter (and also across DCs) ✓
- Originally designed at Facebook ✓
- Open-sourced later, today an Apache project ✓
- Some of the companies that use Cassandra in their production clusters
 - Blue chip companies: IBM, Adobe, HP, eBay, Ericsson
 - Newer companies: Twitter ✓ twitterID → value Cassandra Cluster
 - Nonprofit companies: PBS Kids
 - Netflix: uses Cassandra to keep track of positions in the video.

▶ ▶ | 36:39 / 43:16 Design of Key-Value Stores CC

So, Apache Cassandra is a distributed, database management system or it is also called as a, 'Distributed Key-Value Store'. Now this particular database management system, it is intended to run in the data center or across the data centers, it is not meant for to run on a single node, but it is meant to run on a data center, originally it was designed at the Face book and it was open sourced later, by the Apache project, some of the companies that use Cassandra in their production, clusters are blue chip companies like IBM, Adobe, HP, eBay, Ericsson. Newer companies like Twitter also uses, the Cassandra for restoring their tweets and nonprofit companies like PBS, KIDS also uses it and Netflix, also uses Cassandra to keep track of the positions, in the video while you watch the movies on the Netflix. So, Cassandra is so important that most of the companies, in the reduction cluster, they use this Cassandra for storage system, of the large key value store, as per their requirements, for example we have already seen that the twitter ID the twitter company, uses the twitter ID, this becomes the key and the value becomes, the information about that tweet, this particular key value is stored in by the twitter in their Cassandra, production cluster. We are going to see now the more detailed design of the Cassandra, how it supports the storage of the large volume of key-value store that is how it is going to support the today's workload, by providing the abstraction key value, to the different companies.

Refer slide time :(36: 44)

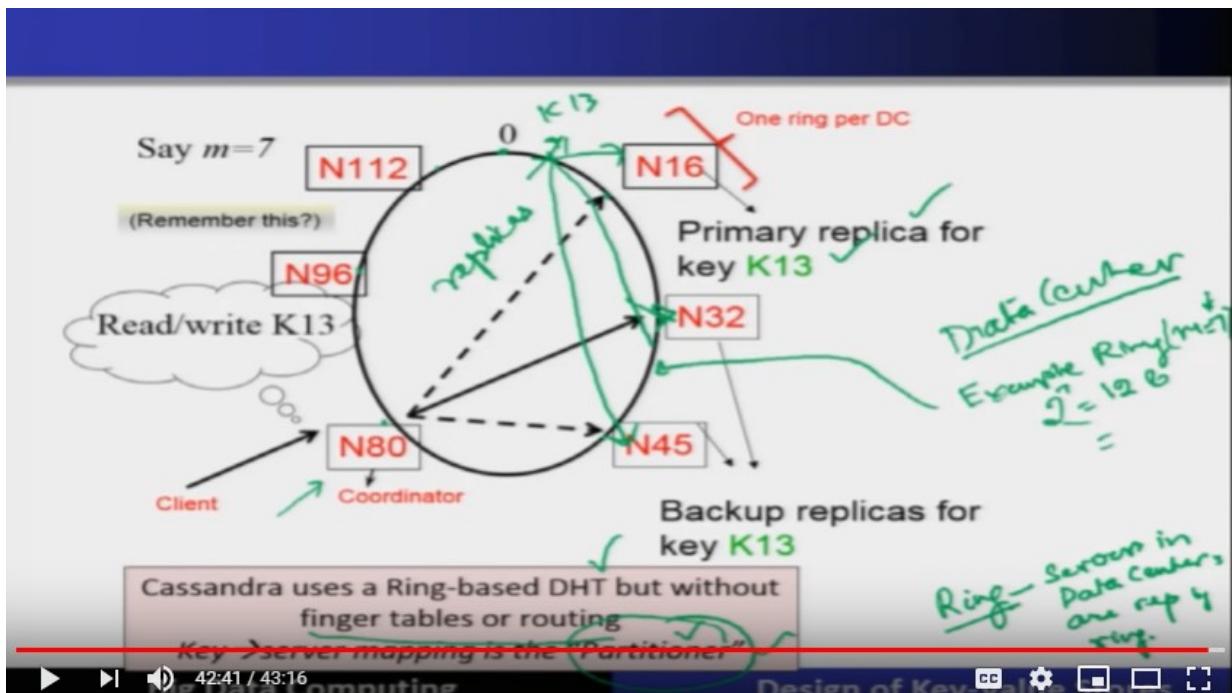
Inside Cassandra: Key -> Server Mapping

- How do you decide which server(s) a key-value resides on?



Let us see, the inside of Cassandra to see the inside of the Cassandra we will divide, the task which is designed inside the Cassandra, the first thing is the key the mapping between key to server mapping. So, the Cassandra supports, the key value abstraction, to support this key value abstraction, this key, to the server mapping, is the first most important task of the Cassandra design. So, that means now you know that this particular database is supported, on the clusters, this is called, 'Clusters', which has several nodes and this runs, the Cassandra. Now here for a particular key, which server really stores, that key with server stores this particular key is going to be very important and how we are going to do this mapping, we are going to see so how do you decide which server or a set of servers, basically stores that key value pair on it. So, to decide this

Refer slide time :(38: 32)



We are going to see the inside of Cassandra. So, Cassandra uses a ring based, distributed hash table, for doing this kind of mapping and key to the server mapping, is done by the concept which is called a, 'Partitioner'. We call it as partitioner so; let us understand what do you mean by the ring? So, the so, the nodes are the servers within the cluster or in a data center they are represented in a form of a ring so, for example here in this, in this case we are making a ring, of size $M = 7$, when $M = 7$ then $2^7 = 128$, different points will be there on the ring and on these points, we can place different servers. So, there will be so, this will be called as a, 'Ring', of all servers which are there in a data center. So, data center is represented as a form of a ring and if, if the number of servers are more than 128 then, M value will, will increase and so on. So, the ring depends upon how many servers are there and it will be organized in the form of a ring. So, so whenever there is a key, required to be stored on these particular servers which are organized in a form of way of a logical ring, then one of these particular server, will become a coordinator, to handle this key to the server mapping, with the help of a program which is called the partitioner.

So, that particular client even the help of partitioner it will decide, where this key is going to be stored or mapped, on which servers. So, it will follow for example key 13 it will follow, the servers following this K13 lies over here. So, the successor of K 13, will store this particular case for example the successor of K 13 will become N 16 it will store this key and another copy, of this key is stored on n 32 and another copy will be stored on N45. So, the copies are called, 'Replicas'. So, it's not called as a, 'Primary Replica' or a secondary replicas'. But they are called as a replica all the copies are same and that can be served this is called a, 'Partitioner'. So, this kind of mapping from, key to the server is called a partitioner, in these terms and this uses the concept of the, the ring based technology which is we have seen in distributed hash table in a peer-to-peer system. So, when's this peer-to-peer systems, that means all the nodes are same it's a pure a distributed system without having any client-server architecture in it. This ring which Cassandra uses, it differs from DHT in one form that it is only using the ring, of DHT and it is not using

the other concept of, distributed hash table such as finger table it is not using a routing is not using. So, Cassandra uses the ring without any routing within it.

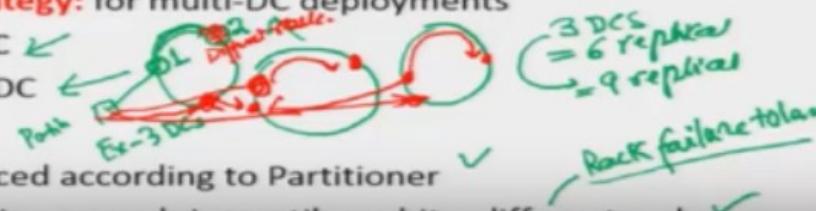
Module 16 - Lecture 13

Data Placement

Strategies

Refer Slide Time :(0: 13)

Data Placement Strategies

- **Replication Strategy:**
 - 1. SimpleStrategy ✓
 - 2. NetworkTopologyStrategy
- 1. **SimpleStrategy:** uses the Partitioner, of which there are two kinds
 - 1. RandomPartitioner: Chord-like hash partitioning ✓ *Randomly across servers*
 - 2. ByteOrderedPartitioner: Assigns ranges of keys to servers.
 - Easier for *range queries* (e.g., Get me all twitter users starting with [a-b])
- 2. **NetworkTopologyStrategy:** for multi-DC deployments
 - ✓ Two replicas per DC
 - Three replicas per DC
 - Per DC ✓
 - First replica placed according to Partitioner
 - Then go clockwise around ring until you hit a different rack ✓

Data placement strategies, replication strategy is of two types, simple strategy and network topology strategy. In the simple strategy uses the partitioner, of which there are two kinds of partitioner, one is called, 'Random Partitioner' and the other one is called, 'Byte Ordered Partitioner'. Random partitioner is caught like has, partitioning and where is the byte order partitioner assigns, the ranges of each to the servers and it is, easier for the range queries for example, let me call, let me call, all the twitter users starting with a to b. But, instead of going for a byte ordered partitioner, random partitioner, is a recommended partitioner, why because, random partitioner, partitions the keys, uniformly across all the servers and it is, more efficient to use the random partitioner, whereas the byte order partitioner , will have the various issues that after So much, of So many, number of storage and operations, what will happen it will form the hotspots, out of these right operations. So, hotspots will create a problems and again load-balancing has to be done and so on and so, forth. So, it is recommended that, the random partitioner is being used, in most of the cases, as far as the simple strategy is concerned. Now, another strategy for the replication is called a, 'Network Topology Strategy'. So, network topology strategy, for data placement, requires for multiple data multiple datacenter, deployments, are considers, multiple data center deployments. So, for example, network topology strategy will use two replicas per data center. So, if there are three data centers, then it requires, two data, two replicas per datacenter that means, six replicas, it will be there, across different data centers. Similarly when, there will be three replicas per data center and if we have, three different data center. So, nine different replicas will be there across all the data centers. Now, in per datacenter, replication or data placement, strategies will be doing that first replica; will be placed according to the partitioner and then go clockwise, around the ring, until you hit a different racks. So, is storing the replicas on a different rack, ensures, the rack failure tolerant. Right? Fully at all it ensures the rack failure tolerance. So, that means, what we have seen is that, when we go for the network topology strategy, we are, we are placing our replicas, in different data center, if there is a

multiple datacenter deployments or geographically distributed, data centers if they are, there then these replicas are separate or stored across these data centers and this particular things can be achieved using the partitioner, what partitioner will do? For example, there are different data centers, let us assume that in, in your multi data center deployments there are three data centers, which are shown in the figure. So, what it does is, the first replica; will be placed, the partitioner, will place the first replica here, first replicas will be placed according to the partitioner, let us, use the two replicas for data center. So and the other replicas will be, will be placed on the ring and the second replica; will be will be placed. So, that this replica one and two this replica one and two, they resides on a different rack. Similarly this partitioner will also, store on these two replicas on the other data center and also, on other datacenter. Now, you see that different datacenters have a ring and there will be a coordinator, which will coordinate among themselves. So, this particular partitioner has to, contact to these coordinators and they will in turn, not carry out, this network topology strategy-based, the, the data placement, in this manner.

Refer Slide Time :(6: 26)

Snitches

- **Maps:** IPs to racks and DCs. Configured in cassandra.yaml config file
- **Some options:** ✓
 1. • **SimpleSnitch:** Unaware of Topology (Rack-unaware) ✓
 2. • **RackInferring:** Assumes topology of network by octet of server's IP address
 - $101.102.103.104 = x.<\text{DC octet}>.<\text{rack octet}>.<\text{node octet}>$
 3. • **PropertyFileSnitch:** uses a config file
 4. • **EC2Snitch:** uses EC2.
 - EC2 Region = DC ✓
 - Availability zone = rack ↙
 - Other snitch options available

Now, the next issue, in this system is called the, 'Mapping of IPs' to the to the racks and data center. So, the first thing is, about key to the server. So, this we have seen key to the server mapping, in the previous slide. Now we are, going to see, the another design issue, which is called, how the IPs are to be mapped, to the racks and data center. So, the mapping of IPs, to the rack and datacenter, is done through the technique which is called a, 'Snitches'. So, this particular configuration, is can be done, in a file which is called a, 'Cassandra.yaml' configuration file and let us see, these technique, which is called the, 'Snitches' which does this mapping of IP to, the racks and data center. So, there are different type of

snitches the, the first type of snitch, is called the, 'Simple Snitch', simple snitch is unaware of the topology and it is rack in aware. So, that means it will, it will be done, in a manner, which will be a random, assignment and doesn't have any inference or doesn't have any information, about from looking from the IP addresses, where it is going to be mapped on, it's called, 'Simple Snitch'. The other type of snitch is called the, 'Rack Inferring Snitch' which assumes, the topology, of the network by an octet, of server IP addresses. So, for example, if this IP address is there, one zero one point, one zero two point, one zero three point, one zero four, this inverts that the, the first octet, is not going to be used for any inference, the second octet, represents the data center address and third octet, represents the rack octet and the fourth one is the node octet. So, if let us say that for example, if let us say the, the IP addresses, is let us say, $101.102.103.104 = \text{x}.\langle\text{DC octet}\rangle.\langle\text{rack octet}\rangle.\langle\text{node octet}\rangle$. 102 that means both these IP addresses, they are mapped to the same data center. Similarly we can see that, 102.103.104 that means within that data center, they are referring to the different racks and within the rack, if let us say that, the fourth octet becomes the, the node octet. So, this way, the IP addresses, will be able to infer about the topology, if it is, used in this signature which is called, 'Rack Inferring Snitch'. Another type of snitch is called the, 'Property File Snitch' which uses, the configuration file and fourth type of the snitch is sued, in the EC2 snitch and which is used in easy to type of storage system, that is the cloud storage system. So, easy to has, to kind of inferencing, one is the EC2 region, which will tell about the which data center is used to store the data, the other is called, 'Availability Zone' and which will tell, about which rack uses, which rack is used to store the data. So, if it is, used the easy to based the cloud system. There are many various other snitches options are available, so we have just introduced some of the most commonly used snitches.

Refer Slide Time :(11: 02)

Writes

- Need to be lock-free and fast (no reads or disk seeks) ✓
 - Client sends write to one coordinator node in Cassandra cluster
 - Coordinator may be per-key, or per-client, or per-query
 - Per-key Coordinator ensures writes for the key are serialized ↗
 - Coordinator uses Partitioner to send query to all replica nodes responsible for key
 - When X replicas respond, coordinator returns an acknowledgement to the client
 - X?
-

In used in the Cassandra. Now, we are going to see the operations, which are supported in the Cassandra in the form of write operations. So, as we have seen that, the writes has to be lightning-fast, writes which is supported by the Cassandra, because it is write a heavy, workloads. So, let us see how the Cassandra supports the write operations. So, the writes need to be log free and fast, hence it not require the right does not require, to be, to read or to perform a disk access, before performing the right operations and the cursor, the clients ends the right, to one of the coordinate or node in the Cassandra cluster, like we have seen, in the previous slide that one of the, one of the server, becomes the coordinator and the client, will send the right operation, to the coordinator and the coordinator, will in perform this right operation. So, coordinator may be a per key basis or a per client, basis or a per query basis, it depends upon different use cases and applications and all these variations, are possible for the coordinator. So, per-key coordinator ensures the right for, right for the keys are serialized. So, that means for per-key, coordinator will ensure that the rights, all the rights which performed, over that particular coordinator for per key basis, they are all serialized and this ensures, the consistency in some of the cases that we will see, in Some of the applications where it is, wherever it is required to be useful. The next, thing is about the coordinator uses the partitioner, to send the queries to all the replicas nodes, responsible for the key. So, this particular coordinator will now, send using partitioner, to send the queries, to all the replicas responsible for the key, where the keys are restored. Now, when X of these replicas, out of them, if X, if out of them, out of n, let us say n replicas and X replies back, what is this X number is when X replies or responds the coordinator returns an acknowledgment to the client. So, what is this value of X? Which is acceptable, that we will see in the further slides?

Refer Slide Time :(13: 58)

Writes (2)

- **Always writable: Hinted Handoff mechanism**
 - If any replica is down, the coordinator writes to all other replicas, and keeps the write locally until down replica comes back up.
 - When all replicas are down, the Coordinator (front end) buffers writes (for up to a few hours).
- **One ring per datacenter**
 - Per-DC coordinator elected to coordinate with other DCs
 - Election done via Zookeeper, which runs a Paxos (consensus) variant

Now, right operation we are seeing that, this way always writable, that means right operation, always is performed and which applies our using the hinted, handoff mechanism. So, what is this hinted handoff mechanism? To basically which uses, the which is used for the right operation that we are going to see now. So, if a replica is down, the coordinator rights to all our replicas, let us understand the same diagram that if this is a partitioner and these are all let us say there are three different replicas and if, this wants to write. So, if any of these replicas, let us say this replica is down, the coordinator rights to all other replicas and keeps the right, locally until the down replica comes up. So, for this, replica which is down, the coordinator in, coordinator himself, coordinator himself keeps, the values, at its end and whereas it will send it, to all other replicas, these values, if the replica is down, the coordinator rights to all other replicas and keeps the right, locally down, locally until the no replica comes up and backs up. When all the replicas are down, let us say then, then the coordinator, will buffer, all the right operations at this end, and, and wait for up to the few hour still, these replicas which are down, they may come up and they may get the updates. So, this is called, 'Hinted Handoff Mechanism' and which is used, to perform this always writable and irrespective of the replicas are down or not, some are down or all are down, even than the right, is performed and using the hinted handoff mechanism. So, we have explained that, what do you mean by always, right able. So, in this particular scenario using hinted handoff and of mechanism. Now, as far as, we know that, there is one ring per data center, So, per a datacenter coordinator, is elected to coordinate with other data center and this election is done via the zookeeper, which runs, a paxos, protocol. So, again we have, already seen and discussed for example, this is a three data center. So, every data center, will have a coordinator and they may these coordinators will communicate with each other. So, this coordinator if they are, they are elected, using the zookeeper and the zookeeper, zookeeper runs, a Paxos, consensus protocol.

Refer Slide Time :(17: 27)

Writes at a replica node

On receiving a write

1. Log it in disk commit log (for failure recovery)
2. Make changes to appropriate memtables
 - **Memtable** = In-memory representation of multiple key-value pairs
 - Typically append-only datastructure (fast)
 - Cache that can be searched by key
 - Write-back as opposed to write-through

Later, when memtable is full or old, flush to disk

- Data File: An **SSTable** (Sorted String Table) – list of key-value pairs, sorted by key
- SSTables are immutable (once created, they don't change)
- Index file: An SSTable of (key, position in data sstable) pairs
- And a Bloom filter (for efficient search)

Cache	Disk
memtable	SSTable

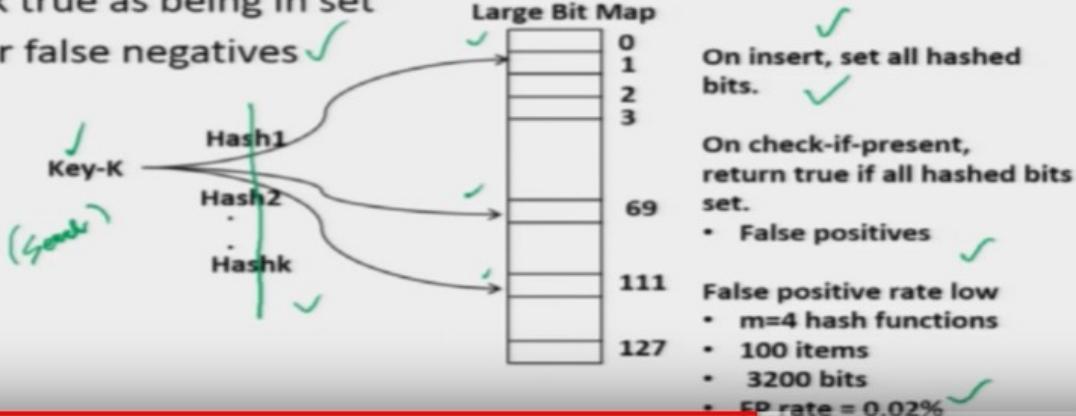
bloom filter for efficient search

Now right said the replica nodes, we are going to see the details of it. Now, when a right is performed, at the replica. So, it will be done in this following manner, on receiving a right, it will be first logged, in to the disk commit log, to ensure the reliability and in case, if it is a failure, of any kind then it can be recovered, from the failure, that is why, the it has to be first log into the disk commit log. Now then, it make changes to the appropriate Memtable. So, Memtable is an in-memory representation, of multiple key value pairs and typically, this particular Memtable, has append only data, structure therefore it is fast, to write, in the Memtable and there is a cache that can be searched, by the key. So, this is Okay? This is maintained in a cache and therefore it can be searched, by the key therefore these operations, writing and searching, is become fast, if it is, done through the Memtable. Now, in Memtable, the operation is, the right back, that means it will be stored, in this in memory Memtable and then later on, it will be, written on the on the disk, whereas another, approach which is called a, ‘write-through’ that means if it is, written to be on right, is to be done on the disk or to make a persistent, then basically it is going to be a time taken and that process is called, ‘Write-through’. But, Cassandra by default supports the right back. So, it writes into the map, into the Memtable and that's it. So, hence the write operations are lightning-fast, by this method, of using cash to, do the to, to write the multiple key value pairs. Now, when the Memtable becomes full or becomes old then automatically it will be flushed, to the disk at a later point of time. So, the data, file will be, stored in the in the form of, the SS table. So, when it is done, then these data, files are stored in the SS table which is nothing but, a list of key value pairs, which are sorted by the key. So, in the cache, the same thing is maintained that is called a, ‘Memtable’ and within the disk, this particular same task is maintained in the form of SS table. So, the data, file which is stored in a in an SS table, that is full form is called, ‘Sorted String Table’ is a list of, key value pairs and which are Sorted by the key. So, SStables are immutable, that is once created they don't change and they are also, maintained maintaining an index file, so that search becomes fast. So, index file is associated with a, with the SStable of which will indicate the key and the positions in the SS table pairs. So, that is maintained in the index, for make it the, the, the retrieval of faster one. Now, to make it, more efficient retrieval using SStable, a bloom filter is, introduced here in Cassandra, bloom filter provides an efficient search, through this SS table.

Refer Slide Time :(21: 28)

Bloom Filter

- Compact way of representing a set of items
- Checking for existence in set is cheap
- Some probability of false positives: an item not in set may check true as being in set
- Never false negatives ✓



Let us see, what do you mean by the bloom filter how, it enhances, the efficiency for searching. So, bloom filter, is a compact way of representing a set of items and in this data structure checking for the existence, in the set whether the element is present, in the set is cheap. So, the sum probability, of false positives are there that means an item, which is not, in there in the set may, may, may it turns out to be too sometimes. But, there will be an hour false a negatives. So, that means if, the item is not there, always it will say, no with 100% confidence, in some of the cases, even if the item, is not present, even then it will say sometimes, present that is called, 'False Positives' The working of, the bloom filter is like this. So, whenever a key, which required to be means searched, we want to search a key, through the bloom filter, then it has to, go through the k different hash, functions and every hash, function will generate a bit and this bit will be, then inserted onto the large bit map. So, that means, when it is written, that these bits are set and when it is searched, then using hash function it will check, whether all the bits are one, if it is, all the bits are one, then it will say the item is present, although if it is not, then it will be a false positive, if it is, zero sum in one of these bits then obviously, it will be false negative that means it is not inserted. So, when we insert, a key into this, then all these index, are the large bit map, will be set, at all the bits, all the key bits and if you want to check, if it is present or not, these hash bits are being checked, if it is 0 or 1 well if all are, zeros if some are zeros then, we will be basically it is not present and sometimes, these false positives, will happen and it will say, it is present, although it is not and if we see the performance, of this bloom filter, the false positive, rates are very, very less that is 0.02% is very, rare. So, that means it will give, that means it will check or search, with a high efficiency and happy the high probability,

Refer Slide Time :(24: 09)

Compaction

Data updates accumulate over time and SSTables and logs need to be compacted

- The process of compaction merges SSTables, i.e., by merging updates for a key
- Run periodically and locally at each server

it gives a correct method. Now then, coming back to the Cassandra' again, we have discussed that, that means data, updates will accumulate, over the time and SS tables and the log need to be compacted, compaction, will is a process, which will merge, different SSTables and locks also, together. So, the process of compaction, merges SSTable by merging updates for a key, it says that, the different SS tables, which are immutable, now required to be merged and all the data, has to be made persistent. So, that other operations becomes efficient. So, this particular compaction runs periodically and locally at each server.

Refer Slide Time :(25: 01)

Deletes

Delete: don't delete item right away

- Add a **tombstone** to the log
- Eventually, when compaction encounters tombstone it will delete item

Now, let us see how, the delete is supported. So, the delete operation do not delete the items right away rather it adds a tombstone, to the log and when the eventually, when compaction encounters a tombstone it will delete these items.

Refer Slide Time :(25: 17)

Reads

Read: Similar to writes, except

- **Coordinator can contact X replicas (e.g., in same rack)**
 - Coordinator sends read to replicas that have responded quickest in past
 - When X replicas respond, coordinator returns the latest-timestamped value from among those X
 - (X? We will check it later.)
- **Coordinator also fetches value from other replicas**
 - Checks consistency in the background, initiating a **read repair** if any two values are different
 - This mechanism seeks to eventually bring all replicas up to date
- **At a replica**
 - A row may be split across multiple SSTables => reads need to touch multiple SSTables => reads slower than writes (but still fast)

Now, let us see the read operation, which is supported in the Cassandra, read is similar to the right except, the coordinator can contact x replicas, may be in the same rack. So, the coordinator sends the read to the to the replicas that have responded quickest, in the past and when X replicas respond the, the coordinator returns the latest timestamp value from among those X replicas. So, we will see, what is the value of X, how we can, exploit this in the efficiency and the design of different application systems. Now, the coordinator also, fetches the values from, other replicas and it will check, for the consistency in the background, initiating read repair if the two values are different, this mechanism seeks, to eventually, bring up, all the replicas up to date. So, at the replicas, we have to see that a road row main, may be split across multiple, STable that means the, the reads need to touch multiple STable, sometimes and therefore, the reads become slower, than the writes but, still they are faster, compared to the other traditional RDBMS is. So, therefore that means, multiple SSTables, sometimes required, to be, to be accessed, for the, the read operation to be made successful that is why the competition Sometimes, is useful.

Refer Slide Time :(26: 54)

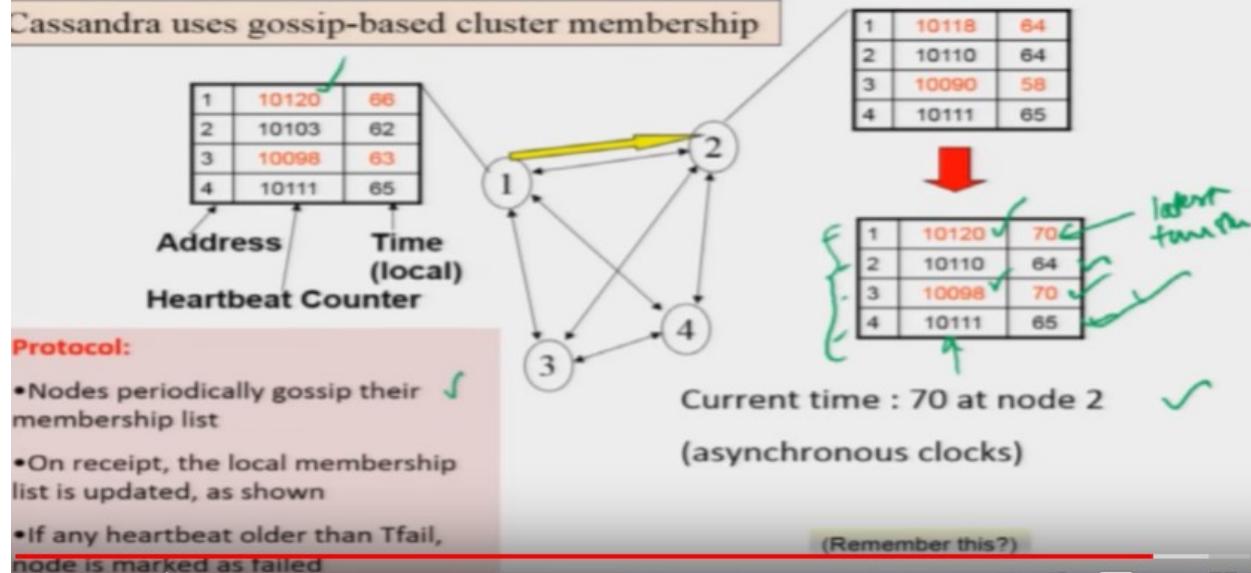
Membership

- Any server in cluster could be the coordinator
- So every server needs to maintain a list of all the other servers that are currently in the server
- List needs to be updated automatically as servers join, leave, and fail

Now, we will see the membership, membership in the sense that, the nodes, which are there in the form of a ring, they come and they go down. So, any server in the cluster could be a coordinator, so every, server needs to maintain a list of other servers that are current in the server. And this list need to be updated automatically, as the server join, leaves and fails.

Refer Slide Time :(27: 19)

Cluster Membership – Gossip-Style



So, the membership, cluster membership is maintained in the form of a gossip and this example, shows that, these hot beat or these details, are exchanged and, and then based on that, they are being updated. So,

here we can see that, this particular protocol here the nodes periodically gossip, their membership list and on receive the local membership list is updated, as shown over here. So, if any heartbeat is older than, T fail known, is mark as failed. So, here we can see that, once no 2 receives, this heartbeat, from node1, then it will try to compare, for example, it has, not seen this one zero, to zero. So, it will be now, added with the latest timestamp. Similarly 1, 1, 8 and 1, 1, 0 will be stored and this 1, 0, 1, 1 is there. So, all these are different heartbeat counters, this is heartbeat counter and these are the notes addresses. So, the note address, is 1, 2, 3, 4 they are being updated, with their heartbeat counts, since they have lost being seen over here. So, the node number1 has more recent, heartbeats over here. So, that value, of the heart weight is updated and similarly for the node number 2, the value is not updated. So, because it has, the information about the, the recent information. So, it is not updated, similarly for node number 3, it is, 1, 0, 98 and this information is more recent so, 1, 0, 98 is updated, with a, with a recent timestamp and the node number 4, is not going to be changed here in this case. So, this requires the, the current clock, time and the current clock time, will be used to update the, the entries. So, this way, the membership so, the node 2 knows that, in this topology, there are 4 different nodes, which are working and they are exchanging their heartbeats and this is called the, 'Cluster Membership and the Cassandra uses, the gossip based cluster membership. Now, suspicion mechanism, which is there in Cassandra.

Refer Slide Time :(30: 11)

Suspicion Mechanisms in Cassandra

- Suspicion mechanisms to adaptively set the timeout based on underlying network and failure behavior
- **Accrual detector:** Failure Detector outputs a value (PHI) representing suspicion
- Applications set an appropriate threshold
- **PHI calculation for a member**
 - Inter-arrival times for gossip messages
 - $\text{PHI}(t) = -\log(\text{CDF or Probability}(t_{\text{now}} - t_{\text{last}})) / \log 10$
 - PHI basically determines the detection timeout, but takes into account historical inter-arrival time variations for gossiped heartbeats
- In practice, $\text{PHI} = 5 \Rightarrow 10\text{-}15 \text{ sec detection time}$

So, suspicions, suspicion mechanism to adaptively, set the time out based on underline network and the failure behavior. So, accrual detector is the failure detector which outputs, the value of PHI, which is representing the suspicion and application set and appropriate threshold Phi, will indicate the calculation for a member. So, it will be, internal times for the gossip, messages and in practice the, this threshold, is of PHI that is, it's 10-15 seconds off in detection time.

Refer Slide Time :(30: 50)

Cassandra Vs. RDBMS

- MySQL is one of the most popular (and has been for a while)
- On > 50 GB data
- **MySQL**
 - Writes 300 ms avg
 - Reads 350 ms avg
- **Cassandra**
 - Writes 0.12 ms avg
 - Reads 15 ms avg
- Orders of magnitude faster
- What's the catch? What did we lose?

Now, let us compare the Cassandra with RDBMS. So, my sequel is one of the most popular, RDBMS as on date and if, it is more than, 50 GB of data, let us see the, the performance, in my sequel, the writes are taking 300 milliseconds on an average heat, is taking 350 milliseconds on an average, if we compare it with, Cassandra where writes is, lightning fast, that is, it will be 0.12 milliseconds on an average, similarly reads are also very, fast that is 15 milliseconds on an average. So, orders of magnitude this Cassandra, provides these operation writes and reads much faster. Now, let us compare, what is the catch? What is the difference? How it is going to achieve, such a fast writes and better reads and what is the catch and where, where we where the Cassandra is losing, compared to the RDBMS that we are going to see using, the CAP Theorem.

Lecture – 14

CAP Theorem

Cap theorem.

Refer Slide Time :(0: 16)

CAP Theorem

- Proposed by Eric Brewer (Berkeley)
- Subsequently proved by Gilbert and Lynch (NUS and MIT)
- In a distributed system you can satisfy atmost 2 out of the 3 guarantees:
 1. **Consistency:** all nodes see same data at any time, or reads return latest written value by any client ✓
 2. **Availability:** the system allows operations all the time, and operations return quickly
 3. **Partition-tolerance:** the system continues to work in spite of network partitions

(Eric Brewer's 3 almost 2 guarantees by system design issue)

Cap theorem was proposed by Eric Brewer, in Berkeley and which was subsequently proved by Gilbert and Lynch. In a distributed systems you can specify, at most two out of three different guarantees; which is now, specified as, which is known as, the cap theorem. Let us see, what these three different guarantees? Three different things are. So, the first one is called, 'Consistency', the second one is called, 'Availability', third one is called, 'Partition tolerance'. CAP theorem says that, out of these three at most two can be the guaranteed, by any systems. So, let us see the, consistency that means all the nodes see the same data, at any time or the read returns the latest return value, by any client. So, that is called, 'Consistency', that means at all points of time, the node will, get the most recent data, at any point of time whenever it is being, referred or it is being accessed, by the read operation. So, whenever the read operation is performed, it will, read it will return the latest write, by the client, if that is, guaranteed at all points of time, then it is called, 'Consistency'. Availability says that the system allows operation, all the time and operations returned quickly. Meaning to say that, the operations, the system always is operating and whenever is being accessed, it will perform, it will return, it will return the operations, very quickly. So, this is called, 'Availability'. Third criteria, is called, 'Partition Tolerance' that is the system continues to work in spite, of network partitions. So, the cap theorem says that, out of three, out of these three. So, cap, c a p that means consistency availability and partition tolerant, in short, it is called, 'Cap Theorem'. Cap theorem says that, out of three, out of these three different parameters, at most, two can be guaranteed, at any time, by the system. So, this is, the design issue that we are going to see, how what is the implication of this cap theorem or different NoSQL systems which are being available, at this point of time.

Refer Slide Time :(3: 26)

Why is Availability Important?

- **Availability** = Reads/writes complete reliably and quickly.
- Measurements have shown that a 500 ms increase in latency for operations at Amazon.com or at Google.com can cause a 20% drop in revenue.
- At Amazon, each added millisecond of latency implies a \$6M yearly loss.
- **User cognitive drift:** If more than a second elapses between clicking and material appearing, the user's mind is already somewhere else → Churn in customer in any business →
- SLAs (Service Level Agreements) written by providers predominantly deal with latencies faced by clients.

Now, let us see, all these three different things, which is specified in the cap theorem that is the first one islet us, understand about the availability and what is the importance of availability? Availability says that read and write will complete reliably and quickly, at all point of time. So, in if we measure, these are read and write operation times and let us see that, this particular measurement will have an increase, of 500 millisecond, latency then let us see, what is the implication of most of the operations in the company. So, this latency, of read and write of 500 milliseconds, it is shown that, for the companies like Amazon or a Google, highest cost, drop of 20% in the revenue model, meaning to say that, so Amazon.com, is dealing with the sales of these items. So, if there is latency, of 500 milliseconds, in these performance, then obviously customers will churn and therefore they will drop in the revenues. So, at imagine each added millisecond, of latency implies, implies a six million yearly loss. So, a user cognitive drip; is there and if more than a second he lapses, between the clicking and the material appearing, the users mind is already somewhere else and this will lead to a churn, in the customers, in any business. So, this is a, most important parameter in some of the cases. So, the companies, like online which are dealing with the e-commerce online sales, that is imagine company or the Google, which taps the advertisements and other such product which are and services online, has to ensure, high availability and not only reliability but, it has to be very, quickly that is the latency also has to be very minimal. So, that the users can get the services. So, therefore a service level agreements, written by the providers, predominantly deal with the latencies, which are faced by the clients and therefore the availability is also, one of the most important parameter or criteria, in the cap theorem.

Refer Slide Time :(6: 11)

Why is Consistency Important?

- **Consistency** = all nodes see same data at any time, or reads return latest written value by any client.
- When you access your bank or investment account via multiple clients (laptop, workstation, phone, tablet), you want the updates done from one client to be visible to other clients.
- When thousands of customers are looking to book a flight, all updates from any client (e.g., book a flight) should be accessible by other clients.

Now, second criteria, which is called a, ‘Consistency’, of a CAP theorem C of that is called, ‘Consistency’. Consistency says that, all the nodes, see the same data, at any time are or reads returns, the latest return values by the client. Meaning to say that, the nodes whatever updates are happening, on the nodes by different write operations. So, they are updated, they are up-to-date. So, whenever read is requested, it will always return, the latest write operations which are done by the any client. So, reads are very latest, returns read returns the very latest, information which are written by the client, if that is maintained at all points of time that is called, ‘Consistency’. Now, when you access your bank account or investment account, by multiple clients wire, laptop, work station, phone, tablets, excess price you want these updates to be done from, one client to be visible to the other clients. It's not that, if you have done through the mobile phone and your mobile phone, updates are and you are, you are now, accessing or referring reading wire, laptop, they may, not get that a decent update, if that is the case, then it is not a consistency. So, similarly when a thousands of customers are looking to book a flight ticket, all the updates from a client, should be accessible by the other client, in the reservations, airline reservation, to booking a tickets. So, consistency also, is going to be an important parameter.

Refer Slide Time :(7: 49)

Why is Partition-Tolerance Important?

- Partitions can happen across datacenters when the Internet gets disconnected
 - Internet router outages
 - Under-sea cables cut
 - DNS not working
 - Partitions can also occur within a datacenter, e.g., a rack switch outage
 - Still desire system to continue functioning normally under this scenario
- Partition tolerance ensures the functioning even in the face of network failures.*

Now, another parameter, which is there in CAP, P stands for the partition. So, partitions can happen across datacenters, when the internet gets disconnected. So, this can happen by way of, Internet routages, internet router outages, are undersea cable cuts or DNS not working, in all these scenarios, you will see that, the network gets partitioned and the entire data center, some of the data, centers may be it is connected and partition. So, partition tolerance, partitions can occur, within the data center, within the rack or switch outages and so on. So, rack switches also, can basically ensure or induce these partitions, in tune to the network. So, still the, the desire system, the desire is that system to continue functioning normally under, the partition that is why? It is called, ‘Partition Tolerance’. Partition tolerance, ensures the functioning, even in the face, of partitions or it is also called as, ‘Network Failures’.

Refer Slide Time :(9: 24)

CAP Theorem Fallout

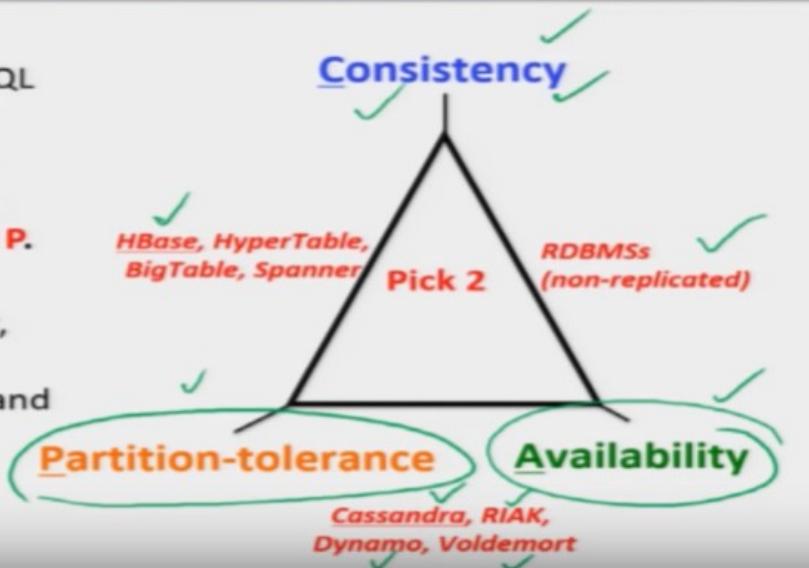
- Since partition-tolerance is essential in today's cloud computing systems, CAP theorem implies that a system has to choose between consistency and availability
- **Cassandra** ✓
 - Eventual (weak) consistency, Availability, Partition-tolerance
- **Traditional RDBMSs** ↙ CAP ↘
 - Strong consistency over availability under a partition

Now, the cap theorem fallout, since the partition tolerant is essential in today's cloud computing systems, the cap theorem implies that a system has to choose between the consistency and the availability. Now, we will see the Cassandra. So, Cassandra uses, eventual that is called a, 'Eventual Consistency' which is also, a weak form of consistency and then it ensures, it also uses the availability and partition tolerance. So, it Cassandra uses, A and P, of a cap, where C is being compromised or that is called, 'Eventual Consistency' or a 'Weak Form of Consistency'. Similarly the traditional databases management system, we see that, provides the strong consistency that, is it provides C and or availability, under the partition since, the since the database, is on the same system hence, the partition is already, partition will not happen hence the cap is, always satisfied or guaranteed, in the traditional RDBMS systems.

Refer Slide Time :(10: 40)

CAP Tradeoff

- Starting point for NoSQL Revolution
- A distributed storage system can achieve **at most two of C, A, and P.**
- When partition-tolerance is important, you have to choose between consistency and availability



So, CAP trade-off let us see that, starting point for this NoSQL system, the cap trade-off has to be, there so, in a distributed storage system can achieve, at most two out of three, at most two out of this consistency availability and partitioning. So, when the partition tolerance is important, then you have to choose between consistency and availability. So. Same thing is happening in the Cassandra system. So, we have to choose, between consistency and availability, so Cassandra chooses availability and a weaker form of consistency, in Cassandra. Similarly in react and dynamo and Voldemort, also uses the partition tolerance and availability with a, with a weaker form of consistency. Now, as far as RDBMS is concerned. So RDBMS, which is not replicated, which ensures the consistency and availability and there is no partition hence, consistency and availability are ensured in RDBMS is now, HBase, prefer reds partition tolerance and consistency or availability. So, we will see that, HBase and hyper table, Big Table and span and uses the, the consistency and partition tolerance or availability.

Refer Slide Time :(12:04)

Eventual Consistency

- If all writes stop (to a key), then all its values (replicas) will converge eventually.
- If writes continue, then system always tries to keep converging.
 - Moving “wave” of updated values lagging behind the latest values sent by clients, but always trying to catch up.
- May still return stale values to clients (e.g., if many back-to-back writes).
- But works well when there are a few periods of low writes – system converges quickly.

wave front of latest values converging



Let us see the, eventual consistency, which is a weak form of consistency. Which is supported in the Cassandra? Let us go and detail see, how the Cassandra is trading off with the consistency, to ensure, the partition, tolerance and the availability. Now, if all the writes stopped, to a particular key, then all its values in the replicas will eventually converge. meaning to say that, if Some of the rights, do not take place or do not get updated at Some of its replicas, then those replicas, will be updated or will become consistent after Some point of time. Hence, they will be eventually converge. So, if the write continues then system always tries to keep converging. So this way, the moving wave, of the updated values, moving wave of updated values, lagging behind the latest values, sent by the, by the client. So, these waves will move over the time. So, that says that, moving wave, of an updated values, will be lagging behind the latest values sent by the client, but always tries to catch up. So this difference will always be there. Now if, the number of new values are not coming, then obviously, eventually it will become the latest one, after eventually it will, be converging it. So if, the write continues then, the system tries to, tries to keep converging and this may still, return some of the stale values to the client, if many back to back write operations are there, but it will work fine when, there are a few, periods of low right's and So, the system can converge quickly.

Refer Slide Time :(14:34)

RDBMS vs. Key-value stores

- While RDBMS provide **ACID**
 - Atomicity
 - Consistency
 - Isolation
 - Durability
- Key-value stores like Cassandra provide **BASE**
 - Basically Available Soft-state Eventual Consistency
 - Prefers Availability over Consistency

Now, let us, compare the RDBMS with the key value stores. So, So RDBMS provides the acid properties, atomicity, consistency, isolation and durability, whereas key value store like Cassandra provides, the base property. BASE is basically available, soft state, eventual consistency. So basically available means, there, they are it provides the availability insurance and Soft state eventual consistency, Soft state that we have seen, in the form of cash caching, in in-memory operations, most of that the table properties are stored in cash and that is in the form of Memtable and eventual consistency means, finally everything, will be updated, that is called, 'Eventual Consistency', not immediately but, eventually it will be updated.

Refer Slide Time :(15:37)

Consistency in Cassandra

- Cassandra has **consistency levels**
- Client is allowed to choose a consistency level for each operation (read/write)
 - **ANY:** any server (may not be replica)
 - Fastest: coordinator caches write and replies quickly to client
 - **ALL:** all replicas
 - Ensures strong consistency, but slowest
 - **ONE:** at least one replica
 - Faster than ALL, but cannot tolerate a failure
 - **QUORUM:** quorum across all replicas in all datacenters (DCs)
 - What?

So, it prefers the availability or the consistency in the base model. Now, let us see the consistency levels which are supported in the Cassandra. Cassandra has different consistency levels. So, the client is allowed to choose the consistency level, for each operation, that read and write, among these different consistency levels. So they are, one that is any, second type of consistency level is all, third is one and fourth is quorum. Let us, understand one by one all these consistency level. So any, consistency level, by mean, any consistency level, that means, any server may not be the replica is can, can allow, this operation to be complete. So this becomes a fastest, because the coordinator caches, the write operation and returns quickly to the, to the client, to perform this read and write operations hence, this is the fastest one. The second one is called, 'All Replicas'. That means, it ensures that the read and write operations, write operations requires, to ensure that, it has to be updated at all the replicas. So this is a kind of strong consistency, so it will provide, if the consistency level is all then it will be as the slowest one. Third one is called 'One', that is at least one of these replicas gets updated, so it is faster than all, but it is slower there any, why because it cannot tolerate the failure of all, of the replicas. And fourth one is called, 'Quorum', quorum says that, quorum that means, a quorum across all the replicas in the data center is to be updated, what do you mean by the quorum? That we are going to see? So quorum is between the all and one, so quorum is some number K. So, we are going to see how many replicas are required to be updated, under the quorum system.

Refer Slide Time :(17:41)

Quorums for Consistency

In a nutshell:

- Quorum = majority ✓
 - > 50%
- Any two quorums intersect✓
 - Client 1 does a write in red quorum
 - Then client 2 does read in blue quorum
- At least one server in blue quorum returns latest write
- Quorums faster than ALL, but still ensure strong consistency

A quorum A second quorum
A server

Five replicas of a key-value pair

Examp 5 = ③

So quorum's for consistency, quorum says that majority, so if there are in this example, there are five different replicas, so the quorum says, the majority means, more than fifty percent, so that, becomes the three. So minimum at least three different quorums, different replicas are to be updated. So, for any to the proper these upper quorum, which has to be satisfied is that, if any to quorum's, if we take the intersection then there must be a common, replica common servers between any two system. So, any two quorum vary intersect, client one does the write operation, in the write quorum and the Client two, reads

from the blue quorum, then it will get the update from, because there is a common server in both the quorum's, So at least one server in the blue quorum, gets the latest write? So quorums are faster than all, but ensure the strong consistency.

Refer Slide Time :(18:46)

Quorums in Detail

- Several key-value/NoSQL stores (e.g., Riak and Cassandra) use quorums.
- **Reads**
 - Client specifies value of **R** ($\leq N$ = total number of replicas of that key).
 - **R** = read consistency level.
 - Coordinator waits for **R** replicas to respond before sending result to client.
 - In background, coordinator checks for consistency of remaining ($N-R$) replicas, and initiates read repair if needed.

So quorum's, let us see in more detail, So several key value pairs, that is NoSQL to react and Cassandra uses, the quorum. So reads; that is the client specifies the value of R, which is the value of the quorum, which is less than the number of replicas, So R is the read consistency level, So coordinator waits for R replicas to respond before sending, the result to the client. In the background, the coordinator, checks for the consistency of remaining and - R replicas and initiate, the read repair if any. So meaning to say that, not only it the, the read is being satisfied by, specifying the R number of replicas, but what about the, the remaining replicas, that is N-R, whether are they consistent or not that also required to be checked, for the consistency, of the remaining replicas. So that has to be done in the background and if they are inconsistent then, the read repair is to be initiated, So that eventually they may be consistent at all the levels.

Refer Slide Time :(19:55)

Quorums in Detail (Contd..)

- Writes come in two flavors
 - Client specifies W ($\leq N$)
 - W = write consistency level.
- Client writes new value to W replicas and returns. Two flavors:
 - Coordinator blocks until quorum is reached.
 - Asynchronous: Just write and return.

So the reads sometimes, does the read repair, to ensure the eventual consistency. now, quorum's the write come in to flavor, when a client writes $W(\leq N)$, then write replication, then write consistency level, we have to specify. So the client writes, a new value to W replicas turn, there are two flavors. So the, coordinator blocks, until the quorum is reached or the it will be in the Asynchronous, that means, it will just write and return, returns back.

Refer Slide Time :(20:34)

Quorums in Detail (Contd.)

- R = read replica count, W = write replica count
- Two necessary conditions:
 1. $W+R > N$
 2. $W > N/2$
- Select values based on application
 - **($W=1, R=1$):** very few writes and reads
 - **($W=N, R=1$):** great for read-heavy workloads
 - **($W=N/2+1, R=N/2+1$):** great for write-heavy workloads
 - **($W=1, R=N$):** great for write-heavy workloads with mostly one client writing per key

- Now, if let us say, R is the replica, read replica count and W is the Write replica count, then there are two necessary conditions in the quorum,

1. $W+R > N$
2. $W > N/2$

Select values based on application

- ($W=1, R=1$): very few writes and reads
- ($W=N, R=1$): great for read-heavy workloads
- ($W=N/2+1, R=N/2+1$): great for write-heavy workloads
- ($W=1, R=N$): great for write-heavy workloads with mostly one client writing per key

Refer Slide Time :(21:54)

Cassandra Consistency Levels (Contd.)

- Client is allowed to choose a consistency level for each operation (read/write)
 - ANY: any server (may not be replica)
 - Fastest: coordinator may cache write and reply quickly to client
 - ALL: all replicas
 - Slowest, but ensures strong consistency
 - ONE: at least one replica
 - Faster than ALL, and ensures durability without failures
 - QUORUM: quorum across all replicas in all datacenters (DCs)
 - Global consistency, but still fast
 - LOCAL_QUORUM: quorum in coordinator's DC
 - Faster: only waits for quorum in first DC client contacts
 - EACH_QUORUM: quorum in every DC
 - Lets each DC do its own quorum: supports hierarchical replies

So, we have seen the quorum's, across all the replicas, in all the datacenters and it ensures the global consistency, which is still a fast one. So local quorum, So Coulomb's in a in, in the coordinated data center are faster, only waits for the quorum in the first datacenter client contacts, each quorum's, the quorum in every data center, let each data center do its own quorum and support the heretical replies.

Refer Slide Time :(22:23)

Types of Consistency

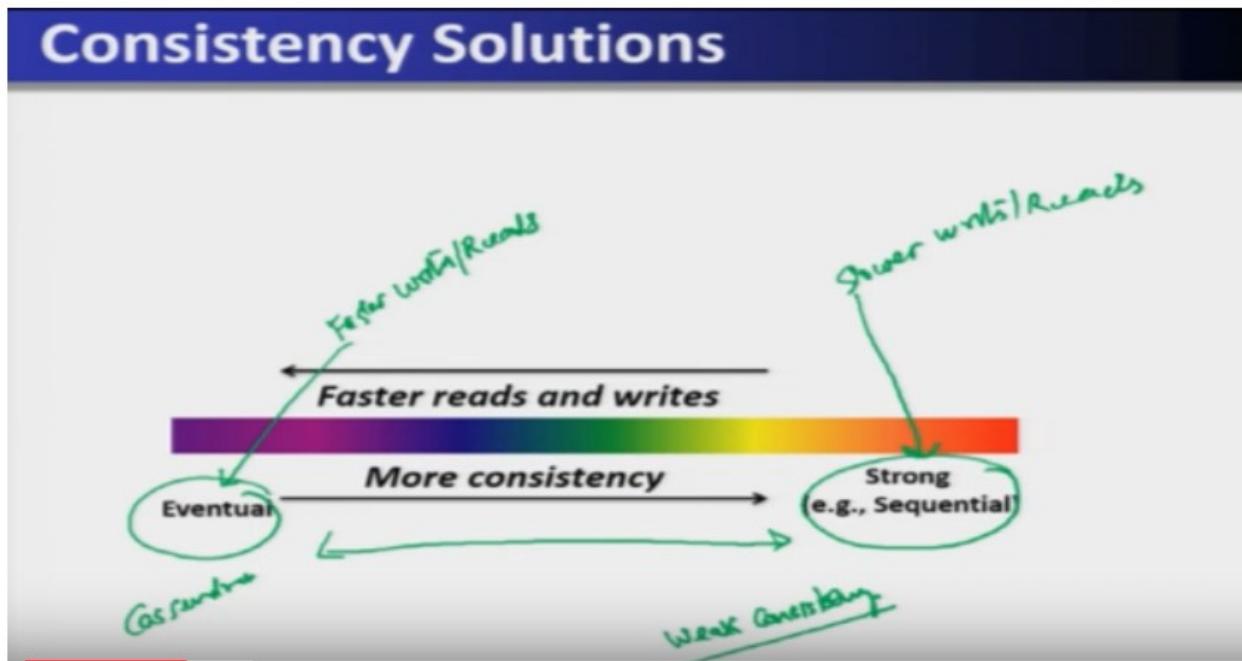
- Cassandra offers **Eventual Consistency**
- Are there other types of weak consistency models?

So the type of consistency, which Cassandra provides is called, ‘Eventual Consistency’. What are the other weak, forms of consistency models, which are available in practice? That we will see, in the, in the next slide.

Lecture 15
Consistency Solutions

Consistency solutions

Refer slide time :(0:15)

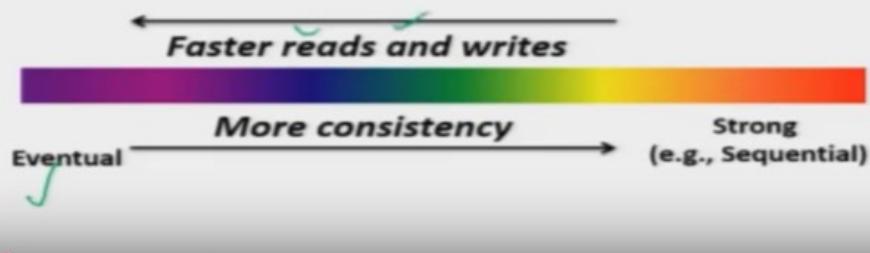


So, we will see the spectrum of different consistency solutions, which are available, under the weak consistency model. And we have seen that eventual consistency, which is supported by the Cassandra, is that one extreme level and strong consistency, at the other extreme level, in between what are the other consistency, models are there. So, before that we, we have to understand, when to use these consistency solutions, now if you want to do a faster read and writes, obviously the eventual consistency, is the good enough for the faster, faster writes and reads, similarly if we require, the achieve the high consistent high, consistency or strong consistency, then we will be having a slower, writes, and reads. Now in between, we will see the other forms of consistency and the performance varies, from strong consistency, to the weak consistency that means we have to ensure, what kind of reads and write whether it is faster or it is slower, read and writes are required.

Refer slide time :(01:50)

Eventual Consistency

- Cassandra offers **Eventual Consistency**
 - If writes to a key stop, all replicas of key will converge *eventually*
 - Originally from Amazon's Dynamo and LinkedIn's Voldemort systems

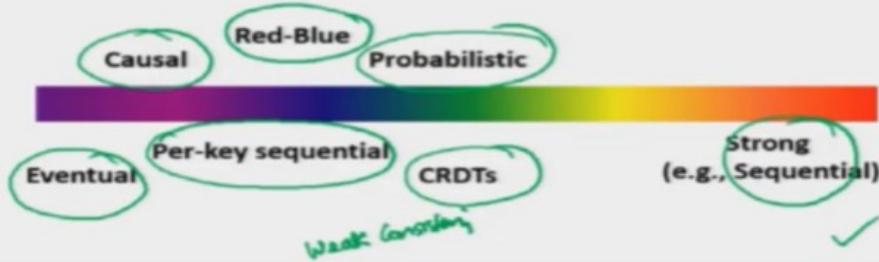


So, eventual consistency Cassandra, offers eventual consistency if, if writes to a key stops all the replicas of a key will converge eventually, originally this eventual consistency, was inspired by, inspired from a Amazon's Dynamo and LinkedIn's Voldemort system, in into the Cassandra system into the Cassandra's design. So, here we have to see that strong eventual consistency supports faster reads and write operations.

Refer slide time :(02:34)

Newer Consistency Models ✓

- Striving towards strong consistency
- While still trying to maintain high availability and partition-tolerance

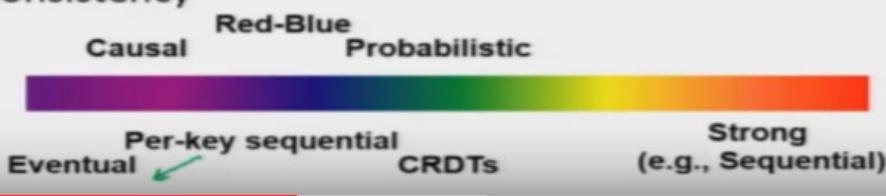


And let us see between the, the eventual consistency and between, the strong consistency, what are the other weak form of consistency, consistency available they are called, 'Newer Consistency', model. So, is striving towards a strong consistency is, very much needed, but different applications, but while still trying to maintain high availability and a partition tolerance, forces to go for some weak consistency models. So, there are different models, which are shown over here, they are called, 'Per-key sequential', consistency model then, then red blue consistency model, then CRDTs and causal consistency model, probabilistic let us see this newer consistency model, how they are now trading the consistency and giving the,

Refer slide time :(03:37)

Newer Consistency Models (Contd.)

- **Per-key sequential:** Per key, all operations have a global order
- **CRDTs (Commutative Replicated Data Types):** Data structures for which commutated writes give same result [INRIA, France]
 - E.g., value == int, and only op allowed is +1
 - Effectively, servers don't need to worry about consistency

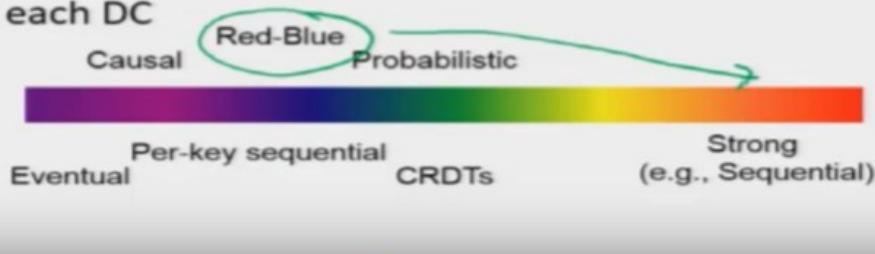


the performance required by the applications. So, as far as the, per-key sequential consistency model, since that per-key all the operations, have the global order. So, this ensures a performance, better than the eventual consistency, because it ensures the, the strong consistency notion, per key bases at global level. Now another type of consistency is called , 'CRDTs', that is commutative replicated data types, here the data structure for which the commutated writes, gives the same result this was given by the INRIA, France for example, if it is if the value ==int and only operation allowed is +1. So, whether the two writes, whether one has happened before the other it doesn't make any difference why because it is only doing the plus operation, +1 operation, increment operation effectively server do not need to worry about, the consistency if it is the commutative, operations commuting, write operations in CRDTs.

Refer slide time :(04:57)

Newer Consistency Models (Contd.)

- **Red-blue Consistency:** Rewrite client transactions to separate operations into red operations vs. blue operations [MPI-SWS Germany]
 - Blue operations can be executed (commutated) in any order across DCs
 - Red operations need to be executed in the same order at each DC

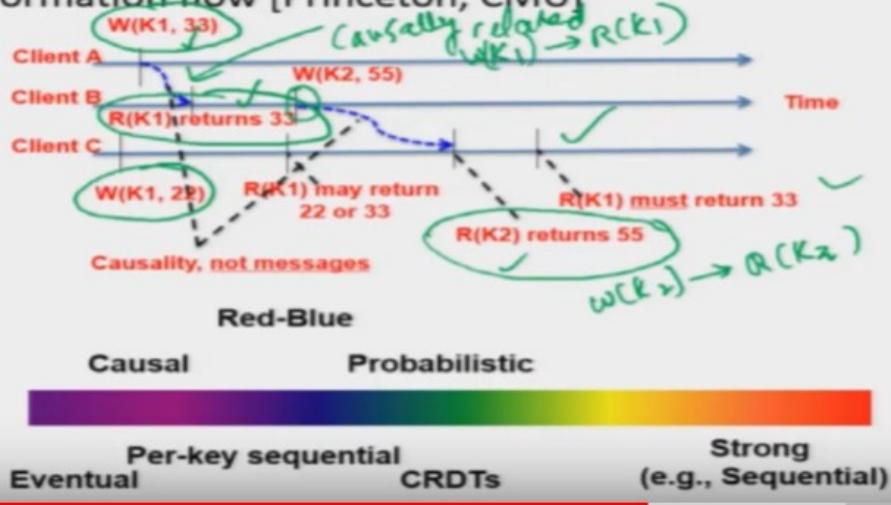


Now all the operations cannot be commutative. So, therefore in the, in the red-blue consistency, it will divide into two different type of operations, as per commutative and the other operations. So, it will rewrite the client's transaction two separate, operations into the red operations, that is vs. blue operations, blue operations can be executed, commutated in any order across datacenter, whereas the red operations, need to be executed in the same order in the data center. So, this classification, of these operations into the blue and red ensures, the red-blue consistency and therefore will make this particular, approach that is the red-blue consistency a much faster than eventual means, it will ensure a strong, much better consistency level, compared to the eventual consistency.

Refer slide time :(06:03)

Newer Consistency Models (Contd.)

Causal Consistency: Reads must respect partial order based on information flow [Princeton, CMU]

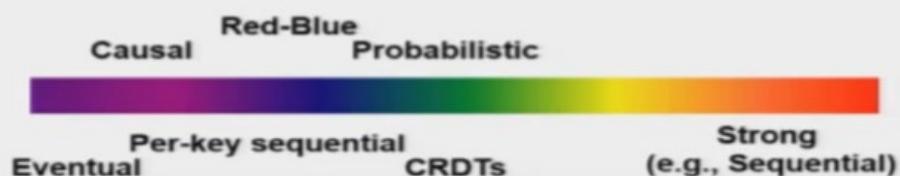


Now causal consistency, causal consistency deals, with the another weak form of consistency, which says that the deed must respect the partial order, based on the information flow, for example here we have seen that the client a $W(K1, 33)$ whereas, the client B will read this value of K, after some point of time $k1$. So, it should return the value 33 so, obviously this right of a client 1, is causally related. So, client this key writing of key $k1$, is causally related to the read operation of K, key one and this there is no message exchange, but internally we have to see this kind of causal dependency, exist similarly, you see that there is a causal flow similarly, between client B, which reads between client B and client C, which reads this value of $R(K2)$, which is written by the client B, at this point of time also are causally related. So, that means right of K key 2, of a key K 2, is happened before the read of key K 2, in this case. So, this particular $R(K2)$, returns the value, which is written by the client B on key K 2 and similarly when, the client C performs, $R(K1)$ it must return using this causal path, the value 33 which, is written by the client a not, the other value, which is the old value, which is written s 22. So, the causality where the messages, are not exchanged even then the causality is protected, then it is called, 'Causal Consistency'. So, this kind of consistency also is a very fast, notion of providing read and write, although they are ensuring some, some strong notion of consistency.

Refer slide time :(08:34)

Which Consistency Model should you use?

- Use the lowest consistency (to the left) consistency model that is “correct” for your application
 - Gets you fastest availability



So, which model, of consistency should be used, since thus the lowest consistency to the left, that is the consistency model that is the correct for your application, gets you the faster availability in this particular scenario.

Refer slide time :(08:49)

Strong Consistency Models

- **Linearizability:** Each operation by a client is visible (or available) instantaneously to all other clients
 - Instantaneously in real time
- **Sequential Consistency [Lamport]:**
 - "... the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program."
 - After the fact, find a “reasonable” ordering of the operations (can re-order operations) that obeys sanity (consistency) at all clients, and across clients.
- Transaction ACID properties, example: newer key-value/NoSQL stores (sometimes called “**NewSQL**”)
 - Hyperdex [Cornell] ✓
 - Spanner [Google] ✓
 - Transactional [Microsoft Project] ✓

NewSQL
- NoSQL + ACID

Now what is the strong consistency morals, let us quickly review them. So, linearizability is the strong consistency model, where each operation, by the client is visible, to all other it just like storing the data, on a single system, where whenever one data is modified it is available to all the clients instantaneously. And in real time, if that is there then it is called, 'Linearizability', is very difficult to achieve in the distributed systems, another form of strong consistency, is called, 'Sequential Consistency', which says that the result of any execution, is the same as if the operations of all the processors, were executed in some sequential order and operations of each individual processor appeared, in this sequence in the order is specified by, by its program, meaning to say that although it is not linearizability approach, but the sequential consistency, says that some order, which is which is to be executed in some sequential order, across all the clients, then if it is unsure then it is called, 'Sequential Consistency', it is given by the lamp or so, after the fact find reasonable ordering of operations, can be reorder operations that obeys the consistency, at all the clients across the clients. So, this is again, again another form of strong constraint that is a sequential consistency that that means all the clients will see the same, kind of the operations, by all the clients called sequential consistency. Now transactions, which are basically following the acid properties and whether the newer, key value, stores that is NoSQL store sometimes called, 'New Sequel', which also ensures the acid transaction properties. So, the nowadays, the application which required, also the acid property to be followed, besides all the properties, which we have seen in the NoSQL stores, if both are in addition, to that ACID is also required and they are called a, 'New SQL Systems'. So, for example new SQL is now a days is also available, in the form of high products given by the Cardinal and the Spain or a database management system, using by the Google and transaction chain by the Microsoft research and they call it as new SQL which supports, besides whatever is there in NoSQL, plus ACID properties, to support the transactions.

Refer slide time :(11:53)

Conclusion

- Traditional Databases (RDBMSs) work with strong consistency, and offer ACID
- Modern workloads don't need such strong guarantees, but do need fast response times (availability)
- Unfortunately, CAP theorem
- Key-value/NoSQL systems offer BASE
[Basically Available Soft-state Eventual Consistency]
 - Eventual consistency, and a variety of other consistency models striving towards strong consistency
- We have also discussed the design of Cassandra and different consistency solutions.

So, in conclusion, we have to see that the traditional, RDBMS work with a strong consistency model and offer the acid properties and whereas the modern workloads do not need, such a strong guarantees, but do need fast response that is availability, unfortunately the cap theorem comes in to an effect, which says that out of consistency availability and partition, you have to choose two of them. That we have seen, in this part of the discussion. So, NoSQL system which are also called, 'Key-Value Store', offers the base property that is called basically available, soft state eventual consistency. So, eventual consistency and a variety of other consistency models are striving, towards a strong consistency, we have also discussed the design, details of Cassandra system and also covered different consistency models, which are the newer consistency models, which are available, for the newer kind of systems. Thank you.

Lecture 16
Design of Zookeeper

Design of Zookeeper.

Refer slide time: (0:18)

Preface

Content of this Lecture:

- In this lecture, we will discuss the '**design of ZooKeeper**', which is a service for coordinating processes of distributed applications.
- We will discuss its basic fundamentals, design goals, architecture and applications.



Preface; Content of this lecture; in this lecture we will discuss the 'Design of Zookeeper', which is a service for coordinating processes in a distributed application. We will discuss; the fundamentals, the design goals, architecture, and the applications of Zookeeper. Zookeeper is also known as, 'Apache Zookeeper', which is an open-source foundation. So it is available, a free to be used, for the open communities.

Refer slide time: (1:00)

ZooKeeper, why do we need it?

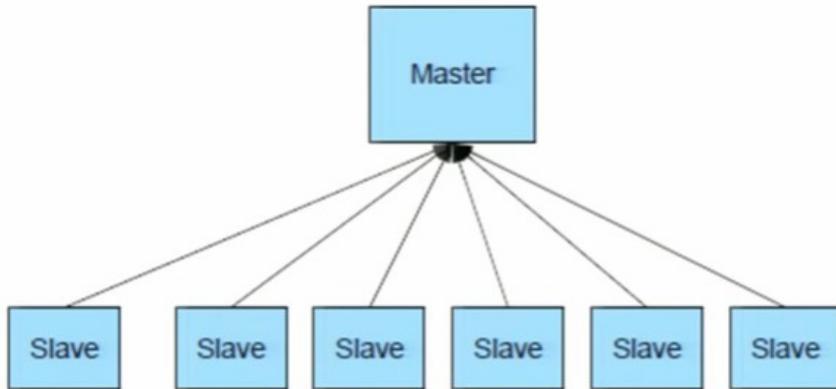
- **Coordination is important**



Zookeeper, why do we need it? So Zookeeper provides the coordination service. What is the coordination? We can understand by looking up, this particular picture, there will be traffic condition at the intersection, so if there is a coordination then, there will not be any such congestion and there will be a smooth flow of the traffic, similarly here, in our scenario of the big data computing, we have the hundreds and thousands of cluster nodes, wherein the processes are executing, these cluster nodes are coming up and going down, that means, they are prone to the failure and different applications require to be run-on, these nodes. Therefore, for the smooth way of resource allocation, in a fault alternate or a reliable manner, there is a requirement of coordination service and Zookeeper is, used for that purpose or designing of different, big data applications. And the libraries, we will see in this part of the discussion, having understood what coordination is? And how important the coordination is?

Refer slide time: (2:24)

Classic Distributed System

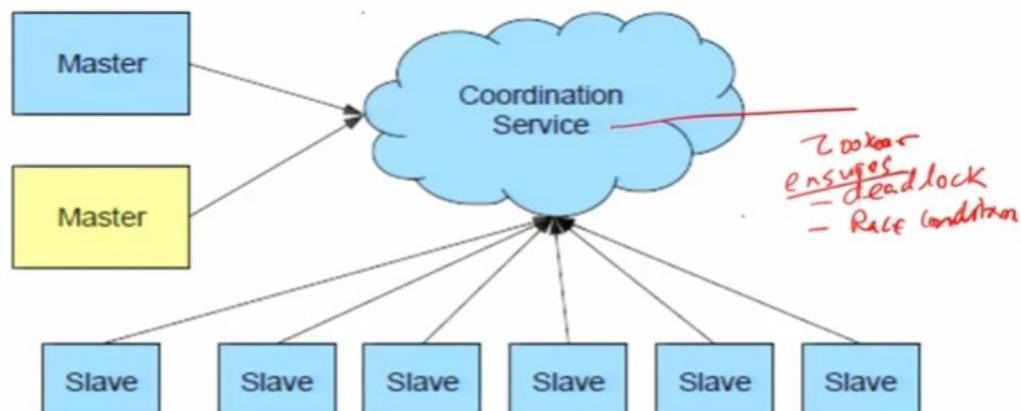


- Most of the system like HDFS have one Master and couple of slave nodes and these slave nodes report to the master.

Now we have to understand, in another scenario, which is of the master and slave relation of a distributed system, here most of the systems, like HDFS has one master and a couple of slave nodes and these slave nodes will report to them, to the paid, to the master. What happens if the master fails? The entire system will collapse to work.

Refer slide time: (2:51)

Fault Tolerant Distributed System



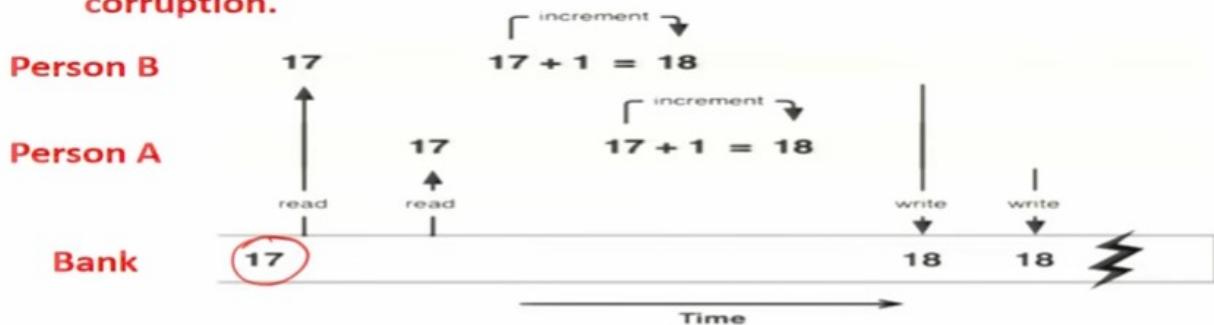
- Real distributed fault tolerant system have Coordination service, Master and backup master.
- If primary failed then backup works for it.

The fault tolerant distributed systems ,where in, between master and slave, there is a coordination service, which coordinates between the master and slave processes, running on different nodes so the irrespective of the node failures, the coordination service will, assign the nodes to these master and therefore, a fault tolerant distributed system can be envisage using such a coordination service. So, this coordination service is very much required, to achieve this far tolerant, coordinated view of any application in a distributed environment, so real distributed fault-tolerant system how a coordination service that is the master and a backup master, so if a master fails then the backup master takes over. and this particular taking over of backup master, as the master is to be monitored and decided by the coordination service, so therefore, the master and the slave, can be used in an application, irrespective of the false. So, if the primary, that is the master primary fails, then the backup can be taken up, to ensure the fault tolerant distributed system. Here in this example, the coordination service is the Zookeeper, which provides the coordination services, to achieve the fault tolerant distributed system. This particular coordination service, has we ensure that, it should not lead to a deadlock and also not suffer from, the race conditions. so and, it ensures that deadlock, can be prevented deadlock should not happen, in any of the case and also, it should not face the risk conditions.

Refer slide time: (2:51)

What is a Race Condition?

- When two processes are competing with each other causing **data corruption**.



As shown in the diagram, two persons are trying to deposit 1 rs. online into the same bank account. The initial amount is 17 rs. Due to race conditions, the final amount in the bank is 18 rs. instead of 19.

To understand these two terms, what do you mean by the race condition? we can see through this particular example, so when two processes are competing with each other, causing data corruption. for example, person A and person B together, they want to update, the same variable that is the bank account, which is having the current value 17, at the same point of time, with the value 1, then what will happen is, they may read the variable 17 and update at their end and finally write back, the values. So instead of, two increments, only one increment is now, taken into the effect. Therefore, together the two processes which are competing may cause, the data corruption here in this example, which is shown, this condition is called a, 'Race Condition'. so as shown in the diagram, two persons are trying to deposit

1 rupees online, on to the same bank account, the initial amount is 17 rupees ,due to the race condition, the final amount is written as 18 rupees, instead of 19. so therefore, this condition is called a, ‘Race Condition’, where competing processes, which are writing or which are accessing the variable may cause, the race conditions and therefore, may corrupt the data. This has to be avoided in the coordination service, why because multiple, multiple servers, multiple systems, they are trying to access or trying to run the same service. Therefore, this race condition may be a possibility and the coordination service has to, be such that this condition should not occur.

Refer slide time: (7:03)

What is a Deadlock?

- When two processes are waiting for each other directly or indirectly, it is called **deadlock**.



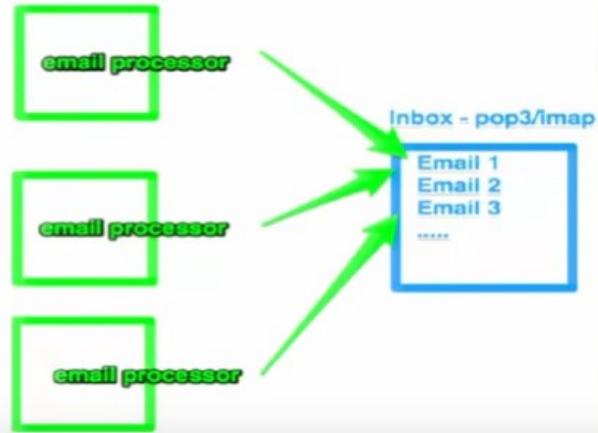
- Here, Process 1 is waiting for process 2 and process 2 is waiting for process 3 to finish and process 3 is waiting for process 1 to finish. All these three processes would keep waiting and will never end. This is called dead lock.

Another problem is about, deadlock. So when two processes are, waiting for each other directly or indirectly. And none of is basically able to progress, then it is called a, ‘Deadlock’. so here, in this example, set up waiting processes 1, 2, 3, they are said to be deadlock, if they are waiting for each other, for the resource, allocation and none of them are, finishing for the others to complete. Hence, this circularly waiting process are said to be a deadlock. So in our scenario, in a distributed, computing where different, servers are running different processes and they may be communicating with each other, they may also suffer to this particular problem, which is called a, ‘Distributed Deadlock Problem’. So hence, the situation of coordination service, should be such that, it should not lead to a deadlock.

Refer slide time: (8:03)

What is Coordination ?

- How would Email Processors avoid reading same emails?
- Suppose, there is an inbox from which we need to index emails.
- Indexing is a heavy process and might take a lot of time.
- Here, we have multiple machines which are indexing the emails. Every email has an id. You can not delete any email. You can only read an email and mark it read or unread.
- Now how would you handle the coordination between multiple indexer processes so that every email is indexed?

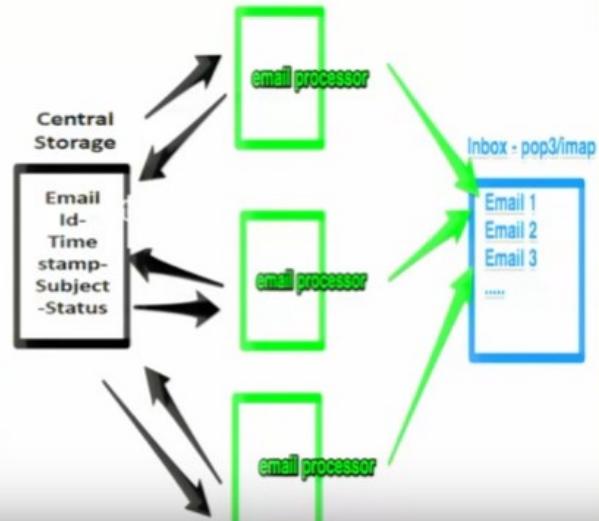


So let us see, some of the example of the coordination service in a distributed system for big data computations. Let us understand the email processors. And how these email process are avoid reading the same email? Using the coordination service, suppose there is an inbox, from which we need to index emails, indexing is a heavy process and might take, a lot of time. Hence, we have multiple machines, which are indexing the emails, every email has an ID and you cannot delete any email, you can only read an email and mark it as, read or unread. Now, how would you handle the coordination, between multiple indexer processes, so that every mail is indexed, that is shown here, in this particular picture.

Refer slide time: (9:10)

What is Coordination ?

- If indexers were running as multiple threads of a single process, it was easier by the way of using synchronization constructs of programming language.
- But since there are multiple processes running on multiple machines which need to coordinate, we need a central storage.
- This central storage should be safe from all concurrency related problems.
- **This central storage is exactly the role of Zookeeper.**



So if the indexers were running, as multiple threads on a single machine or a single processor, then it would be, easily handled with the, help of synchronization constructs of a programming language. But, since there area multiple processes running on multiple machines, we need to coordinate; that is we require, central storage kind of scenario, implemented on a distributed system, which is nothing but a coordination service. So this central storage would be safe from all concurrency related problems ,this central storage is exactly the role of the coordination, in the Zookeeper.

Refer slide time: (9:49)

What is Coordination ?

- **Group membership:** Set of datanodes (tasks) belong to same group
- **Leader election:** Electing a leader between primary and backup
- **Dynamic Configuration:** Multiple services are joining, communicating and leaving (Service lookup registry)
- **Status monitoring:** Monitoring various processes and services in a cluster
- **Queuing:** One process is embedding and other is using
- **Barriers:** All the processes showing the barrier and leaving the barrier.
- **Critical sections:** Which process will go to the critical section and when?

So again, what is the coordination? That is about, group membership that is the setoff tasks, under the data node; belong to the same group: that is also a coordination service. To elect a leader between the primary and the replicas, that also is a coordination service. than dynamic configuration that is multiple services are joining and communicating and there, that is why common, configuration is required, to be maintained, where in the active servers are to be registered is, also called, 'Dynamic Configuration'. And is a part of coordination service. Status monitoring where monitoring, we are various processes and services in a cluster is maintained, dynamically is also a coordination service, queuing that is one process is, embedding and the other is using that is called, 'Coordination Service'. Barriers all the processes showing the barriers and leaving the barrier is, also a coordination service. Critical sections where processes go into the critical section and then, they may, leave the critical section, so that others can enter into the critical section, so this kind of critical section is also kind of coordination service.

Refer slide time: (11:22)

What is ZooKeeper ?

- **ZooKeeper is a highly reliable distributed coordination kernel,** which can be used for distributed locking, configuration management, leadership election, work queues,....
- Zookeeper is a replicated service that holds the metadata of distributed applications.
- **Key attributed of such data**
 - Small size
 - Performance sensitive
 - Dynamic
 - Critical
- **In very simple words,** it is a central store of key-value using which distributed systems can coordinate. Since it needs to be able to handle the load, Zookeeper itself runs on many machines.

So what is the Zookeeper? So having understood the coordination service, Zookeeper is a highly reliable distributed coordination kernel, which can be used for distributed locking, configuration, management, leader election, work queues and so on. So, a Zookeeper is a replicated service that holds the metadata of distributed application, by meaning, to say they are a placated service means, that particular service, runs on the clusters, which is consist of multiple nodes or multiple server machines, that is why it is called the, 'Replicated Service'. So, how that is all done, in the Zookeeper, which is the replicated service, on one hand, while providing the centralized configuration service, that is the coordination service, that is the design goal of the Zookeeper, which we are going to cover. So the key attributes of such a data which the Zookeeper requires to maintain in the replicated services are the size of the data is small and the performance is very sensitive and it is dynamically changing and it is also very critical to manage, in the coordination replicated service by the Zookeeper. So in a very simple word, the Zookeeper is a central store of key-value pairs, using which, the distributed systems can coordinate, since it needs to be able to handle the load, the Zookeeper itself runs on many machines.

Refer slide time: (13:05)

What is ZooKeeper ?

- Exposes a simple set of primitives
- Very easy to program
- Uses a data model like directory tree
- Used for
 - Synchronisation
 - Locking
 - Maintaining Configuration
- Coordination service that does not suffer from
 - Race Conditions
 - Dead Locks

So, the Zookeeper exposes a simple set of primitives, it is very easy to program, uses the data model like, the directory tree and it is used for synchronization, locking, maintaining configuration. So, the coordination service, should also not suffer from, race condition and deadlock, though it has to be providing the synchronization locking and maintaining the configuration, as a service, that is the Zookeeper, does this and it uses, to do this, it uses a data model, which is like a tree, directory tree and it also has a very easy program interface, for so that the programmers can use it, for designing their own distributed application, for big data computation .and it also exposes a simple set of primitives that, we will see.

Refer slide time: (14:01)

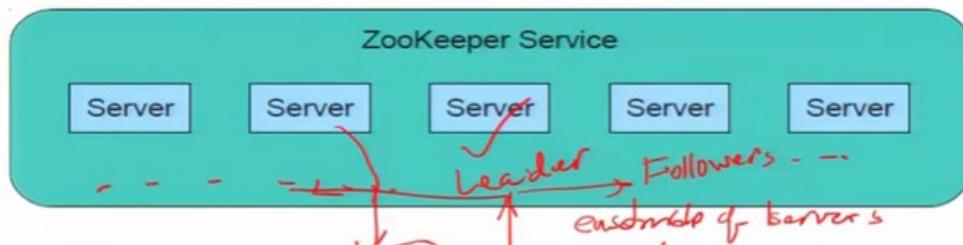
Design Goals: 1. Simple

- A shared hierachal namespace looks like standard file system
- The namespace has data nodes - znodes (similar to files/dirs)
- Data is kept in-memory
- Achieve high throughput and low latency numbers.
- **High performance**
 - Used in large, distributed systems
- **Highly available**
 - No single point of failure
- **Strictly ordered access**
 - Synchronisation

So, the first design goal of Zookeeper is the simple, simplicity. So, it has the shared, hierarchical namespace which looks like, a standard file system of a UNIX. So, the namespace highest the data nodes and these data nodes are, known as znodes, which disturb this, which differentiates from the other type of the data node usage. So, the data is kept in memory and will achieve high throughput and low latency numbers. So high performance is, one of the important features of providing, the simple service that is used in the large and distributed system and it is also highly available, that is not a single point of failure, it suffers and it also strictly ordered the axis that is by providing the synchronization.

Refer slide time: (14:59)

Design Goals: 2. Replicated

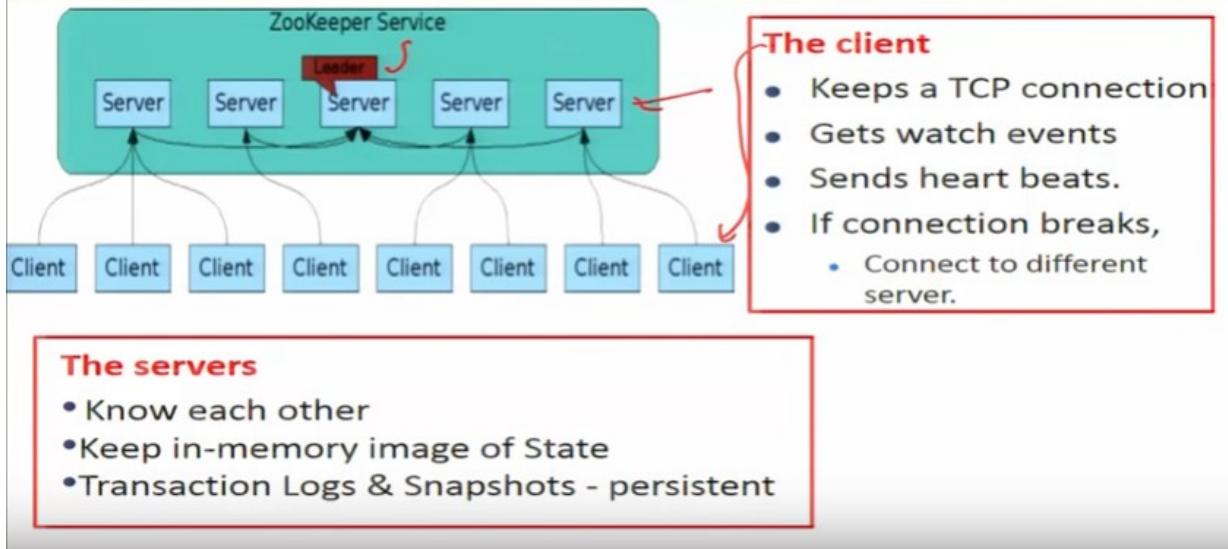


- All servers have a copy of the state in memory
 - A leader is elected at startup
 - Followers service clients, all updates go through leader (Write)
 - Update responses are sent when a majority of servers have persisted the change
- We need $2f+1$ machines to tolerate f failures

So, the second goal of the Zookeeper is the replication or a replicated service. So here, the Zookeeper runs on, the example of servers. So here we have seen, there are five different servers, it can be many more also, which runs the Zookeeper service, so it is an ensemble of servers. so it is not one server, on which the Zookeeper will run, but it is an ensemble of servers on the Zookeeper will run, so all the servers have the copy of the state, in the memory and now, since all servers have the same state. Therefore, leader will be elected, at the startup and at all the time let us say, this is the real leader and all others are the followers. So, the leader is elected as I start up and at all point of time, a leader is always elected, to run this Zookeeper service. Now, the followers service the clients, all the updates go through the leader, meaning to say that, if there is a write operation by the client, it has to go through the leader, whereas the read operation by the client, can be done through any of these available servers, in the end sample. so therefore, it is written that, the followers service the clients, but all the updates has to go through the leader. Updates in the sense, the write operations has to be channeled through, the leader whereas all the read operations, can be performed or can be serviced, to the client from any of the followers therefore. Update responses are sent, when a majority of the servers have persisted the change. So here we need, the majority we need $2f+1$ machines, to tolerate f failures.

Refer slide time: (17:20)

Design Goals: 2. Replicated



So this, concept can be understood here, in this diagram that, you can see that the client can be connected to any of the servers, whether it is follower or it is the leader. But, as far as read, read operations are concerned, the client can be serviced, by any of the follower servers as far as, the write or update operation is concerned, it has to go through the leader elected and out of these servers, at this point of time, this particular server is denoted as, the leader. So here, we have the set of servers, they will know each other and they will keep in memory image of the state and also maintains the transaction log and snapshots, persistently. Whereas the other, this is one such entity, in the Zookeeper, the other entity is called a, 'Client'. So, the client keeps a TCP connection ,gets a watch events and sends the heartbeats, ID of the connection breaks, they will connect to a different servers. So these are, the client's activity in this replicated service, of Zookeeper.

Refer slide time: (18:36)

Design Goals: 3. Ordered

- ZooKeeper stamps each update with a number
- **The number:**
 - Reflects the order of transactions.
 - used implement higher-level abstractions, such as synchronization primitives.

Third important goal of Zookeeper is, about the ordering. So, so the Zookeeper will timestamp each updates, with a number, so the number is nothing but, it reflects the order of the transaction and it is used, to implement the high level abstraction, such as synchronization primitives. We will see this aspect in more detail in the further slides.

Refer slide time: (19:04)

Design Goals: 4. Fast

Performs best where reads are more common than writes, at ratios of around 10:1.

At Yahoo!, where it was created, the throughput for a ZooKeeper cluster has been benchmarked at over 10,000 operations per second for write-dominant workloads generated by hundreds of clients

Now fourth design goal is, about the speed, which is fast. So, it performs best when the reads are more common, than writes, at the ratio of around 10 is to 1. So, at Yahoo! where it was created the Zookeeper, the throughput of the Zookeeper cluster, has been benchmarked at over 10,000 operations per second for the write, dominant workloads generated by hundreds and thousands of clients.

Refer slide time: (19:33)

Data Model

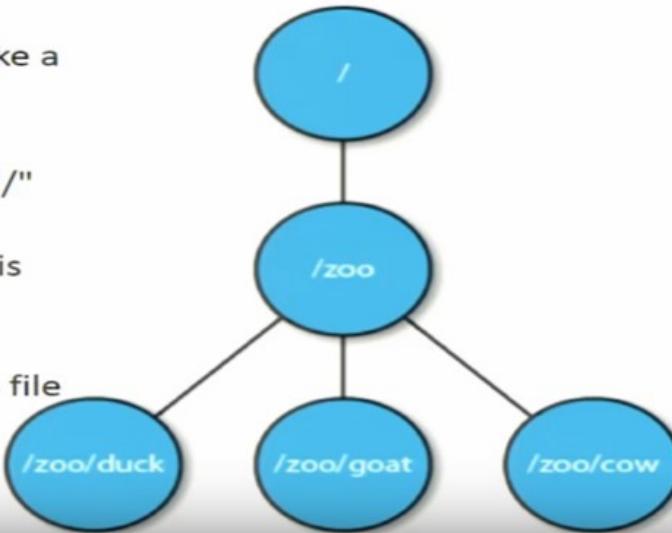
- The way you store data in any store is called **data model**.
- **Think of it as highly available fileSystem:** In case of zookeeper, think of data model as if it is a highly available file system with little differences.
- **Znode:** We store data in an entity called znode.
- **JSON data:** The data that we store should be in JSON format which Java script object notation.
- **No Append Operation:** The znode can only be updated. It does not support append operations.
- **Data access (read/write) is atomic:** The read or write is atomic operation meaning either it will be full or would throw an error if failed. There is no intermediate state like half written.
- **Znode:** Can have children

Let us see the data model. The way you store data in any store is called, ‘Data Model’. Think of it as highly available file system: in case of Zookeeper, think of a data model, as if it is highly available file system, with the little differences. Here, in this file system of Zookeeper, the nodes are called , ‘znodes’, where we store, the data in an entity, which is also called as a,’znode’. The JSON data: the data that means store, should be in the JSON format, which Java Script objection, object notation follows. And it has the append operation, so znode can only be updated, it does not support, the append operation. So, so the data axis here is, that is the read and write operations, they are atomic, so read and write operation is atomic operations, meaning either it will be full or would throw an error a field. Therefore, in no intermediate state, like half written is possible in the atomic node. and znode can have the children’s, so again we are repeating that no append operation, so znode can only be updated, it does not support the append operations, in this particular scenario.

Refer slide time: (21:05)

Data Model Contd...

- So, znodes inside znodes make a tree like hierarchy.
- The top level znode is "/".
- The znode "/zoo" is child of "/" which is the top level znode.
- "duck" is a child znode of "zoo". It is denoted as "/zoo/duck".
- Though ".." or "." are invalid characters as opposed to the file system.



So, here we can see that the znodes, inside the znodes will make like a tree hierarchy. where the top-level znode is the root node, which is the top level znode and let us say that, then child is let us say, a znode of this top node, "zoo" and this is called now the, 'Duck' and so on. So there will be a hierarchical way, these znodes will be organized in the, the namespace. So it provides the data model, of the hierarchical tree structure, to support the namespace.

Refer slide time: (21:52)

Data Model – Znode - Types

- **Persistent**
 - Such kind of znodes remain in zookeeper until deleted. This is the default type of znode. To create such node you can use the command: `create /name_of_my_znode "mydata"`
- **Ephemeral**
 - Ephemeral node gets deleted if the session in which the node was created has disconnected. Though it is tied to client's session but it is visible to the other users.
 - An ephemeral node can not have children nor even ephemeral children.

So, znode has various types, there are two different type of znodes, one is called, ‘Persistent’, such kind of znodes, will remain in Zookeeper until it is deleted. This is the default type of znode. To create such a node you can use the command: create slash name of my znode, my data. Ephemeral type of znode. Ephemeral nodes gets deleted, if the session in which, the node was created has disconnected. Though it is tied to the client session, but it is visible to the other users. So, an ephemeral node, cannot have the children not even the ephemeral children’s. So there are two different type of znode, persistent and ephemeral, we have just discussed. So, ephemeral is related to the session, whereas persistent it will remain in the Zookeeper, until it is deleted.

Refer slide time: (22:50)

Data Model – Znode - Types

- **Sequential**

- Creates a node with a sequence number in the name
- The number is automatically appended.

```
create -s /zoo v  
Created /zoo0000000008
```

```
create -s /zoo/ v  
Created /zoo/0000000003
```

```
create -s /xyz v  
Created /xyz0000000009
```

```
create -s /zoo/ v  
Created /zoo/0000000004
```

- The counter keeps increasing monotonically
- Each node keeps a counter

Now, another type is called, ‘Sequential’, will create a nodes, with the sequence numbers, in the namespace, the number is automatically appended. And the counters will keep on, monotonically increasing. So each node has a counter. so here we can, see that the, the znode which is created has, this kind of sequence numbers and whenever the new nodes are increasing, added, so these numbers are keep on, now increasing, whenever the nodes are, whenever a new node is created with a sequence number.

Refer slide time: (23:33)

Architecture

- Zookeeper can run in two modes: (i) Standalone and (ii) Replicated.

(i) Standalone:

- In standalone mode, it is just running on one machine and for practical purposes we do not use standalone mode.
- This is only for testing purposes.
- It doesn't have high availability.

(ii) Replicated:

- Run on a cluster of machines called an ensemble.
- High availability
- ~~• Tolerates as long as majority.~~

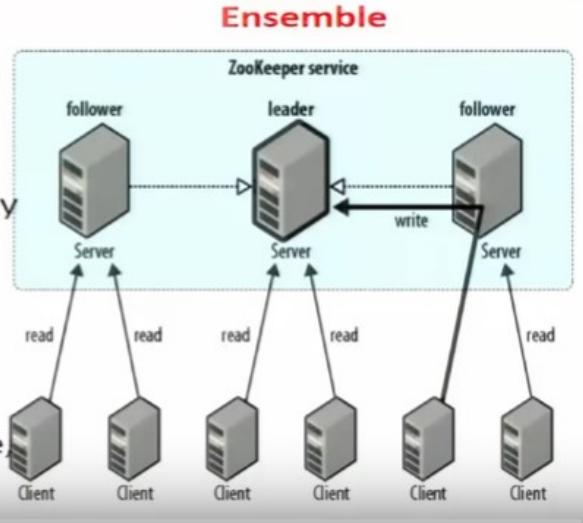
So, the architecture of Zookeeper, we can see here that Zookeeper can run in two modes, one is standalone, the other is called replicated mode. In a standalone mode, it is just running one, one machine and for practical purpose, we do not use this standalone mode. This is only for testing purposes and does not have, the high availability because, it is running on only one machine and if it fails, the entire operation will stop functioning. Therefore, it is only meant for testing purposes. The other mode of operation in the Zookeeper is called, 'Replicated Mode', it is used for the production clusters in, in this mode. So, it runs on the cluster of machines called, 'Ensemble'. And it ensures high availability and also tolerates, as long as, it ensures the majority of the machines are in active service.

Refer slide time: (24:28)

Architecture: Phase 1

Phase 1: Leader election (Paxos Algorithm)

- The machines elect a distinguished member - leader.
- The others are termed followers.
- This phase is finished when majority sync their state with leader.
- If leader fails, the remaining machines hold election. takes 200ms.
- If the majority of the machines aren't available at any point of time, the leader automatically steps down.



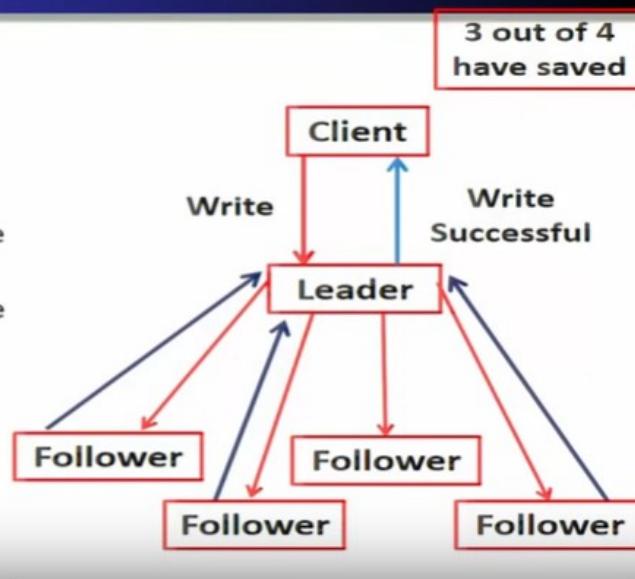
So, let us see the architecture: and in this architecture, this Zookeeper runs in two phases. so the phase one is about the leader election, which uses the Paxos like algorithm, which is called, 'zservice'. So, which is called, 'Job, zoperations'? So the machines will elect the distinguished members, which is called the, 'Leader over here'. Which is shown here, as the Zookeeper service in the phase 1 and the other, servers are termed as the followers, in the ensemble, servers for running the Zookeeper service. Now, this phase is finished when the majority, will sync their state with the leaders and if the leader fails, the remaining machines will hold the election and elect another, follower as the leader and this process of leader election takes only 200. Now, if the majority of the machines are not available at any point of time, the leader election has to be ensured if the majority rule follows and therefore, the majority rule is important, to functioning of the Zookeeper.

Refer slide time: (25:56)

Architecture: Phase 2

Phase 2: Atomic broadcast

- All write requests are forwarded to the leader,
- Leader broadcasts the update to the followers
- When a majority have persisted the change:
 - The leader commits the up-date
 - The client gets success response.
- The protocol for achieving consensus is atomic like two-phase commit.
- Machines write to disk before in-memory



Now in phase 2, there will be an atomic broadcast, that is all right requests are forwarded to the, to the leader. And so, the client you see here is forwarding the right request to the leader. and the leader broadcast the updates to all the follower, when a majority have persisted the change, that is the leader commits, the update and the client gets the success response, then the protocol for achieving the consensus is atomic and this is like two-phase commit protocol. So the machines will write to the disk before in memory.

Refer slide time: (26:48)

Election in Zookeeper

- Centralized service for maintaining configuration information
- **Uses a variant of Paxos called Zab (Zookeeper Atomic Broadcast)**
- Needs to keep a leader elected at all times

Reference: <http://zookeeper.apache.org/>

So, now let us see in more detail about, the phase one, which is the election process in Zookeeper. So, Zookeeper is the centralized service, for maintaining the configuration information. Now, Zookeeper uses, a variant of Paxos which is called, 'Zab', that is Zookeeper atomic broadcast. Now, Zookeeper needs to keep, the lead leader elected at all point of time, how that is leader is elected, we will understand the complete process in the Zookeeper.

Refer slide time: (27:20)

Election in Zookeeper (2)

- Each server creates a new **sequence number** for itself
 - Let's say the sequence numbers are **ids**
 - Gets highest id so far (from ZK(zookeeper) file system), creates next-higher id, writes it into ZK file system
- Elect the highest-id server as leader.

The diagram illustrates the election process for Zookeeper servers. Six servers are shown in boxes: N12, N3, N6, N32, N80, and N5. N80 is labeled 'Master'. The sequence numbers are: N12, N3, N6, N32, N80, and N5. The highest sequence number is 80, which is assigned to the Master server.

So, each server will create at the new sequence number for itself. and this particular sequence number is stored in a file, so let us say, that the sequence numbers are the node IDs and the highest node ID seen so far, from the Zookeeper file system, will create the next higher ID, I had write sit into the Zookeeper file. Now, elect the highest ID server, as the leader. So, here in this example, the different servers will have their, IDs and a , N6 ,N12, N3, N32, N5, among these six different servers, the one with the highest ID is an 80 and it is now, after this election it will be elected as, the leader or the master.

Refer slide time: (28:20)

Election in Zookeeper (3)

- **Failures:**

- One option: everyone monitors current master (directly or via a failure detector)
 - On failure, initiate election
 - Leads to a flood of elections
 - Too many messages

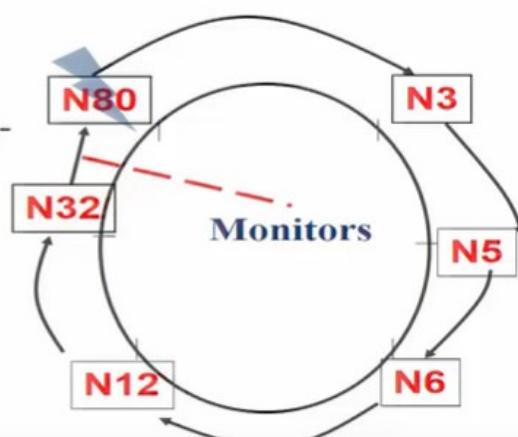


Now, what about the failures? Now, during the election or after the election that the master may crash. So one option is that, everyone monitors, the current master, directly why are the failure detectors. So that means, in normal cases, leader election is not happening, again and again even if the master fails, so every other node monitors the current master. and whenever it detects, that the master is filled, it will initiate the election, that means, if they initiate the election it will lead to a flood of lot of messages and too many messages, therefore this option of whenever there is a failure, everyone start electing the leader, this will require a flood of messages.

Refer slide time: (29:15)

Election in Zookeeper (4)

- **Second option:** (implemented in Zookeeper)
 - Each process monitors its next-higher id process
 - **if** that successor was the leader and it has failed
 - Become the new leader
 - **else**
 - wait for a timeout, and check your successor again.



Therefore, we will see a second option, which is implemented in Zookeeper. Where in the each process monitors its next higher ID process. For example, process number 32 will monitor, process number 80 and 80 will monitor 3 and 3 will monitor 5 and 5 will monitor 6 and 6 will monitor 12 and 12 will monitor 32. So, each process will monitor its next higher ID process. Now, if that successor, to which a process is monitoring, it detected as the field and if that, field process, happens to be the leader, then the, that particular process, who has directed, this leader to be failed becomes the new leader. So, there is no leader election required, here in this case of Zookeeper, otherwise then it will wait for a timeout. And check the, the successor again.

Refer slide time :(30:11)

Election in Zookeeper (5)

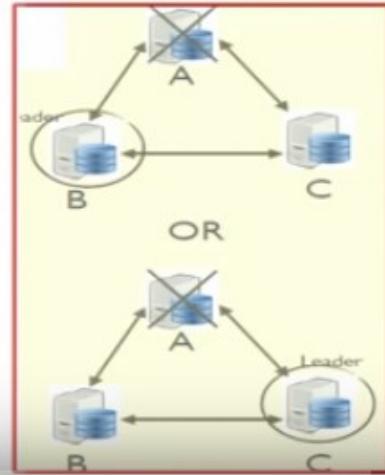
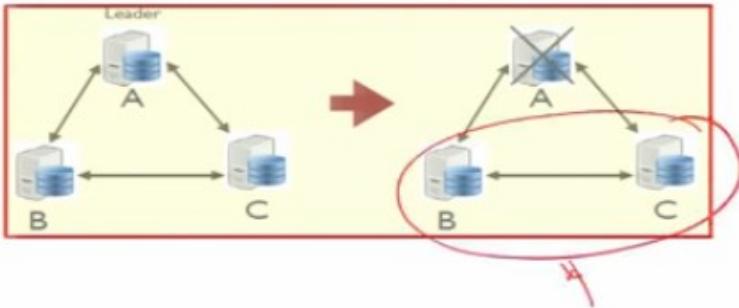
- What about id conflicts? What if leader fails during election ?
- To address this, Zookeeper uses a ***two-phase commit*** (run after the sequence/id) protocol to commit the leader
 - Leader sends NEW_LEADER message to all
 - Each process responds with ACK to at most one leader, i.e., one with highest process id
 - Leader waits for a majority of ACKs, and then sends COMMIT to all
 - On receiving COMMIT, process updates its leader variable
- Ensures that safety is still maintained

So, therefore, what about the IDs conflict, what about if the leader fails during the election in both these cases, to handle these cases the Zookeeper, does this election, using two-phase commit protocol. So, Zookeeper runs after the sequence ID and this particular protocol, used to commit the leader. So, leader will send the new leader message, to all of the other processes and each process, will respond with the acknowledgement, to at most one leader, that is one with the highest process ID seen. So, far the other processes will respond, as an acknowledgement, allowing that particular process, to become the leader. Now leader a waits for the majority of acknowledgments and then sends the commit, to our on receiving the commit protocol, on receiving the commit message, the process updates is leader variable and it ensures that the safety is always maintained in this two-phase leader election protocol.

Refer slide time :(31:20)

Election Demo

- If you have three nodes A, B, C with A as Leader. And A dies. Will someone become leader?

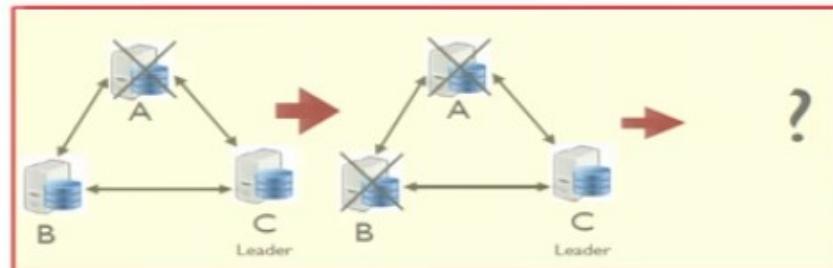


Let us see the election process, demonstration in Zookeeper, now if you have three nodes ABC with leader elected as a node shown on the left side of the diagram. And if a dies will someone become the leader, someone that is the remaining B and C will become the leader. So, if a dies then B and C which becomes, the majority, of three that is two out of three is the majority. So, majority survives hence, out of B and C one will be elected as the leader. So, the answer is yes that means either B or C can become, the elect can become the leader why because, even after the failure of a the remaining node set, remains in the majority.

Refer slide time :(32:16)

Election Demo

- If you have three nodes A, B, C And A and B die. Will C become Leader?



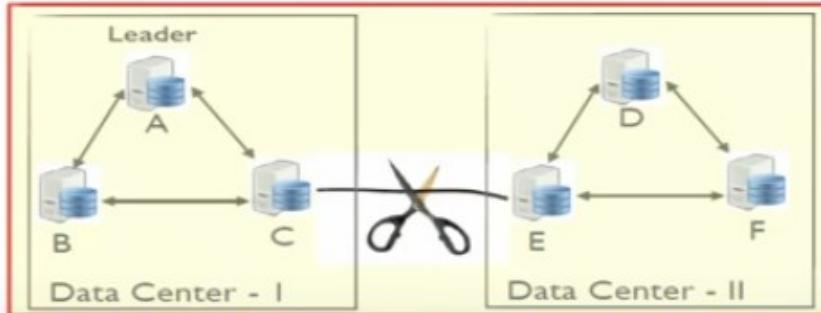
No one will become Leader.
C will become Follower.
Reason: Majority is not available.

Now what about if two nodes are filled, it is not only one but two nodes are filled. So, the remaining one is node number, one out of three, one out of three is not in a majority, therefore if you have three nodes. And two of them dies, then can the C will the C becomes the leader, the answer is no, no one will become the leader, why because C become the follower and the reason is that the majority, is not available if two nodes A dies, therefore the leader election, will happen as long as, the after the failure the majority, serve is majority of the server it is surviving.

Refer slide time :(33:06)

Why do we need majority?

- **Imagine:** The network between data centres got disconnected.
If we did not need majority for electing Leader,
- **What will happen?**



Now let us see why the majority, is important feature or important parameter in the leader election. So, imagine that you have an ensemble, of servers which are spreaded across, two different data centers. So, this example shows that, you have six servers which are spreaded across two data centers that is three on one data center and remaining three out of six, will be on the other data center. Now if let us say there will be a disconnection, which is the network disconnection, which connects these two data center. So, imagine the network between the data center got disconnected, now if we did not need the majority, of electing the leader then what will happen that every data center will be electing one leader.

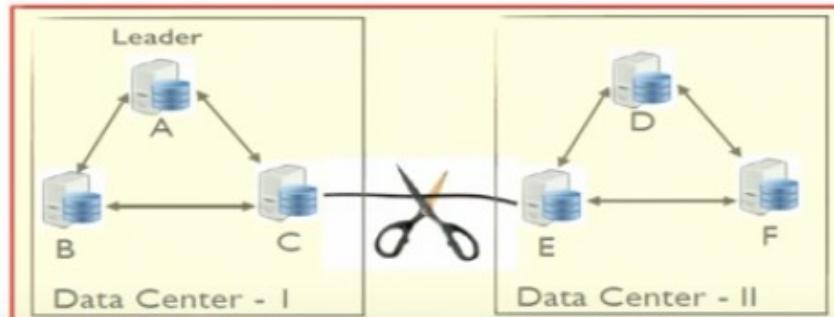
Refer slide time :(33:55)

Why do we need majority?

Each data centre will have their own Leader.

No Consistency and utter Chaos.

That is why it requires majority.



And therefore in this particular scenario, there will be two leaders, appointed and in this case there will be violation of leader election, there cannot be two leaders at the same point of time, hence therefore the majority is required,

Refer slide time :(34:17)

Sessions

- Lets try to understand **how do the zookeeper decides to delete ephemeral nodes** and takes care of session management.
- A client has list of servers in the ensemble
- It tries each until successful.
- Server creates a new session for the client.
- A session has a timeout period - decided by caller

To avoid this kind of scenarios, now we will see about the sessions which are maintained in the Zookeeper. Now let us try to understand, how the Zookeeper decides to delete the ephemeral nodes and take care of session management. So, a client has the list of servers, in the example and it tries each until it becomes successful. And the server creates a new session for the client; a session has the timeout period, which is decided by the caller.

Refer slide time :(34:51)

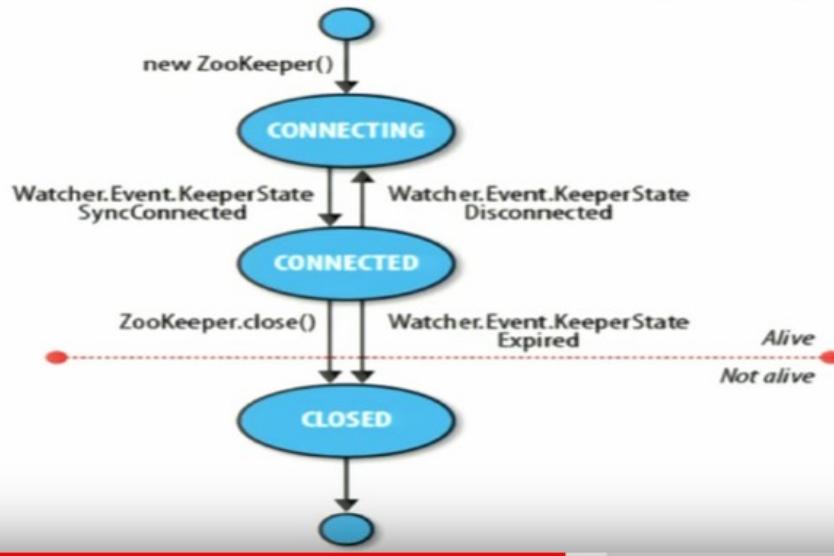
Contd...

- If the server hasn't received a request within the timeout period, it may expire the session.
- On session expire, ephemeral nodes are lost
- To keep sessions alive client sends pings (heartbeats)
- Client library takes care of heartbeats
- Sessions are still valid on switching to another server
- Failover is handled automatically by the client
- Application can't remain agnostic of server reconnections
 - because the operations will fail during disconnection.

So, if the server has not received the request, within a particular timeout period, it may expire the session, on session expiry ephemeral nodes are lost, to keep the session alive, the client sends the ping that is the heartbeats and the client library takes care of heartbeats, sessions are still valid on switching to another server. So, failover is handled automatically, by the client and applications cannot remain agnostic of the server reconnections, because the operations will fail during the disconnection.

Refer slide time :(35:23)

States

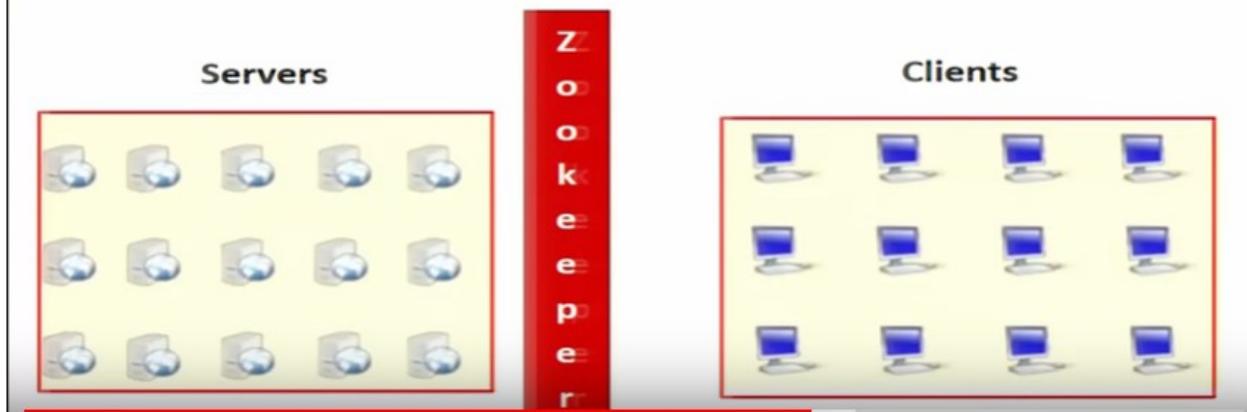


So, this is automatically handled by, the Zookeeper let us understand that when a new state zoo Zookeeper process is starting, about connecting to it then it will watch, the events and keep the state synchronized, with the connected, with the clients that is called, 'Connected'. So, the Zookeeper, when it is closing the connection, then it will be closed. And in between it will watch the events, Keeper State and this particular, when it is expired, then it will send that the watch event state is disconnected.

Refer slide time :(36:10)

Use Case: Many Servers How do they Coordinate?

- Let us say there are many servers which can respond to your request and there are many clients which might want the service.

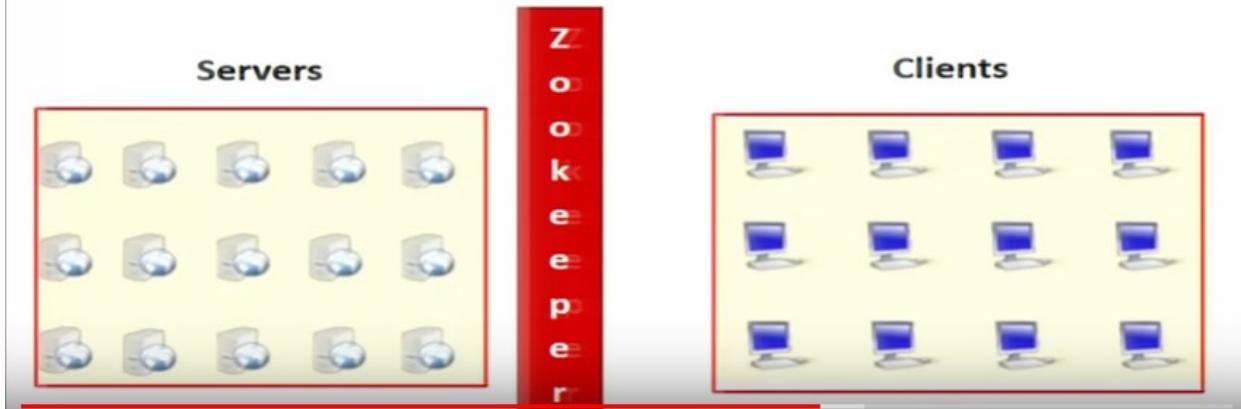


So, these state transitions are handled, now let us see the use case where many servers are available and how the coordination service, will ensure the availability of server to any applications, which is running on the server. So, now let us see that there are many servers which can respond, to your request and there may be a single line, which one this particular service to none. So, the client wants to know which of these servers are alive at a particular point of time and directly, every client instead of maintaining the list of servers, which are active and keep on updating, at their level instead of that let us keep a coordination service, which is called a 'Zookeeper'. Now Zookeeper will coordinate between, all the set of servers and between the set of all different clients.

Refer slide time :(36:58)

Use Case: Many Servers How do they Coordinate?

- Let us say there are many servers which can respond to your request and there are many clients which might want the service.

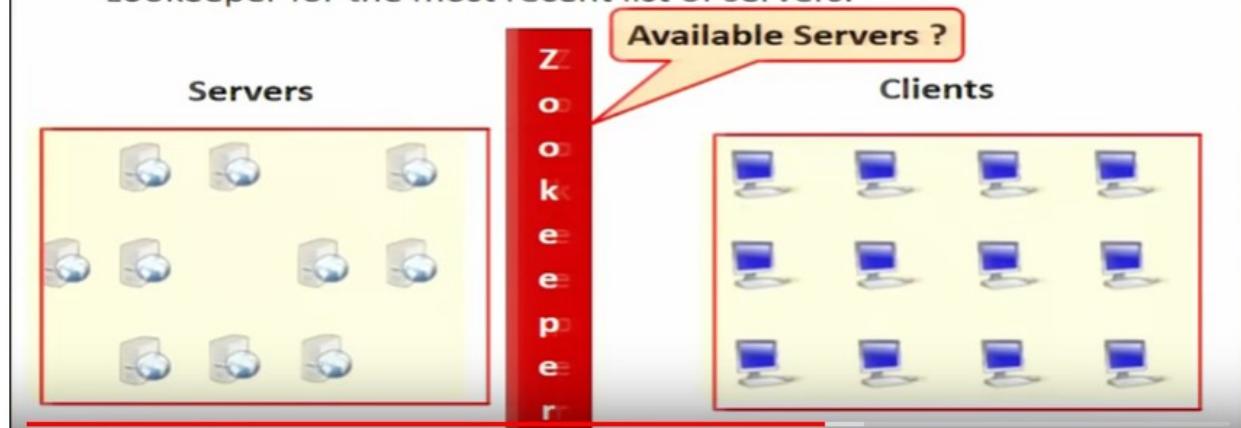


So, from time to time, some of the servers will keep going down. How can all client can keep track of these different servers?

Refer slide time :(37:11)

Use Case: Many Servers How do they Coordinate?

- It is very easy using zookeeper as a central agency. Each server will create their own ephemeral znode under a particular znode say "/servers". The clients would simply query zookeeper for the most recent list of servers.



So, the available servers will be now maintained, not by the client, but by the service which is called a, 'Zookeeper'.

Refer slide time :(37:18)

Use Case: Many Servers How do they Coordinate?

- Lets take a case of two servers and a client. The two server duck and cow created their ephemeral nodes under "/servers" znode. The client would simply discover the alive servers cow and duck using command ls /servers.
- Say, a server called "duck" is down, the ephemeral node will disappear from /servers znode and hence next time the client comes and queries it would only get "cow". So, the coordinations has been made heavily simplified and made efficient because of ZooKeeper.

The diagram shows a sequence of events:

- A client connects to a ZooKeeper server, sending `Connect(host, timeout, Watcher handle)`. The server responds with "okay. I will call you when needed."
- The client processes the event, sending `handle.process(WatchedEvent : Connected)`.
- The client creates two ephemeral nodes under the "/servers" znode: `create("/servers/duck", ephemeral node)` and `create("/servers/cow", ephemeral node)`.
- After some time, the "Duck Server Killed" (indicated by a red box and arrow) and a timer elapse.
- 5 seconds later, the client queries the "/servers" znode again, sending `ls /servers`. The server responds with "cow" (also indicated by a red box and arrow).

Annotations in red highlight the ephemeral nodes and the "Duck Server Killed" event.

Now let us see let us take the case of two servers and a client. So, the two servers duck and cow are created, by the created, they are ephemeral nodes/ server, which is shown over here, the client would simply discover, the alive servers, cow and duck using the commands / servers. Now after some time the server called, 'Duck,' is now down. So, both are now ephemeral nodes which are now created with the name duck and the cow and after, some time, this time flows down, after some time this particular server dog, got killed. So, then this particular client, when it again accesses slash servers, it will see only that cow is visible. So, earlier it was duck and cow two servers, were alive because they have created an ephemeral nodes. So, a server called, 'Duck', is down the ephemeral nodes, will disappear from the / server znode and hence, the next time the client comes up and queries, it will only get the cow. So, the core Nations has to be made heavily simplified, in this Zookeeper and made efficient also, because of the Zookeeper.

Refer slide time :(37:18)

Guarantees

- **Sequential consistency**
 - Updates from any particular client are applied in the order
- **Atomicity**
 - Updates either succeed or fail.
- **Single system image**
 - A client will see the same view of the system, The new server will not accept the connection until it has caught up.
- **Durability**
 - Once an update has succeeded, it will persist and will not be undone.
- **Timeliness**
 - Rather than allow a client to see very stale data, a server will shut down,

Now Zookeeper, guarantees the sequential consistency that is the updates from any particular clients are applied, the order in which these updates are done, that is called, ‘Sequential Consistency’. Now second thing which second property, which the Zookeeper guarantees is about atomicity, we see that the updates are either completely, successful or null that is either it is fully done or it is not done, that is called, ‘Atomicity’ property to ensure the consistency, of the dataset. So, single system image that is a client will see a single, view of the system. So, when a new server will not accept the connection, until it has been cut up. So, therefore every client so, the clients will see the same view, of the set of systems or the state variable here in case of Zookeeper. Now another thing which guaranteed, by the Zookeeper is called, ‘Durability’, that is once the update has been successful it will persist and it will not be undone, another part is called, ‘Timeliness’. So, rather than allowing, the client to see the very steel data, the server will in this case will shut down.

Refer slide time :(40:07)

Operations

OPERATION	DESCRIPTION
create	Creates a znode (parent znode must exist)
delete	Deletes a znode (mustn't have children)
exists/ls	Tests whether a znode exists & gets metadata
getACL, setACL	Gets/sets the ACL for a znode getChildren/ls Gets a list of the children of a znode
getData/get, setData	Gets/sets the data associated with a znode
sync	Synchronizes a client's view of a znode with ZooKeeper

So, these are some of the operations of the Zookeeper: create, delete, exist/ls, getACL that is access control list, setACL get data/set data, sync, will not go in more detail.

Refer slide time :(40:22)

Multi Update

- Batches together multiple operations together
- Either all fail or succeed in entirety
- Possible to implement transactions
- Others never observe any inconsistent state

And you will see we have just shows you the, the type of operations, commands which are supported by the, the Zookeeper which are very simple. Now multi updates here the batches, together supports the multiple operations, either all or either all are none is done in entirety and possible, to implement the transition in this scenario and others will never observe any inconsistent state.

Refer slide time :(40:22)

APIs

- Two core: Java & C
- contrib: perl, python, REST
- For each binding, sync and async available

Synch:

```
Public Stat exists (String path, Watcher watcher) throws KeeperException,  
InterruptedException
```

Asynch:

```
Public void exists (String path, Watcher watcher, StatCallback cb, Object ctx
```

Different APIs available and out of them two core options are Java and C and other contributions are Perl Python rest and for each building, the synchronization and asynchronous things are available, which are therein.

Refer slide time :(41:10)

Watches

- Clients to get notifications when a znode changes in some way
- Watchers are triggered only once
- For multiple notifications, re-register

Now another part which is available in the Zookeeper is called, ‘Watches’, so, clients to get notifications when a znode changes in some way, is done through the watches. So, Watchers are triggered only once for multiple notifications, they have to reread Easter. So, what triggers are available in Zookeeper and this is used, to get the notifications, notifications for the clients. So, whenever there are znode changes, in some way the clients will get the notification.

Refer slide time :(41:58)

Watch Triggers

- The read operations exists, `getChildren`, `getData` may have watches
- Watches are triggered by write ops: `create`, `delete`, `setData`
- **ACL (Access Control List)** operations do not participate in watches

WATCH OF ...ARE TRIGGERED	WHEN ZNODE IS...
<code>exists</code>	created, deleted, or its data updated.
<code>getData</code>	deleted or has its data updated.
<code>getChildren</code>	deleted, or its any of the child is created or deleted

So, what triggers so, the read operations which exists here is to get children, get data may have watches, which are triggered operations. So, watches are triggered by, the right operation, such as create delete set data, then there is an access control list, of operation ,which do not participate in the watches. So, watch of the following are triggered, that is exists create delete or its data is updated, in the znode then it gets triggered, using exists command. Now in the get data that means when in the znode the operation is deleted or it has that data updated, then using this particular gate data, it will get the trigger similarly, when the znode gets deleted or it is any of a children is created or deleted then using a get children the watch triggered is triggered, using get children.

Refer slide time :(43:22)

ACLs - Access Control Lists

ACL Determines who can perform certain operations on it.

- **ACL is the combination**
 - authentication scheme,
 - an identity for that scheme,
 - and a set of permissions
- **Authentication Scheme**
 - **digest** - The client is authenticated by a username & password.
 - **sasl** - The client is authenticated using Kerberos.
 - **ip** - The client is authenticated by its IP address.

So, in access control list, access control list determines, who can perform certain operations on it success control list is a combination, of authentication scheme and identity for that scheme and a set of permissions. So, authentication is scheme contains the digest, that is the client is authenticated by the username and the password, which is nothing but it digest, generates a choices then sasl is that the client is authenticated using curved arrows and it will give a session, ACL list that is called sasl IP is the client is authenticated, by its IP address then it is an IP.

Refer slide time :(44:15)

Use Cases

Building a reliable configuration service

- A Distributed lock service

Only single process may hold the lock

Now use cases, is that building a reliable configuration service, using curvy nodes, now another use case is the distributed lock service, that is only a single process, may hold the lock at any point of time, this distributed lock service is provided by the Zookeeper. Now let us see when not to use this

Refer slide time :(44:40)

When Not to Use?

1. To store big data because:

- The number of copies == number of nodes
- All data is loaded in RAM too
- Network load of transferring all data to all Nodes

2. Extremely strong consistency

Zookeeper is when you want to store, very big data, then the Zookeeper cannot be stored and all also when extremely strong consistency applications are required then also, it is it should not be used.

Refer slide time :(44:54)

ZooKeeper Applications: The Fetching Service

- **The Fetching Service:** Crawling is an important part of a search engine, and Yahoo! crawls billions of Web documents. The Fetching Service (FS) is part of the Yahoo! crawler and it is currently in production. Essentially, it has master processes that command page-fetching processes.
- The master provides the fetchers with configuration, and the fetchers write back informing of their status and health. The main advantages of using ZooKeeper for FS are recovering from failures of masters, guaranteeing availability despite failures, and decoupling the clients from the servers, allowing them to direct their request to healthy servers by just reading their status from ZooKeeper.
- Thus, FS uses ZooKeeper mainly to manage configuration metadata, although it also uses Zoo- Keeper to elect masters (leader election).

Now let us see some of the Zookeeper applications and here, we are going to discuss, the fetching service, as the Zookeeper application. So, the fetching service is nothing but here in this service the crawling is an important part of this fetching service, which is nothing but a search engine and in this particular service the Yahoo crawls billions, of web documents. So, this fetching service is a part, of Yahoo's crawler and it is currently in the production, coessentially it has the master processes that commands, page fetching processes. So, the master provides, the fetchers with the configuration and the features right back, informing of their status and the health, the main advantage of using Zookeeper for fetching service are recovering from the failure of the masters, guaranteeing availability despite failures and decoupling, the client from the servers allowing them to direct their request to the healthy servers by just reading their status from the Zookeeper thus fetching service, uses Zookeeper mainly to manage the configuration metadata although it uses the Zookeeper to elect the masters that is done through the leader election.

Refer slide time :(46:54)

ZooKeeper Applications: Katta

- **Katta:** It is a distributed indexer that uses Zoo- Keeper for coordination, and it is an example of a non- Yahoo! application. Katta divides the work of indexing using shards.
- A master server assigns shards to slaves and tracks progress. Slaves can fail, so the master must redistribute load as slaves come and go.
- The master can also fail, so other servers must be ready to take over in case of failure. Katta uses ZooKeeper to track the status of slave servers and the master (**group membership**), and to handle master failover (**leader election**).
- Katta also uses ZooKeeper to track and propagate the assignments of shards to slaves (**configuration management**).

Now let us see another service which is called, ‘Katta’, so, Katta is the distributed, indexer that uses Zookeeper for the coordination service and it is an example of non Yahoo application. So, Katta divides the work of indexing using shots. So, a master server assigns the shard, to the slaves and tracks the progress. So, slips can fail so, the master must redistribute the loads as the slaves come and go. So, master canal so, fail so, the other servers must be ready to take over in case of the failures. So, Katta uses Zookeeper, to keep the status of slave servers and the master, therefore it is called the group membership, which is done through the, the Zookeeper and to handle the master failure by the leader election. So, the two aspects of leader election of Zookeeper, which is used here in Katta service is called the, ‘Group Membership’, and the leader election. So, Katta also uses Zookeeper to, to crack and propagate the assignments of shots to the client, which is maintained as the configuration management. So, three different, uses of Zookeeper is used in Katta service, first is called the group membership, where the it has to keep track of slave servers and the master servers, which is called a group membership, the second one is to handle the master failover, is done by the leader election, third is that whenever, the to keep track and propagate the assignments, of charge to the to the slaves this configuration management, is used of as a service from the Zookeeper.

Refer slide time :(48:19)

ZooKeeper Applications: Yahoo! Message Broker

- **Yahoo! Message Broker: (YMB)** is a distributed publish-subscribe system. The system manages thousands of topics that clients can publish messages to and receive messages from. The topics are distributed among a set of servers to provide scalability.
- Each topic is replicated using a primary-backup scheme that ensures messages are replicated to two machines to ensure reliable message delivery. The servers that makeup YMB use a shared-nothing distributed architecture which makes coordination essential for correct operation.
- YMB uses ZooKeeper to manage the distribution of topics ~~(configuration metadata)~~, deal with ~~failures of machines in the system (failure detection and group membership)~~, and control system ~~operation~~.

Now another Zookeeper application is Yahoo message broker so, that is called, 'YMB'. So, this is an young product. So, Yahoo's message broker, YMB is a distributed publish subscribe system this particular system manages, thousands of topics that clients can publish messages to receive the messages from, the topics are distributed among the set of servers, to provide the scalability. Now each topic is replicated, using the primary backup scheme that ensures messages are replicated to two machines to ensure reliable message delivery, the servers that make up the broker, that is YMB uses, the sheer nothing distributed architecture, which makes the coordination essential for correct operations. So, YMB uses Zookeeper to manage the distribution, of topics that is done in the configuration metadata, also YMB uses Zookeeper to deal with the failures of the machine, in the system that is the failure detectors and the group membership, is managed using the Zookeeper. And then the control system operation and it also controls the system operation. So, therefore we can summarize that the publish subscribe system, which is made by the Yahoo that is called, 'Yahoo Message Broker System', uses the help or uses the Zookeeper, as the coordination service, for to manage the distribution, of the topics using configuration metadata, then it will deal with the failures of the machine, in the system using the failure detect detection and the group membership, of the Zookeeper. And also it performs the control operation.

Refer slide time :(50:22)

ZooKeeper Applications: Yahoo! Message Broker

- The topics directory has a child znode for each topic managed by YMB.
- These topic znodes have child znodes that indicate the primary and backup server for each topic along with the subscribers of that topic.
- The primary and backup server znodes not only allow servers to discover the servers in charge of a topic, but they also manage leader election and server crashes.

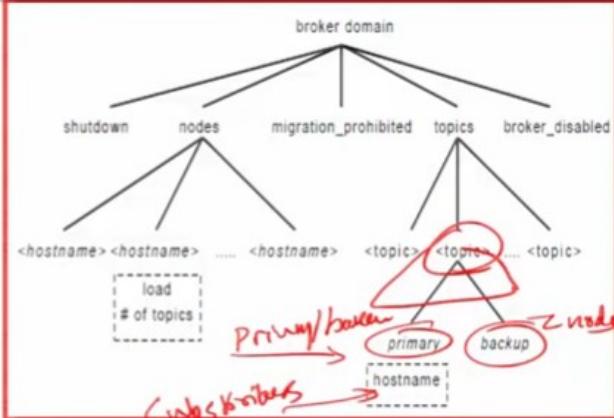


Figure: The layout of Yahoo! Message Broker (YMB) structures in ZooKeeper

So, this particular figure shows the part of znode layout, for Yahoo message broker system. And so, each broker, domain has the znode called the, 'Nodes', that has an ephemeral znodes for each of the active servers that compose of Yahoo message broker service. So, each YMB server creates an ephemeral node, under the nodes with the load and status information, providing both the group membership and the status information, using the Zookeeper of the yahoo message broker. So, this is shown here as per the layout of a yahoo message broker structure in the Zookeeper. So, here you can see that these are the different, ephemeral nodes, which maintains the, the topics and using the ephemeral znodes in this particular scenario. So, the topics directory, has topics directory as, the child znode for each topics, which is managed by the YMB these topics that is znodes have the children's you note that indicate the primary and the backup server, of each of each topic, for his topic along with the subscribers of that particular topic. So, these are the subscribers and these are the primary end and the backup, xenon. So, every topic is maintained as the so, each topic have the child znode that indicates the primary and the backup servers for each topic along with the subscriber of that topic. So, the primary and backup servers, that is znode not only allows the servers to discover the servers in the charge of the topic. But also manages the leader election and server crashes.

Refer slide time :(52:46)

More Details

ZooKeeper: Wait-free coordination for Internet-scale systems

Patrick Hunt and Mahadev Konar
Yahoo! Grid
`{phunt,mahadev}@yahoo-inc.com`

Flavio P. Junqueira and Benjamin Reed
Yahoo! Research
`{fpj,breed}@yahoo-inc.com`

See: <https://zookeeper.apache.org/>



So, more detail of Zookeeper, we can see through this particular paper, given written by Zookeeper weight free coordination for internet scale systems by Yahoo peoples.

Refer slide time :(53:02)

Conclusion

- **ZooKeeper** takes a wait-free approach to the problem of coordinating processes in distributed systems, by exposing wait-free objects to clients.
- **ZooKeeper** achieves throughput values of hundreds of thousands of operations per second for read-dominant workloads by using fast reads with watches, both of which are served by local replicas.
- In this lecture, we have discussed the basic fundamentals, design goals, architecture and applications of ZooKeeper.

Conclusion the Zookeeper takes a weight free approach, to the problem of coordinating processes in the distributed system. By exposing the weight free objects to the client. So Zookeeper achieves throughput values of hundreds and thousands of operation, per second for read dominant workloads, by using fast reads with the watches, both of which are served by the local replicas. So, in this lecture we have discussed the basic fundamentals design goal principles architectures and applications, of the Zookeeper which is a centralized coordination, service and which has these particular properties, such as,

Refer slide time :(53:49)

ZooKeeper Applications: Katta

- **Katta:** It is a distributed indexer that uses Zoo- Keeper for coordination, and it is an example of a non- Yahoo! application. Katta divides the work of indexing using shards.
- A master server assigns shards to slaves and tracks progress. Slaves can fail, so the master must redistribute load as slaves come and go.
- The master can also fail, so other servers must be ready to take over in case of failure. Katta uses ZooKeeper to track the status of slave servers and the master (**group membership**), and to handle master failover (**leader election**).
- Katta also uses ZooKeeper to track and propagate the assignments of shards to slaves (**configuration management**).

The use cases which we have seen about.

Refer slide time :(53:55)

What is Coordination ?

- **Group membership:** Set of datanodes (tasks) belong to same group
- **Leader election:** Electing a leader between primary and backup
- **Dynamic Configuration:** Multiple services are joining, communicating and leaving (Service lookup registry)
- **Status monitoring:** Monitoring various processes and services in a cluster
- **Queuing:** One process is embedding and other is using
- **Barriers:** All the processes showing the barrier and leaving the barrier.
- **Critical sections:** Which process will go to the critical section and when?

We have seen the use cases of the Zookeeper, as the group membership, where the set of nodes where set of data nodes, where belong to the same group, indifferent applications. We have also seen the use cases where the leader election, is done using the Zookeeper. I said as an service, we have also seen the dynamic configuration, aspects of Zookeeper and where in multiple services are joining and some are leaving, we have also seen the status monitoring, aspects our as a service of Zookeeper, where it has monitored various processes and services in the cluster. We have also seen about the queuing, aspect of this coordination service, we have seen barriers we have also seen the critical section. So, in nutshell where our while writing, the distributed application for Big Data computations, these different services are mashed together are collected to the other end is being provided by the Zookeeper. So, writing application, building application, with Zookeeper is going to be very very important.

Refer slide time :(55:19)

What is ZooKeeper ?

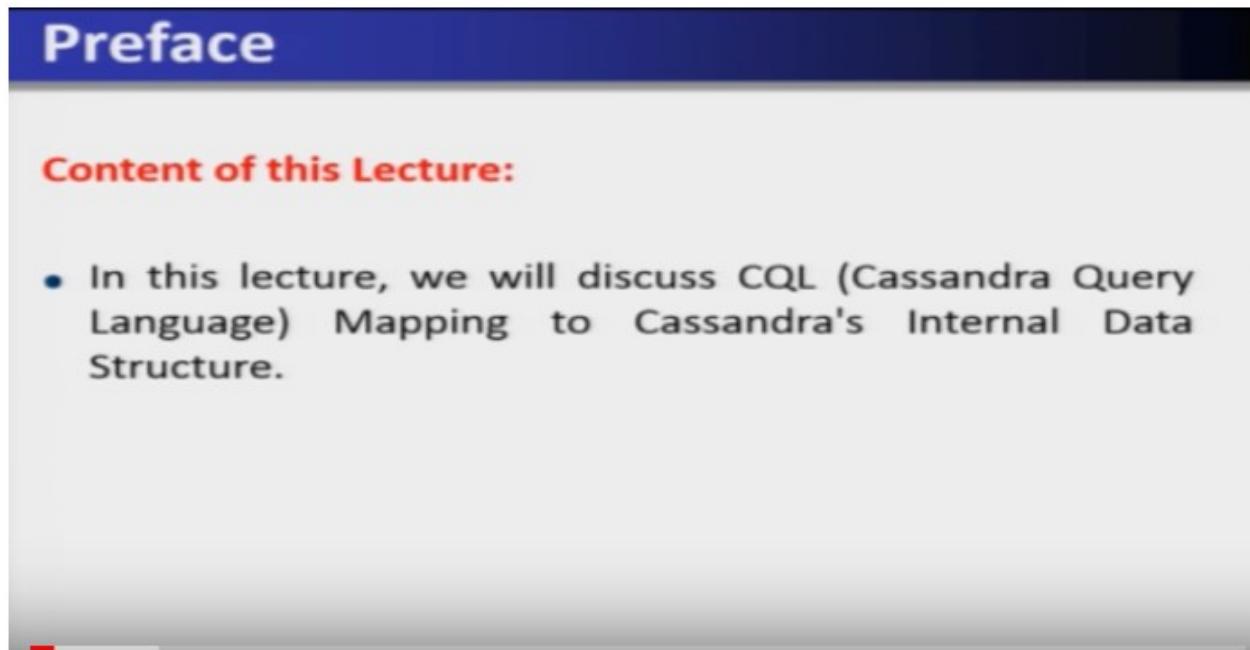
- **ZooKeeper is a highly reliable distributed coordination kernel,** which can be used for distributed locking, configuration management, leadership election, work queues,....
- Zookeeper is a replicated service that holds the metadata of distributed applications.
- **Key attributed of such data**
 - Small size
 - Performance sensitive
 - Dynamic
 - Critical
- **In very simple words,** it is a central store of key-value using which distributed systems can coordinate. Since it needs to be able to handle the load, Zookeeper itself runs on many machines.

And we will see in the further slides, how the Zookeeper is used in different applications for realizing these services. Thank you.

Lecture 17
CQL (Cassandra Query Language)

CQL Cassandra query language.

Refer slide time :(0:16)



The image shows a slide titled "Preface" in a blue header bar. Below the title, the text "Content of this Lecture:" is displayed in red. A bulleted list follows, describing the lecture's content.

Preface

Content of this Lecture:

- In this lecture, we will discuss CQL (Cassandra Query Language) Mapping to Cassandra's Internal Data Structure.

Preface content of this lecture; this lecture we will discuss, CQL that is Cassandra Query Language, its mapping to Cassandra's internal data structure, all these internal details, we are going to see in this part of the lecture.

Refer slide time :(0:34)

What Problems does CQL Solve?

- **The Awesomeness that is Cassandra:**

- Distributed columnar data store
- No single point of failure
- Optimized for availability (through "Tunably" consistent)
- Optimized for writes lightning fast writes
- fast reads
- Easily maintainable
- Almost infinitely scalable — good news to design
of Cassandra

What problem does the CQL is going to solve, to understand this let us first see, the awesomeness that is the Cassandra, Cassandra is a distributed columnar data store, that we have seen in the previous discussion, here in Cassandra there is no single point of failure, that is it is handled, it is handling the fault tolerance effectively, Cassandra is optimized for availability, although it is tunable, it is having a “tunably” consistency also, as far as the cap theorem is concerned, Cassandra is optimized, for the writes, here the writes are lightning, fast writes in the Cassandra, all the reads are also very fast. And this Cassandra is easy to maintain and also this Cassandra, is infinitely scalable, that is this is the good part, of the design of Cassandra. So, all these are adding the awesomeness, to the customer.

Refer slide time :(02:07)

What Problems does CQL Solve? (Contd.)

- **Cassandra's usability challenges**

- NoSQL: "Where are my JOINS? No Schema? De-normalize!?"
- BigTable: "Tables with millions of columns!?"

- **CQL Saves the day!**

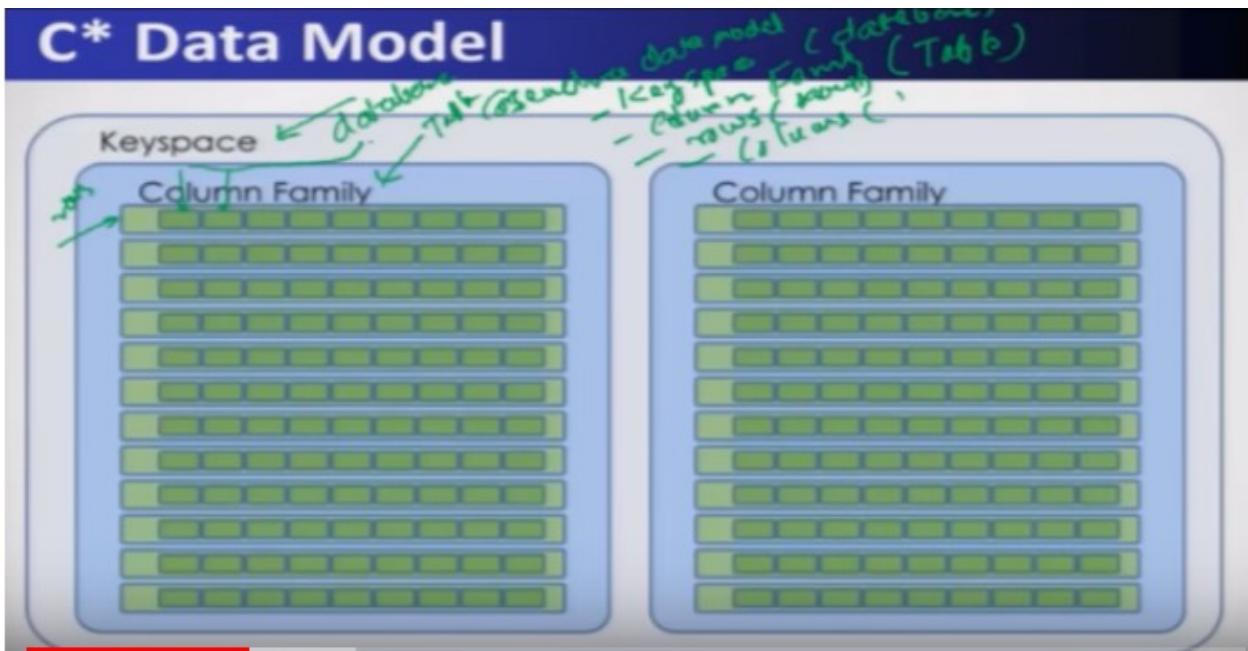
- A *best-practices* interface to Cassandra
- Uses familiar SQL-Like language

Theater mode (t)

Awesomeness to the Cassandra, what problem does CQL solve? We are going to see, further more detail that Cassandra's, usability challenges is that again, is of no CQL that is there are no joints and also no schema and there is a big table, which contains all the, columns hence there is no demon de normalization. So, big table and here, the tables with the millions of column, sexist in this Cassandra's, now the CQL saves the day that is the back best practices, which are interfaced to the Cassandra. And also Cassandra uses SQ Like languages; given all these let us see.

Refer slide time :(02:52)

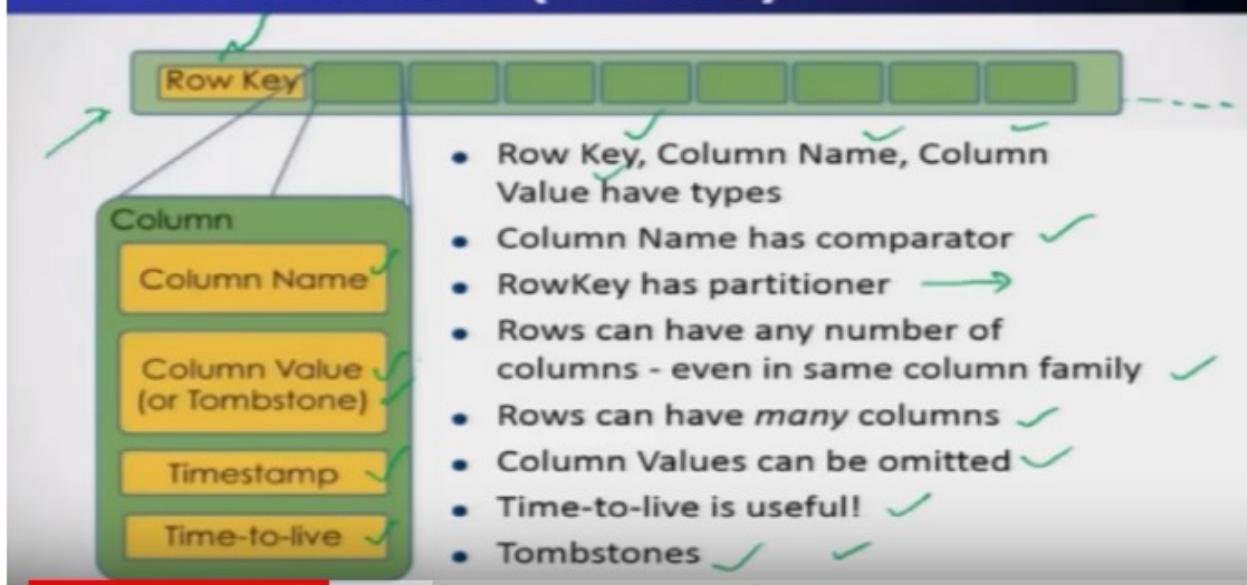
C* Data Model



The more detail of Cassandra data model. So, Cassandra data model consists of a key space key spaces the entire database, of RDBMS system and the column family is nothing, but the tables of database management system. So, so the Cassandra data model, consists of consists of key space and then it is divided, into the column families. So, column family is nothing but a table, in a database and this is the database, then column families will have, different rows. So, this is the rows of a table and you will have different columns. So, that different the rows have multiple columns. So, this particular aspect which we have now seen, this key space refers to the, database of RDBMS, with respect to RDBMS and column family is a reference to the table, in RDBMS and these are all called, 'Rows'. And every row is having multiple columns.

Refer slide time :(04:41)

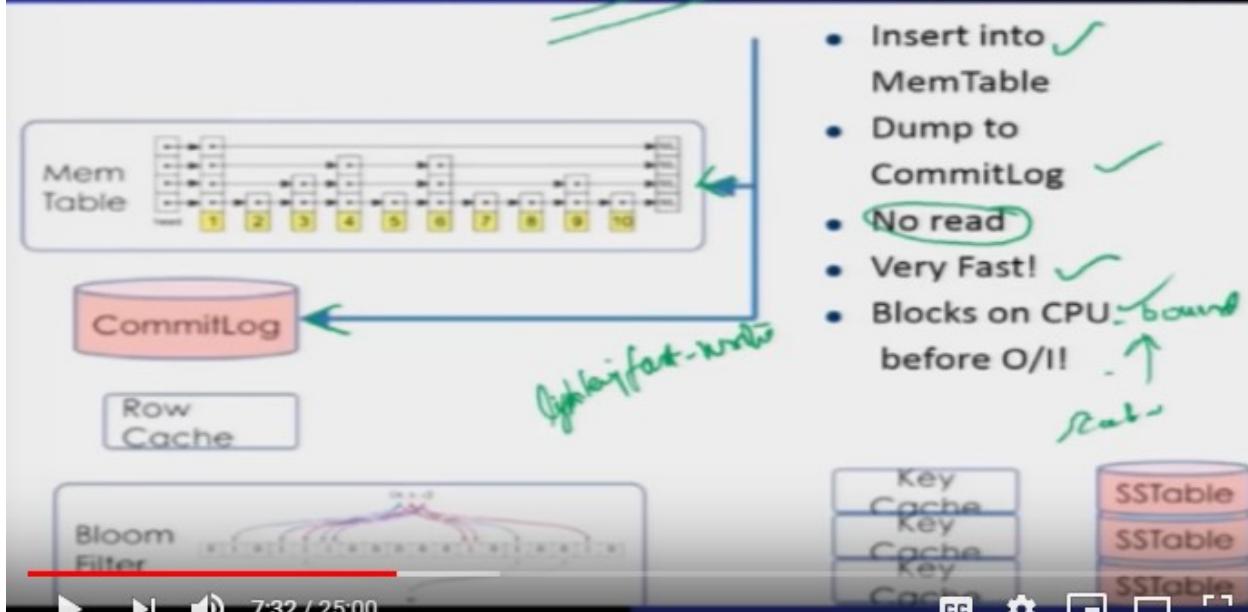
C* Data Model (Contd.)



So, this comprises of the data model, let us consider the row in word row of the column family. So, here the row contains the row key and further the every column contains, the name of the column, then column values or a tombstone, then times, time will tend to live. So, here the row key, column name, column values, how different types? And column name has a comparator and this particular comparator is, defined in Cassandra row key has, the partitioner. So, that means these row key will be, used to store, the rows on to the cluster or on different nodes and rows can have any number of columns, even in the same column families. So, so that means, this row can have any number of columns, it can be a millions of columns, also can be there in a row and rows can have many columns, column values can be, omitted here. And there will be the two more functions, one is called time to live, that is automatically after, this time to leave the data will be deleted. And there will be a tombstone that is if the, the data is deleted that particular flag will be set, it will not be immediately deleted by the system.

Refer slide time :(06:14)

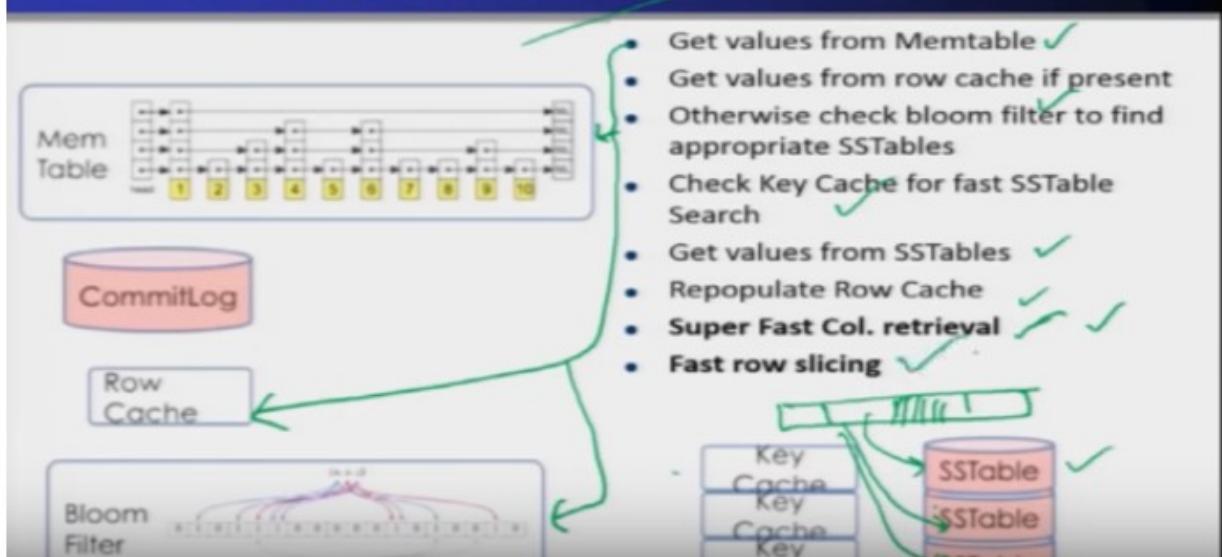
C* Data Model: Writes



Now let us see the data model, of Cassandra and we will see how the right operation is being performed. So, right operation virtually has to do nothing, in the Cassandra data model. So, here we can see that if there is a request to, write. So, it will be first go and write into the Memtable and then it will write into the commit log. So, commit log, will ensure the, reliability against the failures. So, that is why using this but there is no lead is required and hence this particular right operation, is very fast this is called, ‘Lightining’, first read/write operation is supported in the Cassandra, here we can see, also that these operations does not require any IO and all the operation, has to be done is an IO in CPU, bound operations is performed in the right operation, hence this requires, hence this is quite scalable. So, as we add more number of CPUs this will automatically, get scaled in linearly.

Refer slide time :(07:37)

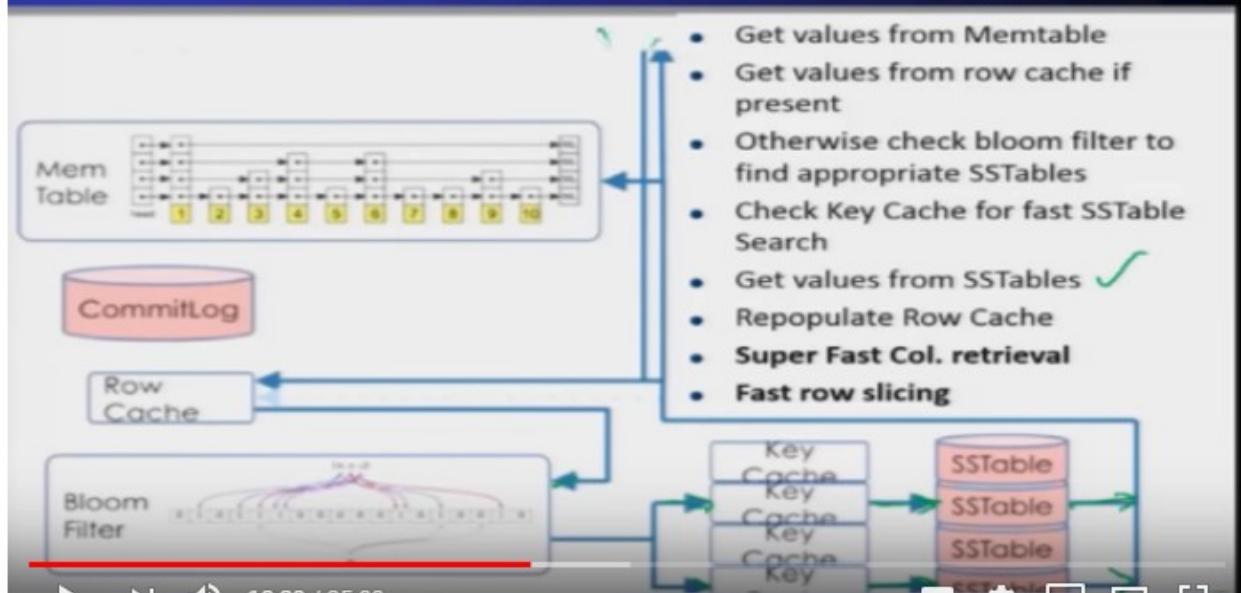
C* Data Model: Reads



As well as the write operation, the read is concerned, reads are supported, with several set of steps, in Cassandra. So, let us see how the read is supported, reads are not as fast as, the writes but still it is faster. So, it when the read operation, is invoked first it will try to get the values from the Memtable, it will first look up the Memtable and if the values are present, then it will be then operation, read operation, is done, if it is not there then get the values from the row cache. And if it is present, then it will be done otherwise, if it is not in this memory or in a row cache, then it has to go and search the disk, for these values, for that it goes and checks, the bloom filter, bloom filter is also a very fast, way of indexing or basically searching whether the data is, resides in which SSTable. So, checking the bloom filter is to find the appropriate as a SSTable. And so, it will, now check the using bloom filter after that it will go and check the key cache, for the first SSTable search and get the values from, after getting get, get the values from SSTable. And then repopulate the row cache, this is super fast, column retrieval and why because, you know that a row, maybe split into different, SSTables because the slices, of the rows of slices of the columns are stored, in different SSTables. So, multiple SSTables need to be consulted, therefore sometimes if, if a particular row contains, that column value which is required by the read operations and it will be extremely fast read, otherwise it has to access more than one as a SSTable and all the row entries, have to be now assessed. So, this way it will be possible, for the first row.

Refer slide time :(09:52)

C* Data Model: Reads (Contd.)



So, all these scenarios interactions, which we have shown here, for the read operation so, first it will go and check, the Memtable and row keys if it is not there, in the row key then it will be checking in into the bloom filter and based on that it will go and check, the in the key cache. So, key cash in turn will index into the SSTable and SSTable in turn will fetch these values which are required, use the read operation, get the values from the SSTable and give it back to the read operation.

Refer slide time :(10:33)

Introducing CQL

- **CQL is a reintroduction of schema** so that you don't have to read code to understand the data model.
- **CQL creates a common language** so that details of the data model can be easily communicated.
- **CQL is a best-practices Cassandra interface** and hides the messy details.

Now let us see, about Cassandra query language to access, this particular key value pair, this database Cassandra provides, easy interface, that is looking that is quite similar to SQL that is called, 'CQL'. Let us go and discuss more detail of it so; CQL is reintroduction of a schema. So, that you don't have to read the code to understand the data model. We need to say that the interface, by which the users are going to access, is quite familiar and this is nothing but similar to the SQL, but internally their implementations are quite different. So, how this CQL is into the internals of Cassandra that we will discuss in more details. So, CQL creates a common language. So, that the details of the data model can easily be communicated. So, CQL is the best practices, the Cassandra interface and hides many messy details.

Refer slide time :(11:35)

Introducing CQL (Contd.)

```
CREATE TABLE users (
    id timeuuid PRIMARY KEY,
    lastname varchar,
    firstname varchar,
    dateOfBirth timestamp );

INSERT INTO users (id,lastname,firstname,dateofbirth)
VALUES (now(),'Berryman','John','1975-09-15');

UPDATE users SET firstname = 'John'
WHERE id = f74c0b20-0862-11e3-8cf6-b74c10b01fc6;

SELECT dateofbirth,firstname,lastname FROM users ;

dateofbirth      | firstname | lastname
-----+-----+
1975-09-15 00:00:00.0400 | John | Berryman
```

So, to give a very, quick overview of what kind of construct CQL provides. So, for just like we have seen in SQL all these commands similar commands are available, in CQL that we will see in the further slides.

Refer slide time :(11:56)

Remember this:

- Cassandra finds rows fast
- Cassandra scans columns fast
- Cassandra *does not* scan rows

So, remember that Cassandra finds the Rows when you fast Cassandra, scans the columns very fast Cassandra does not scan through, the rows these are some of the, the features, of the Cassandra, how it performs how it does. So, we have to see all these particular and three different steps in our operations.

Refer slide time :(12:14)

The CQL/Cassandra Mapping

```
CREATE TABLE employees (
    name text PRIMARY KEY,
    age int,
    role text
);
```

name	age	role
john	37	dev
eric	38	ceo

rows are partitioned in cassandra

john	age	role
	37	dev

eric	age	role
	38	ceo

So, let us see that, what are the CQL commands, of Cassandra and how it is internally being mapped into the Cassandra. So, if there is a create table, CQL command that is the create table with the employee name. And name is the primary key and age and role are the two other attributes. Now let us populate, this particular database, having two tuples one is called, 'John', that is called, 'Eric'. Now as far as internal mapping of this CQL command, is concerned the entire operation. So, this particular name, is a primary key. So, the further couple for the row a key, that is John will become the row key, for the for this particular tuples. And corresponding age and the roll number and corresponding age and role, they will be having their different details, noted. So, this is the, the row key and this entire row is identified using row key and others are the column values, similarly Eric row key will be used as the row key and these are all column values. Now these rows are partitioned in Cassandra. So, these rows are now replicated and stored, in the cluster, having the row keys and they are being accessed very fast.

Refer slide time :(14:33)

Remember this:

- Cassandra finds rows fast ✓ (partition on rowkey)
- Cassandra scans columns fast
- Cassandra *does not* scan rows

So, as far as this particular, important point, which was there that Cassandra finds that was very fast why because the, the row keys, they are partitioned across the, they are partitioned, on the row keys, hence they are very fast to access.

Refer slide time :(14:47)

The CQL/Cassandra Mapping

```
CREATE TABLE employees (
    company text, ←
    name text,
    age int,
    role text,
    PRIMARY KEY (company,name)
);
```

Composite

company | name | age | role

company	name	age	role
OSC	eric	38	ceo
OSC	john	37	dev
RKG	anya	29	lead
RKG	ben	27	dev
RKG	chad	35	ops

	eric:age	eric:role	john:age	john:role
any:age	38	dev	37	dev
RKG	29	lead	27	dev

Now the next example which we will see about CQL is that we are going to have added one more attribute, that is called, 'Company'. And now the primary key, is now the composite key, now let us see the, the table or a column family, which will look like in this manner and it has two of the, there are two tuples with the OSC company name and there are three tuples with RKG company name. Let us see that five different tuples will be converted into two different row keys. So, the first row key is OSC and the name is the Eric. And so, OSC has two different Eric names so, both are here, in that same row key, which is having the value OSC similarly it has the, the age values, column values, those values are also added. So, this becomes a row key similarly RKG will be the another row key and this will have this name Anya and Ben and Chaid there each and roll their values or column values are now, put into the different column values. So, so this will be partitioned across using the row key and will be stored into the Cassandra. So, this is the CQL to the Cassandra, internal mapping.

Refer slide time :(16:42)

The CQL/Cassandra Mapping (Contd.)

CREATE TABLE example (

A text,

B text,

C text,

D text,

E text,

F text,

PRIMARY KEY ((A,B),C,D)

);

A | B | C | D | E | F

-+-----+

a | b | c | d | e | f

a | b | c | g | h | i

a | b | j | k | l | m

a | n | o | p | q | r

s | t | u | v | w | x

	c:d:E	c:d:F	c:g:E	c:g:F	j:k:E	j:k:F
a:b	e	f	h	i	l	m
	o:p:E	o:p:F			u:v:E	u:v:F
a:n	q	r			w	x

Similarly we can extend it into the more complicated, example where in the primary key, is now another a composite of four different keys, A B and then C D. So, A B is one of the primary key and then C D and their values are corresponding, values are now put in this column similarly for a 2 n. Now let us consider about s 2 T so, this is the a B they will become the primary key and the CD UV and their CD values are now noted down, as for evaluate will be now W and for F value it will be X and U and V these values, will come as the column attributes. So, this way the

Refer slide time :(17:41)

CQL for Sets, Lists and Maps

- Collection Semantics
 - Sets hold list of unique elements ✓
 - Lists hold ordered, possibly repeating elements ✓
 - Maps hold a list of key-value pairs
- Uses same old Cassandra datastructure
- Declaring

```
CREATE TABLE mytable(  
    X text,  
    Y text,  
    myset set<text>,  
    mylist list<int>,  
    mymap map<text, text>,  
    PRIMARY KEY (X,Y)
```

Collection fields
can not be used
in primary keys

entire row key will be mapped, now CQL also supports the sets list and map. So, they are the collect they are the collection semantics. So, set holds the list of unique elements, set means the list of unique elements. And the list is nothing but an ordered possibly repeating the elements that is called the, 'Set'. And map is the list of key value pairs, now uses same old Cassandra data structure. Let us see how we can declare, these things, that my set is equal to set and a text and my list is a list of integers and my map will be the map of key value pairs and note that these fields that is set list and map cannot, cannot be the primary key. So, the primary key will be, from the normal attributes, that x and y can be the primary key.

Refer slide time :(18:59)

Updating

```
UPDATE mytable SET myset = myset + {'apple','banana'}
```

```
WHERE row = 123;
```

```
UPDATE mytable SET myset = myset - { 'apple' }
```

```
WHERE row = 123;
```

```
UPDATE mytable SET mylist = mylist + ['apple','banana']
```

```
WHERE row = 123;
```

```
UPDATE mytable SET mylist = ['banana'] + mylist
```

```
WHERE row = 123;
```

```
UPDATE mytable SET mymap['fruit'] = 'apple'
```

```
WHERE row = 123
```

```
UPDATE mytable SET mymap = mymap + { 'fruit':'apple' }
```

```
WHERE row = 123
```

Now there are some operations, all these operations are supported, to insert into the into, the table, these values, whether it is lists or it is set or it is the, the map or it is the normal, very values these values can be inserted into the using insert into the command, of off CQL similarly we can update, it we can add and we can delete the values.

Refer slide time :(19:22)

SETS

```
CREATE TABLE mytable(  
    X text,  
    Y text,  
    myset set<int>, ✓✓✓  
    PRIMARY KEY (X,Y)  
);
```

X | Y | myset

---+---+-----✓

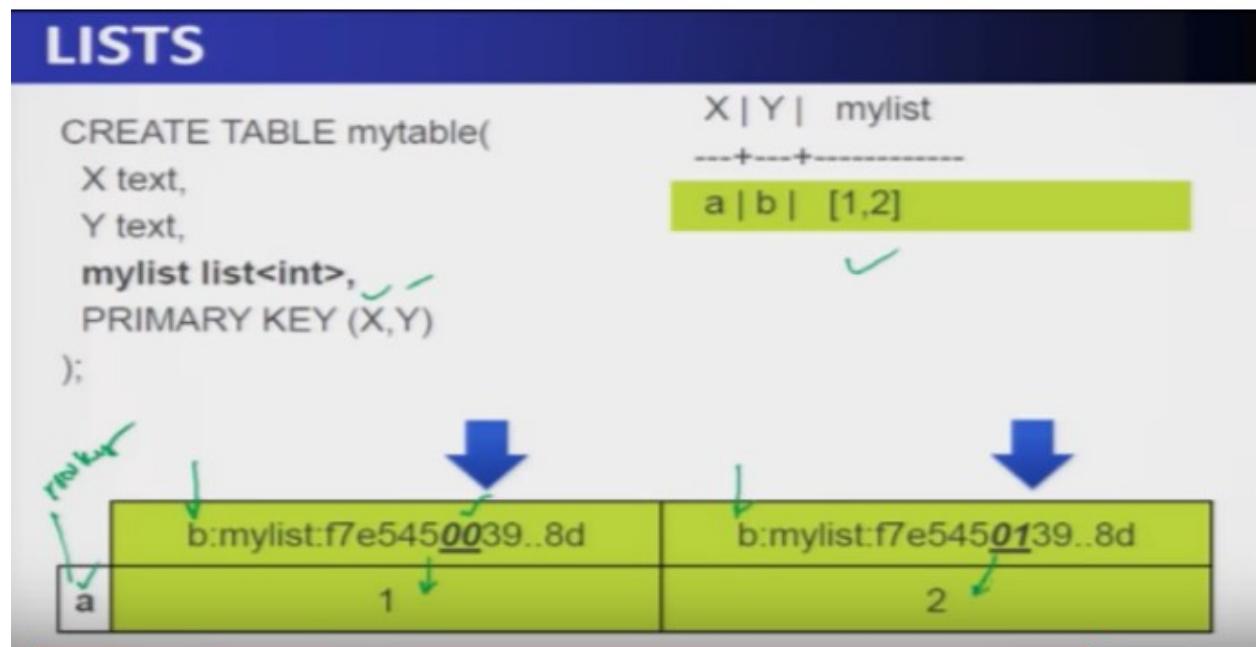
a | b | {1,2} ✓✓

a | c | {3,4,5} ✓

	b:myset:1	b:myset:2	c:myset:3	c:myset:4	c:myset:5
a	-	-	-	-	-

Now let us see about the set how it is being mapped into the Cassandra data model. So, that means if we are given, the CQL command for CQL query for create table, by table and here we are using the set, my set and the primary key will be XY. So, the set will be the set of elements, here two sets will be given. Now this entry is shown like this that a will be the row key. And as far as the columns are concerned so, primary key are x and y. So, Y will be the on the primary key, so, so Y will be 4b and 4c, now for B there are two different members, in, in the set, two different elements in the set. So, my set with the member one, my set with a member tow, with a member three, four and five. Now you see this there will be no values in this column values and everything will be there in the column attributes. So, this will be the internal mapping, of the Cassandra sets into the,

Refer slide time :(20:53)



into the Cassandra details so, another construct which is there in Cassandra is called a, 'List'. Now let us see the example of a list here, the elements of the list may be repeating also. So, the primary key is X and Y so, here a will be there as a row key and the other row key will be the B which will be there on the column, together they will identify, this part in our my list values. So, my list is some arbitrary value. So, in that there will be two values so, 0, 0 and 0 1 will the first value will be 1, the other value will be 2, in this way all these list values are being used here in this case.

Refer slide time :(21:41)

MAPS

```
CREATE TABLE mytable(  
    X text,  
    Y text,  
    mymap map<text,int>,  
    PRIMARY KEY (X,Y)  
)
```

X | Y | mymap

---+---+

a | b | {m:1,n:2}

a | c | {n:3,p:4,q:5}

	b:mymap:m	b:mymap:n	c:mymap:n	c:mymap:p	c:mymap:q
a	1	2	3	4	5

So, Maps means key value store so, key and value will be stored, here in this case let us see this example also. So, A and B they are the primary keys, A and B they are the primary keys they are restored and from the key value pair from the map. So, key will be M and the value will be stored over here. So, therefore in the second row A and C will be the primary key and they are key value pair that is from my map. And 4n the value will be 3 and 4 P with the value will be 4 and it will 5 in this way, this particular map, of Cassandra CQL will be stored internally.

Refer slide time :(22:27)

Example

(in cqlsh)

```
CREATE KEYSPACE test WITH replication =  
    {'class': 'SimpleStrategy', 'replication_factor': 1};  
USE test;  
CREATE TABLE stuff ( a int, b int, myset set<int>,  
    mylist list<int>, mymap map<int,int>, PRIMARY KEY (a,b));  
UPDATE stuff SET myset = {1,2}, mylist = [3,4,5], mymap = {6:7,8:9} WHERE a = 0  
AND b = 1;
```

```
SELECT * FROM stuff;
```

(in cassandra-cli)

```
use test;
```

```
list stuff;
```

(in cqlsh)

```
SELECT key_aliases,column_aliases from system.schema_columnfamilies WHERE
```

So, let us see some examples, of CQL so, we can create first the key space, which is named as a, 'Test', with the replication factor is called 2:1 all these things we have already, studied and the petitioner is simple strategy, which is being used, for replication or storing the data and then we have created a column table, column family that is called, 'Create Table'. And the name is stuff and all these are the elements and the primary key will be a, b and we can perform an update, into it and then we can perform the select, operations. And from select all the values from the stuff which we have just, defined and furthermore we can create, the soul select, using key alias column alias, from the system is schema column families, where key space name is equal to test and column family stuff. So, we have seen that, the CQL is very familial and it looked like SQL commands, but internally it is being handled in a different manner that we have already discussed and covers.

Refer slide time :(23:57)

Conclusion

- CQL is a reintroduction of schema
- CQL creates a common data modeling language
- CQL is a best-practices Cassandra interface
- CQL lets you take advantage of the C* Data structure
- CQL protocol is binary and therefore interoperable with any language
- CQL is asynchronous and fast (Thrift transport layer is synchronous)
- CQL allows the possibility for prepared statements

Conclusion CQL is a reintroduction, of the schema CQL creates a common data Modern Language, CQL is the best practices, the Cassandra' interface and CQL lets you take advantage of the data structure, of the Cassandra CQL protocol, is the Binary and therefore interoperable with the languages, CQL is asynchronous and fast, CQL allows possibility for prepared statements. Thank you.

Lecture - 18
Design of HBase

Design of HBase,

Refer Slide Time :(0: 17)

Preface

Content of this Lecture:

- In this lecture, we will discuss:

- What is HBase?
- HBase Architecture
- HBase Components
- Data model
- HBase Storage Hierarchy
- Cross-Datacenter Replication
- Auto Sharding and Distribution
- Bloom Filter and Fold, Store, and Shift

Preface content of this lecture:

In this lecture we will discuss:

1. What is HBase?
2. HBase architecture
3. HBase components
4. Data model
5. HBase storage hierarchy
6. Cross-data center replication
7. Auto sharding and distribution
8. Bloom filter and fold, store and shift.

Refer Slide Time :(0: 40)

HBase is:

- An open source NOSQL database.
- A distributed column-oriented data store that can scale out horizontally to 1,000s of commodity servers and petabytes of indexed storage.
- Designed to operate on top of the Hadoop distributed file system (HDFS) for scalability, fault tolerance, and high availability.
- Hbase is actually an implementation of the **BigTable** storage architecture, which is a distributed storage system **developed by Google**.
- Works with structured, unstructured and semi-structured data.

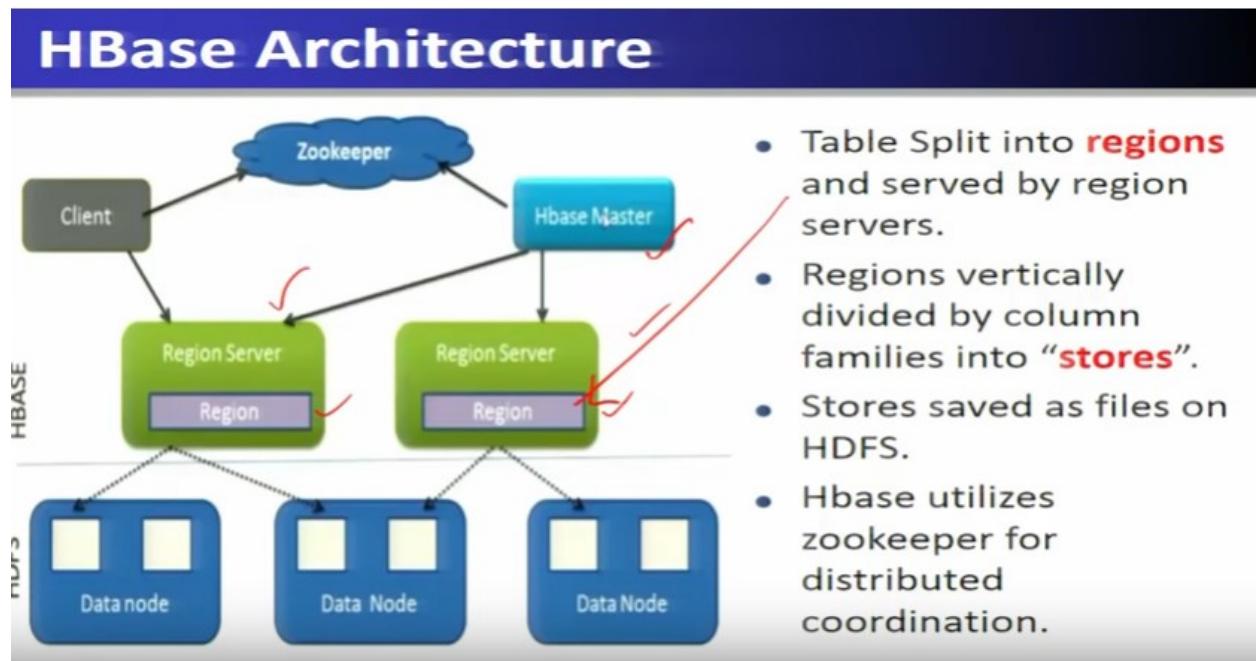
So, HBase is an open source NoSQL database. It is a distributed column-oriented data store that can scale out horizontally to thousands of commodity servers and Petabytes of indexed storage because this amount of commodity machines or the cluster is required to store the large amount of data that is why?. This particular aspect is covered into the distributed column-oriented data store and also it can scale out horizontally to thousands of servers and Petabytes of indexed storage. HBase is designed to operate on top of HDFS, for scalability, fault tolerance and high availability. HBase is actually an implementation of Big Table, which is our storage architecture, given by Google and this is which a distributed storage system, developed at Google. So, HBase works with a structured, unstructured and semi-structured data.

Refer Slide Time :(1: 59)

HBase

- **Google's BigTable** was first “blob-based” storage system
- **Yahoo!** Open-sourced it → HBase
- Major Apache project today
- **Facebook** uses HBase internally
- **API functions**
 - Get/Put(row)
 - Scan(row range, filter) – range queries
 - MultiPut – *multiple Key, value write*
- Unlike Cassandra, HBase prefers consistency (over availability)
Availability *Consistency*

So, HBase is basically designed as the Google's, Big Table, was first “block-based” storage system and Yahoo! Open sourced this particular concept, a block based and which is now known, as HBase. So, HBase is a major Apache project today and Facebook also uses HBase internally and it has various API functions which provides for example: get and put by row that is the key-value pair and scan the row range and filter all these operations which are supported to perform the range queries, it is also having a multiput. So, that multiple key-value store can be stored and handled at the same time. Unlike Cassandra, HBase prefers consistency over availability. Now, Cassandra prefers availability whereas HBase prefers consistency over availability. So, now we are going to see that where consistency is a preferred option for the application HBase is used and wherever availability is more important than Cassandra is used. That is why both of them exist as the NoSQL solution.



51)

Let us see some of the important aspects of the HBase architecture. So, HBase has the region servers and these region servers are basically handling the regions and there is one HBase master and this zookeeper has to interact with the HBase master and all the other component and HBase, then as HBase also, deals with the data nodes. So, HBase master has to communicate with the region servers and zookeeper. So, we will see, in more detail about this. So, HBase architecture here, the table is split into the regions and served by the region servers. So, regions are vertically divided by the column families into the stores that we will discuss later on. And stores saved as files on HDFS. HBase utilizes zookeeper for the distributed coordination service.

Refer Slide Time :(5: 11)

HBase Components

- **Client:**
Finds RegionServers that are serving particular row range of interest
- **HMMaster:**
Monitoring all RegionServer instances in the cluster
- **Regions:**
Basic element of availability and distribution for tables
- **RegionServer:**
Serving and managing regions
In a distributed cluster, a RegionServer runs on a DataNode

The components in more detail. So, client will find the region servers that are serving particularly the row range of interest. So, then HMaster monitors all the region servers instances in the cluster system. Regions are our basic element of availability and distribution for the tables. Our region servers serving and managing the regions and in a distributed cluster region runs on the data node.

Refer Slide Time :(5: 42)

Data Model

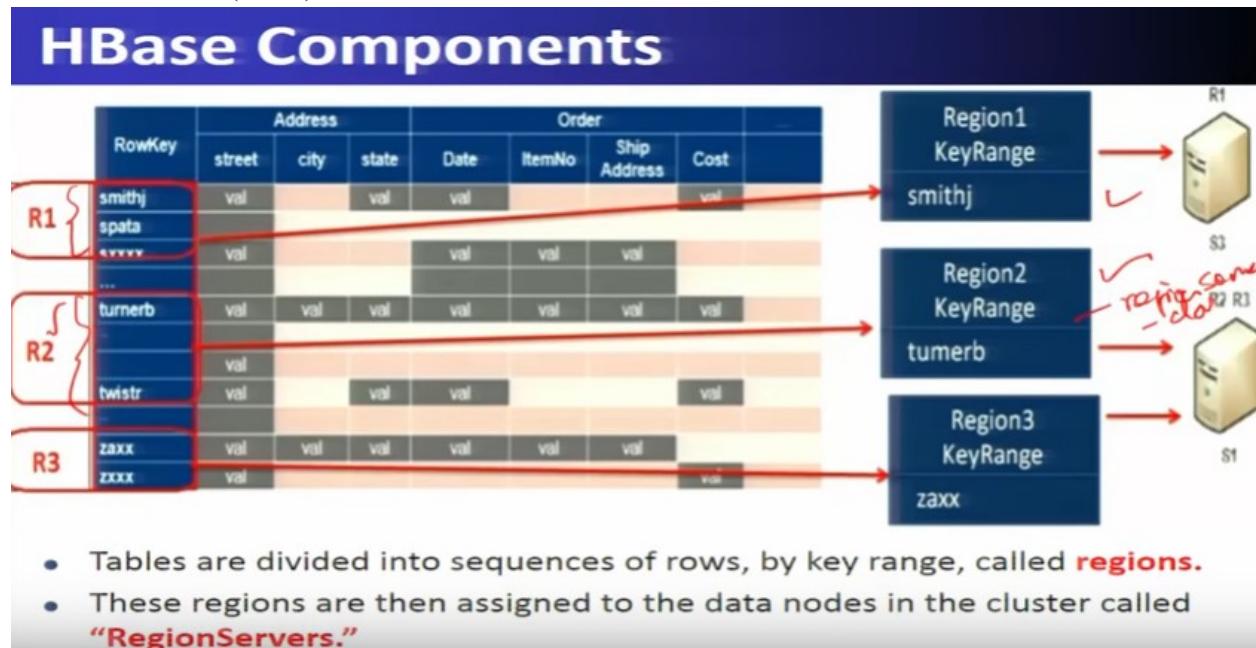
RowKey	Address				Order		
	street	city	state	Date	ItemNo	Ship Address	Cost
mithy	val		val	val			val
pata							
xxx	val			val	val	val	
nerb	val	val	val	val	val	val	val
	val						
istr	val		val	val			val
xx	val	val	val	val	val	val	
xx	val						val

Column families

- Data stored in Hbase is located by its “rowkey”
- RowKey is like a primary key from a relational database.
- Records in Hbase are stored in sorted order, according to rowkey.
- Data in a row are grouped together as Column Families. Each Column Family has one or more Columns
- These Columns in a family are stored together in a low level storage file known as **HFile**

So, this is the typical layout of the data model which is also called as the, ‘Column Families’. So, data is stored in HBase and is located by its “rowkey”. So, “rowkey” is the primary key from the notion of relational database management system. So, the records in HBase are stored in the sorted order according to the rowkeys and the data in a row are grouped together as the column families and each column family has one or more columns. These columns in a family are stored together in a low level storage file which is called as, ‘HFile’.

Refer Slide Time :(6: 28)



So, the tables are divided into the sequence of rows, by the key range called ‘Regions’. So, here we can see that, this particular key range is basically nothing but, the row 1 and this will be, stored together into the region. So, these regions are then assigned to the data node in the cluster called the ‘Region Servers’. Now, that is shown over here. So, again for example, let us say that the range the sequence or the key range, the “rowkey” range R2 will store these set off the rows, sequence of the rows and this will be stored in another region server and the region servers, these regions are managed by the data nodes, they are called they are called the ‘Region Servers’. Region servers as the data nodes.

Refer Slide Time :(7: 42)

Column Family

Row Key	Personal Data		ProfessionalData	
Emp Id	Name	City	Designation	Salary
I01	John	Mumbai	Manager	10L
I02	Geetha	New Delhi	Sr.Software Engineer	8L
I03	Smita	Pune	Programmer Analyst	4L
I04	Ankit	Bangalore	Data Analyst	12L



- A column is identified by a Column Qualifier that consists of the Column Family name concatenated with the Column name using a colon. ex-personaldata:Name *Column name*
- Column families are mapped to storage files and are stored in separate files, which can also be accessed separately.

Now, column family a column is identified by the column qualifier that consists of the column family name concatenated with the column name using the colon ex- personaldata:name. So, here we can see that, this particular column family name column is identified by the columns of the column family name concatenated with the column name using the colon. So, this is the column name and the column family name is shown over here this is the column family name. So, this is shown over here. So, column family name and the name will qualify, will identify the particular column. So, column families are mapped to the storage files and are stored in separate files which can also be accessed separately.

Refer Slide Time :(8: 53)

Cell in HBase Table

Row Key	Column Family	Column Qualifier	Timestamp	Value
John	PersonalData	City	123456790123	Mumbai

KEY ✓ VALUE

- Data is stored in HBASE tables Cells.
 - **Cell is a combination of row, column family, column qualifier and contains a value and a timestamp**
 - The key consists of the row key, column name, and timestamp.
 - The entire cell, with the added structural information, is called **Key Value**.

Now, cell in HBase. A table data is stored in HBase table cells. And cell is a combination of row, column family, column qualifier and contains a value and a timestamp. So, the key consists of the row key, column name and a timestamp that is shown here and the entire cell, with the added, structural information is called the, ‘Key Value Pair’.

Refer Slide Time : (9: 21)

HBase Data Model

- **Table:** Hbase organizes data into tables. Table names are Strings and composed of characters that are safe for use in a file system path.
- **Row:** Within a table, data is stored according to its row. Rows are identified uniquely by their row key. Row keys do not have a data type and are always treated as a byte[] (byte array).
- **Column Family:** Data within a row is grouped by column family. Every row in a table has the same column families, although a row need not store data in all its families. Column families are Strings and composed of characters that are safe for use in a file system path.

So, HBase data model consists of a table. HBase organizes data into the table and table names are the strings composed of the characters that are safe for use in the file system path. And the rows within the table, the data is stored according to its row. Rows are identified by the arrow key and row keys do not have the data type and are always treated as the byte [] (byte array). So, this aspect we have already covered. So, column family the data within the row is grouped by the column family. Every row in the table has the same column family, although the row need not store the data in all its families. Column families are the strings and composed of characters that are safe for use in the file system path.

Refer Slide Time :(10: 08)

HBase Data Model

- **Column Qualifier:** Data within a column family is addressed via its column qualifier, or simply , column, Column qualifiers need not be specified in advance. Column qualifiers need not be consistent between rows. Like row keys, column qualifiers do not have a data type and are always treated as a byte[].
- **Cell:** A combination of row key, column family, and column qualifier uniquely identifies a cell. The data stored in a cell is referred to as that cell's value.

Now, column qualifier: the data within the column family is addressed via the column qualifier and or simply the column qualifier need not be specified in the advance and column qualifier need not be consistent between the rows. Like "row keys" column qualifier do not have the data type and is always treated as a byte []. Cell is a combination of row key, column family and column qualifier uniquely identifies the cell. The data is stored in the cell is referred to as the cell value.

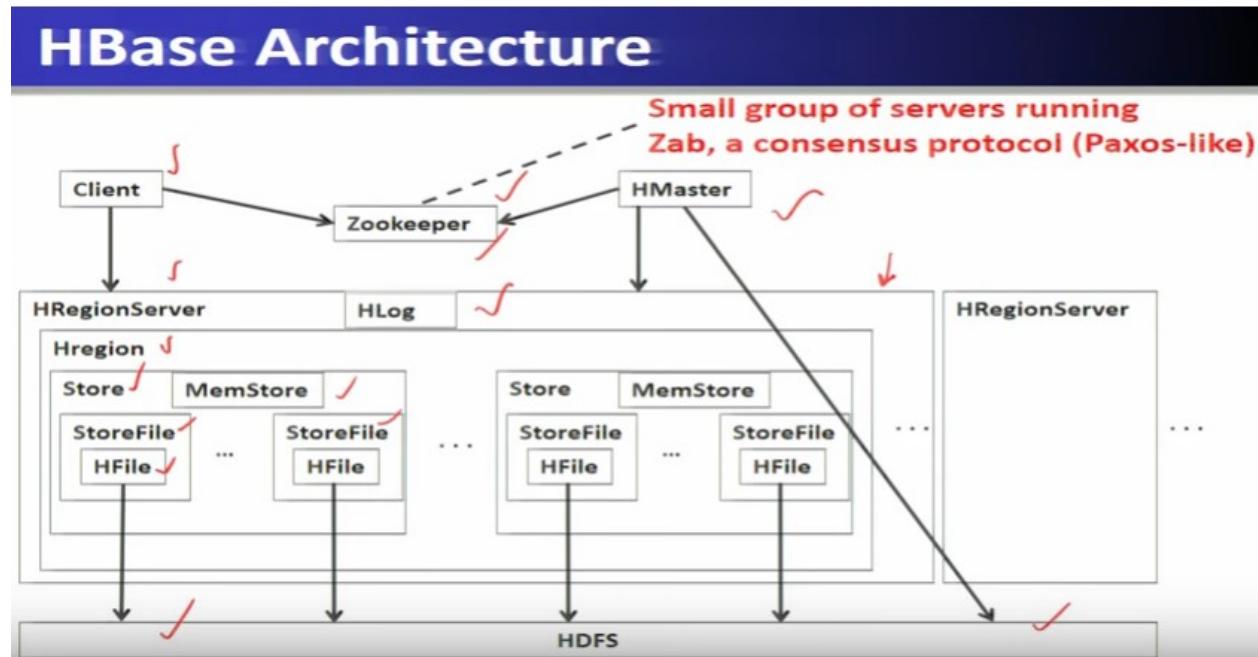
Refer Slide Time :(10: 41)

HBase Data Model

- **Timestamp:** Values within a cell are versioned. Versions are identified by their version number, which by default is the timestamp is used.
- If the timestamp is not specified for a read, the latest one is returned. The number of cell value versions retained by Hbase is configured for each column family. The default number of cell versions is three.

And timestamp the values within the cell are versioned. Versions are identified by their version number, which by default is the timestamp. If the timestamp is not specified for the read, the latest one is returned. The number of cell values versions retained by the HBase is configured for each family. The default number of cell versions is three.

Refer Slide Time :(11: 03)



Let, us see in more detail once again the HBase architecture. So, HBase has this one a client. Client can access to the HRegionServer and this HRegionServer are many HRegionServers. one such HRegionServer is shown over here which has HLog and each HRegionServer is further divided into different Hregions. One such Hregion is shown over here and Hregions will contain the store and also a MemStore. So, within the store it will be having the StoreFile and StoreFile will contain basic storage that is called, 'HFile' and 'HFile' is stored in HDFS. Now, as far as there is one HMaster and HMaster communicates with the zookeeper and with the HRegionServers and HDFS, we will see, we have seen about the HMaster and what is zookeeper? Is a small group of servers which runs consensus protocol like Paxos and the zookeeper is the coordination service for HBase and assigns the different nodes and servers to this particular service if zookeeper is not there then HBase will stop functioning.

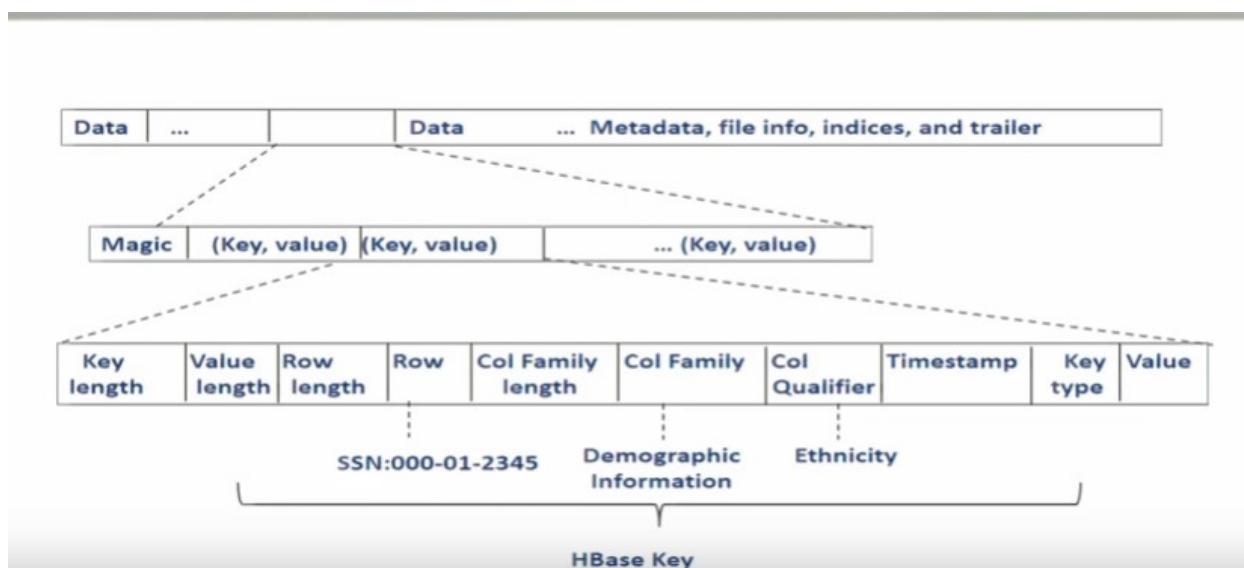
Refer Slide Time :(12: 38)

HBase Storage Hierarchy

- **HBase Table**
 - Split it into multiple regions: replicated across servers
 - ColumnFamily = subset of columns with similar query patterns
 - One Store per combination of ColumnFamily + region
 - Memstore for each Store: in-memory updates to Store; flushed to disk when full
 - » StoreFiles for each store for each region: where the data lives
 - Hfile
 - **HFile**
 - SSTable from Google's BigTable

Now, let us see the HBase storage hierarchy. So, HBase has HBase table which is split into the multiple regions which is replicated across the servers. Now, then it will be having a column family which is a subset of the columns with similar query and one store per combination of column family plus a region is there. And also, it has the Memstore for each store: in-memory updates to store and flush to the disc when full, that like we have seen in the Cassandra. So, StoreFiles for each store for each region where the data lives and within that contains the basic and basic HFile and as HFile will be stored in HDFS. So, each HFile is a SSTable from the Google's Big Table.

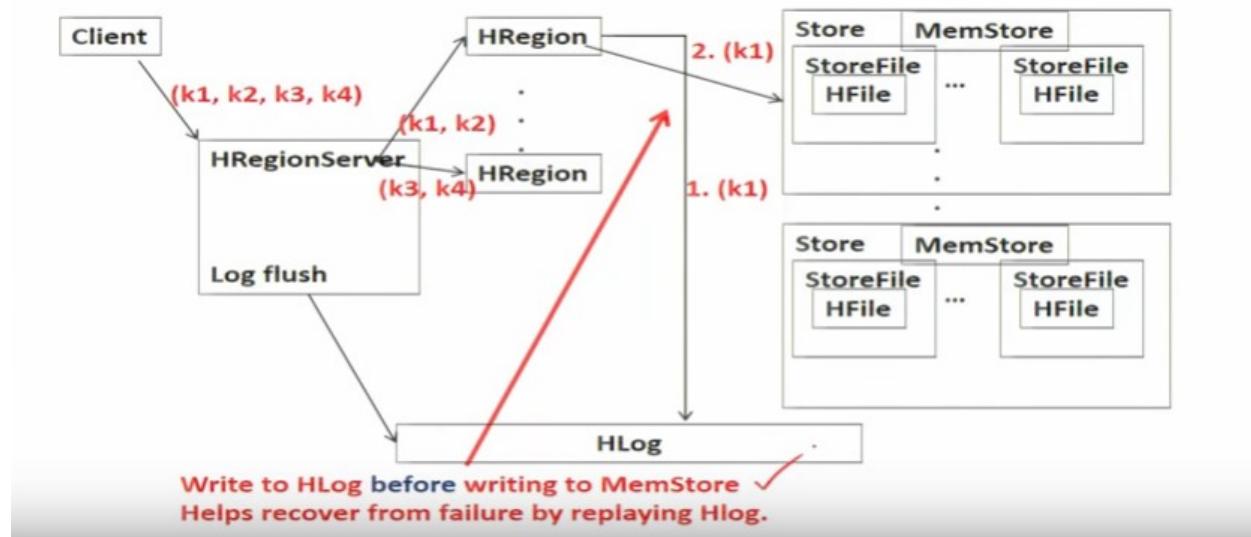
Refer Slide Time :(13: 41)



So, this is about the HBase HFile. So, HFile comprises, of data and Meta file, Metafile information indices and trailer where the data portion consists of the magic value and (key, value) pairs. So, the key value pair is having the key length, value length, row length, row and column family length, column family, column qualifier, timestamp, key type and value. And these rows will have SSN numbers and column family will have the demographic information and column qualifier will have ethnicity information and this becomes the HBase key.

Refer Slide Time :(14: 30)

Strong Consistency: HBase Write-Ahead Log



Now, HBase prefers the strong consistency or availability. So, HBase prefers the write-ahead log and whenever a client comes with the key (K 1, K 2, K 3, K 4) gives to the HRegionServer then this let us say, (K 1 and K 2) will be on a particular HRegion and (K 3, K4) will have another H Region and then this particular aspect will be stored in the store and these store will have the MemStore, StoreFiles and internally they are stored in as HFile. So, right to HLog before writing to MemStore is there to ensure the fault tolerance aspect and recovery from the failures. So, this helps recover from the failure by replaying HLog.

Refer Slide Time :(15: 35)

Log Replay

- After recovery from failure, or upon bootup (HRegionServer/HMaster)
 - Replay any stale logs (use timestamps to find out where the database is with respect to the logs)
 - Replay: add edits to the MemStore

So, Log Replay after the recovery from the failure or upon boot up, HRegionServer and HMaster will do the replay. So, replay any stale logs using the timestamp to find out where the database is with respect to the logs and replay will add it to the MemStore.

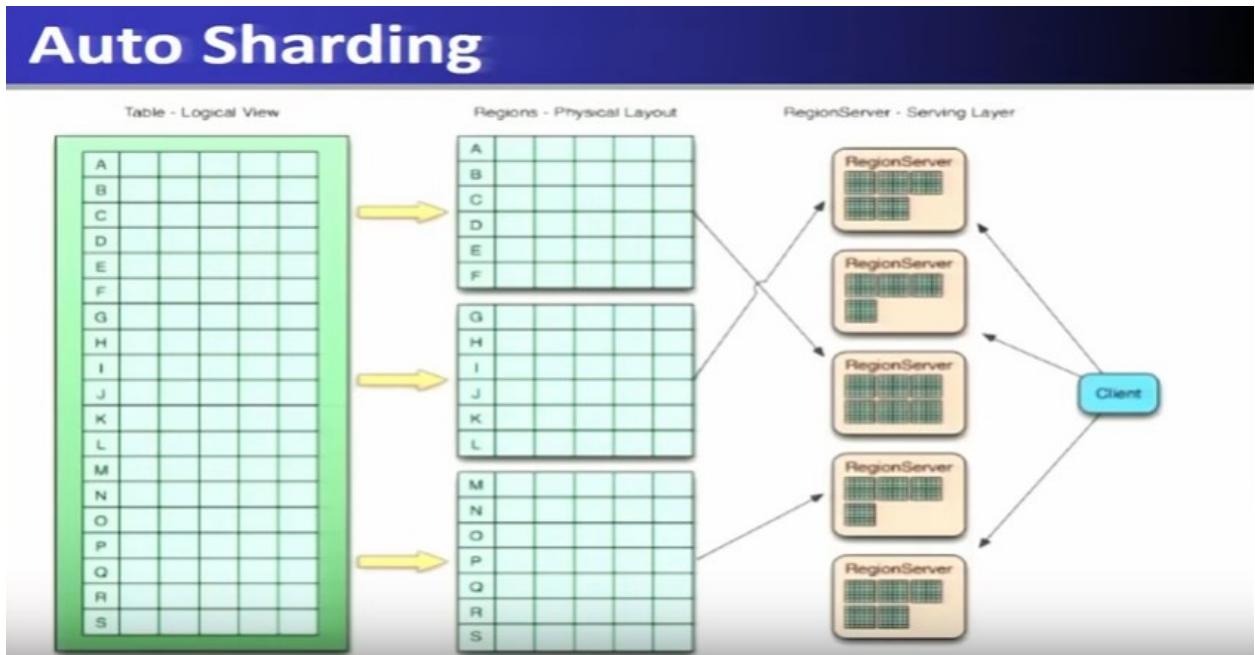
Refer Slide Time :(15: 53)

Cross-Datacenter Replication

- Single “**Master**” cluster
- Other “**Slave**” clusters replicate the same tables
- Master cluster synchronously sends HLogs over to slave clusters
- Coordination among clusters is via Zookeeper
- Zookeeper can be used like a file system to store control information
 - 1. `/hbase/replication/state`
 - 2. `/hbase/replication/peers/<peer cluster number>`
 - 3. `/hbase/replication/rs/<hlog>`

Now, cross-datacenter replication now there will be a single “Master” cluster, others “Slave” cluster replicate the same table, master cluster synchronously sends HLogs over the slaves clusters, coordination among the cluster is done by a zookeeper. Zookeeper can be used like a file system to store the control information and also this particular zookeeper will use different paths for invoking the state and peer cluster number and each log all this information is there.

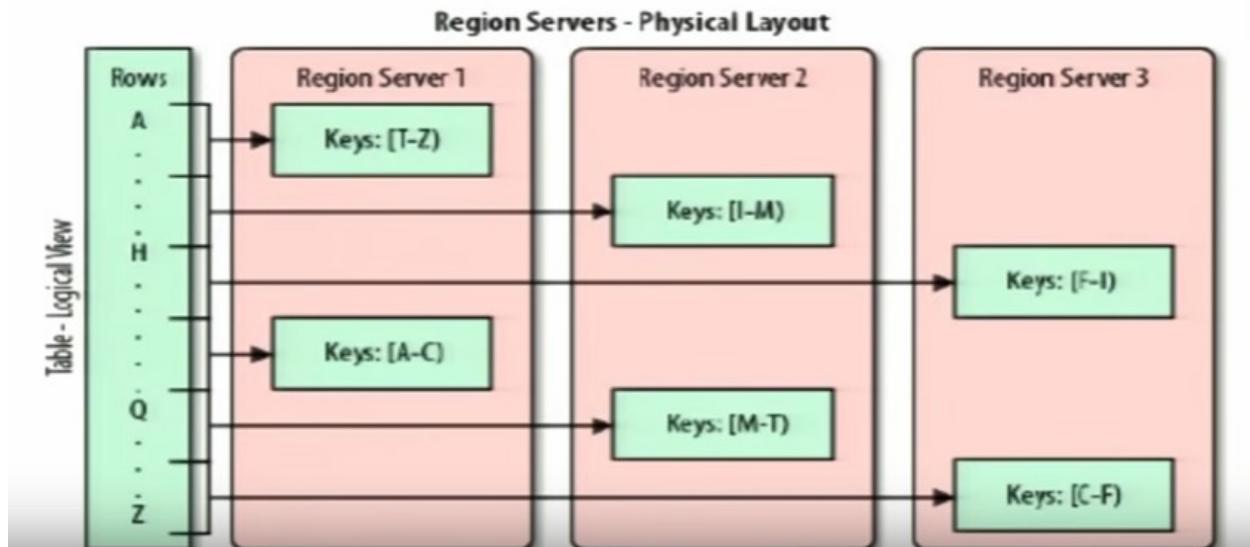
Refer Slide Time :(16: 29)



Now, let us see how the auto sharding is done and Auto sharding means that tables is divided into the row, range or a range keys and they are being stored in the region servers and we and these region servers are now solved with the client.

Refer Slide Time :(16: 50)

Distribution



Now, similarly the table logical view, we can see that the rows are now split into the row keys, in the range keys and these range keys are given charted into the region servers and we have shown here that these rows from A to Z are stored in three different region servers.

Refer Slide Time :(17: 11)

Auto Sharding and Distribution

- Unit of scalability in HBase is the Region
- Sorted, contiguous range of rows
- Spread “randomly” across RegionServer
- Moved around for load balancing and failover
- Split automatically or manually to scale with growing data
- Capacity is solely a factor of cluster nodes vs. Regions per node

And this layout is there and it is automatically done by that is called, ‘Auto Sharding’ and ‘Distribution’. So, unit of the scalability in HBase is the region that we have seen which is managed by the region

servers and they are sorted and contiguous range of rows spread randomly across the RegionServers, moved around for the load balancing and failover that we have already seen in the previous slide. So, is split automatically or manually to scale with the growing data and capacity is only a factor of cluster nodes versus the regions per node.

Refer Slide Time :(17: 46)

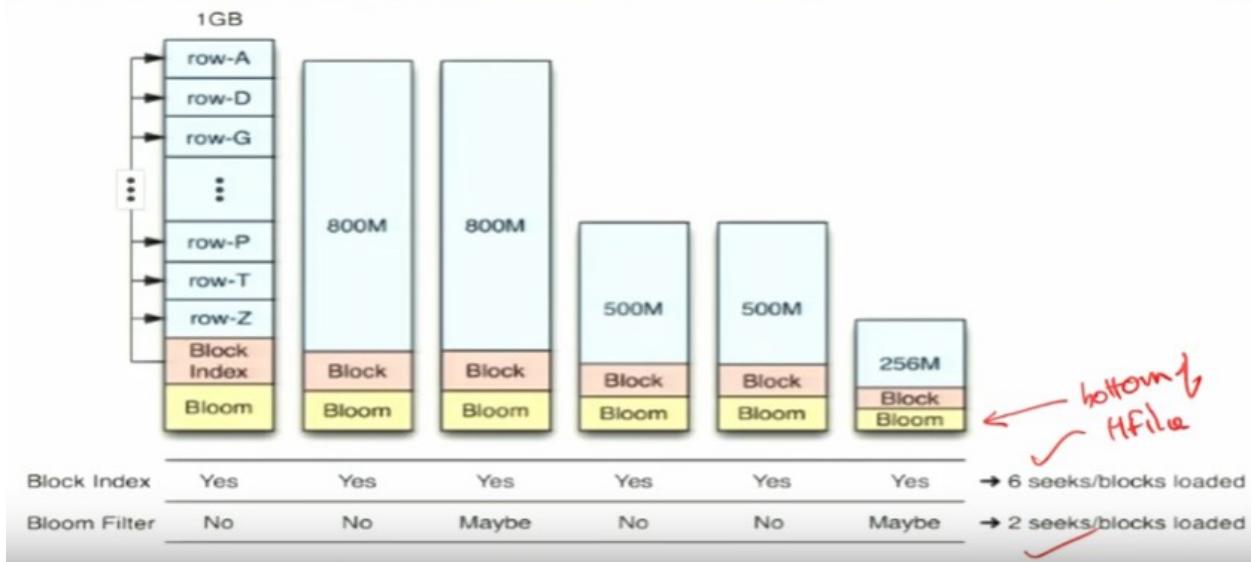
Bloom Filter

- Bloom Filters are generated when HFile is persisted
 - Stored at the end of each HFile
 - Loaded into memory
- Allows check on row + column level
- Can filter entire store files from reads
 - Useful when data is grouped
- Also useful when many misses are expected during reads (non existing keys)

Now, there is a use of bloom filter here in HBase also. So, bloom filters are generated when HBase when HFile is persisted, stored at the end of HFile, loaded into the memory. This bloom filter will allow to check on the rows and from the column level and they can filter the entire store files from the reads, useful when the data is grouped and also useful when many misses are expected during the reads.

Refer Slide Time :(18: 15)

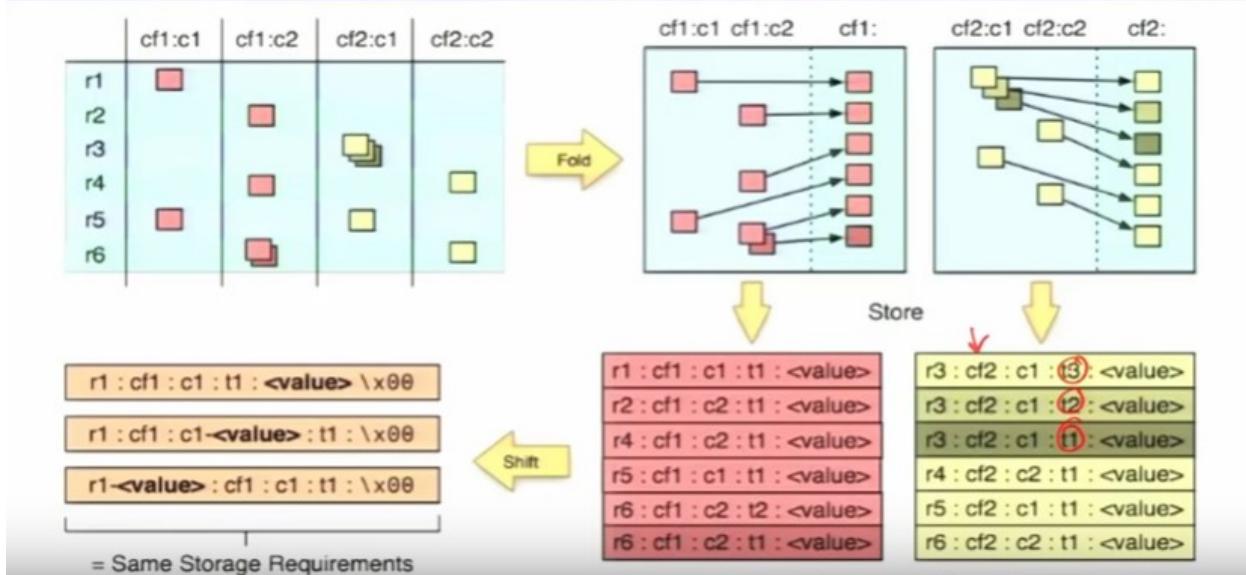
Bloom Filter



So, let us see the positioning of bloom filter into this HFile. So, at the bottom of HFile this particular the bloom filter information is stored and this will help in identifying which of these different blocks are basically containing the block index or information. So, using bloom filter you can access at least two at most two blocks not the entire set of block sequentially has to be read. So, hence the access is reduced from 6 seeks per block to the 2 seeks per block using the bloom filter and bloom filter stone we is restored as a part of s five.

Refer Slide Time :(19: 05)

Fold, Store, and Shift



Now, as far as fold store and shift is concerned you can see here that, for a particular row, the same data is stored, multiple instances because the time series data is also stored at different time stamp the data is stored for a particular row key. So, that is why? Several data on a particular row key is now appearing they may be varying at their timestamp T 1, T 2 and T 3 and they are a store, into the, into the five.

Refer Slide Time :(19: 44)

Fold, Store, and Shift

- **Logical layout does not match physical one**
- **All values are stored with the full coordinates, including: Row Key, Column Family, Column Qualifier, and Timestamp**
- **Folds columns into “row per column”**
- **NULLs are cost free as nothing is stored**
- **Versions are multiple “rows” in folded table**

Logical layout does not match the physical one and here all the values are stored with the full coordinates including the row key, column family, column qualifier and the timestamp. Folds column into the “row per column” and nulls are cost free as nothing is stored, versions are multiple rows in a folded table.

Refer Slide Time :(20: 06)

Conclusion

- Traditional Databases (RDBMSs) work with strong consistency, and offer ACID
- Modern workloads don't need such strong guarantees, but do need fast response times (availability)
- Unfortunately, CAP theorem
- **Key-value/NoSQL systems offer BASE**
 - Eventual consistency, and a variety of other consistency models striving towards strong consistency
- **In this lecture, we have discussed:**
 - HBase Architecture, HBase Components, Data model, HBase Storage Hierarchy, Cross-Datacenter Replication, Auto Sharding and Distribution, Bloom Filter and Fold, Store, and Shift

Conclusion: Traditional databases (RDBMSs) works with strong consistency and offers acid property and in the modern workload doesn't need such strong guarantees, but do need fast response time that is the availability. Unfortunately, CAP provides three out of two that is here given the partition tolerance, the HBase prefers the consistency over the availability. So, key-value pair or a NoSQL system offers the BASE property in these scenarios. So, Cassandra offers the eventual consistency and the other variety of consistency models are striving towards the strong consistency. So, in this lecture we have covered about the HBase architecture components, data model, storage hierarchy, cross datacenter replication auto-starting, distribution and bloom filter and fold store and shift operation. Thank you.

Lecture-19

Spark Streaming and Sliding Window Analytics (Part-I)

Spark Streaming and Sliding Window Analytics.

Refer slide time: (0:17)

Preface

Content of this Lecture:

- In this lecture, we will discuss Real-time big data processing with Spark Streaming and Sliding Window Analytics.
- We will also discuss a case study based on Twitter Sentiment Analysis with using Streaming.

Preface

Content of this lecture: In this lecture we will discuss real time big data processing with Spark Streaming and sliding window analytics. We will also discuss a case study based on Twitter sentiment analysis using Streaming.

Refer slide time: (0:35)

Big Streaming Data Processing

Fraud detection in bank transactions



Anomalies in sensor data

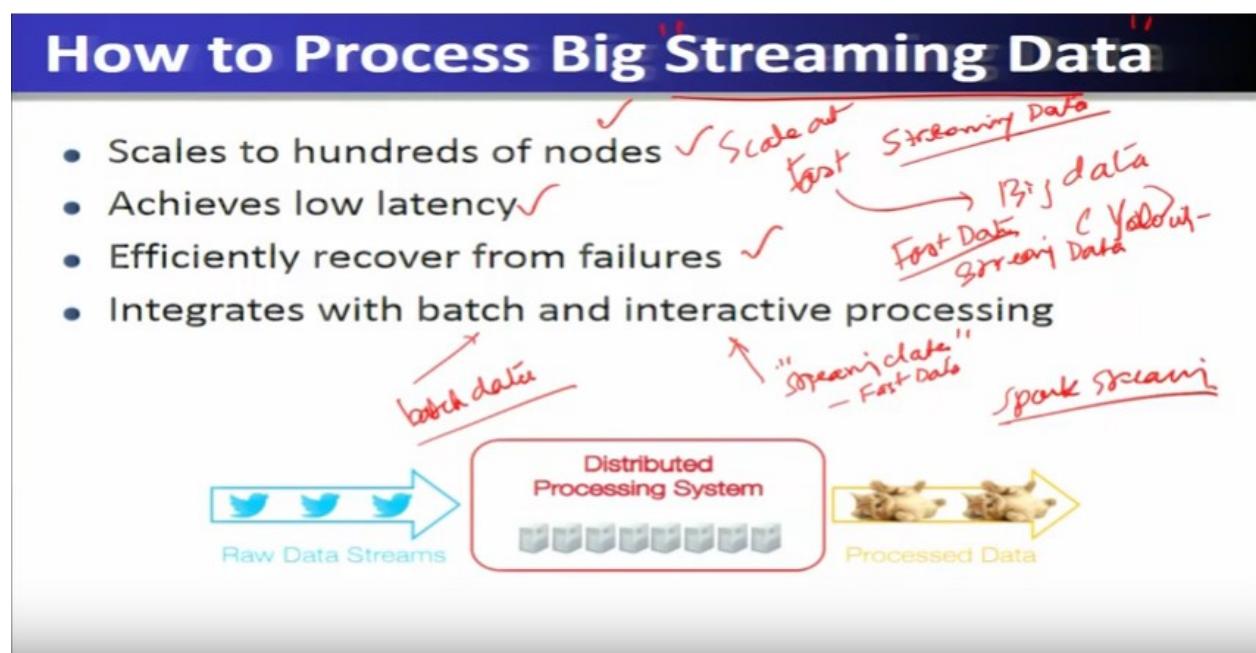


Cat videos in tweets



Big Streaming data processing the motivation of going for Streaming data processing is deep-rooted and is based on the motivation is based on the application such as fraud detection in different banking transactions which are happening online and in real time how this particular fraud detection can be done? That becomes a challenge. So, therefore this motivates Streaming data processing system and how using that? We can do this online or in a real time the fraud detection in the banking transactions. Another such application which has motivated the use of Streaming data processing is detecting the anomalies in the sensor data. Now, you have seen that lot of IOT deployments are happening these days which basically collects the sensor data and then based on these sensing data the anomalies are to be detected in real time. This also has given the new way of solving these problems using the Streaming data processing in real time. Similarly we want to find out we want to do the analysis of tweets and based on these analysis we can find out the sentiments or the mood of the people at a particular instance so this online in a real-time this tweet analysis also requires the Streaming data processing. So, these applications are newer applications are now given a reason to think of providing Streaming data processing for analysis for analytics in the real time. And therefore, we are going to understand the new concepts and the very new theory that is Spark Streaming how that is being used in today's workload and then ever applications.

Refer slide time: (3:03)



Now, the question is how to process the big data big Streaming data? So, now here we are going to deal in this part of the lecture the data which is called a, 'Streaming Data'. So, if the stream of the data is very fast and also a continuous stream then it is categorized under the Big Data scenario. Where in this particular characteristic which is called a, 'Volume', sorry, 'Velocity', requires the infrastructure of a big data to handle it so that means we require hundreds to the thousands of these nodes to scale that means to scale out to deal with this first data which is also called as the, 'Streaming Data'. So, for the scaling and scalability and for we require thousands of hundreds of nodes to process this big Streaming data another

aspect is about achieving the low latency. Obviously this requires the insight into the technology we will see how we can achieve the low latency in this Streaming data processing framework. Now, another requirement is about how to deal with the failures? And how to deal not only with the failures? But, how to efficiently recover from the failures? So, that it can be tolerated by the applications which are monitoring in the real time the events and for different applications. So, we have to deal with the failure recovery which is to be done. So, that it can be catered to the application in a real-time applications which are based on streaming data. Now, another thing is that now there are various interactive processing happening with the Streaming data. And this particular Streaming data is also called as a ‘Fast Data’. Now, besides this fast data sometimes you also require to be integrated with another stream of batch data. So, if there are two different modes of data that means one is through the batch data the other is called ‘Streaming data’ together there are some applications which require the integration of both these different type of data and they are different and they are having a different stack to be processed. So, how are we going to integrate them in the traditional in the previous technologies they are having a different stacks and therefore the integration is basically time taking and may not be useful for real-time applications. So, here we are going to see the new technology which is called the ‘Spark Streaming’. Which will combine or integrate this requirement of processing simultaneously the batch and the interactive data.

Refer slide time: (7:16)

What people have been doing?

- Build two stacks – one for batch, one for streaming
 - Often both process same data
- Existing frameworks cannot do both
 - Either, stream processing of 100s of MB/s with low latency
 - Or, batch processing of TBs of data with high latency

Now, let us see how what people have been doing? So, far that means what are the other systems before this Spark is Streaming. And how it was being done. So, actually as we see that for batch processing and for stream processing there is a requirement of different stacks and often different stacks will be optimized for different type of data so integration is not very common in the previous generation of systems. And, for example for batch processing we have the Spark system and we have the Map Reduce framework for doing the batch processing and for the Streaming in the earlier systems were such

as storm. So, now integration requires two different stacks to be combined together. So, hence it requires it has lot of latency involved within it and may not be require may not be sufficient for different real-time applications. That, is why this particular framework which we are going to discuss in today's lecture that is called, 'Spark Streaming' which will integrate both batch and Streaming applications using the same stack. Therefore, it will be the most efficient way of dealing with multiple type of data that is the batch data and the stream data when they are required to be processed at the same time. So, the existing framework such as storm and Map Reduce cannot do both of the the processing that is for the batch and the Streaming data at the same point of time. So, either the swimming stream processing of hundreds of megabytes with a low latency or the batch processing of terabytes of data with high latency are to be dealt separately and the combined or integrated viewpoint is not available as of date before this Spark is Streaming. So, Spark Streaming is the new technology which we are going to discuss and we will be also seeing different use cases where this kind of batch processing and stream processing together are required in many applications.

Refer slide time: (10:22)

What people have been doing?

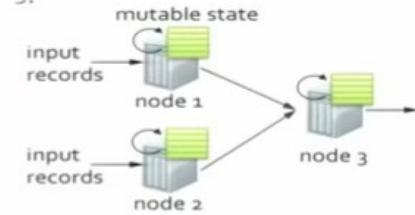
- Extremely painful to maintain two different stacks
 - Different programming models
 - Doubles implementation effort
 - Doubles operational effort

So, therefore there are it is required to maintain two different stacks if the the integration is not supported and also they may require different programming models and also different efforts are required and also requires an operational cost. So, that is not feasible.

Refer slide time: (10:45)

Fault-tolerant Stream Processing

- Traditional processing model
 - Pipeline of nodes ✓
 - Each node maintains mutable state ✓
 - Each input record updates the state and new records are sent out ✓
- Mutable state is lost if node fails ✓
- Making stateful stream processing fault-tolerant is challenging! ✓



— failure of nodes is norm rather exception in commodity hardware

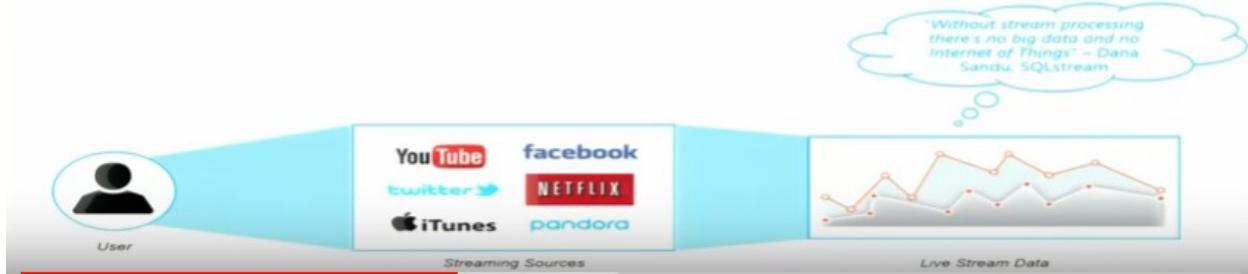
Spark Streaming →

Therefore, the Spark is Streaming we are going to discuss today how that is all integrated and is being useful for various applications. Now, let us see the before going in more detail about the Spark Streaming. So, how we will see let us see that how the fault-tolerance is achieved in the stream processing systems. So, traditional processing models use the pipeline of different nodes and each node maintains the mutable States and each input record updates the state and new records are sent out. Now, the problem with the previous systems were that the mutable States were lost if the node are filled. And, this is a normal failure of the nodes is a norm rather than exception in the commodity hardware. So, therefore when the node fields the mutable states which are maintaining this fault tolerance of for the mutable states will be lost. So, some things are basically lost which are in the traditional the previous systems. Therefore, the stream therefore making the stateful stream processing fault tolerant is also very much needed and in in Spark Streaming system. Will, we will see that how this stateful in stream processing is done and in a fault tolerant manner.

Refer slide time: (12:44)

What is Streaming?

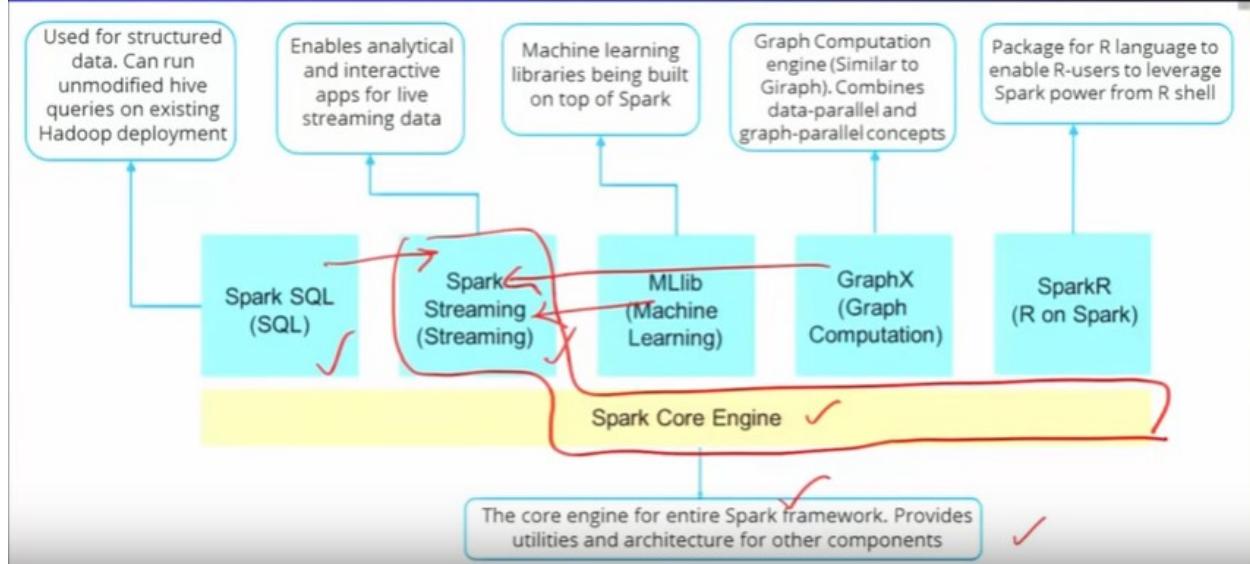
- Data Streaming is a technique for transferring data so that it can be processed as a steady and continuous stream.
- Streaming technologies are becoming increasingly important with the growth of the Internet.



Now, let us see what is a Streaming? So, Data Streaming is a technique for transferring data. So, that it can be processed as a steady and continuous stream which is incoming to the system in a stream of data which is incoming to the system. So, you can visualize as if the data is flowing continuously on the pipes and as it process as it passes through these pipes it has required to be processing in a real-time. And, if that is cause if that is there then it is called the, 'Data Streaming' or 'Streaming Data'. Now, the sources which generates the streaming data are many for example the internet traffic which is flowing also if it is seen then it will also be a network Streaming data similarly the Twitter Streaming data also can be taken up in some of the applications. Similarly, the Netflix which is in real-time online movie watching that also generates the streaming data similarly YouTube data also a kind of the Streaming data and there are many other many many other ways this Streaming data can be generated Streaming data can also be generated from the database so data is read and is being transmitted in the form of the Streaming heater, that is ETL. So, companies normally does this for the analysis so Streaming data technologies are becoming increasingly important I with the growth of the Internet and the Internet enabled different services which are available in the form of Netflix, Facebook, Twitter, YouTube, Pandora, iTunes and so on. There are tons of such different nowadays services available through the internet.

Refer slide time: (14:49)

Spark Ecosystem

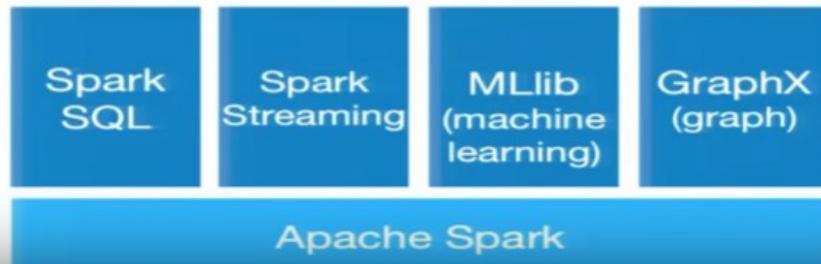


Now, let us see the Spark ecosystem. And, we will see the positioning of the Spark streaming system where it lies. So, from in this Spark ecosystem you will see that on the core there is a spark engine and on top of it is you can see that spark streaming system is running. Now, the spark is streaming system runs over the Spark core and this enables the analytical and interactive applications for the live Streaming data. So, therefore these core components of the Spark framework provide the utilities and architecture for the other components also. Therefore, dealing with this Spark Streaming or Spark gives many other advantages. For example, the analytics which is required to be performed on the Streaming data may sometimes need the machine learning on that to be applied on the own Streaming data. Now, with one single common stack that is a Spark core engine it is now possible that we can apply or it is possible that the machine learning libraries which are built on top of the Spark can also be used for analytics of the Streaming data. Similarly, the graph processing applications such as graph computation engines which combine data parallel and graph parallel concepts can also be useful for the analysis of our analytics of the Streaming data. And, they are all integrated similarly the Spark SQL which is also the structured way of accessing the key value store can also be useful to be used in the Spark Streaming for storage and retrieval purposes. So, therefore having the same in stack this Spark Streaming will gain a lot of advantages not only for doing the batch processing and the Streaming data together there can be integrated and a lot of other that means libraries can be used such as machine learning can be applied for the Streaming analysis and graph also can be used as for the Streaming data analysis. Which can be represented in a graph and can be done an analysis. We, will see this Spark ecosystem. So, the Spark Streaming the positioning on top of the Spark core will gain not the advantages of other such utilities like machine learning that is MLlib, GraphX, Spark SQL and so on. We, will see in further slides how we are going to utilize the integration of all these together in solving today's workload problems.

Refer slide time: (17:54)

What is Spark Streaming?

- Extends Spark for doing big data stream processing
- Project started in early 2012, alpha released in Spring 2017 with Spark 0.7
- Moving out of alpha in Spark 0.9
- Spark Streaming has support built-in to consume from Kafka, Flume, Twitter, ZeroMQ, Kinesis, and TCP/IP sockets.
- In Spark 2.x, a separate technology based on Datasets, called Structured Streaming, that has a higher-level interface is also provided to support streaming.



So, what is this Spark Streaming? So, is it Spark it extends the Spark for doing the data big data Streaming processing. So, big data stream processing can be now done with the help of Spark Streaming which is an extension which extends the Spark core to perform this Spark Streaming. Now, this Spark Streaming project was started in 2012 and it was released in the form of this Spark 0.9. And, Spark Streaming highest lot of built-in support to consume the data from different sources such as from Kafka is also one of the receiver which can feed the live stream data to the Spark stream similarly flume, Twitter, Zero MQ, kinesis and TCP/IP sockets. Are, some of the inbuilt receiver system through API is they can be plugged into the Spark Streaming system. Now, to get the stream data for computations in this scenario. So, in a spark to point acts a separate technology based on the data set called, ‘Structured Streaming’, has been designed and that has a higher level interface provided to provide the the support of Streaming.

Refer slide time: (19:26)

What is Spark Streaming?

- Framework for large scale stream processing
- Scales to 100s of nodes ✓
- Can achieve "seconds" scale latencies ✓ *Seconds not in min*
- Integrates with Spark's batch and interactive processing
- Provides a simple batch-like API for implementing complex algorithm ✓
- Can absorb live data streams from Kafka, Flume, ZeroMQ, etc. ✓

So, let us see the Streaming Spark Streaming framework for a large scale processing of Streaming data. And, we will again let us review that it should be scale to the hundreds and thousands of the node it can achieve can achieve second scaled the latencies. So, that means latencies are to be in the form of seconds. So, latencies are in the in the scale of seconds not in the minutes are not latency should be in the minutes or hours and so on. Now, achieving this is the scale of seconds latency is not going to be easy task and it requires the new design. Therefore, we are going to understand the Spark Streaming system. Now, this is Spark Streaming system also integrates the batch and the interactive processing together with a unified view with the same stack therefore with unification on this batch and interactive application it is now possible to achieve the latencies in the in the scale of seconds. Similarly it will provide a simple batch like API is for implementing the complex algorithms. And, it can also integrate it with the live data streams from different other live other tools such as Kafka, flume, ZeroMQ and so on. So, that is why this Spark Streaming is now-a-days required very much.

Refer slide time: (21:22)

What is Spark Streaming?

- Receive data streams from input sources, process them in a cluster, push out to databases/ dashboards
- Scalable, fault-tolerant, "second-scale latencies"



Now, Sparky Streaming receives the data stream from different input sources process them in the cluster and push it out to the databases dashboard for its output. Therefore, Spark Streaming is a scalable fault tolerant and having the the latencies of the scale of the seconds time. So, let us see through this particular diagram that the input the data can be received through different input sources. So, it can be received either from Kafka, flume, HDFS, kinesis or Twitter this live stream of data is now fed into the spark Streaming system for the computation of Streaming data computation. And, after that the output will be either stored on the database or it will be pushed on the dashboard for the output. So, we will now see in this part of the discussion. What is the spark Streaming? How it is handling this kind of Streaming of Streaming data of different sources together? And for which is useful for various applications?

Refer slide time: (22:54)

Why Spark Streaming ?

- Many big-data applications need to process large data streams in realtime



So, therefore again are going to understand about why the Spark is Streaming? So, many big data applications need to process the large data streams in the real time such as website monitoring sometimes require is required to see the the loads, the hits which are basically coming on the website and whether the performance of the website is going well or not for has to be monitored. Similarly, the online transactions for the bank or from for the credit card also really needs to be analyzed in real time. So, that becomes Streaming data applications. Similarly, various ad monetization also require to be processed in real time. So, whenever a user clicks on a particular ad and so those ads are to be so when the user will click. So, all that data the click-through data has to be analyzed in real time and therefore different ads are to be pushed in that manner.

Refer slide time: (24:13)

Why Spark Streaming ?

- Many important applications must process large streams of live data and provide results in near-real-time *(in seconds)*
 - Social network trends
 - Website statistics
 - Intrusion detection systems
 - etc.
- Require large clusters to handle workloads
- Require latencies of few seconds



So, the Spark Streaming is very much needed and many important application must process the large stream of live data and provides the results in a near real-time near real-time means this particular system which we are going to discuss that is Spark Streaming has the latency is up to the half of the second. So, those latencies which are tolerable up to half of the seconds can be used here by with the help of Spark Streaming even lesser than this can be used in a separate Streaming systems which are known as the storm and so on which has very less latency already available. So, let us see that if the the different application such as social network trends where statistics intrusion detection system and so on. They they need this kind of streaming system and require the large cluster to handle these workloads and also require the the latencies of the order seconds.

Refer slide time: (25:29)

Why Spark Streaming ?

- We can use Spark Streaming to stream real-time data from various sources like Twitter, Stock Market and Geographical Systems and perform powerful analytics to help businesses.



So, we can use the Spark is Streaming to stream the real-time data from various sources like Twitter a stock market geographical system for doing the powerful analytics. So, spark sleeping is used to stream the real-time data from various sources like Twitter, stock market, geographical systems perform powerful analytics to help various business.

Refer slide time: (25:51)

Why Spark Streaming?

Need a framework for big data
stream processing that

Scales to hundreds of nodes

Achieves second-scale latencies

Efficiently recover from failures

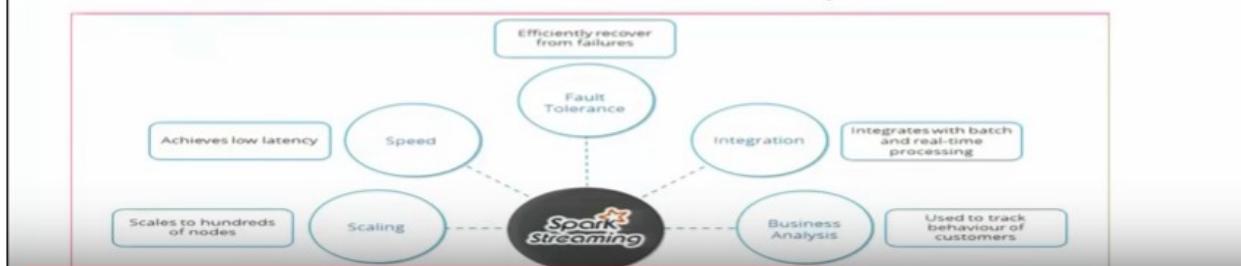
Integrates with batch and interactive processing

So, therefore there is a need of a framework for big data stream processing that scales to the hundreds thousands of the nodes achieves second scale latency sufficient to recover from the failure integrate, with a batch and interactive processing.

Refer slide time: (26:06)

Spark Streaming Features

- **Scaling:** Spark Streaming can easily scale to hundreds of nodes.
- **Speed:** It achieves low latency.
- **Fault Tolerance:** Spark has the ability to efficiently recover from failures.
- **Integration:** Spark integrates with batch and real-time processing.
- **Business Analysis:** Spark Streaming is used to track the behavior of customers which can be used in business analysis



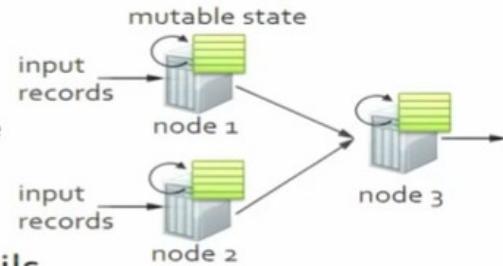
Let us see, how? What are the different features which are handled which are able to cater all these parts? So, spark is tripping features first is scaling. So, Spark Streaming can easily scale to hundreds and thousands so speed also is a low latency and fault tolerance is achieved here to recover from the failure ending it is also integrated with the real-time and Business Analytics is also supported.

Refer slide time: (26:45)

Stateful Stream Processing

- Traditional model

- Processing pipeline of nodes
- Each node maintains mutable state
- Each input record updates the state and new records are sent out



- Mutable state is lost if node fails

- Making stateful stream processing fault tolerant is challenging!

So, let us see another part besides the integration with the batch and the batch processing and the real-time Streaming data processing other than that we will see another requirement which is about stateful stream processing. In, the traditional model as we have seen that it provides a pipeline and if the mutable state is lost then it has to handle.

Refer slide time: (27:15)

Modern Data Applications approach to Insights

Traditional Analytics

Structured & Repeatable
Structure built to store data



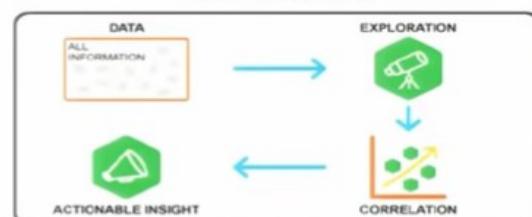
Start with hypothesis
Test against selected data



Analyze after landing...

Next Generation Analytics

Iterative & Exploratory
Data is the structure



Data leads the way
Explore all data, identify correlations



Analyze in motion...

So, let us see the modern data applications approaches for more insights. So, traditional analytics is basically require these kind of analysis.

Refer slide time: (27:32)

Existing Streaming Systems

- Storm ✓
 - Replays record if not processed by a node
 - Processes each record at least once
 - May update mutable state twice!
 - Mutable state can be lost due to failure!
- Trident – Use transactions to update state
 - Processes each record *exactly once*
 - Per-state transaction to external database is slow

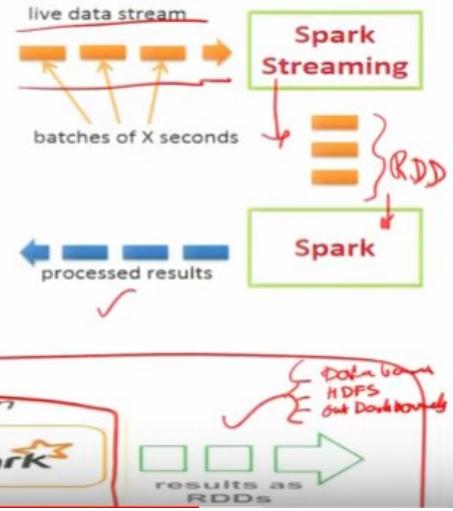
Now, the existing system for the Streaming data analysis is called a, ‘Storm’. And, it replays the record if not processed by the node and therefore sometimes it provides the at least ones semantics. So, that means some of the updates if the mutable states and the nodes are filled to achieve the updates in the mutable state. So, that means it will update twice so that becomes a problem in at least one semantics and the mutable states can be lost due to the failures. So, this at least once semantics and will create problems in some of the times where the updates are to be done twice and in the existing systems like storm and does achieve only up to that state of the art which is called at least once. So, exactly once is the semantics which is required and it is supported in the spark Streaming system. There, are other streaming system such as Trident which use the transactions to update the state and there also has exactly ones semantics and for estate transaction to external database is slow.

Refer slide time: (28:58)

How does Spark Streaming work?

Run a streaming computation as a **series of very small, deterministic batch jobs**

- Chop up the live stream into batches of X seconds
- Spark treats each batch of data as RDDs and processes them using RDD operations
- Finally, the processed results of the RDD operations are returned in batches



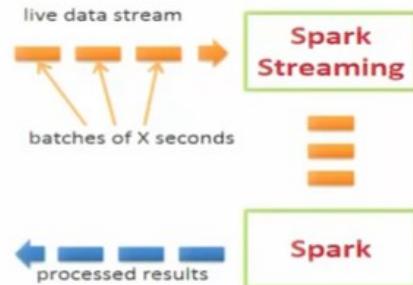
Now, how does this Spark Streaming work? Let us understand this. So, it runs a Streaming computations as the series of small, deterministic bad jobs and the live streams which are taken up or which are considered in the system is to be divided into the X seconds. And, the Spark treats each batch of data is an RDD and process them using an RDD operation. Finally, the process results of RDD operations are returned in the batches. So, let us see all these things using the diagram so here the live stream data when it is entered it will be divided into the batch of of X seconds. And, this particular Spark Streaming system will now divide into the batches earned this batches in the form of RDD is will be given to the Spark engine for processing and the process result will finally be emitted. So, again let us discuss this entire scenario which says that you to run this Spark Streaming system requires the the Streaming data to be entered into the system that is called data streams. Data streams are received by the receivers of the Spark Streaming system and then it will divide into the batches of X seconds and they are called, 'Micro Batches' or a 'Micro RDDs'. So, after dividing into the batches of X seconds this particular micro batches will be given to the Spark. And, spark reads each batch of data as RDD and process them using the different RDD operations which are provided by the Spark engine. And, finally the process result of the RDD operations are returned in the form of batches and will be taken care either they are to be stored in the database or they are stored in HDFS or they are output on dashboards. So, there are different ways, this output can be or the result of this Spark Streaming can be now used indifferent applications.

Refer slide time: (31:47)

How does Spark Streaming work?

Run a streaming computation as a **series of very small, deterministic batch jobs**

- Batch sizes as low as $\frac{1}{2}$ second, latency of about 1 second
- Potential for combining batch processing and streaming processing in the same system



So, therefore we have seen that to run a Spark Streaming computation as a series of very small deterministic batch jobs. And, these batch sizes are as low as half of the second. So, it cannot be lesser than half a second latency why because this is to be set by the system. So, let us see if let us say batch size is a half of a second it will finally end-to-end latency it comes out to be only one second. So, therefore seconds of latency is achieved due to this batch size of a half a second size is being computed and processed. So, therefore the potential for combining batch and Streaming data processing in the same system exists why because finally these batches are also micro batch RDD's and which are given to the spark system and the batch data also is processed to the SPARC system so finally they have the single engine that is the spark engine which process both the batch as well as the Streaming data of after the processing of the Spark is Streaming. So, therefore it is possible to combine or integrate both batch processing and Streaming processing in the same system.

Refer slide time: (33:12)

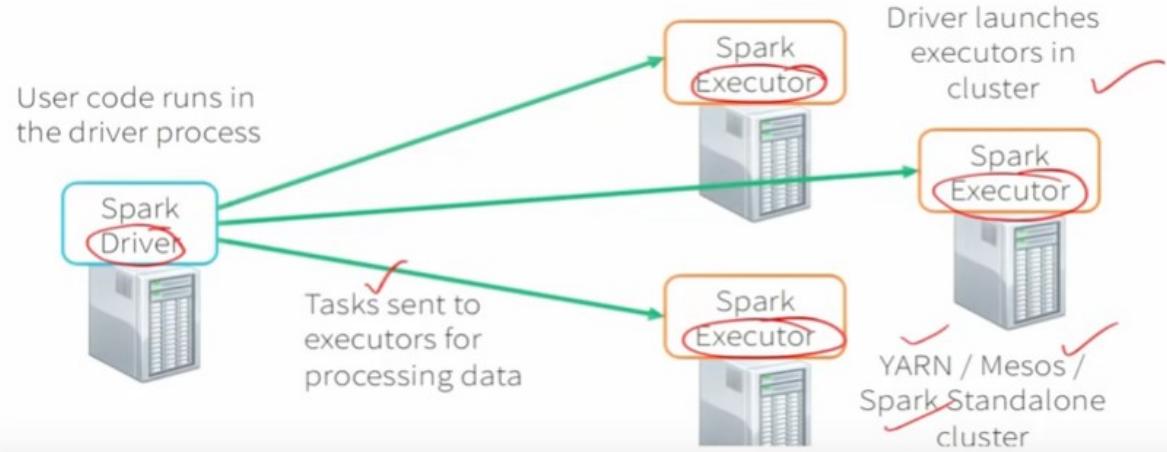
Word Count with Kafka

```
object WordCount {
    def main(args: Array[String]) {
        val context = new StreamingContext(new SparkConf(), Seconds(1))
        val lines = KafkaUtils.createStream(context, ...)
        val words = lines.flatMap(_.split(" "))
        val wordCounts = words.map(x => (x,1)).reduceByKey(_ + _)
        wordCounts.print()
        context.start()
        context.awaitTermination()
    }
}
```

And, now let us see an example of a word count with Kafka as the live input source. And, how this is all done in the Spark Streaming system. So, this word count application we can see we have to create the Spark Streaming context and with a one-second size of batch sizes and the Sparkle and then this particular context will be bound to the to the Kafka utilities and which will give the live stream in the form of lines and this particular line will be further processed by the by the Spark Streaming using flat map. Which, will split this line into the words and this word will now be applied on a map function. So, map function is doing the transformation on all the words, which will be received by the Kafka live stream. And, this map function will now perform the transformation that for every word it will emit the word and a value 1 which in turn will be taken reduced by key and for every word it will now try to count or it will or it will now do the sum of all such words appearing number of ones that means and therefore every word will now have a particular count which will be printed. And, then we have to do this context start and finally the context of wait termination also these. So, you see that what count with the Kafka is quite easy to understand and you can write down the program address of the things that is the entire operation of this stream computation is completely abstracted from the user.

Refer slide time: (35:12)

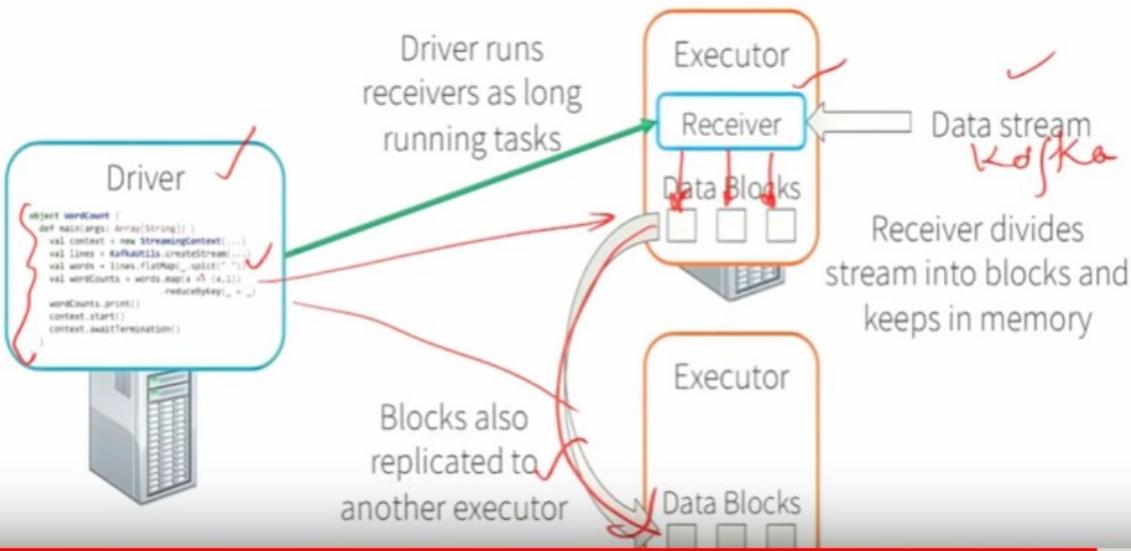
Any Spark Application



Now, let us see the internal details of the calf the Spark stream execution before that let us understand first the Spark application. So, in the Spark application that is a Spark engine will have the processing in the terms of the driver program which will now communicate with the executors running on different nodes on to the cluster systems. And, this particular driver will communicate to the executor and this particular driver will assign the tasks send to the executor for processing this particular data. Now, this particular driver launches the executors in this particular cluster and this can be done with the help of yarn and Mesos or Spark standalone clusters. So, this way the Spark launches using the driver program and driver and executors driver will assign the task to the executors and executors in turn will perform the execution of the processing of data in the Spark system.

Refer slide time: (36:25)

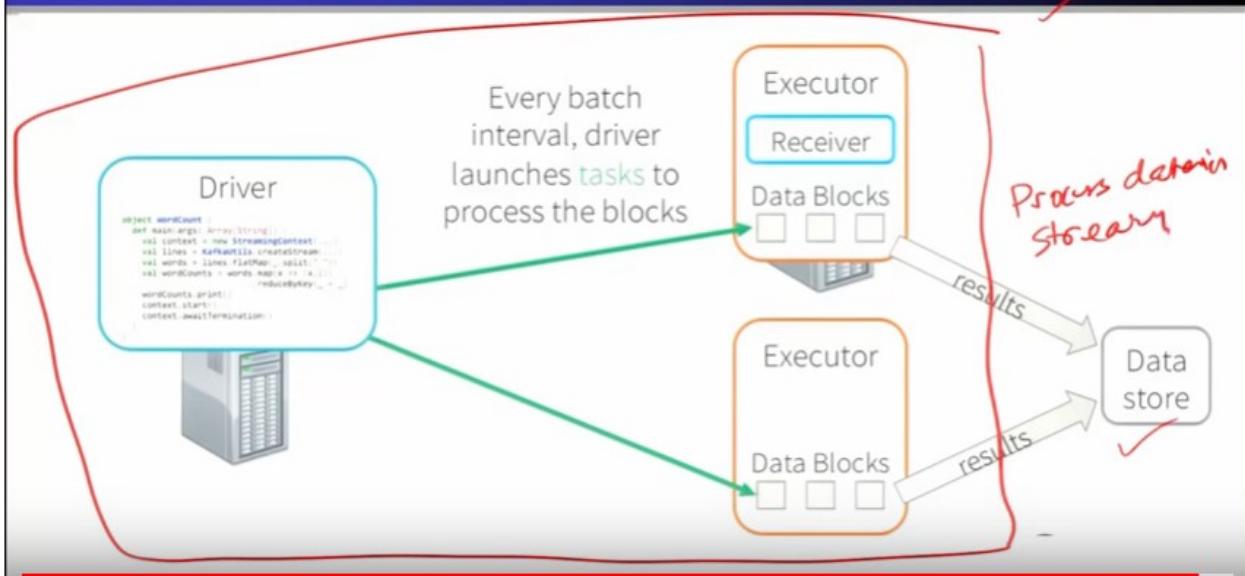
Spark Streaming Application: Receive data



Now, in Spark Streaming runs on top of a Spark engine. So, let us see how the Spark Streaming application will now progress in this scenario again there will be a driver program and the same word count program which we have written is given to the driver and Driver will now in turn runs the receiver as long as as long as running the task. So, this particular tasks are given to the executor and executors in turn will have the receiver which will receive the live data stream from Kafka. So, after receiving the data from the Kafka now the data will be means this particular live stream data will be received in the form of lines and these lines will be now done in and that will be in the form of data blocks. Now, then it will assign the task of a flat map on these executors these data blocks are to be replicated on to the other executors. So, replication also is required for achieving the fault tolerance of these data blocks. Now, further the driver will also assign the tasks to be performed by the executors on these data blocks. And, for example we have seen the flat map and performing the the Map Reduce operation on the word counts all these tasks will be executed at the executor. So, so all these are the components of the spark Streaming system. So, here we see that how it will receive the data using the receiver.

Refer slide time: (38:12)

Spark Streaming Application: Process data

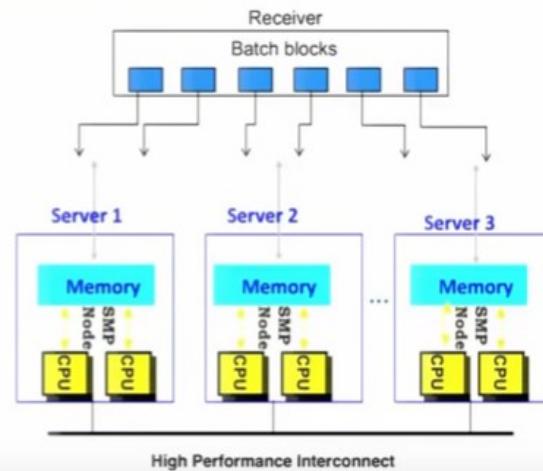


And, then we have also seen now see that how the data will be processed so each batch interval the driver launches the task to process these particular data blocks. And, these processing will now give other results to the results back and that result will be stored in the in the data store. So, these parts we have seen and now we have also seen internally how the data is to be processed in the spark in the Streaming in the spark Streaming system.

Refer slide time: (38:49)

Spark Streaming Architecture

- Micro batch architecture.
- Operates on interval of time
- New batches are created at regular time intervals.
- Divides received time batch into blocks for parallelism
- Each batch is a graph that translates into multiple jobs
- Has the ability to create larger size batch window as it processes over time.

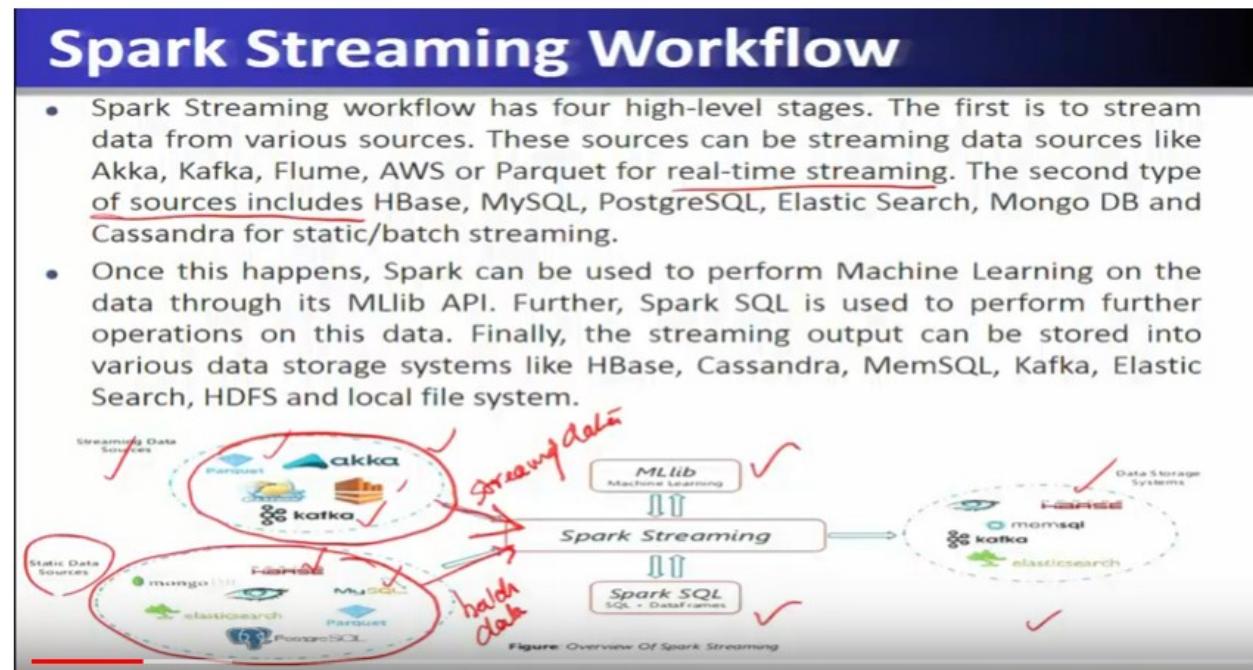


So, let us review the entire scenario as the spark is streaming architecture. So, it is based on micro batch architecture operates in the interval of time whenever a new batch are created at a regular time interval it divides the received time batch into the blocks for parallelism. And, each batch is draft that translates into the multiple jobs. And, has the ability to create the larger sized batch window as it processes over the time.

Lecture - 20
Spark Streaming and Sliding Window Analytics (Part-II)

Spark Streaming Workflow.

Refer slide time :(0:15)



So, spark streaming workflow has four high level stages and let us sees all these stages in in a brief manner. So, the first is to stream the data from various sources and these sources are such as akka system, Kafka system, flume, AWS or Parquet for the real-time streaming data input. So, this particular streaming data is to be input using these different sources akka, Kafka, flume and Parquet, they will give the live feed of streaming data to the to the system. Now, second type of sources of the data include they are called the static data sources. So, they include HBase that we have discussed, MySQL, PostgreSQL, then Mongo DB, Cassandra and they are for the static are the batch streaming data feed into the SPARK streaming system. Once, these once these data is streamed into the spark system the spark and we used to perform on on it the machine learning using the machine learning API that is MLlib further the spark SQL can also be used to perform further operations on this particular data finally the streaming output can be stored into the into the various data storage system like HBase and Cassandra, MemSQL, Kafka, elastic search, HDFS, local system and so on.

Refer slide time :(02:40)

Spark Streaming Workflow

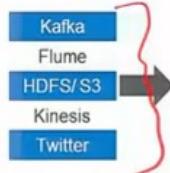


Figure: Data from a variety of sources to various storage systems

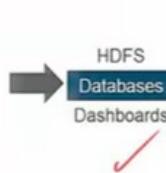


Figure: Incoming streams of data divided into batches

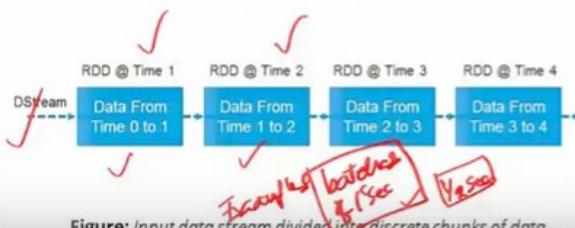


Figure: Input data stream divided into discrete chunks of data

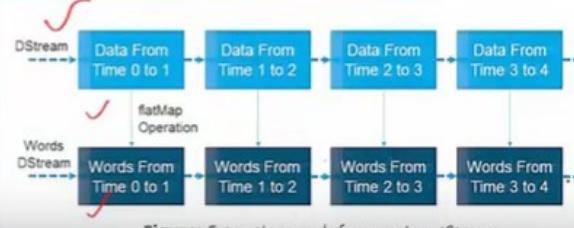


Figure: Extracting words from an InputStream

So, therefore the the spark is streaming workflow looks like this that input of the streaming data comes from Kafka, flume, HDFS, Kinesis, Twitter. And, finally after processing they will be stored on to the either HDFS databases dashboard and so on. So, incoming these incoming data streams are now divided into the into the batches of input data which is given back to the spark engine for computation and they will process it and gives back output this particular data. Now, this batch the input data streams is now divided into the discrete chunks of data for example the streams which is handled by the spark streaming is called the, ‘ Discretized Stream’, or a dstream. So, dstream is is the batches of data of X seconds. So, here the batch is of let us say one second so from zero to one second batch is called as ‘RDD’ at the rate time one. So, the batch of timing from time one to time two that is that next one second is called the ‘ RDD’ at the rate time 2 and so on. So, the batches are divided into the into the discrete chunks in this example this is the batches of one second it is divided into the batches of one second in this example it can be minimum half a second batches for better latency from n to n. So, once this particular data stream or dstream is decided or is broken by the Spark streaming system then various transformation can be applied which is also a part of the Spark streaming system for example a flat map operation can be applied on every dstream. So, for example when a flat map operation is applied it will give different words which are input or which are divided in the form of dstream of one second duration similarly this flat map is when applied on all the case then it will extract the words from the input stream.

Refer slide time :(05:19)

Example 1 – Get hashtags from Twitter

```
val tweets = ssc.twitterStream()  
val hashTags = tweets.flatMap(status => getTags(status))
```

new DStream

transformation: modify data in one DStream to create another DStream

tweets DStream

hashTags Dstream
[#cat, #dog, ...]

batch @ t batch @ t+1 batch @ t+2 →

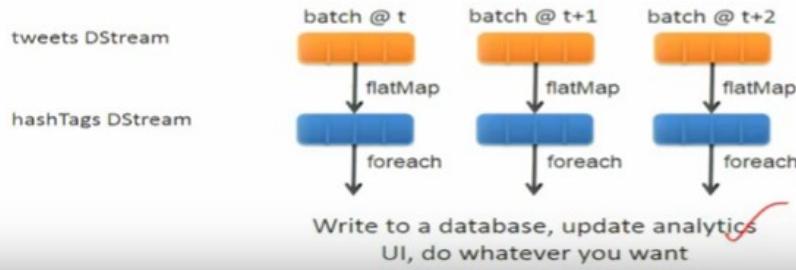


And, this can be shown here in this example of getting the hashtags from on the Twitter. So, Twitter stream is input into the into the system into the spark streaming system. So, this is called. So, Twitter stream is given into the system and then after dividing into the different Dstreams it will perform the transformation on top of it which is called the 'Flat Map'. And, the flat map will be defined on the tweets which is given into the system and the flat map will perform the get tag the hashtag it will get as the status and this hashtag will be extracted as per the transformation. So, the transformation will modify one D stream to create another dstream in this particular manner and finally it will from this tweet dstream it will extract the hashtags in this particular example. So, this is shown over here that this tweet Dstreams when a flat map is applied it will give the hashtag Dstreams which is hashtag in in the terms of for example number of #cat #dog and so on different topics it will extract and it will generate the new RDD

Refer slide time :(07:03)

Example 1 – Get hashtags from Twitter

```
val tweets = ssc.twitterStream()  
val hashTags = tweets.flatMap(status => getTags(status))  
hashTags.foreach(hashTagRDD => { ... })
```



out of every batch therefore after that these hashtags will be saved into the Hadoop as a file and this will be the output which is to be. So, output operation is to push this particular transform data in the form of a hashtag to the external storage. And, here that is shown over here but not every time we are going to store we can perform various analytics on this transform hashtag. And, maybe that sometimes this transformed hashtag analytics will require to update the website or perform various other applications depending upon whatever we want to do on this output so for each. So, therefore various,

Refer slide time :(08:01)

Java Example

Scala

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap(status => getTags(status))
hashTags.saveAsHadoopFiles("hdfs://....")
```

Java

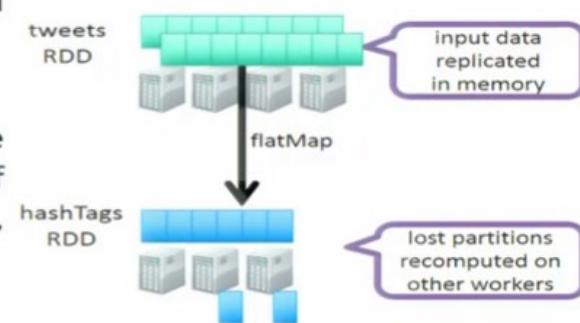
```
JavaDStream<Status> tweets = ssc.twitterStream()
JavaDStream<String> hashTags = tweets.flatMap(new Function<...> { })
hashTags.saveAsHadoopFiles("hdfs://....")
```

different programming languages are supported. So, the example which we have seen was written in scala the same application it can be written using the Java. So, Java API also is available with the Spark streaming system.

Refer slide time :(08:21)

Fault-tolerance

- RDDs remember the sequence of operations that created it from the original fault-tolerant input data
- Batches of input data are replicated in memory of multiple worker nodes, therefore fault-tolerant
- Data lost due to worker failure, can be recomputed from input data



And, now let us see about the fault tolerance. So, RDDs remember the sequence of operation and that created it from the original fault tolerant data. So, therefore RDDs are knowing using lineage about the sequence of operation how they are created from the original. So, this we know from the spark fault tolerance system and when the batches of input data are replicated in the memory of multiple worker nodes and therefore we are trying to achieve the fault tolerance in this case. So, whenever the data is lost to the worker failure it can be recomputed using lineage from the input why because these RDDs remember all that things. So, so therefore this fault tolerance can also be ensured here in the part of this spark system in a spark streaming.

Refer slide time :(09:17)

Key concepts

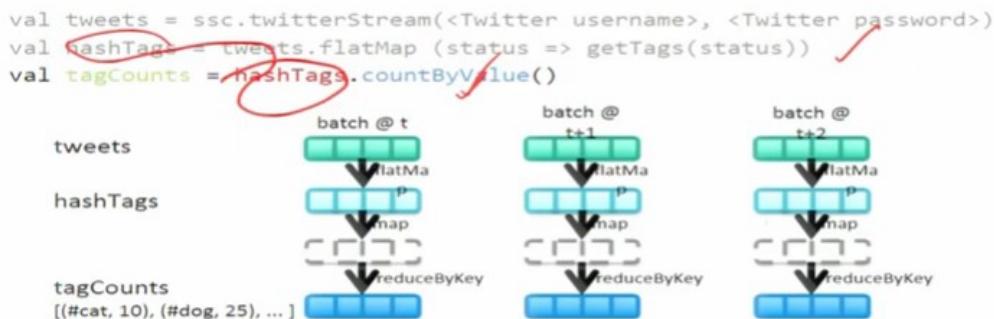
- **DStream** – sequence of RDDs representing a stream of data
 - Twitter, HDFS, Kafka, Flume, ZeroMQ, Akka Actor, TCP sockets
- **Transformations** – modify data from one DStream to another
 - Standard RDD operations – map, countByValue, reduce, join, ... ✓ ✓ ✓ ✓
 - Stateful operations – window, countByValueAndWindow, ... ✓
- **Output Operations** – send data to external entity
 - saveAsHadoopFiles – saves to HDFS ✓
 - foreach – do anything with each batch of results ✓

So, let us see the key concepts. So, key concepts so far we have seen about the Dstream which is a sequence of RDDs representing the stream of data and these Dstreams are created out of the stream of data from Twitter, HDFS, Kafka, flume and so on. And, there are various transformations can be applied on Dstream which can modify one from from the given Dstream to another form of these team. So, the standard RDD transformations which are operations which are available for the transformations are the map, count by value, reduce, join and so on similarly there are other stateful operations which are available in the form of transformations such as window operations. And, count by value and window and so on we will see all these stateful operations. Now, besides the transformation there are actions or the output operations also to be performed on the Dstreams which are available as part of the spark streaming system. Now, these output operation will send the data to the external entities will save as Hadoop files

will say to HDFS for each do anything with each batch of results. So, we will see that whenever an action or an output operation which we are going to perform either it will save or to the HDFS file save as a file or it will further actions, using for each command.

Refer slide time :(10:53)

Example 2 – Count the hashtags

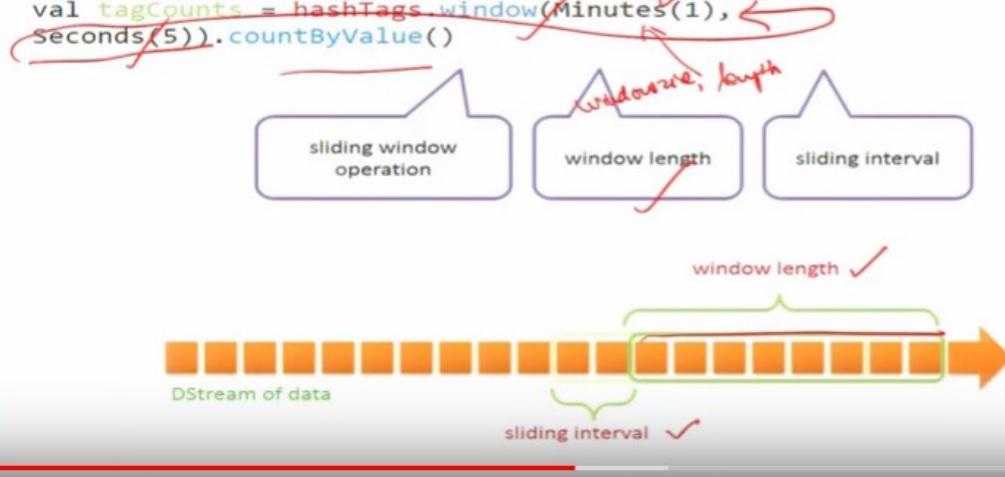


So, again we will now count the hashtags in this particular example. So, that means once we get the hashtags that we are going to count these hashtags. So, these hashtags which is now available using this spark streaming system. Now, we are going to perform an action or the operation as an output. So, the output here is to be the count by value. So, it will count how many hashtags about these hashtags are there into the stream processing into the data. So, here we can see that we perform this count by value.

Refer slide time :(11:42)

Example 3 – Count the hashtags over last 10 mins

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap(status => getTags(status))
val tagCounts = hashTags.window(Minutes(1),
seconds(5)).countByValue()
```

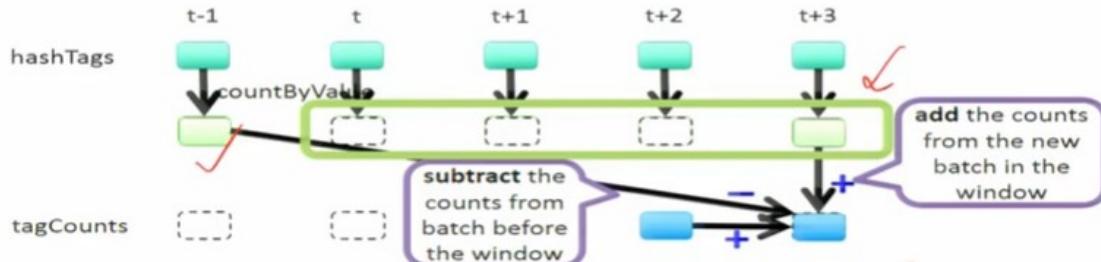


Now, we will also see some more functionality in the terms of this example which will count the hashtag over the last 10 minutes. So, is so basically for that we have to use the been doing operation. So, we have to use the window of one minute and within the one minute after every five seconds we are now monitoring this hashtag and then performing the count by value. So, this particular command this operation will have the windows window has two different arguments one is the length of the window, window size and window length. And, so here it is in the window of one minute duration for every five second we are going to count this by value. So, just window is can be seen here the window has sliding window operation, has the window length and the window interval, sliding interval. So, we can see using this particular diagram that this is the window length this is of one minute and sliding interval is of that another duration. So, sliding interval will give the data for processing so whenever we say count by value. So, this particular window length and sliding interval together will give the data for this computation.

Refer slide time :(13:35)

Smart window-based countByValue

```
val tagCounts = hashtags.countByValueAndWindow(Minutes(10), Seconds(1))
```



So, here we can see that in this particular example that that we are going to count all the data in that particular window. Now, one important thing is that when the window slides can see the sliding interval there are two things one is called ‘Window Length’ and the ‘sliding interval’. So, when the window slides then the this particular old data will be out of that window and a new data will enter into the system. So, the values which we are now counting every time is going to be changed in this way. So, what will be the new count word count count by value that is required to subtract this is the previous value and the new value is to be added this concept requires the window based different algorithm to do the analysis.

Refer slide time :(14:48)

Smart window-based *reduce*

- Technique to incrementally compute count generalizes to many reduce operations
 - Need a function to “inverse reduce” (“subtract” for counting)
- Could have implemented counting as:
`hashTags.reduceByKeyAndWindow(_ + _, _ - _, Minutes(1), ...)`

That, we will see in the further slides. So, smart window based reduction we will see different commands are there so techniques to incrementally compute count generally generalizes to the many reduce operations that needs a function to inverse the reduce that is subtract for counting. I, could have implemented counting as the hashtag reduced by key and window. So, within that particular time that will do this operation that is after sliding the new value will come and the old value will be subtracted. And this is performed in the reduced by key and window operation.

Refer slide time :(15:31)

Arbitrary Stateful Computations

Specify function to generate new state based on previous state and new data

- Example: Maintain per-user mood as state, and update it with their tweets

```
def updateMood(newTweets, lastMood) => newMood  
moods = tweetsByUser.updateStateByKey(updateMood)
```

Arbitrary stateful computation, this is also very important computation in the spark streaming system why because this allows you to maintain a count of particular words or event which is occurring into the stream of data. So, this is to maintain the stateful computation. So, especially by function to generate the new state based on the previous state and a new data. So, for example maintain per user mood as the state and update it when when it sees that tweet. So, update mode definition of a function can be defined as the new tweets and the last mood which it has seen and then it will update to the new mood and so this is an update function so whenever. So, this particular function will do this update mood using the parameters which is given in the in the tweet so so whenever a new tweet it comes it will perform this update move function. And, this will be the update state by the key and so whenever the Twitter tweets by the user is given. So, here the mood will be extracted and and the state variable is maintained to measure the to understand the mood. So, here that is that is why it is called a ‘Stateful Computation’. So, state is man is maintained all the time so and this state will be updated whenever not did the stream of data is coming and now there is a change of mood. So, it will be extracted out of the stream and updated as the state.

Refer slide time :(17:34)

Arbitrary Combinations of Batch and Streaming Computations

Inter-mix RDD and DStream operations!

- Example: Join incoming tweets with a spam HDFS file to filter out bad tweets

```
tweets.transform(tweetsRDD => {
    tweetsRDD.join(spamHDFSFile).filter(_.value)
})
```

So, arbitrary combination of batch and stream computation example we are going to see now. So, there will be an intermix RDDs and the dstream computation operation. So, for example join incoming stream with the with a spam HDFS file to filter out the bad tweets. So, using this particular way of joining we can see that so so the tweet RDDs is now joined with the HDFS file system and we will perform various filter on top of it and this will transform the tweets and the transform to X will be given as the output. So, all these functions are written in the scalar to understand these particular operations or the commands or the programs. So, I advise you to refer the scala programming language to have a better understanding of the streaming.

Refer slide time :(18:36)

Spark Streaming-Dstreams, Batches and RDDs



- These steps repeat for each batch.. Continuously
- Because we are dealing with Streaming data. Spark Streaming has the ability to “remember” the previous RDDs...to some extent.

So, spark streaming these 3 dstreams batches and RDDs. So, again let us summarize all this is that whenever the data input data stream is coming input streaming data is coming. So, in the spark streaming system it will divide into the batches of one second duration in this example and these batches are now performed various transformations. And, an action and given back to the spark engine for giving the output. So, these steps are repeated for each batch continuously because we are dealing with the streaming data. So, the data is continuously coming, in the form of streams. So, spark streaming has the ability to remember the previous RDDs and to some extent.

Refer slide time :(19:26)

DStreams + RDDs = Power

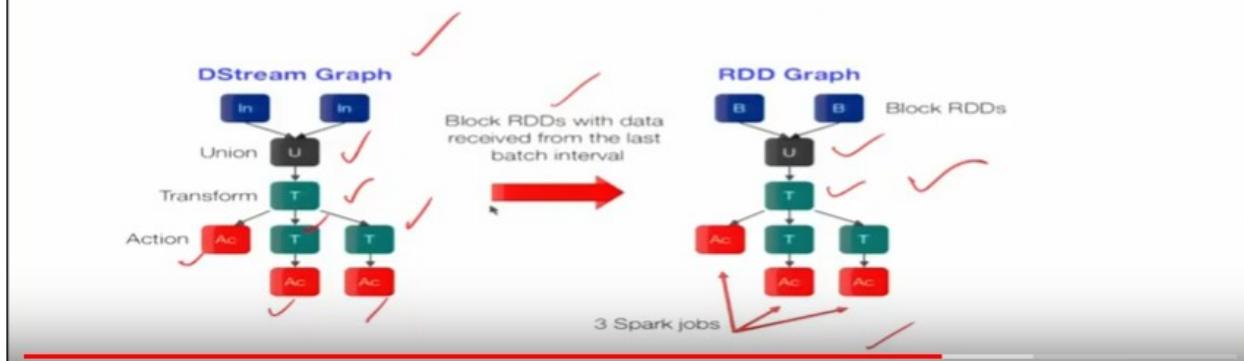
- Online machine learning
 - Continuously learn and update data models (*updateStateByKey* and *transform*)
- Combine live data streams with historical data
 - Generate historical data models with Spark, etc.
 - Use data models to process live data stream (*transform*)
- CEP-style processing
 - window-based operations (*reduceByWindow*, etc.)

Therefore, this dstream and RDDs together that is dstream is basically the the streaming data and RDDs are the batch data together if we add it will become the more power for example we can apply or we can introduce the online machine learning. So, that means we can apply the the RDDs when it becomes an RDD we can apply the machine learning that the library is on top of it hence it becomes a online machine learning technique that means machine learning applied on the dstreaming using this model. So, continuously learn and update the data model this can be performed using update state by the key and transformations. So, also there is a in this manner we can combine the live stream data with the historical data we generally we can generate the historical data model with the spark etc. Now, we can use the data model to process live data streams using transformations we can also do the CP style processing such as window based operations reduced by window etc.

Refer slide time :(20:26)

From DStreams to Spark Jobs

- Every interval, an RDD graph is computed from the DStream graph
- For each output operation, a Spark action is created
- For each action, a Spark job is created to compute it



So, from Dstream to the spark jobs we can see that every interval an RDD graph is computed from Dstream graph. And, for each output operation spark action is created for each action a spark job is computed. So, this is shown here in this particular example graph that it is a Dstream graph that input streams are coming and all these input different streams we can perform a union on it. And, we can perform different transformations and then perform various actions. So, these block of RDDs are then with the data received from the last batch interval is given back to the spark system in the form of RDD graph. And, RDD again will perform these RDDs and perform the union and again apply the transformation and give the output. So, this particular from a spark streaming weather and when the when the jobs are given for this part then again another level of transformations, can be applied before it can output.

Refer slide time :(21:34)

Input Sources

- Out of the box, we provide
 - Kafka, HDFS, Flume, Akka Actors, Raw TCP sockets, etc.
- Very easy to write a *receiver* for your own data source
- Also, generate your own RDDs from Spark, etc. and push them in as a “stream”

receivers

So, there are input streams which we have already seen let us summarize that that out of the box we have provided for the input sources, Kafka, HDFS, flume, Akka actors, raw TCP sockets and it is very easy to write a receiver for your own data source. So, these are different receivers which are inbuilt and you can also write down your own receiver for your data source also generate your own RDDs from the spark and push them as the stream.

Refer slide time :(22:09)

Current Spark Streaming I/O

- Input Sources
 - Kafka, Flume, Twitter, ZeroMQ, MQTT, TCP sockets
 - Basic sources: sockets, files, Akka actors
 - Other sources require receiver threads
- Output operations
 - Print(), saveAsTextFiles(), saveAsObjectFiles(), saveAsHadoopFiles(), foreachRDD()
 - foreachRDD can be used for message queues, DB operations and more



So, current a spark streaming input output we can say now summarizes as Kafka, Flume, Twitter, ZeroMQ and so on the basic sources are sockets file a character and so on. So, the output operations are print save as a file save as object file save as Hadoop files for each RDDs for each RDD can be used as a message queue and DB operations and many more things.

Refer slide time :(22:38)

Dstream Classes

- Different classes for different languages (Scala, Java)
- Dstream has 36 value members
- Multiple types of Dstreams
- Separate Python API

```
org.apache.spark.input      hide focus
  G PortableDataStream

org.apache.spark.serializer  hide focus
  G DeserializationStream

org.apache.spark.streaming.api.java
  G JavaDStream
  G JavaDStreamLike
  G JavaInputDStream
  G JavaPairDStream
  G JavaPairInputDStream
  G JavaPairReceiverInputDStream
  G JavaReceiverInputDStream

org.apache.spark.streaming.dstream
  G ConstantInputDStream
  G DStream
  G InputDStream
  G PairDStreamFunctions
  G ReceiverInputDStream
```

So, dstream classes different classes for different languages are ported Scala and Java these three miles36 different values value members and multiple type of rest reams and separate Python API will be provided.

Refer slide time :(22:54)

Spark Streaming Operations

- All the Spark RDD operations

- Some available through the transform() operation

map/flatmap	filter	repartition	union
count	reduce	countByValue	reduceByKey
join	cogroup	transform	updateStateByKey

- Spark Streaming window operations

window	countByWindow	reduceByWindow
reduceByKeyAndWindow	countByValueAndWindow	

- Spark Streaming output operations

print	saveAsTextFiles	saveAsObjectFiles
saveAsHadoopFiles	foreachRDD	

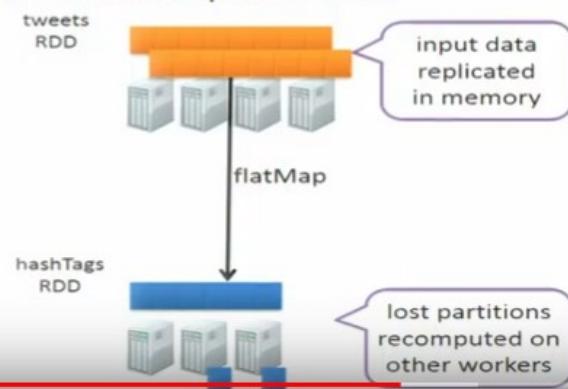
Now, Spark streaming operations are summarized here as the RDD operations and some or the transformations such as map and flat map that we have seen filter also we have seen repartition, Union, count, reduce, countbyvalue, reducebykey, join, cogroup, transform, and updatestatebykey are different transformations available in the spark RDDs. Now, it's spark streaming window operations are available such as window, countbywindow, count really reducebywindow, reducebykeyandwindow, countby valueandwindow. Similarly, for output operations in spark streaming various commands such as print, saveastextfiles, saveasobjectfiles, saveashadoopfiles, foreachRDD and so on.

Refer slide time :(23:46)

Fault-tolerance

- Batches of input data are replicated in memory for fault-tolerance
- Data lost due to worker failure, can be recomputed from replicated input data

- All transformations are fault-tolerant, and *exactly-once* transformations



So, therefore the batches of input data are replicated in the memory for fault tolerance. Data, lost due to the worker can be recomputed from the replicated input data. And, all transformed transformation or fault tolerant and follow the exactly ones transformations.

Refer slide time :(24:02)

Fault-tolerance

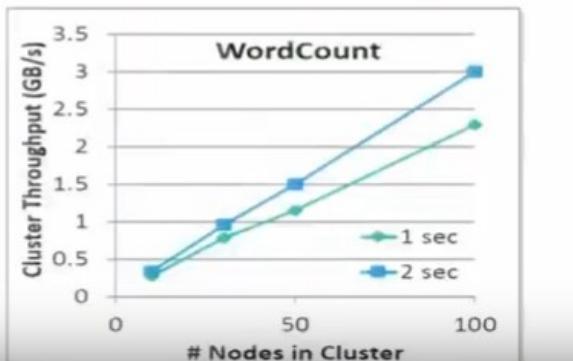
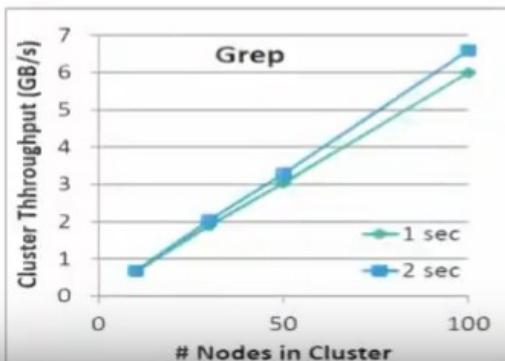
- Received data is **replicated** among multiple Spark executors
 - Default factor: 2
- **Checkpointing**
 - Saves state on regular basis, typically every 5-10 batches of data
 - A failure would have to replay the 5-10 previous batches to recreate the appropriate RDDs
 - Checkpoint done to HDFS or equivalent
- Must protect the **driver program**
 - If the driver node running the Spark Streaming application fails
 - Driver must be restarted on another node.
 - Requires a checkpoint directory in the StreamingContext
- **Streaming Backpressure**
 - spark.streaming.backpressure.enabled
 - spark.streaming.receiver.maxRate

So, fault tolerance is that is the receive data is replicated among multiple spark executors the default is to and this must protect the driver program there is only one driver running. So, if the driver node is running on the spark streaming and application fails driver must be restarted on another node. And, this can be handled using the zookeeper our yarn this was engine requires a check point directly here in the streaming context. So, check pointing saves the state on the regular intervals typically every five to ten batches of data the check point's being made. So, a failure would have to replay the five to ten previous batches to recreate the appropriate RDDs. So, checkpoint done to HDFS or equivalent so streaming back pressure proclaiming backpressure will be enabled and all these will achieve the fault tolerance. So, in nutshell we can say that a check pointing and replication together will ensure the the recovery of the failures.

Refer slide time :(25:19)

Performance

Can process **60M records/sec (6 GB/sec)** on
100 nodes at sub-second latency



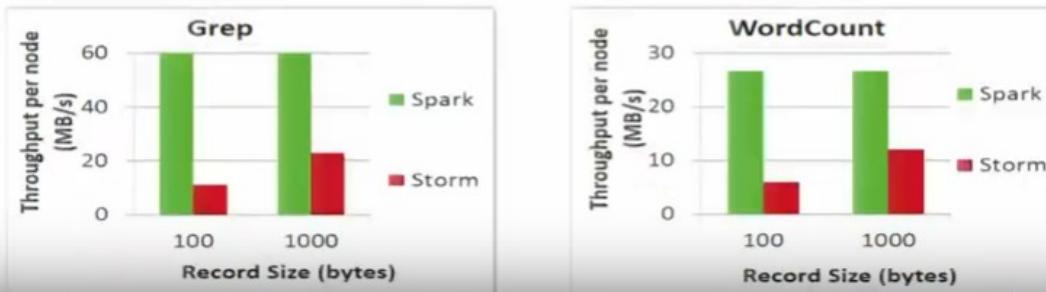
So, performance if we see that even the grep and the wordcount performance. So, on sub second latency it is four it is achieved in one second and two second a very good performance that we are seeing here.

Refer slide time :(25:39)

Comparison with other systems

Higher throughput than Storm

- Spark Streaming: **670k** records/sec/node
- Storm: **115k** records/sec/node
- Commercial systems: **100-500k** records/sec/node

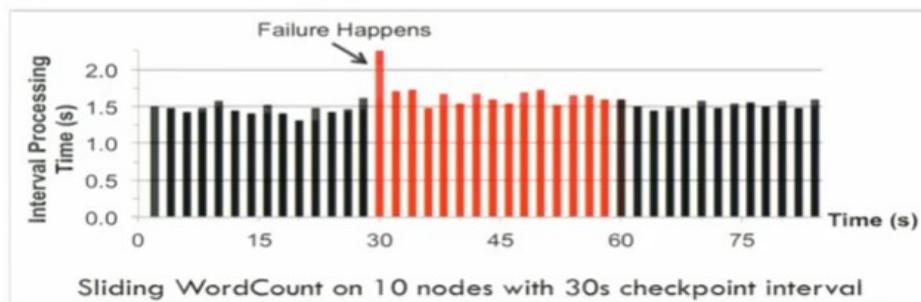


Compared, to the other systems this is also showing the better performance that is a spark system is having good performance and storm is basically achieving and the but the higher throughput a low throughput and a spark is having achieved the better throughput.

Refer slide time :(26:02)

Fast Fault Recovery

Recover from faults/stragglers within **1 sec**



So, fault tolerance recovery systems are there so it recovers from false or stragglers within one second.

Refer slide time :(26:10)

Real time application: Mobile Millennium Project

Traffic transit time estimation using online machine learning on GPS observations

- Markov-chain Monte Carlo simulations on GPS observations
- Very CPU intensive, requires dozens of machines for useful computation
- Scales linearly with cluster size

# Nodes in Cluster	GPS observations per sec
10	150
20	400
40	850
80	1700

So, this also is reported and

Refer slide time :(26:18)

Spark program vs Spark Streaming program

Spark Streaming program on Twitter stream

```
val tweets = ssc.twitterStream(<Twitter username>, <Twitter password>)
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.saveAsHadoopFiles("hdfs://...")
```

Spark program on Twitter log file

```
val tweets = sc.hadoopFile("hdfs://...")
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.saveAsHadoopFile("hdfs://...")
```

Now, let us see the spark program versus spark streaming program and here again those issues we have seen that. So, whether it is coming from the spark streaming or coming from the Hadoop file these particular things are going to be handled in the spark's training system,

Refer slide time :(26:50)

Advantage of an unified stack

- Explore data interactively to identify problems
- Use same code in Spark for processing large logs
- Use similar code in Spark Streaming for realtime processing

```
$ ./spark-shell
scala> val file = sc.hadoopFile("smallLogs")
...
scala> val filtered = file.filter(_.contains("ERROR"))
...
scala> val mapped = filtered.map(...)

object ProcessProductionData {
  def main(args: Array[String]) {
    val sc = new SparkContext(...)
    val file = sc.hadoopFile("productionLogs")
    val filtered = file.filter(_.contains("ERROR"))
    val mapped = filtered.map(...)

  }
}

object ProcessLiveStream {
  def main(args: Array[String]) {
    val sc = new StreamingContext(...)
    val stream = sc.kafkaStream(...)
    val filtered = stream.filter(_.contains("ERROR"))
    val mapped = filtered.map(...)

  }
}
```

both as a batch as well as streaming data. So, all these things we have already discussed there that is for having the unified stack and explore the data interactively to identify the problems and use the same code in the spark for processing large logs and use similar code in the Spark streaming for the real time. And, in the code we can see that we can apply the filter and then we can also invoke this particular fault tolerant aspect and then invoking the spark context and once the spark context is invoked then the driver and the executors are allocated

Refer slide time :(27:42)

Roadmap

- Spark 0.8.1
 - Marked alpha, but has been quite stable
 - Master fault tolerance – manual recovery
 - Restart computation from a checkpoint file saved to HDFS
- Spark 0.9 in Jan 2014 – out of alpha!
 - Automated master fault recovery
 - Performance optimizations
 - Web UI, and better monitoring capabilities
- Spark v2.4.0 released in November 2, 2018

and they are basically able to get ready for the computation.

Lecture – 21
Sliding Window
Analytics

Sliding-Window Analytics.

Refer Slide Time :(0: 15)

Sliding Window Analytics

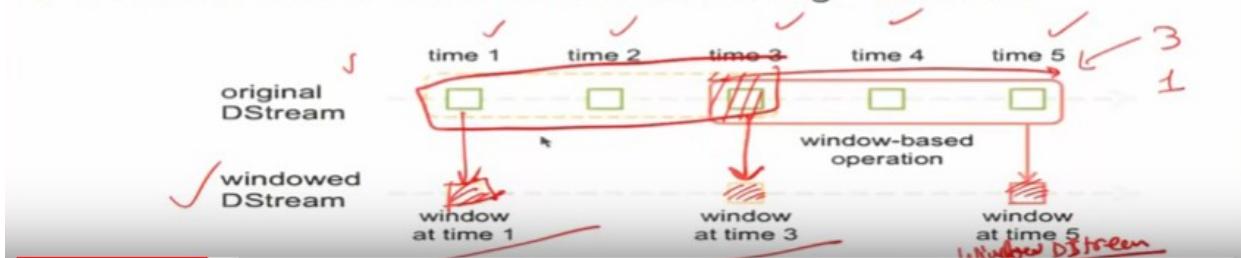
Spark Streaming → Function → Window
- many updates
- Analytics
- Sliding window analytics
Example ↓

So, in this spark streaming we will see that the most important functionality is the window based operations. So, using this window based operation we can design many application and also we can design many new algorithms for doing the analytics and therefore they are called as a ‘Sliding Window Analytics’. Some, of the some examples we are going to see in the further slides on this what are the different applications programs we can build, using this feature which is called a ‘Window’ and in hence we will see in more detail about the sliding window analytics.

Refer Slide Time :(1: 27)

Spark Streaming Windowing Capabilities

- **Parameters**
 - **Window length:** duration of the window
 - **Sliding interval:** interval at which the window operation is performed
 - Both the parameters must be a multiple of the batch interval
- A window creates a new DStream with a larger batch size



So, spark streaming windowing capabilities that is the window has window function has two parameters the first one is called ‘Window Length’ which is nothing but the duration of window, the second parameter is called the the ‘Sliding Interval’ which is an interval at which the window operation has been performed. So, both the parameters must be in a multiple of batch intervals. So, we have to see how these particular parameters we can use that is the window length and sliding interval and using this we can design many applications. So, a window creates a new DStream with a larger batch size. So, for example for example the if this is the input DStream and this particular input DStream is divided into the batches of time one, time two, time three, time four, time five and this window length is three of three batches which includes the window length is of three size three and sliding interval is also is basically the two then obviously there will be one overlap. So, that is shown over here. So, the original date DStream is shown here which is divided into the batches one, two, three, four, five and when the window operation is being performed that is the window head windowed DStream which will be based on per window this particular operation will be available. So, there will be an overlap of one. So, that means you see that in window at time one there will be a window DStream will be available and at time three they again there will be a window Dstream at time three will be available and then window at time four this window DStream will be available. So, these particular window based sliding interval will tell that the window operation is being performed this is the interval. So, sliding interval of size two will give so sliding interval of size one will give this kind of

Refer Slide Time :(4: 08)

Spark Window Functions

Spark Window Functions for DataFrames and SQL

Introduced in Spark 1.4, Spark window functions improved the expressiveness of Spark DataFrames and Spark SQL. With window functions, you can easily calculate a moving average or cumulative sum, or reference a value in a previous row of a table. Window functions allow you to do many common calculations with DataFrames, without having to resort to RDD manipulation.

Aggregates, UDFs vs. Window functions

Window functions are complementary to existing DataFrame operations: aggregates, such as sum and avg, and UDFs. To review, aggregates calculate one result, a sum or average, for each group of rows, whereas UDFs calculate one result for each row based on only data in that row. In contrast, window functions calculate one result for each row based on a window of rows. For example, in a moving average, you calculate for each row the average of the rows surrounding the current row. ~~this can be done with window functions.~~

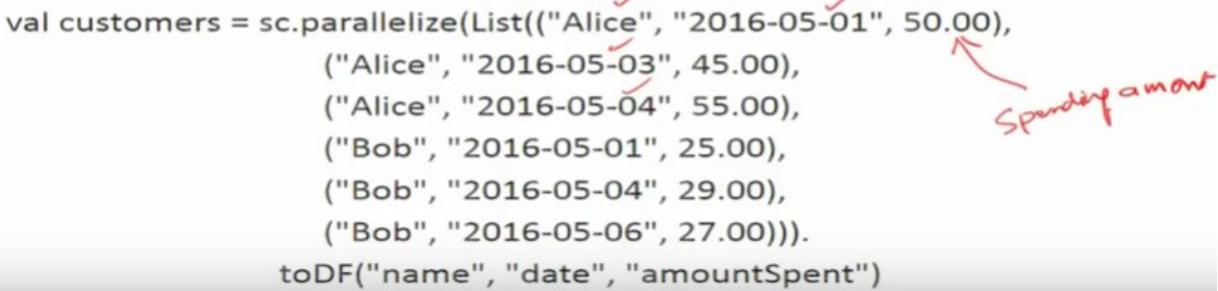
Now, Spark window functions spark window functions for data for data frame and SQL. So, which is introduced in a Spark 1.4 spark window functions improve the expressiveness of Spark data frames and Spark SQL. With, window functions you can easily calculate the moving average or cumulative sum or reference value in the previous row of a table. So, window function allows you to do many common calculations with it friends without having and without having to resort to RDDs manipulations. So, aggregates, UDF versus window functions. So, window functions are complementary to existing data frame operations: such as aggregates aggregate such as sum and average and UDF give you the aggregate calculations one result one result some are the average for each group of rows, whereas the UDF calculate one result for each row. So, based on only the data on in that row in contrast window functions calculate one result for each row based on the window off rows. For example, in a moving average you can calculate each row the average of the rows surrounding the current row and this can be done using the window functions.

Refer Slide Time :(5: 30)

Moving Average Example

- Let us dive right into the moving average example. In this example dataset, there are two customers who have spent different amounts of money each day.
- // Building the customer DataFrame. All examples are written in Scala with Spark 1.6.1, but the same can be done in Python or SQL.

```
val customers = sc.parallelize(List(("Alice", "2016-05-01", 50.00),
                                    ("Alice", "2016-05-03", 45.00),
                                    ("Alice", "2016-05-04", 55.00),
                                    ("Bob", "2016-05-01", 25.00),
                                    ("Bob", "2016-05-04", 29.00),
                                    ("Bob", "2016-05-06", 27.00))).  
toDF("name", "date", "amountSpent")
```



So, let us see this moving average example. So, let us dive into the moving average example in this example the data set there are two customers who have different spending amounts of money each day. So, let us build the the customer dataframe and all the examples are written in the scala and the same can be visualized using Python or SQL. So, let us see the customer and its data is given in the form of the list such as the customer is Alice and 2015, 2016 5th month and and the date is given over here and the time is also B and the spending is given. So, this is the the amount of spending amount is given and this is the name of the customer and these are their dates when the customer has done this spending. So, given this particular detail that this one the name of the customer a date and the amount which is spent. So, given this particular data of customers and.

Refer Slide Time :(6: 46)

Moving Average Example

```
// Import the window functions.  
import org.apache.spark.sql.expressions.Window  
import org.apache.spark.sql.functions._
```

// Create a window spec.

```
val wSpec1 =  
  Window.partitionBy("name").orderBy("date").rowsBetween(-1, 1) |||
```

- In this window spec, the data is partitioned by customer. Each customer's data is ordered by date. And, the window frame is defined as starting from -1 (one row before the current row) and ending at 1 (one row after the current row), for a total of 3 rows in the sliding window.

Now, we can perform the window operations on top of it to calculate the moving average of spending. So, moving average. So, create the window specifications and so we have to specify the window. So, so the window will partition by the name and order by the date and the windows will calculate we will consider the rows between -1 and 1. So, -1 and 1 means in this window specs the data is partitioned by the customer and each customer's data is ordered by the date and the window frame is defined as starting from -1 that is one row before the current row and ending at one that is one row after the current row. So, the total there are three different rows are considered in the sliding window.

Refer Slide Time :(8: 01)

Moving Average Example

- Let us dive right into the moving average example. In this example dataset, there are two customers who have spent different amounts of money each day.
- // Building the customer DataFrame. All examples are written in Scala with Spark 1.6.1, but the same can be done in Python or SQL.

```
val customers = sc.parallelize(List(("Alice", "2016-05-01", 50.00),
  ("Alice", "2016-05-03", 45.00),
  ("Alice", "2016-05-04", 55.00),
  ("Bob", "2016-05-01", 25.00),
  ("Bob", "2016-05-04", 29.00),
  ("Bob", "2016-05-06", 27.00))).  
  toDF("name", "date", "amountSpent")
```

Annotations:

- (Current row) points to the first row: ("Alice", "2016-05-01", 50.00)
- Spending amount points to the "amountSpent" column
- window(-1, +1) points to the range of rows used for the window: -1, 0, +1
- 3 rows into window for operation points to the total number of rows in the window: 3

So, three rows means that for example if this is the current row then this row then the row above this row will be the -1 and the row below this will be the +1. So, the window of size one and -1 will require three rows into the window for operations. So, therefore the the window frame is defined as starting from -1 that is one row before the current row and ending at one that is one to after the current row. So, total there are three different rows are there in the sliding window. So, having understood this notion in this particular statement or written in the Scala we have to specify using window specification. So, window partition by the name order by the row and the the rows between -1 and 1. So, that will specify the window operations.

Refer Slide Time :(9: 26)

Moving Average Example

```
// Calculate the moving average  
customers.withColumn( "movingAvg",  
  avg(customers("amountSpent")).over(wSpec1) ).show()
```

This code adds a new column, "movingAvg", by applying the avg function on the sliding window defined in the window spec:

name	date	amountSpent	movingAvg
Alice	5/1/2016	50	-
Alice	5/3/2016	45	50
Alice	5/4/2016	55	50
Bob	5/1/2016	25	27
Bob	5/4/2016	29	27
Bob	5/6/2016	27	28

$$\begin{aligned} \text{Row 1: } & \frac{50+45}{2} = 47.5 \\ \text{Row 2: } & \frac{50+45+55}{3} = 50 \\ \text{Row 3: } & \frac{45+55+29}{3} = 50 \end{aligned}$$

Now, let us calculate the moving average using this particular scala command that is the customers with column and moving average we have to calculate using the average of all of the customers by the account amount amount spent or the windows specification which we have shown and that we have to show. So, this code will now add a new column which is called ‘Moving Average’. So, moving average column will be now added in the customers table and by applying the average function. So, this average function on the sliding window which is defined in the window specs in the previous slide. So, that means let us see how 47.5 which we have obtained is that if this is the customer its spending is 50. So, before that is the row number -1 which is missing. But, after this there is a row which is called ‘45’ which is available. So, the window of so this window will consider 50 plus 45 divided by 2 that is nothing but 47.5. Now, consider the now consider this particular entry that is 45 will consider this row as -1 and this row as +1. So, this window will contain all three values and they will be $50 + 45 + 55$ and if you take this average it comes out to be 50. So, this value will come over here again. Now, you will consider this third entry that is 55. So, this particular row will be -1 and the further row will be +1. So, for this to calculate the average we require $45 + 55 + 25$ divided by 3 that comes out to be 50. Similarly, for 25 we can see that for 25 we can see that this becomes -1 and this becomes +1. So for 25 these 3 will these three values will be taken up into the window for everything $55 + 25 + 29$ no not that actually. So, so up to this point 25 will not be used why because? It belongs to another customer. So, so here $45 + 55$ divided by 2 that will be about 24. So, let us see about the Bob. So, Bob the previous values will be let us see this case of Bob again. So, in Bob case, the previous value is not there and +1 will be there. So, only these 2 values will be used, why because? Before that it is an, Alice values.

Refer Slide Time :(13: 05)

Window function and Window Spec definition

- As shown in the above example, there are two parts to applying a window function: (1) specifying the window function, such as avg in the example, and (2) specifying the window spec, or wSpec1 in the example. For (1), you can find a full list of the window functions [here](#):
- [https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.sql.functions\\$](https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.sql.functions$)
- You can use functions listed under “Aggregate Functions” and “Window Functions”.
- For (2) specifying a window spec, there are three components: partition by, order by, and frame.
 - “Partition by” defines how the data is grouped; in the above example, it was by customer. You have to specify a reasonable grouping because all data within a group will be collected to the same machine. Ideally, the DataFrame has already been partitioned by the desired grouping.
 - “Order by” defines how rows are ordered within a group; in the above example, it was by date.
 - “Frame” defines the boundaries of the window with respect to the current row; in the above example, the window ranged between the previous row and the next row.

So, 25 so its its average will be $25 + 29$ divided by 2 that is 27. And, similarly for this value we will consider it as -1 and this is +1. So, $25 + 29 + 27$ divided by 3 that comes out to be 27. And, for 27 we will

see that for 27 it will only use -1. So, $29 + 27$ divided by 2 that comes out to be, 28. So, this way the moving average is automatically calculated and now this is the the small number of entries that we have to see that that the window is sliding and is the moving average is now calculated and filled in this particular table. So, window based function and window specification definition as shown in the example that there are two parts to the two applying the window function first is specifying the window function such as the average in the example and the second is specifying the window specs that is windows specification. So, for the average you can find the full list of window functions available on the this one streaming, Spark Streaming functions which you can apply you can also apply the aggregate functions and the window functions together for two that is specifying the window aspects there are three different components partitioned by order by and frame. So, partition by defines how the data is grouped in the above example it was customer you can you have to specify reasonable grouping because all the data within the group will be collected on to the same machine. So, ideally the dataframe has already being partitioned by the desired groupings. So, when you go for order by it will define how the rows are ordered within the group in the above example it was the by the date. Similarly, the frames defines the boundaries of the window with as but to the current row in the above example the window reached between the previous row and the next row. So, all these are very much needed and once it is understood then the partition by and order by together will perform the optimizations in the data movement from different cluster systems and secondly the the frame that is the window operations is very very much needed or moving average and it depends upon the application how this particular window is used for that application.

Refer Slide Time :(16: 08)

Cumulative Sum

Next, let us calculate the cumulative sum of the amount spent per customer.

// Window spec: the frame ranges from the beginning (Long.MinValue) to the current row (0).

```
val wSpec2 =
Window.partitionBy("name").orderBy("date").rowsBetween(Long.MinValue, 0)
// Create a new column which calculates the sum over the defined window
frame.
customers.withColumn( "cumSum",
sum(customers("amountSpent")).over(wSpec2) ).show()
```

name	date	amountSpent	cumSum
Alice	5/1/2016	50 ✓	50 ✓
Alice	5/3/2016	45 ✓	95 →
Alice	5/4/2016	55 →	150 ✓
Bob	5/1/2016	25 ✓	25 ✓
Bob	5/4/2016	29 ✓	54 ✓
Bob	5/6/2016	27 ✓	81 ✓

Now, we will see another application of window based analysis analytics which is called ‘Cumulative Sum’. Let, us calculate the cumulative sum of the amounts spent per customer here the window aspects the the frame ranges from beginning that is the long and minimum value to the current row that is zero and window specification is aspects to we are now saying that you know to partition by the name like in the previous example order by the date also by the previous as per previous example and the rows

between the long mean value zero. So, let us create the new column which calculates the sum or that define window frame. So, the customer with the column with the column that is cumulative sum that is sum we have to define as function of the customer amount is spent over the window specs which we are now going to show. So, minimum value along minimum value and zero we are going to so. So, Alice has spent 50. So, the accumulated amount is 50 only then the as Alice has spent 45. So, cumulative will be 95 automatically will be calculated then Alice has spent 45 and 55 then it becomes 150 in this case. Then, Bob has spent 25. So, it becomes 25 it has this point 29 it becomes 54 then it becomes then it has expired 27 it and the cumulative amount becomes 81. So, let us see that the new row which is called ‘Cumulative Sum’ and this will be this will be created using Queue using this particular command customer with the column a new column will be added and will be calculated in this,

Refer Slide Time :(18: 12)

Data from previous row

In the next example, we want to see the amount spent by the customer in their previous visit.

// Window spec. No need to specify a frame in this case.

```
val wSpec3 = Window.partitionBy("name").orderBy("date")
```

// Use the lag function to look backwards by one row.

```
customers.withColumn("prevAmountSpent",
lag(customers("amountSpent"), 1).over(wSpec3) ).show()
```

name	date	amountSpent	prevAmountSpent
Alice	5/1/2016	50	null
Alice	5/3/2016	45	50
Alice	5/4/2016	55	45
Bob	5/1/2016	25	null
Bob	5/4/2016	29	25
Bob	5/6/2016	27	29

using the the window functions.

Refer Slide Time :(18: 20)

Rank

- In this example, we want to know the order of a customer's visit (whether this is their first, second, or third visit).

// The rank function returns what we want.

```
customers.withColumn( "rank", rank().over(wSpec3) ).show()
```

name	date	amountSpent	rank
Alice	5/1/2016	50	1
Alice	5/3/2016	45	2
Alice	5/4/2016	55	3
Bob	5/1/2016	25	1
Bob	5/4/2016	29	2
Bob	5/6/2016	27	3

Now, we are going to see the “rank” in this example we want to know the order of the customers visit to the store whether this is the first time, second time or third visit. So, that is called the ‘rank’. So, rank function returns what we want? So, customer with the column and we are going to add the rank within that particular column. So, you see the rank is headed over here and we want to rank or the window specification and and then we have to show. So, window a specification we are doing the same window specification which we have specified in the previous example and now let us see that is the Alice when it has a span when it has visited on this date and spend 50 amount then first time it has visited then next day it has spent 45 then second time it has visited and the next time it is third time. Similarly, the the Bob has be stayed three times. So, as it is visiting and spending on another date the automatically the rank of that customer is increasing. Now, this kind of analysis or analytics is very much useful in many of the websites which are doing the e-commerce or doing the business and based on for example the first time a customer is logging into the system or doing the business with that or taking the service then as far as the promotions are concerned using this particular information it will give some promotional discount for the new customer to be attracted and for the old customers these particular values are changed accordingly based on the policies of the business.

Refer Slide Time :(20: 12)

Case Study: Twitter Sentiment Analysis with Spark Streaming

Now, let us see another case study of this Spark Streaming analytics using twitter sentiment analysis using the Spark Streaming.

Refer Slide Time :(20: 23)

Case Study: Twitter Sentiment Analysis

- Trending Topics can be used to create campaigns and attract larger audience. Sentiment Analytics helps in crisis management, service adjusting and target marketing.
- Sentiment refers to the emotion behind a social media mention online.
- Sentiment Analysis is categorising the tweets related to particular topic and performing data mining using Sentiment Automation Analytics Tools.
- We will be performing Twitter Sentiment Analysis as an Use Case or Spark Streaming.



So, twitter sentiment analysis that means here we want to find out the trending topics can be used to create the campaigns and attract large larger audience sentiment analysis helps in the crisis management and service adjusting and and target marketing. So, therefore the sentiment analysis is going to become very important thing in the the web and in all businesses and the sentiment referred to the emotion emotion behind the social media mentioned online sentiment analysis is categorizing the tweets related to the particular topic and performing data mining using sentiment automation analytics tools. We, will be performing twitter sentiment analysis as an use case or streaming data.

Refer Slide Time :(21: 16)

Problem Statement

- To design a Twitter Sentiment Analysis System where we populate real-time sentiments for crisis management, service adjusting and target marketing.

Sentiment Analysis is used to:

- Predict the success of a movie
- Predict political campaign success
- Decide whether to invest in a certain company
- Targeted advertising
- Review products and services

So, let us see the what we are going to see here is to design a sentiment a twitter sentiment analysis system where we populate the real-time sentiments for crisis management service adjusting and the target marketing. Sentiment analysis is used to predict the success of a movie, predict the political campaign, success decide whether to invest in a certain company target, advertising, review the product and services. Refer Slide Time :(21: 46)

Twitter Token Authorization

```
object mapr {  
  
  def main(args: Array[String]) {  
    if (args.length < 4) {  
      System.err.println("Usage: TwitterPopularTags <consumer key>  
<consumer secret> " +  
        "<access token> <access token secret> [<filters>]")  
      System.exit(1)  
    }  
  
    StreamingExamples.setStreamingLogLevels()  
    //Passing our Twitter keys and tokens as arguments for authorization  
    val Array(consumerKey, consumerSecret, accessToken,  
    accessTokenSecret) = args.take(4)  
    val filters = args.takeRight(args.length - 4)  
  }  
}
```

So, let us go and see this particular details. So, we have to follow the pipeline this is called token ‘Token Authorization’ and here the twitter, which are given as the input,
Refer Slide Time :(22: 05)

DStream Transformation

```
// Set the system properties so that Twitter4j library used by twitter stream
// Use them to generate OAuth credentials
System.setProperty("twitter4j.oauth.consumerKey", consumerKey)
System.setProperty("twitter4j.oauth.consumerSecret", consumerSecret)
System.setProperty("twitter4j.oauth.accessToken", accessToken)
System.setProperty("twitter4j.oauth.accessTokenSecret",
accessTokenSecret)

val sparkConf = new ✓
SparkConf().setAppName("Sentiments").setMaster("local[2]")
val ssc = new StreamingContext(sparkConf, Seconds(5) ✓)
val stream = TwitterUtils.createStream(ssc, None, filters)

//Input DStream transformation using flatMap
val tags = stream.flatMap { status =>
status.getHashtagEntities.map(_.getText) }
```

we are going to apply and now we perform the part the Streaming DStreaming transformations on this dstream Twitter D stream which is now captured and in the as the stream and given here in the system. So, the first statement is about we have to start a new Spark configuration and we have to set the Spark Streaming or DStreaming context. So, Streaming context, is just like a Spark context here we are using the Streaming context and this particular Streaming context we are creating the window of 5 seconds. Now, this utility Twitter utility will create the stream and this stream will will use the Twitter utility and this will be the input stream to the Spark system is Streaming system and it will apply some filters filters are none and so all the stream will be taken up into the Streaming system. Now, this particular Dstream will be transformed using flat map. So, flat map will use the will get the hash tag entries and it will get the text out of the hash tag entries and this is called the ‘tags’. So, out of the stream it will capture the hash tag entries.

Refer Slide Time :(23: 44)

Results

```

Markers Properties Servers Data Source Explorer Snippets Console Scala Interpreter (TwitterStreaming)
<terminated> mapr$ [Scala Application] /usr/lib/jvm/java-8-openjdk-1.8.0_131-b13-2~14.04.1~Ubuntu~14.04.1-b02 /java (09-Feb-2017, 11:56:26 AM)
debug: weighted: 1.0
-----
Time: 1486621640000 ms
-----
(# 葉足、半導体新機を着工→メモリー製造、18年夏完成へ https://t.co/DU5goZAp25 #不動産 #投資 #アート→ #地元 #経営 NEGATIVE ([Ljava.lang.String:@1a25ec3])
#RT @Bts_bighit: [奉呈] O. 투표하는 걸 침자?
#OUI: 좋아요~ 카트리▶ 늘 새로운!▶ 재탄생스만 투표하는 게 최고야

#기운차트어워드 https://t.co/0HDwR2xt4
#ShortyAwards https://t... NEGATIVE ([Ljava.lang.String:@121986a)
#RT @MukePL: Jezeli na typie edycji widzisz swoj swiat to daj RT. #oneBestfans & #5505bestfans https://t.co/rn2EMWvPjP NEGATIVE ([Ljava.lang.String:@1c3681d)
#RT @Morocasts: #Cancer most enduring quality is an unexpected silly sense of humor. POSITIVE ([Ljava.lang.String:@174e1a2)
(I'm listening to "A Song For Mama" by @BezyIDMen on @PandoraMusic. #pandora https://t.co/7In5w43CYB NEUTRAL ([Ljava.lang.String:@95f6d4)
('Greenwashing' Costing Walmart $1 Million https://t.co/DBK02R0MNP #Biodegradability #Compostability #biobased NEGATIVE ([Ljava.lang.String:@1511e25)
#RT @CamilaXaxidiah: Serayah representando a las camilizers cuando un hombre se le acerca a Camila #CamilaBestFans https://t.co/8LgppQ0RGe NEGATIVE ([Ljava.lang.String:@78c835)
#RT @CamilaVoteStats: #CamilaBestFans https://t.co/qSLx0p0ln NEUTRAL ([Ljava.lang.String:@16e7255)
(@tos 六甲道駅 https://t.co/0rk18rlSb3 #TFB NEGATIVE ([Ljava.lang.String:@1a3fe)
(Imar pro Marcos: "Vai dormir puta.. Bebe e fica ai com o cu quente." XXXXXXXXXXXXXXXXXXXXXXXXX #888817 NEGATIVE ([Ljava.lang.String:@1516ece)
...

```

So, this is the some results which we have gathered into the captured into the system.

Refer Slide Time :(23: 56)

Sentiment for Trump

```

mapr.scala earth.scala
57   val tags = t.getText.split(" ").filter(_.startsWith("#")).map(_.toLowerCase)
58   tags.contains("#obama") && tags.contains("anyc")
59 }
60 */
61 val tweets = stream.filter { t =>
62   val tags = t.getText.split(" ").filter(_.startsWith("Trump")).map(_.toLowerCase)
63   tags.exists { x => true }
64 }
65 }
66
67
68 val data = tweets.map { status =>
69   val sentiment = SentimentAnalysisUtils.detectSentiment(status.getText)
...

```

```

Markers Properties Servers Data Source Explorer Snippets Console Scala Interpreter (Twitt
<terminated> mapr$ [Scala Application] /usr/lib/jvm/java-8-openjdk-1.8.0_131-b13-2~14.04.1~Ubuntu~14.04.1-b02 /java (09-Feb-2017, 6:06:24 PM)
debug: weighted: 1.0
-----
Time: 1486645210000 ms
-----
(#USA Trump Suggests That Supreme Court Nominee's Criticism of Him Misrepresented: Trump questioned whether... https://t.co/1Zctok4P43 #news NEGATIVE ([Ljava.lang.String:@10f96b1)
#RT @WorldStarLaugh: Compilation of Donald Trump's greatest accomplishments as president https://t.co/Got6efwiMH POSITIVE ([Ljava.lang.String:@e5a4fe)
(BBCNewsnight: Should the UK roll out the red carpet for President Trump? Here's what Hillary Clinton's campaign ma... https://t.co/hjKNuJJu3s NEUTRAL ([Ljava.lang.String:@146dc81)
#RT @Jdxthompson: Ellen DeGeneres' response to Donald Trump screening "Finding Dory" at The White House is everything b... https://t.co/k0QCPuL NEGATIVE ([Ljava.lang.String:@116c5fd)
#RT @Calilelia: Trump: Ivanka "always pushing me to do the right thing." He needs a push to do the right thing? @anavarro @VanJones68 @Ch... NEUTRAL ([Ljava.lang.String:@129dc11)

```

And, now we are going to analyze analyzing the sentiment of a particular politician
Refer Slide Time :(24: 05)

Applying Sentiment Analysis

- As we have seen from our Sentiment Analysis demonstration, we can extract sentiments of particular topics just like we did for 'Trump'. Similarly, Sentiment Analytics can be used in crisis management, service adjusting and target marketing by companies around the world.
- Companies using Spark Streaming for Sentiment Analysis have applied the same approach to achieve the following:
 - Enhancing the customer experience
 - Gaining competitive advantage
 - Gaining Business Intelligence
 - Revitalizing a losing brand

using this and now we are applying the sentiment analysis as we have seen from our sentiment analysis demonstration we can extract the sentiment of a particular topic and we did for a politician 'Trump'. Similarly, sentiment analysis can be used in the crisis management and service adjusting and target marketing by the companies around the world. Companies, use using the Spark stream for sentiment have applied the same approach to achieve the following enhancing the customer experience, gaining the competitive advantage, gaining business intelligence, revitalizing losing brand.

Refer Slide Time :(24: 45)

References

- <https://spark.apache.org/streaming/>
- Streaming programming guide –
spark.incubator.apache.org/docs/latest/streaming-programming-guide.html
- <https://databricks.com/speaker/tathagata-das>

So, these are the following references for further details to refer the Spark spark.apache.org/streaming and the Streaming programming guide is also their spark.incubator.apache.org and also we have used some of the material from databricks.com speaker the Tathagata Das.

Refer Slide Time :(25: 18)

Conclusion

- Stream processing framework that is ...
 - Scalable to large clusters
 - Achieves second-scale latencies
 - Has simple programming model
 - Integrates with batch & interactive workloads
 - Ensures efficient fault-tolerance in stateful computations

Conclusion, stream processing framework that is scalable to the large clusters achieves seconds of scaled latency has the programming simple programming model will integrate into the batch and interactive workloads, ensures efficient power tolerance in the stateful computations. Thank you.

Lecture 22 Introduction to Kafka

Introduction to Kafka

Refer slide time :(0:17)

The screenshot shows a presentation slide with a blue header bar containing the word 'Preface'. Below the header, the text 'Content of this Lecture:' is displayed in red. A bulleted list follows, detailing the topics to be covered:

- Define Kafka
- Use cases for Kafka
- Kafka data model
- Kafka architecture
- Types of messaging systems
- Importance of brokers

At the bottom of the slide, there is a navigation bar with icons for back, forward, and search, along with a timestamp '0:29 / 33:34'. To the right of the timestamp, the text 'Big Data Computing' is visible. On the far right, there is a footer bar with the text 'Introduction to Kafka' and several small icons.

Preface content of this lecture: we will define what is a Kafka, we will see the use case of Kafka and Kafka is a data model, Kafka is a architecture, types of messaging system, importance of Kafka as the broker.

Refer slide time :(0:33)

Introduction to Kafka



*Data ingestion phase
of Big data Computing
Kafka can be used for
Data ingestion*

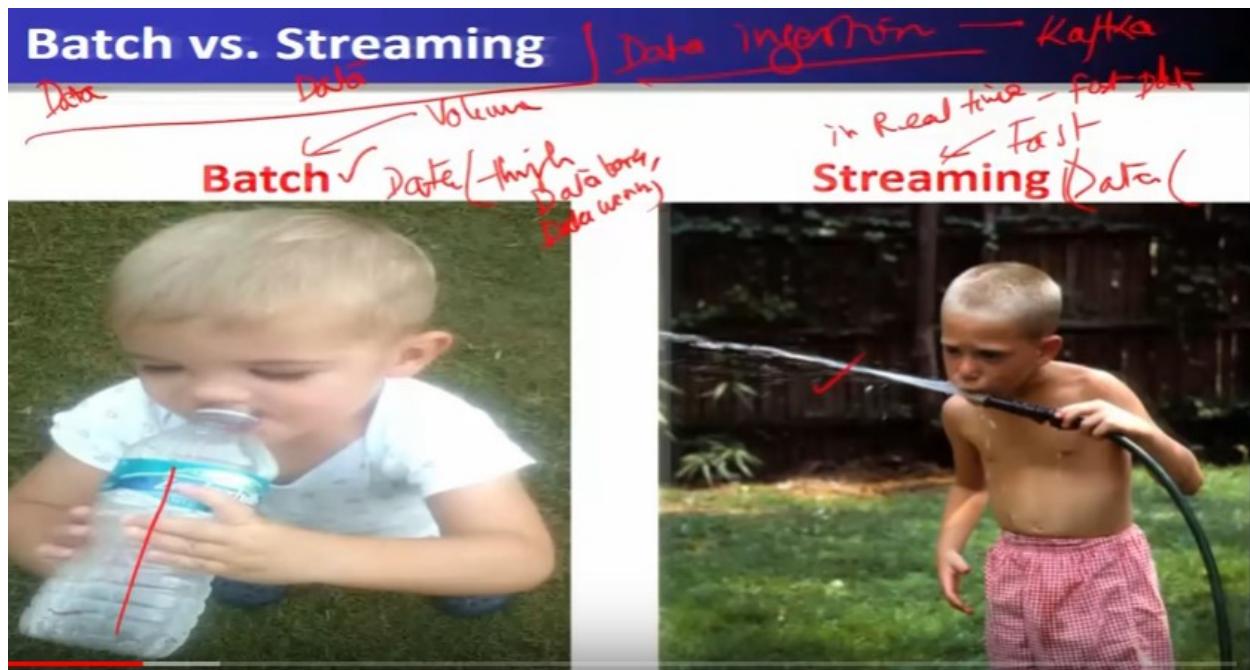
Dr. Rajiv Misra

Dept. of Computer Science & Engg.
Indian Institute of Technology Patna
rajivm@iitp.ac.in

Introduction to Kafka

So, before we go ahead let us see where what is the use of Kafka in big data. So, the Kafka is used for the data injection, data ingestion of big data computing. So, in this particular discussion we will see the details of the Kafka and how the Kafka can be used for data injection.

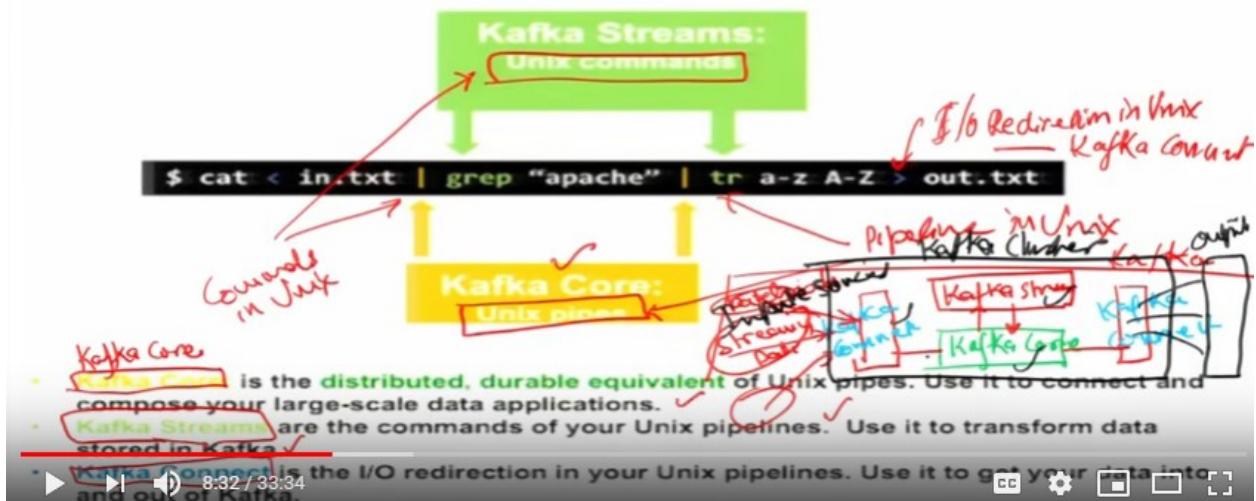
Refer slide time :(01:23)



Before we understand before we go in more detail let us understand two important things using this particular picture. One, is the batch batch jobs means data stored this is the stored water this is if it is consumed then it is called ‘Batch Processing’ whereas the the water which is in the form of stream if you are consuming then it is called ‘Stream’. So, it is the data if we are consuming in our scenario it is called a batch ‘Data’ if we are using the the stored data for its consumption that is through the through the databases data warehouses and so on. And, this is the streaming data if we are consuming the data for any application such as the Twitter stream data or the socket API socket connection which brings into the network data and so on. So, those data is called the ‘Streaming Data’. So, as the data is flowing we have to compute this particular data then it is called the ‘Streaming Data Processing’. So, this kind of data here we have the big volume of data which is we have to store that is the volume we have to deal in the big data similarly the streaming data that means data is moving very fast and we have to compute this processing which cannot be done by a single computer system. So, that means how we have to compute in the real time this fast data. So, we will see here what do you mean by the how batch data and what is the streaming data and how we are going to handle these cases in the data injection system. So, what is the technology? So, we will be discussing the Kafka as the technology which deals with the both type of input sources of the data that is the best data as the input source and the streaming data as the input source.

Refer slide time :(03:54)

1. Apache Kafka: a Streaming Data Platform Unix Pipelines Analogy



So, before we go ahead let us see the Apache Kafka as the streaming data platform having the UNIX pipeline analogy. So, just if you have familiar with the UNIX pipeline analogy now we will understand using that analogy about the Kafka Apache Kafka. So, here in the UNIX domain you might have noticed that there are certain Unix commands and we can now form a pipeline of these set of unit command Unix command together and this is called the 'Unix Pipeline'. So, similar to this analogy of Unix we will see different components of Kafka. So, there is a Kafka core which is like UNIX pipes this is called 'Kafka Core'. So, Kafka core is a distributed durable equivalent of UNIX pipes. So, it used to connect and compose your large-scale data application the second aspect here is called 'Kafka Streams' which is having the similar analogy as the commands you use in the UNIX pipelines. So, this particular commands are used to transform the data which is stored in the Kafka third component is called 'Kafka Connect'. So, Kafka Connect is an I/O redirection just like here and this indicates Kafka connector. So, Kafka connect is the I/O it is having analogy with the I/O to direction in the UNIX pipelines. So, it is used to get your data into and out of the Kafka system. So, if we draw a diagram of a Kafka system we will how this is called 'Kafka Connect' the next thing is about the Kafka streams, Kafka core and then we have the Kafka streams. So, this is the complete picture and here there will be the input sources input sources may be generating the batch data, streaming data. And, this is called the 'Input Sources'. And, now here there will been output or a consumers or output sources this is called the 'Kafka cluster'. Now, let us go and understand the design of each and every component of these Kafka that is what do you mean by the Kafka connects. And, Kafka stream and Kafka core and how together all of these components can be used to build the data, ingestion and data computation pipeline in a big data scenario.

Refer slide time :(08:40)

Introduction: Apache Kafka

- **Kafka is a high-performance, real-time messaging system. It is an open source tool and is a part of Apache projects.**
- The characteristics of Kafka are:
 1. **It is a distributed and partitioned messaging system.**
 2. **It is highly fault-tolerant**
 3. **It is highly scalable.**
 4. **It can process and send millions of messages per second to several receivers.**



9:01 / 33:34

Big Data Computing

Introduction to Kafka



So, Kafka is the high performance real time messaging system and it is an open source tool and is the part of Apache project. So, the characteristics of Kafka are it is the distributed and partition messaging system, it is highly called tolerant, it is highly scalable, it can process and send millions of messages per second to the several receivers.

Refer slide time :(09:01)

Kafka History

- Apache Kafka was originally developed by LinkedIn and later, handed over to the open source community in early 2011.
 - It became a main Apache project in October, 2012.
 - A stable Apache Kafka version 0.8.2.0 was released in Feb, 2015.
 - A stable Apache Kafka version 0.8.2.1 was released in May, 2015, which is the latest version.

So, the history of apache Kafka was originally developed by Lyndon and later on handed over to the open source community in the 2011 it became a main Apache project in October 2012. A stable Apache Kafka version, 0.8.2 released in February 2015 a stable Kafka version 0.8.2.1 was released in May 2015 which is the latest version.

Refer slide time :(09:43)

Kafka Use Cases

- Kafka can be used for various purposes in an organization, such as:

Messaging service	Millions of messages can be sent and received in real-time, using Kafka.	
Real-time stream processing	Kafka can be used to process a continuous stream of information in real-time and pass it to stream processing systems such as Storm.	
Log aggregation	Kafka can be used to collect physical log files from multiple systems and store them in a central location such as HDFS.	
Commit log service	Kafka can be used as an external commit log for distributed systems.	
Event sourcing	A time ordered sequence of events can be maintained through Kafka.	

So, let us see some of the use cases of Kafka so Kafka can be used for various purposes for ended in the industries such as Kafka can be used as a messaging server messaging service that is millions of messages can be sent and received in the real time using Kafka. So, real time stream processing that means Kafka can be used to process a continuous stream of information in real time and pass it to the stream processing system such as storm. So, log aggregation that is Kafka can be used to collect the physical log files with from multiple systems and store them in the central location such as HDFS commit log service Kafka can be used as an external commit log for distributed systems even sourcing time ordered sequence of events can be maintained through the Kafka. So, these are some of the important use cases of Kafka which is used in different organizations and is used in applications. So, let us summarize it that Kafka can be used as a messaging server for handling with the multiple numbers of messages and in the real time its connotations. Real-time stream processing cover can be used to process continuous stream of information and the the log aggregation that is Kafka can be used to collect the physical log files from multiple systems. And, store them to the central agencies so commit log service Kappa can be used as an external commit log for distributed systems event sourcing that is a time ordered sequence of events can be maintained through the Kafka Apache Kafka a streaming data platform.

Refer slide time :(11:49)

Apache Kafka: a Streaming Data Platform

- Most of **what a business does** can be thought as **event streams**. They are in a
 - **Retail system**: orders, shipments, returns, ...
 - **Financial system**: stock ticks, orders, ...
 - ✓ **Web site**: page views, clicks, searches, ... (Stream data)
 - **IoT**: sensor readings, ...

and so on.



So, most of what a business does can be thought of as an event streams. So, they are used in the retail system in the form of orders shipments and returns in the financial system such as stock ticks orders etc the web site such as the page views clicks searches and IoT that means sensor readings and so on. So, all these are the data which is called the streaming data which is required to be input into the systems which is shown here in this particular diagram that the the input streams are stream data which basically is in the form of the sales and the shipment details also is a streaming form of data than various price adjustments reordering and and inventory adjustments. So, all these are some of the data system which is out of the retail system similarly in the financial systems the stop ticks different stock orders that will be the stream data which is required to be computed as an event for the different business application similarly as far as the website is concerned page views page clicks searches on the website also will generate the stream of the data as an event IoT devices wherein the sensors are reading taking the readings and now generating the events streams out of this IoT sensor data is also one of the streaming data platform and is supported in the Kafka.

Refer slide time :(14:03)

Enter Kafka

- Adopted at 1000s of companies worldwide

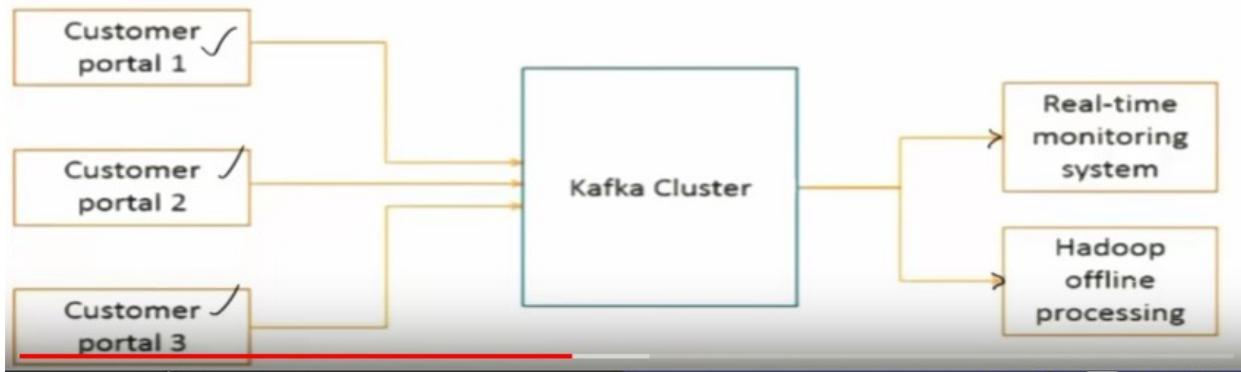


So, the Kafka we see here more than 1000 different companies worldwide uses Kafka in some form or the others such as Netflix, eBay, Adobe, Cisco and so on all these big data companies they uses the Kafka in some form of analysis and computations.

Refer slide time :(14:28)

Aggregating User Activity Using Kafka-Example

- Kafka can be used to aggregate user activity data such as clicks, navigation, and searches from different websites of an organization; such user activities can be sent to a real-time monitoring system and hadoop system for offline processing.



So, let us see some more Kafka example so aggregating users activity using the Kafka as an example. So, Kafka can be used to aggregate the user activity data such as clicks navigation searches from different website of an organization such user activities can be sent to a real-time monitoring system and Hadoop for. So, we can see here as an example of user activities aggregating user activities that means different users who are accessing through different portals will generate the stream of data and that will enter into the Kafka cluster and Kafka will after computation now it will give it to the consumers either for the real-time monitoring or it will store in the Hadoop offline file process.

Refer slide time :(15:21)

Kafka Data Model

The Kafka data model consists of messages and topics.

- Messages represent information such as, lines in a log file, a row of stock market data, or an error message from a system.
- Messages are grouped into categories called topics.
Example: LogMessage and Stock Message.
- The processes that publish messages into a topic in Kafka are known as producers.
- The processes that receive the messages from a topic in Kafka are known as consumers.
- The processes or servers within Kafka that process the messages are known as brokers.
- A Kafka cluster consists of a set of brokers that process the messages.

Data model
- messages (1)
- Topics (2)

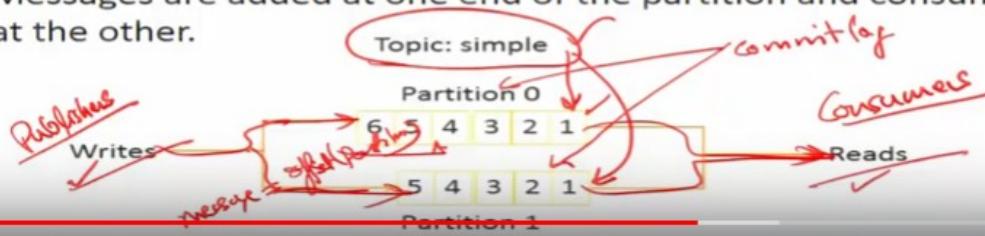


So, Kafka data model consists of the messages and the topics. So, messages represent the information such as lines in a log file rows of a stock market data or an error message from a system they are called ‘Messages’. So, messages are sometimes groups together into the into the topics. So, for example log message or a stock message together are grouped together and called as the ‘Topics’. So, the processes that publish messages into the into the topic in Kafka is known as the producers. And the processes that receive the message from the topic in the Kafka is known as the consumer the processes or the servers within the Kafka that processes the messages are known as the Kafka brokers in the Kafka cluster consists of set of brokers that processes the messages let us understand all these different components of Kafka through this particular picture. So, the so the group of messages is called a ‘Topic’. And, so here in this particular example we have shown two different groups of messages that is topic 1 and topic 2 these topics are or these messages are generated and those by the processes which we call it as producers. So, these particular topics are now processed using things which is called the ‘Brokers’. So, brokers process them and again these particular topics after the transformation will be passed on to the consumers. So, again let us understand that the data model in Kafka is either messages and the collection or the group of messages which is called the ‘Topics’. So, data model in Kafka comprises of messages and topics now these messages represents the information such as lines in a log file row of a stock market data or error messages. So, these are called messages these messages are grouped together and into the different categories and called the ‘Topics’. So, related messages are grouped together and they called as ‘Topics’ furthermore the processes that publishes the messages into the into a topic in the Kafka is known as the producers. So, producers publishes the that the messages into the into the topic. So, the processes those receive the message that receives the messages from the topics in the Kafka is known as the consumers. Now, the processes and the servers within the Kafka that processes the messages are known as the brokers. So, a kafka cluster consists of set of brokers that processes the messages. So, let us see what you mean by the the topic which is nothing but the data model,

Refer slide time :(19:13)

Topics (Data model in Kafka)

- A topic is a category of messages in Kafka.
- The producers publish the messages into topics.
- The consumers read the messages from topics.
- A topic is divided into one or more partitions.
- A partition is also known as a commit log.
- Each partition contains an ordered set of messages.
- Each message is identified by its offset in the partition.
- Messages are added at one end of the partition and consumed at the other.



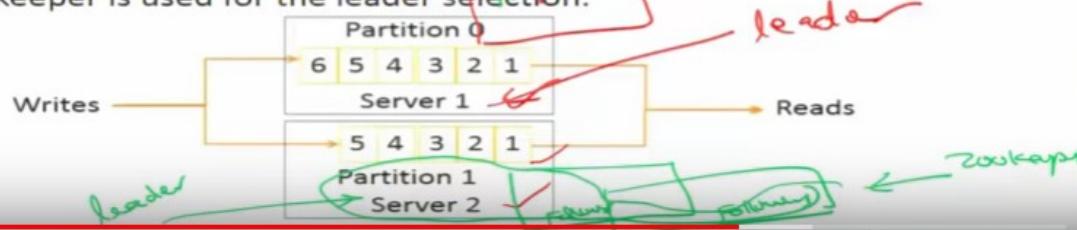
in Kafka so the topic is the category of the messages in Kafka. So, the producers published the messages into the topics and consumers who read the messages from the topics. So, the topic is divided into one or more partitions and partition is also known as the commit log. So, each partition contains an ordered set of messages. So, each message is identified by its offset in the partition and the messages are added at one end of the partition and consumed at the other end of the partition. Let, us understand the concept of topics and partition by this particularly simple example. So, here the the topics are nothing but the the publishers publishes the messages into the topics and once they do this then they have to write down. So, the publishers they will write they will do the write operations on the topic and the consumers will read the messages out of from the topic. So, these operations write is related to the publisher read operation is related to the consumer and the topic is divided into one or more partition for example this is the topic a simple, it is divided into two partitions partition 1 and partition 2 .So, each partition is known as the commit log. So, partition is nothing but a commit log. So, each partition contains an ordered set 0 of messages that we see the order in which the messages are received so, each message is identified by its offset in two. So, each message is identified by its offset in the in the partition message is nothing but an offset into the partition. So, messages are added at one end of the partition like here it will be added and consumed at the other end of the partition that is added with the help of using the write command and

consumed with the help of read command having understood the topics the partition the offset the write and read the publisher and consumer let us understand the partition in more details.

Refer slide time :(22:07)

Partition Distribution

- Partitions can be distributed across the Kafka cluster.
- Each Kafka server may handle one or more partitions.
- A partition can be replicated across several servers for fault-tolerance.
- One server is marked as a leader for the partition and the others are marked as followers.
- The leader controls the read and write for the partition, whereas, the followers replicate the data.
- If a leader fails, one of the followers automatically becomes the leader.
- Zookeeper is used for the leader selection.



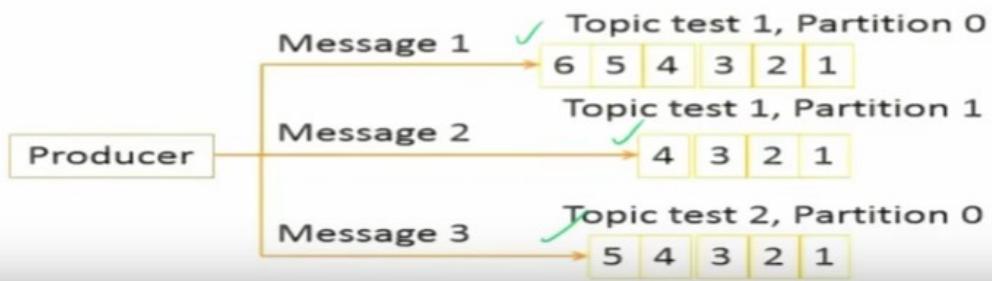
So, topics are divided into the partitions which are the unit of parallelism in Kafka. So, partitions allow the messages in a topic to be distributed allowing the messages in a topic to be distributed to multiple servers. So, a topic can have any number of partitions and each partition should fit in a single Kafka cluster. And, the number of partition decides the parallelism of that particular topic. So, partition distribution partition can be distributed across Kafka cluster and each Kafka server may handle one or more partition and here in this example which is shown that this server one will handle partition 0 and any other server will handle the partition 1 are both partition 0 and 1 can be handled by one server. So, each a partition, can be replicated across several servers for fault tolerance. So, one server is marked as the leader for the partition others are marked as the followers. So, reader controls the read and write operation for the partition whereas the followers which replicate the data. Now, if the leader fails one of the followers automatically becomes the leader zookeeper is used to do the leader selection process here to maintain. So, here which is shown as the server this is nothing but a leader and the same partitions are replicated in the other servers also and they are being replicated and called as the followers similarly this particular server is nothing but the leader and same partition is replicated on the other servers which are called 'Followers'. And, this particular leader election across different followers is done by the zookeeper this is called 'Partition Distribution'.

Refer slide time :(24:30)

Producers

The producer is the creator of the message in Kafka.

- The producers place the message to a particular topic.
- The producers also decide which partition to place the message into.
- Topics should already exist before a message is placed by the producer.
- Messages are added at one end of the partition.



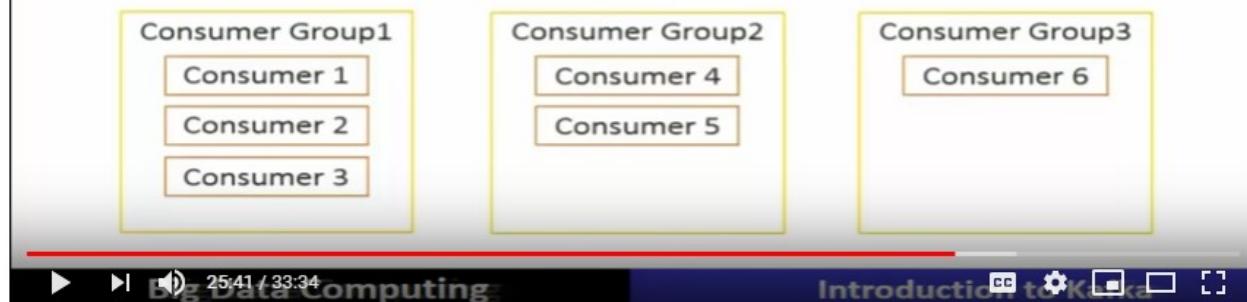
Now, let us see the producers. So, producer is the creator of message in Kafka. So, the producer plays the message into a particular topic so producers also decide which partition to place the message into it. And, the topic should already exist before a message is placed by the producer messages are added at one end of the topic one end of the partition. So, this is the example which is shown here that the producer is producing the messages and different messages are now joining the different topics depending upon the topic which are to be partitioned. So, here the producer will add the messages to different topics.

Refer slide time :(25:21)

Consumers

The consumer is the receiver of the message in Kafka.

- Each consumer belongs to a consumer group.
- A consumer group may have one or more consumers.
- The consumers specify what topics they want to listen to.
- A message is sent to all the consumers in a consumer group.
- The consumer groups are used to control the messaging system.



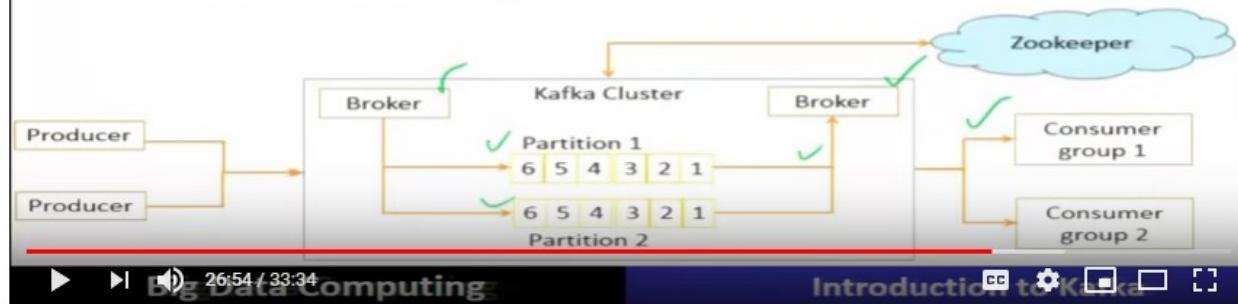
And, consumer is the receiver of the message in the in the Kafka. So, each consumer belongs to a consumer group of consumer group a how one or more consumers. So, consumers specify what topic they want to listen and the message is sent to all the consumers in a particular consumer group consumer groups are used to control the messaging system.

Refer slide time :(25:41)

Kafka Architecture

Kafka architecture consists of brokers that take messages from the producers and add to a partition of a topic. Brokers provide the messages to the consumers from the partitions.

- A topic is divided into multiple partitions.
- The messages are added to the partitions at one end and consumed in the same order.
- Each partition acts as a message queue.
- Consumers are divided into consumer groups.

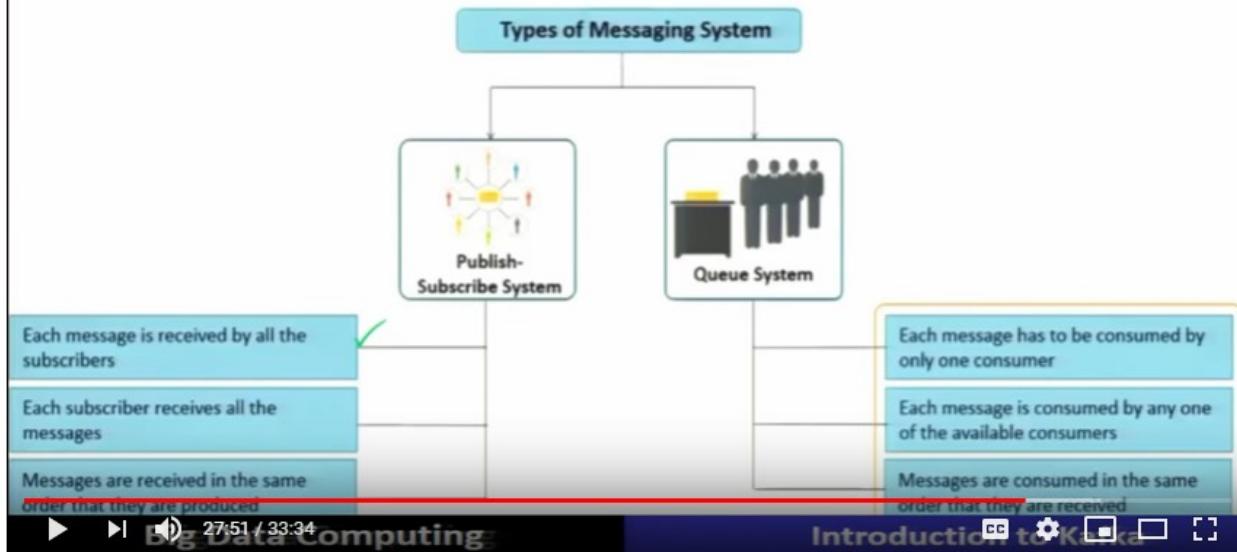


Now, let us see all these components together and see the overall Kafka architecture. So, Kafka architecture comprises of the brokers that take the messages from the producer and add to the partition of a topic broker provides the messages to the consumer from the partition. So, a topic is divided into multiple partitions. So, the message is added to the partition at one end and consumed at the other end that we have already seen and that is being coordinated by or that is handled by the brokers. So, each partition will act as the message queue and the consumers are divided into the consumer groups here in this scenario which is shown over here. And, all the brokers together are called the 'Kafka Clusters'. And, the different server machines are used with the help of zookeeper in this scenario which is shown.

Refer slide time :(26:59)

Types of Messaging Systems

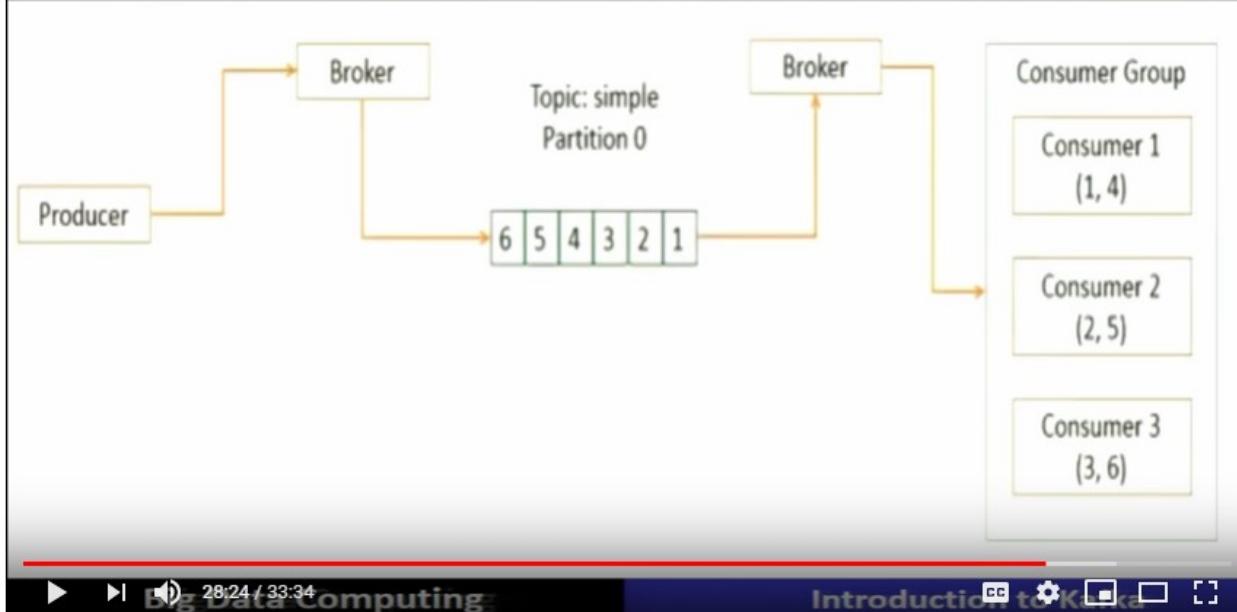
Kafka architecture supports the publish-subscribe and queue system.



Now, different type of messaging systems are available as on date. So, Kafka architecture supports publish subscribe and queuing system. So, publish subscribe system on the left is shown that each message is received by all the subscribers that is are the consumers. So, each subscriber receives all the messages and the messages are received in the same order that are in the same order that they are being produced by in the publish subscribe system. Whereas, in the queuing system each message is consumed by only one consumer and each message is consumed by any one of the available consumers and messages are consumed in the order in which they are removed. So, here there is a difference between queuing and publish subscribe system both the models of messaging system is supported in the Kafka architecture.

Refer slide time :(27:56)

Example: Queue System

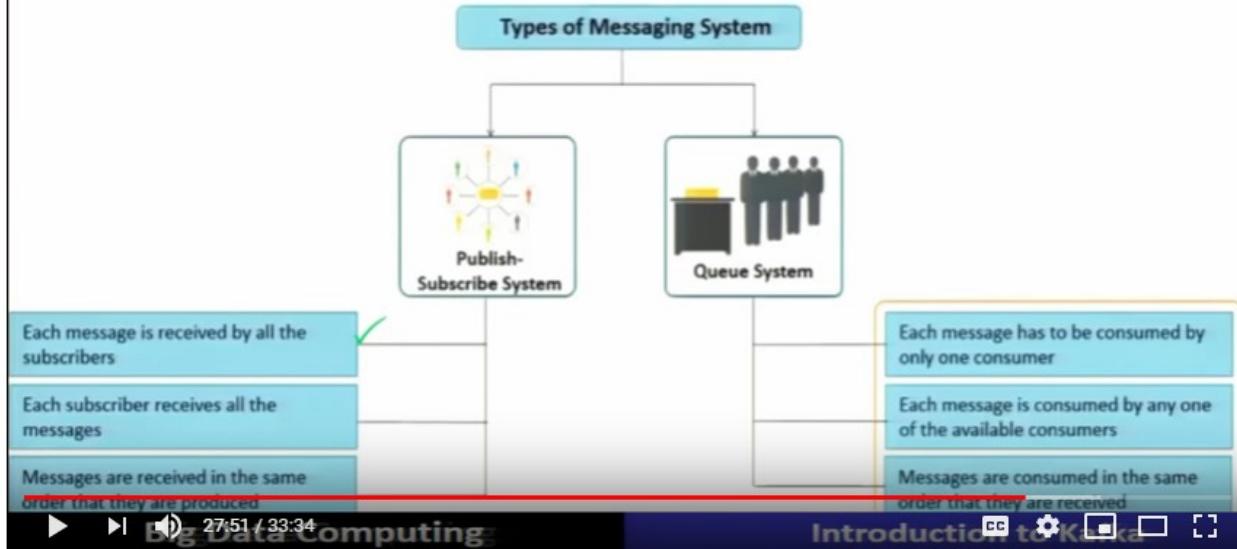


Now, let us see the Queen queuing queue system which is supported by the Kafka. So, here the producer will produce the message and this will be put into the partition or it will be divided into the topic with the help of the broker and a particular consumer will consume this particular message in this particular system.

Refer slide time :(28:24)

Types of Messaging Systems

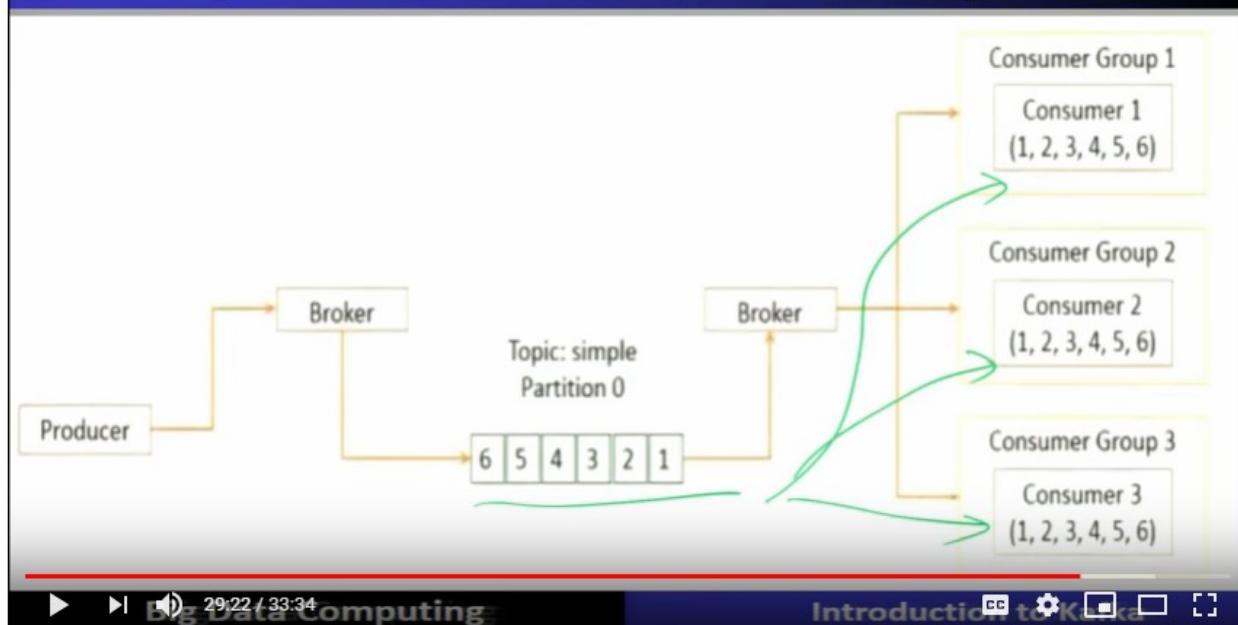
Kafka architecture supports the publish-subscribe and queue system.



So, we have to see that each message has to be consumed by only one consumer and each message is consumed by any one of the available consumers. So, and the messages are consumed in the same order that they are being received.

Refer slide time :(28:35)

Example: Publish-Subscribe System



So, here you can see that consumer number one is available. So, it will consume my message 1 and 4 similarly consumer number 2 is now when it is available it will consume 2 and 5 and when consumer 3 is available it will consume 3 and 6. So, any of these consumers will consume and every message is consumed by only one consumer. So, that is called the 'Queuing System'. And, it is being supported over here whereas in publish subscribe system every message is received by all these consumers. So, you see here in this publish subscribe system so this message 1 2 3 4 5 6 is being consumed in the same order by all the consumer groups.

Refer slide time :(29:27)

Brokers

Brokers are the Kafka processes that process the messages in Kafka.

- Each machine in the cluster can run one broker.
- They coordinate among each other using Zookeeper.
- One broker acts as a leader for a partition and handles the delivery and persistence, whereas, the others act as followers.

Now, as far as the brokers are concerned brokers are the Kafka processes that process the messages in Kafka each message each machine in the Kafka in the cluster can run one broker. So, they coordinate among each other using zookeeper one broker is act as the leader for the partition and handles the delivery and persistence whereas others are act as the followers.

Refer slide time :(29:50)

Kafka Guarantees

- Kafka guarantees the following:
 1. **Messages sent by a producer to a topic and a partition are appended in the same order**
 2. **A consumer instance gets the messages in the same order as they are produced.**
 3. **A topic with replication factor N, tolerates upto N-1 server failures.**

So, Kafka guarantees the following first is the message is sent by the producer to a particular topic and partitions are appended in the same order. Second, guarantees is the consumer instance gets the message in the same order that they are being produced. Third, guarantees is that a topic with the replication factor and all rates up to n minus one failures.

Refer slide time :(30:15)

Persistence in Kafka

Kafka uses the Linux file system for persistence of messages

- Persistence ensures no messages are lost.
- Kafka relies on the file system page cache for fast reads and writes.
- All the data is immediately written to a file in file system.
- Messages are grouped as message sets for more efficient writes.
- Message sets can be compressed to reduce network bandwidth.
- A standardized binary message format is used among producers, brokers, and consumers to minimize data modification.

Now, replication in Kafka. So, Kafka uses primary backup method for the replication that is one machine which is one replica is called the leader and chosen as the primary and the remaining machine are the replicas chosen as the followers and X as the backup. So, the leader propagates the right to the followers and the leader waits until all the rights are completed on the replicas if the replica is down it is skipped for the right until it comes back if the leader fails one of the followers will be chosen as the new leader and this mechanism can tolerate n minus one failures if the replicas of their application factor is n. Now, persistence in Kafka so Kafka uses Linux file system for persistence of the messages. So, persistence ensure no messages are lost Kafka relies on the file system page caching for fast reads and writes all the messages are immediately written to a file in the file system and messages are grouped as the messages set for the most efficient writes. So, messages sets can be compressed to reduce the network bandwidth a standard binary message format is used among the producers, brokers, consumers, to minimize the data modification.

Refer slide time :(31:32)

Apache Kafka: a Streaming Data Platform

- Apache Kafka is an open source streaming data platform (a new category of software!) with **3 major components**:
 1. **Kafka Core**: A **central hub** to **transport** and **store** event streams in real-time.
 2. **Kafka Connect**: A **framework** to **import** event streams from other source data systems into Kafka and **export** event streams from **Kafka** to destination data systems.
 3. **Kafka Streams**: A **Java library** to **process** event streams live as they occur.



So, again in a nutshell let us see how the streaming data platform is supported in Kafka. So, Apache capitals and open source streaming data platform and its suppose three with the three major components Kafka core is a central hub to transport and store the event streams in a real-time Kafka connect well now is the framework to import the event streams from other sources into the Kafka and export the event streams from Kafka to the destination data systems. Kafka stream is Java library to process event streams

Refer slide time :(32:07)

Further Learning

- **Kafka Streams code examples**
 - Apache Kafka <https://github.com/apache/kafka/tree/trunk/streams/examples/src/main/java/org/apache/kafka/streams/examples>
 - Confluent <https://github.com/confluentinc/examples/tree/master/kafka-streams>
- **Source Code** <https://github.com/apache/kafka/tree/trunk/streams>
- **Kafka Streams Java docs** <http://docs.confluent.io/current/streams/avadocs/index.html>
- **First book on Kafka Streams (MEAP)**
 - Kafka Streams in Action <https://www.manning.com/books/kafka-streams-in-action>
- **Kafka Streams download**
 - Apache Kafka <https://kafka.apache.org/downloads>
 - Confluent Platform <http://www.confluent.io/download>

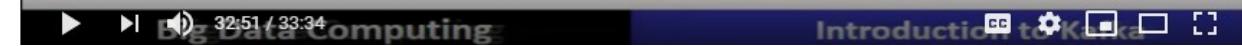


live as they occur iron for further reading this is given here the references, for Kafka streams code examples and Kafka stream java docs is on first book on Kafka streams. And, Kafka streams download is given at <https://kafka.apache.org/downloads>.

Refer slide time :(32:07)

Conclusion

- **Kafka is a high-performance, real-time messaging system.**
- **Kafka can be used as an external commit log for distributed systems.**
- **Kafka data model consists of messages and topics.**
- **Kafka architecture consists of brokers that take messages from the producers and add to a partition of a topics.**
- **Kafka architecture supports two types of messaging system called publish-subscribe and queue system.**
- **Brokers are the Kafka processes that process the messages in Kafka.**



Conclusion: Kafka is high-performance real-time messaging system. Kafka, can be used as an external commit law for distributed system Kafka data model consists of messages and topics Kafka architecture consists of brokers that take messages from producers. And, to add to the partition of a topic. Kafka, architecture supports two types of messaging system called ‘Publish Subscribe’. And, queuing system brokers are the Kafka processes that process the messages in the Kafka. Thank you.

Lecture – 23
Big Data Machine
Learning (Part-I)

Big data, Machine Learning.

Refer Slide Time :(0: 16)

Preface

Content of this Lecture:

- In this lecture, we will provide an overview of machine learning techniques to explore, analyze, and leverage data.
- We will also discuss tools and algorithms that you can use to create machine learning models that learn from data, and to scale those models up to big data problems.

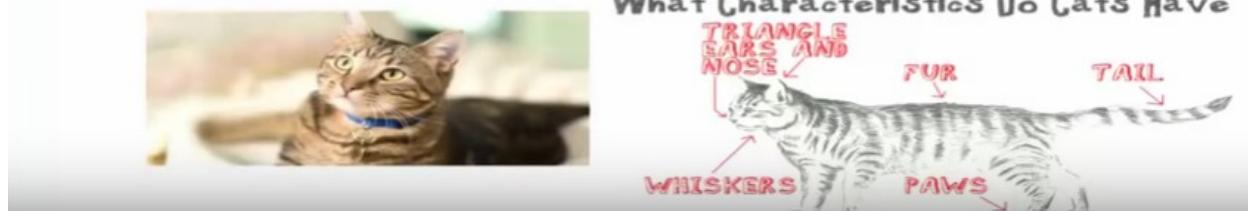
Preface content of this lecture: In this lecture we will provide an overview of machine learning techniques to explore, analyze and leverage big data processing. We will also discuss the tools and algorithm that you can use to create machine learning models and learn from big data and to scale these models up to the Big Data platforms.

Refer Slide Time :(0: 39)

What is Machine Learning?

- **Learning from data:**

- Machine learning is the field of study that focuses on computer systems that can learn from data. That is the system's often called models can learn to perform a specific task by analyzing lots of examples for a particular problem. For example, a machine learning model can learn to recognize an image of a cat by being shown lots and lots of images of cats.



Machine learning overview: What is a machine learning? Machine learning is the learning from the data. So, machine learning is the field of study that focuses on computer system that can learn from the data. That is the systems often called the model can learn to perform a specific task by analyzing the lot of examples for a particular problem. For example, machine learning model can learn to recognize an image of a cat by being shown a lots and lots of example and images of the cat for example cat can be characterized by from the examples that it has the as a triangle ears and nose it has a fur it has a tail, paws and whiskers. So, with these features you can identify characterize the cat.

Refer Slide Time :(1: 36)

What is Machine Learning ?

1. Learning from Data

2. **No explicit programming:** This notion of learning from data means that a machine learning model can learn a specific task without being explicitly programmed. In other words, the machine learning model is not given the step by step instructions on how to recognize the image of a cat.

- Instead, the model learns what features are important in determining whether it picture contains a cat from the data that has analyzed. Because the model learns to perform this task from data it's good to know that the amount and quality of data available for building the model are important factors in how well the model learns the task.

So, another feature of machine learning is. So, another aspect which identifies or which distinguish machine learning is that it does not have any programming. So, it has no explicit programming involved in problem Solving. So, this much notion of learning from the data means that machine learning models can learn a specific task without being explicitly programmed. In, other words the machine learning models is not given any step-by-step instruction on how to recognize the image of a of a cat. Instead, the modern learns with the features that are important in determining whether the picture contains the cat from the data that has well analyzed because the model learns to perform this task from the data it is good to know that the amount and quality of data available for building the model are important factors in how well the model learns the task therefore it does not have any explicit programming and we have also seen that it machine learning is the learning from from the data. So, there are two different aspects of a machine learning we have covered that is it learns from the data and second thing is it there is no explicit programming involved in Solving the problem using machine learning.

Refer Slide Time :(3: 11)

What is Machine Learning ?

• **Discovering hidden patterns:** Because machine learning models can learn from data that can be used to discover hidden patterns and trends in the data.

• **Data-driven decisions:** These patterns and trends lead to valuable insights into the data. Thus the use of machine learning allows for data driven decisions to be made for a particular problem.

- So to summarize, the field of machine learning focuses on the study and construction of computer systems that can learn from data without being explicitly programmed. Machine learning algorithms and techniques are used to build models, to discover hidden patterns and trends in the data allowing for data-driven decisions to be made.

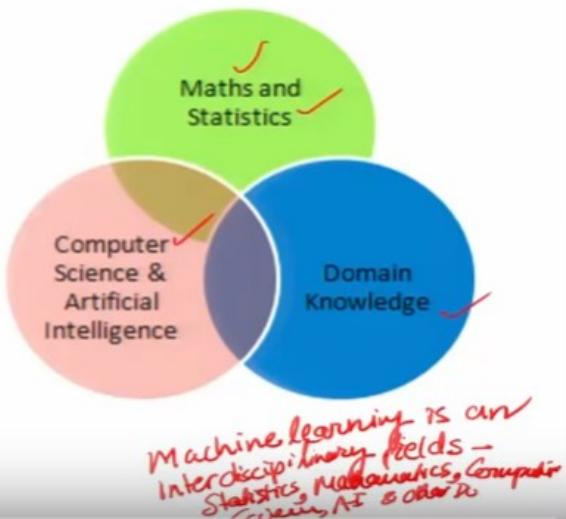
And, third important characteristics of the machine learning is about discovering the hidden patterns. Now, because machine learning models can learn from the data that can be used to discover hidden patterns and trends in the data which is hidden. So, the discovering hidden patterns is also an important aspects of the big data wise of the machine learning why because? it learns from the data. So, it can also discover the hidden patterns which are embedded into the data and also which and also captures the trend which is there inside the or within the data important characteristics which identifies or which characterizes the machine learning is about the data driven decisions. So, these patterns and the trends get to the valuable insight into the data thus the use of machine learning allows the data driven applications or decisions to be made for a particular problem. Thus, to summarize the field of machine learning focuses on the study and the construction of computer system that can learn from the data without being explicitly programmed machine learning algorithm techniques are used to build models to discover hidden pattern and to identify the trends in the data allowing for data driven decisions to be made.

Refer Slide Time :(4: 47)

Machine Learning (ML) is an Interdisciplinary Field

In applying machine learning to a problem, domain knowledge is essential to the success of end results. By domain knowledge we mean an understanding of the application or business domain.

Knowledge about the application, the data related to the application, and how the outcomes will be used are crucial to driving the process of building the machine learning model. So domain knowledge is also an integral part of a machine learning solution.

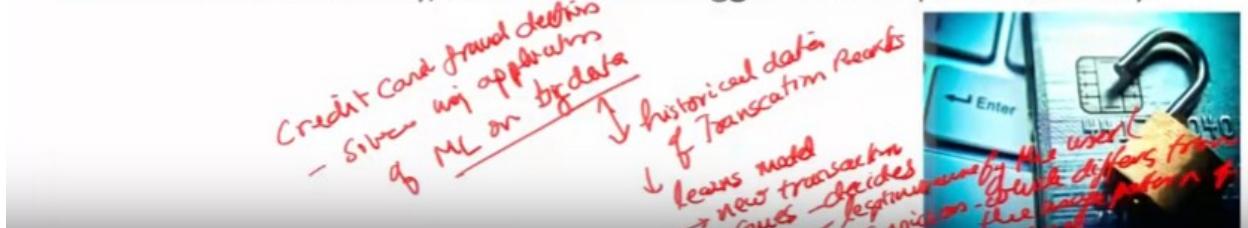


So, machine learning is an interdisciplinary field. So, in applying machine learning to a problem the domain knowledge is essential to success of the end results by domain knowledge we mean an understanding of the application or the business domain. In which we are solving the problem. So, the knowledge about the application the data related application and how the outcomes will be used are crucial in driving the process of building machine learning model. So, domain knowledge is also an integral part of a machine learning solution also we can see that the machine learning is an interdisciplinary field that is it requires the understanding and the background of the roots of machine learning draws from applying the statistics and also the computer science AI these are computer science AI statistics had also the domain knowledge is the intersection of this or its an interdisciplinary field of all three together defines the machine learning.

Refer Slide Time :(6: 49)

Example Application of Machine Learning

- **Credit card fraud detection:** One application of machine learning that you likely used this past weekend, or even just today, is credit card fraud detection. Every time you use your credit card, the current purchase is analyzed against your history of credit card transactions to determine if the current purchase is a legitimate transaction or a potentially fraudulent one. If the purchase is very different from your past purchases, such as for a big ticket item in a category that you had never shown an interest in or when the point of sales location is from another country, then it will be flagged as a suspicious activity.

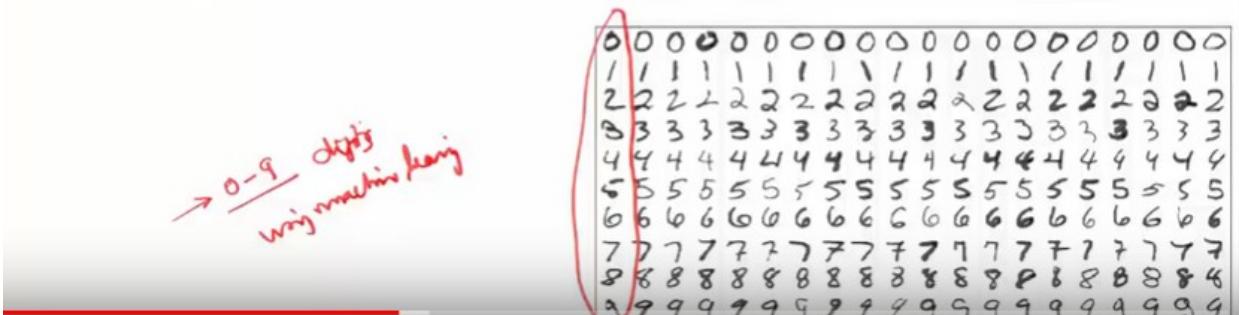


So, the example application of machine learning that means where the machine learning can be applied is in the credit card fraud detection that is one of the machine learning application is that you are likely to use the in the weekend or even today is the credit card fraud detection. Every, time you use the credit card the current purchase is analyzed against your history of the car transaction to determine whether if the current purchase is legitimate of the previous history transactions or potentially a fraudulent one if in this process the credit card fraud can be detected by applying the machine learning on this particular problem. So, if it is did if it is if the purchase is very different from your past purchases that learned from the machine learning algorithm using using the analysis of the data such as for the big ticket item in the category that you had never shown an interest in or when the point of sales is from another country then it will be flagged as the suspicious activity therefore such applications like credit card fraud detection can be solved using using an application of machine learning on the big data. So, the big data is the historical data of the transaction records. So, that means from this particular historical data of transaction records the machine learning learns the model and whenever a new whenever a new transaction comes then it decides whether it is a legitimate use by the user or it is suspicious that one which differs from the usage pattern of that user.

Refer Slide Time :(9: 57)

Example Application of Machine Learning

- **Handwritten digit recognition:** When you deposit a handwritten check into an ATM, a machine learning process is used to read the numbers written on the check to determine the amount of the deposit. Handwritten digits are trickier to decipher than typed digits due to the many variations in people's handwriting.



Now, another example is about handwriting recognition. So, handwritten digit recognition can be easily done with the help of machine learning. So, when you deposit a handwritten check in an ATM the machine learning process is used to read the numbers written on the check to determine the amount of the deposit handwritten digits are trickier to be deciphered than typed digit due to the many variations in the people's handwriting. So, so this handwritten. So, handwritten characters can be identified between 0 to 9 the digits using machine learning techniques. So, this handwritten digit recognition is another such application of a machine learning.

Refer Slide Time :(10: 56)

Example Application of Machine Learning

- **Recommendations on websites:** After you buy an item on a website you will often get a list of related items. Often this will be displayed as customers who bought this item also bought these items, or you may also like.
- These related items have been associated with the item you purchased by a machine learning model, and are now being shown to you since you may also be interested in them.



Now, further examples of machine learning is about recommendations on the website. So, after you buy an item on a website you will often get a list of related items. Often, this will be displayed as the customers who bought this item also bought these items. You may also like that these related items have been associated with them with the item you purchased by the machine learning model and now being shown to you since you are also interested to buy in them for example when you purchase an item let us say as hoodie then you are also displayed there recommended items maybe based on your past purchases or the other customer who purchased this item along with this item what are the other items? They are purchased based on that these recommendations are being made. So, the specific item and its associated items are being identified by the machine learning and is to be recommended on the website for online purchases of the related items and these recommendations are given to the customers. So, such kind of recommendations on the website is also an application of the machine learning.

Refer Slide Time :(12: 27)

More Applications of Machine Learning

- Targeted ads on mobile apps
- Sentiment analysis
- Climate monitoring
- Crime pattern detection
- Drug effectiveness analysis

Further, applications of machine learning include the targeted advertisements on the mobile apps, sentiment analysis, climate monitoring, crime pattern detection, drug effectiveness analysis.

Refer Slide Time :(12: 41)

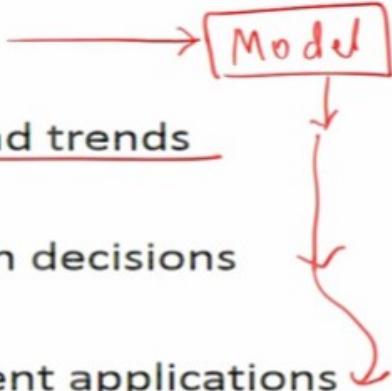
What's in a Name of Machine Learning?

- **Machine learning:** Machine learning has its roots since statistics, artificial intelligence, and computer science among other fields. Machine learning encompasses the algorithms and techniques used to learn from data. ✓
- **Data mining:** The term data mining became popular around the time that the use databases became common place. So data mining was used to refer to activities related to finding patterns in databases and data warehouses. There are some practical data management aspects to data mining related to accessing data from databases. But the process of finding patterns in data is similar, and can use the same algorithms and techniques as machine learning.
- **Predictive analytics** refers to analyzing data in order to predict future outcomes. ✓ This term is usually used in the business context to describe activities such as sales forecasting or predicting the purchasing behavior of a customer.
- **Data science** is a new term that is used to describe processing and analyzing data to extract meaning. Again machine learning techniques can also be used here. Because the term data science became popular at the same time that big data began appearing, data science usually refers to extracting meaning from big data and so includes approaches for collecting, storing and managing big data. *M.L. can be applied in Data Science, Predictive Analytics, ... so the field of Data Science is extended M.L.*

Now, let us decipher what is there in the name behind the machine learning and other such terms which are related terms which are available. So, machine learning is machine learning has its roots since the statistics artificial intelligence and computer science among other fields together they have contributed and the machine learning field. So, machine learning and encompasses algorithms are techniques that is used to learn from the data. Now, a related field which is called ‘Data Mining’. The, data mining became popular around the time when the databases were in common use at databases and data warehouses were at the main streams in the business operations. So, at that time this the data mining are of great importance and there are and there are other practical data management aspect to the data mining related to accessing the data from the databases. But, the process of finding pattern in the data is similar and similar to the machine learning and can be used and can and therefore the same algorithms which are there in data mining can also be used as some of the algorithmic techniques in the machine learning that we will see. So, therefore data mining is the field which was around the databases and data warehouses and the therefore there in that term there in that technique the algorithms and techniques are common therefore some of these techniques from data mining and algorithms are also used in the machine learning. So, data mining is differs from the machine learning in the sense that it is it was focusing around the databases and data warehouses, data retrieval and finding the patterns within that particular databases and data management. Now, the another term which is called ‘Predictive Analytics’. So, predictive analytics is the term which is used in the business houses. So, predictive analysis refers to analyzing the data in order to predict the future outcomes this term is usually used in the business context to determine to describe the activities such as sales forecasting or predicting the purchase behavior of the customer. This, particular predictive analytics of analyzing the data also uses the machine learning. But, it is the the it is the term which is used to predict the future outcomes and data mining is to find the patterns in the databases and data warehouses and machine learning is about the algorithms a technique used to learn from the data furthermore the data science is also related terms. So, data science is a new term that is used to describe the processing and analyzing the data, to extract the meaning out of it again the machine learning techniques can also be used here in the data science because the term data science became popular at the same time the big data began appearing. So, data science usually refers to extracting the meaning from the big data and so includes the approaches for collection, storing and managing the big data. So, therefore data science and also uses the techniques of machine learning to extract the meaning out of the data. Therefore, all these terms related to the which are coined around the machine learning we have just summarized again we can say that the machine learning is applied in data science and in predictive analytics whereas the algorithms and techniques which are used in machine learning can be drawn from the data mining. So, all are or they are all related terms. So, we can say that machine learning can be applied in data science, predictive analytics whereas the algorithms and techniques of data mining is used in machine learning.

Refer Slide Time :(17: 49)

Machine Learning Models

- Learn from data
 - Discover patterns and trends
 - Allow for data-driven decisions
 - Used in many different applications
- 

So, let us see the machine learning what do you mean by machine learning models? So, when it runs learns from the data and discovers the patterns and trends it allows also it allows the data-driven decisions to be used in many applications. So, how to learn from the data after learning from the data it forms the algorithm which is called ‘Model’. So, we will see what do you mean by the model. So, it learns from the data which we can also say that the model fits on that data. So, once the model is identified then when a new data is given it will discover the patterns and the trends and this way it allows the data-driven decisions to be made and this particular way that means learning from the data or this particular technique is very useful, in many different applications that we have seen in the previous slides.

Refer Slide Time :(18: 58)

Categories Of Machine Learning Techniques

- Classification
- Regression
- Cluster Analysis
- Association Analysis

Now, categories of machine learning techniques. So, machine learning techniques can be categorized, into four different types. So, first type is called the ‘Classification’. So, second type is called ‘Regression’, ‘Cluster Analysis’ and ‘Association Analysis’. Let, us understand these categories of machine learning techniques and then we will go in more detail of the different algorithms which falls into these different categories.

Refer Slide Time :(19: 31)

Classification *of ML*

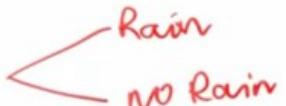
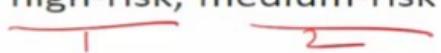
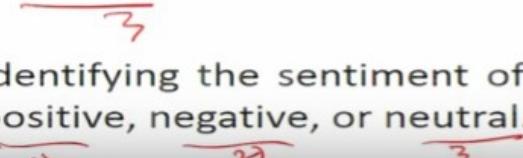
- **Goal:** Predict category
- In classification, the goal is to predict the category of the input data. An example of this is predicting the weather as being sunny, rainy, windy, or cloudy.



So, categories of machine learning is the first one is called ‘Classification Technique’ of ‘Machine Learning’. So, here in this classification the goal is to protect is the goal is to predict the category. So, in classification the goal is to predict the category of category of the input data for example an example of this classification is to predict the weather as it is sunny, rainy, windy, or cloudy. So, the weather forecasts can be. So, weather condition can be a sunny or it can be a foggy, it can be a windy or it can be stormy. So, this kind of categorization that is the weather which can be categorized into sunny, windy, foggy and stormy is to that means given this weather input you can classify into different categories then it is called ‘Classification Problem’.

Refer Slide Time :(20: 42)

Classification

- Some more examples of classification are classifying a tumor from a medical image as being benign or malignant.
- Predicting whether it will rain the next day. 
- Determining if a loan application is high-risk, medium-risk or low-risk. 
- Identifying the sentiment of a tweet or review as being positive, negative, or neutral. 

Some, other examples of classification a reclassifying a tumor from the medical image as being benign or malignant. So, predicting whether it is whether it will rain the next day is also a classification problem. So, that means it is predicting whether it will rain or it will not rain or no rain. So, this becomes a binary classification problem. So, determining if the loan application is a high risk, medium risk and the low risk. So, if the loan application can be classified into three different categories one of these three different categories then it is also a classification problem. Now, identifying the sentiments of a tweet or the review as being positive, negative and neutral is also is also a classification problem. So, given a tweet and we can if we can identify the sentiments whether it is the positive, negative or neutral then this again can be classified or this can also be a classification problem of machine learning.

Refer Slide Time :(22:00)

Regression *of ML*

- When your model has to predict a numeric value instead of a category, then the task becomes a regression problem.
- An example of regression is to predict the price of a stock. The stock price is a numeric value, not a category. So this is a regression task instead of a classification task.
- If you were to predict whether the stock price will rise or fall, then that would be a classification problem. But if you're predicting the actual price of the stock, then that is a regression problem.
- That is the main difference between classification and regression. In classification, you're predicting a category and in regression, you're predicting a numeric value.

Classification - predicting Category
Regression - predicting value



Now, let us see the regression problem of machine learning. So, when your model has to predict the numerical value like in the previous slide we have seen that it is predicting the category. But, now if it is predicting a numerical value instead of category then the task becomes the regression problem. So, for example an example of regression is to predict the price of a stock. So, the stock price is a numerical value not a category. So, this is a regression task instead of the classification task. So, if you are predicting whether the stock price will rise or fall that means it will classify as rise or fall then it would be a classification problem. But, if you are finding out the price of a stock or to predict the price of a stock then it becomes a regression problem. But, if you are predicting the actual price of that stock then it is regression. So, that we have already understood. So, that is the main difference between classification and regression in classification you are predicting a category and in regression you are predicting a value a numeric value. So, in regression we are predicting a numeric value whereas in classification problem we are predicting category.

Refer Slide Time :(24: 06)

Regression Examples

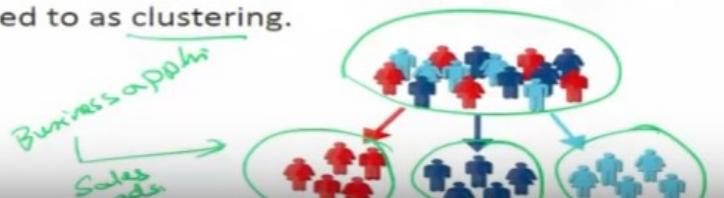
- Estimating the demand of a product based on time or season of the year.
- Predicting a score on a test.
- Determining the likelihood of how effective a drug will be for a particular patient.
- Predicting the amount of rain for a region.

Now, further examples of regression problems are estimating the demand of a product based on the time or the season of a year. I, am predicting a score of a test determining the likelihood of how effective a drug will be for a particular patient. I, am predicting the amount of rain for a particular region.

Refer Slide Time :(24: 26)

Cluster Analysis & ML.

- In cluster analysis, the goal is to organize similar items in your data set into groups. A very common application of cluster analysis is referred to as customer segmentation. This means that you're separating your customer base into different groups or segments based on customer types.
- For example it would be very beneficial to segment your customers into seniors, adults and teenagers. These groups have different likes and dislikes and have different purchasing behaviors. By segmenting your customers to different groups you can more effectively provide marketing adds targeted for each groups particular interests. Note that cluster analysis is also referred to as clustering.



Now, the third category of machine learning is called ‘Cluster Analysis’. So, let us see about in cluster analysis the goal is to organize similar items in your data set into the groups. A, very common application of cluster analysis is referred to as the customer segmentation. This, means that you are separating your customer base into different groups and segments based on customer types. For example, it would be very beneficial to segment your customers into seniors, adults and teenagers. These, groups have different likes and dislikes and have different purchasing behaviors. By, segmenting your customers to different groups you can target more advertisements and to this particular group of group having common interest. Note, that the cluster analysis also referred as the clustering. So, in this example we have shown that if it is a group of people we can segment them or we can cluster them into different groups of having the common interest and these groups can further be used for various business applications to target them for the sales advertisements having the common interest and so on.

Refer Slide Time :(25: 26)

Cluster Analysis Examples

- Some other examples of cluster analysis are:
- Identifying areas of similar topography, such as desert region, grassy areas, mountains etc.
- Categorizing different types of tissues from medical images. Determining different groups of weather patterns, such as snowy, dry, monsoon and
- Discovering hot spots for different types of crime from police reports.

*- Grouping together
similar data items
are called Cluster Analysis
clustering*

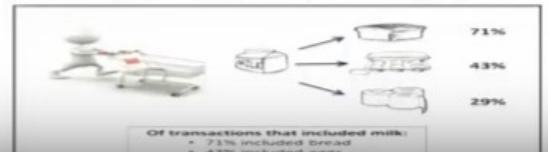
Other, examples of the cluster analysis includes somehow are: identifying the areas of similar topography such as desert region and grassy areas and mountain regions and categorizing, different type of tissues from the medical images, reminding different groups of weather pattern, such as snowy, dry and monsoon and also discovering the hotspot for various type of crime from the police records. Therefore, this

classification this grouping together the common or similar data items is called ‘Clustering’. And, this cluster analysis is also called as a ‘Clustering’.

Refer Slide Time :(27: 10)

Association Analysis

- The goal in association analysis is to come up with a set of rules to capture associations between items or events. The rules are used to determine when items or events occur together.
- A common application of association analysis is known as market basket analysis. Which is used to understand customer purchasing behavior.
- For example, association analysis can reveal that banking customers who have CDs, or Certificates of Deposits, also tend to be interested in other investment vehicles such as money market accounts.
- This information can be used for cross selling. If you advertise money market accounts to your customers with CDs they are likely to open such an account.



Now, another category of machine learning is called the ‘Association Analysis’ of Machine Learning. So, in association analysis the goal is to come up with a set of rules to capture association between the items or the events. These, rules are used to determine whether the items or events occur together. So, a common application of association analysis is known as market basket analysis which is used to understand the customer purchasing behavior. So, for example association analysis can reveal the banking customers who have CDs or certificates deposits and also trending to be interested in investment in other in other investment instruments such as money market accounts and so on. So, therefore the information this association analysis inside information can be used for cross selling. So, if you advertise money market account to your customers and likely open they will likely open such an instrument that is already seen in these applications.

Refer Slide Time :(28: 27)

Association Analysis Examples

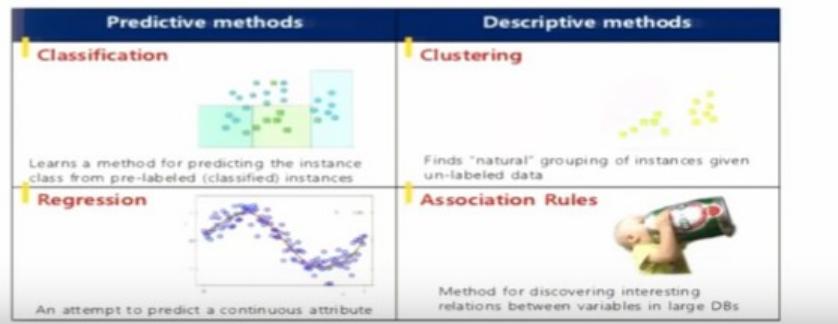
- Some other applications of association analysis are recommending similar items based on the purchasing behavior or browsing histories of customers.
- Finding items that are often purchased together, such as garden hose and potting soil, and offer sales on these related items at the same time to drive sales of both items.
- Identifying web pages that are often accessed together so that you can more efficiently offer up these related web pages at the same time.

So, association analysis other examples are such as recommending items which are similar items based on the purchasing behavior or the browsing histories of customers. Finding items that are often purchased together and identifying the web pages that are of an access together. So, that you can more efficiently offer these web pages at the same time.

Refer Slide Time :(28: 55)

Classification of Machine Learning Techniques

- So the different categories of machine learning techniques are classification, regression, cluster analysis, and association analysis.



So, the classification of machine learning technique we have different categories of machine learning techniques scene let us say that they are classification, regression, cluster analysis and association analysis. And, therefore depending upon different problems we have to apply one of these different categories to solve the problem in the machine learning landscape for example for predictive methods we can apply the classification or regression depending upon whether the prediction of a category it is there or a prediction of a value similarly when the descriptive methods are to be applied then we have to choose either between the clustering or between the association rules.

Refer Slide Time :(29: 50)

Supervised vs. Unsupervised

- **In supervised approaches** the target, which is what the model is predicting, is provided. This is referred to as having labeled data because the target is labeled for every sample that you have in your data set.
- Referring back to our example of predicting a weather category of sunny, windy, rainy or cloudy, every sample in the data set is labeled as being one of these four categories. So the data is labeled and predicting the weather categories is a supervised task. In general, classification and regression are supervised approaches.

Now, furthermore the machine learning is also categorized as supervised and unsupervised. So, in supervised approaches the target which is what the model is predicting is being provided. So, this is referred to as having the data with a with a label and because the target is labeled for every sample that you have given in the dataset. So, referring back to our example of predicting the weather category of sunny, windy, rainy and cloudy every sample of the weather data is labeled as being one of these four categories. So, the data is labeled by the domain expert and predicting the weather categories is also called the ‘Supervised Task’. So, in general the classification regression both are the supervisor activity and approaches.

Refer Slide Time :(30: 48)

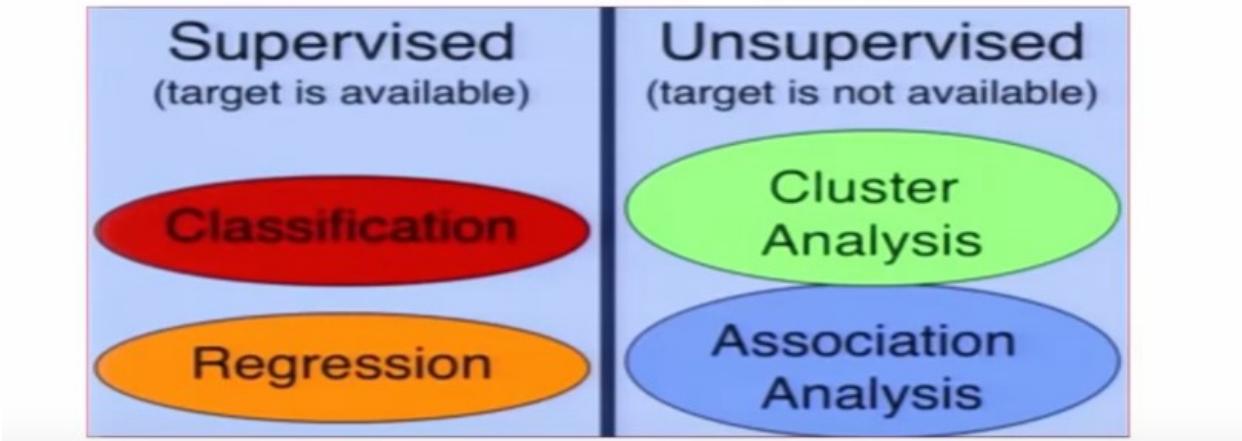
Supervised vs. Unsupervised

- In **unsupervised approaches** on the other hand, the target that the model is predicting is unknown or unavailable. This means that you have unlabeled data.
- Remember the cluster analysis example of segmenting customers into different groups. The samples in your data are not labeled with the correct group. Instead, the segmentation is performed using a clustering technique to group items based on characteristics that they have in common.
- Thus, the data is unlabeled and the task of grouping customers into different segments is an unsupervised one. In general, cluster analysis and association analysis are unsupervised approaches.

In, unsupervised approach on the other hand the target that the model is predicting is unknown or unavailable. This, means that you have unlabeled data. Now, remember the cluster analysis example of segmenting customers into the different groups of common of having similarities. So, the sample in your data are not labeled with the correct group illustrate the segmentation is performed using the clustering techniques to group items based on the similarities or it is also called as a characteristics. That, they have in common thus the data is unlabeled on the task of grouping customers into different segments is called ‘Unsupervised’. In, general the cluster analysis and association analysis are unsupervised approaches.

Refer Slide Time :(30: 41)

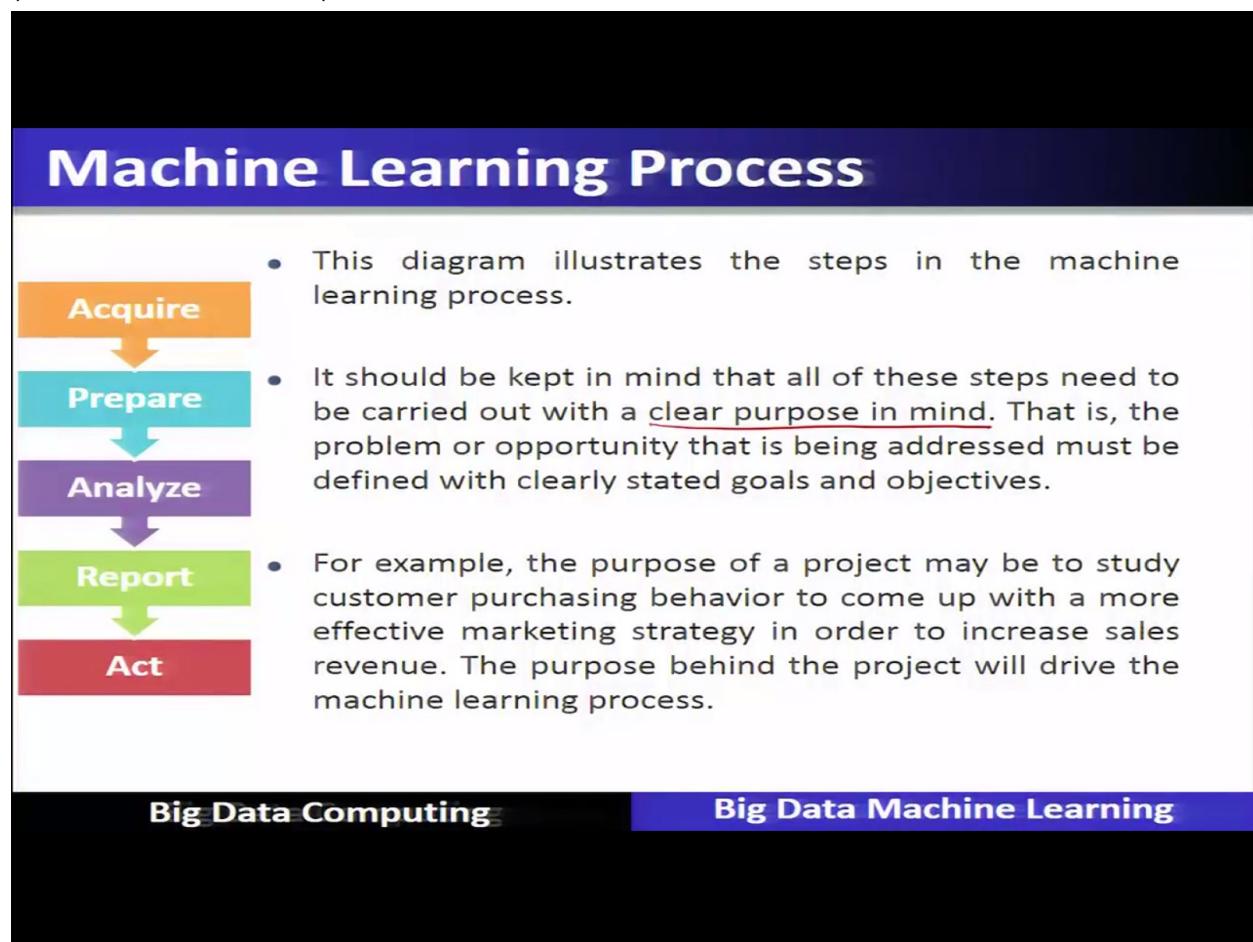
Classification of Machine Learning Techniques



So, in short we can classify all four techniques of a machine learning that is classification, regression as the supervised machine learning technique whereas cluster analysis and Association analysis unsupervised machine learning technique depending upon whether the target is available with that data set or not if the target is available then it is called ‘Supervised Learning’. If, the target or label is not available then it is called ‘Unsupervised Learning’. So, cluster analysis and association analysis there the target, is not available with the data set and hence it is called ‘Unsupervised’. Various, in classification and regression the target or the label is there with the data set hence this activity of applying the dataset is supervised machine learning technique.

Machine learning process. Machine learning process this diagram explains the steps in the machine learning process which has Acquire, Prepare, Analyze, Report, and Act. So, it should be kept in mind that all these steps are needed to be carried out with a clear purpose in mind that is the problem or the opportunity that is being addressed must be defined with a clear stated goals and objectives. For example, the purpose of a project may be to study the customer purchasing behavior. Now, to come up with more effective marketing strategy in order to increase the sales so this particular aspect must be there at all the steps. So, the purpose behind the project will drive the machine learning process.

(Refer Slide Time: 01:08)



Now, the first step here is called acquire the data. So, the first step in the machine learning process is to get all the available data related to the problem at hand. Hence, we need to identify all data sources, collect the data and finally integrate the data from these multiple sources for that particular problem. Therefore, the clear purpose or the clear objectives of a goal in a given problem must be there in the mind while in the step no. 1 to acquire the data. This, step is also called data acquisition. So, data acquisition is the process of gathering, filtering and cleaning the data before it is being put in the data warehouse or any other storage solution on which the data analysis can be carried out. So, data acquisition is one of the major big data challenges.

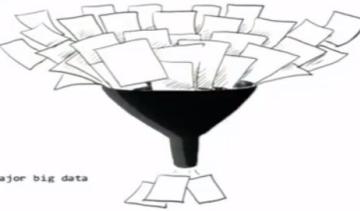
(Refer Slide Time: 2:05)

Step 1: Acquire Data

- The first step in the machine learning process is to get all available data related to the problem at hand. Here, we need to identify all data sources, collect the data, and finally integrate data from these multiple sources.

What is Data Acquisition

It is the process of gathering, filtering and cleaning the data before it is put in a data warehouse or any other storage solution on which data analysis can be carried out.



Data acquisition is one of the major big data challenges

Big Data Computing

Big Data Machine Learning

Step no. 2 in this particular machine learning process is to prepare the data. This, step is further divided into two parts to explore the data and preprocess the data. Let, us understand these two aspects in more detail. Now, the first part of the data preparation involves the preliminary exploration of the data to understand the nature of the data that we have to work with. This, is also called data exploration. That, is the things we want to understand about the data are its characteristics, formats, quality, as per as now these a good understanding of the data will lead to a more informed analysis and will become a successful outcome. So, to have the better successful outcome of the machine learning result it is necessary that very careful data exploration steps has to be performed. So, once we know about data through the exploratory analysis the next part is the preprocessing of the data for the analysis that is called preprocess. Now, this pre processing includes the cleaning data, selecting the variables to use and transforming the data to make the data more suitable for the analysis in the next step. (Refer Slide Time: 3:58)

Step 2: Prepare Data

- This step is further divided into two parts, explore data and pre-process data.

Data Exploration
The first part of data preparation involves preliminary exploration of the data to understand the nature of the data that we have to work with. Things we want to understand about the data are its characteristics, format, and quality. A good understanding of the data leads to a more informed analysis and a more successful outcome.

- Once we know more about the data through exploratory analysis, the next part is pre-processing of the data for analysis. This includes cleaning data, selecting the variables to use, and transforming data to make the data more suitable for analysis in the next step.



Big Data Computing

Big Data Machine Learning

Third, step is to analyze the data. Now, the prepared data then would be boxed on to the analysis step. This, involves selecting the analytical technique to use building a model using the data and assessing the result. Meaning, to say that to analyze the data we require to select one of these techniques analytical techniques to be used. This, is one of the important decision here in this step which is to analyze the data. So, techniques involves again all the machine learning techniques which we have which we have seen that is we have to identify which of these different categories or classes of machine learning is to be applied to analyze the problem that is whether it is a classification or it is a regression or it is association or it is clustering. So, all these techniques we have to identify and within each technique different algorithms are there that is required to be selection that required to be selected. So, this becomes the selection of analytical technique to be used in building the model that means we have to fit the model in the data using these selected techniques. So, this is one of the most important step of a machine learning.

(Refer Slide Time: 05:59)

Step 3: Analyze Data ✓

- The prepared data then would be passed on to the analysis step. This step involves selecting the analytical techniques to use, building a model using the data, and assessing the results.

ML techniques
- classification
- regression
- clustering
- association

Selection

fit model in data w/ selected techniques



Big Data Computing

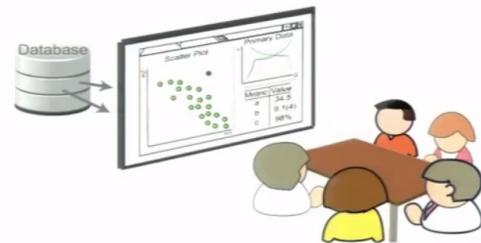
Big Data Machine Learning

Fourth, data is to communicate results this include evaluation evaluating the results with respect to the goal sets for the project. So, presenting the result is in an easy to understand way and communicating the result to others is also one of the important aspect.

(Refer Slide Time: 06:16)

Step 4: Communicate Results

- This includes evaluating the results with respect to the goal set for the project. Presenting the results in a easy to understand way and communicating the results to others.



Big Data Computing

Big Data Machine Learning

Step 5, is to apply the result. So, the last step is to apply the result this is to bring us back to the purpose of the project how can the insight from our analysis be used to provide effective marketing to increase the sell revenue and so on. So, this is also going to be very useful in using the result for different application.

(Refer Slide Time: 06:44)

Step 5: Apply the Results

- The last step is to apply the results. This brings us back to the purpose of the project. How can the insights from our analysis be used to provide effective marketing to increase sales revenue?
- Determining actions from insights gained from analysis is the main focus of the act step.



Big Data Computing

Big Data Machine Learning

So, determining the actions for insight gain from the analysis is the main focus of this step. Now, we have to see that all these steps are iterative in the nature that means either all these steps or some of the steps they have to be repeated. So, that we will see further that that machine learning process is very iterative and finding from results from 1 step may require a previous step to be repeated with the new information. So, for example during the preparation step we may find some data quality issues that may require us to move back and to acquire further information to address some issues with the data collection or to get additional data that we did not include in the first round and so on. So, therefore this iteration is required in the machine learning process.

(Refer Slide Time: 07:35)

Iterative Process

- Note that the machine learning process is a very iterative one. Findings from one step may require a previous step to be repeated with new information.
- For example, during the prepare step, we may find some data quality issues that may require us to go back to the acquire step to address some issues with data collection or to get additional data that we didn't include in the first go around.

Big Data Computing

Big Data Machine Learning

Now, goals and activities in the machine learning process. So, the goals and activities in the machine learning process we will describe the goals from each step and the key activities performed in these steps.

(Refer Slide Time: 07:55)

Goals and Activities in the Machine Learning Process

- Here we will describe the goals of each step and the key activities performed in each step.



Big Data Computing

Big Data Machine Learning

So, to in the first step that is data acquisition or to acquire the data the first step this is used also in the in the data science to acquire the data. The, goal of this system is to identify and obtain all the data related to the problem at hand. First, we need to identify all related data sources. Keep, in mind that data can come from difference sources such as files, databases, internet, mobile devices and so on. So, remember to include all the data related to the problem that you are addressing. So, after you have identified your data and data sources the next step is to collect the data and integrate the data from different sources. This, may require conversion as the data can come from different with different formats so it is required to convert the format which is required to be use in your machine learning process. So, this may require conversion as the data can come from different sources. And, it may also require to align the data as the data from different sources may have different timing or special resolution. So, once you have collected and integrated your data you now have to make this data core and for your analysis.

(Refer Slide Time: 09:30)

Acquire Data

- The first step in the data science process is to acquire the data. The goal of the step is to identify and obtain all data related to the problem at hand. First, we need to identify all related data and the sources. Keep in mind, that data can come from different sources such as files, databases, the internet, mobile devices. So remember to include all data related to the problem you are addressing.

Data Sources related to problem at hand

- After you've identified your data and data sources, the next step is to collect the data and integrate data from the different sources. This may require conversion, as data can come in different formats. And it may also require aligning the data, as data from different sources may have different time or spatial resolutions. Once you've collected and integrated your data, you now have a coherent data set for your analysis.

Big Data Computing

Big Data Machine Learning

Second, step is to prepare the data the next step after acquiring the data is to prepare it to make it suitable for the analysis. There, are two parts to this step to explore the data and pre-process that we have already discussed.

(Refer Slide Time: 09:42)

Prepare Data

- The next step after acquiring data is to prepare it to make it suitable for analysis.
- There are two parts to this step, explore data and preprocess data.

Big Data Computing

Big Data Machine Learning

So, data exploration you want to do some preliminary investigation in order to gain the better understanding of specific characteristics of your data. This, in turn will guide the rest of the process. With, data exploration you will want to look the things like correlations, general trends, outliers etc. Correlations, provide the information about the relationship between the variables in your data trends in your data will reveal if the variable is moving in a certain direction such as transaction volume is increasing through the year. Outliers, indicate the potential problems with the data or it may indicate an interesting data point that needs to be further examination. Without, this exploration activity you will not be able to use your data effectively. So, the data exploration requires preliminary investigation which needs which involves the steps like to how to gain the better understanding of the data and by that process you will find out the correlation, general trend, and outliers. So, correlations will provide the information about the relationship between different variables in your data and also it will reveal the trends if the data if the variable is moving in certain directions such as the transaction volumes increasing throughout throughout the year that is it will identify the trends with the data. And, third thing is very important which will be processed in the data exploration part is called outliers. So, by this data exploration you can identify the outliers. And, this outliers have to be dealt with at this stage that is in the data data exploration stage.

(Refer Slide Time: 11:44)

Describe your Data

- One way to explore your data is to calculate summary statistics to numerically describe the data.
- Summary statistics are quantities that capture various characteristics of a set of values with a single number, or a small set of numbers. Some basic summary statistics that you should compute for your data set are mean, median, mode, range and standard deviation. Mean and median are measures of the location of a set of values. Mode is the value that occurs most frequently in your data set, and range and standard deviation are measures of spread in your data. Looking at these measures will give you an idea of the nature of your data. They can tell you if there's something wrong with your data.
- For example, if the range of the values for age in your data includes negative numbers, or a number much greater than a hundred, there's something suspicious in the data that needs to be examined

Big Data Computing

Big Data Machine Learning

Next, part is to describe your data. Once, you explore your data to calculate the summary statistics to numerically describe the data now this summary statistics are the are the quantities that capture various characteristics of the severe of values with a single number or a set of small numbers. Some, basic summary statistics that should compute your data set are mean, median, mode range, range and standard deviation. They, are the summary statistics of your data. So, mean and median will measure are the measures of location of the set of values. So, mean location is for example if this is the data this is the mean so it will give the central Centrality, location of the data, location of the set of values. Similarly, mode is the values that occurs most most frequently in your data set. And, the range and the standard deviation are the measures of the spread of your data. Looking, at these measures you will you will get an idea of the nature of your data and they will tell you that if your that if there is something wrong in your data set then you have to basically refine your data at this stage. For, example if the range of values for the age in your data set includes a negative number then obviously it is an outlier or the number is greater than a 100 in the age then it is something suspicious in the data and that needs to be corrected upon.

(Refer Slide Time: 13:28)

Describe your Data

- One way to explore your data is to calculate summary statistics to numerically describe the data.
- Summary statistics are quantities that capture various characteristics of a set of values with a single number, or a small set of numbers. Some basic summary statistics that you should compute for your data set are mean, median, mode, range and standard deviation.
✓ Mean and median are measures of the location of a set of values. Mode is the value that occurs most frequently in your data set, and range and standard deviation are measures of spread in your data. Looking at these measures will give you an idea of the nature of your data. They can tell you if there's something wrong with your data.
- For example, if the range of the values for age in your data includes negative numbers, or a number much greater than a hundred, there's something suspicious in the data that needs to be examined

Big Data Computing

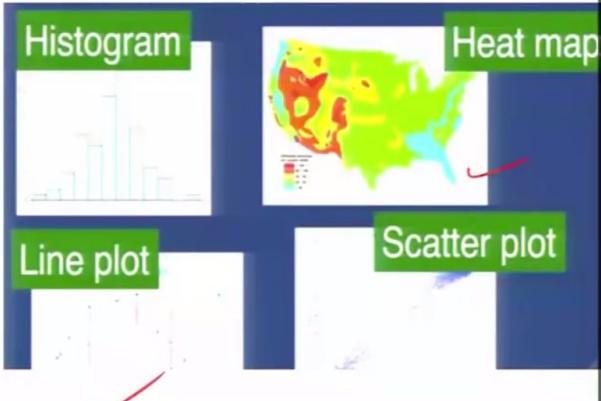
Big Data Machine Learning

Now, the next step is called the visualization visualize your data. Visualization techniques, also provide the quick and effective ways to explore your data. For, example the use of histogram plots will understand the distribution of data or the skewness or unusual dispersion and outliers of the data. So, line plot like the one in the lower plot shown over here can be used to look the trends in the data as the change of the prize on the stock. The, heat map also is one of the visualization technique of your data give an idea about where the hot spots are. Scatter plots, effectively show the correlation between the two variables and overall there are many other type of plots to visualize your data. They, are very useful in helping to understand behavior of your data.

(Refer Slide Time: 14:33)

Visualize Your Data

- Visualization techniques also provide quick and effective ways to explore your data. Some examples are, a histogram, such as the plot shown here, shows the distribution of the data and can show skewness or unusual dispersion in outliers.
- A line plot, like the one in the lower left, can be used to look at trends in the data, such as, the change in the price of a stock. A heat map can give you an idea of where the hot spots are.
- A scatter plot effectively shows correlation between two variables. Overall, there are many types of plots to visualize data. They are very useful in helping you understand the data you have.



Big Data Computing

Big Data Machine Learning

Now, pre-processing step. So, after you have explored the data the next step is the pre-process the data to prepare for the analysis. The, goal here is to create the data that will be used for the analysis. The, main activities of this pre-processing is to clean the data, select the appropriate variables to use and transform the data as needed.

(Refer Slide Time: 14:53)

Step-2-B: Pre-Process

- The second part of the prepare step is preprocess. So, after we've explored the data, we need to preprocess the data to prepare it for analysis.
- The goal here is to create the data that will be used for analysis.
- The main activities on this part are to clean the data, select the appropriate variables to use and transform the data as needed.

Big Data Computing

Big Data Machine Learning

So, let us see what do you mean by cleaning the data or that data cleaning. A very important of data preparation is to clean the data to address many quality issues. Real world data, is nothing and there are many examples of a quality issues with the data from the real applications including missing values such as income in the survey duplicate data such as two different records of the same customers with different addresses. So, inconsistent or invalid data such as security zip code, noise in the collection of data that distorts the true values outlier such as numbers larger than 100 on someone's age is also an essential to detect and address these issues that can negatively affect the quality of the data.

(Refer Slide Time: 15:39)

Data Cleaning

- A very important part of data preparation is to clean the data to address quality issues. Real world data is nothing. There are many examples of quality issues with data from real applications including missing values, such as income in a survey, duplicate data, such as two different records for the same customer with different addresses.
- Inconsistent or invalid data, such as a six digit zip code. Noise in the collection of data that distorts the true values. Outliers, such as a number much larger than 100 for someone's age. It is essential to detect and address these issues that can negatively affect the quality of the data.

Big Data Computing

Big Data Machine Learning

The next important part is about the feature selection. Feature selection, first choosing the set of features to use that is appropriate for the application. Feature selection, can involve removing redundant or irrelevant features combining features or creating the new features. So, during the exploration data step you may have discovered that two features are very correlated. In, that case one of these features can be removed without negatively affecting the analysis result. For example, the purchase price of a product and the amount of sales tax are very likely to be correlated similarly the age and the weight of a person are highly correlated. So, eliminating the sales tax would then be beneficial. So, removing redundant or irrelevant features will make the subsequent analysis more simpler and more accurate. So, you may also want to combine the features to create the new ones. For example, adding the applicants educational level as a feature to a loan approval application would make more sense and there are also algorithm to automatically determine the most relevant feature that based on various mathematical properties.

(Refer Slide Time: 16:55)

Feature Transformation

- Feature transformation maps the data from one format to another. Various transformation operations exist. For example, scaling maps the data values to a specified range to prevent any one feature from dominating the analysis results.
- Filtering or aggregation can be used to reduce noise and variability in the data.
- Dimensionality reduction maps the data to a smaller subset of dimensions to simplify the subsequent analysis. We will discuss techniques to prepare data in more detail later in this course.

Big Data Computing

Big Data Machine Learning

Then, feature transformation. Feature transformation, maps the data from one format to another format various transformation operations exist for example scaling map the data values to a specific range to prevent any feature from dominating the analysis result. This, is very important to be handled. So, filtering or aggregation can be used to reduce the noise and variability in the data. Dimensionality reduction, maps the data to a smaller subset of dimensions to simply file the subsequent analysis. We, will discuss these techniques how to prepare the data using future transformation in further slides.

(Refer Slide Time: 17:36)

Feature Transformation

- Feature transformation maps the data from one format to another. Various transformation operations exist. For example, scaling maps the data values to a specified range to prevent any one feature from dominating the analysis results.
- Filtering or aggregation can be used to reduce noise and variability in the data.
- Dimensionality reduction maps the data to a smaller subset of dimensions to simplify the subsequent analysis. We will discuss techniques to prepare data in more detail later in this course.

Big Data Computing

Big Data Machine Learning

Now, another step is to analyze the data. So, after preparing the data to address address the quality issues and pre-process it to get in appropriate format the next step would be to use that particular data for the machine learning process to analyze. The, goal of this the goal of this step is to build the machine learning model to analyze the data and to evaluate the results that you get from the model. So, the analysis step starts with this determining the type of problem you have you began selecting the appropriate machine learning techniques. So, the selection of machine learning techniques is the important step of analyzing the data. Then, you construct the model using the data that is also called to fit the machine learning techniques with the data, fit the model on the data, fit the machine learning. So, that is called the construction of the a model with the data you have prepared. So, once the model is build you will want to apply this model to a new data sample to evaluate how well your model behaves. Thus, the data analysis involves selecting the appropriate techniques for your problem, building the model and evaluating the results.

(Refer Slide Time: 19:01)

Step-3: Analyze

- After preparing the data to address data quality issues and preprocess it to get it in the appropriate format, the next step in the machine learning process is to analyze the data. The goals of the staff are to build a machine learning model, to analyze the data and to evaluate the results that you get from the model.
- The analyze steps starts with this determining the type of problem you have. You begin by selecting appropriate machine learning techniques to analyze the data.
fit the ML with Data
- Then you construct the model using the data that you've prepared. Once the model is built, you will want to apply it to new data samples to evaluate how well the model performs. Thus data analysis involves selecting the appropriate technique for your problem, building the model, then evaluating the results.

Big Data Computing

Big Data Machine Learning

Step number four is about the report. The next step in the machine learning process is reporting the result from your analysis. In, reporting your result it is important to communicate the insight to make a case for that action to follow. In reporting your result, you will be think about what to present and how well to present? In deciding what to present, you consider what main results are what insights you have gained from your analysis and what added the values to these insight to the application. Keep in mind that, even the negative results are the value will learning and suggest further avenues or additional analysis. Remember, that all the findings must be presented so that it informs the decision makers for the further steps. So, in deciding how to present remember that the visualization is also an important tool in presenting your results. So, plots and summary statistics discussed earlier, explore in the data exploration can be used effectively here as well. You, should have the tables with the details from your analysis as backup if someone wants to take the deeper dive into the results for that purpose. So, in summary you want to report your findings by presenting your results and the value added with the graphs using the visualization tools.

(Refer Slide Time: 20:30)

Step-4: Report

- The next step in the machine learning process is reporting results from your analysis. In reporting your results, it is important to communicate your insights and make a case for what actions should follow.
- In reporting your results, you will want to think about what to present, as well as how to present. In deciding what to present, you should consider what the main results are, what insights were gained from your analysis, and what added value do these insights bring to the application.
- Keep in mind that even negative results are valuable lessons learned, and suggest further avenues for additional analysis. Remember that all findings must be presented so that informs decisions can be made for next steps.
- In deciding how to present, remember that visualization is an important tool in presenting your results.
- Plots and summary statistics discussing the explore step can be used effectively here as well. You should also have tables with details from your analysis as backup, if someone wants to take a deeper dive into the results.
- In summary, you want to report your findings by presenting your results and the value added with graphs using visualization tools.

Big Data Computing

Big Data Machine Learning

Fifth, step is to act. Final step, in the machine learning process is to determine what action should be taken based on the insights gained. So, what action should be taken based on the results of your analysis? Should you market certain products to specific customer segment to increase the sales? What inefficiency can be removed from your process? What incentive should be affective in attracting new customers? So, these are some of the important actions you can perform based on this analysis of the outcome. So, once the specific action has been determined the next step is to implement the action. And, things to consider here include how can the action be added to your application. How will this and users be affected?

Accessing assessing the impact of implementing actions is then necessary to evaluate the result benefit gained. So, the result of this assessment determines the next step which would suggest the additional analysis for further opportunities which can which would begin another cycle of machine learning process.

(Refer Slide Time: 21:42)

Step-5: Act

- The final step in the machine learning process is to determine what action should be taken based on the insights gained.
- What action should be taken based on the results of your analysis? Should you market certain products to a specific customer segment to increase sales? What inefficiency can be removed from your process? What incentives would be effective in attracting new customers?
- Once a specific action has been determined, the next step is to implement the action. Things to consider here include, how can the action be added to your application? How will end users be affected?
- Assessing the impact of the implemented action is then necessary to evaluate the benefits gained. The results of this assessment determine next steps, which could suggest additional analysis or further opportunities, which would begin another cycle of the machine learning process.

Big Data Computing

Big Data Machine Learning

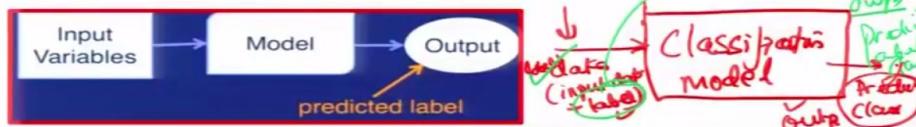
Generalization and overfitting. So, it is about the errors in the classification of machine learning. So, recall that the machine learning model maps the input it receives to an output. So, for a classification model the model's output is predicted class labels for the input variables and the true labels are the target labels. So, again let us see that in the classification model the model's output is the predicted class labels for the input variables and the true labels are the target labels meaning to say that if this is the classification model which is shown as the box whereas the input is the data which is basically having the input variable input data with label or it is also called as the label data and which is prepared in this format. So, this particular classification model will give the predicted class label which is also there with the data. So, it will give the predicted class. Now, this predicted class is an output label. Now, this label can be compared with the input label. So, that is what is mentioned over here. Let, me summarize this. So, the model's output is called the predicted label for the input variable. So, the model's output is called the predicted label class. So, this is the output of a model. For, the input variable this is the input variable and the true label is the target. So, the true label is already there. True label is known. So, the predicted label and the true label both are now compared. The, true label and the predicted output level if both are same so we have to now compare about these two different labels. So, then if the classifier predicts the correct classes label for the sample that is a success. And, if the predicted label is different from the true class label then there is an error. Now, we have to see these two different cases. So, the

error rate then is the percentage of the error made over the entire data sets. That, is the number of errors divided by the total number of samples in the data set. So, error rate is known as the misclassification rate or simply the error. So, error is the error rate by the classification model.

(Refer Slide Time: 24:40)

Errors in Classification

- Recall that a machine learning model maps the input it receives to an output. For a classification model, the model's output is the predicted class label for the input variables and the true class label is the target.
- Then if the classifier predicts the correct classes label for a sample, that is a success. If the predicted class label is different from the true class label, then that is an error.
- The error rate, then, is the percentage of errors made over the entire data set. That is, it is the number of errors divided by the total number of samples in a data set.
- Error rate is also known as misclassification rate, or simply error.



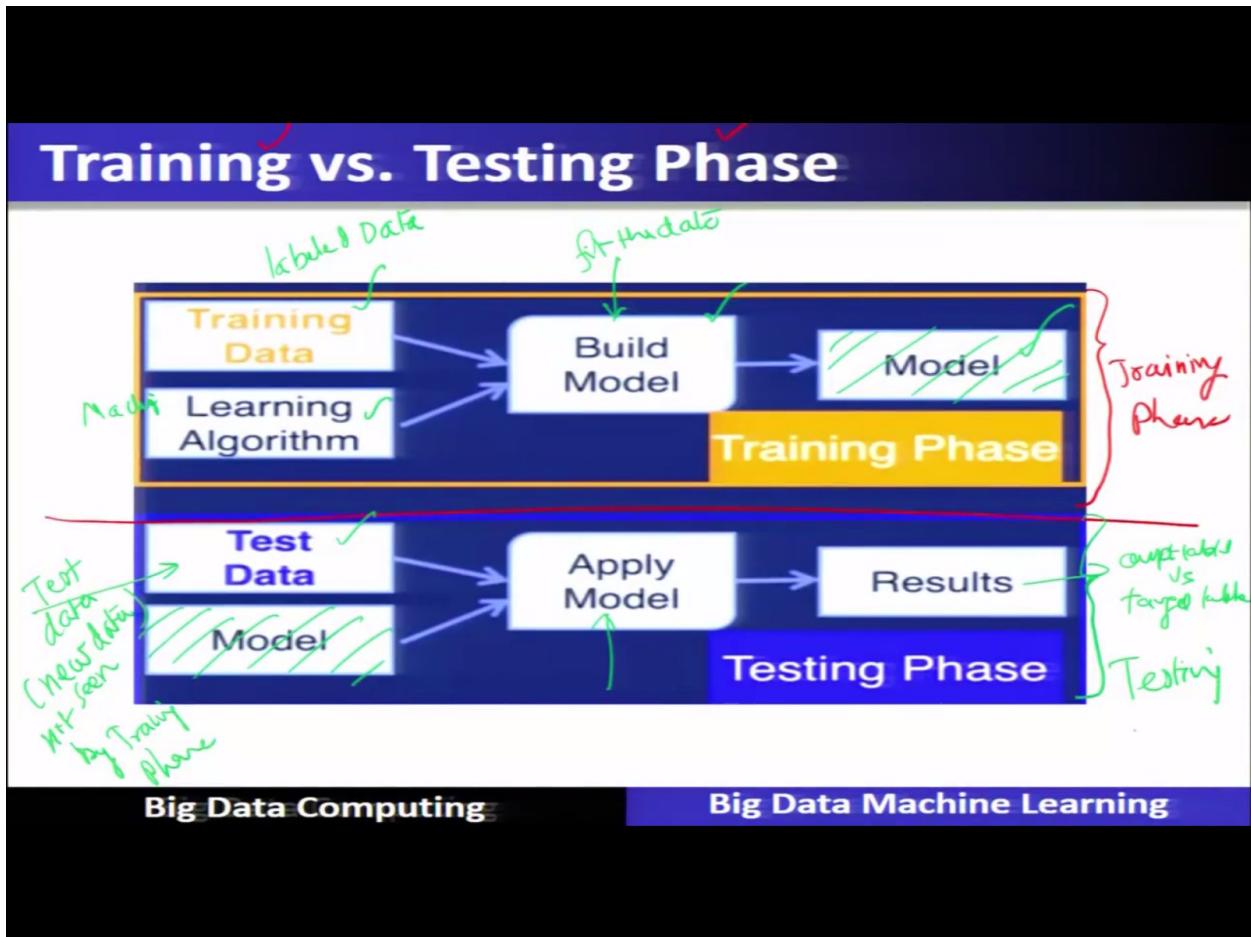
Big Data Computing

Big Data Machine Learning

So, let us discuss this machine learning stage that is the classification. Now, this classification stage is divided into the two parts one is called the training phase and the other is called testing phase. So, let us see here this is called the training phase and the training phase includes the training data which is the labeled data and also requires one machine learning algorithm. Using, this particular machine learning algorithm will fit on the data and therefore it will therefore it will build the model based on the the machine learning technique which is applied on the training data. So, once the model is build this particular model is now available for the evaluation of the training phase. This, is called training phase. Now, the other phase is called the testing phase. In testing phase, also we have the the test data and the model which is now being created. So, the model is the same which is created over the training phase. So, that particular model will now be given new data and this is called as the test data. It, is the new data which is not seen by the training phase. So, when a new data is given to the model now this model will do the predictions based on this test data and it will give the result that is the output label and it will be compared with the with the target label output label to be compared

with target label. So, this particular way we are going to do the testing. So, based on these particular analysis of these particular model we are going to see the activities in the testing phase.

(Refer Slide Time: 28:40)



So, therefore let us see more insight into the errors of errors and the classification. The, model is built using the training data and evaluated on the test data that we have seen in the previous slides. So, the training and test data are the two different data sets. The, goal in building the machine learning mold is to have the model perform well on the training as well as on the test data. So, the model which is which is created will perform well on the training as well as it will be also perform well in the test data. Now, error rate are simply the error on the training is refer to as the training error similarly the error on the test data is referred to as the test error. So, there are two types of errors. So, training error or the test error. Again, to understand we have to just you have to recall. What, we have done is there were two different phases one was called the training phase. Training phase, will take the training data and the machine learning algorithm and it will build a model this particular model will also give an error. Classification errors which are collected and it is called a training error. Now, this particular model is given back again is used again in the testing phase and we will have another data that is called test data. Test data, when it is given to the model now it will perform and it will give an error and this error is called testing error. So, there are two different type of errors one is called training

error and the other is called testing error. So, the error on the test data is an indication how well the classifier will perform on the new data.

(Refer Slide Time: 31:11)

Errors in Classification

- The model is built using training data and evaluated on test data. The training and test data are two different data sets. The goal in building a machine learning model is to have the model perform well on training, as well as test data.
- Error rate, or simply error, on the training data is referred to as training error, and the error on test data is referred to as test error. The error on the test data is an indication of how well the classifier will perform on new data.

Big Data Computing **Big Data Machine Learning**

Now generalization. This, is known as generalization. Generalization, refers to how well your model performs on a new data that is the data not used to train the model. Now, you want your model to generalize well to the new data. So, if you model generalizes well then it will perform well on the data set. That, is similar in the structure to the training data but does not contain exactly the same sample in the training data. Since, the test error indicates how well your model generalizes to the new data note that the test error is also called the generalization error.

(Refer Slide Time: 31:50)

Generalization

- This is known as generalization. Generalization refers to how well your model performs on new data, that is data not used to train the model.
- You want your model to generalize well to new data. If your model generalizes well, then it will perform well on data sets that are similar in structure to the training data, but doesn't contain exactly the same samples as in the training set.
- Since the test error indicates how well your model generalizes to new data, note that the test error is also called generalization error.

Big Data Computing

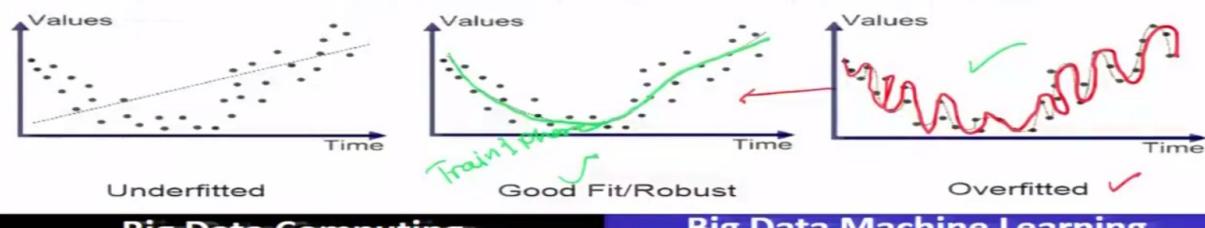
Big Data Machine Learning

Now, there is a concept which is called as overfitting. So, related concept to the Generalization is called overfitting. So, if your model has a very low training error but has the high generalization error then it is called the overfitting. Meaning, to say that during the training phase the error is less but during the testing phase the error is more hence it is called overfitting. This, means that the model has learned to model has learned to model the noise in the training data instead of learning the underlying structure of the data hence it is called overfitting. So, overfitting at these plots illustrates what happens when a model overfits. So, the training samples are shown as the points and the input to the input to output mapping that the model has learned is indicated as the curve. So, the plot on the left shows that the model has learned the underlying data structure as the curve follows the trend of the sample data point as well. So, here you can see that this curve has learned the the behavior of a data well. Hence, it is a good fit during the training phase. Now, this there is another example the plot on the right however shows that the model has learned to model the model has learned to the noise in the data set for example this is this curve is fitting to the to the noise and it has and this is different from this particular curve which has fit to the to the underlying structure of the data. So, this is an example of the overfitting and the left side graph which shows this particular is a good fit. So, the model which tries to capture every sample point instead of general trend of the samples together that is the training error and that is generalization error are plotted together during the model training.

(Refer Slide Time: 34:15)

Overfitting

- These plots illustrate what happens when a model overfits. Training samples are shown as points, and the input to output mapping that the model has learned is indicated as a curve. The plot on the left shows that the model has learned the underlying structure of the data, as the curve follows the trend of the sample point as well. The plot on the right, however, shows that the model has learned to model the noise in a data set.
- The model tries to capture every sample point, instead of the general trend of the samples together. The training error and the generalization error are plotted together, during model training.



Big Data Computing

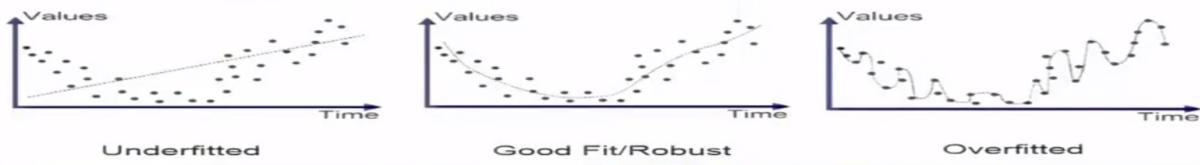
Big Data Machine Learning

A model that overfits will not generalize well to the new data. So, the model will do well on just the training data but not perform well on the new data set or rather it will perform poorly. So, a classifier that performs well on just the training data set will not be only useful. So, it is essential that the goal of the good generalization performance should be kept in mind before building the model. So, overfitting and underfitting. So, underfitting occurs when the model is fitted to the noise in the training data this result is this results in a low training error and high test error so that is called overfitting. Now, there is another thing which is called underfitting. Underfitting, on the other hand occurs when the model has not learned the structure of the data. This, results in high training error and high test error. When, both the errors are very high that is the training error is also very high and test error is very high then it is called underfitting. Now, both are undecidable overfitting and underfitting are undecidable. Since, both mean that the model will not generalize well to the new data. So, overfitting generally occurs when the model is too complex that is it has too many parameter relative to the number of training samples. So, to avoid overfitting the model needs to be kept as simple as possible and yet will solve the input output mapping for a given data set.

(Refer Slide Time: 35:50)

Overfitting and Underfitting

- Overfitting occurs when the model is fitting to the noise in the training data. This results in low training error and high test error.
- Underfitting on the other hand, occurs when the model has not learned the structure of the data. This results in high training error and high test error. ✓
- Both are undesirable, since both mean that the model will not generalize well to new data. Overfitting generally occurs when a model is too complex, that is, it has too many parameters relative to the number of training samples. So to avoid overfitting, the model needs to be kept as simple as possible, and yet still solve the input/output mapping for the given data set.



Big Data Computing

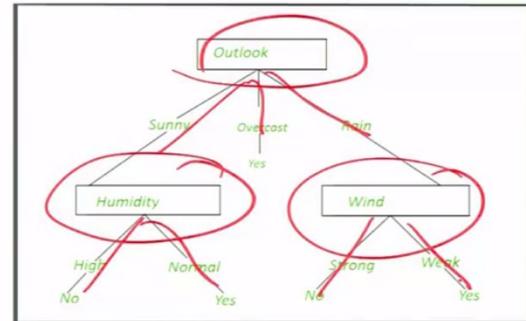
Big Data Machine Learning

Now, let us see what causes overfitting? In, summary overfitting is when your model has learned the noise in the training data instead of underlying structure of the data. So, you want to avoid overfitting so your model will generalize well to the new data. Now, let us discuss the overfitting in the decision trees. Now, decision tree induction that is the decision tree also referred as the tree induction the tree repeatedly splits the data in a node in order to get successfully paired subsets of data. So, this is here the note gets split into three cases. Here, this note splits into two. So, this particular process of splitting the note and growing the tree is called tree induction.

(Refer Slide Time: 36:40)

Decision Tree Induction

- A decision tree, also referred to as tree induction, the tree repeatedly splits the data in a node in order to get successively paired subsets of data.



Big Data Computing

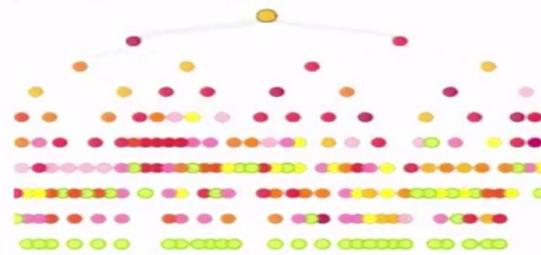
Big Data Machine Learning

Now, let us see the concept of overfitting in this decision trees. Note, that the decision tree classifier can potentially expand its notes until it can perfectly classify the samples in the training data. But, if the tree grows the nodes to fit the noise in the training data then it will not classify the model for a new sample. This, is because the tree has partitioned the input space according to the noise in the in the data instead of the true structure of the data in other words it overfits.

(Refer Slide Time: 37:15)

Overfitting in Decision Trees

- Note that a decision tree classifier can potentially expand its nodes until it can perfectly classify samples in the training data.
- But if the tree grows nodes to fit the noise in the training data, then it will not classify a new sample well.
- This is because the tree has partitioned the input space according to the noise in the data instead of to the true structure of a data. In other words, it has overfit.



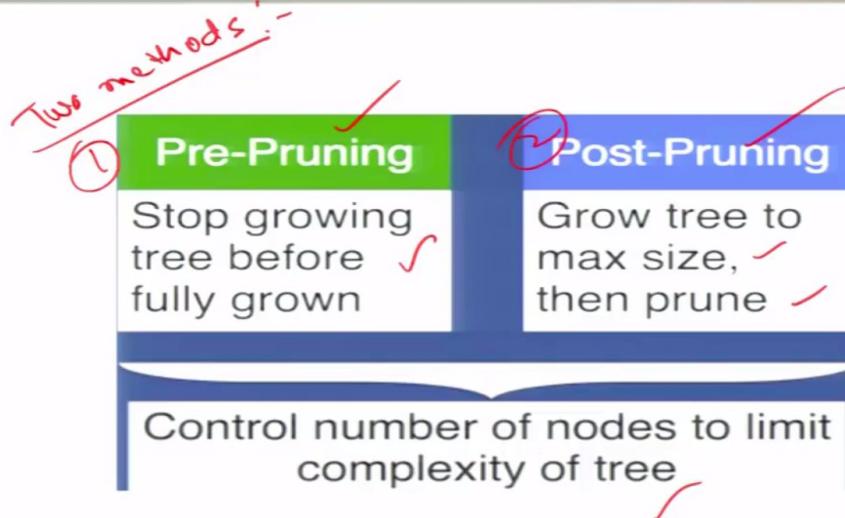
Big Data Computing

Big Data Machine Learning

So, how to avoid overfitting in the decision tree. So, there are two methods one is called pre-pruning the other method is called post-pruning. There, are two methods to avoid the overfitting in decision trees. The, first one is called pre-pruning the other one is called post-pruning. So, pre-pruning says that to stop growing the tree before it is fully grown why because in this particular case it is the data is so the tree is fully grown that means it is growing according to the noise in the data hence it has to be stopped before it is fully grown and so it will control the number of notes to limit the complexity of the tree. The, second method of avoiding the overfitting in the decision tree is called post-pruning. Post-pruning, means let the tree grow to its maximum size and then prune the tree so that it can reduce or it can overcome from the overfitting.

(Refer Slide Time: 38:23)

Avoiding Overfitting in Decision Trees



Big Data Computing

Big Data Machine Learning

Now, let us see the pre-pruning step. So, with the pre-pruning the idea is to stop the tree induction before fully grown tree is build that perfectly fits the training data. To, do this restrictive stopping conditions for growing tree must be used. For, example noise stops expanding if the number of samples in the node is less than some minimal threshold. Another example is to stop expanding the node if the improvement in the impurity measures also below a certain threshold.

(Refer Slide Time: 39:03)

Pre-pruning

- With pre-pruning, the idea is to stop tree induction before a fully grown tree is built that perfectly fits the training data.
-
- To do this, restrictive stopping conditions for growing nodes must be used. For example, a node stops expanding if the number of samples in the node is less than some minimum threshold.
- Another example is to stop expanding a node if the improvement in the impurity measure falls below a certain threshold.

Big Data Computing

Big Data Machine Learning

Post-pruning, in the post-pruning the tree is grown to a maximum size and then the tree is pruned by removing the nodes using the bottom up approach that is the tree is trimmed starting with the leaf nodes. The pruning, is done by replacing the subtree with a leaf node. If, this improves the generalization error or if there is no change to the generalization error with this replacement.

(Refer Slide Time: 39:25)

Post-pruning

- In post-pruning, the tree is grown to its maximum size, then the tree is pruned by removing nodes using a bottom up approach.
- That is, the tree is trimmed starting with the leaf nodes. The pruning is done by replacing a subtree with a leaf node if this improves the generalization error, or if there is no change to the generalization error with this replacement.

Big Data Computing

Big Data Machine Learning

Now, overfitting in the decision tree in other words if removing the sub trees does not have the negative effect on generalization error then the nodes in that subtree only add to the complexity of the tree and not to its overall performance. So, those notes showed be removed in practice post-pruning tends to give better results. This, is because pruning decisions are based on the information from the full tree. Pre-pruning, on the other hand may stop the tree growing process prematurely however post-pruning is more computationally expensive since the tree has to be expanded on its full size.

(Refer Slide Time: 40:05)

Overfitting in Decision Trees

- In other words, if removing a subtree does not have a negative effect on the generalization error, then the nodes in that subtree only add to the complexity of the tree, and not to its overall performance.
- So those nodes should be removed. In practice, post-pruning tends to give better results. This is because pruning decisions are based on information from the full tree. Pre-pruning, on the other hand, may stop the tree growing process prematurely. However, post-pruning is more computationally expensive since the tree has to be expanded to its full size.

Big Data Computing

Big Data Machine Learning

Let, us see how the validation set is used for these purposes. So, now we have again looking back to the same diagram that is the training phase and the testing phase and this particular process. So, how to avoid the overfitting and recall that the model that over fit does not generalize well to the new data recall also that overfitting generalizes generally occurs when the data is too complex. And, how to determine when it should occur.

(Refer Slide Time: 40:44)

Avoiding Overfitting

- Recall that a model that overfits does not generalize well to new data.
- Recall also that overfitting generally occurs when a model is too complex.
- So to have a model with good generalization performance, model training has to stop before the model gets too complex.
- How do you determine when this should occur?

Big Data Computing

Big Data Machine Learning

So, there is a technique which is called using the validation set. A validation set, can be used to guide the training process to avoid the overfitting and deliver the good generalization performance. Now, we have discussed having the training set and a test set. So, the training set is used to build the model and a test set is used to see how the model performs in a new data. So, the training data is now further divided into two different sets one is called the training data and the other is called validation data. So, training data is not fully used for the training purpose but it is divided into the validation data. Let, us see what do you mean by the validation data which is taken out from the training data. So, this particular validation set is used to guide the training process to avoid the overfitting and deliver the good generalization how that is all done we will see.

(Refer Slide Time: 41:41)

Validation Set

- A validation set can be used to guide the training process to avoid overfitting and deliver good generalization performance.
- We have discussed having a training set and a separate test set. The training set is used to build a model and the test set is used to see how the model performs a new data.



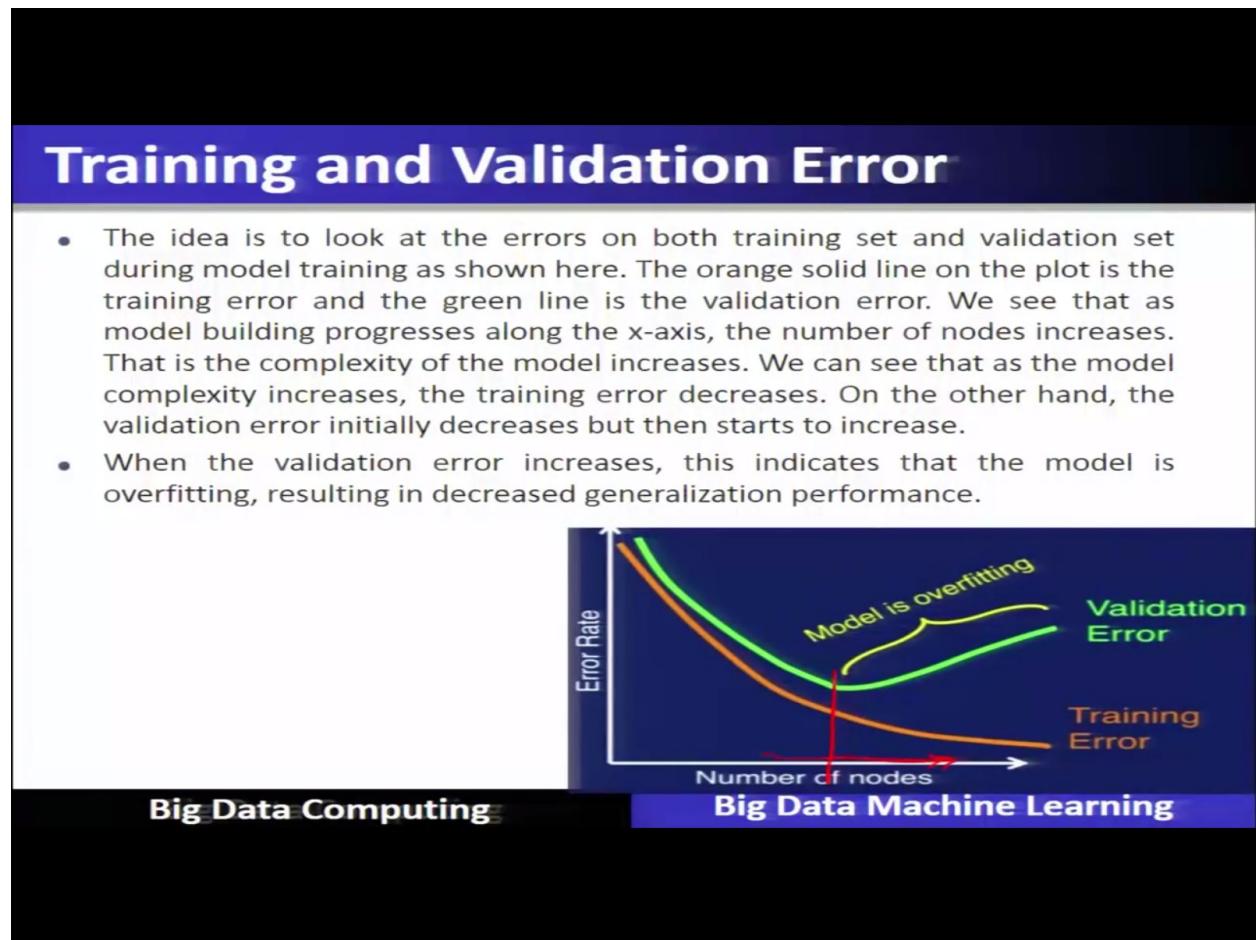
Big Data Computing

Big Data Machine Learning

So, training and validation error. Now, we want to further divide up the training into the training and the validation set. So, the training set is used to train the model as before and the validation is used to determine when to stop the training model to avoid the overfitting in order to get the best result performance. So, let us see that there are two different data sets one is the training data set which is shown over here that it is giving the errors which is called the training error. Now, with the validation data set we are also getting the errors and we can now compare these to graphs. What, we see here at this stage this validation is not consistent with the training error here it diverges. So, this is the position where this particular this is the position when the model is not behaving consistently with the validation set and the training set and there validation and the training error are differs at this point where validation error rate is more compared to the to the training set error. One, is growing the other is decreasing. So, therefore this particular at this point the model overfits. So, we have to identify when to at what stage the growing of the tree has to be used using this particular method. So, now let us understand this. Then, so the training set is used to train the model as before and the validation is used to determine when to stop the training model to avoid the overfitting in order to get the best generalization. So, the so the data is to the idea is to look at the errors of both the training and the validation set during the model training and the orange line here on the plot is shown as the training error and the green line is validation. We, see that the model building progress is along X axis so the number of node increases that is the complexity of the node of the model

increases here as the number of node the complexity. So, we can see that the model complexity increases at this after beyond this particular stage. The, training error decreases whereas the validation error increases start increasing. So, when the validation error increases this indicates that the model is overfitting and resulting into the decreased generalization performance.

(Refer Slide Time: 44:19)

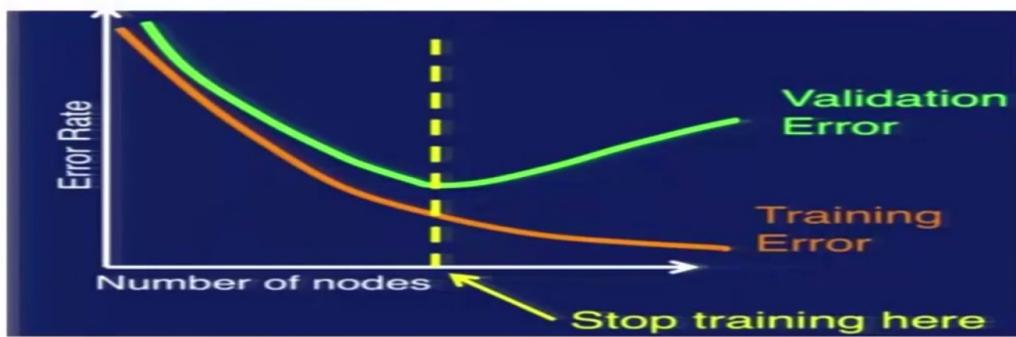


So, when to stop this particular training or tree induction. So, this can be used to determine when to stop the tree induction where the validation error starts to increase is when you get the best generalization performance. So, the training so the tree induction should be stopped here. This, method is using the validation set to determine when to stop the training is referred to as model selection since you are selecting one from many varying complexities. Note, that this was illustrated for the decision tree classifier and the same method can be applied to other machine learning models.

(Refer Slide Time: 45:01)

When to Stop Training ?

This can be used to determine when to stop training. Where validation error starts to increase is when you get the best generalization performance, so training should stop there. This method of using a validation set to determine when to stop training is referred to as model selection since you're selecting one from many of varying complexities. Note that this was illustrated for a decision tree classifier, but the same method can be applied to any type of machine learning model.



Big Data Computing

Big Data Machine Learning

What, are the ways to create and use the validation sets? There, are several ways to create and use the validation sets to avoid the overfitting. The different models are hold out method, random subsampling, K-fold cross-validation and leave-one-out cross-validation LOOC.
(Refer Slide Time: 45:25)

Ways to create and use the validation set

- There are several ways to create and use the validation set to avoid overfitting. The different methods are:
- Holdout method
- Random subsampling
- K-fold cross-validation, and
- Leave-one-out cross-validation

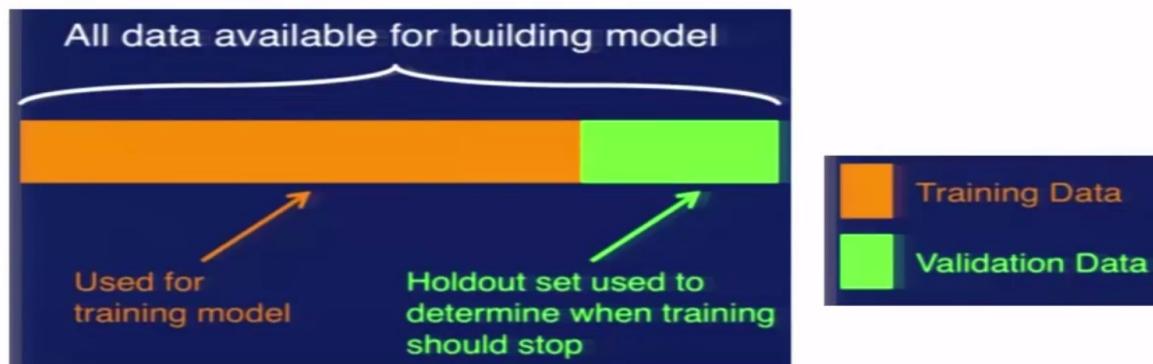
Big Data Computing

Big Data Machine Learning

So, hold out method the first way to use the validation set is a hold out method. This, describes the scenario that we are that we have discussed where the part of the training is reserved as the validation set the validation is then is then the hold out set.
(Refer Slide Time: 45:45)

Holdout Method

- The first way to use a validation set is the holdout method. This describes the scenario that we have been discussing, where part of the training data is reserved as a validation set. The validation set is then the holdout set.



Big Data Computing

Big Data Machine Learning

This, method we have already seen.

(Refer Slide Time: 45:47)

Repeated Holdout Method

- Repeating holdout method several times
- Randomly select different hold out set each iteration
- Average validation errors over all repetitions.

Big Data Computing

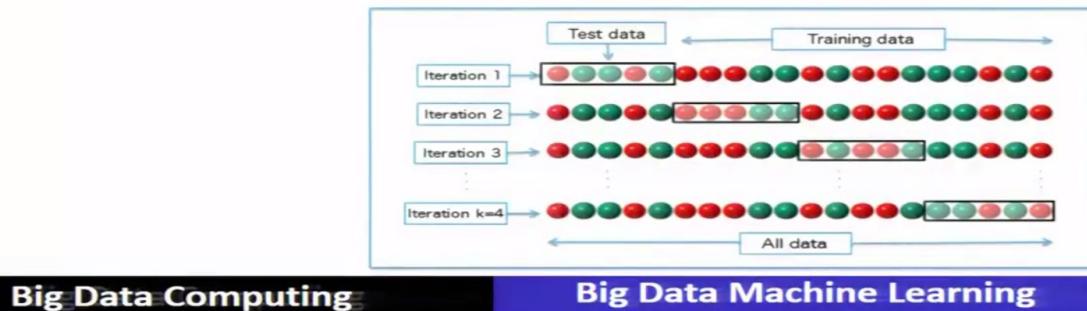
Big Data Machine Learning

Now, K-fold cross-validation the way to improve the repeated hold out hold-out method is to use cross-validation. Cross-validation, works as follows. Segment, the data into K number of disjoint partitions. During, each partition during each iteration one partition is used as the validation set. Repeat, the process K times each time a different partition for the validation so each partition is used for validation exactly once this is illustrated in the figure. In the first iteration, the first partition is used for the validation. In the second iteration, the second partition is used for the validation and so on.

(Refer Slide Time: 46:29)

K-Fold Cross-Validation

- A way to improve on the repeated holdout method is use cross-validation. Cross-validation works as follows. Segment the data into k number of disjoint partitions. During each iteration, one partition is used as the validation set. Repeat the process k times. Each time using a different partition for validation. So each partition is used for validation exactly once. This is illustrated in this figure. In the first iteration, the first partition, is used for validation. In the second iteration, the second partition is used for validation and so on.



Now, K-fold cross-validation. The overall validation error is calculated by averaging the validation for K different iteration. The model with the smallest average validation error is then selected. The process is just described referred the as K-fold cross-validation. This is very commonly used approach to model selection in practice. This particular approach gives you the more structured way to divide available data up between the training and the validation data set provide a way to overcome the variability in the performance that you can get when using a single partition of the data.

(Refer Slide Time: 47:06)

K-Fold Cross-Validation

- The overall validation error is calculated by averaging the validation errors for all k iterations.
- The model with the smallest average validation error then is selected. The process we just described is referred to as k-fold cross-validation. This is a very commonly used approach to model selection in practice.
- This approach gives you a more structured way to divide available data up between training and validation datasets and provides a way to overcome the variability in performance that you can get when using a single partitioning of the data.

Big Data Computing

Big Data Machine Learning

Leave out, leave-one-out cross-validation Leave-one-out cross-validation is a special case of K-fold cross-validation where K=N, where N is the size of data set. Here for each iteration, the validation set has exactly one sample. So, the model is trained to use N-1 sample is validated on the remaining. So, the rest of the process works the same as K-fold cross mode that the cross-validation is often abbreviated as CV and leave out leave-one-out cross-validation is referred as LOOCV and pronounced LOOCV.

(Refer Slide Time: 47:45)

Leave-One-Out Cross-Validation

- Leave-one-out cross-validation is a special case of k-fold cross-validation where k equals N, where N is the size of your dataset.
- Here, for each iteration the validation set has exactly one sample. So the model is trained to using N minus one samples and is validated on the remaining sample.
- The rest of the process works the same way as regular k-fold cross-validation.
- Note that cross-validation is often abbreviated CV and leave-one-out cross-validation is in abbreviated L-O-O-C-V and pronounced LOOCV.

Big Data Computing

Big Data Machine Learning

Uses of validation set. Note, that the validation error that comes out of this process can also be used to estimate the generalization performance of the model. In other words, the error on the validation set provides an estimate of the error on the test set.
(Refer Slide Time: 48:04)

Uses of Datasets

- With the addition of the validation set, you really need three distinct datasets when you build a model. Let's review these datasets.
- The training dataset is used to train the model, that is to adjust the parameters of the model to learn the input to output mapping.
- The validation dataset is used to determine when training should stop in order to avoid overfitting.
- The test data set is used to evaluate the performance of the model on new data.

Big Data Computing

Big Data Machine Learning

Uses of data sets with the addition of validation set you really need three distinct data sets when you build a model. Let us review these data sets. So, we have seen the first type of data set is called the training data set which is used to train the model that is to adjust the parameters of the model to learn the input to the output mapping that is called the training data set. The validation data set is used to determine when the training should stop in order to avoid overfitting so this is called the validation data set. Third is called the test data set is used to evaluate the performance of the model on the new data set. So, that means the data set is divided into three different type of data sets one is called the testing or training data set then it is the validation data set third is called test data set. Let, us see that the training data set is used to train the model that is to adjust the parameter of the model to learn the mapping from input to the output. So, that means training data set will build the model and the validation data set is used to determine when the training should stop in order to avoid the overfitting. So, while building the model it will refine this particular model so that it will be improved model which will not have the overfitting. Finally, the test data set is used to evaluate the performance of the model. So, this evaluation of this improved model will be done using the test data set. Now, let us see that how this data set or given data set is divided into three different parts which are called training data set validation data set and test data set.
(Refer Slide Time: 51:03)

Uses of Datasets

- With the addition of the validation set, you really need three distinct datasets when you build a model. Let's review these datasets.

- 1 The training dataset is used to train the model, that is to adjust the parameters of the model to learn the input to output mapping.
 - 2 The validation dataset is used to determine when training should stop in order to avoid overfitting.
 - 3 The test data set is used to evaluate the performance of the model on new data.
-
- ```
graph TD; Dataset --> Training[Training Data - 75%]; Dataset --> Validation[Validation Data - 15%]; Training --> Builds[Builds Model]; Validation --> Builds; Builds --> Overfitting[Overfitting]; Overfitting --> Improved[Improved Model]; Improved --> Evaluate[Evaluating w/ test dataset];
```

Big Data Computing

Big Data Machine Learning

So, note that the test data set should never be used in any way to create or tune the model. So, it should be a new data set which is shown. For example, in a cross-validation process to determine when to stop the training it should not be used there. So, the test data set must always remain independent from the model training and remain untouched until the very end when all the training has been completed. Note, that the sampling the original data set to created training validation test all the data set must contain the same distribution of the target class. For example, if the original data set if in the original data 75% samples belong to one class 30% samples to the other class then this same distribution should approximately be present in each of the training validation and test set otherwise the analysis will be misleading.  
(Refer Slide Time: 52:08)

## Uses of Datasets

- Note that the test data set should never, ever be used in any way to create or tune the model. It should not be used, for example, in a cross-validation process to determine when to stop training.
- The test dataset must always remain independent from model training and remain untouched until the very end when all training has been completed. Note that in sampling the original dataset to create the training, validation, and test sets, all datasets must contain the same distribution of the target classes.
- For example, if in the original dataset, 70% of the samples belong to one class and 30% to the other class, then this same distribution should approximately be present in each of the training, validation, and test sets. Otherwise, analysis results will be misleading.

**Big Data Computing**

**Big Data Machine Learning**

So, validation set summary. So, we have discussed the need for three different data sets in the model building. So, the training set is to train the model the validation set is to determine when to stop the training process and test to evaluate the performance on the new data. We, have learned how a validation set can be used to avoid overfitting and in the process to provide an estimate general performance estimate of generalized performance and the generalization performance. And, we cover different types to create and use validation set such as K-fold cross-validation.

(Refer Slide Time: 52:48)

## Validation Set Summary

- We have discussed the need for three different datasets in building model. A training set to train the model, a validation set to determine when to stop training, and a test to evaluate performance on new data.
- We learned how a validation set can be used to avoid overfitting and in the process, provide an estimate of generalization performance.
- And we covered different ways to create and use a validation set such as k-fold cross-validation.

**Big Data Computing**

**Big Data Machine Learning**

Now metrics to evaluate the model performance.

## **Metrics to Evaluate Model Performance**

**Big Data Computing**

**Big Data Machine Learning**

So, here we give the class labels to the data set in supervised learning that is done is the animal mammal yes or no is the class label.

(Refer Slide Time: 53:02)

## Class Labels



**Big Data Computing**

**Big Data Machine Learning**

Now, using this particular target label we now can see the difference between the true target label this is called target label this is the true label and this is the label which is the predicted label out of the classification algorithm this is called predicted label. Now, if you compare them if both the labels are same then the error type is called there is no error and it is called true positive. When, the true label is no and the predicted is also no then also it is called the true negative. So, both are not having any errors. Now, when the true label is no but the predicted label is yes then it is called the false positive. So, this positive is falsely given hence it is called a false positive. Now, if the true label is yes and the predicted label is no so it is false negative. So, these different classes we are now going to see this is called the types of classification error. So, there are four different types of classification error true positive, true negative is there there is no error and the predicted label is wrongly saying yes then it is called false positive and if the predicted label is saying wrongly no yeah it is saying wrongly no then it is called false negative. (Refer Slide Time: 54:43)

## Types of Classification Errors

| True Label | Predicted Label | Error Type          |
|------------|-----------------|---------------------|
| Yes        | Yes             | True Positive (TP)  |
| No         | No              | True Negative (TN)  |
| No         | Yes             | False Positive (FP) |
| Yes        | No              | False Negative (FN) |

Big Data Computing

Big Data Machine Learning

So, let us see calculate the accuracy rate out of this error types. So, accuracy rate is nothing but the number of correct predictions divided by total predictions. So, number of correct predictions here you can see that the true value is yes so it is also predicting yes true is no it is also saying no and number of correct predictions and here in all other 2 cases it is wrongly predicted. So, the number of correct predictions is 2 divided by how there are 4 different. So, in that particular sample we have to find out how many such false positives, false negatives, true positive and true negatives are there and based on that we can find out the accuracy rate and error rate is (1- accuracy rate) that is 0.3 here in this case.

(Refer Slide Time: 55:48)

## Error Rate

### Error Rate

| True | Predicted |
|------|-----------|
| Yes  | Yes       |
| No   | No        |
| No   | Yes       |
| Yes  | No        |

$$\begin{aligned}\text{Error Rate} &= \frac{\# \text{ incorrect predictions}}{\# \text{ total predictions}} \\ &= 1 - \text{Accuracy Rate} \\ &= 1 - 0.7 = 0.3\end{aligned}$$

Error  
True Positive (TP)  
True Negative (TN)  
False Positive (FP)  
False Negative (FN)

Big Data Computing

Big Data Machine Learning

So, let us see another thing is called recall and precision says that (

$\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$ ). True positives divided by all positives which is predicted by the prediction is called precision. And recall is true positives divided by that is true positives and false negatives ( $\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$ ) that is all samples with so this is called recall.  
(Refer Slide Time: 56:21)

## Precision and Recall

### Precision & Recall

| True | Predicted |                |
|------|-----------|----------------|
| Yes  | Yes       | True Positive  |
| No   | No        | True Negative  |
| No   | Yes       | False Positive |
| Yes  | No        | False Negative |

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \checkmark \quad \leftarrow \begin{array}{l} \text{All samples with} \\ \text{Predicted} = \text{Yes} \end{array}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \checkmark \quad \leftarrow \begin{array}{l} \text{All samples with} \\ \text{True} = \text{Yes} \end{array}$$

Big Data Computing

Big Data Machine Learning

So, precision and recall, precision is considered as the major of exactness because it calculates the percentage of samples predicted as positive which are actually positive in the class. Recall, is considered as the major of completeness because it calculates the percentage of positive samples that that the model correctly identifies.

(Refer Slide Time: 56:45)

## Precision and Recall

- Precision is considered a measure of exactness because it calculates the percentage of samples predicted as positive, which are actually in a positive class.
- Recall is considered a measure of completeness, because it calculates the percentage of positive samples that the model correctly identified.

**Big Data Computing**

**Big Data Machine Learning**

So, besides precision and recall there is something which is called the F-Measure, which combines both of them. So,  $F_1$  measure is nothing but  $2 * \left( \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \right)$  where  $F_1$  is evenly weighted that means the precision and recall are given equal weights then it is called  $F_1$  measure. And,  $F_2$  measure here the weighs recall more in this case and  $F_{0.5}$  measure gives the weights precision more in this case.  
(Refer Slide Time: 57:25)

## F-Measure

Precision



Recall

$$F_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

- $F_1$  : evenly weighted
- $F_2$  : weights Recall more
- $F_{0.5}$ : weights Precision more

Big Data Computing

Big Data Machine Learning

Let, us see the confusion matrix.

## **Confusion Matrix**

**Big Data Computing**

**Big Data Machine Learning**

So, again we will see that if the animal is the animal mammal so this data set of animal we are going to label that yes or no.

(Refer Slide Time: 57:38)

# Classification



**Big Data Computing**

**Big Data Machine Learning**

So, again there are four different classification errors. Based, on these classification errors we can draw a confusion matrix. So, confusion matrix on the X axis gives the true class label and whereas the Y is called predicted class label. So, if the true class label is yes and the predicted class label is also yes then it is called true positive and if the true class label is yes and the predicted is no then it is called the the false negative. Similarly, if the true class label is no and the predicted class label is yes then it is called false positive. And, if the true label is no and the predicted is also no then it is called true negative. So, this particular matrix is called the confusion matrix.

(Refer Slide Time: 58:36)

# Confusion Matrix



|                    |       | Predicted Class Label ✓ |                       |
|--------------------|-------|-------------------------|-----------------------|
|                    |       | Yes ✓                   | No ✓                  |
| True Class Label ✓ | Yes ✓ | True Positive (TP) ✓    | False Negative (FN) ✓ |
|                    | No ✓  | False Positive (FP) ✓   | True Negative (TN) ✓  |

Big Data Computing

Big Data Machine Learning

So, let us see the confusion matrix for this particular example and here in this particular example we can see that this is the true label and this true times, so so this is the correct this is correctly predicted and this is also correctly saying. So, yes if it is yes then it is correctly predicted how many times 1 2 3 so it is 3 times it is appearing and when it is when it is no it is also correctly predicting negative how many times? 1 2 3 and 4 i.e. 4 times it is correctly predicting. And, it is when it is no and yes so there it is no and it is there it is no and yes it is one times it is false positive and the remaining is yes or no 1 and 2 times. So, this we have calculated the confusion matrix. So, total number of samples is 1 2 3 4 5 6 7 8 9 10 so it is 0.3, 0.2, 0.4, 0.1 so that means number of false positive is 10% and number of true negative so true positive is 30% and so this is the correct outcomes so that means 70% is the accuracy of this particular case whereas 30% is the inaccuracy.

(Refer Slide Time: 60:51)

| True Label | Predicted Label |
|------------|-----------------|
| Yes /      | No ✓            |
| No /       | No ✓            |
| No /       | No ✓            |
| Yes /      | Yes ✓           |
| Yes /      | Yes ✓           |
| No /       | No ✓            |
| Yes /      | No ✓            |
| Yes /      | Yes ✓           |
| No /       | No ✓            |
| No /       | Yes ✓           |

|                  |     | Predicted Class Label |                     |
|------------------|-----|-----------------------|---------------------|
| True Class Label | Yes | True Positive (TP)    | False Negative (FN) |
|                  |     | False Positive (FP)   | True Negative (TN)  |
| Yes              | ✓   | 3                     | 2                   |
| No               | ✓   | 1                     | 4                   |

Confusion matrix

10% 70%

Big Data Computing

Big Data Machine Learning

So, let us see the machine learning tools.

## **Machine Learning Tool**

**Big Data Computing**

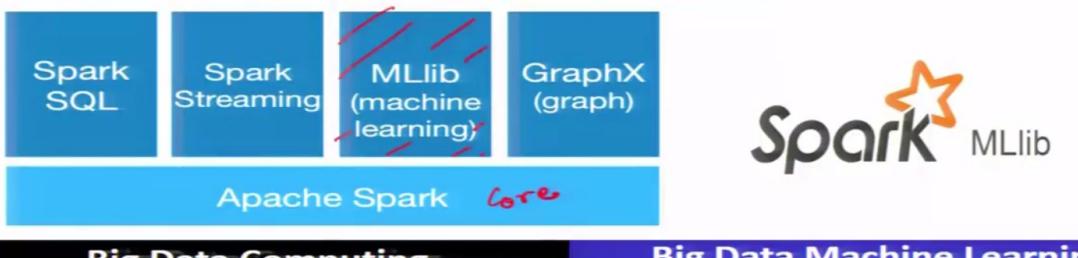
**Big Data Machine Learning**

Spark MLlib. Spark MLlib, is the distributed machine learning framework on top of the spark core. So, this is the spark core and on top of it is the machine learning MLlib library is positioned. So, machine learning spark MLlib is a distributed machine learning framework. So, what do you mean by distributed machine learning. We will see that, this machine learning that means the data is big that the machine learning requires the data so that it will build the model it will learn the model and it will solve the problem. So, the data is a big data which cannot be stored on a single system therefore the data is to be stored on a distributed platform and the machine learning algorithm also to be applied on that distributed way hence it is called the distributed machine learning. So, spark MLlib is the the distributed machine learning framework build on top of the spark core. MLlib, is a spark scalable machine learning library consisting of common machine learning algorithms and utilities including classification, regression, clustering, collaborative filtering and dimensionality reduction. So, what do you mean by scalable machine learning library is that the cluster machines can follow the property of scale out hence the particular algorithm that is machine learning algorithm follows that that particular property hence it is called as scalable machine learning libraries which is the spark MLlib.

(Refer Slide Time: 62:45)

## Spark MLlib

- **Spark MLlib** is a distributed machine-learning framework on top of Spark Core.  
*→ Big Data*
- MLlib is Spark's scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction.



Now, spark MLlib has library has four different components. So, first component is called different analysis algorithms for machine learning purpose. So, the all the algorithms which is machine learning algorithms are available as a part of spark MLlib that is the classification, regression, clustering and collaborative filtering. Now, besides these algorithm it is also having a feature which is called a pipeline. So, the so the data scientist working in the big data scenario they can write their own pipeline and do the and apply the machine learning approach on the big data using pipeline. We will see in this particular part of the discussion that Spark MLlib provides much pipeline and how pipeline can be used for big data analytics. Then, another component of MLlib is called featurization which has two parts very important parts which are called extraction and transformation. So, we will see what do you mean by these extraction and transformation in this building up the pipeline. And, it has also the utilities such as linear algebra and forming different statistics.

(Refer Slide Time: 64:11)

# Spark MLlib Component

## Algorithms

- Classification
- Regression
- Clustering
- Collaborative Filtering

## Pipeline

- Constructing
- Evaluating
- Tuning
- Persistence

## Featurization

- Extraction
- Transformation

## Utilities

- Linear algebra
- Statistics

**Big Data Computing**

**Big Data Machine Learning**

Now, let us see the classification algorithm which are available in the spark. First, let us see what are the steps of our doing the classification in spark. So, these are the steps we will follow and we will see an example how the classification is performed in the spark. Classification, is to first step is to load the data into the data frame, second is to drop unused and missing data item. So, this is the data preparation. Here, is to read the data then we have to create a categorical variable for the low humidity days so that means we will transform the data to be required in our format. Then, we have to perform the aggregate features used to make predictions and that means you have to do a feature engineering or feature selection. And, then split the data into the training and test data sets. Then, we will create and train the decision tree and then we will save the predictions to the file  
(Refer Slide Time: 65:55).

## Steps for Classification in Spark

1. Load weather data into DataFrame (Read Data)
2. Drop unused and missing data. (Data preparation)
3. Create a categorical variable for low humidity days (Training)
4. Aggregate features used to make predictions. (Feature Selection)
5. Split the data into training and test sets. (Split Data-set into Train & Test)
6. Create and train the decision tree.
7. Save the predictions to a CSV file

**Big Data Computing**

**Big Data Machine Learning**

Let, us see how to read the weather data into the data in spark MLlib or in spark format. So, here we can see that this is the data file and we have to load using the load data file using SQL context. So, we are using SQL spark SQL using SQLcontext we will read the data file which is shown over here using these commands of spark. So, using SQL context we will read this particular file and the format is given as .CSV and other information. Let, us say header is also there and infer schema is also available as far as spark SQL. So, we will now read in the form of a dataframe. So, the weather data will be loaded into dataframes and now if you want to see what are the different columns after performing this read operation, what are the columns read then these are the different columns of the data set which is being read into the data frames. That is, air pressure at 9 am, air pressure air temperature at 9 am, average wind direction at 9 am, average wind speed at 9 am, average wind direction at 9 am, maximum wind speed 9 am, rain accumulation 9 am, rain duration 9 am. So, these are different columns of your data set and this column will be used to construct the decision trees later on.  
(Refer Slide Time: 67:47)

## 1. Load weather data into DataFrame

The first cell contains the classes we need to load to run this exercise. Next we create SQL context and load the weather data CSV into a data frame.

The second cell also prints all the columns in this data frame.

The third cell defines the columns in the weather data we will use for the decision tree classifier.

```
In []: from pyspark.sql import SQLContext
 from pyspark.sql import DataFrameNaFunctions
 from pyspark.ml import Pipeline
 from pyspark.ml.classification import DecisionTreeClassifier
 from pyspark.ml.feature import Binarizer
 from pyspark.ml.feature import VectorAssembler, StringIndexer, VectorIndexer

In []: sqlContext = SQLContext(sc)
 df = sqlContext.read.load('file:///home/cloudera/Downloads/big_data-4/daily_weather.csv',
 format='com.databricks.spark.csv',
 header='true', inferSchema='true')
 df.columns

In []: featureColumns = ['air pressure 9am', 'air temp 9am', 'avg wind direction 9am', 'avg wind speed 9am',
 'max wind direction 9am', 'max wind speed 9am', 'rain accumulation 9am',
 'rain duration 9am']
```

Big Data Computing      Big Data Machine Learning

So, the data frame columns are again summarized here in a nice manner.

(Refer Slide Time: 67:57)

The screenshot shows a Jupyter Notebook interface with a Python 3 kernel. The code cell contains the following:

```
header='true',inferSchema='true')
df.columns
Out[2]: ['number',
 'air_pressure_9am',
 'air_temp_9am',
 'avg_wind_direction_9am',
 'avg_wind_speed_9am',
 'max_wind_direction_9am',
 'max_wind_speed_9am',
 'rain_accumulation_9am',
 'rain_duration_9am',
 'relative_humidity_9am',
 'relative_humidity_3pm']

In []: featureColumns = ['air_pressure_9am','air_temp_9am','avg_wind.direction_9am','avg_wind_speed_9am'
 'max_wind.direction_9am','max_wind_speed_9am','rain_accumulation_9am',
 'rain_duration_9am']
```

The output cell (Out[2]) displays a list of column names. A red bracket is drawn around the list to highlight it. The notebook interface includes a toolbar at the top and a navigation bar with links like Hue, Hadoop, HBase, Impala, Spark, Solr, Oozie, Cloudera Manager, and Getting Started.

And, after that then we will prepare the data so that all the unused or the missing data can be removed. So, use the column name and drop use the column name number and drop from drop that from the data frame and if when whenever there is a row which is missing and then we will print the number of rows and columns in the let us see how we can do this to drop this one data which is having missing data. So, let us see we will find out this one the total count. So, total count of this one data value is 1064 and we will also see that. So, using df.count() and the length of that data frame of the columns we will find out the total number of data set and we will find out how many columns or data is available and if there is a difference so we will find out those the missing data item and remove it. So, therefore we will calculate the total number of and we will drop all those we will drop the the missing data item from that particular column. And, then after that we will see that it comes out to be the data set which does not have any missed data.

(Refer Slide Time: 69:45)

## 2. Drop unused and missing data

Use the column name number.

Drop that from the data frame, df = df.drop Number.

Rows with missing data, df = df.na.drop.

Print the number of rows and columns in our resulting data frame,  
df.count(), len(df.columns).

```
Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Edit View Insert Cell Kernel Help
Cell Toolbar
In [3]: featureColumns = ['air pressure_9am', 'air temp_9am', 'avg wind direction_9am', 'avg wind speed_9am',
 'max wind direction_9am', 'max wind speed_9am', 'rain accumulation_9am',
 'rain duration_9am']
In [4]: df = df.drop('number') Drop the missing data
In [5]: df = df.na.drop() Drop the missing data
In [6]: df.count(), len(df.columns)
Out[6]: (1064, 10)
```

**Big Data Computing**

**Big Data Machine Learning**

Now, third step is to create the categorical variable for the low humidity days. So, meaning to say that what we will do here is that wherever the numbers are wherever the values categorical values are there these variables are to be converted into the number and this conversion can be performed from categorical data to the numerical data. And here, we can see that these labels are binarized. So these values are binarized and the values are shown as the categorical data.

(Refer Slide Time: 70:15)

### 3. Create a categorical variable for low humidity days

We will enter `binarizer = Binarizer ()`. The first argument specifies a threshold value for the variable. We want the categorical variable to be 1, if the humidity is greater than 25%. So we'll enter a threshold=24.9999. The next argument specifies the column to use to create the categorical variable. We'll input, `inputCol = relative_humidity_3pm`. The final argument specifies the new column name, `outputCol = label`. A new data frame is created with this categorical variable. `binarizedDF = binarizer.transform df`.

The screenshot shows a Jupyter Notebook interface with the following code in cell In [7]:

```
Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Edit View Insert Cell Kernel Help
In [3]: featureColumns = ['air pressure 9am', 'air temp 9am', 'avg wind direction 9am', 'avg wind speed 9am', 'max wind direction 9am', 'max wind speed 9am', 'rain accumulation 9am', 'rain duration 9am']
In [4]: df = df.drop('number')
In [5]: df = df.na.drop()
In [6]: df.count(), len(df.columns)
Out[6]: (1064, 10)
In [7]: binarizer = Binarizer(threshold=24.9999, inputCol='relative_humidity_3pm', outputCol='label')
binarizedDF = binarizer.transform(df)
```

Below the code, there are two tabs: "Big Data Computing" and "Big Data Machine Learning".

Now, aggregate features will be used to make the predictions and now this particular data set will be split into once the data is prepared then it will be split into the training and the test data. So, this will be having 80% and 20%. 80% will be training and 20% will be test data and then we will randomly split that and using the training data we will now fit we will process it to build the decision trees. Let, us see how we will do this. So, after doing this here we can see that we are building the we are setting up the decision tree and in the decision tree pipeline we have to build and now we will perform the the this one to fit the model. So, model is when it is fit that training on a training data the model will be the decision tree and once the model is fit then we will perform the predictions on that particular data.

(Refer Slide Time: 71:52)

## 6. Create and train the decision tree. 1:48 / 1:13:07

Next, we can create a model by training the decision tree. We can do this by executing it in a pipeline. Enter pipeline=Pipeline (stages=[dt]). We will create them all by putting a training data, model = pipeline.fit(trainingData). Let's run this Now, we can make predictions using our test data. Enter predictions = model.transform(testData).

The screenshot shows a Jupyter Notebook interface with the following code:

```
Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Edit View Insert Cell Kernel Help
assembled = assembled.transform(binarizedour)

n [10]: (trainingData, testData) = assembled.randomSplit([0.8,0.2], seed=13234)

n [11]: trainingData.count(), testData.count()
out[11]: (854, 210)

n [12]: Difer(labelCol='label', featuresCol='features', maxDepth=5, minInstancesPerNode=20, impurity="gini")
<-->
n [13]: pipeline = Pipeline(stages=[dt])
model = pipeline.fit(trainingData)
model
n [14]: predictions = model.transform(testData)
In []:
```

The code performs the following steps:

- Imports the assembled dataset.
- Splits the dataset into trainingData and testData in a 80/20 ratio.
- Creates a decision tree model using the Difer class with parameters: labelCol='label', featuresCol='features', maxDepth=5, minInstancesPerNode=20, and impurity="gini".
- Creates a Pipeline object with the decision tree stage.
- Fits the pipeline to the training data.
- Transforms the test data using the fitted model.

Big Data Computing

Big Data Machine Learning

So, once the predictions are made we will see the labels and the prediction values. So, here you see that all are having same and now we will see prediction accuracy using. So, these predictions values are now stored on the on the file CSV file.

(Refer Slide Time: 72:15)

## Conclusion

- In this lecture, we have discussed the machine learning techniques.
- We have also discussed tools and algorithms that you can use to create machine learning models that learn from data, and to scale those models up to big data problems.

**Big Data Computing**

**Big Data Machine Learning**

So, conclusion: in this lecture we have discussed the machine learning techniques. We have also discussed the tools and algorithms that you can use to create the machine learning models that learned from the data and to scale those models up to the big data problem. Thank you.  
(Refer Slide Time: 72:32)

## Conclusion

- In this lecture, we have discussed the machine learning techniques.
- We have also discussed tools and algorithms that you can use to create machine learning models that learn from data, and to scale those models up to big data problems.

**Big Data Computing**

**Big Data Machine Learning**

## **Lecture-25**

### **Machine Learning Algorithm K-means using Map Reduce for Big Data Analytics**

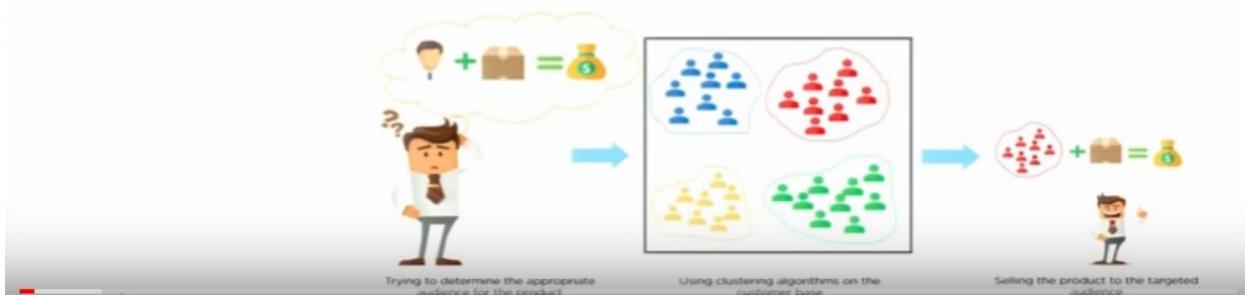
Machine Learning Algorithm, Parallel k-means, using Map Reduce, for big data analytics.

Refer slide time: (0:21)

## Preface

### Content of this Lecture:

- In this lecture, we will discuss machine learning classification algorithm k-means using mapreduce for big data analytics

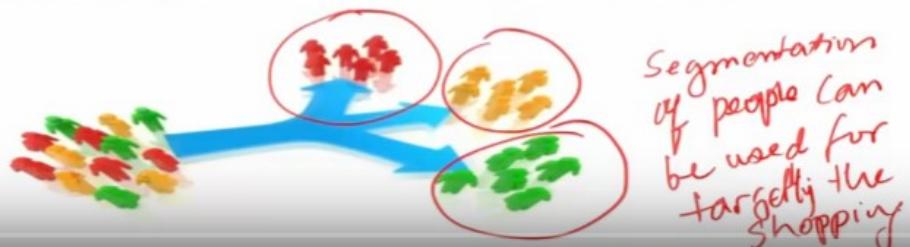


Preface, content of this lecture: In this lecture we will discuss machine learning algorithm, k-means using Map Reduce for big data analytics.

Refer slide time: (0:33)

# Cluster Analysis Overview

- **Goal:** Organize similar items into groups.
- In cluster analysis, the goal is to organize similar items in given data set into groups or clusters. By segmenting given data into clusters, we can analyze each cluster more carefully.
- Note that cluster analysis is also referred to as clustering.



Before, we go into the deeper of k-means algorithm let us understand some of the preliminaries which are required to understand this k-means big data analytics algorithm. Now, the term which is called ‘Cluster Analysis’ is a widely known term in the machine learning framework and the cluster analysis goal is to organize the similar items in a given data set into the groups or they are also called ‘Cluster’. By, segmenting this way the given data into the clusters we can gain the insight into the data set about the clusters or the groups and this is called the ‘Segmentation’. And, we can analyze each cluster or a group with more careful inside. Now, this cluster analysis is also known as clustering. For example, here we can see here there is a group of people. Now, as far as these group of people can be segmented into different groups based on their purchasing capacity or are maybe that different interest groups and soon. So, here we can see that all the interest groups of a particular interest they are classified into four different classes. And, this can be used this segmented can be used for targeting the the shopping businesses.

Refer slide time: (2:48)

## Applications: Cluster Analysis

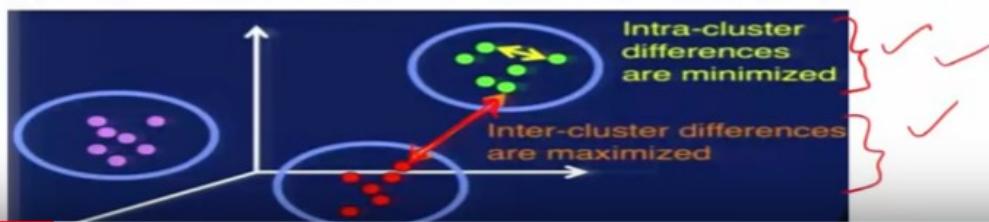
- **Segment customer base into groups:** A very common application of cluster analysis is to divide your customer base into segments based on their purchasing histories. For example, you can segment customers into those who have purchased science fiction books and videos, versus those who tend to buy nonfiction books, versus those who have bought many children's books. This way, you can provide more targeted suggestions to each different group.
- **Characterize different weather patterns for a region:** Some other examples of cluster analysis are characterizing different weather patterns for a region.
- **Group news articles into topics:** Grouping the latest news articles into topics to identify the trending topics of the day.
- **Discover crime hot spots:** Discovering hot spots for different types of crime from police reports in order to provide sufficient police presence for problem areas.

Let us see some other applications of this cluster analysis. Here, the first such application which is well known is called 'Segment Cluster Base' into various groups. So, hereafter identifying these particular customers into different groups based on their purchasing histories these segment customers into who purchased? Which of these items can be useful for different business applications for targeting advertisements or for targeting different promotions? So, that their businesses can be directly dealt with the interested set of people with their known preferences. Now, similarly the other application is to characterize different weather patterns for a region. So, this particular way the cluster analysis can characterize different weather patterns of a region and therefore different disaster management or different applications can be based on this kind of clustering or segmentation. Similarly, as far as the group news articles can also be used so that they can be identified that means different news articles can be grouped according to that trending topics of the day and also to discover the crime hotspots it is possible that to identify different type of crimes and they are located and using cluster analysis identifies the hottest part so that the sufficient number of police personals can be put on their presence to solve any problem related to that spot or for that crime spot.

Refer slide time: (4:49)

# Cluster Analysis

- **Divides data into clusters:** Cluster analysis divides all the samples in a data set into groups. In this diagram, we see that the red, green, and purple data points are clustered together. Which group a sample is placed in is based on some measure of similarity.
- **Similar items are placed in same cluster:** The goal of cluster analysis is to segment data so that differences between samples in the same cluster are minimized, as shown by the yellow arrow, and differences between samples of different clusters are maximized, as shown by the orange arrow. Visually, we can think of this as getting samples in each cluster to be as close together as possible, and the samples from different clusters to be as far apart as possible.

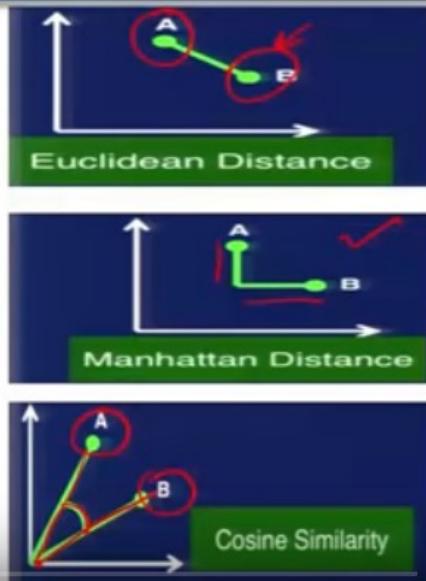


Similarly, we can divide the data into the cluster. So, cluster analysis divides the samples of in two groups. And, in this particular picture we can see here there are three different groups which are shown with their members as red, green, purple data points. Now, they are grouped together based on some measure which is called a 'Similarity Measure'. We, are going to see what do you mean by the similarity measures? So, this particular clustering or the cluster analysis is with reference to some measure of similarity we are going to see about this particular term in more detail in further slides. Now, in this particular cluster analysis we are going to keep the similar items together in the same cluster. So, the goal of cluster analysis is to segment the data. So, that the so that the difference is between the between the samples in the same cluster are minimized. So, here we can see by an example that for example all the green ones if we see the differences in terms of measuring the similarities. So, they are basically having a very little difference hence they are grouped together in a form of a cluster. So, this way the differences between the samples of different clusters are maximized here we can see by this particular arrow. Now, the differences between the elements across different clusters are maximized. So, there will be a different separation and different clusters can be easily identifiable so that means in a nutshell we can summarize by saying that the similar items are placed are placed in the same cluster .that is their intra cluster differences are minimum and also across different clusters what we can say that the inter cluster differences are to be maximized and when these two things these two conditions are met then we are done with that cluster and clustering analysis is about dividing into these clusters so that similar item can be placed together and the items which are in different clusters they are separated with the maximum differences.

Refer slide time: (7:24)

# Similarity Measures

- Cluster analysis requires some sort of metric to measure similarity between two samples. Some common similarity measures are **Euclidean distance**, which is the distance along a straight line between two points, A and B, as shown in this plot.
- Manhattan distance**, which is calculated on a strictly horizontal and vertical path, as shown in the right plot. To go from point A to point B, you can only step along either the x-axis or the y-axis in a two-dimensional case. So the path to calculate the Manhattan distance consists of segments along the axes instead of along a diagonal path, as with Euclidean distance.
- Cosine similarity** measures the cosine of the angle between points A and B, as shown in the bottom plot. Since distance measures such as Euclidean distance are often used to measure similarity between samples in clustering algorithms, note that it may be necessary to normalize the input variables so that no one value dominates the similarity calculation.



Now, let us see the similarity measures which is the which is the main parameter by which this particular clustering or cluster analysis or is particularly performed. So, this is called 'Similarity measures'. So, here cluster analysis required some sort of measures to some sort of metric to measure similarity between two samples. Some, common similarity measures are equilibrium distance which is the distance along the straight line between two points A and B which are shown over here. So, this is called 'Equilibrium Distance' here the measure is about the the distance between the any two points which are shown here by the straight line. So, there is a equation and equilibrium distance follows the triangle inequality properties of mathematics and this kind of measure is used here for doing the clustering and in the cluster analysis. Another, similarity measure is called the 'Manhattan Distance' which is calculated on strictly following the horizontal or on the vertical path as shown here in this particular diagram. Now, from to go from A to B it has to follow either the horizontal or the vertical path it cannot follow the diagonal path in this two-dimensional space that is the case that it is called Manhattan distance and using this particular similarity measure you can also do the clustering in certain applications. Finally, there is another dissimilarity measure which is also very popular besides all these is called 'Cosine Similarities'. So, in the cosine similarities there is a measure of cosine of the angle between two points. For example, the points are A and B and if with the reference you can plot, you can draw an angle and the cosine of this angle is, basically called a cosine similarity which is shown here in this particular diagram. Now, you can see that, Euclidean distances are used to measure the similarity between the samples in the clustering algorithm and it may be necessary to normalize this input. Now, so that no no one value will dominate the similarity calculations and here as far as cosine similarity is concerned on this kind of similarity is used in some of the applications for example the documents which are closely related to some topic their cosine similarity will be less and they are they are together will form a cluster or a cluster of topics. So, if a corpus of documents is given then you can apply this cosine similarity and this cosine similarity will segment all that topics segment the corpus into topic wise or distribution or classification and this can be used in news reading promotions.

Refer slide time: (10:47)

## Another natural inner product measure

$x_i$

|   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 5 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

$x_q$

**Similarity**

$$= \mathbf{x}_i^T \mathbf{x}_q$$

$$= \sum_{j=1}^d \mathbf{x}_i[j] \mathbf{x}_q[j]$$

$$= 13$$

Now, we can see this particular cosine similarity is nothing but it is a dot product of two vectors.

Refer slide time: (10:57)

## Cosine similarity- normalize

$$\text{Similarity} = \frac{\sum_{j=1}^d \mathbf{x}_i[j] \mathbf{x}_q[j]}{\sqrt{\sum_{j=1}^d (\mathbf{x}_i[j])^2} \sqrt{\sum_{j=1}^d (\mathbf{x}_q[j])^2}}$$

$$= \frac{\mathbf{x}_i^T \mathbf{x}_q}{\|\mathbf{x}_i\| \|\mathbf{x}_q\|} = \cos(\theta)$$

- Not a proper distance metric
- Efficient to compute for sparse vecs

So, you can see that these vectors are nothing but they are the features. And, if you take the cosine similarity which by normalization it will form this equation that is:

$$\frac{X_i^T X_q}{\|X_i\| \vee \|X_q\| \vee \text{ii}}$$

which is nothing but all this is the summation of all points of excise and their square root of it. So, this particular way this cosine similarity will formulate the normalized view normalized way of the measuring the similarities among the two points or a to two topics and so on can be used in various applications. Now, this particular cosine similarity is used in the document classification why because these are very sparse. Hence, this kind of similarity measure that is called ‘Cosine Similarity Measure’ is very useful in such scenarios.

Refer slide time: (12:10)

## Cosine Normalize

The diagram shows three separate 2D Cartesian coordinate systems. In each system, two vectors originate from the same point on the negative x-axis. The first vector is short and the second is long, resulting in a very small angle  $\theta$  between them. A handwritten note indicates this is 'Similar' with a cosine value of 1 labeled as  $\cos(\theta) = 1$ . The second system shows two vectors of equal length forming a 90-degree angle ( $\pi/2$ ). A handwritten note indicates this is 'Very similar' with a cosine value of 0 labeled as  $\cos(\theta) = 0$ . The third system shows two vectors pointing in opposite directions, forming a 180-degree angle ( $\pi$ ). A handwritten note indicates this is 'Very dissimilar' with a cosine value near -1 labeled as  $\cos(\theta) \approx -1$ .

In general,  $-1 < \text{similarity} < 1$   
 For positive features (like tf-idf)  
 $0 < \text{similarity} < 1$

Define **distance** =  $1 - \text{similarity}$

A small diagram of a unit circle centered at the origin of a coordinate system. A radius vector is drawn from the origin to a point on the circumference. The angle between the negative x-axis and this vector is labeled  $\theta$ . A handwritten note next to the angle is labeled 'cosine'. The text 'Define distance = 1 - similarity' has a red checkmark pointing to this diagram.

To, have more understanding about cosine similarity and how this cosine similarity is normalized? We can see here that if this angle is very small then this cosine value of that similarity of these two points will become 1. Hence, higher is the value the mean it is closer or more similar the documents are. And, if they are separated apart then the cosine value of that angle will become 0 hence the this similar. This similar, objects or topics are having the highest cosine values and otherwise it will become zero. So, the distance is often measured because 1-similarity hence the smaller values are taken as so here in this particular method.

Refer slide time: (13:18)

# Cluster Analysis Key Points

**Unsupervised ✓**

Cluster analysis is an unsupervised task. This means that there is no target label for any sample in the data set.

**There is no  
'correct' clustering ✓**

In general, there is no correct clustering results. The best set of clusters is highly dependent on how the resulting clusters will be used.

**Clusters don't  
come with labels ✓**

Clusters don't come with labels. You may end up with five different clusters at the end of a cluster analysis process, but you don't know what each cluster represents. Only by analyzing the samples in each cluster can you come out with reasonable labels for your clusters. Given all this, it is important to keep in mind that interpretation and analysis of the clusters are required to make sense of and make use of the results of cluster analysis.

**Interpretation and analysis required to  
make sense of clustering results! ✓**

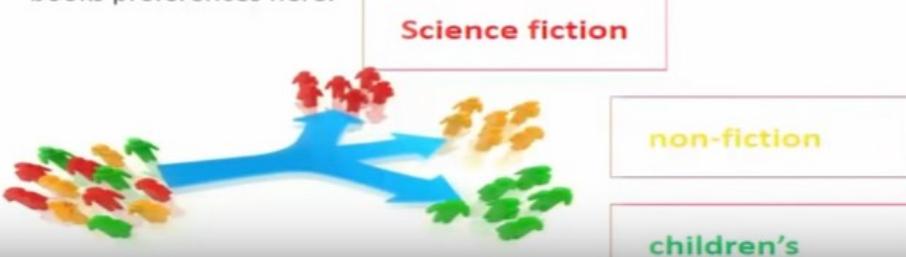
*Cluster  
Analysis  
Learning  
ML  
→ insight of data  
Set  
→ Application  
specific*

Now, cluster analysis the key points. We can see that, this particular technique of clustering or a cluster analysis is categorized as the unsupervised machine learning why because, the the data points or a dataset is not having any target label hence it is called 'Unsupervised Learning'. Another thing is, the clustering there is no correct way of the of being doing the clustering. Hence, it requires a different application and their interpretations based on this particular technique. Now, another key point about this cluster analysis is that clusters do not come with the label that we have seen and using this particular cluster analysis we do not know what each cluster will represent so hence by analyzing the samples in each cluster you can come out with a reasonable label for your clusters given all these it is important to keep in mind that interpretation and analysis of the clusters are required to make sense of and make use of the result of the cluster analysis. So, interpretations and analysis is required to make sense of these clustering results. So, that means these cluster analysis is a machine learning technique to gain the insight of the data set. And, by doing this interpretation and this analysis you can use this insight for any application or the business. Hence, the interpretation and the analysis is very easy application is specific.

Refer slide time: (15:22)

## Uses of Cluster Results

- **Data segmentation**
- **Analysis of each segment can provide insights:** There are several ways that the results of cluster analysis can be used. The most obvious is data segmentation and the benefits that come from that. If we segment your customer base into different types of readers, the resulting insights can be used to provide more effective marketing to the different customer groups based on their preferences. For example, analyzing each segment separately can provide valuable insights into each group's likes, dislikes and purchasing behavior, just like we see science fiction, non-fiction and children's books preferences here.

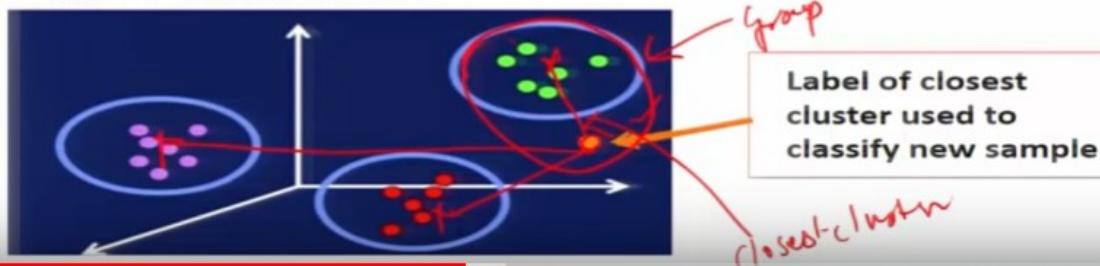


Now, let us see some of the uses of these cluster analysis results. Now, the first we use is called 'Data Segmentation' that is the analysis of each segment can provide an insight that we have already seen or we have already discussed. So, there are several ways that the results of cluster analysis can be used one obvious is data segmentation and the benefits that come from that. So, if we segment your customer base into different type of readers the resulting insight can be used to provide a more effective marketing to different customer groups based on their preferences. For example, analyzing each segment separately can provide when you will insight into each groups likes, dislikes, purchasing behavior, just like we see a science fiction, non-science fiction and childrens book preferences here in this particular picture.

Refer slide time: (16:25)

## Uses of Cluster Results

- Categories for classifying new data ✓
- New sample assigned to closest cluster: Clusters can also be used to classify new data samples. When a new sample is received, like the orange sample here, compute the similarity measure between it and the centers of all clusters, and assign a new sample to the closest cluster. The label of that cluster, manually determined through analysis, is then used to classify the new sample. In our book buyers' preferences example, a new customer can be classified as being either a science fiction, non-fiction or children's books customer depending on which cluster the new customer is most similar to.

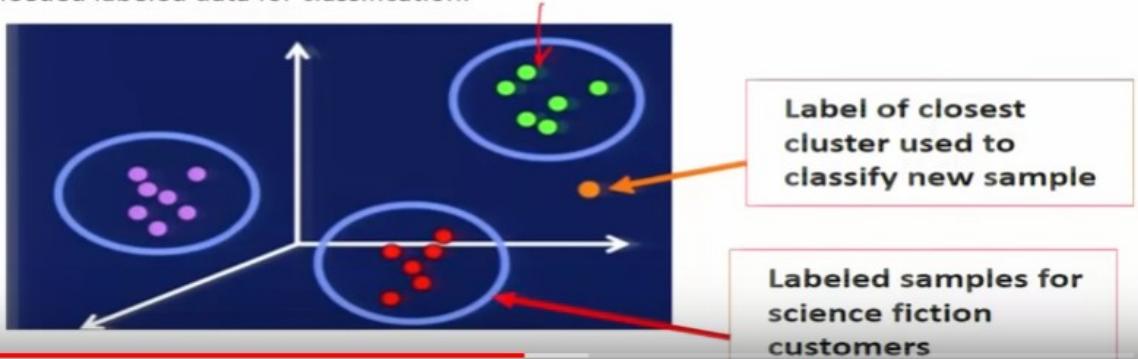


Now, categories of classifying the new data. So, when a new sample assigned to closest cluster then cluster can be used to classify the new data sample. So, when a new sample is received like an orange sample here in this example compute the similarity measure between it and the centers of all the clusters similarities between the between it and the centers of our cluster and assign a new sample to the closest cluster for example if this is the closest cluster in this particular let us assume that this is the closest cluster. So, this will become a new cluster member after that. So, the label of that cluster manually determines through the analysis is then used to classify the new sample. So, in the book buyers preference example a new customer can be classified as being either the science fiction, non-science fiction or a childrens of books customer depending on which cluster, the new customer is similar to. So, this particular cluster will identify this particular user to be a part of this particular cluster group and hence this particular classification will now know that this customer is belongs to this kind of group for example here the groups which we have identified in the previous example is about the readers of science fiction, non-science fiction or a childrens books.

Refer slide time: (18:18)

## Uses of Cluster Results

- **Labeled data for classification**
- **Cluster samples used as labeled data:** Once cluster labels have been determined, samples in each cluster can be used as labeled data for a classification task. The samples would be the input to the classification model. And the cluster label would be the target class for each sample. This process can be used to provide much needed labeled data for classification.

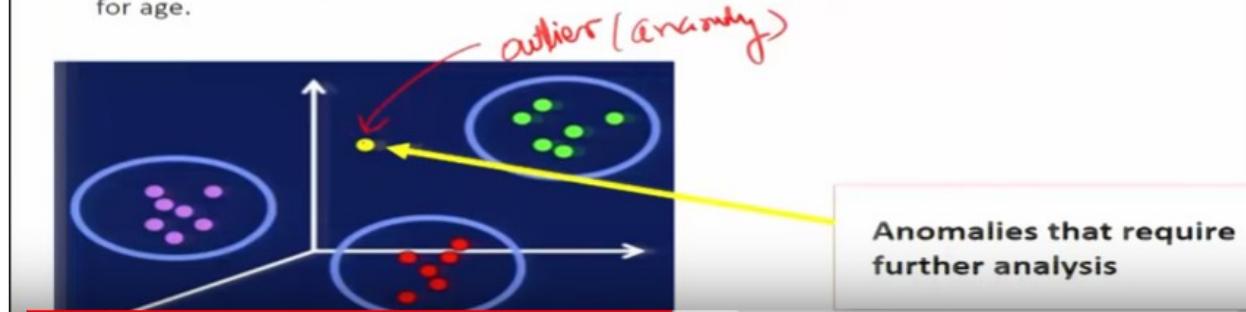


Now, label data for classification. So, cluster samples used as the label data. Now, once the cluster labels have been determined samples in each cluster can be used as label data as for a classification task. The sample would be the input to the classification model and the cluster label would be the target class for each sample this process can be used to provide much needed labeled data for the classification take this particular example here, the label of the closest cluster is used to classify the new sample and the label sample for the science-fiction customers are labeled in this case so after clustering these particular different clusters will now be used to assign the level for each data points and also when a new data points comes it will be getting a new label. And, this label data can be used for supervised learning.

Refer slide time: (19:27)

## Uses of Cluster Results

- **Basis for anomaly detection**
- **Cluster outliers are anomalies:** Yet another use of cluster results is as a basis for anomaly detection. If a sample is very far away, or very different from any of the cluster centers, like the yellow sample here, then that sample is a cluster outlier and can be flagged as an anomaly. However, these anomalies require further analysis. Depending on the application, these anomalies can be considered noise, and should be removed from the data set. An example of this would be a sample with a value of 150 for age.



Now, another basis this cluster can be used cluster analysis can be used as a basis for anomaly detection. So, cluster outliers are basically the anomalies. So, yet another use of cluster results is as a basis for anomaly detection. So, if a sample is very far away or very different from any of the cluster centers like here shown as the yellow data point. Then, the sample is a cluster outlier and can be flagged as the anomaly however these anomalies required further analysis depending on the application these anomalies can be considered noise and should be removed from the data set. An example, of this would be sample with the value 150 for the age so after doing this analysis and doing the anomaly detection the quality of data will be improved and hence the quality of the predictions, accuracy of the predictions will further increase in the machine learning context of analytics.

Refer slide time: (20:50)

## Cluster Analysis Summary

- Organize similar items into groups
- Analyzing clusters often leads to useful insights about data
- Clusters require analysis and interpretation

Now, let us summarize the cluster analysis is that it will organize the similar items into the groups and it will analyze the clusters often leads to a useful insight about the data. And, the cluster required the analysis and interpretation.

Refer slide time: (21:06)

## k-Means Algorithm

- Select k initial centroids (cluster centers)



- Repeat

*Classify data into clusters*

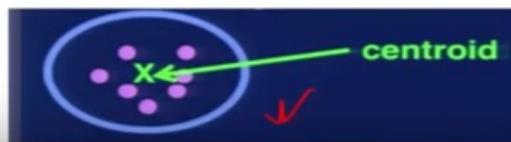
- Assign each sample to closest centroid

*Recompute new centroid*

- Calculate mean of cluster to determine new centroid

*2nd iteration*

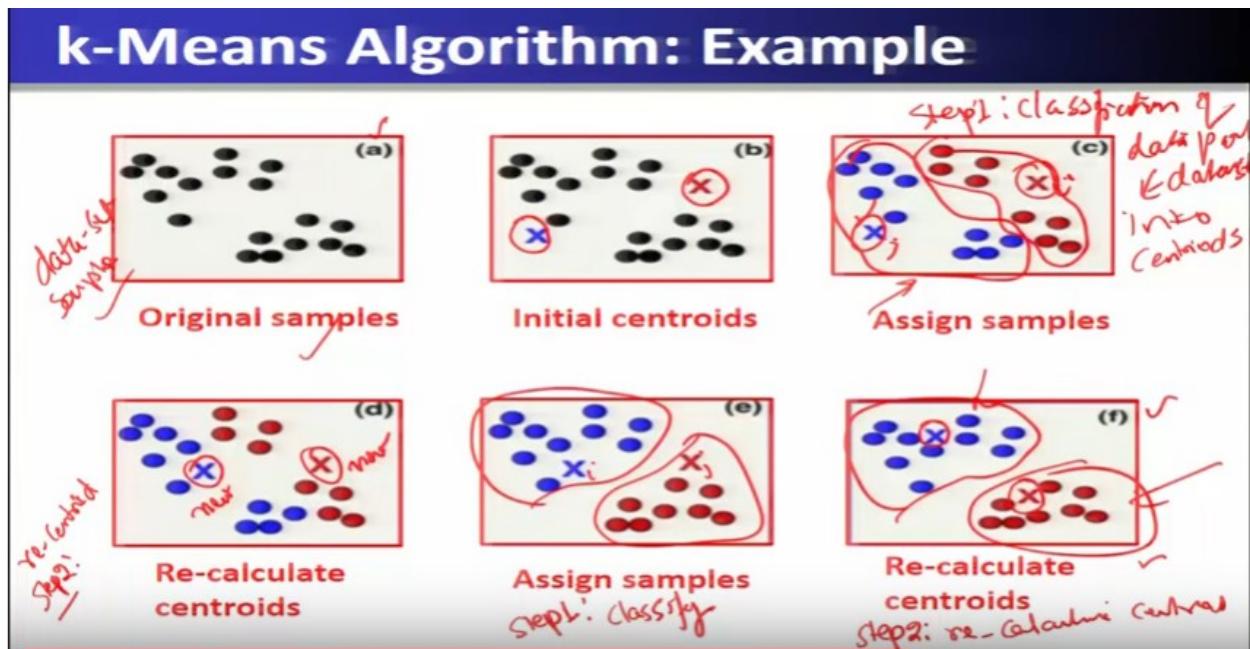
- Until some stopping criterion is reached



Now, let us see the one algorithm that is called k-means algorithm for doing the clustering. And, for performing the cluster analysis as we have seen in the previous slides different applications of it. Now,

here the k-means algorithm initially selects  $k$  initial centroids which are nothing but the cluster centers. And the next step, would be iterative iteration of two steps which says that, assign each sample data sample to the closest centroid and then it will then calculate the means of the clusters to determine the new centroid and this two iterations will repeat until some stopping criteria is met. So, let us define that this particular step is called ‘classify the data into the clusters’ and the second step is called ‘Re Clustering’ which says that now having defined the clusters in step number one. Now, step number two will identify by taking the means of all the data points into a particular cluster and its mean will give a new centroid so it is called ‘Re Clustering’ that is computing the new centroids. Now, these two steps will further repeat and this particular repetition will stop until the stopping criteria is met this is that when we are going to compute the new centroid there is no new there is no much differences are happening further that means the points are almost saturated and fixed into the clusters not going to change. Hence, the stopping of that algorithm is required at that point that means all the plus all the clusters are now being identified and fixed with the proper centroid hence, this otherwise this particular step repeats. Therefore, the clustering of that data points is highly dependent upon the initial positioning or initialization of different centroids.

Refer slide time: (23:59)



So, let us see the example of this k-means, algorithm on a original data set. Let us say, figure 1 or figure a denotes the original data samples data set points this is the original data set sample. Now, let us identify or let us place two clusters these are the initial centroids which are placed into that original data set. Now , then we will perform the second step or we will now this was the initialization that is let us identify two centroids after identifying the two centroids in step one of the algorithm which says that a classification of data points in the data set into these two centroids. So, for example this particular every data point will identify will find out the distance between these different centroids. Let us say,  $i$  and this is  $j$  every data point will compute these distances, this is called ‘Similarity Measures’. So, the

data point which is closer to a particular centroid will join that centroid. So, here you can see in this particular diagram that that these red dots shows that these data points are closer to the to the centroid  $X_i$ , compare to  $X_j$  where whereas all the blue poor depth dots they are closer to  $X_j$  compared to  $X_i$ . Hence, the two different groups that means the initial data set samples is divided into two different groups and they are being identified as the centroid that is  $X_i$  and  $X_j$ . Now, with this particular center now we will execute the step number two which says that which says that it will recompute the centroid or a new centroid will be computed. So, based on these other points mean or an average will be calculated. And, it will identify a new centroid. And, then the iteration repeats for the step number one and step number one will now identify the data points which are closer to this new centroid  $X_i$  and the data points which are closer to the new centroid  $X_j$ . Hence, two different clusters or segments is being identified. Now, in the figure f we see that again we have to calculate the new centroids and now we can see that new centroid is not going to change these membership of these data points hence the stopping criteria will be it work to stop this algorithm at this end. Therefore, these two different data points or a cluster of these two points is being used to identify the insight into this particular data which was given as the original data set. So, they are basically quite similar in their similarity measures.

Refer slide time: (28:29)

## Choosing Initial Centroids

- **Issue:**

Final clusters are sensitive to initial centroids

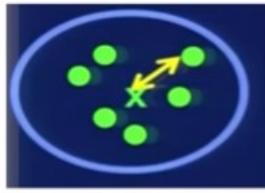
- **Solution:**

Run k-means multiple times with different random initial centroids, and choose best results

Now, we we have seen that choosing the initial centroid is going to affect the final clustering. So, final clustering is very sensitive to the initial centroid so the issue is the final clusters are very sensitive to the initial centroids. So, what is the solution? To run the k-means multiple times with different random initial centroid we have to choose the best results hence we start from that initial centroid after taking the best ones which gives the best possible results.

Refer slide time: (29:07)

## Evaluating Cluster Results



↓  
 $\text{error} = \text{distance between sample \& centroid}$

Squared error =  $\text{error}^2$

Sum of squared errors  
between all samples & centroid

Sum over all clusters → WSSE  
(Within-Cluster Sum  
of Squared Error)

Now, after doing this clustering using k-means now we have to do this we have to evaluate these are not these clusters. And, we have to also interpret the results which are in nothing in the form of clustering. So, for to evaluate that results. Let us, use a term which is called the 'Error'. So, error is the term the difference between the the sample and its centroid. So, for example we can see that we can measure the distance between the centroid and data points for all data points and we have to take the sum of these squared errors. And, this is sum of squared errors between all samples and centroid is being calculated for all over all the clusters hands within the cluster sum of square error is the new term which we are now saying WSSE. So, that means this particular term which we have calculated is used to do this analysis.

Refer slide time: (30:25)

## Using WSSE

- **WSSE1 < WSSE2 → WSSE1 is better numerically**
  - **Caveats:**  
Does not mean that cluster set 1 is more 'correct' than cluster set 2
- Larger values for  $k$  will always reduce WSSE

Now, we can say that a clustering which has WSSE1 and there is another clustering which is WSSE 2. Now, if we compare these two parameters these two values of that clustering and if WSSE1 is less than WSSE 2 then basically WSSE 1 is a better numerically. And, we can see that this does not mean that cluster 1 is more correct than cluster 2. Larger, the values of K will always reduce this value of WSSE. Now, therefore the value of K which we are now going to fix will also has an implication on WSSE.

Refer slide time: (31:10)

# Choosing Value for k

k = ?

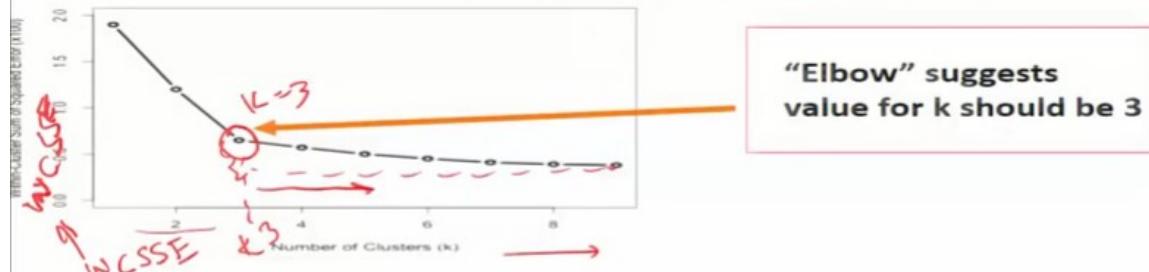
**Approaches:** Choosing the optimal value for k is always a big question in using k-means. There are several methods to determine the value for k.

- **Visualization:** Visualization techniques can be used to determine the dataset to see if there are natural groupings of the samples. Scatter plots and the use of dimensionality reduction are useful here, to visualize the data.
- **Application-Dependent:** A good value for k is application-dependent. So domain knowledge of the application can drive the selection for the value of k. For example, if you want to cluster the types of products customers are purchasing, a natural choice for k might be the number of product categories that you offer. Or k might be selected to represent the geographical locations of respondents to a survey. In which case, a good value for k would be the number of regions you are interested in analyzing.
- **Data-Driven:** There are also data-driven methods for determining the value of k. These methods calculate symmetric for different values of k to determine the best selections of k. One such method is the elbow method.

So, now we will see how to choose this value of K and so on and let us see the different approaches. So, choosing an optimal value for K is always a big question and using the k-means algorithm and there are several methods to determine the value of K. So, the first method is called 'Visualization' technique can be used to determine the data set to see if there are natural groupings of the sample scatter plots and the use of dimensionality. Dimensionality reduction can be useful here to visualize the data. The second approach is called 'Application Dependent' that is a good value for K is application dependent so the domain knowledge of the application can derive the selection for the value of K. For example, if you want to cluster the types of products customers are purchasing a natural choice for K might be the number of products categories you offer or K might be selected to represent the geographic locations of respondent to survey in which case a good value of K would be the number of regions you are interested in analyzing. Third approach is called 'Data-Driven'. So, there are also data-driven methods for determining the value of K. These methods calculate symmetric for different values of K to determine the best selections of K.

Refer slide time: (32:51)

## Elbow Method for Choosing k



The elbow method for determining the value of  $k$  is shown on this plot. As we saw in the previous slide, WSSE, or within-cluster sum of squared error, measures how much data samples deviate from their respective centroids in a set of clustering results. If we plot WSSE for different values for  $k$ , we can see how this error measure changes as a value of  $k$  changes as seen in the plot. The bend in this error curve indicates a drop in gain by adding more clusters. So this elbow in the curve provides a suggestion for a good value of  $k$ .

Note that the elbow can not always be unambiguously determined, especially for complex data. And in many cases, the error curve will not have a clear suggestion for one value, but for multiple values. This can be used as a guideline for the range of values to try for  $k$ .

One such method is called ‘Elbow Method’ which is shown over here for choosing the value of  $K$ . In this elbow method, we can see that this  $W$  within cluster sum of squared errors that is  $w$  WCSSE that we have seen again we are writing WCSSE this term is shown on the vertical axis and on the horizontal side the number of cluster that is the value of  $K$ . So, as we increase the value of  $K$  we can see that it is coming down but beyond this particular point if we increase the value of  $K$  that is the number of clusters we see that this particular reduction in the WSSE is very nominal it is not very much. So, this is the point which is called an elbow point. So, elbow point shows that the value of  $K$  should be this elbow point and this is nothing but a  $k$ ,  $k$  is equal to 3, at 3 we see that the WSSE will not drop much beyond that if we increase the value of  $K$ . So, even if them the number of clusters are more beyond 3 it is not going to affect much on reducing this WSSE hence the value of  $K$  is chosen based on this elbow method. So, in the elbow method for determining the value of  $K$  is shown on this particular plot as we see on the previous slide WSSE or within cluster sum of square or measures how much data sample deviate from their respective centroid in a set of clustering results. So, if we plot W SSE for different values of  $K$  we see that how this error measure changes as the value of  $K$  changes as seen in the plot. The bend, in this error curve indicates a drop in the gain by adding more number of clusters. So, this elbow in the curve provides cessation for a good value of  $K$ . Note, that the elbow cannot always be unambiguously determined especially for a complex data in many cases the error curve will not have the clear solution for any value for multiple values. This can be used as a guideline for trying the value of  $K$ .

Refer slide time: (35:37)

# Stopping Criteria

## When to stop iterating ?

- **No changes to centroids:** How do you know when to stop iterating when using k-means? One obviously stopping criterion is when there are no changes to the centroids. This means that no samples would change cluster assignments. And recalculating the centroids will not result in any changes. So additional iterations will not bring about any more changes to the cluster results.
- **Number of samples changing clusters is below threshold:** The stopping criterion can be relaxed to the second stopping criterion: which is when the number of sample changing clusters is below a certain threshold, say 1% for example. At this point, the clusters are changing by only a few samples, resulting in only minimal changes to the final cluster results. So the algorithm can be stopped here.

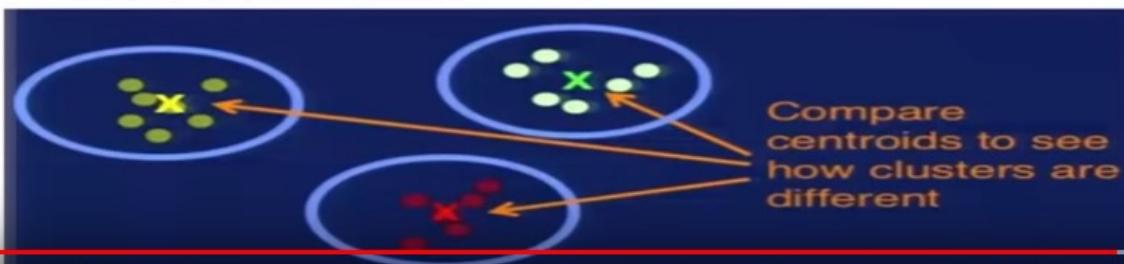
Now, another parameter here in k-means algorithm is about stopping criteria. When, to stop these iterations of that that is two iterations which two steps in an iteration which we are making classification and Recomputing the new centroid. Now, we will stop iterating when no changes to the cluster to the centroid is observed. So, how do you know when to stop iterating when using the k-means? One obviously stopping criteria is when there is no changes in the centroids. This means, that no samples would change the cluster assignments and recalculating the centralized would not result in any change so additional iterations will not bring out any more changes in the results. Second way to stop, these iterations in the k-means clustering is determined using the number of samples which are changing the clusters is below a threshold. So, the stopping criteria can be relaxed to these second stopping criteria which is when the number of sample changing clusters is below a certain threshold say 1% for example in at this point the clusters are changing by only a few samples resulting only minimal changes to the final cluster result. So, the algorithm can be stopped here.

Refer slide time: (36:57)

# Interpreting Results

## Examine cluster centroids

- **How are clusters different ?** At the end of k-means we have a set of clusters, each with a centroid. Each centroid is the mean of the samples assigned to that cluster. You can think of the centroid as a representative sample for that cluster. So to interpret the cluster analysis results, we can examine the cluster centroids. Comparing the values of the variables between the centroids will reveal how different or alike clusters are and provide insights into what each cluster represents. For example, if the value for age is different for different customer clusters, this indicates that the clusters are encoding different customer segments by age, among other variables.



Now, finally interpretation of the results is also going to be very important. Now, examine the cluster centroids how the clusters are different? So, at the end of k-means we have a set of clusters and each with a centroid. Each centroid, is the mean of the samples assigned to that cluster you can think of the centroid is the representative sample for that cluster. So, to interpret the cluster analysis result we can examine the cluster centroids comparing the values of variables between the centroid will reveal how different are alike the clusters are and provide insight into what clusters represent? For example, if the value for the age is different for different for different customers cluster this indicates that the clusters are encoding different customer segments by age and among other variables. So, here we can see here to compare the centroid to see how the clusters are different.

Refer slide time: (38:06)

## K-Means Summary

- **Classic algorithm for cluster analysis**
- **Simple to understand and implement and is efficient**
- **Value of k must be specified**
- **Final clusters are sensitive to initial centroids**

So, let us summarize the k-means so it is a classical algorithm for doing the cluster analysis or the clustering. It is simple to understand and interpret and it is efficient also the value of K here in k-means algorithm must be specified. And, the final clusters are very sensitive to the initial centroid.

**Lecture - 26**  
**Parallel K-means using Map Reduce on Big Data Cluster Analysis**

Parallel k-means using MapReduce on big data cluster analysis we have seen in this particular section you will see the parallel k-means algorithm that means we will use the MapReduce.

Refer slide time :( 0:36)

**Parallel K-means using Map Reduce on Big Data Cluster Analysis**

- Apply Map Reduce on k-Means Algorithm
- Data Parallel k-Means Algorithm via mapreduce
- Big data
- Extract the insights of Big data

▶ ▶ ⏪ 2:14 / 19:29 Big Data Computing Machine Learning Classification Algorithms

We will apply, the MapReduce apply the matter MapReduce on k-means algorithm. Now, important thing here is that this particular method of applying the MapReduce on k-means will make this data parallel algorithm. Now, after doing this data parallel k-means algorithm using MapReduce now we can use this particular algorithm for the analysis of a big data. So, we can use this particular algorithm that is called ‘Parallel k-means’ algorithm using MapReduce with the big data. And, we can extract the insights of big data. So, let us see the more details of how MapReduce can be applied on the k-means to achieve this parallel k-means using MapReduce which is used for the big data analysis.

Refer slide time :( 02:19)

# MapReducing 1 iteration of k-means

~~Step 1~~ **Classify:** Assign observations to closest cluster center

$$z_i \leftarrow \arg \min_j ||\mu_j - x_i||^2$$

**Map:** For each data point, given  $(\{\mu_j\}, x_i)$ , emit  $(z_i, x_i)$

- Recenter:** Revise cluster centers as mean of assigned observations

$$\mu_j = \frac{1}{n_j} \sum_{i: z_i = j} x_i$$

**Reduce:** Average over all points in cluster  $j$  ( $z_i = j$ )

Now, this particular iteration of the k-means which is the step number one which we have seen in the previous slide that is to classify that means all the we have to assign the the data set the data points in the data set to the closest centroid or to the closest cluster Center which is represented by this particular equation let us understand this particular equation. Now, if we are given a set of centroids then for a particular data item  $X_i$  we are calculating the similarity measure and we have to identify that particular centroid which is having that minimum distance. So, this particular arm in this particular formula will identify that particular cluster which is having that closest closest or which is very similar or which is having the least distance or a minimum distance with that data set compared to the other plus other centroids. Hence, this particular  $j$  is the label which will be identified for a particular  $X_i$ . Now, we are now going to get  $Z_i$  as the label or as a centroid for this data point  $X_i$ . Now, the next step in the k-means algorithm which we have seen in the previous slides is called ‘Recenter’ that means after identifying the clusters that means after classifying the data set points into the different clusters. Now, for every cluster we are going to calculate the mean of that particular cluster. So, for every cluster we are going to compute the mean and that will become the new centroid for every cluster this is called ‘Recenter’. And, these two step is going to iterate. So, this these two step will form the first iteration of the MapReduce k-means algorithm. Now, let us use the map function and reduce function which can be applied now on k-means algorithm. Now, the map function would be here for each data point we have to we will identify this particular centroid which is closest to this a point  $X_i$  and let us assume that  $Z_i$  is the closest centroid which is being identified for a data point  $X_i$  and the map function will perform this operation in a data parallel operation as a data parallel operation. And, it will emit this key value pair. So, key will be key will be the the centroid or you can also call it as a label and the value will be that data point data point is in the data set or the data sample. Now, as far as the step number 2 of the first iteration which is called ‘Computing’ the recenter which is nothing but identifying the mean of that particular cluster mean of the cluster points. So, mean of a cluster point will become the new centroid which is nothing but we call it as recenter. Now, as far as the reduced function for this operation will be performed by taking this key value pair from the

previous MapReduce and for that particular that means it will group by that particular centroid all data points and perform the average. So, average of all data points for that but your cluster which is  $Z_i$ .

Refer slide time :( 07:24)

## Classification step as Map

- Classify:** Assign observations to closest cluster center in data parallel operation

```

map([μ₁, μ₂, ..., μₖ, xᵢ])
zᵢ ← arg minⱼ ||μⱼ - xᵢ||₂²
emit(zᵢ, xᵢ)

```

Now, let us see in more details how this map and reduce function is being implemented. So, the first step is called classification or a classifying classify the data points data points in the data set to the closest centroid in a data parallel operation. So, in this particular equation you can see that if you are given the set of centroids. So, for a given data point it will calculate the distance measure the differences and the minimum of that particular difference will be that data will be that centroid which is very closest to the data point compared to all other one and that will become as the label or the centroid which is closest to the data point. So, this way if we perform this activity for all data sets in data parallel operation then we can specify using my function. So, map function will have the input as the set of all data all all centroids. So, this is the set of all centroids ( $\mu_1, \mu_2$ ) these are all let us seek a different centroids are given in the map function and  $X_i$  is the data point. Now, this  $X_i$  will be computing the the similarity measure or that the distance measure with all the centroids and the minimum distance centroid let us assume it is  $Z$  which will be identified which will be assigned to  $Z_i$  and the map function will omit that  $Z_i$  and this  $X_i$ . So, the key which this map function will emit is the centroid ID which is the closest to this data point and the value will be that data point.

Refer slide time :( 10:18)

## Recenter step as Reduce

- **Recenter:** Revise cluster centers as mean of assigned observations

*Centroid*

$$\mu_j = \frac{1}{n_j} \sum_{i: z_i = k} x_i$$

```

reduce(j, x_in_cluster j : [x1, x3, ...,])
 sum = 0
 count = 0
 for x in x_in_cluster j
 sum += x
 count += 1
 emit(j, sum/count)

```

*set of datapoint assigned to centroid*

*normalization or mean*

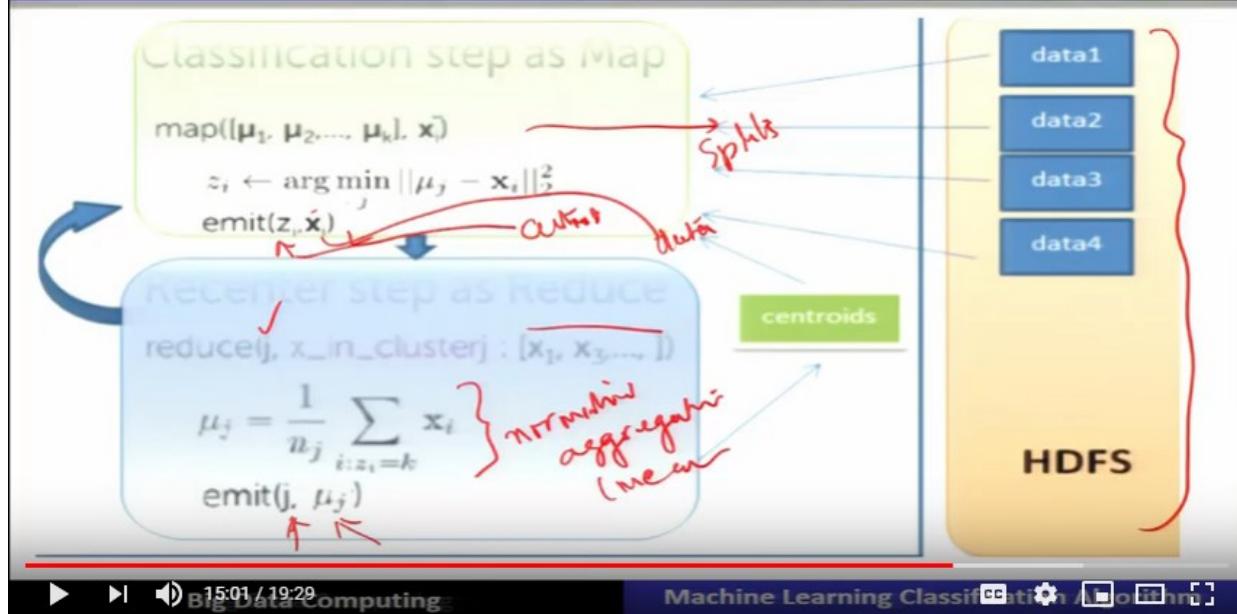
*Key - Centroid  
Value → new\_Centroid*

▶ ▶ ⏪ 13:42 / 19:29 Big Data Computing Machine Learning Classification Algorithms

Now, let us see the reduce function which is nothing but computing the recenter. Now, after doing the mean or it will identify the mean of all the data points within a particular data within a particular cluster and after doing the normalization summation and the normalization it will identify the new centroid. So, for all the cluster ID all the clusters there will be are center or computing the the summation of all the data points and normalizing it that is nothing but the calculating the mean for all things. Let, us see how it will be done performed using a reduced function. Now, as we have seen from the map function that there will be key value pair which will be emit from the map. So, map will emit the key value pair. So, key will be that centroidID let us say this is the centroid and this value will be the set of all set of data points which are assigned to the centroid J. So, what it will do is it will compute it will find out the total it will find out the summation of all data points X within it within the clusters and then it will do the normalization. So, here for that there will be a for loop. So, it will iterate through all the all the data points in that particular set of that cluster and it will do this particular summation that means it will do the summation of it this particular equation it will operate in this way and then it will also do a counting and finally the count value will contain the total number of points which are doing which are being summed up for normalization. So, now it will do this mean or the normalization normalized aggregation is being done normalized aggregation or it will be computed the mean for every cluster. And, it will emit this is the key key means it will emit the centroid and value will be the normalized value which is a new centroid. So, after calculating the new centroid now again it will go to the next iteration and this iteration will repeat. So, we have seen that how the map and reduce has been implementing this k-means algorithm.

Refer slide time :( 13:46)

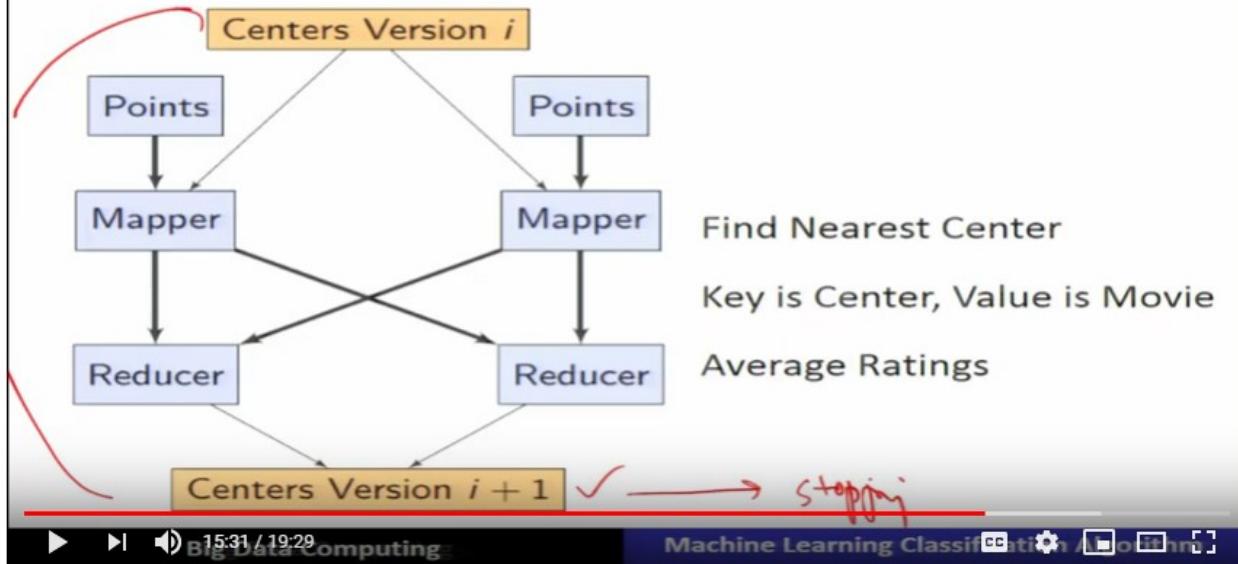
## Distributed KMeans Iterative Clustering



Let us see the same thing in the form of a picture. So, you can see that this is the data set and this particular map function will be performed on different data splits. So, map function will be applied on a different data splits and this will map will emit the key value pair. So, key will be the centroid and for all the values of this data points it will emit reduce function what it will do it will now group by key. And, it will collect all such data points which belong to a particular cluster. And, it will form the normalized aggregation that is nothing but it will calculate the mean of it. So, for that particular centroid  $J$  it will compute the mean that is nothing but the new centroid and this particular way this k-means algorithm will iterate to the next step until it converges.

Refer slide time :( 15:06)

## Distributed KMeans Iterative Clustering



So, this particular new data centers centers once it is being identified then again it will go and classify all the data set accordingly. And, so on this particular iterations keeps on iterating unless there is a stopping criteria and once the stopping criteria is met then these iterations will stop and the algorithm.

Refer slide time :( 15:36)

## Summary of Parallel k-means using MapReduce

- **Map: classification step;**
  - data parallel over data points

(key, value)

- **Reduce: recompute means;**
  - data parallel over centers

{ new centroid  
based on mean of  
data-point in  
cluster (i) }

▶ ▶ ⏪ 17:01 / 19:29 Big Data Computing Machine Learning Classification Algorithms

will terminate in summary we have seen the parallel k-means algorithm which is nothing but an implementation of MapReduce for k-means algorithm. So, we have also seen that the two steps that is the using map and reduce. We, have implemented we have seen the implementation using map function for the classification step wherein the data parallel operation is performed or different data points to classify the data points into the key value pair. Now, second operation which is called a ‘Reduce Operation’ which is nothing but to come to recomputed the means are the new centroids the new centroids are computed based on the mean of data points in the clusters which is identified in the step number one. So, after identifying it then it will go to the next iteration and so on. So, these iteration these two operations that is the classification and recomputation we have seen how using map and reduce is being implemented.

Refer slide time :( 17:05)

## Some practical considerations

- k-means needs an iterative version of MapReduce  
Not standard formulation

Spark.

- Mapper needs to get data point and all centers

A lot of data!

Better implementation:

mapper gets many data points

Spark



Now, we see that this k-means algorithm is an iterative version is an iterative algorithm and which requires an iterative version of a MapReduce. And, in the previous one chance of Hadoop and MapReduce version 1 we have we have there were several new implementation of iterative MapReduce. Now, with the new version of Hadoop which internally supports the iterative MapReduce in is spark this particular iteration is not going to be a problematic and is going to be very straight forward that we will see in further lectures more detail that this is not that we can use it iterative version of MapReduce using some other means such as is spark. Now, another practical consideration is about the mapper which needs to get the data points and all the data and all the centers lot of data is going to be moved in the old or in the previous version of MapReduce but now in a newer implementation like in spark and all. So, in memory configuration is being used to store the data and over the iterations all these are implemented and handled for better optimization and efficient implementation.

Refer slide time :( 18:36)

# Conclusion

- In this lecture, we have given an overview of cluster analysis and also discussed machine learning classification algorithm k-means using Mapreduce for big data analytics

Conclusion in this lecture: We have given an overview of the parallel k-means algorithm as an implementation of Map Reduce and which is one of the very important unsupervised machine learning for different big data analytics application. Thank you.

**Lecture - 27**

**Decision Trees for Big Data Analytics**

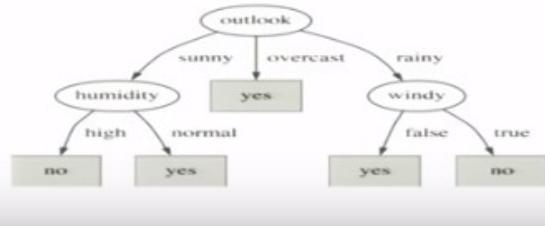
Decision trees for big data analytics.

Refer Slide Time (0:18)

## Preface

### Content of this Lecture:

- In this lecture, we will discuss Decision Trees for Big Data Analytics and also discuss a case study of medical application using a Decision Tree in Spark ML



Preface-Content of this lecture: In this lecture we will discuss decision trees for big data analytics. We will also cover a case study of a medical application using decision trees in Spark ML.

Refer Slide Time (0:37)

## Decision Trees

Before, that let us understand about the basics of decision trees.

Refer Slide Time (0:42)

## Predict if Sachin will play cricket

Consider the following data set our task is to predict if Sachin is going to play cricket on a given day. We have observed Sachin over a number of days and recorded various things that might influence his decision to play cricket so we looked at what kind of weather it is. Is it sunny or is it raining humidity is it high as normal is it windy So this is our training set and these are the examples that we are going to build a classifier from.

Training examples: **9 yes / 5 no**

| Day | Outlook  | Humidity | Wind   | Play |
|-----|----------|----------|--------|------|
| D1  | Sunny    | High     | Weak   | No   |
| D2  | Sunny    | High     | Strong | No   |
| D3  | Overcast | High     | Weak   | Yes  |
| D4  | Rain     | High     | Weak   | Yes  |
| D5  | Rain     | Normal   | Weak   | Yes  |
| D6  | Rain     | Normal   | Strong | No   |
| D7  | Overcast | Normal   | Strong | Yes  |
| D8  | Sunny    | High     | Weak   | No   |
| D9  | Sunny    | Normal   | Weak   | Yes  |
| D10 | Rain     | Normal   | Weak   | Yes  |
| D11 | Sunny    | Normal   | Strong | Yes  |
| D12 | Overcast | High     | Strong | Yes  |
| D13 | Overcast | Normal   | Weak   | Yes  |
| D14 | Rain     | High     | Strong | No   |

Refer Slide Time (0:46)

## Decision Trees

So, decision tree is a tree which is nothing but an acyclic graph which has the root node and other nodes. This is called the root node of a tree and these are all called internal nodes of a tree and this is called the leaf. So, the square boxes represent the node which is called a leaf node in the decision tree. The predictions are there at the leaf nodes and the internal nodes and the root nodes they are called the

decision nodes. Now, given a data set we are going to construct this particular decision tree using the decision tree algorithm and traversing the tree we can predict for a given query. So, whenever a new data set we get we will be able to handle this particular data set for doing the predictions. So, details we are going to see in this discussion and then how this decision tree is going to be useful for doing the big data analytics also we will cover through an example.

Refer Slide Time (2:39)

## Predict if Sachin will play cricket

Training examples: 9 yes / 5 no

| Day | Outlook  | Humidity | Wind   | Play |
|-----|----------|----------|--------|------|
| D1  | Sunny    | High     | Weak   | No   |
| D2  | Sunny    | High     | Strong | No   |
| D3  | Overcast | High     | Weak   | Yes  |
| D4  | Rain     | High     | Weak   | Yes  |
| D5  | Rain     | Normal   | Weak   | Yes  |
| D6  | Rain     | Normal   | Strong | No   |
| D7  | Overcast | Normal   | Strong | Yes  |
| D8  | Sunny    | High     | Weak   | No   |
| D9  | Sunny    | Normal   | Weak   | Yes  |
| D10 | Rain     | Normal   | Weak   | Yes  |
| D11 | Sunny    | Normal   | Strong | Yes  |
| D12 | Overcast | High     | Strong | Yes  |
| D13 | Overcast | Normal   | Weak   | Yes  |
| D14 | Rain     | High     | Strong | No   |

New data:

|     |        |        |        |   |
|-----|--------|--------|--------|---|
| D15 | Rain ✓ | High ✓ | Weak ✓ | ? |
|-----|--------|--------|--------|---|

Let us see, simple data set and let us see what kind of predictions this decision tree will be able to make. So, to understand in a very simple manner. Let us consider, this data set about a query. So, the queries that we want to predict if searching will play the cricket or not to answer this particular query that means to get the prediction whether sachin will play the cricket or not. We are going to have observed the details of Sachin playing and also the weather conditions over 15 days. So, day 1 to day 14 or 14days wherein we have considered that what is the outlook of the weather if it is sunny, overcast, raining and the humidity of every day that is whether it is high, normal or the wind speed whether it is weak or a strong and we have also seen that in these conditions whether that player such in has played or not. So, on day one it has not played he has not played day three he has played well. The weather was outlook was overcast humidity was high and wind was weak and so on. So, we have observed the 14 days of these different parameters and also we have seen whether the player Sachin has played under these conditions or not. Now, using this particular data set, can we predict whether Sachin will play to the days which are a new day or a next day or not so decision tree. Let us see, how it is going to be useful? So, consider this data set and our task is to predict if the Sachin is going to play the cricket on a given day. So, we have observed the playing the game playing of Sachin our

number of days. In this example, we have monitored or we have observed 14 days and recorded various conditions that might influence his decision to play the cricket. So, under different weather conditions. So, is it sunny or is it raining or it is having humidity and then in that case or it is having a normal windy temperature what conditions are required and necessary and to predict whether certain will play or not? Even, for this is a small or a simple data set this becomes quite complex. So, let us see how we are going to simplify using decision tree for this particular purpose. Now, one thing is that this particular data set is basically observation of the last 14 days. So, that becomes the training set and using these examples can be predict this particular decision of a search in whether he will play the game or not? So, that means the query will be that on day 15<sup>th</sup> if outlook is, raining humidity is high and wind is weak whether sachin will play or not. So, from this data set also it seems that it is not easy to get or understand this entire data set and come out with the decision to predict or not? But, let us understand this particular data set because decision tree is the technique which will understand these different parts of the data set or the behaviour of that player Sachin it will understand and based on that it will be just the predictions. So, on day 15 if it is raining and the humidity is high and the wind speed is not windy slow or a week wind is capacity then weather Sachin is going to play or not and by looking to this particular data and this kind it will become very hard to decide. The right kind of predictions, whether Sachin is playing in other days on one of these parameters and how this kind of condition whether he will be playing or not? It is very difficult to predict.

Refer Slide Time (7:44)

## Predict if Sachin will play cricket

- Hard to guess ✓
- Try to understand when Sachin plays ✓
- Divide & conquer:
  - Split into subsets
  - Are they pure ? (all yes of all no)
  - If yes: stop
  - If not: repeat
- See which subset new data falls into

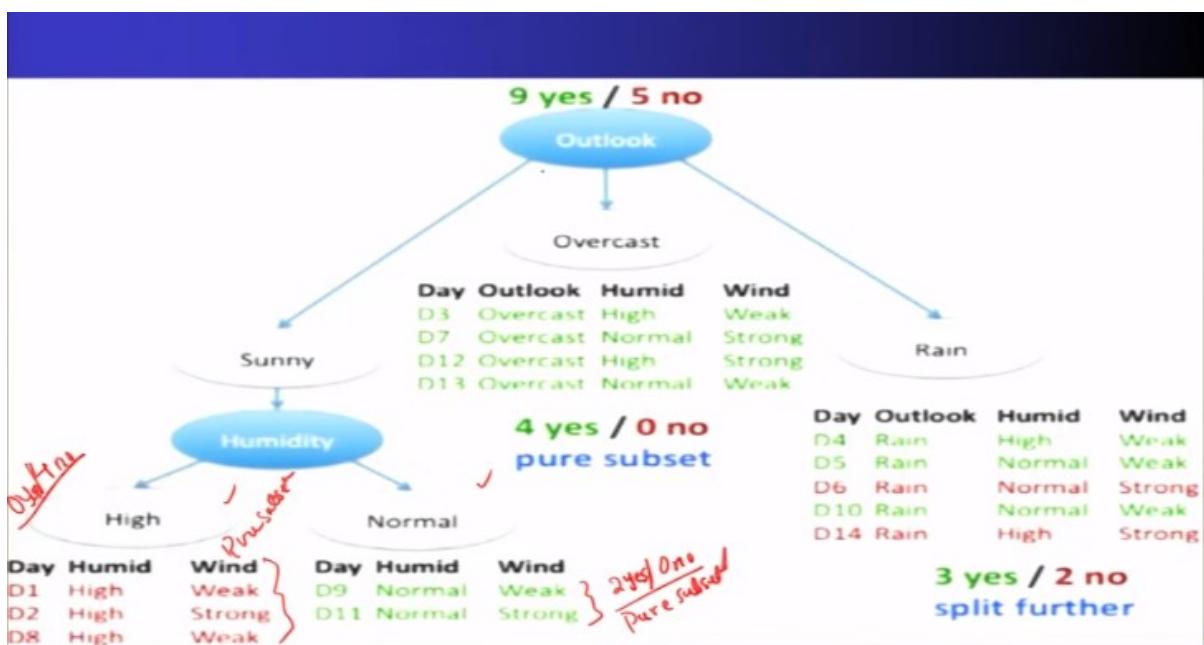
| Training examples: 9 yes ✓ / 5 no ✓ |          |          |        |      |
|-------------------------------------|----------|----------|--------|------|
| Day                                 | Outlook  | Humidity | Wind   | Play |
| D1                                  | Sunny    | High     | Weak   | No   |
| D2                                  | Sunny    | High     | Strong | No   |
| D3                                  | Overcast | High     | Weak   | Yes  |
| D4                                  | Rain     | High     | Weak   | Yes  |
| D5                                  | Rain     | Normal   | Weak   | Yes  |
| D6                                  | Rain     | Normal   | Strong | No   |
| D7                                  | Overcast | Normal   | Strong | Yes  |
| D8                                  | Sunny    | High     | Weak   | No   |
| D9                                  | Sunny    | Normal   | Weak   | Yes  |
| D10                                 | Rain     | Normal   | Weak   | Yes  |
| D11                                 | Sunny    | Normal   | Strong | Yes  |
| D12                                 | Overcast | High     | Strong | Yes  |
| D13                                 | Overcast | Normal   | Weak   | Yes  |
| D14                                 | Rain     | High     | Strong | No   |

New data:  
D15      Rain      High      Weak      ? ✓

Let us see, how in decision tree we are going to do this kind of predictions? So, as we have seen it is very hard to guess whether Sachin will play on day 15 and but for this we have to understand when

sachin play? What are the condition when certain will be when Sachin is playing? So, there are 9 different samples when Sachin has played and under different conditions 9 and these 9 different conditions when Sachin has played is required as the training to understand when he has played and where there are 5 different condition when he has not played in those weather conditions. Now, we will to understand this behaviour of a player Sachin out of this data set which is an observation of last 14 days. We can do this by splitting into the subset into subsets of different classes and then we can extract this information and the the decision information we can then basically use this information for predicting a new data set.

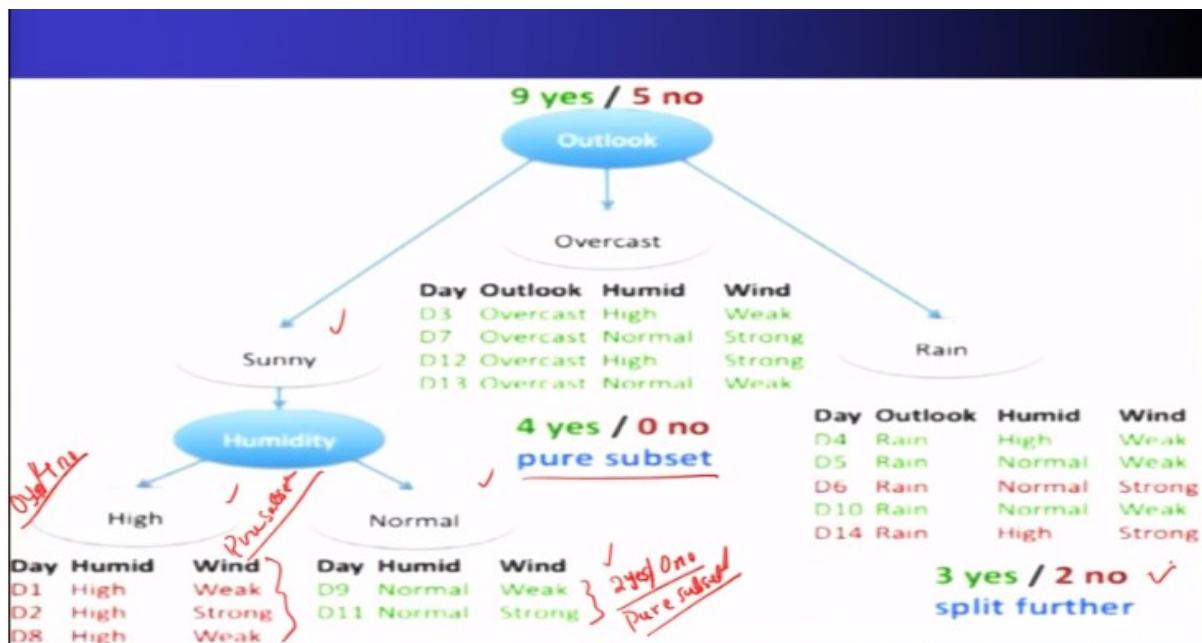
Refer Slide Time (9:07)



So, let us understand more details of this entire data set which is given. So, if we classify this data set based on the outlook parameter then there are 3 different outlook conditions. One is sunny then overcast and raining and these subsets which we are going to get in these different conditions. So, we basically have obtained three different subsets. Now, out of these three different subsets we see that in when the outlook is overcast we can see that here. There are 4 different entries or 4 different days are there when this becomes yes and there is no entry which is having 0 no's. So, hence this kind of subset is called a pure subset which has all yes. When, the outlook sunny then we can see this particular subset which has three no's and two yes. So, this kind of subset is called impure subset. That means, it is mixed whether to are saying yes when the outlook is sunny then Sachin has played in two of these conditions when the humid is normal and the event is a weak or strong and in three different cases he has in played. So, this kind of subset is called impure subset. Similarly, when the outlook is rain then there are three different cases when Sachin has played and two different cases

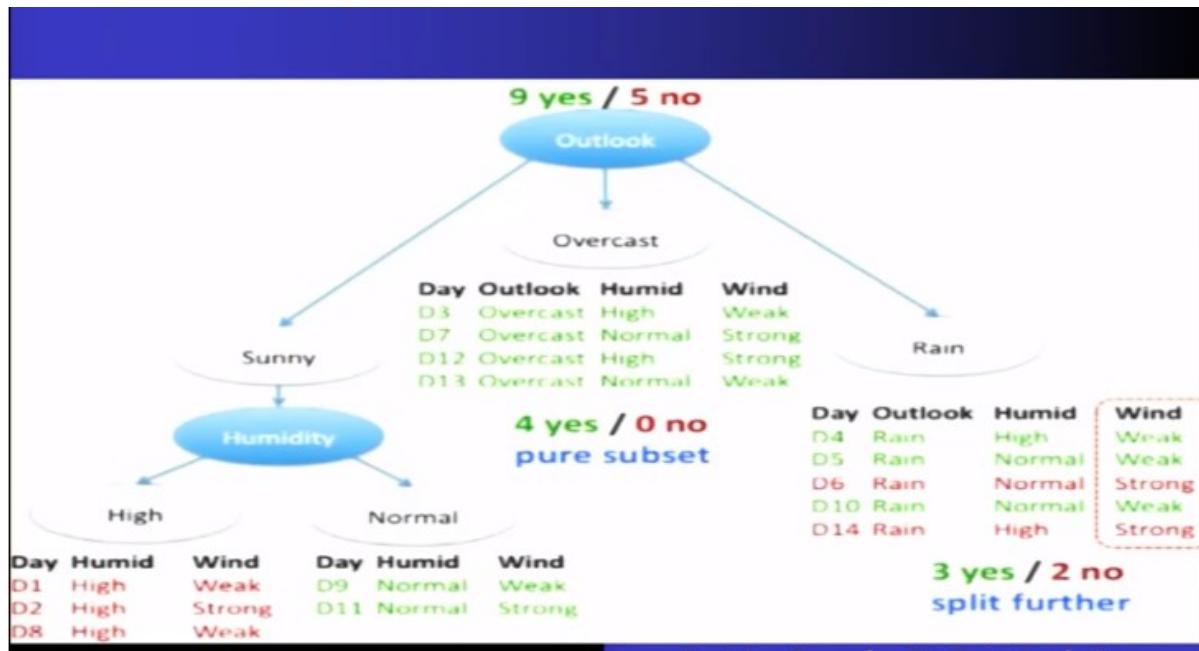
when Sachin has not played. So, hence this becomes an impure subset. Now, we can further split these impure subsets and pure subsets need not to be splitted further.

Refer Slide Time (11:28)



So, now we can see that we can split so that now we have splitted this outlook sunny based on the humidity when the humidity is high or normal. So, what we can see here that when the humidity is high all three cases the Sachin has not played. So, 4 no and 0 yes. Hence, this is called a pure subset whereas when the humidity is normal then there are 2 different in this subset that means 2 yes is there and there is 0 no. Hence, this is also called a pure subset. So, when it is a pure subset then decision can be easily made. For example, when the outlook is sunny and the humidity is normal then Sachin has always played. So, whenever a data of this category comes we can decide about this. So, here also when under this split humidity. We, have obtained both the pure subsets of drop out of this split of sunny. Now, then let us see the rain when the outlook is rain then it has basically a mixed or impure subset which can be splitted further.

Refer Slide Time (13:00)

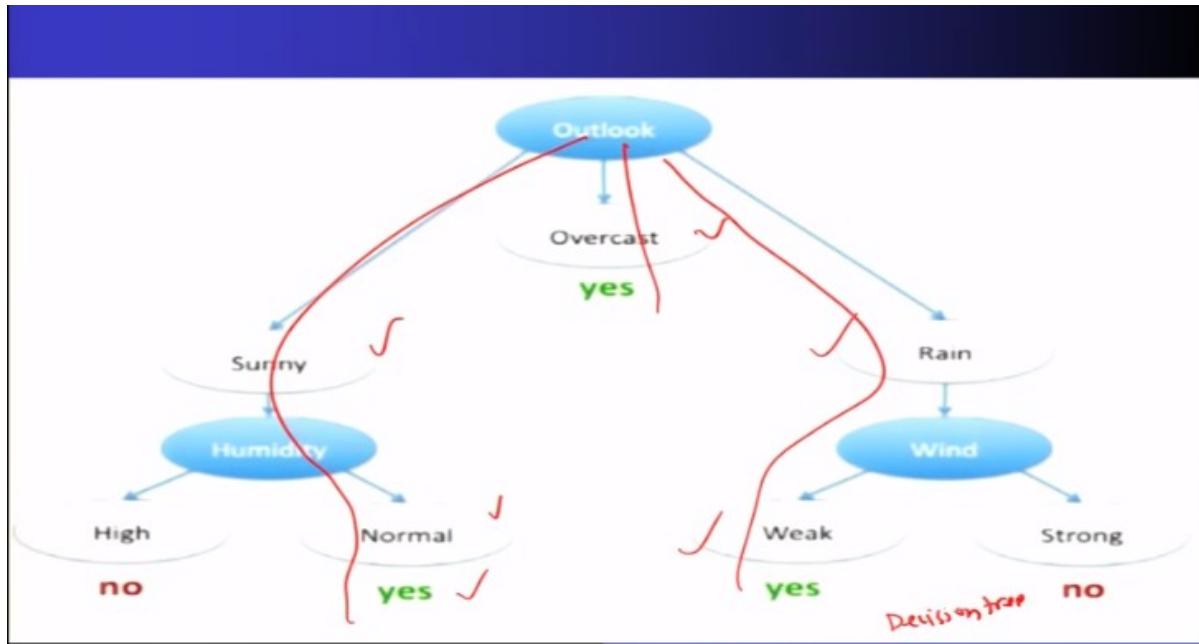


Refer Slide time (13:02)



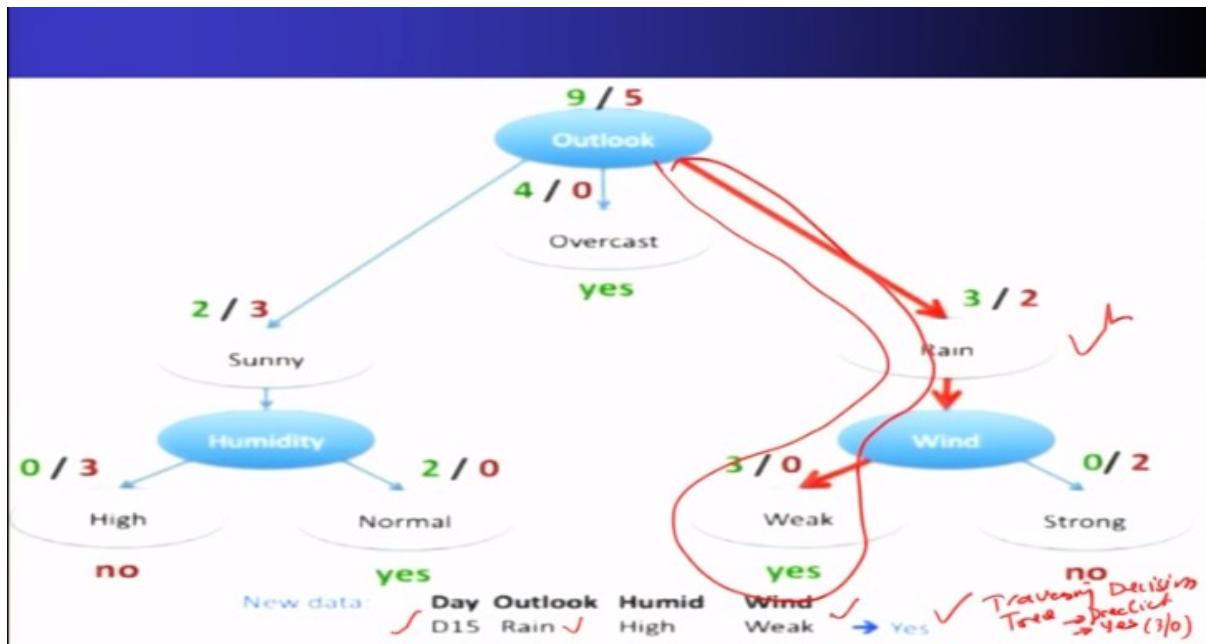
So, if we split further on this wind what we obtain is that when the outlook is rain and the wind is weak then we have got 3 different subsets. In this subset, the 3 yes is there. 3 times he has played and 0 times it he has not played. So, this is also called a pure subset. Now, as far as when the wind is strong then we have obtained a subset where the zero yes is there and two no are there and this is also called pure subsets. Now, there is no further division or split required and we have constructed decision tree of this use case.

Refer Slide Time (14:01)



Now, if we summarize this particular decision that decision tree which is constructed out of that dataset. We have obtained, that if when the humidity is normal and when outlook is sunny then always that player Sachin hence played where the outlook is overcast in Sachin has played and when the wind is weak and the outlook is rain then also Sachin has played. So, there are three different cases when Sachin has played and which is shown over here. In, all other cases Sachin has not played.

Refer Slide Time (14:56)



Now, not only this particular decision tree has yes and no cases but it also has how many cases which are having yes. So, it will add to the confidence on this particular decision using these parameters. So, let us see the query which was given to us on day 15 that if the outlook is rain and outlook is rain and humidity is high and wind is weak. When, the outlook is rain and and the wind is weak. Obviously, it is going to play in this particular case. So, we are going to say yes. Hence, the decision is yes in this particular case using the traversing of the decision tree. So, by traversing the decision tree we can predict that Sachin will play and the confidence with which we are going to predict is 3 by 0. So, that means with the high confidence this particular prediction can be made.

Refer Slide Time (16:23)

## ID3 Algorithm *for Decision Tree*

- In decision tree learning, ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset. ID3 is the precursor to the C4.5 algorithm, and is typically used in the machine learning and natural language processing domains.
- Split (node, {examples}):
  1. A  $\leftarrow$  the best attribute for splitting the {examples}
  2. Decision attribute for this node  $\leftarrow$  A
  3. For each value of A, create new child node
  4. Split training {examples} to child nodes
  5. For each child node / subset:
    - if subset is pure: STOP
    - else: Split (child\_node, {subset})
- Ross Quinlan (ID3: 1986), (C4.5:1993)
- Breimanetal (CaRT:1984) from statistics

Now, as far as the decision trees are concerned we are going to discuss one such algorithm which is called ID3 algorithm for decision tree. So, this algorithm was invented by Ross Quinlan used to generate the decision tree from the data set ID3 is the precursor to see 4.5 algorithm and is typically used in the machine learning domain. So, let us see how the split of a node is done in ID3. Now, let us consider A which is assigned to the best attribute for the splitting and the decision attribute for this particular node is assigned from A. For each value of A let us create a new child node. Now, split the training examples to the child node and for each child node or a subset if the subset is pure then stop splitting further else we have to split the node child into the subsets further. Now, Ross Quinlan which has given this ID3 in 1986 and also C4.5 in 1993 and Breinmanetal has given CaRT in 1984 from statistics about this entry algorithm.

Refer Slide Time (18:17)

## Which attribute to split on ?



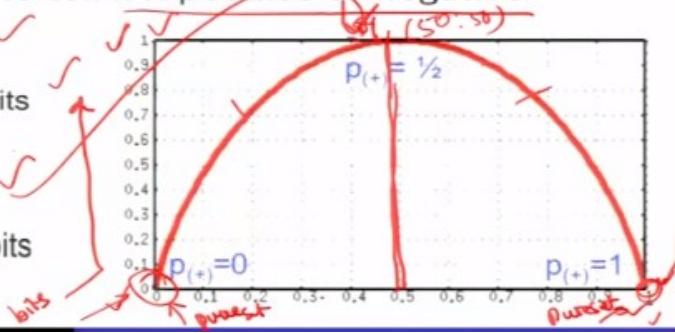
- Want to measure “purity” of the split
  - More certain about Yes/No after the split
    - pure set (4 yes / 0 no) → completely certain (100%) ✓ ↗
    - impure (3 yes / 3 no) → completely uncertain (50%) ✓
  - Can't use  $P(\text{"yes"} | \text{set})$ :
    - must be symmetric: 4 yes / 0 no as pure as 0 yes / 4 no  
Pure subsets (100% correct)

Now, the question is which attribute to split on. So, let us see that we have seen that there are three different attributes in that dataset. So, which one we are going to use to split the attributes on that particular question is going to be that is the split decision which attribute is split on is a decision which we are going to see how decision tree is going is taking here in this case. Now, here let us see that it wants to measure the purity of the split and this is the way it is going to take the decision of which attribute to is split on. So, based on the purity of the split the decision is to be evaluated. So, more certain about yes or no after the split. So, for example if we have a pure set where for yes is there and 0 no are there then we can say with hundred percentage certainty that this is a pure and pure set and the decision is very certain. Similarly, if the set is impure where three yes is there and three no are there then the decision which we are going to take is having completely uncertain that is 50% chance of making the decision correct is basically is there. So, this way we are going to measure using how many yes is there whether the set is pure or impure and this is going to be the measure to go for the split. Now, so this particular must be the use of this probability must be the symmetric that is four yes and 0 no as pure as 0 yes and 4 no. So, that means whether it is all yes or whether it is all no are called pure subsets wherein the decision is 100% certain to be correct. Whereas, if the set is impure then it is uncertain that the decision is correct. So, we are going to evaluate this measure of purity to split on.

Refer Slide Time (21:05)

# Entropy

- Entropy:  $H(S) = -p(+)\log_2 p(+) - p(-)\log_2 p(-)$  bits
  - $S \dots$  subset of training examples ✓
  - $p(+)/p(-) \dots$  % of positive/negative examples in  $S$
- Interpretation: assume item  $X$  belongs to  $S$  ✓
  - How many bits need to tell if  $X$  positive or negative
- Impure (3 yes / 3 no): ✓
 
$$H(S) = -\frac{3}{6}\log_2 \frac{3}{6} - \frac{3}{6}\log_2 \frac{3}{6} = 1 \text{ bits}$$
- Pure set (4 yes / 0 no): ✓
 
$$H(S) = -\frac{4}{4}\log_2 \frac{4}{4} - \frac{0}{4}\log_2 \frac{0}{4} = 0 \text{ bits}$$



Now, this decision tree uses different measures for make the decision whether to split or not? One such way is called entropy that we are going to discuss. How using entropy, we can decide whether to split further or to not to split? So, entropy is represented by HS here and there are two variables. Let us say,  $P(+)$  and  $P(-)$ . So, that is  $-p \text{ bits}$ . This is nothing but the number of bits. So, if we calculate the entropy of a particular set of training examples then we will get the bits. So, this the interpolation of the bits we are going to use to infer whether to split or not how this is all done we are going to see now. Now  $p \text{ bits} \dots$  % of the positive and negative examples we are going to evaluate. Now, interpolation is let us assume an item  $X$  belongs to a set  $S$  and we have to ensure or basically estimate how many bits are needed to tell if  $X$  is positive or the negative? So, for example if it is impure set of three 3 yes / 3 no then if we calculate the entropy of this impure set  $S$  then it comes out to be what bit? So, one bit if it is the entropy is one bit then it is highly uncertain for the decision whether it is yes or no. So, the number of bits which basically will represent it here it is 1. So, 1 means it is highly uncertain in this particular case. Now, if it is a pure set let us hear if it is 4 yes and 0 no and we want to find out the entropy of that set. So, entropy will be zero in this case. So, if the set is pure the entropy will be zero bits. So, zero bits means the decision is very certain and if it is uncertain that is highly uncertain then the bit will be one. Now, and rest of the decisions are in between 0 and 1 that is what we have seen. Now, so this particular diagram shows that if the number of bits is 1 if the number of bit is 1 here in this case if it is so if it is 1 then the the decision will be half 50% chance that the decision is going to be correct 50% chance. So, it will be a 50-50 that means 50-50 and if all are positive or all are negative then the number of bits which is required is 0 here in this case. For example, here in the pure set all are all are yes and 0 are no. Similarly, another pure set when all are all are yes all are no and zero yes then also it is a pure set. In, both the cases the number of bits here this represents the number of bits in both the cases the number of bits are zero hence, this is going to be a pure set and

this is also going to be a pure set and this is the impure set condition and so basically either it is pure set then it will take zero bits or it is highly uncertain then it will take one bit. Whereas, all other conditions are lying on either on the ellipse. On the left side or on the right side where the number of bits is varying between 1 to 0.5.

Refer Slide Time (26:00)

## Information Gain

- Want many items in pure sets
- Expected drop in entropy after split:

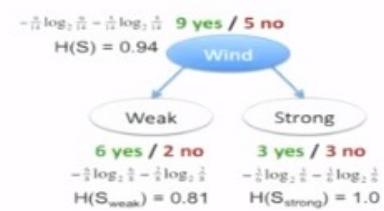
$$Gain(S, A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

$V$  ... possible values of A  
 $S$  ... set of examples {X}  
 $S_V$  ... subset where  $X_A = V$

- Mutual Information  
-between attribute A and class labels of S

Gain (S, Wind)

$$\begin{aligned} &= H(S) - 8/14 * H(S_{weak}) - 6/14 * H(S_{strong}) \\ &= 0.94 - 8/14 * 0.81 - 6/14 * 1.0 \\ &= 0.049 \end{aligned}$$

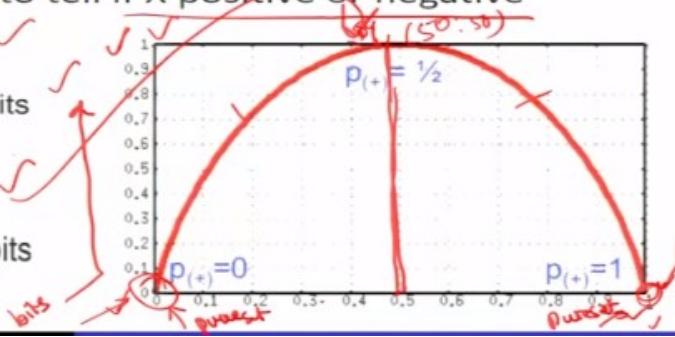


So, with this we have understood that entropy can be useful to estimate whether we can go for the split or not? So, if it is a pure set then entropy the size of the entropy is zero that means we are not going to a split. If, it is a highly uncertain that is fifty percentage are saying yes and fifty percentage are saying no. Then, the value of entropy will become half. So, then entropy will become 1 that is highly uncertain and all other in between which are lying in between. So, their values are between 0 and 1 that we have seen in the previous example.

Refer Slide Time (26:50)

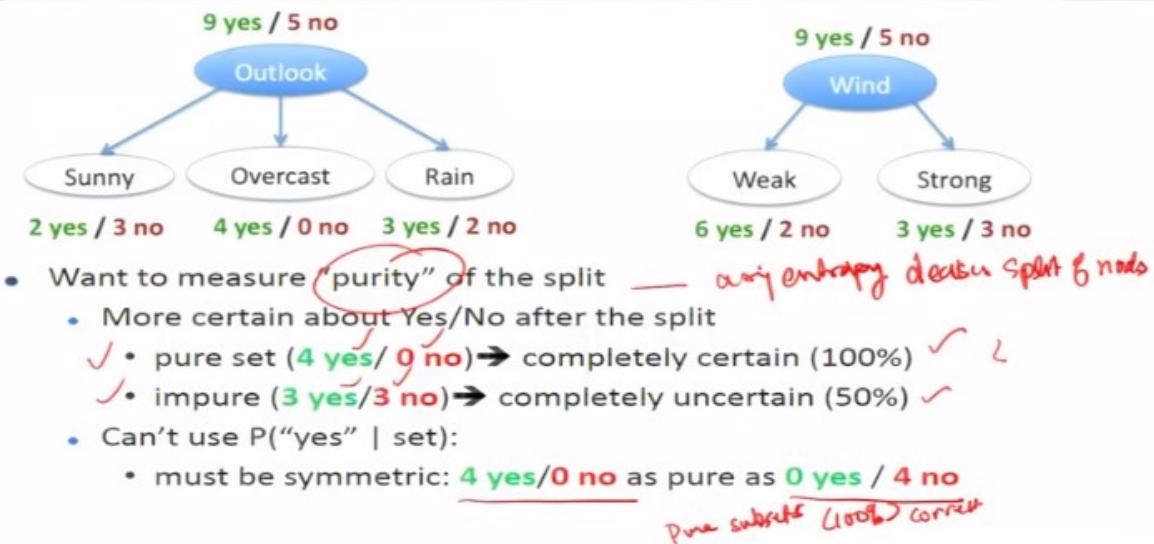
## Entropy

- Entropy:  $H(S) = - p_{(+)} \log_2 p_{(+)} - p_{(-)} \log_2 p_{(-)}$  bits
  - $S$  ... subset of training examples ✓
  - $p_{(+)} / p_{(-}$  ... % of positive/negative examples in  $S$
- Interpretation: assume item  $X$  belongs to  $S$  ✓
  - How many bits need to tell if  $X$  positive or negative
- Impure (3 yes / 3 no): ✓
 
$$H(S) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1 \text{ bits}$$
- Pure set (4 yes / 0 no): ✓
 
$$H(S) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0 \text{ bits}$$



Refer Slide Time (26:51)

## Which attribute to split on ?

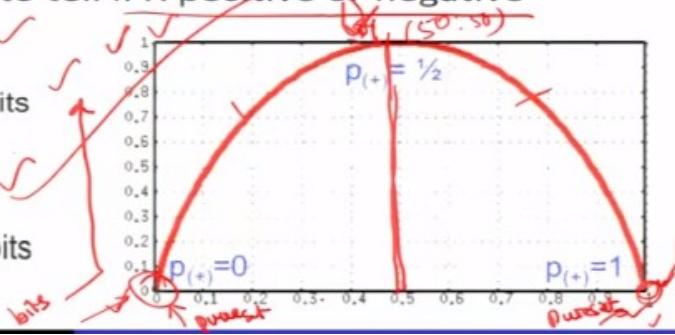


So, we are going to evaluate this purity of the set using entropy to decide the split of the nodes.

Refer Slide Time (27:09)

## Entropy

- Entropy:  $H(S) = -p(+)\log_2 p(+) - p(-)\log_2 p(-)$  bits
  - $S$  ... subset of training examples ✓
  - $p(+)/p(-)$  ... % of positive/negative examples in  $S$
- Interpretation: assume item  $X$  belongs to  $S$ 
  - How many bits need to tell if  $X$  positive or negative
- Impure (3 yes / 3 no): ✓
 
$$H(S) = -\frac{3}{6}\log_2 \frac{3}{6} - \frac{3}{6}\log_2 \frac{3}{6} = 1 \text{ bits}$$
- Pure set (4 yes / 0 no): ✓
 
$$H(S) = -\frac{4}{4}\log_2 \frac{4}{4} - \frac{0}{4}\log_2 \frac{0}{4} = 0 \text{ bits}$$



Refer Slide Time (27:10)

## Information Gain

- Want many items in pure sets
- Expected drop in entropy after split:

$$\text{Gain}(S, A) = H(S) - \sum_{V \in \text{Values}(A)} \frac{|S_V|}{|S|} H(S_V)$$

✓ ... possible values of  $A$   
✓ ... set of examples  $\{X\}$   
✓ ... subset where  $X_A = V$

- Mutual Information

-between attribute  $A$   
and class labels of  $S$

$$\begin{aligned} \text{Gain}(S, \text{Wind}) &= H(S) - \frac{8}{14} * H(S_{\text{weak}}) - \frac{6}{14} * H(S_{\text{strong}}) \\ &= 0.94 - \frac{8}{14} * 0.81 - \frac{6}{14} * 1.0 \\ &= 0.049 \quad \text{(0.049)} \end{aligned}$$



Let us see, how this decision tree has used this concept? This concept is termed as information gain. So, we want to see how many items are there in that pure set and how many and this particular expectation will drop after the split in that entropy value and this can be termed as the gain and that is

called information gain. So, information gain will be  $H(S) - \sum_{i \in \text{values}(A)} \frac{|S_i| H(S_i)}{|S|}$  and that is the summation of all the values of  $A$  and the set of all samples divided by that means subsets of all the

samples divided by the set of all examples. Let us see, how with this information gain we are going to do this kind of decision whether to split or not? So, let us see here in this case of the wind we are in the wind we have seen that there are 9 yes and 5 no. So, the positive variable is 9 which will be fed

and the total number of samples are 14. So,  $\frac{-9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14}$  comes out to be 0.94. So,

information gain is 0.94 here in the wind. How about calculating this same information gained for weekend for the strong? For the weak we obtained 0.81 and for the strong we are obtained 1. Now, we can see what is the information gain here in this case of wind whether we can split or not? So,  $H(S)$  is basically the information gain of a wind. So, that is 0.94 and this particular then for the weak case it is

0.81 and it will be divided by  $\frac{8}{14}$ . So, so total number of samples if we count here they are fourteen.

So, that total number of 14 and out of 14 we are 4 we case we are taking only

$0.94 - \frac{8}{14} * 0.81 - \frac{6}{14} * 1.0 \cdot \frac{8}{14}$  and - this comes out to be  $\frac{6}{14}$  and if we evaluate the gain comes out

to be 0.049 information gain. So, if we split then the net gain will be the positive gain of 0.049. Hence, it is better to split.

Refer Slide Time (30:41)

## Decision Trees for Regression

So, this way using information gain we can estimate or the decision tree will make the decision whether to split or not?

Refer Slide Time (30:51)

## How to grow a decision tree

- The tree is built **greedily** from top to bottom
- Each split is selected to **maximize information gain (IG)**

$$IG = \text{Impurity}(Z) - \left( \underbrace{\frac{|Z_L|}{|Z|} \text{Impurity}(Z_L)}_{\text{Error before split}} + \underbrace{\frac{|Z_R|}{|Z|} \text{Impurity}(Z_R)}_{\text{Error after split}} \right)$$

Now, let us see how to grow, how to build this particular decision tree. So, the trees is built greedily from top to the bottom and each split is selected to maximize the information gained. So, what we have seen here is that in the previous example that after the split the gain is positive. So, we have to maximize after the split. Hence, we are going to split in this particular case.

Refer Slide Time (31:16)

## Decision Tree for Regression

- Given a training set:  $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}$   
 $y_i$ -real values
- Goal is to find  $f(x)$  (a tree) such that

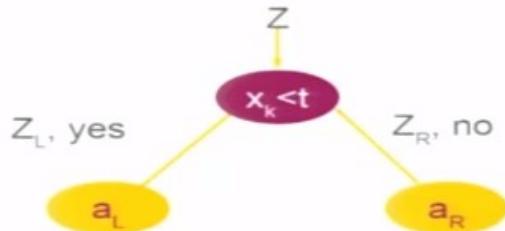
$$\min \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2$$

- How to grow a decision tree for regression ?

Refer Slide Time (31:20)

## How to find the best split

- A tree is built from top to bottom. And at each step, you should find the best split in the internal node. You get a test dataset  $Z$ . And here is a splitting criteria  $x_k < t$  or  $x_k \geq t$ . The problem is how to find  $k$ , the number of feature and the threshold  $t$ . And also you need to find values in leaves  $a_L$  and  $a_R$ .



Refer Slide Time (31:29)

## What happens without a split?

- What happens without a split?**
- Without a split, the best you can do is to predict one number, 'a'. It makes sense to select such a number 'a' to minimize the squared error.
- It is very easy to prove that such variable  $a$  equals an average of all  $y_i$ .
- Thus you need to calculate the average value of all the targets.
- Here  $\bar{a}$  ( $a$ -hat) denotes the average value.
- Impurity  $z$  equals the mean squared error if we use  $\bar{a}$  ( $a$ -hat) as a prediction.

**Without split:** predict one number  $a$

$$\hat{a} = \min_a \sum_{i \in Z} (a - y_i)^2$$
$$\hat{a} = \frac{1}{|Z|} \sum_{i \in Z} y_i$$



$|Z|$  - number of elements in  $Z$

$$\text{Impurity}(Z) = \frac{1}{|Z|} \sum_{i \in Z} (\hat{a} - y_i)^2$$

Refer Slide Time (31:35)

## Find the best split ( $x_k < t$ )

- What happens if you make some split by a condition  $x_k < t$ ? For some part of training objects  $Z_L$ , you have  $x_k < t$ . For other part  $Z_R$  holds  $x_k \geq t$ .
- The error consists of two parts for  $Z_L$  and  $Z_R$  respectively. So here we need to find simultaneously  $k$ ,  $t$ ,  $a_L$  and  $a_R$ .
- We can calculate the optimal  $a_L$  and  $a_R$  exactly the same way as we did for the case without a split. We can easily prove that the optimal  $a_L$  equals an average of all targets which are the targets of objects which get to the leaf. And we will denote these values by  $\hat{a}_L$  and  $\hat{a}_R$  respectively.

$$\min_{k, t, a_L, a_R} \sum_{i \in Z_L} (a_L - y_i)^2 + \sum_{i \in Z_R} (a_R - y_i)^2$$

Values in leaves

$$\hat{a}_L = \frac{1}{|Z_L|} \sum_{i \in Z_L} y_i, \quad \hat{a}_R = \frac{1}{|Z_R|} \sum_{i \in Z_R} y_i$$

$|Z_L|$  - number of elements in  $Z_L$ ,  
 $|Z_R|$  - number of elements in  $Z_R$

$$\min_{k, t} \sum_{i \in Z_L} (\hat{a}_L - y_i)^2 + \sum_{i \in Z_R} (\hat{a}_R - y_i)^2$$

So, we can see here by this way we can find out the splits and we have already seen that if we can do this kind of predictions to find out this whether to split a node or not. So, whenever there will be information gained and to maximize the gain according to that maximizing the information gained this particular splitting is to be done. So, we have to find out the best split and we have to do the decision based on that.

Refer Slide Time (32:00)

## Stopping rule

- The node depth is equal to the **maxDepth** training parameter.
- No split candidate leads to an information gain greater than **minInfoGain**.
- No split candidate produces child nodes which have at least **minInstancesPerNode** training instances ( $|Z_L|, |Z_R| < \text{minInstancesPerNode}$ ) each.

Now, there is another criteria which is called a stopping criteria that the growth of the spanning tree or the building of the spanning tree it has to be stopped. So, the node depth is equal to the maximum and depth of the training parameter and no split candidate leads to an information getting greater than minimum information gain and no candidate produces the child which have at least the minimum instances per node training instances.

Refer Slide Time (32:33)

## Summary: Decision Trees

- **Automatically handling interactions of features:** The benefits of decision tree is that this algorithm can automatically handle interactions or features because it can combine several different features in a single decision tree. It can build complex functions involving multiple splitting criteria.
- **Computational scalability:** The second property is a computational scalability. There exists efficient algorithms for building decision trees for the very large data sets with many features.
- **Predictive power:** Single decision tree actually is not a very good predictor. The predictive power of a single tree is typically not so good.
- **Interpretability:** You can visualize the decision tree and analyze this splitting criteria in nodes, the values in leaves, and so one. Sometimes it might be helpful.

So, in summary what we have seen in the decision tree is that here the automatic handling of interactions of the features is done and computationally scalability also we have seen and interpretability aspects also.

Refer Slide Time (32:47)

## Building a tree using MapReduce

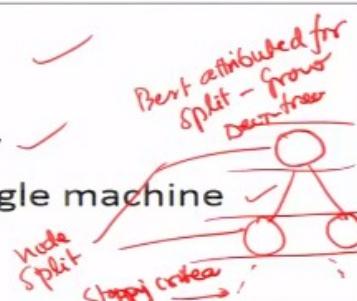
Now, let us see how to build tree using MapReduce.

Refer Slide Time (32:53)

### Problem: Building a tree

- Given a large dataset with hundreds of attributes ✓
- Build a decision tree! ↗ Private
- General considerations:
  - Tree is small (can keep it memory):
    - Shallow (~10 levels) ✓
  - Dataset too large to keep in memory ✓
  - Dataset too big to scan over on a single machine ✓
  - MapReduce to the rescue! ✓

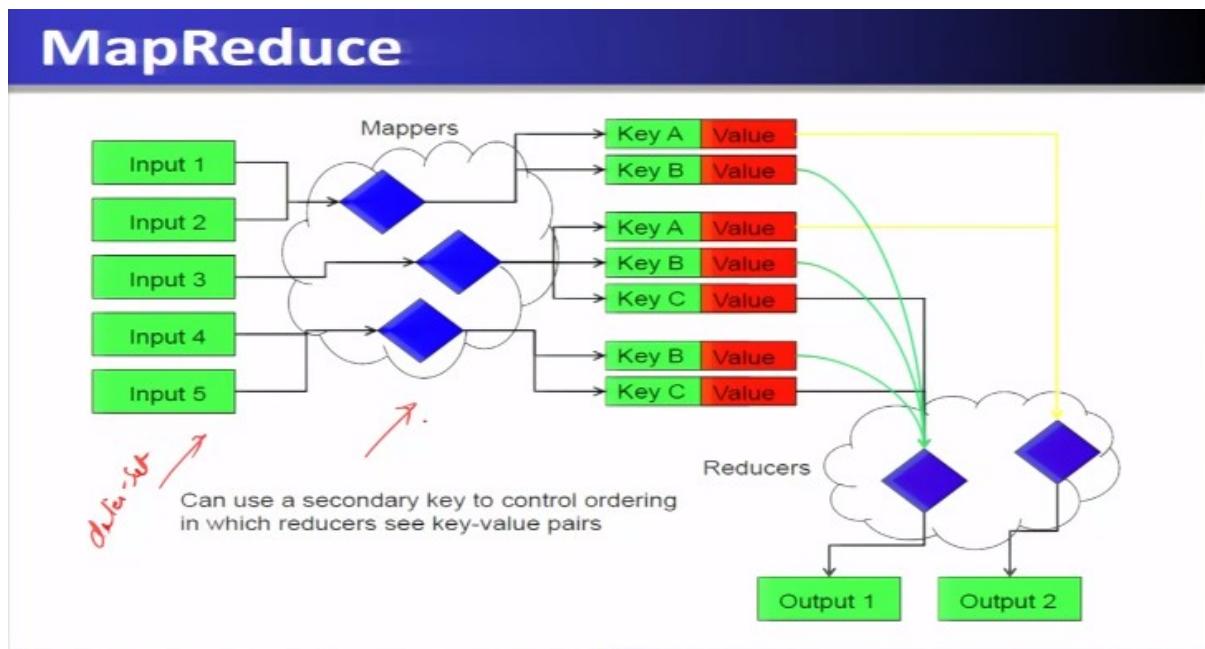
```
Algorithm 1 FindBestSplit
Require: Node n, Data D ⊆ D*
1: (n → split, D_L, D_R) = FindBestSplit(D) ✓
2: if StoppingCriteria(D_L) then
3: n → left_prediction = FindPrediction(D_L)
4: else
5: FindBestSplit(n → left, D_L) ✓
6: if StoppingCriteria(D_R) then
7: n → right_prediction = FindPrediction(D_R)
8: else
9: FindBestSplit(n → right, D_R) ✓
```



So, now MapReduce will be applied for the decision tree algorithm and here we will see the approach and we will see more detail of it. How this MapReduce will be applied in the decision tree. So, that the big data analytics can be performed with the help of this parallel a decision tree implementation. Now, here we are given a large data sets and which has a lot of attribute the size of the data set is also big and the number of attributes are also more and we want to build a decision tree and that is what is called the Big Data applying that is the big data analysis. So, here we can we have to see is, that the tree all the data is very big but the tree is small and it can be stay in the memory and maximum levels

of that particular tree which is extracted out of big data is on off out of the ten levels and also we have to keep the data set the very large data set into the memory and this data set is too big to be kept and one machine. So, basically MapReduce we are going to use which is going to go over the clusters of machines. Now, here in this particular technique which is called the decision tree. We have to find out the best splits and then we have to find out which attribute along with which we are going to build the tree that is the single tree which we are going to build. So, we have to first find out which is the best variable around which the split has to be done or the tree has to be constructed. So, the best we have to find out the best attribute for the split to grow the decision tree. So, after having identified a particular attribute. Now, we have to identify the how the best split is to be done? So, that in the left side and the right side of the tree can basically grow and so on. So, these particular growth of the spanning of the decision tree which we are going to see level wise growth of the tree and also the stopping criteria when we have to stop further growth of the tree. So, this particular decision is called when we are going to split. So, this is whether the node what is the condition in which we are going to split the node along whether to split or not and if it is then which particular attribute will be the best to split. So, all these things we are going to discuss which are going to be implemented using MapReduce in.

Refer Slide Time (36:17)



So, MapReduce we have already seen that it is it will the input data set is now divided into the chunks and these chunks are given to the map or functions and the mapper function will generate the key value pair which is further given to the reduce functions and they will generate the output.

Refer Slide Time (36:47)

## PLANET

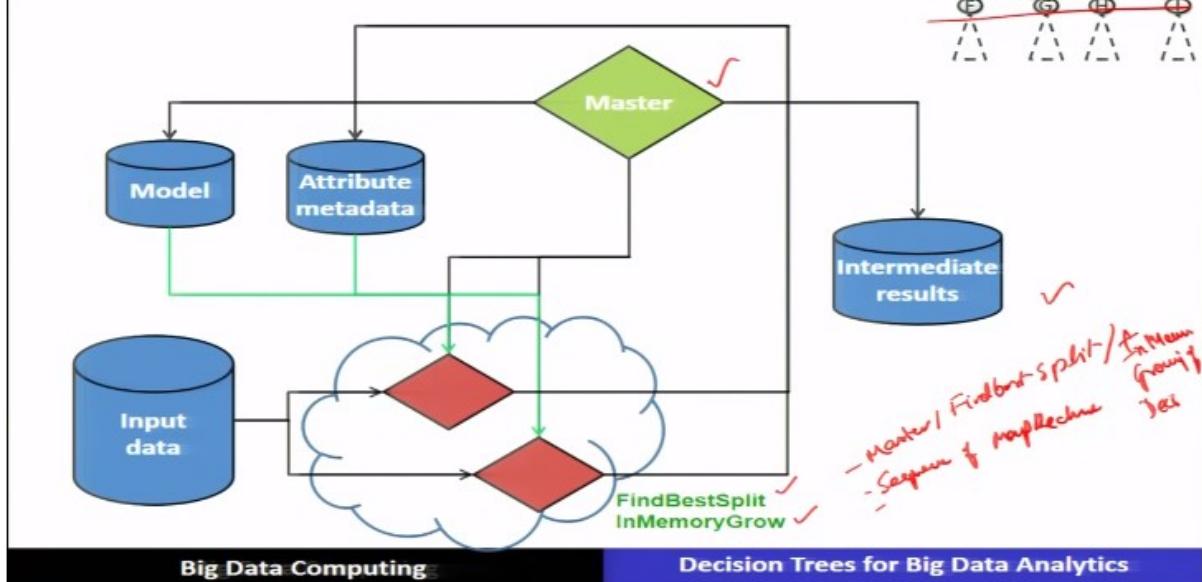
### Parallel Learner for Assembling Numerous Ensemble Trees [Panda et al., VLDB '09]

- A sequence of MapReduce jobs that build a decision tree ✓
- Setting:
  - Hundreds of numerical (discrete & continuous) attributes
  - Target (class) is numerical: **Regression**
  - Splits are binary:  $X_j < v$
  - Decision tree is small enough for each Mapper to keep it in memory ✓ *binary tree*
  - Data too large to keep in memory

Now, this particular implementation of a MapReduce for decision tree is published in a paper which is called PLANET by the Google. So, Parallel Learner for Assembling Numerous Ensemble Trees which we are going to see here as the MapReduce implementation of a decision tree. So, this is nothing but a sequence of MapReduce job that will build the decision trees and here we assume that the values which are given in the data set they are basically hundreds of numerical attributes and we are going to build the binary tree which is the decision tree which is a binary which is a binary tree and this particular tree which we assume that is small enough to be for each mapper to be kept in the memory and data is too large.

Refer Slide Time (37:45)

# PLANET Architecture

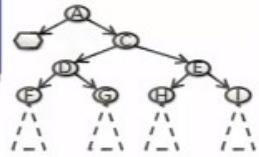


Let, us see the different constituents in this architecture. So, here we see that there will be a master node and this particular master node in turn will decide about finding the best split and also the in memory growth of that tree at the mapper end and the intermediate results are also stored and this all happens in the coordination of the master. So, the tree is built level wise using these different MapReduce sequence of MapReduce operations. MapReduce operations are being coordinated by the master. There will be a master node which will decide the best is split and also tries to decide about in memory growing of the decision tree had the mapper.

Refer Slide Time (39:08)

## PLANET Overview

- We build the tree level by level
  - One MapReduce step builds one level of the tree
- **Mapper**
  - Considers a number of possible splits  $(X_i, v)$  on its subset of the data
  - For each split it stores partial statistics
  - Partial split-statistics is sent to **Reducers**
- **Reducer**
  - Collects all partial statistics and determines best split.
- **Master** grows the tree for one level

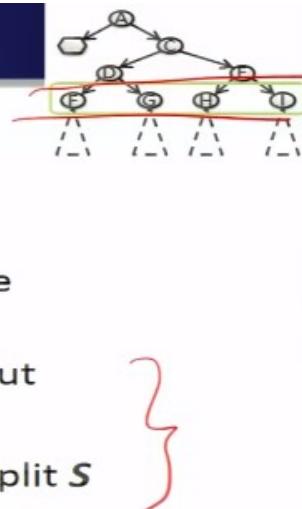


So, let us see the planet overview. So, here in this particular method we build the tree level by level and one MapReduce step will build one level of a tree. So, if there is a 10 level. So, 10 different MapReduce in sequence has to execute. As far as the mapper is concerned, mapper will consider the number of possible splits on the subset of the data and this comprises of  $(X_i, V)$  and for each is split. It is stores the partial statistics partial split statistics is then sent to the reducers reducers we collect all the partial statistics and determines the best split and master grows the tree for one more level.

Refer Slide Time (40:05)

## PLANET Overview

- **Mapper** loads the **model** and info about which **attribute splits** to consider
- Each mapper sees a subset of the data  $D^*$
- Mapper “drops” each datapoint to find the appropriate leaf node  $L$
- For each leaf node  $L$  it keeps statistics about
  - 1) the data reaching  $L$
  - 2) the data in left/right subtree under split  $S$
- **Reducer** aggregates the statistics (1) and (2) and determines the best split for each node



So, here the mapper loads the model and information about the attribute splits to consider. So, each mapper sees a subset of the data  $D^*$  and the mapper drops each data point to find the appropriate leaf node  $L$  and for each node  $L$  it keeps the statistics about the data reaching  $L$  and the data in the left and right subtree under the splits  $S$  this way the mapper. So, each mapper will now has to deal with a particular level of the nodes and has to decide whether it has to be further split or not. If it is to be splitted further then another iteration will go on with the next set of MapReduce job. Reducer aggregates the statistics of one and two and determines the best split at each node.

Refer Slide Time (41:00)

## PLANET: Components

- **Master**
  - Monitors everything (runs multiple MapReduce jobs)
- **Three types of MapReduce jobs:** ✓

**(1) MapReduce Initialization (run once first)**

- For each attribute identify values to be considered for splits ✓  
*(root node)*

**(2) MapReduce FindBestSplit (run multiple times)** ✓ *for each level of Decision Tree*

- MapReduce job to find best split when there is too much data to fit in memory

**(3) MapReduce InMemoryBuild (run once last)**

- Similar to FindBestSplit (but for small data)
- Grows an entire sub-tree once the data fits in memory

So, overall different components in this MapReduce implementation of decision tree is like this. So, master will monitor everything and runs multiple MapReduce jobs. So, besides master there are three different type of MapReduce jobs which will be executed. So, first is the initialization of MapReduce that is it will be run once. So, for each attribute it will identify the values to be considered for the split. So, for example the root node so for the root node decision has to be done in during the initialization phase and then it will be a second set of MapReduce job to find out the best is split and it will run multiple times for each level of decision tree growth. So, the second step will keep on increasing keep on growing the the levels of the tree as they are being invoked. So, the number of time is it it is invoked. So, that indicates that it will be growing in that levels. So, that particular so MapReduce job is to find the best displayed when there are too much of data to fit in the memory. So, this way this MapReduce will try to find out the best split and keep on splitting and also it will be then the next step is finally the MapReduce will build the in-memory build that is subtrees are built at each MapReduce

then they will be runs once. So, similar to find the best split for small data it will grow and entire subtree once the data fits in the memory. So, these three different MapReduce jobs one is for the root node that is the initialization. The second one is at the level wise splitting of the nodes, and growing the decision tree and finally it has to take this subtrees which are in memory and built during each run. So, it will be done once.

Refer Slide Time (43:33)

## Reference

- B. Panda, J. S. Herbach, S. Basu, and R. J. Bayardo. PLANET: Massively parallel learning of tree ensembles with MapReduce. VLDB 2009.
- J. Ye, J.-H. Chow, J. Chen, Z. Zheng. Stochastic Gradient Boosted Distributed Decision Trees. CIKM 2009.

These are the references for this discussion which we have just seen.

Refer Slide Time (43:43)

**Example: Medical Application  
using a Decision Tree in Spark  
ML**

Example medical application using decision tree in spark machine learning.

Refer Slide Time (43:50)

## Create SparkContext and SparkSession

- This example will be about using Spark ML for doing classification and regression with decision trees, and in samples of decision trees.
- First of all, you are going to create SparkContext and SparkSession, and here it is.

```
In []: from pyspark import SparkContext
from pyspark.sql import SparkSession
```

Consider Medical application w/  
Decision trees for  
Big data Analytics

```
In []: sc = SparkContext(appName = "module3_week4")
```

1. Create  
SparkContext

```
In []: ! echo $PYSPARK_SUBMIT_ARGS
```

```
In []: spark = SparkSession.Builder().getOrCreate() # required for dataframes
```

2. Build  
Spark  
Session

Now, we will consider an example of the medical application of the application of decision trees for using spark a ML. So, in this example we will see how the classification using the decision trees is done. So, let us see the steps for doing this decision tree based analytics for the medical application. So, here we are going to consider the medical application using decision trees for big data analytics. Now, here we have to see that we have to give the name of the application and now we have to create the sparkcontext using this application name. So, we call it as by sparkcontext. We have to create a sparkcontext. After creating the sparkcontext, then we have to build a session that is called a spark session. This is step number one and this is step number two which is essential in all such applications which runs under spark. First, we have to create the context then a spark session and this is required to build the data frames.

Refer Slide Time (46:10)

## Download a dataset ✓

- Now you are downloading a dataset which is about breast cancer diagnosis, here it is.

```
Download a dataset (breast cancer diagnosis)
In []: !wget https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wis
In []: head -n 3 wdbc.data
In []: !wc -l wdbc.data
In []: #1) ID number
#2) Diagnosis (M = malignant, B = benign)
#3) Features
In []: from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import StringIndexer
In []: # Load a text file and convert each line to a Row.
data = []
with open("wdbc.data") as infile:
```

✓ ✓

M/B  
Diagnosis features  
numeric

The next step would be to download the data set which is used here in this example. So, we will download the data set from some link and let us call it as this data set which is being downloaded. Once the data set is available and is downloaded into the spark system. Then we have to see, that this particular data set has three different parts that is in the form of key value pair. It has an ID number and then it will have the key as whether it is having a medical problem yes or no means malign or having benign and then different features of it. So, the data will be in this particular form. It will be a diagnosis and then it will be a features. Now, this diagnosis if it is M or B it has to be converted into the numeric form. So, we have to convert we have to load into file and we have to convert this field into the numeric field.

Refer Slide Time (47:36)

## Exploring the Dataset

- Let's explore this dataset a little bit. The first column is ID of observation. The second column is the diagnosis, and other columns are features which are comma separated. So these features are results of some analysis and measurements. There are total 569 examples in this dataset. And if the second column is M, it means that it is cancer. If B, means that there is no cancer in a particular woman.

```
In [6]: !head -n 3 wdbc.data
842302,M,17.99,10.38,122.8,1001.0,1184.0,2776.0,3001.0,1471.0,2419.0,0.07871,1.09
5.0,0.9053,8.589,153.4,0.006399,0.04904,0.05373,0.01587,0.03003,0.006193,25.38,1
7.33,184.0,2019.0,1622.0,0.656,0.7119,0.2654,0.4601,0.1189
842517,M,28.57,17.77,132.9,1326.0,0.08474,0.07864,0.0869,0.07017,0.1812,0.05667,
0.5435,0.7339,3.398,74.08,0.005225,0.01308,0.0186,0.0134,0.01389,0.003532,24.9
9.23,41.158.8,1950.0,1238.0,1866.0,2416.0,186.0,275.0,0.08902
84300903,M,19.69,21.25,138,1203.0,1096.0,1599.0,1974.0,1279.0,2069.0,0.05999,0.74
55.0,7869.4,585.94,0.03,0.00015,0.04006,0.03832,0.02058,0.0225,0.004571,23.57,25.
55,152.5,1709.0,1444.0,4245.0,4504.0,243.0,3613.0,0.08758

In []: !wc -l wdbc.data
In []: #1) ID number
#2) Diagnosis (M = malignant, B = benign)
#3) Features

In []: from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import StringIndexer

In []: # Load a text file and convert each line to a Row.
```

So, out of this particular data set that is 569 examples in a data set we have to convert in this particular field of diagnosis into the numeric values and then we have to see this particular explore the data set.

Refer Slide Time (47:48)

## Exploring the Dataset

- First of all you need to transform the label, which is either M or B from the second column. You should transform it from a string to a number. We use a StringIndexer object for this purpose, and first of all you need to load all these datasets. Then you create a Spark DataFrame, which is stored in the distributed manner on cluster.

```
In []: from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import StringIndexer

In []: # Load a text file and convert each line to a Row.
 data = []
 ✓
 with open("wdbc.data") as infile:
 for line in infile:
 tokens = line.rstrip("\n").split(",")
 y = tokens[1]
 features = Vectors.dense([float(x) for x in tokens[2:]])
 data.append((y, features))
 ✓

In []: inputDF = spark.createDataFrame(data, ["label", "features"])

In []: inputDF.show()
 .
 ✓

In []: stringIndexer = StringIndexer(inputCol = "label", outputCol = "labelIndexed")
si_model = stringIndexer.fit(inputDF)
inputDF2 = si_model.transform(inputDF)
```

So, after having read this particular data set which we have downloaded. Now, it will be creating the data frames with the label and the feature and the label will be converted into into the numeric values.

Refer Slide Time (48:19)

## Exploring the Dataset

- inputDF DataFrame has two columns, label and features. Okay, you use an object vector for creating a vector column in this dataset, and then you can do string indexing. So Spark now enumerates all the possible labels in a string form, and transforms them to the label indexes. And now label M is equivalent 1, and B label is equivalent 0.

```
In [11]: inputDF = spark.createDataFrame(data, ["label", "features"])

In [12]: inputDF.show()

+-----+-----+
|label| features|
+-----+-----+
|M|[17.99,10.38,122.0]
|M|[20.57,17.77,132.0]
|M|[19.69,21.25,130.0]
|M|[11.42,20.38,77.5]
|M|[20.29,14.34,135.0]
|M|[12.45,15.7,82.57]
|M|[18.25,19.98,119.0]
|M|[13.71,20.83,90.2]
|M|[13.0,21.82,87.5]
|M|[12.46,20.04,83.9]
|M|[16.02,23.24,102.0]
|M|[15.78,17.89,103.0]
|M|[19.17,24.8,132.4]
|M|[15.85,23.95,103.0]
|M|[13.73,22.61,93.6]
|M|[14.56,27.54,96.7]
|M|[14.88,20.13,94.7]
+-----+-----+
```

Refer Slide Time (48:24)

## Train/Test Split

- We can start doing the machine learning right now. First of all, you make training test splitting in the proportion 70% to 30%. And the first model you are going to evaluate is one single decision tree.

### train/test split

```
In [15]: (trainingData, testData) = inputDF2.randomSplit([0.7, 0.3], seed = 123)
```

### Training Decision Tree

```
In []: from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

In []: decisionTree = DecisionTreeClassifier(labelCol = "labelIndexed")

In []: dtModel = decisionTree.fit(trainingData)

In []: dtModel.numNodes

In []: dtModel.depth

In []: dtModel.featureImportances
```

Refer Slide Time (48:27)

## Exploring the Dataset

- inputDF DataFrame has two columns, label and features. Okay, you use an object vector for creating a vector column in this dataset, and then you can do string indexing. So Spark now enumerates all the possible labels in a string form, and transforms them to the label indexes. And now label M is equivalent 1, and B label is equivalent 0.

```
In [11]: inputDF = spark.createDataFrame(data, ["label", "Features"])

In [12]: inputDF.show()

+---+-----+
|label| Features|
+---+-----+
M	[17.99,10.38,122...
M	[20.57,17.77,132...
M	[19.69,21.25,130...
M	[11.42,20.38,77.5
M	[20.29,14.34,135...
M	[12.45,15.7,82.57
M	[18.25,19.98,119...
M	[13.73,20.83,99.2
M	[13.6,21.82,87.5...
M	[12.46,24.04,83.9
M	[16.02,23.24,102...
M	[15.78,17.89,103...
M	[19.17,24.8,132.4...
M	[15.85,23.95,103...
M	[13.73,22.61,93.6...
M	[14.54,27.54,96.7...
M	[14.66,20.13,94.7...
+---+-----+
```

and after that this particular data set which will having this values converted into so these M fields will be converted into the numeric values and then now after converting it now it will be ready. So, the conversion will be M is equivalent to 1 and V is equivalent to 0. So, wherever 1 is there M is there. So, it will be 0 and wherever V is there. So, after this conversion now this particular data set will be now ready for the use of machine learning.

Refer Slide Time (48:58)

## Train/Test Split

- We can start doing the machine learning right now. First of all, you make training test splitting in the proportion 70% to 30%. And the first model you are going to evaluate is one single decision tree.

### train/test split

```
In [15]: (trainingData, testData) = inputDF2.randomSplit([0.7, 0.3], seed = 23)
```

### Training Decision Tree

```
In []: from pyspark.ml.classification import DecisionTreeClassifier
 from pyspark.ml.evaluation import MulticlassClassificationEvaluator

In []: decisionTree = DecisionTreeClassifier(labelCol = "labelIndexed")

In []: dtModel = decisionTree.fit(trainingData)

In []: dtModel.numNodes

In []: dtModel.depth

In []: dtModel.featureImportances
```

So, the first after preparing the data. Now, we have to use the data for the machine learning for that we have to split this data into the training data set and the test dataset into the proportion of seventy

and thirty. So, to split this data set into the training and test data splits. Now, we have to use this particular spark come on randomsplit that is seventy percent is of training data and thirty percent is the test data. So, after this splitting of the training data and the test data. Now, we are going to apply this decision tree algorithm. So, we will now use the decision tree classifier and now we will use this labelindexed which we have seen in the previous slide and then we will do this decision tree fit on the decision on the training data set which we have already obtained from the the data set is split seventy percent of it. So, after this step we are ready with the the decision tree. Now, we can see out of this decision tree it is different parameters such as how many nodes are there in the decision tree, what is the depth of the decision tree, also the what are the features which are of importance and this will build the model of a decision tree. So, decision tree model is being built here after this execution.

## Refer Slide Time (51:14)

# Train/Test Split

use data for ML

- We can start doing the machine learning right now. First of all, you make training test splitting in the proportion 70% to 30%. And the first model you are going to evaluate is one single decision tree.

```
In [15]: (trainingData, testData) = inputDF2.randomSplit([0.7, 0.3], seed = 23)
```

train/test split ✓

test data ✓

trainData ✓

Training Decision Tree

Decision Tree model

```
In []: from pyspark.ml.classification import DecisionTreeClassifier
In []: from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
In []: decisionTree = DecisionTreeClassifier(labelCol = "labelIndexed")
```

```
In []: dtModel = decisionTree.fit(trainingData)
```

```
In []: dtModel.numNodes
```

```
In []: dtModel.depth
```

```
In []: dtModel.FeatureImportances
```

### Refer Slide Time (51:13)

## Train/Test Split

- We are making import DecisionTreeClassifier object. We create a class which is responsible for training, and we call the method fit to the training data, and obtain a decision tree model.

```
In [17]: decisionTree = DecisionTreeClassifier(labelCol = "labelIndexed")
In [18]: dtModel = decisionTree.fit(trainingData)
In [19]: dtModel.numNodes ✓
Out[19]: 29 ✓
In [20]: dtModel.depth
Out[20]: 5 ✓
In [21]: dtModel.featureImportances ✓
Out[21]: SparseVector(30, {1: 0.0589, 6: 0.0037, 10: 0.0112, 13: 0.0117, 20: 0.0324, 21: 0.0302, 22: 0.7215, 24: 0.01, 26: 0.0191, 27: 0.1013})
In [22]: dtModel.numFeatures ✓
Out[22]: 30 ✓
In []: print dtModel.toDebugString
```

Now, now we can see that if we find out how many in this particular data set of around six hundred different samples. We have seen, that there are 29 nodes are there and the number of depth is only 5 and different important features which we have obtained in the form of the graph and number of features are 30.

Refer Slide Time (51:46)

## Train/Test Split

- Number of nodes and depths of decision tree, feature importances, total number of features used in this decision tree, and so on.
- We can even visualize this decision tree and explore it, and here is a structure of this decision tree. Here are splitting conditions If and Else, which predict values and leaves of our decision trees.

```
Out[21]: SparseVector(30, {1: 0.0589, 6: 0.0037, 10: 0.0112, 13: 0.0117, 20: 0.0324, 21: 0.0302, 22: 0.7215, 24: 0.01, 26: 0.0191, 27: 0.1013})
In [22]: dtModel.numFeatures
Out[22]: 30
In []: print dtModel.toDebugString
In []: predictions = dtModel.transform(testData)
In []: predictions.select('label', 'labelIndexed', 'probability', 'prediction').show()
In []: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predicti
accuracy = evaluator.evaluate(predictions) ✓
print("Test Error = %g" % (1.0 - accuracy))
```

So, after constructing this particular decision tree. Now, we have to see that we have to evaluate this particular decision tree whether it is the quality wise whether it is good or bad. How it is good for the

predictions? So, evaluation is done. So, multiclass classifier MulticlassClassificationEvaluator is used for this particular purpose to evaluate the predictions accuracy.

## Refer Slide Time (52:20)

## Train/Test Split

- Now we are applying a decision tree model to the test data, and obtain predictions. And you can explore these predictions. The predictions are in the last column.
  - And in this particular case, our model always predicts zero class.

```
In [23]: print dtModel.toDebugString

DecisionTreeClassificationModel (uid=DecisionTreeClassifier_4653be4c1bd9e589b6
4) of depth 5 with 29 nodes
 If (Feature 22 <= 114.2)
 If (feature 27 <= 0.1613)
 If (feature 20 <= 16.57)
 If (feature 27 <= 0.1258)
 If (feature 10 <= 0.9289)
 Predict: 0.0
 Else (feature 10 > 0.9289)
 Predict: 1.0
 Else (feature 27 > 0.1258)
 If (feature 21 < 32.85)
 predict: 0.0
 Else (feature 21 > 32.85)
 Predict: 1.0
 Else (feature 20 > 16.57)
 If (feature 1 <= 16.54)
 Predict: 0.0
 Else (feature 1 > 16.54)
 If (feature 24 <= 0.1084)
 Predict: 0.0
 Else (feature 24 > 0.1084)
 Predict: 1.0
```

Refer Slide Time (52:23)

# Predictions

```
In [1]: predictions = dtModel.transform(testData)
```

```
In [1]: predictions.select('label', 'labelIndexed', 'probability', 'prediction').show()
```

```
In []: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predictionCol = "prediction")
accuracy = evaluator.evaluate(predictions)
```

```
print("Test Error = %g" % (1.0 - accuracy))
```

## Refer Slide Time (52:29)

# Predictions

So, we want to evaluate the prediction accuracy and by that by this particular method we have to see that the prediction accuracy which is calculated for this decision tree construction is around 96% and this is fairly good which we have generated.

## Refer Slide Time (52:41)

# Predictions

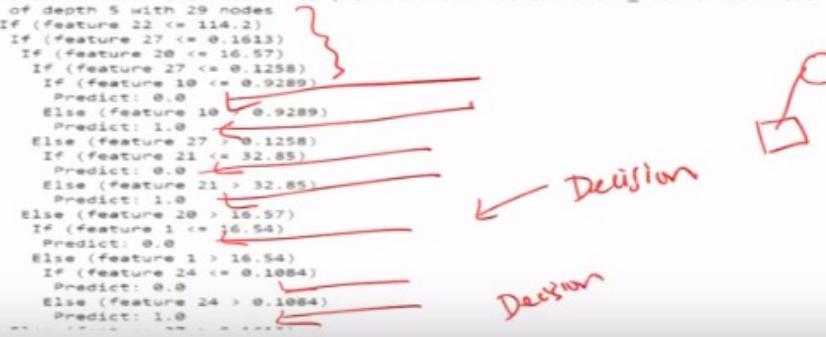
Here, we can see that these predictions are shown here in this case. So, this is the decision tree which is generated which is nothing but here you can see all the predictions that means predictions are there at the leaf nodes and all other nodes are there the decision nodes are there at the internal nodes

Refer Slide Time (52:53)

## Train/Test Split

- Now we are applying a decision tree model to the test data, and obtain predictions. And you can explore these predictions. The predictions are in the last column.
- And in this particular case, our model always predicts zero class.

```
In [23]: print(dtModel.toDebugString)
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_4653be4ce1bd9e589d6
4) of depth 5 with 29 nodes
If (feature 22 <= 114.2)
 If (feature 27 <= 0.1613)
 If (feature 20 <= 16.57)
 If (feature 27 <= 0.1258)
 If (feature 10 <= 0.9289)
 Predict: 0.0
 Else (feature 10 > 0.9289)
 Predict: 1.0
 Else (feature 27 > 0.1258)
 If (feature 21 <= 32.85)
 Predict: 0.0
 Else (feature 21 > 32.85)
 Predict: 1.0
 Else (feature 20 > 16.57)
 If (feature 1 <= 16.54)
 Predict: 0.0
 Else (feature 1 > 16.54)
 If (feature 24 <= 0.1084)
 Predict: 0.0
 Else (feature 24 > 0.1084)
 Predict: 1.0
 ...
```



So, this is the prediction node this is also the prediction nodes this is also the prediction node and you see that this is the decision tree which is being calculated out of the training data set

Refer Slide Time (53:32)

## Predictions

```
In []: predictions = dtModel.transform(testData)
In []: predictions.select('label', 'labelIndexed', 'probability', 'prediction').show()
In []: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predicti
accuracy = evaluator.evaluate(predictions)
print("Test Error = %g" % (1.0 - accuracy))
```

and you and using the test data set we can evaluate this particular model of a training of a decision tree which is being generated based on its prediction accuracy

Refer Slide Time (53:45)

## Accuracy

```
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
8	0.0	[0.99986757999867...] 0.0
+-----+
only showing top 20 rows

In [26]: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predicti
accuracy = evaluator.evaluate(predictions)

print("Test Error = %g" % (1.0 - accuracy))
Test Error = 0.0451977 ✓ a b%
```

and we have seen that the prediction accuracy comes out to be 96% which is fairly well.

Refer Slide Time (53:50)

## Conclusion

- In this lecture, we have discussed Decision Trees for Big Data Analytics.
- We have also discussed a case study of Breast Cancer Diagnosis using a Decision Tree in Spark ML.

So, conclusion in this lecture we have discussed the decision trees for our big data analytics and we have also seen one case study of a medical application wherein we have applied the decision tree based on machine learn is part machine learning. Thank you.

**Lecture - 28**  
**Big Data Predictive Analytics**  
**Part - I**

Big data

Refer Slide Time: (00:15)

## Big Data Predictive Analytics



Dr. Rajiv Misra

Dept. of Computer Science & Engg.  
Indian Institute of Technology Patna  
[rajivm@iitp.ac.in](mailto:rajivm@iitp.ac.in)



0:16 / 39:21

Big Data Computing



Predictive analytics

Refer Slide Time: (00:17)

## Preface

### Content of this Lecture:

- In this lecture, we will discuss the fundamental techniques of predictive analytics.
- We will mainly cover Random Forest, Gradient Boosted Decision Trees and a Case Study with Spark ML Programming, Decision Trees and Ensembles.

Preface- content of this lecture: This lecture we will discuss the fundamental techniques of predictive analytics basically we will cover mainly the techniques such as random forest, gradient boosted decision trees and a case studies of various implementations using spark ML on decision trees and ensemble trees learning for predictive analytics.

Refer Slide Time: (00:49)

## Decision Trees

Decision trees,

Refer Slide Time: (00:51)

## Summary: Decision Trees

- **Automatically handle interactions of features:** It can combine several different features in a single decision tree. It can build complex functions involving multiple splitting criteria.
- **Computational scalability:** There exists effect of algorithms for building decision trees for the very large data sets with many features. But unfortunately, single decision tree actually is not a very good predictor.  
*Implemented on spark cluster 10s 100s nodes - Scale out DT on SPARK*
- **Predictive Power:** The predictive power of a single tree is typically not so good.  
*overfitting/noise in dataset good ✓ Predictive by Tree Learning by myself*
- **Interpretability:** We can visualize the decision tree and analyze this splitting criteria in nodes, the values in leaves, and so one.

▶ ▶ 532 / 39:21 Big Data Computing Predictive Analytics CC BY NC ND

Let us summarize the decision tree which we have covered in the previous session. So, in that decision tree what we have seen is that it automatically handled the interaction of features that is it can combine several different features in a single decision tree that we have already seen. So, the decision tree which is covered or which is drawn as the tree induction which is called is basically the tree which is splitted on different features hence there is a interaction or combine different features in a single tree, now it can build a complex functions involve multiple splitting criteria. So, that means if the tree is very big obviously it is built based on very complex functions. So, all these things are automatically handled as far as the interaction of the features is concerned that is the important aspect of the decision tree which we have covered. Second part is called computationally scalability that is there exists an effect of algorithm for building the decision tree where every large data set and with many features but unfortunately single decision tree actually is not a good predictor why because of the overfitting errors and so on and due to the noise. So, computation and scalability is there that means this particular decision tree can be implemented can be implemented on the clusters and using a spark system of hundreds and thousands of nodes which can be further killed therefore it has the computational scalability of implementing the decision tree we have also seen the predictive power. So, the predictive power of a single decision tree typically is not good. So, due to the overfitting, because of the noise in the data set therefore the predictive power is of a single tree is not typically good due to the overfitting. So, a lot of techniques we have seen about overcoming from the overfitting and thereby improving the predictive power in the single decision tree and fourth important part of this the decision tree which we have seen is about

interpretability. So, we can visualize the decision tree and analyze this splitting criteria of the nodes hence it has a good interpretability that means once a decision tree is built we can understand about the intricacies of the data with the help of the decision tree rules and we can also do the analysis of the splitting criteria on the nodes and also the values on the leaves they basically are giving the predictions on the new data set by the tree traversing in decision trees. So, all these are basically the summary of a decision tree and we have already discussed. So, important part here is that it has though it has a good interpretability but it has not that good not so good predictive powers. So, here we are now going to see if let us say we are going to solve the predictive analytics problems and this single decision tree having not that good predictive power needs to be augmented. So, we are going to see in this part of the discussion what is the techniques which can be used for predictive analytics of the Big Data.

Refer Slide Time: (05:34)

## Bootstrap and Bagging

So, before going ahead into the basic techniques some of the fundamentals which are required to build the new algorithms or which is used for the predictive analytics we will see some of the features of it as the bootstrap and the bagging.

Refer Slide Time: (05:53)

# Bootstrap

- Bootstrapping is an algorithm which produces replicas of a data set by doing random sampling with replacement. This idea is essential for the random forest algorithm.

- Consider a dataset  $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}$

*Dataset*  
*features*      *label*

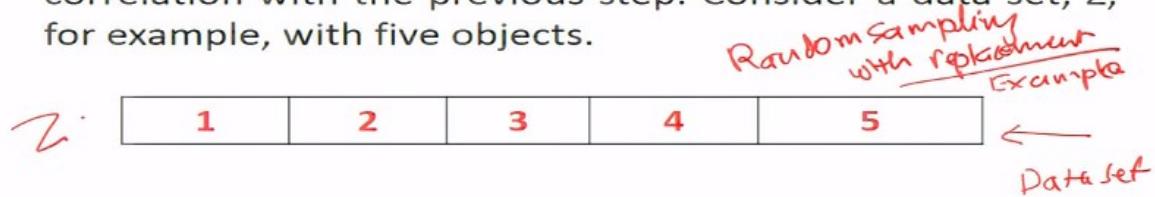
- Bootstrapped dataset  $Z^*$ - It is a modification of the original dataset  $Z$ , produced by random sampling with replacement.

So, bootstrapping is an algorithm which produces the replicas of the data set by doing the random sampling with the replacement. So, that means we are using the technique which is called the random sampling with the replacement we can build the replica of the data set that means if the data set is given we can produce another data set using bootstrap techniques by applying the random sampling with replacement. So, this idea is one of the important ideas or essential for the building the random forest algorithm. So, let us consider the data set  $Z$ , which consists of  $(x_1, y_1)$ . So, here  $x_1$  is the set of features and  $y_1$  is nothing but the labels and this particular data set is called  $Z$ . So, this particular data set is called the bootstrap data set if out of this data set we can generate another one let us see the bootstrap data set which is built out of this particular data set  $Z$ . It is the modification of the original data set produced by applying the random sampling with replacement that is called bootstrap data set let us understand how this is generated.

Refer Slide Time: (07:19)

## Sampling with Replacement

- Each iteration pick an object at random, and there is no correlation with the previous step. Consider a data set, Z, for example, with five objects.

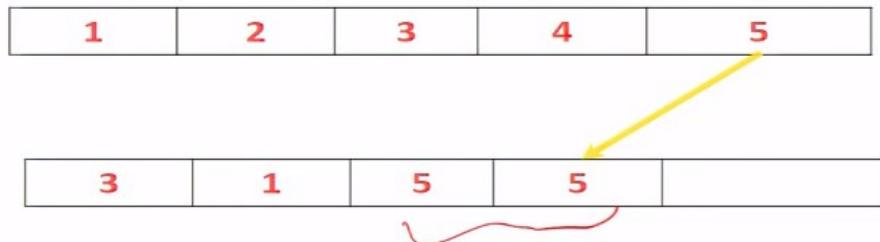


So, let us see the random sampling with replacement techniques. So, let us understand this random sampling with replacement using an example. So, let us say that this is the data set which is given let us call it as Z

Refer Slide Time: (07:59)

## Sampling with Replacement

- Then possibly you can pick again the object number five, because at each iteration you pick an object at random, and there is no correlation with the previous step.



then we can build another data set which is called  $Z^*$  by the application of the same data set with the replacement. So, let us see that we will randomly pick an object of a data set let us say that it is an object at number 3 and this is replicated here in the  $Z^*$  which is the replicated data set which we are building. Now, next we will pick another object in random let us say that this time we are picking the object

number 1 and we will now replicate it at the position number 2 in the replicated data set. Similarly, we will now pick the object number 5 and we will replicate at the position number 3 and we can also because this is the sampling random sampling with the replacement we can pick the the object number 5 again second time. So, it is being duplicated. So, that allowed in this random sampling with replacement. So, then possibly you can pick again the object number 5 because at each iteration you pick an object at random and there is no correlation with the previous step . Hence, it is called random sampling with replacement. So, therefore 5 we the object number 5 can be picked again

Refer Slide Time: (9:47)

## Sampling with Replacement

- And finally, you pick the object number two.

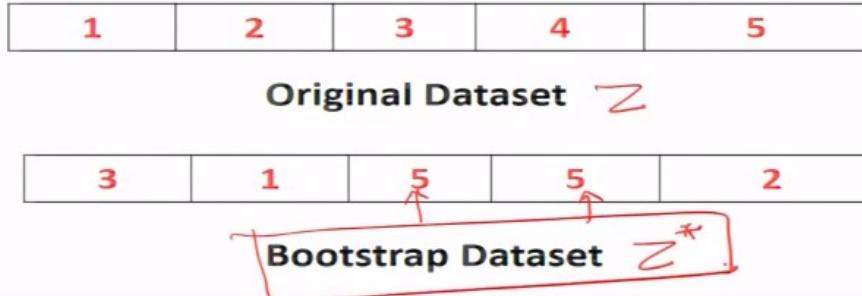


and this is shown in this example finally we will pick another object let us say object number 2 in random and we will replace it we will replicate it on the 5<sup>th</sup> position.

Refer Slide Time: (10:02)

## Sampling with Replacement

After bootstrapping we have a new data set. The size of this data set is the number of elements in the original data set. But its content, as you can see, is slightly different. Some objects may be missing and other objects may be present several times, more than once.



So, this particular data set which is now generated out of the original data set which is called  $Z$  and the bootstrap data set which we have generated called  $Z^*$  it is called a bootstrap data set. So, after bootstrapping we have a new data set the size of the data set is the same number of elements as the original data set but its contents as you see is slightly different. Now, some object may be missing and in some cases and other objects may be present several times that is more than once that we see here that this object is repeated or is being to  $n$  more than once that is allowed in random sampling with replacement to generate the bootstrap data set this particular data set is called bootstrap data set which is generated using boot strapping. So, after understanding the boot strapping,

Refer Slide Time: (10:50)

# Bagging

- It was the second idea essential for understanding of the random forest algorithm.
- Bagging (Bootstrap Aggregation): It is a general method for averaging predictions of other ~~learning~~ algorithms, not decision trees, but any other ~~learning~~ algorithm in general.
- Bagging works because it reduces the variance of the prediction.

*automatically overcomes  
from overfitting problem of DT*

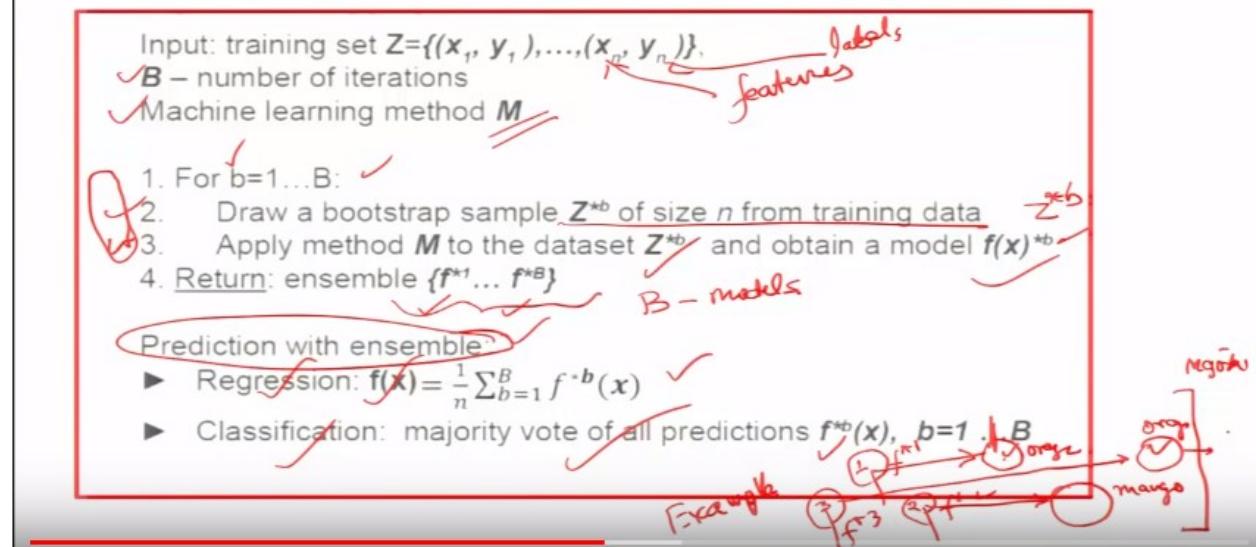
*Bagging( Ensemble of models ) → av( ensemble of models )  
less error*

Let us see what is another technique which is called Bagging. So, it was the second idea essential for an understanding of random forest algorithm. So, bagging is in short hand it is called bootstrap aggregation. So, it is a general method for averaging predictions of other algorithms and not only for decision tree but for any other models it can be applied. So, the bagging works because it reduces the variance in the prediction. So, bootstrap aggregation bagging is called bootstrap aggregation or it is the method for averaging the predictions based on the different models which are based on different learning algorithms. So, the different learning algorithms not necessarily the decision trees but any any algorithm it can be a learning algorithm. So, different learning algorithm will now give out the models. So, this bootstrap aggregation that is called bagging is nothing but everything of ensemble of of models. So, it will give an average of different ensemble of models. So, these models will be generated by different these models will be generated by different algorithms learning algorithms may be that these models are nothing but an n-sample of different decision trees also. So, hence the bagging works because it reduces the variance of the predictions. So, this particular method reduces the variance of the predictions and by doing this averaging it automatically overcomes overcomes from the overfitting of problems of which we have seen in the single decision tree we will see more details in the further slides. So, bagging is an important idea which is essential to the understanding of the random forests. So, bagging is nothing but a bootstrap aggregation it is a general method for averaging the predictions of different learning algorithms and models and thereby. So, it is not only confined to the decision tree but any other algorithm many any

other learning algorithm is used here can be used for bootstrap aggregation. So, by bagging works because it reduces the variance of the predictions

Refer Slide Time: (14:20)

## Algorithm: Bagging



Let us understand the bagging algorithm in more details. So, input to the bagging algorithm is nothing but a training set that is  $Z$ , which comprises of  $\{(x_1, y_1), \dots, (x_n, y_n)\}$  and so on. So, where  $X$  is the features or in a data set and  $Y$  is basically the labels which are given as part of the target. So, the input as the training set is given and let us assume that another input which is also required in the bagging algorithm is the number of iteration let us say it is  $B$  and also we require one machine learning algorithm method let us say it is  $M$ . So, now the bagging says that for, for the iterations from 1 to  $B$  and in each in the step number 2 it says that it draws a bootstrap sample  $Z^*$   $Z^{*b}$  of size  $n$  from the training data. So, this particular  $Z^{*b}$  is a bootstrap sample that we have seen how to generate in the previous slide. So, once we draw a bootstrap sample that is called  $Z^{*b}$  the  $b$  is the  $b^{\text{th}}$  iteration we have generated this particular training data to be used in this iteration  $b^{\text{th}}$  iteration that is  $Z^{*b}$ . Now, we will apply the machine learning method  $M$  on this data set that is  $Z^{*b}$  and obtain a model let us call it as  $f(X)^{*b}$  that is at the  $b^{\text{th}}$  iteration the model will be  $f(X)^{*b}$  and finally after carrying out the iterations  $b$  we will generate the that is the ensemble collection of different models how many models  $b$  models capital  $B$  one for each. So, the first iteration will have  $f^{*1}, f^{*2}, \dots, f^{*B}$ . So, it will be ensemble of  $B$  different models we have generated in this iteration

then this particular model will be used in the prediction with ensemble. So, let us see that if the problem is the regression problem then the effects that is the prediction that is the predicted model the model for the regression model called  $f(x)$  means nothing but it is an average of all the prediction menus with the ensemble and if it is a classification then we will go for the majority of all the predictions and which is shown as every model will give out one prediction and the majority of that will be the other prediction with the ensemble to understand this let us consider an example. So, let us say that this is the decision tree and every DCL let us say decision tree number 1, 2, 3 and they are represented as  $f^{i_1} f^{i_2} f^{i_3}$  now these different trees let us say will give out result in the form of let us say fruits let us say this is orange and this particular tree will give a fruit which is called mango and third tree also will give out the result the prediction result let us say that it is again orange. Now, we have to do a voting majority voting. So, in that majority voting we will see that this orange is appearing twice hence the prediction output will be in the form of orange.

Refer Slide Time: (18:37)

## Why does Bagging work ?

- Model  $f(x)$  has higher predictive power than any single  $f^{xb}(x)$ ,  $b=1,\dots,B$
- Most of situations with any machine learning method in the core, the quality of such aggregated predictions will be better than of any single prediction.

### Why does bagging works?

- This phenomenon is based on a very general principle which is called the bias variance trade off. You can consider the training data set to be random by itself.

Now, the question is why the bagging works? So, the model  $f(x)$  has the higher predictive power than any single decision tree or any single model and so most of the situations with any machine learning method is the core the quality of such aggregated prediction will be much better than any single prediction and why does the bagging works this phenomena is based on the very general principle what is called as a bias variance trade off. So, you can consider the training data set to be the random by itself.

Refer Slide Time: (19:17)

## Why does Bagging work ?

- Why is it so? What is the training data set?
- In the real situation, the training data set may be a user behavior in Internet, for example, web browsing, using search engine, doing clicks on advertisement, and so on.
- Other examples of training data sets are physical measurements. For example, temperature, locations, date, time, and so on. And all these measurements are essentially stochastic.
- If you can repeat the same experiment in the same conditions, the measurements actually will be different because of the noise in measurements, and since user behavior is essentially stochastic and not exactly predictable. Now, you understand that the training data set itself is random.

▶ ⏪ 19:17 / 39:21 Big Data Computing

Predictive Analytics CC BY NC ND

So, in the real situation the training data set maybe the user behavior on the internet for example while web browsing and which web pages the user clicks using search engine while doing the while clicking on the advertisement and so on. Similarly, other examples of the training set data are the physical measurements for example temperature, location, date, time and so on. All these measurements are also essentially stochastic which are not same. So, if you can repeat the same experiment in the same condition the measurement actually will be different because of the noise in the measurement and since the user behavior essentially are stochastic. So, not exactly predictable and so you can understand this particular training data itself is random. So, because of this particular nature of the randomness in the data or a stochastic nature.

Refer Slide Time: (20:17)

## Why does Bagging work ?

- **Bagging:** It is an averaging over a set of possible datasets, removing noisy and non-stable parts of models.
- After averaging, the noisy parts of machine learning model will vanish out, whereas stable and reliable parts will remain. The quality of the average model will be better than any single model.

So, the bagging is also works. So, the bagging is an averaging over the set of possible data sets and removing noisy and non-stable part of the models will be achieve during bagging. So, after everything the noisy part of the machine learning model will vanish out whereby the stable and reliable part will remain in an effect. So, the quality of the average model will be much better than a single model. So, by this way we have understood that you know what is the bagging and by averaging the bagging due to the stochastic nature of the the data set generation will allow the bagging method to work and but the important part of the bagging is that so it will be averaging of different example predictions. So, this averaging is more accurate because the noisy part of that machine learning model will vanish out in this averaging process whereby the stable and reliable part will remain. So, the quality of average model will be much better than any single model here in this case of the bagging scenario

Refer Slide Time: (21:29)

## Summary

- **Bootstrap:** A method for generating different replicas of the dataset
- **Bagging (Bootstrap Aggregation):** A method for averaging predictions and reducing prediction's variance
- Bagging improves the quality of almost any machine learning method.
- **Bagging is very time consuming for large data sets.**

now in short let us summarize these two basic concepts which are going to be used in the random forest algorithm which we are going to discuss next. So, the bootstrap is a method for generating different replicas in the data set and bagging is an bootstrap averaging aggregation. So, this method is based on averaging predictions and reducing the predictions variance. So, bagging improves the quality of almost any machine learning method quality of prediction based on any machine learning method. So, bagging is very time consuming for the large data set

Refer Slide Time: (22:10)

## Random Forest

Now, let us see the random forest algorithm which uses the bagging and boosting methods.

Refer Slide Time: (22:20)

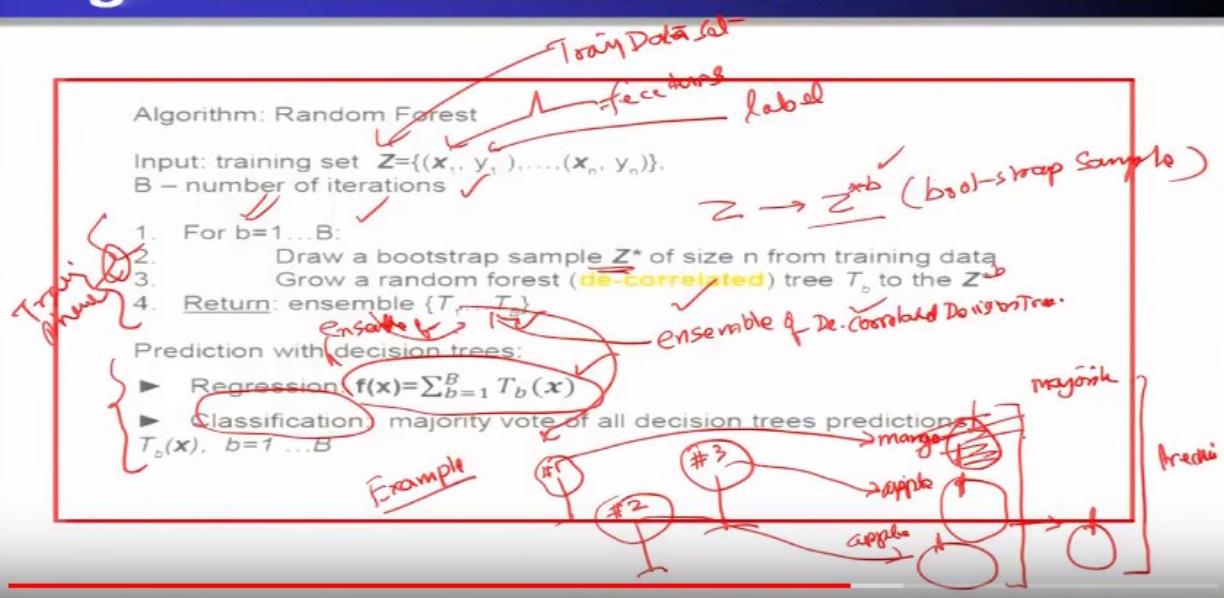
# Random Forest

- Random Forest Algorithm is a bagging of de-correlated decision trees.

So, random forest algorithm is bagging of the decorrelated decision trees. So, again i am repeating random forest algorithm is a is a bagging of decorrelated decision trees

Refer Slide Time: (22:36)

## Algorithm: Random Forest



what you mean by this let us understand the random forest using the algorithm of random forests. So, the algorithm of a random forest requires the input in the form of the training data set again  $x_i$  it represents the features of a data set and why  $y_i$  represents the label which is given in the training data set. So,  $Z$  is represented  $Z$  is a training data set as an input to the random forest algorithm and total number of

iterations let us call it as B now we will see how what steps are followed in each iterations for b is from 1 to B total number of iteration. Let, us see in each iteration that is let us say step number 2 it will draw a bootstrap sample which is called  $Z^*$  of the size n from the training data set. So, training data set Z will produce  $Z^*$ , let us say that  $Z^{*b}$  is the bootstrap sample for the iteration number b now then we will apply then we will grow the random forest that is nothing but decorrelated decision tree and let us call it as  $T_b$ . On this particular data set  $Z^{*b}$  and finally we will return this particular iteration will be repeated at 1 to B different iterations. So, that means we will generate a B different decorrelated decision trees and call it as  $T_1, T_2$  and so on up to capital B different trees and this is ensemble of the different decorrelated decision tree now using this particular training phase model which is called an ensemble of different decorrelated decision trees that will be the ensemble of models which will be generated by the training phase now this model will be used for the prediction with these ensemble tree decision trees ensemble of decision trees. So, if it is regression then we will now as far as the prediction is concerned which is nothing but an average of the prediction of all the models which will be given as the outputs and if it is a classification than the majority vote of all the decision trees will be prediction let us see again by the an example to understand what do you mean by the so classification will require the categorical output from every ensemble tree let us say that this particular ensemble or crease is represented by let us say a tree number one a tree number two and let us say it is having a tree number three these trees will generate an output let us say this tree will generate an mango as an output this tree will also generate let us say apple as an output and tree number two will again generate an apple as an output. So, now if we take at the majority then majority comes as an apple. So, this will be the prediction. So, prediction will suppress the noise part therefore these random forest trees are much more accurate as far as the prediction compared to the single decision trees

Refer Slide Time: (27:25)

## How to grow a random forest decision tree

- The tree is built **greedily** from top to bottom
- Select  $m \leq p$  of the input variables at random as candidates for splitting *Decorrelated decision trees will use subset of input variable in random order*
- Each split is selected to **maximize information gain (IG)**

$$IG = \text{Impurity}(Z) - \left( \frac{|Z_L|}{|Z|} \text{Impurity}(Z_L) + \frac{|Z_R|}{|Z|} \text{Impurity}(Z_R) \right)$$

**Error before split      Error after split**

▶ ▶ ⏪ 29:18 / 39:21 Big Data Computing Predictive Analytics CC BY HD

how to grow a random forest decision tree? So, the trees is built greedily from top to the bottom as we have seen in the in the decision trees but the difference here is that it will select  $m$  out of  $p$  different features. So, select  $m$  out of  $p$  input variables at random as the candidates for the splitting. So, in decision tree all variables are used as the candidate for splitting but here this is called decorrelated decision trees will use the subset of the input variables in random order. So, that is why it is called decorrelated decision tree because not all input variables are used but subset of the input variables are used and they are used for the tree induction that is the they will be used to decide the candidates for the splitting now each split is selected to maximize the information gained. So, information gain as we know that is dependent up on the impurity( $Z$ ) and that is the errors before the split and then errors after the split and which is shown by this particular equation.

Refer Slide Time: (29:20)

## How to grow a random forest decision tree

- Select  $m \leq p$  of the input variables at random as candidates for splitting
- **Recommendations from inventors of Random Forests:**
- $m = \sqrt{p}$  for classification, minInstance PerNode = 1 ✓
- $m = p/3$  for regression, minInstancePerNode=5

*Thumb rule*

$$\begin{cases} \text{Classification} - m = \sqrt{p} \\ \text{Regression} - m = \frac{p}{3} \end{cases}$$

Now, how to grow the random forest decision trees. So, select  $m$  out of  $p$  input variables at random as the candidate for the splitting. So, here the recommendations from inventors of random forests are summarized here so for classification the value of  $m$  has to be chosen as  $\sqrt{p}$  for the classification problem

and the minimum instance will be per node is 1 and  $m$  will be  $\frac{p}{3}$  for regression problems. So, for

classification the rule of thumb is thumb rule says that the value of  $m$  should be  $\sqrt{p}$  and for regression the

value of  $m$  is equal to  $\frac{p}{3}$ .

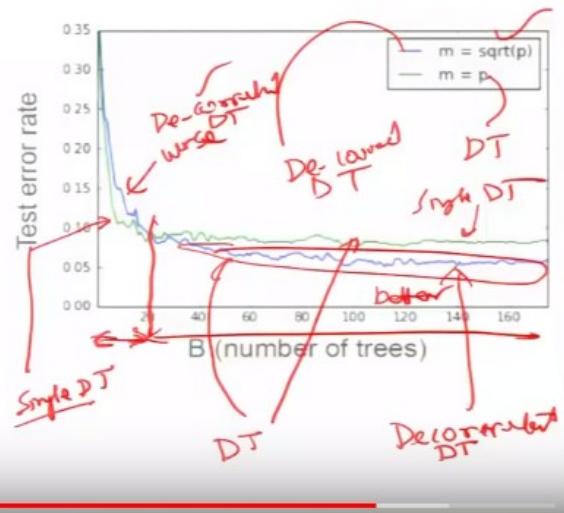
Refer Slide Time: (30:30)

## Random forest

Here are the results of training of two random force. The first variant is marked with green here, and either the variant were at each step  $m$  equals speed.

It means that at each step, we grow a regular decision tree and you find the best split among all the variables. And the blue line, it is a de-correlated decision tree.

In this situation, we randomly pick  $m$  equals square root of  $b$ . And all the trees can be built using different subsets of variables. As we can see at this diagram, at the initial stage, the variant is  $m$  equals square root  $b$  is worse before 20 iterations. But eventually, this variant of the Random Forest algorithm converges to the better solution.

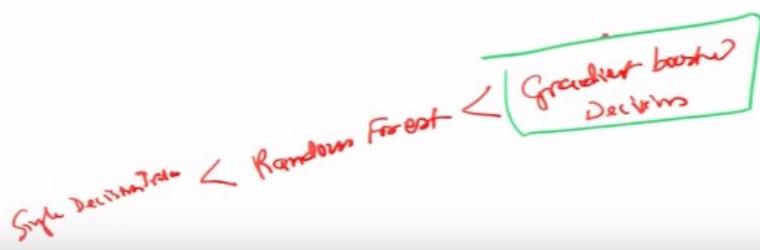


So, random forest here the results of the training of two random forest is shown the first variant is marked with the with the green one and and either the variant where each step  $m$  equals this particular case and it means that at each step we grow the regular decision trees and you find the best split among all the variables and the blue line that is it is the correlated here it is the decision tree and the blue one is decorrelated decision tree. So, this is a decorrelated decision tree and this is the decision tree which is shown as the green one. So, this means that at each step we grow the regular decision tree and you find the best split among all the variables and the blue line it is decorrelated decision tree. In this situation, we randomly pick  $m=\sqrt{b}$  that is shown over here square root and all the trees can be built using different subset variables and we can see in this particular diagram that initial stage the variant is is  $m=\sqrt{b}$  is worse before 20 iterations. So, before 20 iterations if we do this kind of analysis up to this point here this decorrelated is worse is going worse decorrelated decision tree is performing worse compared to the single decision tree but for the value which is more than 20 before the 20 iterations that means if the number of trees are less than 20 then the the decorrelated decision tree will perform not that good compared to the single decision tree but as the number of trees grows that is nothing but the number of iterations. So, if the number of trees are more than 20 then we see that this your decorrelated decision tree is performing much better start performing better compared to the single decision tree but equivalently this variant of random forest algorithm converges to a better solution that means after beyond 20 iterations or beyond 20 number of decision trees this particular random forest algorithm converges to a better solution.

Refer Slide Time: (33:34)

## Summary

- Random Forest is a good method for a **general purpose classification/regression problems** (typically slightly worse than gradient boosted decision trees)



So, random forest is a good method for general purpose classification regression problem typically, slightly worse than gradient boosted decision tree we will see that both the cases. So, that means we have seen that random forest is performing better is better than the single decision tree but the random forest is slightly worse than the gradient boosted decision tree. So, that means gradient boosted decision tree is going to be the best approach for a general purpose regression problems and for prediction problems that we will see in the next slide.

Refer Slide Time: (34:32)

## Summary

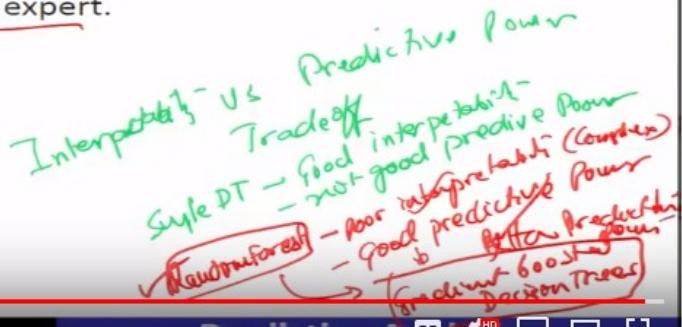
- **Automatically handle interactions of features:** Of course, this algorithm can automatically handle interactions of features because this could be done by a single decision tree.
- **Computational scalability:** Of course, this algorithm can automatically handle interactions of features because this could be done by a single decision tree.
- In the Random Forest algorithm, each tree can be built independently on other trees. It is an important feature and I would like to emphasize it. That is why the Random Forest algorithm is essentially parallel.
- The Random Forest could be trained in the distributed environment with the high degree of parallelization.

So, summary that is the automatically handles the interaction of the features in the decision tree and of course this algorithm can automatically handle the interaction of the features because this could be done by the single decision tree. So, also that particular part or is also covered in the random forest now computational scalability is also there in the random forest of course this algorithm can automatically handle the fraction of features because this could be done by the single decision tree. So, therefore in random forests also it ensures computationally scalability in the random forest algorithm each tree can be split independently on the other trees it is an important feature and i would like to emphasize it that is why a random forest algorithm is easiest and essentially it is parallel. So, random forests could be trained in a distributed environment with high degree of parallelization.

Refer Slide Time: (35:37)

## Summary

- **Predictive Power:** As far as predictive power of the Random Forest is, on the one hand better than a single decision tree, but it is slightly worse than gradient boosted decision trees.
- **Interpretability:** Here you'll lose the interpretability because their composition of hundreds or thousands Random Forest decision trees cannot be analyzed by human expert.



So, so another aspect of random forest is the predictive power as the predictive power of random forest is on one hand better than a single decision trees but it is slightly worse than gradient boosted decision trees. Another part is called interpretability, here the interpretability is not that good that means you lose the interpretability because of their composition of hundreds and thousands of simple decision trees involved in the random forest and therefore it cannot be analyzed in the interpretation compared to the single decision tree problem which are more interpreted, interpretable, but less predictive power, and so therefore interpretability what says predictive power is having a trade off. So, as far as the single decision tree will have the good interpretability but not that good predictive power whereas the random forest has poor interpretability that means due to the complex nature of the tree and sample and so it is not possible to be understood by the or analyzed by the human expert whereas the prediction is very good good predictive power is there in the random forest therefore for good prediction random forest is preferred compared to the single decision tree but there is another better scheme as far as the predictive power is concerned because random forest is already lost its good interpretability aspect. So, let us see how better, how the best we can do the prediction? So, another improvement of random forest is called a gradient boosted decision tree which is having much better predictive power compared to the decision trees.

**Lecture - 29**  
**Big Data Predictive Analytics**  
**(Part-II)**

Gradient boosted decision trees for regression.

Refer Slide Time :( 0:18)

## Boosting

- **Boosting:** It is a method for combining outputs of many weak classifiers to produce a powerful ensemble.
- There are several variants of boosting algorithms, AdaBoost, BrownBoost, LogitBoost, and Gradient Boosting.

Boosting: Boosting it is a method of combining outputs of many week classifiers, to produce a powerful and ensemble. There are several variants of boosting algorithms, AdaBoost, BrownBoost, LogitBoost and Gradient Boosting.

Refer Slide Time :( 0:37)

## Big Data

- Large number of training examples.
- Large number of features describing objects.
- In this situation, It is very natural to assume that you would like to train a really complex model, even having such a great amount of data and hopefully, this model will be accurate.
- There are two basic ways, in machine learning, to build complex models.
- The first way is to start with a complex model from the very beginning, and fit its parameters. This is exactly the way how neural network operates.
- And the second way is to build a complex model iteratively. You can build a complex model iteratively, where each step requires training of a simple model. In context of boosting, these models are called weak classifiers, or base classifiers.

Now, we see in the big data that, the size of the training examples is a very large. And also, a large number of, large number of features describing the objects are present. So, therefore, this one kind of scenario occurs in a situation of a big data. So, in this situation it is very natural to assume that, you would like to train a really complex model even having such a great amount of data and hopefully, this model will be accurate. There are two basic ways in the machine learning to build the complex models. The first one is that, you have to start with a very complex model from the very beginning and fits its parameters with the data. So, this is exactly the way how the neural networks operate. So they operate with the very complex model, in the beginning itself and they will fit the data, they will fit the parameters and the data will be now, then predicted based on that fitting of its parameters. Now, the second way is to build a complex model iteratively. So, we can build a complex model iteratively, that is, with each step requires the training of a simple model. So in the context of boosting, these models are called, ‘Weak Classifiers or the Base Classifiers. Therefore, it is an ensemble of weak classifiers and which makes this particular model iteratively.

Refer Slide Time :( 2:20)

## Regression

Given a training set:  $Z=\{(x_1, y_1), \dots, (x_n, y_n)\}$   
 Xi- features, yi-targets (real values)

Goal is to find  $f(x)$  using training set, such as

$$\min \sum_{(x, y) \in T} (f(x) - y)^2$$

At test set  $T=\{(x_1, y_1), \dots, (x_n, y_n)\}$   
 How to build  $f(x)$

Let us see, the regression. How this particular concept can be applied in the regression, in a gradient boosted our decision trees? So, let us assume that there are samples which are given, which are specified by  $Z$ , where  $x_i$  is the set of features, in the dataset and  $y_i$  is the label and  $x_1$  to  $x_n$  are  $n$  different samples, in the training dataset. Now, the goal here is to find  $f(x)$ , using this training set such that, this particular

$\min \sum_{(x,y) \in T} (f(x) - y)^2$ , the minimum such error will be there and so at the test set given, the test set, this error should be the minimum one, here in this case. How to build this  $f(x)$  is a question?

Refer Slide Time :( 3:28)

## Gradient Boosted Trees for Regression

### How to build such $f(x)$ ?

In boosting, our goal is to build the function  $f(x)$  iteratively. We suppose that this function  $f(x)$  is just a sum of other simple functions,  $h_m(x)$ .

And particular, you assume that each function  $h_m(x)$  is a decision tree.

$$f(\mathbf{x}) = \sum_{m=1}^M h_m(\mathbf{x})$$

$h_m(\mathbf{x})$  - a decision tree

So, how to build such an  $f(x)$ ? And boosting our goal is to find, is to build the function  $f(x)$  iteratively. So, we suppose that, this function  $f(x) = \sum_{m=1}^M h_m(x)$ . So, in particular you have a let us assume that, each function  $h_m$  is a decision tree. So, here each function is a decision tree. And the aggregation of it is basically the  $f(x)$  function.

Refer Slide Time :( 4:00)

## Algorithm: Gradient Boosted Trees for Regression

Algorithm: Gradient Boosted Trees for Regression

Input: training set  $\{(x_1, y_1), \dots, (x_n, y_n)\}$

✓ M – number of iterations

✓ 1.  $f_0(x) = \frac{1}{n} \sum_{i=1}^n y_i$  ✓ initial  $f_0(x)$  ← mean of label values from Training dataset

2. For  $m=1 \dots M$ :

3.  $\hat{y}_i = y_i - f_{m-1}(x_i)$  (residual)

4. Fit a decision tree  $h_m(x)$  to the targets  $\hat{y}_i$  (auxiliary training set  $\{(x_1, \hat{y}_1), \dots, (x_n, \hat{y}_n)\}$ )

5.  $f_m(x) = f_{m-1}(x) + v h_m(x)$  ✓ regularization coefficient ✓ new labels (residuals.)

6. Return  $f_m(x)$  ✓ regularization coefficient, recommended  $\leq 0.1$

Let us understand: The gradient boosted decision trees for regression in more details. So, gradient boosted trees for regression problem. So, let us take the input set of n different data that is training dataset, which consists of  $x$  is the set of features and  $y_i$  consists of the labels, so in turn this is the training dataset of n different samples which are given, as an input to the gradient boosted trees for regression. And M is the number of iterations and now, the step number one assumes the initial calculate the initial,  $f_0(x)$ . So that will be calculated by summing up all the label values and their average or their mean is assigned over here. So, it will take the mean of the label values from the training dataset and that becomes  $f_0$  or the initial  $f_0$  function, initial effects function. Now, then it will iterate for M iterations, wherein it will calculate the residual, which is  $\hat{y}_i$  is nothing but  $y_i - f_{m-1}(x_i)$  that is, of a previous iteration and so here, it will be 0 residual and so residuals are nothing but, the differences of values in the actually labels and the predicted labels that is called the ‘Residual’. Now, this after that, the next step would be to fit a decision tree  $h_m$ , to the targets that is  $y_i$ . And then, it that means, it will generate an auxiliary training set, out of this particular set where in,  $y_i$  will be replaced here as the labels, as the new labels which are nothing but the residuals. And then,  $f_m(x)$  will be calculated by giving up particular regularization, coefficient and therefore, it will calculate the value of  $f_m$  and in this manner it will iterate and compute all that things. Now, as far as the regularization, coefficient that is, nothing but I mean, it is recommended  $\leq 0.1$  here in this case, this is going to be important parameter in this gradient boosted trees for regression.

Refer Slide Time :( 7:37)

# Optimization Theory

- You may have noticed that **gradient boosting is somewhat similar to the gradient descent in the optimization theory.** If we want to minimize a function in the optimization theory using the gradient descent, we make a small step in the direction opposite to the gradient.
- Gradient of the function, by definition, is a vector which points to the direction with the fastest increase. Since we want to minimize the function, we must move to the direction opposite to the gradient. To ensure convergence, we must make very small steps. So you're multiplying each gradient by small constant, which is called step size. It is very similar to what we do in gradient boosting.
- And gradient boosting is considered to be a minimization in the functional space.



$$f_m(x) = f_0(x) + v h_1(x) + v h_2(x) + \dots$$

**Boosting**-Minimization  
in the functional space

So here, you might have noticed that, gradient boosting is somewhat similar to the gradient descent in the optimization theory. That if we want to minimize the function in the optimization theory using gradient descent, we make a small step in the direction opposite to the gradient. So, gradient of a function by definition is a vector which points to the direction with the fastest increase. Since we want to minimize the function, we must move to the direction opposite to the gradient, to ensure the convergence, we must make very small steps, so you are multiplying each gradient by a small constant, which is called a, 'Step Size'. It is very similar to what we do in the gradient boosting. So, gradient boosting is considered, minimization in the functionality space. That is  $f_m(x) = f_0(x) + v h_1(x) + v h_2(x) + \dots$ . So, boosting is the minimization in the functional space.

Refer Slide Time :( 8:38)

## Summary

- **Boosting** is a method for combining outputs of many weak classifiers or regressors to produce a powerful ensemble.
- **Gradient Boosting** is a gradient descent minimization of the target function in the functional space.
- **Gradient Boosting with Decision Trees** is considered to be the best algorithm for general purpose classification or regression problems.

So, let us see the summary and boosting is the method for combining the outputs of many weak classifiers or the regressor to produce a powerful and ensemble. And gradient boosting is a gradient decent minimization of the target functions in the functional space. So, gradient boosting with the decision tree is considered to be the best algorithm for general purpose classification or the regression problem. Now, let us see the gradient boosted decision trees for classification problem.

Refer Slide Time :( 9:13)

## Classification

Given a training set:  $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}$

$x_i$ - features,  $y_i$ -class labels (0,1)

**Goal** is to find  $f(x)$  using training set, such as

$$\min \sum_{(x, y) \in T} [f(x) \neq y]$$

At test set  $T = \{(x_1, y_1), \dots, (x_n, y_n)\}$

How to build  $f(x)$  ?

$f(x)$   $\rightarrow$   $\min$   $\rightarrow$  Aggregate  $\rightarrow$  mis-classification

Let us assume that, the training set which is given that is  $Z=\{(x_1, y_1), \dots, (x_n, y_n)\}$  and we are in  $x_1$  is nothing but they are the features and  $y_1$  is nothing but the class labels. So, here there will be a class label  $y$  because this is a classification problem so the classes lie between lies 0 & 1. So, class labels, let us assume that it is 0 & 1, so these will be assigned as the labels, in the training dataset. Now, goal here is to find out that, effects using the training dataset such that, it will minimize this particular function that is the  $\sum_{(x,y) \in T} [f(x) \neq y]$ . So that means, the label  $f(x)$  is the predicted value and  $Y$  is that the target label, if it is not matching, so that becomes an error and the summation of all such miss classification. So, the summation of so the aggregation of miss classification is to be minimized, so that should be the value of  $f(x)$ , which can achieve this so that, this particular prediction can be applied on the test dataset. So, test dataset  $T=\{(x_1, y_1), \dots, (x_n, y_n)\}$  So, how to build this  $f(x)$  is? The problem of gradient boosted decision trees.

Refer Slide Time :( 11:12)

## Gradient Boosted Trees for Classification

**How we are going to build such the function,  $f(x)$ ?**

We use a probabilistic model by using the following expression.

$$P(y = 1|x) = \frac{1}{1 + \exp(-\sum_{m=1}^M h_m(x))}$$

$h_m(x)$  - a decision tree

$$0 < P(y = 1|x) < 1$$

We model the probability of belonging of an object to the first class. And here inside the exp, there is the sum of  $h_m(x)$ , and each  $h_m(x)$  is a decision tree.

We can easily check that such expression for probability will be always between zero and one, so it is normal regular probability.

So, gradient boosted trees for classification. Here, we have to see how we are going to build, such a function  $f(x)$ . So, we use the probabilistic method by using the following expression, so the probability

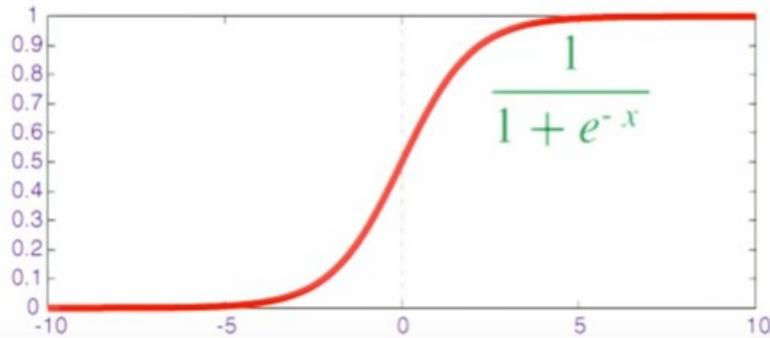
$$P(y=1|x) = \frac{1}{1 + \exp(h(x))}$$

where  $h(x)$  is the decision tree. So, therefore this the value of  $0 < P(y=1|x) < 1$ . Therefore, we model the probability of belonging of an object to the first class and here inside the exponential, there is a sum of  $h_m$ 's and each  $h_m(x)$  is a decision tree. So, we can easily check that each expression or the probability will always be between 0 and 1. So, it is the normal regular probability.

Refer Slide Time :( 12:12)

## Sigmoid Function

This function is called the sigmoid function, which maps all the real values into the range between zero and one.



This particular function is called the, ‘Sigmoid Function’, which maps all the real values, into the range between 0 & 1. So, this particular sigmoid function is  $\frac{1}{1+e^{-x}}$ .

Refer Slide Time :( 12:29)

Let us denote the sum of all  $h_m(x)$  by  $f(x)$ . It is an ensemble of decision trees. Then you can write the probability of belonging to the first class in a simple way using  $f(x)$ . And the main idea which is used here is called the principle of maximum likelihood.

**What is it? First of all, what is the likelihood?**

Likelihood is a probability of absorbing some data given a statistical model. If we have a data set with  $n$  objects from one to  $n$ , then the probability of absorbing such data set is the multiplication of probabilities for all single objects. This multiplication is called the likelihood.

$$f(\mathbf{x}) = \sum_{m=1}^M h_m(\mathbf{x})$$
$$P(y = 1 | \mathbf{x}) = \frac{1}{1 + \exp(-f(\mathbf{x}))}$$

Likelihood:

$$\prod_{i=1}^n P(y_i | \mathbf{x}_i) = P(y_1 | \mathbf{x}_1) \cdot \dots \cdot P(y_n | \mathbf{x}_n)$$

So, here also the same equation is appearing  $\frac{1}{1+\exp(-f(x))}$ . This is called sigmoid function. So, let us

denote  $f(x) = \sum_{m=1}^M h_m(x)$ . It is an ensemble of decision trees, then you can write the probability of the

belonging to a 1st class in a simple way using  $f(x)$  and the main idea which is used here is called the, ‘Principle of Maximum Likelihood’. So, what is this principle of like maximum likelihood? So, likelihood is a probability of absorbing some data, given the statistical model, if we have the dataset, with an object from 1 to n, then the probability of absorbing such dataset is the multiplication of probabilities of all single objects, the multiplication is called, ‘Likelihood’. And here, that can be expressed, likelihood

as the multiplication of the probabilities for all single objects that is nothing but the  $\prod_{i=1}^n p_i = p$ . So, therefore it comes out to be the multiplication of all the probabilities.

Refer Slide Time :( 13:53)

## The principle of maximum likelihood

- **Algorithm:** find a function  $f(x)$  maximizing the likelihood
- **Equivalent:** find a function  $f(x)$  maximizing the logarithm of the likelihood
- (since logarithm is a monotone function)

$$Q[f] = \sum_{i=1}^n \log(P(y_i|x_i))$$

$$\max Q[f]$$

And now, likelihood function we have to calculate. So, the principle of maximum likelihood, can be given by this algorithm, to find a function  $f(x)$  which maximizing the likelihood, which is equivalent to find  $f(x)$  maximizing the logarithmic of the likelihood. Since, the logarithmic is the monotone function. So that can be represented here in this case that,

$$Q[f] = \sum_{i=1}^n \log p_i$$

And we have to find out  $\max Q[f]$ . That is, which will maximize the likelihood. So, we have to find out that  $f(x)$ , which will maximize the likelihood.

Refer Slide Time :( 14:45)

## The principle of maximum likelihood

We will denote by  $Q[f]$  the logarithm of the likelihood, and now, it is sum of all logarithms of probabilities and you are going to maximize this function.

We will use shorthand for this logarithm  $L(y_i, f(x_i))$ . It is the logarithm of probability. And here, we emphasize that this logarithms depend actually on the true label,  $y_i$  and our prediction,  $f(x_i)$ . Now,  $Q[f]$  is a sum of  $L(y_i, f(x_i))$ .

$$L(y_i, f(x_i)) = \log(P(y_i|x_i))$$
$$Q[f] = \sum_{i=1}^n L(y_i, f(x_i))$$

predicted label  
True label for i<sup>th</sup> data object

Hence, we have to fit our distribution in this dataset, by way of principle of maximum likelihood. So, we will denote it by  $Q(f)$ , the logarithmic of the likelihood and it is now, it is the sum of all the logarithm logarithms of the probabilities and you are going to maximize this particular function. Now, you will use the shorthand, for this logarithmic that is  $L$ . So, it is the logarithmic of the probability. And here, we emphasize that this logarithms, depends actually on the true label that is  $y_i$  and our predicted values that

$$is f(x_i) for i. And now, Q(f) = \sum_{i=1}^n L(y_i, f(x_i))$$

So, logarithmic, a  $L$  is nothing but, given as likely, given as the log of the probability of  $Y$  given  $x_i$  and which is nothing but,  $Q(f) = \sum_{i=1}^n L(y_i, f(x_i))$ . So,  $y_i$  is the, the true label, for the idea the data object in the set and  $f(x_i)$  is, the predicted label and this logarithmic of, of this is represented by this likelihood and the summation of this is represented by this, likelihood which has to be maximized for that key way.

Refer Slide Time :( 16:42)

## Algorithm: Gradient Boosted Trees for Classification

*feature → label → category*

Input: training set  $Z = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ ,  
 $M$  – number of iterations

1.  $f_0(x) = \log \frac{p_1}{1-p_1}$ ,  $p_1$  - part of objects of first class
2. For  $m=1 \dots M$ :
3.  $\checkmark g_i = \frac{dL(y_i, f_m(x_i))}{df_m(x_i)}$  ← gradient
4. Fit a decision tree  $h_m(x_i)$  to the target  $g_i$   
(auxiliary training set  $\{(\mathbf{x}_1, g_1), \dots, (\mathbf{x}_n, g_n)\}$ )
5.  $\rho_m = \underset{\rho}{\operatorname{argmax}} Q[f_{m-1}(x) + \rho h_m(x)]$  ←
6.  $f_m(x) = f_{m-1}(x) + \rho_m h_m(x)$
7. Return:  $f_M(x)$

$v$  - regularization (learningRate), recommended  $\leq 0.1$

Let us see the gradient boosted trees for classification, by putting it everything whatever we have discussed. Now, the algorithm for gradient boosted trees for classification. Here, the input set  $Z$  is given as,  $x_i, y_i$  where in  $x_i$  you know that, it is a features in the dataset and why is the labels, which are assigned as, the categorical type that is labels given. And  $M$  is the number of iterations and the for, the first class, the part of the objects of the first class is

$$f_0(x) = \log \frac{p_1}{1-p_1}$$

and for, iterating between from 1 to  $M$ , so we will find out the gradient

$$g_i = dL \quad \text{gradient}$$

So, this will calculate divided by differentiation of  $f_m(x)$ . So, this is called the, ‘Gradient’. So, gradients are calculated and then, it will fit a decision tree  $h_m$  of  $x$ , to the target, to the target  $G_i$ . So, auxiliary training dataset, which will be used here, is that,  $x_1$  and  $x_1$  and then it will be replaced by, label will be replaced by the gradients and this will call an, ‘Auxiliary dataset’. So, given the other rate dataset, it will fit the decision tree that is called, ‘ $h_m(x_i)$ ’. And wherein the role value will be

$$\rho_m = \underset{\rho}{\operatorname{argmax}} Q[f_{m-1}(x) + \rho h_m(x)]$$

So,  $\rho_m$  will be calculated and  $f_m(x) = f_{m-1}(x) + v \rho_m h_m(x_i)$ ,  $v$  is the regularization coefficient  $\rho_m$  and  $h_m(x_i)$ . So, this process will in turn, will give the  $x$  symbol of different values. And so, it will do this kind

of classification in this particular manner. So, let us see the stochastic boosting, so gradient boosting trees for classification, if we use this stochastic boosting.

Refer Slide Time :( 19:49)

## Algorithm: Gradient Boosted Trees for Classification + Stochastic Boosting

Input: training set  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ , M – number of iterations

1.  $f_0(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n y_i$
2. For  $m=1 \dots M$ :
3.  $g_i = \frac{dL(y_i, f_m(\mathbf{x}_i))}{df_m(\mathbf{x}_i)}$
4. Fit a decision tree  $h_m(\mathbf{x}_i)$  to the target  $g_i$ . ✓  
(auxiliary training set  $\{(\mathbf{x}_1, g_1), \dots, (\mathbf{x}_k, g_k)\}$ ),  $k=0.5n$   
*created by random sampling with replacement* || random forest
5.  $p_m = \underset{p}{\operatorname{argmax}} Q[f_{m-1}(\mathbf{x}) + p h_m(\mathbf{x})]$
6.  $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + v p_m h_m(\mathbf{x})$
7. Return  $f_M(\mathbf{x})$

$v$  - regularization (learningRate), recommended  $\leq 0.1$

Then it will be represented here in this case, we are this observe a set, which is used as the training set will be now,  $k = 0.5n$  will be created by the random sampling with the replacement, so this particular part here, we are going to use the, the concept of the random forest, for the bootstrap generation of the data side. So, this is the gradient boosting, gradient boosted trees + stochastic boosting is shown over here.

Refer Slide Time :( 20:32)

## Sampling with Replacement

n=8

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|

k=4

|   |   |   |   |
|---|---|---|---|
| 7 | 3 | 1 | 3 |
|---|---|---|---|

And this way, sampling with the replacement is there.

Refer Slide Time :( 20:37)

## Algorithm: Gradient Boosted Trees for Classification + Stochastic Boosting

Input: training set  $\{(x_1, y_1), \dots, (x_n, y_n)\}$ , M – number of iterations

1.  $f_0(x) = \frac{1}{n} \sum_{i=1}^n y_i$
2. For m=1...M:
  3.  $g_i = \frac{dL(y_i, f_m(x_i))}{df_m(x_i)}$
  4. Fit a decision tree  $h_m(x_i)$  to the target  $g_i$ ,  
(auxiliary training set  $\{(x_1, g_1), \dots, (x_k, g_k)\}$ )  $k=0.5n$   
*created by random sampling with replacement*
  5.  $p_m = \underset{p}{\operatorname{argmax}} Q[f_{m-1}(x) + p h_m(x)]$
  6.  $f_m(x) = f_{m-1}(x) + v p_m h_m(x)$
7. Return  $f_M(x)$

v - regularization (learningRate), recommended  $\leq 0.1$

random forest

So, here we have to see that the size of the sample will be reduced by  $k=0.5n$  that is here, the author a training set, will be reduced by the size half and it will be created, by random sampling with the replacement. And this is called the, ‘Concept of Bagging’, which will be used over here in the stochastic boosting.

Refer Slide Time :( 21:07)

# Sampling with Replacement

n=8 ✓

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|

k=4 ✓

|   |   |   |   |
|---|---|---|---|
| 7 | 3 | 1 | 3 |
|---|---|---|---|



So, sampling with the replacement is used but here, the value of K will become half of, the size of the total features. So here, it will be half of that particular features, will be taken up into the considerations.

Refer Slide Time :( 21:26)

## Algorithm: Gradient Boosted Trees for Classification + Stochastic Boosting

Input: training set  $\{(x_1, y_1), \dots, (x_n, y_n)\}$ , M – number of iterations

1.  $f_0(x) = \frac{1}{n} \sum_{i=1}^n y_i$
2. For m=1..M:
  3.  $g_i = \frac{dL(y_i, f_m(x_i))}{df_m(x_i)}$
  4. Fit a decision tree  $h_m(x_i)$  to the target  $g_i$   
(auxiliary training set  $\{(x_1, g_1), \dots, (x_k, g_k)\}, k=0.5n$   
*created by random sampling with replacement*)
  5.  $p_m = \underset{p}{\operatorname{argmax}} Q[f_{m-1}(x) + p h_m(x)]$
  6.  $f_m(x) = f_{m-1}(x) + v p_m h_m(x)$
  7. Return  $f_M(x)$

v - regularization (learningRate), recommended  $\leq 0.1$

A handwritten note on the slide: "random forest" with an arrow pointing to the final step of the algorithm.

Hence, as we reduce the size, this becomes more efficient this particular case.

Refer Slide Time :( 21:33)

## Tips for Usage

- First of all, it is important to understand how the regularization parameter works. In this figure, you can see the behavior of the gradient boosted decision trees algorithm with two variants of this parameter, 0.1 and 0.05.
- What happens here, at the initial stage of learning the variant with parameter 0.1 is better because it has lower testing error.

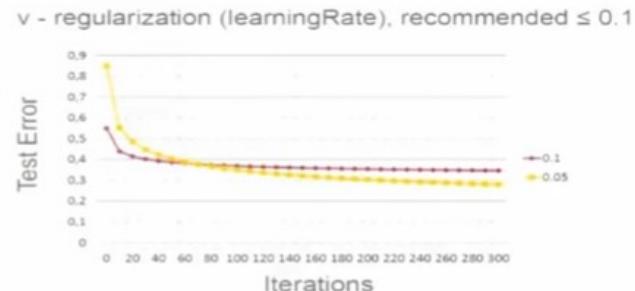


So, let us see the tips for the usage, first of all it is important to understand how the regularization parameter works. In this figure, we can see that the behavior of the gradient boosted, decision trees with the, with, with the two variants we can see with the parameter, .1 & 0 & 0.5 that is 0.5. So, we can see that, this one 0.1 is not that accurate and 0.5 is more accurate. So, what happens, here is that initially stage of the learning, at the initial stage the learning, the, the variant that is 0.1 is better but because, it has the lower testing error.

Refer Slide Time :( 22:21)

## Tips for Usage

- At each iteration, you measure a testing error of our ensemble on the hold out data set.
- But eventually, the variant with lower regularization, 0.05, reach lower testing error. Finally, this variant turns out to be superior.
- It is a very typical behavior, and you should not stop your algorithm after several dozen iterations. You should proceed until convergence.
- Convergence happens when your testing error doesn't change a lot. The variant with lower regularization converges more slowly, but eventually it builds a better model.



But, later on so at each iteration, you measure the testing error, up our example of on the holed out dataset. But, eventually the variant with the lower regularization that is 0.5, reach the lower testing error, finally this variant turns out to be the superior. So, it is very typical behavior and you should not stop your algorithm after several dozens of iterations, so you should proceed over until it converges. So, convergence happens when you're testing error does not change a lot. The variant with the lower regularization convert more slowly but eventually it builds a better model.

Refer Slide Time :( 23:04)

## Tips for Usage

- The recommended **learningRate** should be less or equal than 0.1.
- The bigger your data set is, the larger **number of iterations** should be.
- The recommended **number of iterations** ranges from several hundred to several thousand.
- Also, the more features you have in your data set, the deeper your decision tree should be.
- These are very general rules because the bigger your data set is, the more features you have, the more complex model you can build without overfitting.

So, the recommended learning rate should be less than or equal to 0.1 that we have already seen. And the bigger your dataset is the larger it will be the number of iterations it should have. So, the recommended number of iteration ranges from several hundred to the several thousands. Also, the more features you have in your dataset, the deeper will be your decision tree. And there are many general rules because the bigger your dataset is the more features you have the more complex model you can build without over fitting in this particular scenario.

Refer Slide Time :( 23:39)

## Summary

- It is a **best method** for a general purpose classification and regression problems.
- It **automatically handles interactions** of features, because in the core, it is based on decision trees, which can combine several features in a single tree.
- But also, this algorithm is **computationally scalable**. It can be effectively executed in the distributed environment, for example, in Spark. So it can be executed at the top of the Spark cluster.

So, somebody it is the best method, for general-purpose classification and regression problems. So, it automatically handles the interaction of the features, because in the core, it is based on the decision trees, which can combine several features in a single tree. But also, this algorithm is computationally scalable. It can be effectively executed in the distributed environment, for example, in the SPARK. So, it can be executed on top of the spark cluster.

Refer Slide Time :( 24:09)

## Summary

- But also, this algorithm has a very **good predictive power**. But unfortunately, the models are not interpretable.
- The **final ensemble effects is very large** and cannot be analyzed by a human expert.
- There is always a tradeoff in machine learning, between predictive power and interpretability, because the more complex and accurate your model is, the harder is the analysis of this model by a human.

But also, this algorithm has very good predictive power. But, unfortunately the model is not interpretable. And that problem we have also seen in the random forest but here, the predictive power is better than that

in the forest. So, the final and simple effect is very large and cannot be analyzed by the human experts. So, obviously it is not having the good interpretation of this due to the complex, nature of this particular model. So, there are always a trade-off in the machine learning, between the predictive power and interpretability, because the more complex and accurate your model is the harder is the analysis of this model, to be interpreted by the humans.

Refer Slide Time :( 24:53)

## **Spark ML, Decision Trees and Ensembles**

Spark ML, based decision trees and examples, we have to see the programming aspect of it, how using is part ml we will now use, the decision tree and decision tree and ensemble.

Refer Slide Time :( 25:10)

### **Introduction**

- This lesson will be about using Spark ML for doing classification and regression with decision trees, and in samples of decision trees.

So here, we will talk about a spark ML for doing classification regression, with the ensemble trees.

Refer Slide Time :( 25:19)

- First of all, you are going to create SparkContext and SparkSession

```
In []: from pyspark import SparkContext
 from pyspark.sql import SparkSession

In []: sc = SparkContext(appName = "module3_week4")

In []: ! echo $PYSPARK_SUBMIT_ARGS
 ✓

In []: spark = SparkSession.Builder().getOrCreate() # required for dataframes
```

And first we will see that, we have to, we have to first see the decision trees how the decision tree will be implemented over the spark image and then we can extend it for the random forest and gradient boosted trees. So, the first step here is to create the spark context and in the spark session. Which is shown here, in this particular steps that we have to create the spark context and we have to also create the spark session.

Refer Slide Time :( 25:59)

- Now you are downloading a dataset.

```
In [*]: !wget https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data
 ✓
--2017-07-31 11:09:06-- https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.249
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.249|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 124103 (121K) [text/plain]
Saving to: 'wdbc.data.11'

100%[=====] 124,103 158KB/s in 0.8s
2017-07-31 11:09:07 (158 KB/s) - 'wdbc.data.11' saved [124103/124103]
```

```
In []: !head -n 3 wdbc.data
```

```
In []: !wc -l wdbc.data
```

And this is spark session is required for building the data frame. And then, we can download the dataset and now we can see the dataset after downloading.

Refer Slide Time :( 26:10)

- Let's explore this dataset a little bit. The first column is ID of observation. The second column is the diagnosis, and other columns are features which are comma separated.

```
In [6]: !head -n 3 wdbc.data
842302,M,17.99,10.38,122.8,1001,0.1184,0.2776,0.3001,0.1471,0.2419,0.07871,1.09
5,0.9953,8.589,153.4,0.006399,0.04904,0.05373,0.01587,0.03003,0.006193,25.38,1
7.33,184.6,2019,0.1622,0.6656,0.7119,0.2654,0.4601,0.1189
842517,M,20.57,17.77,132.9,1326,0.08474,0.07864,0.0869,0.07017,0.1812,0.05667,
0.5435,0.7339,3.398,74.08,0.005225,0.01308,0.0186,0.0134,0.01389,0.003532,24.9
9,23.41,158.8,1956,0.1238,0.1866,0.2416,0.186,0.275,0.08902
84300983,M,19.69,21.25,130,1203,0.1096,0.1599,0.1974,0.1279,0.2069,0.05999,0.74
56,0.7869,4.585,94.03,0.00615,0.04006,0.03832,0.02058,0.0225,0.004571,23.57,25.
53,152.5,1709,0.1444,0.4245,0.4504,0.243,0.3613,0.08758

In [7]: !wc -l wdbc.data
#1) ID number
#2) Diagnosis (M = malignant, B = benign)
#3) Features

In [8]: from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import StringIndexer

In [9]: # Load a text file and convert each line to a Row.
```

The dataset into data frames and then we can see the features that, it has the data has an ID number and the label that is categorical label, which can be denoted by 0 & 1 or you can see that it is, it is M and B, finally which will be converted into 1 & 0. So, this particular dataset has the, the features and it has the labels and it has an ID. So, there are three different important parameter sighs. So, let us take this particular case, where all these important parts are, there in the dataset. So, now we can further look into the dataset.

Refer Slide Time :( 27:05)

- So these features are results of some analysis and measurements. There are total 569 examples in this dataset.

```
In [6]: !head -n 3 wdbc.data
842302,M,17.99,10.38,122.8,1001,0.1184,0.2776,0.3001,0.1471,0.2419,0.07871,1.09
5,0.9953,8.589,153.4,0.006399,0.04904,0.05373,0.01587,0.03003,0.006193,25.38,1
7.33,184.6,2019,0.1622,0.6656,0.7119,0.2654,0.4601,0.1189
842517,M,20.57,17.77,132.9,1326,0.08474,0.07864,0.0869,0.07017,0.1812,0.05667,
0.5435,0.7339,3.398,74.08,0.005225,0.01308,0.0186,0.0134,0.01389,0.003532,24.9
9,23.41,158.8,1956,0.1238,0.1866,0.2416,0.186,0.275,0.08902
84300983,M,19.69,21.25,130,1203,0.1096,0.1599,0.1974,0.1279,0.2069,0.05999,0.74
56,0.7869,4.585,94.03,0.00615,0.04006,0.03832,0.02058,0.0225,0.004571,23.57,25.
53,152.5,1709,0.1444,0.4245,0.4504,0.243,0.3613,0.08758

In [7]: !wc -l wdbc.data
569 wdbc.data

In [8]: #1) ID number
#2) Diagnosis (M = malignant, B = benign)
#3) Features

In [9]: from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import StringIndexer
```

You can see that, these datasets. How, so these datasets, so these features are the result of the analysis and it has around 569 different examples, in the dataset.

Refer Slide Time :( 27:24)

- First of all you need to transform the label, which is either M or B from the second column. You should transform it from a string to a number.
- You use a StringIndexer object for this purpose, and first of all you need to load all these datasets.

```
In [6]: !head -n 3 wdbc.data
842302,M,17.99,10.38,122.8,1001,0.1184,0.2776,0.3001,0.1471,0.2419,0.07871,1.09
5.0,9853,8.589,153.4,0.006199,0.04984,0.05373,0.01587,0.03003,0.006193,25.38,1
7.33,184.6,2019,0.1622,0.6656,0.7119,0.2654,0.4601,0.1189
842517,M,20.57,17.77,152.9,1326,0.08474,0.07864,0.0869,0.07917,0.1812,0.05667,
0.5435,0.7339,3.398,74.08,0.005225,0.01308,0.0186,0.0134,0.01389,0.003532,24.9
9.23,41,158.6,1956,0.1238,0.1866,0.2416,0.186,0.275,0.08902
84300903,M,19.69,21.29,136,1203,0.1096,0.1599,0.1974,0.1279,0.2069,0.05999,0.74
56,0.7869,4.585,94.03,0.00615,0.04906,0.03832,0.02858,0.0225,0.004571,23.57,25.
53,152.5,1709,0.1444,0.4245,0.4504,0.243,0.3613,0.08758

In [7]: !wc -l wdbc.data
569 wdbc.data

In []: #1) ID number
#2) Diagnosis (M = malignant, B = benign)
#3) Features
```

```
In []: from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import StringIndexer ✓
```

And what we, we all need to transform this label, which is given in M or B to the, to the numeric value and using string index or object, we can do this and that is the string index our object will create this, into the label values.

Refer Slide Time :( 27:48)

- Then you create a Spark DataFrame, which is stored in the distributed manner on cluster.

```
In [9]: from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import StringIndexer

In [10]: # Load a text file and convert each line to a Row.
data = []
with open("wdbc.data") as infile:
 for line in infile:
 tokens = line.rstrip("\n").split(",")
 y = tokens[0]
 features = Vectors.dense([float(x) for x in tokens[2:]])
 ✓ data.append((y, features))

In [11]: inputDF = spark.createDataFrame(data, ["label", "features"]) ✓
In [*]: inputDF.show()

In []: stringIndexer = StringIndexer(inputCol = "label", outputCol = "labelIndexed")
si_model = stringIndexer.fit(inputDF)
inputDF2 = si_model.transform(inputDF)
```

Into the numeric values and then we can create the data frame, which is stored in the distributed manner on the on the cluster. So, this particular data frame will be, created with the label and the feature, so these two part will be used up in the data frame, which will be input to our system. So, string indexer and then it will transform,

Refer Slide Time :( 28:11)

- **inputDF DataFrame has two columns, label and features. We use an object vector for creating a vector column in this dataset.**

```
In [11]: inputDF = spark.createDataFrame(data, ["label", "features"])

In [12]: inputDF.show()

+---+-----+
|label| features|
+---+-----+
|M|[17.99,10.38,122....]
|M|[28.57,17.77,132....]
|M|[19.69,21.25,130....]
|M|[11.42,20.38,77.5....]
|M|[20.29,14.34,135....]
|M|[12.45,15.7,82.57....]
|M|[18.25,19.98,119....]
|M|[13.71,20.83,90.2....]
|M|[13.0,21.82,87.5,...]
|M|[12.46,24.04,83.9....]
|M|[16.02,23.24,102....]
|M|[15.78,17.89,103....]
|M|[19.17,24.8,132.4....]
|M|[15.85,23.95,103....]
|M|[13.73,22.61,93.6....]
|M|[14.54,27.54,96.7....]
|M|[14.68,20.13,94.7....]
```

so that, all the values will become here. So, all these labels will be converted into the values using string indexer.

Refer Slide Time :( 28:18)

- And now label M is equivalent 1, and B label is equivalent 0.

```
In [14]: inputDF2.show()
```

| label | features               | labelIndexed |
|-------|------------------------|--------------|
| M     | [17.99,10.38,122....]  | 1.0          |
| M     | [28.57,17.77,132....]  | 1.0          |
| M     | [19.69,21.25,130....]  | 1.0          |
| M     | [11.42,20.38,77.5....] | 1.0          |
| M     | [20.29,14.34,135....]  | 1.0          |
| M     | [12.45,15.7,82.57....] | 1.0          |
| M     | [18.25,19.98,119....]  | 1.0          |
| M     | [13.71,20.83,90.2....] | 1.0          |
| M     | [13.0,21.82,87.5,...]  | 1.0          |
| M     | [12.46,24.04,83.9....] | 1.0          |
| M     | [16.02,23.24,102....]  | 1.0          |
| M     | [15.78,17.89,103....]  | 1.0          |
| M     | [19.17,24.8,132.4....] | 1.0          |
| M     | [15.85,23.95,103....]  | 1.0          |
| M     | [13.73,22.61,93.6....] | 1.0          |
| M     | [14.54,27.54,96.7....] | 1.0          |
| M     | [14.68,20.13,94.7....] | 1.0          |
| B     | [19.81,22.15,130....]  | 0.0          |
| B     | [13.54,14.36,87.4....] | 0.0          |

So now, after index, label index and all these label values are converted, from letter to the numeric values. So, here can have this numeric value either 1 or it is 0, as the label values. So, we have now two important things, one is the feature and the other is label indexed, in the data frame which is shown over here, this particular data frame will be now used for further analysis.

Refer Slide Time :( 28:55)

- First of all, you make training test splitting in the proportion 70% to 30%. And the first model you are going to evaluate is one single ~~decision tree~~ ~~Text~~.

Train Test

```
train/test split
In [15]: \trainingData, testData = inputDF2.randomSplit([0.7, 0.3], seed = 23)

Training Decision Tree
In []: from pyspark.ml.classification import DecisionTreeClassifier
 from pyspark.ml.evaluation import MulticlassClassificationEvaluator

In []: decisionTree = DecisionTreeClassifier(labelCol = "labelIndexed")

In []: dtModel = decisionTree.fit(trainingData)
In []: dtModel.numNodes
In []: dtModel.depth
In []: dtModel.featureImportances
```

Now, then we will make the training test is splitting, into the proportion 70 to 30 that when 70% is the training portion and 30% will be the test portion of the dataset. So, first we model with that 70% dataset to train the model on a single decision tree and then we will use the 30% of the dataset for testing purpose. So, this is to be done in this particular manner using random split, of 70 by 30 and then we will train the decision tree model, using decision tree classifier and we'll apply the label in text, in this particular case and then we will fit, the decision tree on the training dataset. So, this will become this will build a model called, 'Decision Tree Model'. It will build and this will be represented as DT model, as the variable. Now, you can see this particular model has how many nodes? What is the depth? And what are the important features? And so on.

Refer Slide Time :( 30:14)

- Here we are making import DecisionTreeClassifier object.
- We create a class which is responsible for training, and
- Then call the method fit to the training data, and obtain a decision tree model.

train/test split

```
In [15]: \trainingData, testData = inputDF2.randomSplit([0.7, 0.3], seed = 23)

Training Decision Tree
In []: from pyspark.ml.classification import DecisionTreeClassifier
 from pyspark.ml.evaluation import MulticlassClassificationEvaluator

In []: decisionTree = DecisionTreeClassifier(labelCol = "labelIndexed")

In []: dtModel = decisionTree.fit(trainingData)
In []: dtModel.numNodes
In []: dtModel.depth
In []: dtModel.featureImportances
```

And so after the, so these things now we are making, the import decision tree classifier we can create the class that is responsible for the training and then we call the, 'Method fit', to the training data and obtain the decision tree model that we have shown.

Refer Slide Time :( 30:36)

- So the training was quite fast, and here are some results.
- Number of nodes and depths of decision tree, feature importance, total number of features used in this decision tree, and so on.

```
In [17]: decisionTree = DecisionTreeClassifier(labelCol = "labelIndexed")
In [18]: dtModel = decisionTree.fit(trainingData)
In [19]: dtModel.numNodes
Out[19]: 29
In [20]: dtModel.depth
Out[20]: 5
In [21]: dtModel.featureImportances
Out[21]: SparseVector(30, {1: 0.0589, 6: 0.0037, 10: 0.0112, 13: 0.0117, 20: 0.0324, 21: 0.0302, 22: 0.7215, 24: 0.01, 26: 0.0191, 27: 0.1013})
In [22]: dtModel.numFeatures
Out[22]: 30
In [23]: print dtModel.toDebugString
```

Now, so the training was done in a very fast manner and now the we have to see the results, so the number of nodes and depth out the decision tree and a feature importance and number of features used in the decision tree and so on. These different things we can inspect.

Refer Slide Time :( 31:01)

- We can even visualize this decision tree and explore it, and here is a structure of this decision tree.
- Here are splitting conditions If and Else, which predict values and leaves of our decision trees.

```
In [23]: print dtModel.toDebugString
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_4653be4ce1bd9e58906
4) of depth 5 with 29 nodes
If (feature 22 <= 114.2)
 If (feature 27 <= 0.1613)
 If (feature 20 <= 16.57)
 If (feature 27 <= 0.1258)
 If (feature 10 <= 0.9289)
 Predict: 0.0
 Else (feature 10 > 0.9289)
 Predict: 1.0
 Else (feature 27 > 0.1258)
 If (feature 21 <= 32.85)
 Predict: 0.0
 Else (feature 21 > 32.85)
 Predict: 1.0
 Else (feature 20 > 16.57)
 If (feature 1 <= 16.54)
 Predict: 0.0
 Else (feature 1 > 16.54)
 If (feature 24 <= 0.1084)
 Predict: 0.0
 Else (feature 24 > 0.1084)
 Predict: 1.0
```

And now, we are going to see the decision tree and we can take this one, so we can print this decision, decision tree model and you can see that, it is nothing but an if-then-else and then that means all the internal nodes of a DC entry and finally the leap will be having the predictions. So, once the decision tree is built,

Refer Slide Time :( 31:23)

- Here we are applying a decision tree model to the test data, and obtain predictions.

```
In [*]: predictions = dtModel.transform(testData)
In []: predictions.select('label', 'labelIndexed', 'probability', 'prediction').show()
In []: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predicti
accuracy = evaluator.evaluate(predictions)
print("Test Error = %g" % (1.0 - accuracy))
```

now we can use this decision tree, to for the test data, to obtain the predictions. So, now we can use the test data and use the same decision tree model and now, we will perform the, the predictions. So, so the predictions are now, there and then we will evaluate this particular, the predictions using multi class, multi class classifier evaluator, we will evaluate these predictions, which are done by this one decision tree for its accuracy. So, what we will find here is that?

Refer Slide Time :( 32:08)

- Now we can evaluate the accuracy of model.
  - For this purpose, we use a MulticlassClassificationEvaluator with a metric named accuracy.
  - The testing error is 3%, the model is quite accurate.

```
In [26]: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predictionCol = "prediction")
accuracy = evaluator.evaluate(predictions)
print("Test Error = %g" % (1.0 - accuracy))

Test Error = 0.045197
```

That accuracy is very high and the error is very less. And so, the testing error is only thirty percent, so the model is quite accurate. So, with only three percent error, the model is quite accurate.

Refer Slide Time :( 32:28)

# Gradient Boosted Decision Trees

- First of all we import it, we create an object which will do this classification. Here you are specifying `labelCol = "labelIndexed"`, and `featuresCol = "features"`. Actually, it is not mandatory to specify feature column, because its default name is features. We can do it either with this argument or without this argument.

```
In [27]: from pyspark.ml.classification import GBTClassifier
In []: gbdt = GBTClassifier(labelCol = "labelIndexed", featuresCol = "features", maxIter
In []: gbdtModel = gbdt.fit(trainingData)
In []: gbdtModel.featureImportances
In []: gbdtModel.toDebugString
In []: predictions = gbdtModel.transform(testData)
predictions.select("label", "labelIndexed", "prediction").show()
In []: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predicti
accuracy = evaluator.evaluate(predictions)
print("Test Error = %g" % (1.0 - accuracy))
```

*Gradient Boosted Decision Tree Model*

And now, we are going to see the another method, which is called a, ‘Gradient Boosted Decision Trees’. So, first of all we import, the data and then create the object which will do the classification. And here, we are specifying the label column is a label indexed that we have seen in the previous example also and the features column as the features. Actually, it is not mandatory to specify the feature column, we can do it either using the assignment or without this argument, now the thing is, so after having done this, now we will fit this particular model and apply this particular gradient boosted, classifier using label indexed

the features that we have seen, now we will fit this particular model, onto the training dataset and we will get the model, gradient boosted a decision tree model. So, once we obtain the model, then we will see, its where then we will be using this particular model, we will now apply the test data on it this particular model and now we using a multi-class classification evaluator, we will evaluate its predictions.

Refer Slide Time :( 34:09)

# Gradient Boosted Decision Trees

- In this case, accuracy is a bit lower than the one of the single decision tree.

```
In [33]: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predictionCol = "prediction")
accuracy = evaluator.evaluate(predictions)
print("Test Error = %g" % (1.0 - accuracy))
.
.
.
Test Error = 0.0451977
```

So, let us see, so this we are going to do 100 iterations and default step is 0.1. So, after that, so we will see that here the accuracy, here the error is here the, the test error is quite. So, basically this has improved or the DC entry model.

Refer Slide Time : ( 34:33)

## Random Forest

- We are importing the classes which are required for evaluating random forest.
  - We are creating an object, and we are fitting this method.

```
In [34]: from pyspark.ml.classification import RandomForestClassifier, RandomForestClassifi
 .

In [35]: rfClassifier = RandomForestClassifier(labelCol = "labelIndexed", numTrees = 100)
 ✓

In []: rfModel = rfClassifier.fit(trainingData)
 ✓

In []: rfModel.featureImportances

In []: rfModel.toDebugString

In []: predictions = rfModel.transform(testData)
 ✓
predictions.select('label', 'labelIndexed', 'prediction').show()

In []: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predicti
 ✓
accuracy = evaluator.evaluate(predictions)

 print("Test Error = %g" % (1.0 - accuracy))
```

Now, we are going to see the random forest. And random forest again, the same dataset we will take and but here the classifier we are going to change as random forest classifier and we will fit the training data on random forest classifier and we will get a model, which is called, ‘Random Forest’, ‘Forest Model’. Now, using this particular model we will, now we will transform the test data we will apply the test it on this particular model and get the predictions. So, now after getting the predictions we are not evaluating these particular predictions, using multi-class classification evaluator and this will now, evaluate its prediction accuracies.

Refer Slide Time :( 35:28)

# Random Forest

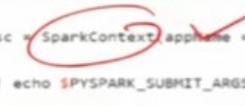
- We can see in this example, testing accuracy of random forest was the best. But with the other dataset, the situation may be quite different.
  - And as a general rule, the bigger your dataset is, the more features it has, then the quality of complex algorithms like gradient boosted decision trees or random forest will be better.

And this we will now, be able to see that, the test error is very less that is, than the previous decision trees. Hence, this we can see that in this example, the testing accuracy of the random forest was the best with the other data the situation may be quite different. In general case, the bigger your dataset is more features, it has the quality of complex algorithm like gradient boosted or the forest will be much better.

Refer Slide Time :( 36:02)

## Cross Validation

- Cross-validation helps to assess the quality of a machine learning model and to find the best model among a family of models.
- First of all, we start SparkContext.

```
In [1]: from pyspark import SparkContext
from pyspark.sql import SparkSession
*
In [2]: sc = SparkContext(appName = "module3_week4")
 
In [3]: ! echo $PYSPARK_SUBMIT_ARGS
--deploy-mode client --master local[2] --executor-memory 512m --driver-memory 5
12m --executor-cores 1 --num-executors 2 --conf spark.driver.maxResultSize=256
pyspark-shell
In [4]: spark = SparkSession.Builder().getOrCreate() # required for dataframes
```

Now, let us see the cross-validation using spark ml. So, class validation helps to assess, the quality of machine learning model and to find the best model, among the family of the models. First of all, we have to create the spark context that is, shown over here and then we will, do the spark session.

Refer Slide Time :( 36:27)

# Cross Validation

We use the same dataset which we use for evaluating different decision trees.

```
In []: !wget https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin.arff
In []: from pyspark.ml.linalg import Vectors
 from pyspark.ml.feature import StringIndexer
In []: # Load a text file and convert each line to a Row.
 data = []
 with open("wdbc.data") as infile:
 for line in infile:
 tokens = line.rstrip("\n").split(",")
 y = tokens[0]
 features = Vectors.dense([float(x) for x in tokens[2:]])
 data.append((y, features))
In []: inputDF = spark.createDataFrame(data, ["label", "Features"])
In []: stringIndexer = StringIndexer(inputCol = "label", outputCol = "labelIndexed")
 si_model = stringIndexer.fit(inputDF)
```

Then we'll build the spark session, will build the spark session, which will create the data frames and then we will use the dataset, read the dataset, load the data file and now, we will create the data frames, DF as the input data frame.

Refer Slide Time :( 36:50)

# Cross Validation

- What steps are required for doing cross-validation with this dataset?  
For example, we want to select the best parameters of a single decision tree. We create an object decision tree then we should create a pipeline.

```
Cross-Validation
In []: from pyspark.ml.classification import DecisionTreeClassifier
In []: decisionTree = DecisionTreeClassifier(labelCol = "labelIndexed")
In []:
In []: from pyspark.ml import Pipeline
In []: pipeline = Pipeline(stages = [decisionTree])
In []:
In []: from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
In []: paramGrid = ParamGridBuilder()\n .addGrid(decisionTree.maxDepth, [1, 2, 4, 5, 6, 7, 8])\n .build()
```

And now, we will apply we after doing various transformations. Now, we will input data we can see it is already indexed now we have to do the cross validation. So, what steps are required for doing cross validation with the dataset? So, for example, we want to select the best parameters of a single decision tree. And we create the object decision tree then we should create the pipeline. So, here we are going to create the pipeline and in this pipeline stage we are only using the decision tree based pipeline.

Refer Slide Time :( 37:25)

## Cross Validation

- Pipeline, in general, may contain many stages including feature pre-processing, string indexing, and machine learning, and so on.
- But in this case, pipeline contains only one step, this training of decision tree.

### Cross-Validation

```
In []: from pyspark.ml.classification import DecisionTreeClassifier
In []: decisionTree = DecisionTreeClassifier(labelCol = "labelIndexed")
In []:
In []: from pyspark.ml import Pipeline
In []: pipeline = Pipeline(stages = [decisionTree])
In []:
In []: from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
In []: paramGrid = ParamGridBuilder()
 .addGrid(decisionTree.maxDepth, [1, 2, 4, 5, 6, 7, 8])\n .build()
```

So, then we will apply this cross validation using parameter builder and we have to also see the cross validation.

Refer Slide Time :( 37:39)

## Cross Validation

- Then we import their cross-validator and ParamGridBuilder class and you are creating a ParamGridBuilder class.
- For example, we want to select the best maximum depths of a decision tree in the range from 1-8.
- We have now the ParamGridBuilder class, we create an evaluator so we want to select the model which has the best accuracy among others.

```
In [15]: from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

In [16]: paramGrid = ParamGridBuilder()
 .addGrid(decisionTree.maxDepth, [1, 2, 4, 5, 6, 7, 8])
 .build()

In [17]: paramGrid = ParamGridBuilder()
 # addGrid(decisionTree.maxDepth, [1, 2, 4, 5, 6, 7, 8])
 # addGrid(decisionTree.minInstancesPerNode, [1, 2, 4, 5, 6, 7, 8])
 .build()

In [18]: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predictionCol = "prediction")
crossval = CrossValidator(estimator = pipeline,
 estimatorParamMaps = paramGrid,
 evaluator = evaluator,
 numFolds = 10)

In [19]: cvModel = crossval.fit(inputDF2)
```

So, then we import their class validator and parameter great builder class and you are now creating the parameter grid builder class. For example, you want to select the best maximum depth of the decision tree in the range from 1 to 8. And we don't have the parameter build grid builder class and we create the evaluator, so we want to select the model that which has the best accuracy among others. And using this, all these parameters whatever we have said now we are going to evaluate using multi-class classification evaluator.

Refer Slide Time :( 38:22)

## Cross Validation

- We create an evaluator so we want to select the model which has the best accuracy among others.
- We create a cross-validator class and pass a pipeline into this class, a ParamGrid and evaluator.

```
In [1]: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predictionCol = "prediction")
crossval = CrossValidator(estimator = pipeline,
 estimatorParamMaps = paramGrid,
 evaluator = evaluator,
 numFolds = 10)

In [2]: cvModel = crossval.fit(inputDF2)

In [3]: cvModel.avgMetrics

In [4]: print cvModel.bestModel.stages[0]

In [5]: cvModel.transform(....)
```

And now, we will see the evaluator will now, evaluate the cross validator. And cross validator will fit it to the input function and it will select the best model into the different stages. So, we create the evil water so we want to select the model which has the best accuracy among others so we create a cross validator class and pass a pipeline into this class parameter great and the evaluator.

Refer Slide Time :( 38:55)

## Cross Validation

- And finally, we select the number of folds and the number of folds should not be less than 5 or 10.

```
In []: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predictionCol = "predictedLabel", metricName = "accuracy")
crossval = CrossValidator(estimator = pipeline,
 estimatorParamMaps = paramGrid,
 evaluator = evaluator,
 numFolds = 10)

In []: cvModel = crossval.fit(inputDF2)

In []: cvModel.avgMetrics

In []: print cvModel.bestModel.stages[0]

In []: cvModel.transform(...)
```

And finally, we select the number of folds and the number of holes should be not less than five or ten. And the number of folds we have selected and after that,

Refer Slide Time :( 39:07)

## Cross Validation

- We create cvModel and it takes some time because Spark needs to make training and evaluating the quality 10 times.

```
In [17]: crossval = CrossValidator(estimator = pipeline,
 estimatorParamMaps = paramGrid,
 evaluator = evaluator,
 numFolds = 10)

In [18]: cvModel = crossval.fit(inputDF2)

In [19]: cvModel.avgMetrics
Out[19]: [0.8924563921120648,
 0.9203744192767783,
 0.9418223975919139,
 0.9419915318207598,
 0.9457320946210345,
 0.938926791703486,
 0.9361037147804891]

In []: print cvModel.bestModel.stages[0]

In []: cvModel.transform(...)
```

we create the CV model and takes some time because spark need to make the training and evaluation, the qualified the ten times.

Refer Slide Time :( 39:21)

## Cross Validation

- You can see the average accuracy, amount of folds, for each failure of decision tree depths.

```
In []:

In [15]: from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
 from pyspark.ml.evaluation import MulticlassClassificationEvaluator

In [16]: paramGrid = ParamGridBuilder()\n .addGrid(decisionTree.maxDepth, [1, 2, 4, 5, 6, 7, 8])\n .build()

In []: #paramGrid = ParamGridBuilder()\n# .addGrid(decisionTree.maxDepth, [1, 2, 4, 5, 6, 7, 8])\n# .addGrid(decisionTree.minInstancesPerNode, [1, 2, 4, 5, 6, 7, 8])\n# .build()

In [17]: evaluator = MulticlassClassificationEvaluator(labelCol = "labelIndexed", predicti
```

Now, we can see that the average accuracy amount of fold for each of the failure of the three depths.

Refer Slide Time :( 39:30)

## Cross Validation

- The first stage of our pipeline was a decision tree and you can get the best model and the best model has depth 6 and it has 47 nodes.

```
crossval = CrossValidator(estimator = pipeline,
 estimatorParamMaps = paramGrid,
 evaluator = evaluator,
 numFolds = 10)

In [18]: cvModel = crossval.fit(inputDF2)

In [19]: cvModel.avgMetrics
Out[19]: [0.8924563921120648,
 0.9203744192767783,
 0.9418223975919139,
 0.9419915318207598,
 0.9457320946210345,
 0.938026791703486,
 0.9361037147804091]

In [20]: print cvModel.bestModel.stages[0]
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_4bce0658f839baff82f
#) of depth 6 with 47 nodes

In []: cvModel.transform(...)
```

Now, we can see and the first stage of our pipeline was the decision tree, so you can get the best model and the best model has the depth six and it has 47 nodes in this case.

Refer Slide Time :( 39:42)

## Cross Validation

- Then we can view this model to make predictions at any other dataset. In ParamGridBuilder, we can use several parameters. For example, maximum depths and some other parameters of a decision tree, for example, minInstancesPerNode and select some other grid here, but in this simple example, we did not do it for the simplicity. And if we evaluate only one parameter, the training is much faster.

```
In [13]: from pyspark.ml import Pipeline
In [14]: pipeline = Pipeline(stages = [decisionTree])
In []:
In [15]: from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
In [16]: paramGrid = ParamGridBuilder()\
 .addGrid(decisionTree.maxDepth, [1, 2, 4, 5, 6, 7, 8])\
 .build()
In []: #paramGrid = ParamGridBuilder()\
.addGrid(decisionTree.maxDepth, [1, 2, 4, 5, 6, 7, 8])\
.addGrid(decisionTree.minInstancesPerNode, [1, 2, 4, 5, 6, 7, 8])\
.build()
```

Then we can view this model, to make the prediction at any other datasets. So, in parameter great builder we can use several parameters. For example, maximum depth and some other parameter of the decision tree, for example, minimum instances per node and select some other grid here. But, in this example we did not do it due to the simplicity and if we evaluate only one parameter the training is much faster.

Refer Slide Time :( 40:09)

## Conclusion

- In this lecture, we have discussed the concepts of Random Forest, Gradient Boosted Decision Trees and a Case Study with Spark ML Programming, Decision Trees and Ensembles.

So, conclusion in this lecture we have discussed the concept of random forests, gradient boosted decision trees and we have also covered a case study using a Spark ML programming, on decision trees and other learning tree ensembles. Thank you.

## **Lecture - 30**

### **Parameter Servers**

Parameter servers.

Refer Slide Time :( 0:17)

## Preface

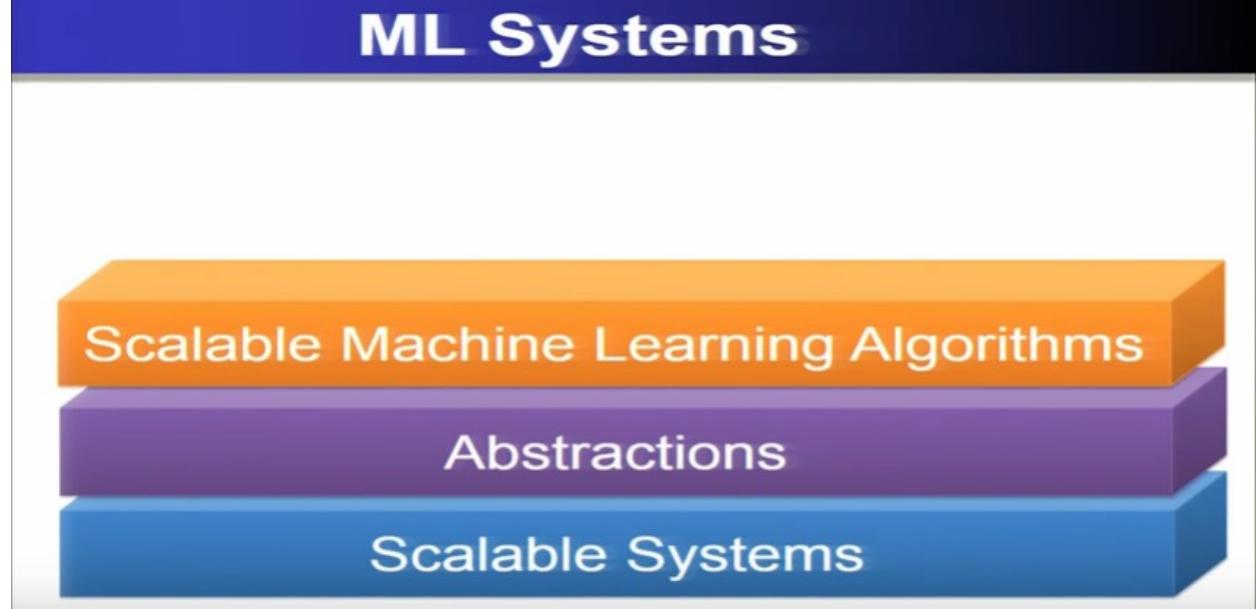
### Content of this Lecture:

- In this lecture, we will discuss the '**Parameters Servers**', also discuss its Stale Synchronous Parallel Model.

Preface, content of this lecture: In this lecture, we will discuss the parameter servers. And also, discuss the stale synchronous parallel model.

Refer Slide Time :( 0:27)

## ML Systems



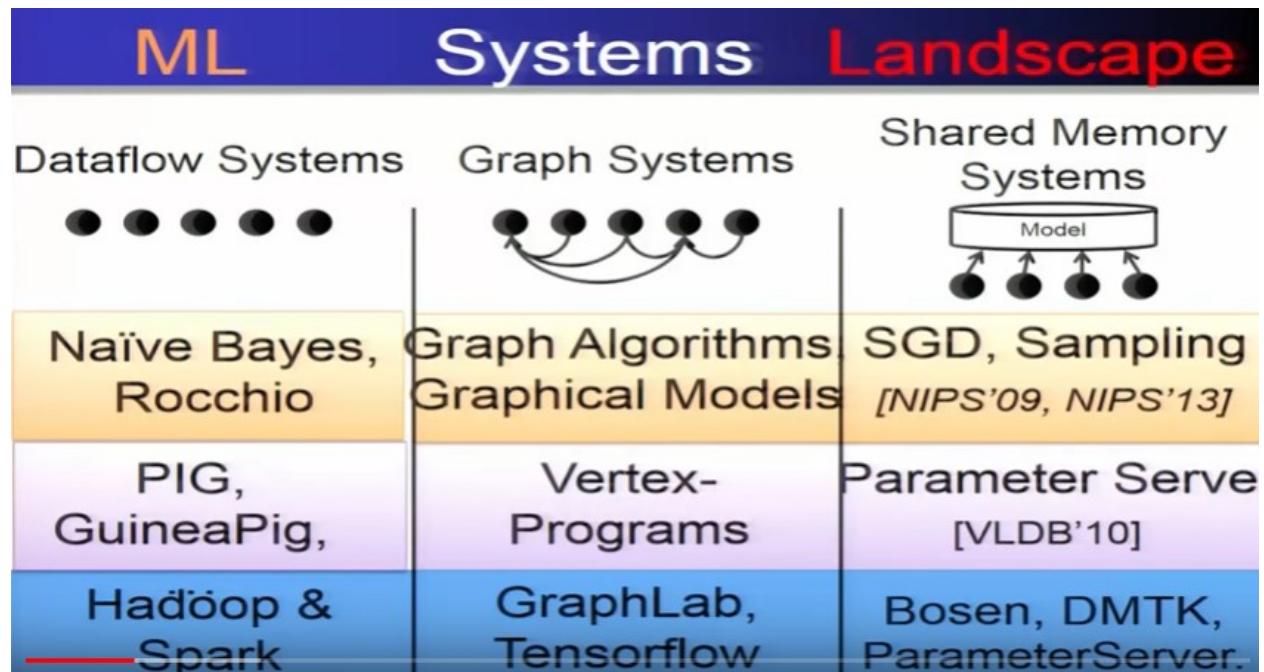
Scalable Machine Learning Algorithms

Abstractions

Scalable Systems

In machine learning systems, especially the scalable machine learning algorithms that we are seeing for, big data computing, requires the abstractions, for the scalable, building scalable systems.

Refer Slide Time :( 0:48)



And here, we can see that, there are three different sections, the machine learning, systems and landscape. We have to see that there are data flow systems which are, built over Hadoop and stack platforms, using the graph system algorithms and shared memory systems. And these machine learning algorithms, which are nothing but supported as the Dataflow systems, such as Naive Bayes and requires an abstraction from the underlying systems like, Hadoop and Spark and for graph systems it requires abstractions from the GraphLab and TensorFlow. And similarly for Shared Memory Systems and these are Bosen and DMTK also requires the parameter servers.

Refer Slide Time :( 1:51)

# ML

# Systems

# Landscape

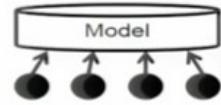
## Dataflow Systems

## Graph Systems

Simple case: Parameters of the ML system are stored in a **distributed hash table** that is accessible thru the **network**

Param Servers used in Google, Yahoo, ....  
Academic work by Smola, Xing, ...

## Shared Memory Systems



[NIPS'09, NIPS'13]

## Parameter Server

[VLDB'10]

So, what we will see here is that, in a simple case, the parameters of machine learning are stored in a distributed hash table that is accessible through the network, which is there in the shared memory systems. So, parameter servers which are used in Google, Yahoo and also is used as an academic work by Smola and Xing.

Refer Slide Time :( 2:20)

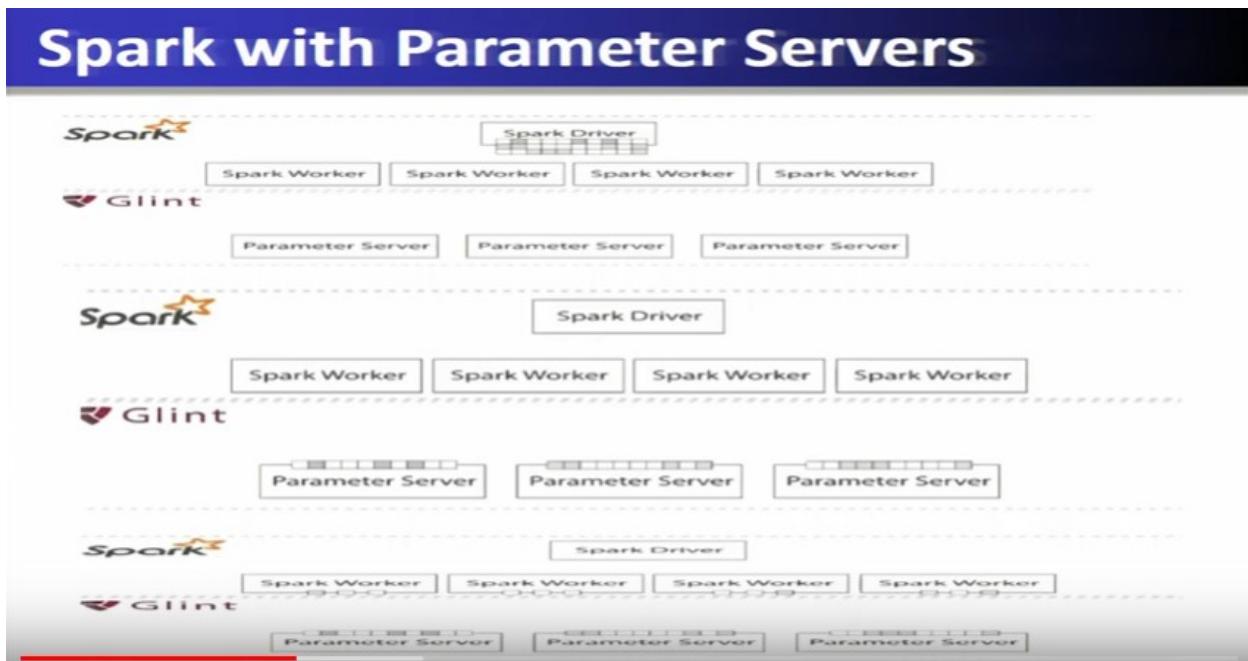
## Parameter Server

- A machine learning framework
- Distributes a model over multiple machines
- Offers two operations:
  - **Pull:** query parts of the model
  - **Push:** update parts of the model
- Machine learning update equation  
$$w_i \leftarrow w_i + \Delta$$
- (Stochastic) gradient descent
- Collapsed Gibbs sampling for topic modeling
- Aggregate **push** updates via addition (+)

And now, we have to see the parameter server and how it is going to be used as the framework for machine learning. So, to build the scalable machine learning systems, so skill machine learning framework requires, the use of parameter server and it distributes, a model over multiple machines and

offers two different operations, one is called, ‘Pull’, that is the query part of the model. And the, ‘Push’, means, update part of the model. So, in any machine learning update equation, which is shown here a,  $w_i \leftarrow w_i + \Delta$  is to be handled into the model, by push that is in the form of plus operation. So, in a stochastic gradient descent and all in a collapsed Gibbs sampling for a topic modeling, this aggregate, that is the push updates is done via the addition operation. So, whenever this addition operation is there, so these parameters are to be pushed into the model and that is handled using the parameter server.

Refer Slide Time :( 3:41)

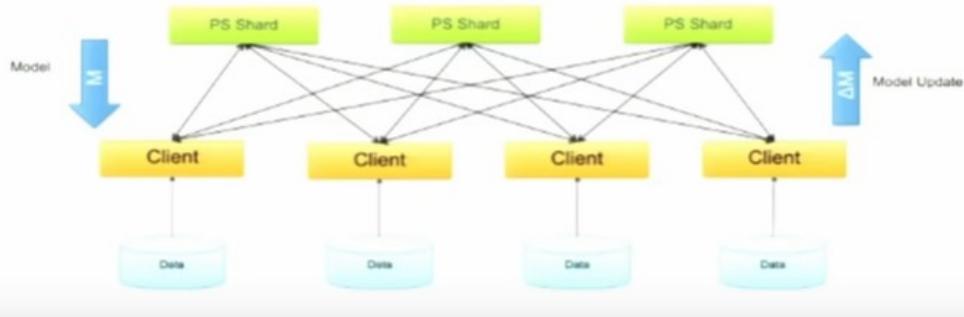


Now, this spark machine learning is done with the parameter servers. So, here in this particular scenario, we see that spark driver and spark workers, they together work and support this parameter server notion. Similarly, this kind of parameter server notion is also supported in the glint. We can see there in spark, there is a parameter server. And in a spark there is a spark driver and different spark workers are there. So, we can see here in this particular example that, these parameter servers are either maintained in the glint, as the queues or as the distributed queue and in the spark as a Driver worker combination.

Refer Slide Time :( 4:38)

## Parameter Server (PS)

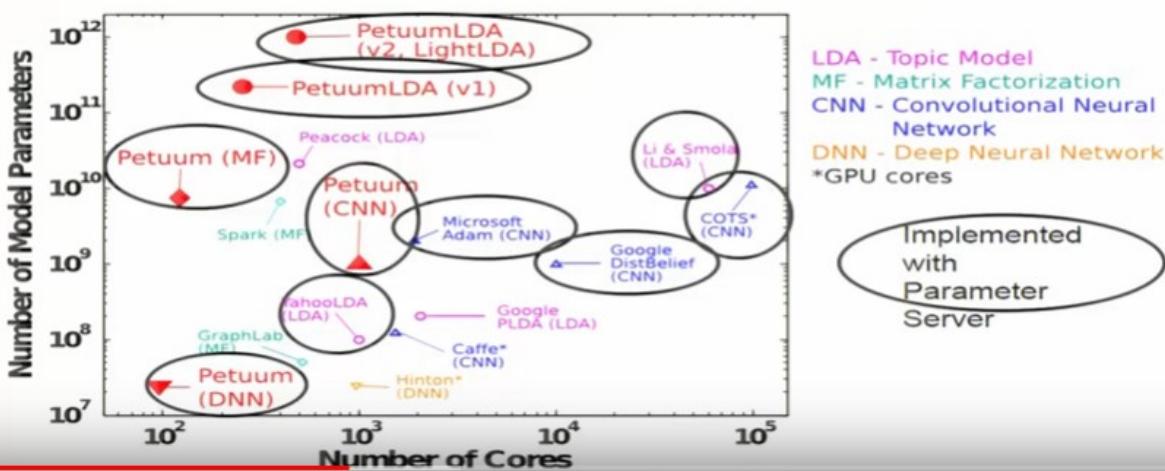
- Training state stored in PS shards, asynchronous updates



So, parameter server is used in the training phase also. So in the training state it is stored in the parameter shard and asynchronous updates are being handled. So, this is the example, figure which shows that, the models are stored as a part of parameters shards, parameter server charts and whenever, there is an update. So, it has to be updated at all the shards which is managing the parameter server. So, the parameter servers are maintained in this big data scenario, that is through the clusters, that we are going to understand here in this part of the discussion.

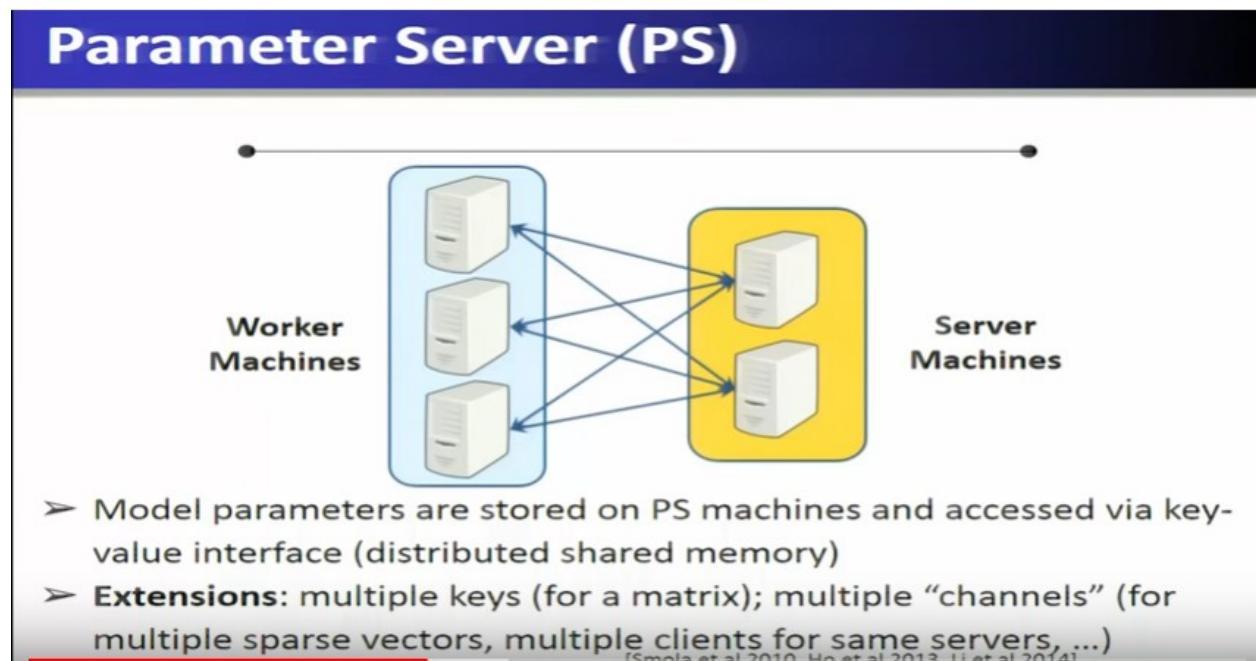
Refer Slide Time :( 5:21)

## Parameter Servers Are Flexible



So, parameter servers are flexible. And here, the number of model parameters which is being supported is quite large and number of cores, which are there which supports this parameter server is shown, for different machine learning landscape. For example, topic modeling and matrix factorization and CNN that is convolutional neural networks and DNN-deep neural networks, all requires flexible parameter servers and huge amount of parameters are complex parameter servers are being managed, efficiently therefore it is shown here in this example figures.

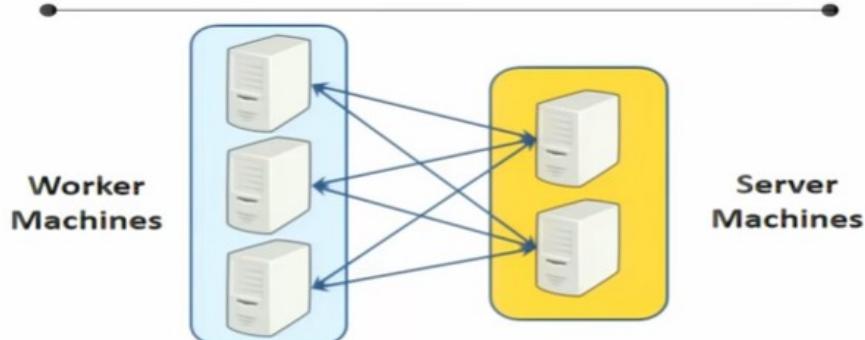
Refer Slide Time :( 6:07)



So, let us understand about the parameter servers. So, parameter server now comprises in its spark, comprises of the server machines and the worker machines and it models the parameter servers, as are stored in the parameter server machines and accessed via the key/value, interface that is in the distributed, as per the distributed shared memory. So, extensions of it are multiple keys are available for matrix operations and multiple channels for multiple sparse vectors and multiple clients for the same servers.

Refer Slide Time :( 6:47)

## Parameter Server (PS)

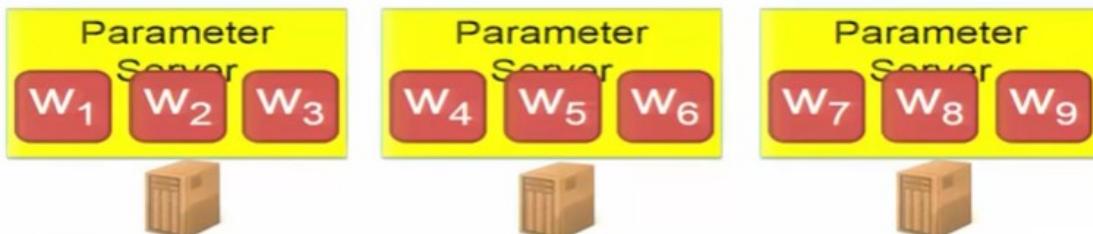


- **Extensions:** push/pull interface to send/receive most recent copy of (subset of) parameters, blocking is **optional**
- **Extension:** can block until push/pulls with clock  $< (t - \tau)$  complete

So, the extensions which are available here in the parameter server is in the form of push-pull interface, to send or receive most recent copy of the subset of the parameters and the blocking is optional. And the extension, can block until the push-pulls with a clock which is bounded by  $t$  minus rho is completed.

Refer Slide Time :( 7:14)

## Data parallel learning with PS



1. Different parts of the **model** on different servers.
2. Workers retrieve the part needed **as needed**



Split Data Across Machines

So, let us see the data parallel learning with the parameter server. And here, the parameter servers are shown in this particular picture that they are being exchanged, through a different part of the model is available on different servers and the workers, they retrieve the part of the model as it is needed.

Refer Slide Time :( 7:39)

## Abstraction used for

- Key-Value API for workers:

1.  $\text{get}(\text{key}) \rightarrow \text{value}$

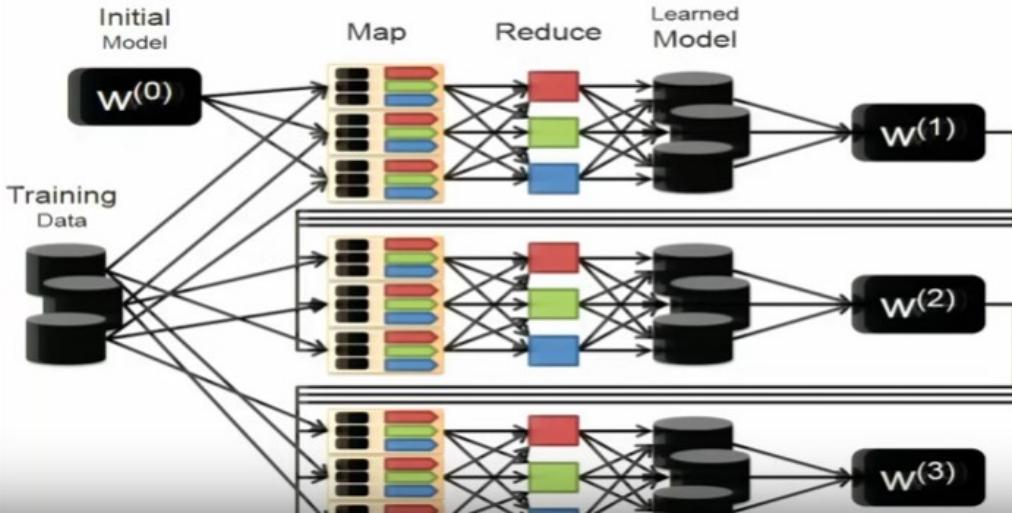
2.  $\text{add}(\text{key}, \text{delta})$

$\text{Mode} \leftarrow \text{Mode} \oplus \delta_i$

So therefore, the abstractions are provided which are internally implemented as the data parallel operations in the parameter server, such as the key/value API is for the workers that is they can get the key. And also, that means they may get the model values and by get key operation. Similarly, I had key with the value  $\delta$ , to be updated in the parameter server and this particular addition can be performed here using add operation.

Refer Slide Time :( 8:18)

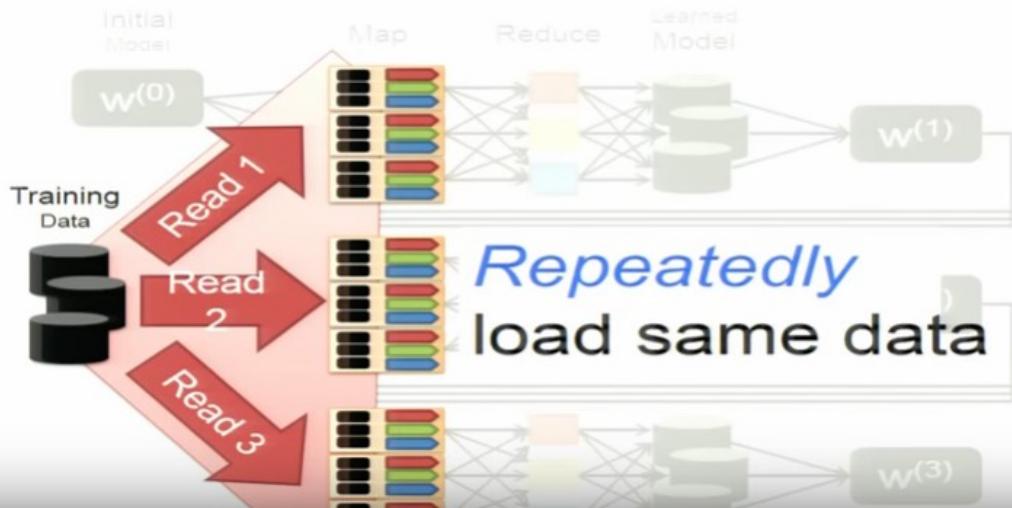
# Iteration in Map-Reduce (IPM)



So, let us see how the iterations in a Map Reduce is supported, using the parameter server. So, this particular iterations in the Map Reduce is often required, to implement the scalable machine learning and this requires the training data to be provided

Refer Slide Time :( 8:43)

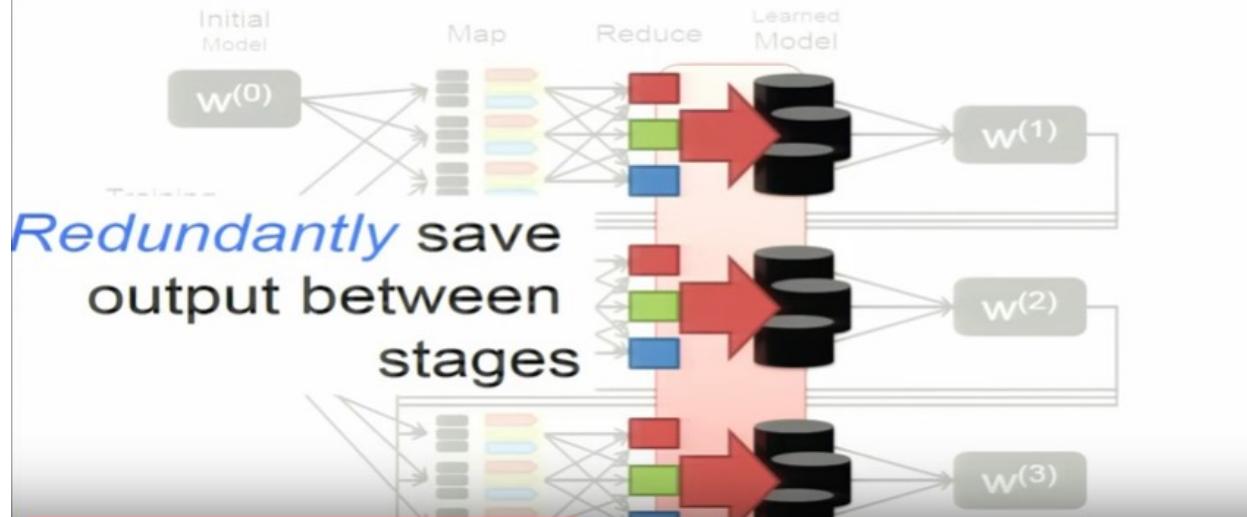
## Cost of Iteration in Map-Reduce



to the different map phase and which are to be read, into different splits or in the form of the shots.

Refer Slide Time :( 8:55)

## Cost of Iteration in Map-Reduce



And then, they will redundantly save the output between these stages in the Map Reduce function and therefore increase the cost of this iterations of the Map Reduce.

Refer Slide Time :( 9:10)

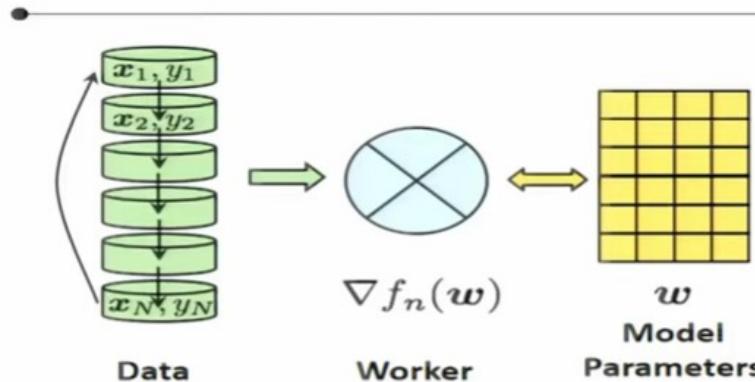
## Parameter Servers

### Stale Synchronous Parallel Model

And therefore the parameter servers are very much needed, to make this machine learning, scalable machine learning every support. So, this particular parameter servers requires, still synchronous parallel model we will understand about them. So here, we have to see these model parameters are stored on the, parameter server machines and, and accessed via the key/value, interface using distributed memory.

Refer Slide Time :( 9:42)

## Iterative ML Algorithms



➤ Topic Model, matrix factorization, SVM, Deep Neural Network...

So, iterative machine learning, if we see the workload which comprises of these parameter transformations that is the data is transformed and by the worker nodes and in between they are accessing the model parameters. And for example, in the topic modeling and matrix factorization, support vector machine and deep neural networks, they all are iterative machine learning algorithms, wherein the workers require these parameters to be updated, parameters mean model parameters are to be, updated. So therefore, a parameter server has to maintain this model parameters. And all the workers will now communicate with the model parameters server and these values are, either read or that means, they will either get or they will put these values on this particular parameter server.

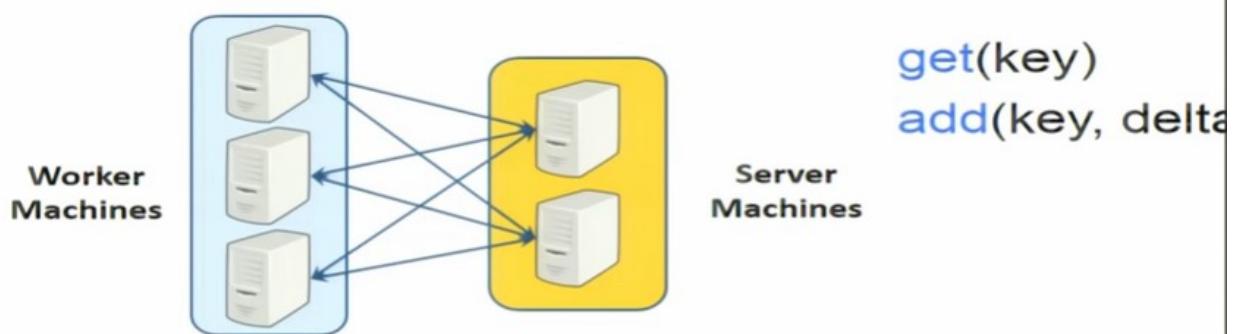
Refer Slide Time :( 10:52)

|                         |                                 |                                             |
|-------------------------|---------------------------------|---------------------------------------------|
|                         | Map-Reduce                      | Parameter Server                            |
| Data Model              | Independent Records             | Independent Data                            |
| Programming Abstraction | Map & Reduce                    | Key-Value Store (Distributed Shared Memory) |
| Execution Semantics     | Bulk Synchronous Parallel (BSP) | ?                                           |

So, here we can see that in typical Map Reduce model, where the data model is the independent records and the programming abstraction is in the form of Map Reduce and the execution semantics, which is used as bulk synchronous parallel, in the Map Reduce to support the map, to support the iterative machine learning algorithms. Now, if we compare this with the parameter server here, the data model in the parameter server is also independent records and the programming abstraction is not the Map Reduce, but it is simply the key value store that is in the form of a distributed shared memory to be provided. Now, what is the execution semantics is it, the bulk synchronous parallel or some other synchronous that we are going to see.

Refer Slide Time :( 11:42)

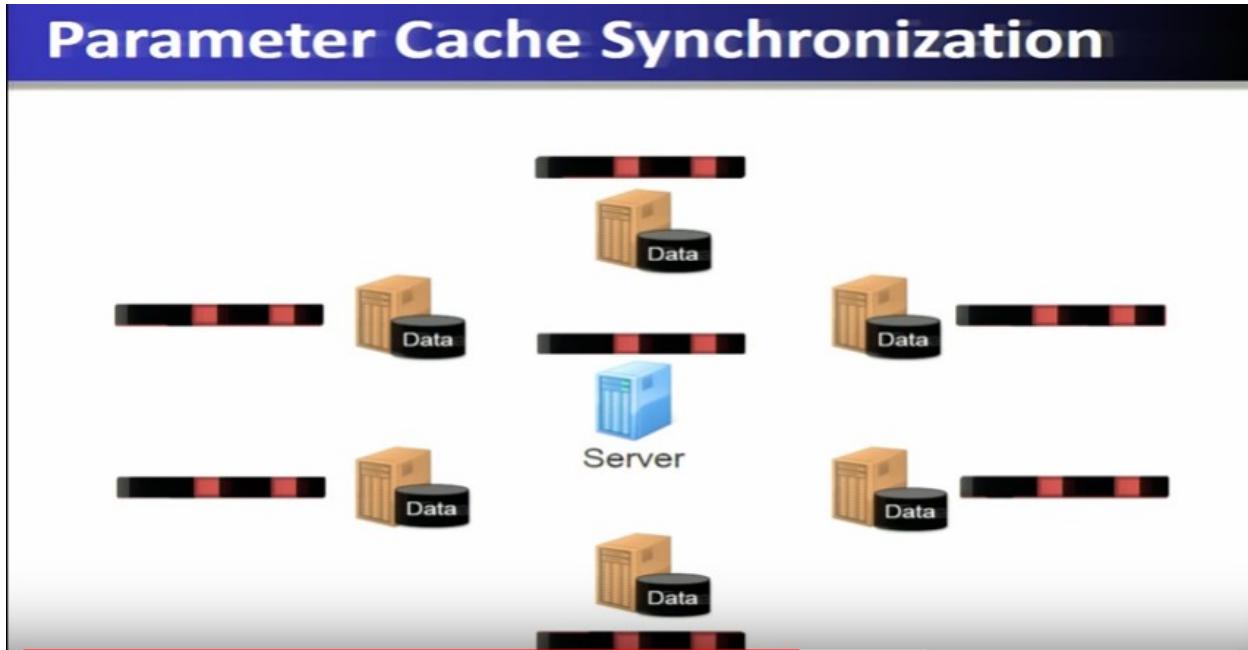
## The Problem: Networks Are Slow!



- Network is slow compared to local memory access
- We want to explore options for handling this....

Now, the problem here is that, the networks are slow. That means, when these parameter servers which are maintaining the parameters, of the model and using gate and adding the keys are the operations, which are to be performed on the network, there becomes quite slow. As the network is slow compared to the local memory axis. So, we want to explore the options for handling, this with the help of the caching.

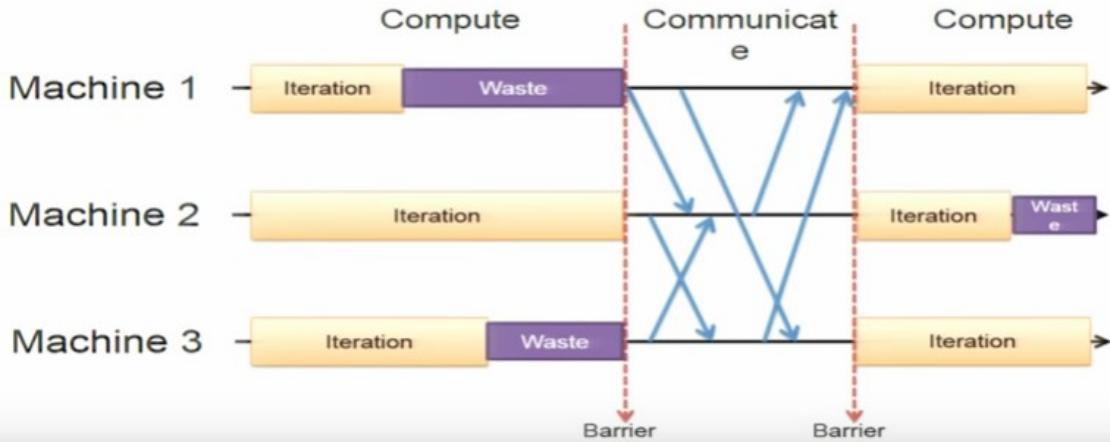
Refer Slide Time :( 12:09)



Now, the caching will introduce a problem that is to be handled using, the cache synchronization and here we can see that, the cache which is maintained in and whenever sparse changes, to the model is happening and repeat is done through the cache. Then, these parameter servers requires this particular cache to be synchronized that is called, 'Cache Synchronization'. So, having implemented the cache synchronization, in the parameter servers the problem of accessing the parameter server, across network also can become efficient if the access is done through the cache.

Refer Slide Time :( 12:57)

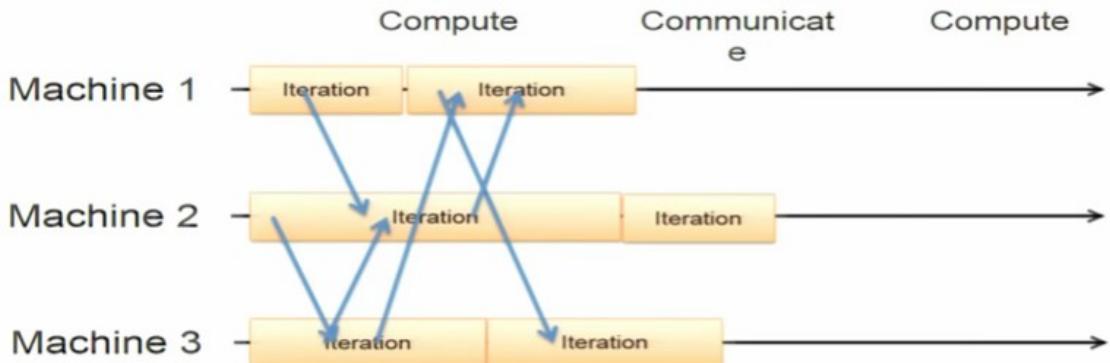
## Solution 2: Asynchronous



The second solution, to implement the efficient access of the parameter server on the network is about the asynchronous operations. So here, the machine one, machine two, machine three, they will perform another computation, they will iterate and now they have to wait, till others, other iteration they complete their iterations. And then, they will interchange their communication and then, the computation can go on.

Refer Slide Time :( 13:32)

## Solution 2: Asynchronous

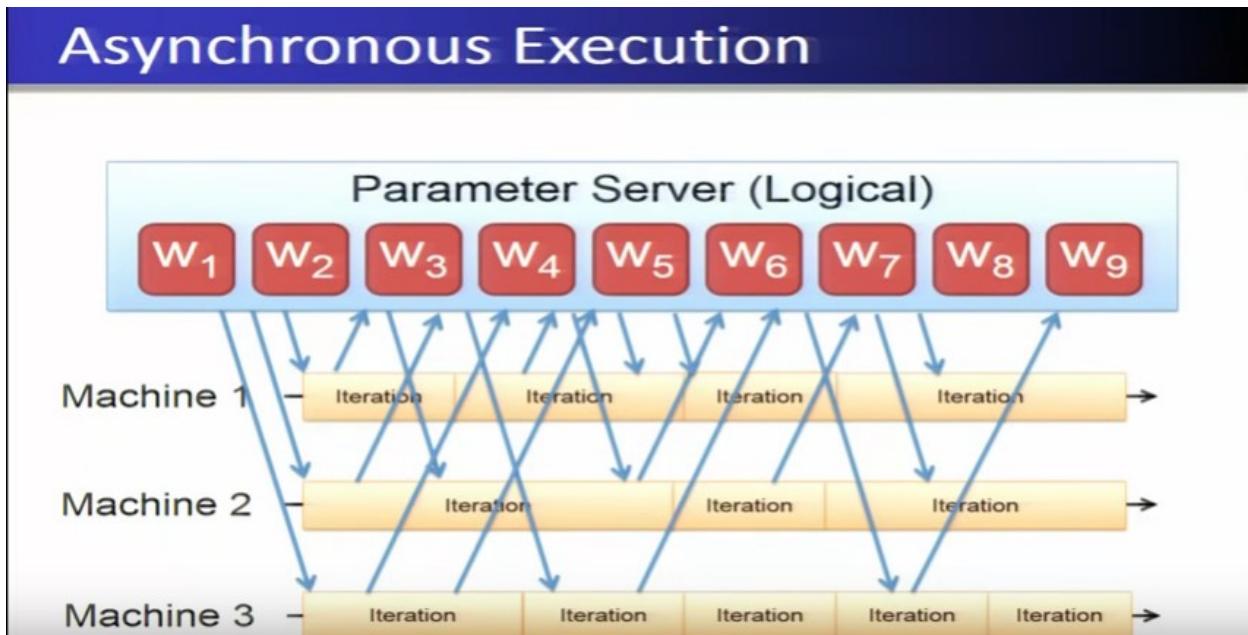


Enable more frequent coordination on parameter values

So, the in every iteration we see that, there will be a barrier, after the machine finishes the iteration and start the communication and before, all the communication completes, so that the next iteration will begin and these are the barriers and if we collapse these barriers, now we can introduce a much efficient way, of

implementing this a synchronous, asynchronous here in this case. So, enable the more frequent coordination on, the parameter servers.

Refer Slide Time :( 14:15)



So, if we support this asynchronous execution, environment with the parameter servers, then the, the scenario will look like this that, the efficient, the, the operations in the parameter server will become more efficient. For example, here the machine one, machine two, machine three, they access different parameters  $W_1$  and  $W_2$  and  $W_3$ , from a different parameter server, which are shown logically  $W_1$ ,  $W_2$ ,  $W_3$ , they are different parameter servers. And they keep on, accessing by different machines and they internally they communicate, the results back to this particular server.

Refer Slide Time :( 14:58)

# Asynchronous Execution

- Problem:
- Async lacks theoretical guarantee as distributed environment can have arbitrary delays from network & stragglers
- But....

So, this asynchronous operations are, if it is there then, let us see the problems, in this asynchronous execution. So, asynchronous execution lacks the theoretical guarantees, as the distributive environment can have, arbitrary delays from the network and stragglers.

Refer Slide Time :( 15:16)

**RECAP**

$f$  is loss function,  $x$  is parameters

1. Take a gradient step:  $x' = x_t - \eta_t g_t$
2. If you've restricted the parameters to a subspace  $X$  (e.g., must be positive, ...)  
find the closest thing in  $X$  to  $x'$ :  $x_{t+1} = \operatorname{argmin}_X \operatorname{dist}(x - x')$
3. But.... you might be using a "stale"  $g$  (from  $\tau$  steps ago)

---

**Algorithm 1** Delayed Stochastic Gradient Descent

**Input:** Feasible space  $X \subseteq \mathbb{R}^n$ , annealing schedule  $\eta_t$ .  
**Initialization:** set  $x_1, \dots, x_\tau = 0$  and compute  $g_1, \dots, g_\tau$ .  
**for**  $t = \tau + 1$  **to**  $T + \tau$  **do**

- Obtain  $f_t$  and incur loss  $f_t(x_t)$
- Compute  $g_t := \nabla f_t(x_t)$
- Update  $x_{t+1} = \operatorname{argmin}_{x \in X} \|x - (x_t - \eta_t g_{t-\tau})\|$  (Gradient Step and Projection)

**end for**

delay  $\tau \in \mathbb{N}$   
using  $g_t = \nabla f_t(x_t)$ .

So therefore, what we see here is that, if we see a particular implementation of the scalable machine learning algorithm. Let us say that, we say the delayed stochastic gradient descent algorithm, here we will see that,  $F$  is a function loss and  $X$  is a parameter, which is being computed in this algorithm and which required to be updated. And this particular updation.

Refer Slide Time :( 15:51)

| Map-Reduce vs. Parameter Server |                                             |
|---------------------------------|---------------------------------------------|
| Data Model                      | Independent Records                         |
| Programming Abstraction         | Map & Reduce                                |
| Execution Semantics             | Bulk Synchronous Parallel (BSP)             |
|                                 | Key-Value Store (Distributed Shared Memory) |
|                                 | Bounded Asynchronous                        |

Let us see what problem will create, during the using asynchronous communication. And therefore, we require to make about execution semantics and decided so that, this particular a synchronous, operation should not be a bounded, unbounded. So, it has to be a bounded synchronous operation, which has to be supported.

Refer Slide Time :( 16:14)

## Parameter Server

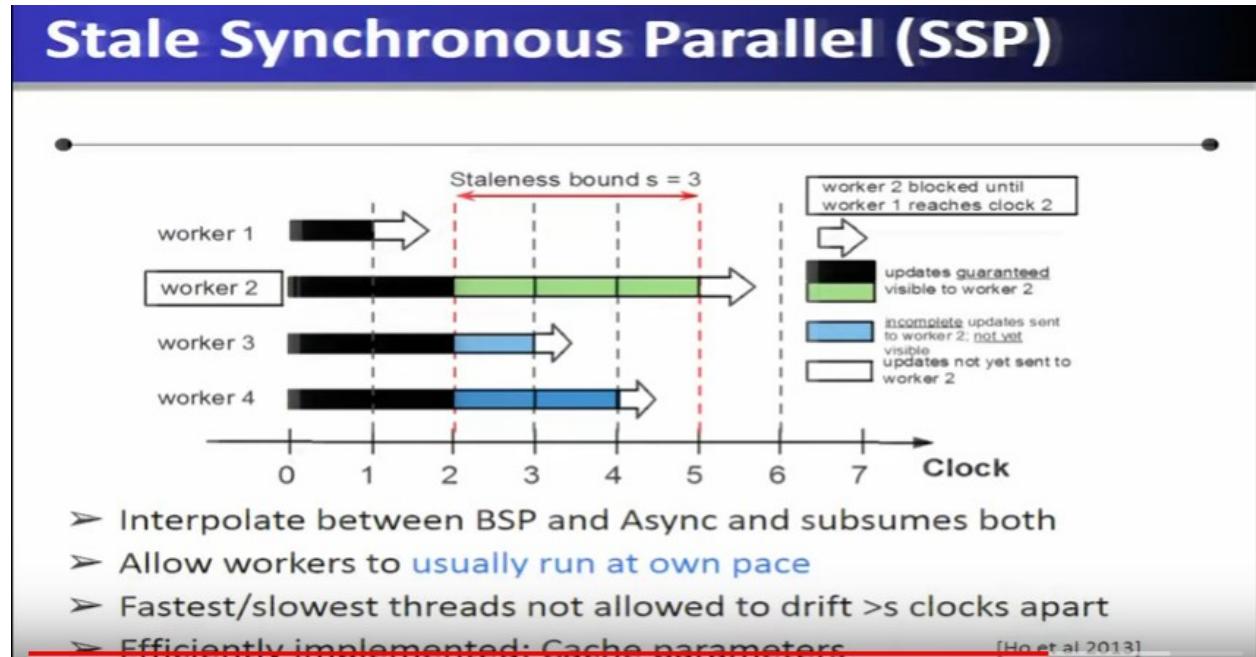
Stale synchronous parallel (SSP):

- Global clock time  $t$
- Parameters workers “get” can be out of date
- but can’t be older than  $t-\tau$
- $\tau$  controls “staleness”
- aka stale synchronous parallel (SSP)

Bounded  
Asynchronous

So, the parameter servers, will use the model which is called, ‘Stale Synchronous Parallel’. Model where in the global time is let us say,  $t$  and the parameter workers, will get and can be out of the date, but cannot be older than  $t - \tau$ . So,  $\tau$  is the tolerance, which is introduced and therefore, it is called, ‘Bounded Synchronous’. So, tau controls the Staleness, also called as, ‘Stale Synchronous Parallel’. So, stale synchronous parallel, execution semantics. We are going to see here,

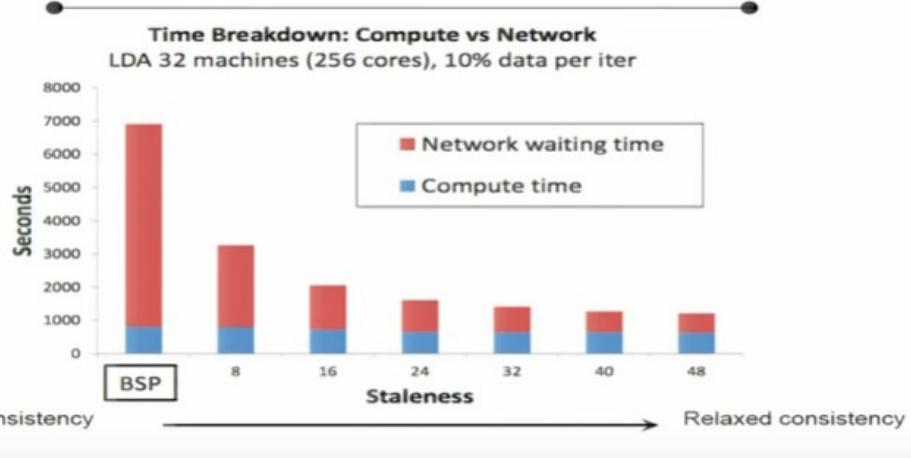
Refer Slide Time :( 16:55)



which is being supported in the parameter servers. So, in the stale synchronous parallel, we see here is that, they will these are workers 1 ,2, 3 & 4. So, these workers now, they will be changing the parameters, as synchronously and the stillness is also, a bounded by  $s$  is equal to 3 that means, now the here we can see here is that, the worker 2 will be blocked until, worker 1 reaches the clock of, clock of 2. And so that means, the updates will be guaranteed and this is black means, updates are guaranteed and green means it is visible to the, worker 2 and blue will indicate that it is in complete. And incomplete updates sent to the worker 2 and not yet visible, updates not yet sent to the worker too. So therefore, this particular slots which is divided and stainless bounds  $s = 3$  which is being calculated. So, this way the interoperate, interpolate between BSP and the as synchronous modes and they subsumes both. Allow the workers to usually run at their own pace. And the fastest, slowest threats not allowed to drift, the clock UI that means more than,  $s$  clocks apart. So, efficiently it implements the cache parameters, in this particular manner.

Refer Slide Time :( 18:42)

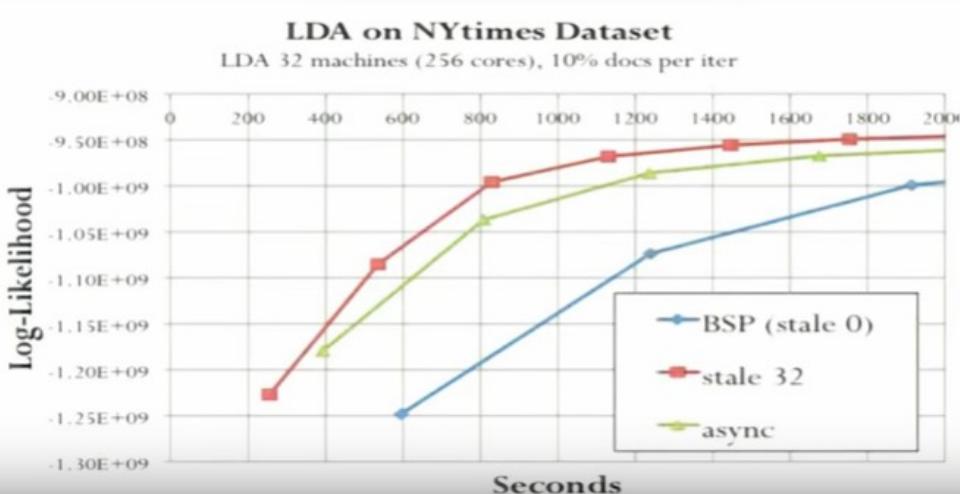
# Consistency Matters



So, consistency here matters. Because, staleness has to be bounded by BSP, which is a strong consistency but we relaxed the consistency. And staleness is bounded, so therefore, suitable delay will give a bigger speed up here in this particular case.

Refer Slide Time :( 19:08)

## Stale Synchronous Parallel (SSP)



So, it stale synchronous parallel, if we introduce then we will see that, it is, it is giving the better result. And ensuring the problems of the asynchronous operations.

Refer Slide Time :( 19:20)

## Conclusion

- In this lecture, we have discussed the parameters servers.

So, in this lecture, we have discussed the parameter servers. And the is stale synchronous parallel model of operations, which is supported in the parameter server. Thank you.

## **Lecture 31**

### **PageRank Algorithm in Big Data**

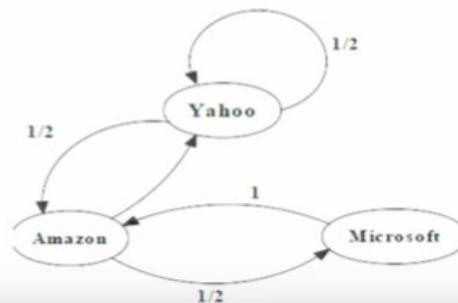
PageRank algorithm in Big Data.

Refer Slide Time :( 0: 17)

## Preface

### Content of this Lecture:

- In this lecture, we will discuss PageRank Algorithm in Big Data using different framework with different ways and scale.

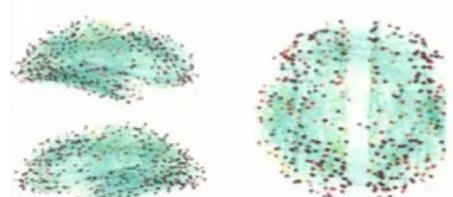


Preface content of this lecture, in this lecture we will discuss PageRank algorithm in big data using different framework, with different ways and the scale.

Refer Slide Time :( 0: 32)

## Big Graphs

- Social scale graph
  - 1 billion vertices, 100 billion edges
- Web scale graph
  - 50 billion vertices, 1 trillion edges
- Brain scale graph
  - 100 billion vertices, 100 trillion edges



Human connectome.  
Gerhard et al., Frontiers in



Social graph from Facebook  
<https://medium.com/@johnrobb/facebook-the-complete-social-graph-b58157ee6594>



Web graph from the SNAP database  
<https://snap.stanford.edu/data/>



Let us consider the big graphs, so in the social scale graph, you can see 1 billion of vertices and 100 million of edges. So, vertices represents the set of people and the edges represents, the relations. So, in a socially scale graph so billions of people are interacting with each other, therefore the social scale graph consists of 1 billion vertices and 100 million edges. In a Webby scale graph which, contains the webpages, as the vertices, are much more than the number of people which are there, in the universe,

hence 50 billion vertices that is the webpages, are there in web scale graph and one trillion edges. Similarly in a brain scale graph that is, neurons, in a human brain, is much more than either the social scale graph or wavy scale graph: that is 100 billion of vertices, are there which are neurons and the 100 trillion edges, are the links between them, connecting those neurons. So, what is there in these graph is that, they are very, big size, therefore the computation of these very big huge graphs, is becoming a challenge and in this particular part of the discussion we will see, how we are going to process and do the computation, on such a big graphs.

Refer Slide Time :( 2: 42)

## What is PageRank?

- Why is Page Importance Rating important?
  - New challenges for information retrieval on the World Wide Web.
  - Huge number of web pages: 150 million by 1998  
1000 billion by 2008
  - Diversity of web pages: different topics, different quality, etc.
- What is PageRank?
  - A method for rating the importance of web pages objectively and mechanically using the link structure of the web.
  - History:
    - PageRank was developed by Larry Page (hence the name Page-Rank) and Sergey Brin.
    - It is first as part of a research project about a new kind of search engine. That project started in 1995 and led to a functional prototype in 1998.

One such algorithm which is quietly, which is well used is called a, ‘PageRank Algorithm’. And PageRank algorithm on using Hadoop technology, is useful for computation on such a graph, of very big size. So, let us understand, about this particular algorithm, which is now, available, which is fairly used in, such a big graph in using this Hadoop technology. So, what we see here is that, the PageRank, which is also called as a, ‘Ranking of the Pages’ is why important, because, the new challenges of information retrieval, on the World Wide Web. Is there and it is very, common in use in most of the applications and the services. So, which considers a huge number of web pages: that is on the World Wide Web: that is 150 million, by 1998 and 1008. So, this particular graph, which is considered, for this computation, is going to be very, very huge and big. So, diversity of web pages, varies between different topics, different quality and so on. Now let us see, what do you mean by PageRank? Page Rank is a method for rating the importance of webpages, objectively and mechanically, using the link structure of the web, meaning to say that, the one might where if it is represented, in the form of the graph, of web pages, where the pages which are linked to each other, which are referring to each other, are having the link. So now, the page, which is being linked most of the other pages, is going to be very important page, whereas the other pages, which are not that much, linked by the other web pages are not that important. So, the method for rating the importance of web pages, using this link structure of the web, is called as the, ‘PageRank’. So, the history of the PageRank, was can be can be traced back. So, the PageRank algorithm was developed by Larry Page and Sergey and Sergey Brin, of Google. It was it is the first as part of the research project, about the

new kind of search engine: that where this Page rank algorithm was used: that project started 1995 and led to the functional prototype in 1998.

Refer Slide Time :( 5: 40)

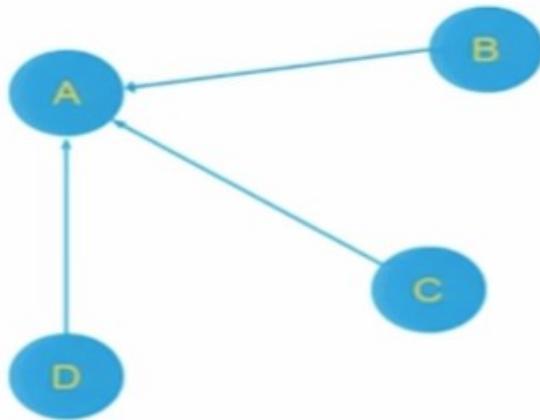
## PageRank of E denotes by PR (E)

- Give pages ranks (scores) based on links to them
- Links from many pages → high rank (important)
- Links from a high-rank page → high rank (Contributes to higher rank)

So, let us see that, using this particular picture. So, the pages which are very important, are shown with a, with a, with the bigger size, compared to the pages, which are not that important, shown by the smaller size. Now, the page rank is denoted by E, let us say, so the page rank we are denoting by the symbol PR. PR is the page rank of the page E. So give, this page rank algorithm will give, the pages the ranks, the score based on the number of links which are pointing to it. For example, this particular, this particular page which is shown, as a very fat, node you see that, most of these links are pointing to it. Therefore the size or the rank or the score of this particular page is higher, compared to the all other pages, we are not that many number of links are pointing to it. And if this particular, fat page, is pointing to some other web page, then that page also, get more importance, hence this kind of ranking, is computed through an algorithm, which gives the ranks or to the pages or this is also called as a, 'Score' which is fairly, based on the number of links, which are pointing to them. So, the number of links, from many pages, will indicate that it is a very high rank, web page or it is an important page and the links from a high, rank page is also, going to be contributing to a high rank. So, considering these two aspect or the rating, into the algorithm, the PageRank has combined these two different notions, to give, the rating or the score, to the webpages.

Refer Slide Time :( 7: 57)

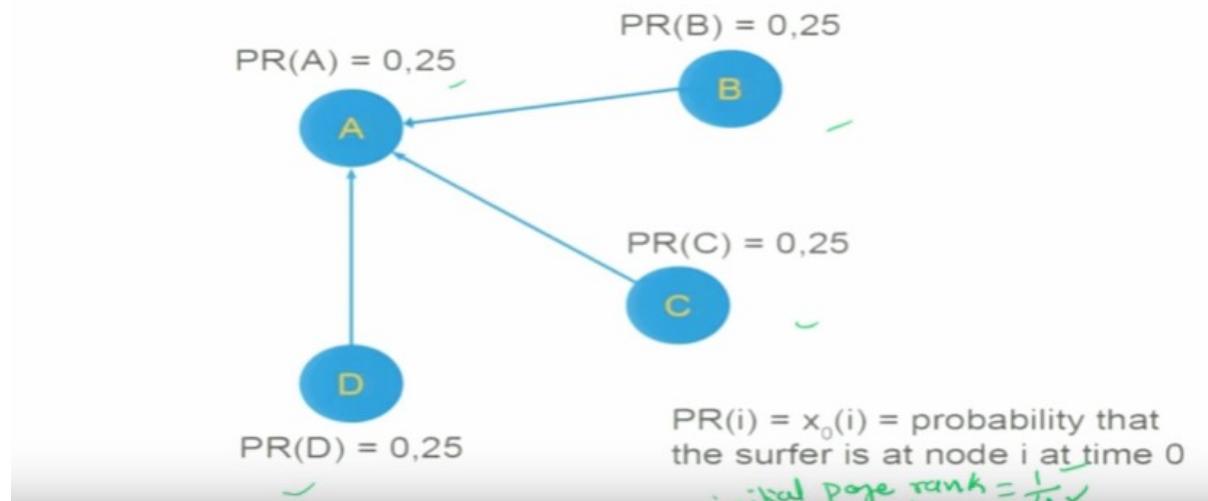
## Example



Now, let us go in more detail of an example, let us say that, node A, is pointed by all other nodes that is B C and D. Now, then in that case, the initial PageRank here, there are four nodes, so initial PageRank is equally divided 1/4 that is, 0.25.

Refer Slide Time :( 8: 22)

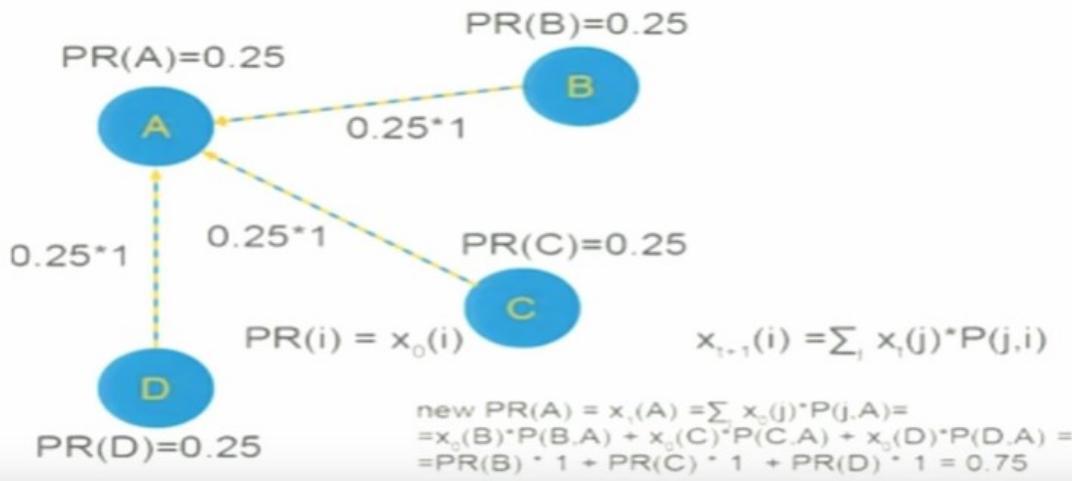
## Example



So, 0.25 is given the initial PageRank of every page: that is the initial page rank is equal to 1/4, for all the nodes, because the total number of nodes is 4 and let us say that, initially, it is to be 1/4. So, 0.25 is the initial page rank. Now, these values, are to be given back, to the page rank, so page rank of a will be recomputed again in the first iterations.

Refer Slide Time :( 9: 02)

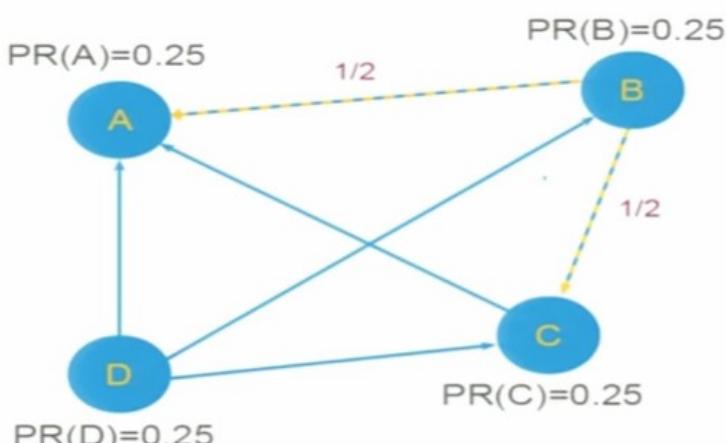
## Example



So, after completing the first iteration, the page rank, of a will be recalculated in this manner.

Refer Slide Time :( 9: 13)

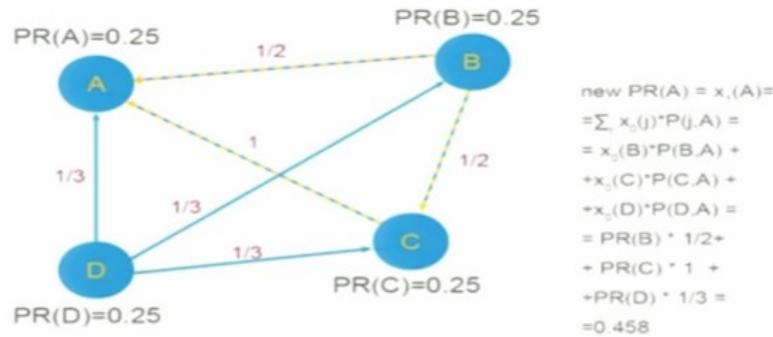
## Example



So, for example, if let us say, B has also added the links, to a and C, so the page rank of B that is 0.25, will be equally divided into two parts and this page rank will be given back, to C and A.

Refer Slide Time :( 9: 35)

## Example



So, if we calculate it.

Refer Slide Time :( 9: 40)

## Page Rank

$L(v)$  - the number of vertex  $v$  outbound links  
 $\Gamma(v)$  - the set containing all pages linking to page  $v$

$$PR(u) = \sum_{v \in \Gamma(u)} \frac{PR(v)}{L(v)}$$

In this manner and the page ranks, values, which are calculated, which is iterated in each iteration in this particular manner.

Refer Slide Time :( 9: 51)

## Page Rank

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \dots$$

$$PR(A) = \frac{1-d}{N} + d \left( \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \dots \right)$$

*dampifactor*

So, PageRank value, is given by this particular formula, where each page for example, page A is pointed by page B, page C, page D, and so on. So, their page rank

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \dots$$

$$PR(A) = \frac{1-d}{N} + d \left( \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \dots \right)$$

Refer Slide Time :( 10: 30)

## Page Rank

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \frac{PR(p_j)}{L(p_j)}$$

where  $p_1, p_2, \dots, p_N$  are the pages under consideration ✓

$\Gamma(p_i)$  is the set of pages that link  $p_i$  ✓

$L(p_i)$  is the number of outbound links on page  $p_i$  ✓

and  $N$  is the total number of pages

So, this particular way the page rank is calculated using this particular formula where  $P1, P2$  and so on  $P$  and are the pages under the consideration and  $\rho_P$  is the set of pages that links to the  $P$  and  $L_P$  is the number of outbound links on page  $P$  and  $N$  is the total number of pages.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \left( \frac{PR(p_j)}{L(p_j)} \right)$$

Refer Slide Time :( 10: 55)

## Page Rank

At t= 0, an initial probability distribution is assumed, usually  $PR(p_i; 0) = \frac{1}{N}$

At each time step  $PR(p_i; t+1) = \frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \frac{PR(p_j; t)}{L(p_j)}$

The computation ends:

1. After fixed number of iterations ✓
2. When convergence is assumed ✓

$$\left( \sum_{i=0}^N |PR(p_i, t+1) - PR(p_i, t)| \right) < \epsilon \quad \checkmark$$

$$PR(p_i; t + 1) \xrightarrow[t \rightarrow \infty]{} x^*(p_i)$$

Now, this particular, after the fixed number of iterations or when the convergence, is assumed, to be a very small value, of changes then the,

Refer Slide Time :( 11: 08)

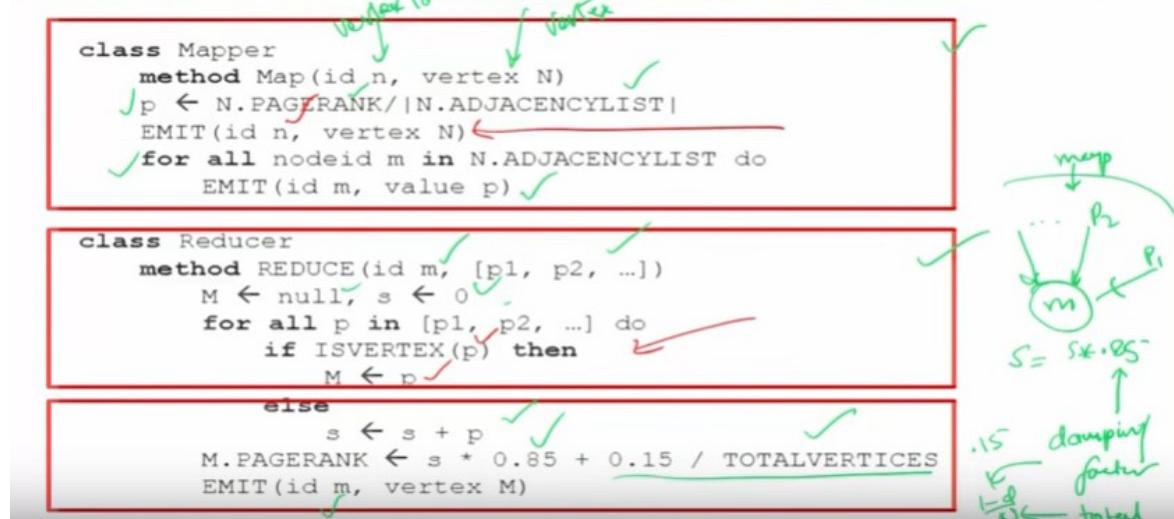
## Summary

- PageRank (PR)- a first algorithm by Google Search to rank websites in their search engine results.
- We have shown how to calculate iteratively PageRank for every vertex in the given graph.

execution of the PageRank algorithm, converges and the final values of the page rank, we the page rank of those algorithm. So, in summary the page rank is the first algorithm, given by the Google search engine to rank the web pages, in their search engine results. So, we have shown how to calculate iteratively the page rank using formula for every vertex in a given graph.

Refer Slide Time :( 11: 35)

## MapReduce for PageRank



Let us see how, Map Reduce can be applied to this PageRank algorithm, so that large scale graphs, we can apply this algorithm PageRank on a very big size crops. So, PageRank may produce applied on a page rank has two components one is called, ‘Mapper’ the other is called the, ‘Reducer’. In the mapper phase, we have taken the input as the node IDs and so, it is the vertex ID and this is the vertex for which we are going to calculate the PageRank. So, for a vertex and what we will do here is that We will find out or we will consider the current page rank of node n and also, you will find out that, total number of adjacent, at the sensi list or the total number of neighbours, of n. So, this particular page rank will be divided by the total number of neighbours: that will be calculating its page rank. And then, for all values of the node IDs, M in its neighbour list or it is as NC list, it will emit, this values, of this ID, m and the value of P. Now, as far as, the reducer is concerned after getting these values M, for the node M, all these values, when it is collected: that is Page ranks from all its neighbours. So, like let us say that this, when it collects the page rank from different  $p_1, p_2$  and so, on up to. So, these page ranks ment, is being sent by the map function and it when it is received at M, then the reducer will now calculate the page ranks. So, it will start with M is equal to null and s, the calculating the value of page rank, in s variable. So, what it will do is, for all p values, which are there receiving the page ranks and if it is, not a vertex: that means a p is not a vertex, then it is the page rank which is being sent. So basically, it will add, the value of s to p and the page rank will be calculated, based on, the PageRank is equal to  $s * 0.85 + 0.15 / \text{TotalVertices}$ , 0.85 is a damping factor and the other portion, where you have to add  $(1 - D) / n$ , so n is the total number of vertices, total vertices and  $1 - D$  that is,  $1 - 0.85$  it becomes, 0.15. So, this will be the total summation of these two will become the PageRank, of no DOM and it will be emitted, from the reducer function, in vertex ID. Now, there is one more function, one more statement which we have omitted is that, this not only will send the PageRank of the neighbouring node, but, the node itself, as the complex object will be emitted out here in this algorithm. So, it will emit N and vertex n, so here it will be checked, if p is the vertex, is p is the vertex, so if p is the vertex, then M will be, added to the p: that means it is the complete complex node object, which is being emitted out of the mapper, not the page ranks. So, both of them,

are being used so p is the complex object is the state variable, which are also, transferred or emitted, along with the PageRank values, so this was the Map Reduce for implementation for PageRank.

Refer Slide Time :( 16: 13)

## Problems

- The entire state of the graph is shuffled on every iteration  
*node object*
- We only need to shuffle the new rank contributions, not the graph structure  
*Pringle, GraphLab, Giraph, GraphX*
- Further, we have to control the iteration outside of MapReduce

So, here let us see the problems, in this implementation of a Map Reduce version of a PageRank here we have seen that, entire state of a graph is shuffled on every iteration. State in the sense we have seen some, the node object as the state variable also in along with the PageRank values, also is being communicated. So, we only need to suffer the new rank contribution not the entire graph structure, so this is the problem of internal implementation so, in a newer version, like Pringle, then Graph Lab or Giraph or graphics has given concept of the internal implementation, of the iterations of a graph, hence the transferring of the entire state, during the shuffle is not required, therefore more efficient versions, are now available the initial of this implementation problem, which we have just seen. Further we only need to shuffle, the new rank contribution not the entire graph structure: that we have seen that in the newer, framework of a graph computation, this is resolved. Further we have to control the iterations outside of the Map Reduce.

Refer Slide Time :( 17: 47)

# Pregel

- Originally from Google
  - Open source implementations
  - Apache Giraph, Standard GPS, Jpregel, Hama
- Batch algorithms on large graphs

```
while any vertex is active or max iterations not reached:
 for each vertex: ← this loop is run in parallel ✓
 ✓ process messages from neighbors from previous iteration
 ✓ send messages to neighbors
 ✓ set active flag appropriately
```

Let us see how the pregel, the PageRank is implemented. Pregel, is originally from the Google it is an open source implementation, the other such open source implementation, which are available called, ‘Apache Giraph’ and standard GPS, then J Pregel and Hama. Batch algorithms on the large-scale graph, processing it is how is being used the Pregel. Now let us see the, iterations in the Pregel, so Pregel all does not, use the Map Reduce notion of the concept which we have seen in the previous slide. So, here in the Pregel, all the entire vertex, is being programmed, only one vertex program is written. And here, we have to see for any vertex, which is active or if the maximum iterations not reached, than for each vertex, this particular loop will, run in parallel, what it will do, in this iteration is that, it will process the messages, from the neighbours, from the previous iterations. So, this is required why because, it will bring the page ranks, from the neighbours and then, it will send the message to the neighbours and set the active flag, appropriately. So, what it will do is, it will collect, it will, it will receive the messages, the messages of the previous iteration and compute or oblique update, the new page ranks and then again, send the message, to the neighbours, about the new PageRank and it will set, the active flag appropriately. So, that means when a particular node, is not doing any of these actions, then the node is not active, its passive and it will be out of the action, in the further iterations.

Refer Slide Time :( 20: 18)

# PageRank in Pregel

```
class PageRankVertex: public Vertex<double, void, double> {
public:
 virtual void Compute(MessageIterator* msgs) {
 if (superstep() >= 1) {
 double sum = 0;
 for (; !msgs->Done(); msgs->Next())
 sum += msgs->Value();
 *MutableValue() = 0.15 / NumVertices() + 0.85 * sum;
 }
 if (superstep() < 30) {
 const int64 n = GetOutEdgeIterator().size();
 SendMessageToAllNeighbors(GetValue() / n);
 } else {
 VoteToHalt();
 }
 }
};
```

*iteration*

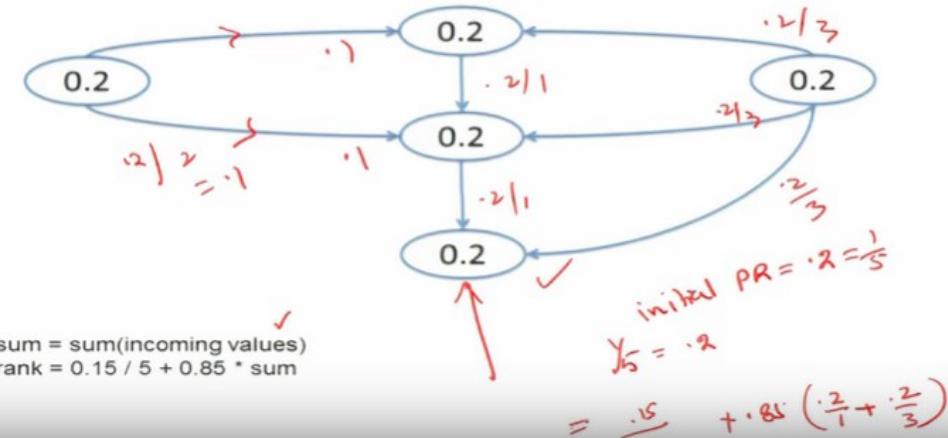
*newPageRate*

*Vertex program*  
in parallel execution of all vertices in Graph.

So, let us go and see the detail algorithm of page rank in the pregel. So, here we, we have, we see that, it is a super step, super step means, this is the iteration, of the PageRank algorithm, on a particular vertex or a node, what it does it will start, it will initialize, the value of sum and it will collect, the messages, the values which comes into the messages and then using this particular summation of all the values, of the PageRank which it has collected, it will apply the damping factor that is 0.85 and then it will calculate  $0.15 / \text{total number of nodes}$  and this will be the new PageRank. Now, if this particular in number of iterations  $< 30$  and then, it will that means it is not terminated, still it has to send the messages. So, it will send the messages, to all the neighbours and send to all the neighbours, divided by n why because, the PageRank will be equally divided, across all the outgoing edges. And this particular iteration will keep on repeating and in the end, it will go to the halt, if no action is being taken. So, this is the vertex program, of the PageRank in prison, you see that, it is not becoming complicated as you have seen in the Map Reduce. So, a vertex program, we can write down and which will calculate the PageRank, in parallel of all the vertices in the graph. So, the parallel execution, parallel graph execution, for this particular code, will run on all the vertices and they will calculate the page rank or iterations and once the iterations finishes: that is, it converges, the PageRank and it will be the value.

Refer Slide Time :( 22: 57)

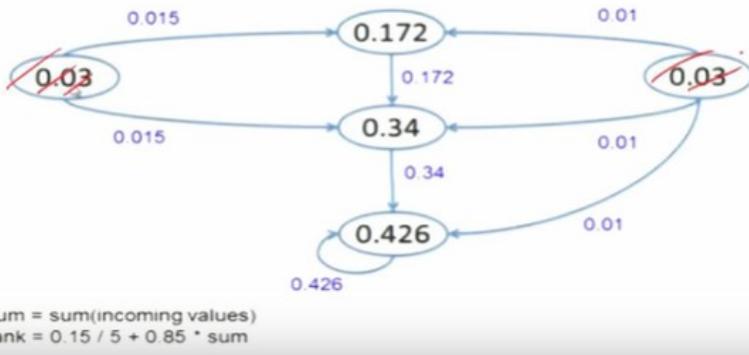
## Example



So, this is the PageRank implementation. Let us see the example, of this algorithm and this approach. So, this is the initial condition, so initial page ranks, of all the roads will be, so there are five nodes, so  $1 / 5$  that is 0.2. So, 0.2 will be the initial PageRank and that is  $1 / 5$ , it is given to all the nodes. Now, let us see that, this particular node has 2 outgoing, 2 outgoing links, so hence, so it is 2 outgoing links, so basically the share of the PageRank which will go along this link is, 0.2 divided by 2 that becomes 0.1. So, 0.1 and 0.1 will be the contribution, of the PageRank, on all these links similarly here, 1, 2, 3. So,  $0.2 / 3$ , will be the PageRank which will be sent along that link. And here 0.2, will be there is only one outgoing edge, so the  $0.2 / 1$  and here also,  $0.2 / 1$ , this is there. Now, as far as, this node is concerned, let us consider this node. So, what it does is, it will calculate, the sum of incoming page ranks: that is  $0.2 / 1 + 0.2 / 3$ . So and then it will multiply by 0.5 that is, a damping ratio and plus, what it does, is that, it will add 0.15/total number of nodes, total number of nodes is: that is three, here in this case. So, in this manner the incoming,

Refer Slide Time :( 25: 06)

## Example



so basically you can see here, the value will become 0.426, over here and this is 0.1 and why these values are changed is because, this contribution is 0. So,  $0.15 / 5$  will become 0.03 and this is the iteration number one and in the next iteration, you see that, these values, these two values are not changing in the next iteration.

Refer Slide Time :( 25: 43)

## Example



So, these two values are not changing in the rate in the next iteration so, they will become red. Red in the sense, they will stop participating in the further iterations and these values, have to be which are sent by these nodes earlier they will be stored in the buffer, because in the in the further iteration they will not participate but, these values are, going to be fixed and a constant which is going to be used. And you see that, here now in this iteration this particular node is out of the action, so it will be not active and finally, at this stage the page rank algorithm, will terminate, with these values.

Refer Slide Time :( 26: 30)

## Conclusion

- Graph-structured data are increasingly common in data science contexts due to their ubiquity in modeling the communication between entities: people (social networks), computers (Internet communication), cities and countries (transportation networks), or corporations (financial transactions).
- In this lecture, we have discussed PageRank algorithms for extracting information from graph data using MapReduce and Pregel.

So, in conclusion graph structure data, are increasingly common in data, in data science context, due to their ubiquity in modeling the communication between entities: that is the people, in the social network computers, in the internet communication cities and countries in the transportation network. And the web page is no word white web or the corporations in the financial car operations. So, in this lecture, we have discussed important algorithm: that is a PageRank algorithm for extracting the information, of the big graphs, big graph data. Using the Map Reduce and using the pregel paradigm. Thank you.

**Lecture – 32**  
**Spark GraphX & Graph Analytics (Part-I)**

Spark GraphX and Graph Analytics.

Refer slide time :( 0:18)

## Preface

**Content of this Lecture:**

- In this lecture, we will discuss GraphX: a distributed graph computation framework that unifies graph-parallel and data parallel computation for Big Data Analytics and also discuss as a case study of Graph Analytics with GraphX.

The diagram illustrates the components of Apache Spark. It features four blue rectangular boxes arranged horizontally at the top, each containing a component name: "Spark SQL", "Spark Streaming", "MLlib (machine learning)", and "GraphX (graph)". Below these four boxes is a larger blue rectangular box labeled "Apache Spark" in white text. The "GraphX" box is positioned above the "Apache Spark" box, indicating its status as a core component.

Preface; content of this lecture; in this lecture, we will discuss GraphX or distributed graph, computation framework. That unifies graph, parallel and data parallel computation, for big data analytics. I had also discussed in this lecture, a case study of a graph analytics, with GraphX here, on the bottom right, we have shown the position of GraphX, which is above the spark, apache spark, core. We are going to discuss, this component that is the GraphX, which is a part of core spark in this discussion.

Refer slide time :( 0:18)

# Introduction

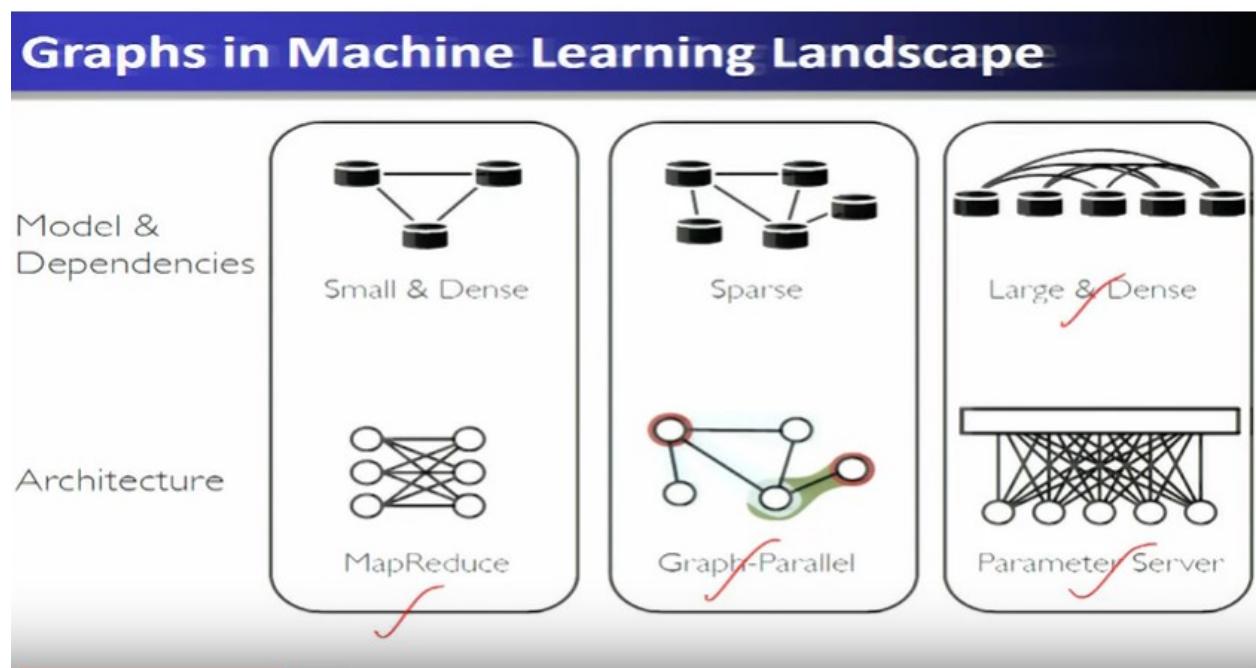
- A piece of technology build on top of spark core. — *GraphX*
  - Graphs like our social network of graphs in the computer science/network sense.
  - However, Graphs are only useful for specific things.
    - It can measure things like “connectedness”, degree distribution, average path length, triangle counts-high level measures of a graph.
    - It can count triangles in the graph, and apply the PageRank algorithm to it.
    - It can also join graphs together and transform graphs quickly
    - It supports the Pregel API(google) for traversing a graph.
    - Introduces VertexRDD and EdgeRDD and the Edge data type.
- 
- 7:14 / 41:15

Introduction; so, a piece of technology, which is, built on top of the spark core, we are going to discuss that is called, ‘GraphX’, the graphs are like, our social network graphs, which are very well-known in computer science and in the network, complex networks, however the graphs are only useful for specific things, for example it can measure the things like connectedness, within a particular domain, the degree distribution, average path length, triangle counts, high level measures, of a graph in the social network graphs. The graphs can also count triangles in a graph, which basically will tell about, the cohesiveness, properties exist within a particular scenario. And also it can apply; we can apply the PageRank algorithm, to the graph to find out the ranking of the nodes. PageRank algorithm is also well known algorithm, which is by the Google to find out the most relevant pages, web pages which a user wants at a particular instance of time. So, for any search engine this PageRank algorithm is a well known, algorithm and is going is I will used in the web graph, it can also, join the graph, together and transforms, the graph quickly we will see this particular aspect of the transformation and joining of a graph together, to solve various problems, in the further slides. It can also support the pregel API that is given by the Google, for traversing a graph.

Now people also introduce here, about how the vertices and edges of a graph, what are the different data models, which are supported in the GraphX, which runs on top of the spark core. So, as you know that spark supports, the data type which is called a ‘Resilient Distributed Data Set’. That is called RDD’s, now this particular RDD’s will use the vertex RDD. So, that means the vertices of a graph are represented as resilient distributed, vertex data sets that is called, ‘Vertices Vertex RDD Similarly’, the edges of a graph are represented as the edge RDD that is the edge component, of a graph, which is represented as the edge resilient distributed, as data set. Together these particular components, the virtus vertex RDD and edge RDD’s, which are supported, to construct a graph that is called a, ‘Property Graph’, in GraphX. So, graph is called, ‘Property Graph’. Which is nothing but the vertex RDD and edge RDD, which is supported as a graph, in the GraphX, also there is another viewpoint, where in vertex and edges are combined together that is called a, ‘Triplet’. The concept of triplet is very useful, in building these graph algorithms, which

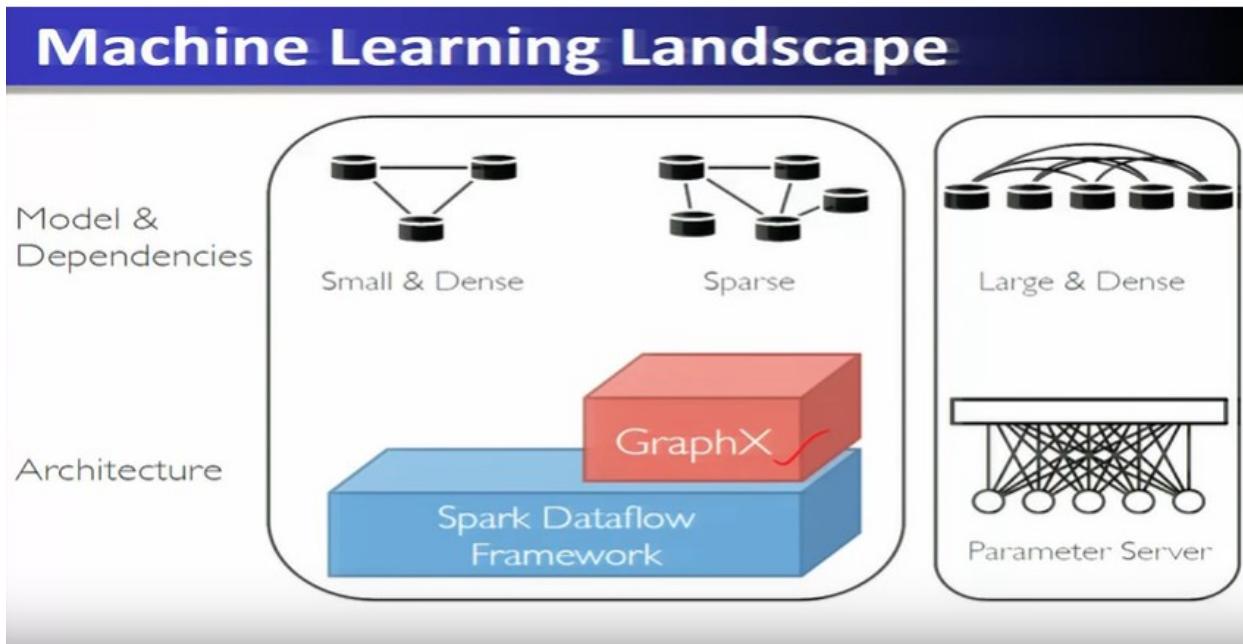
we have discussed like PageRank algorithm connectedness and finding out, the triangle counts degree distribution and the connected components, which are some of the important graph algorithms, such as the PageRank, algorithm, then connected components, algorithm and then triangle, counts counting algorithm and degree distribution, average path length and for graph traversal, algorithms, which are going to be very much useful single, source shortest paths and so, on there are various graph algorithms. Which we are going to see and these graph algorithms, are iterative in nature, therefore, the support of the graph that is called a, 'Property Graph'. And the triplets that we will, see how this GraphX will provide the important basic features and also the libraries of GraphX, which support all these algorithms, in an optimized manner.

Refer slide time :( 07:18)



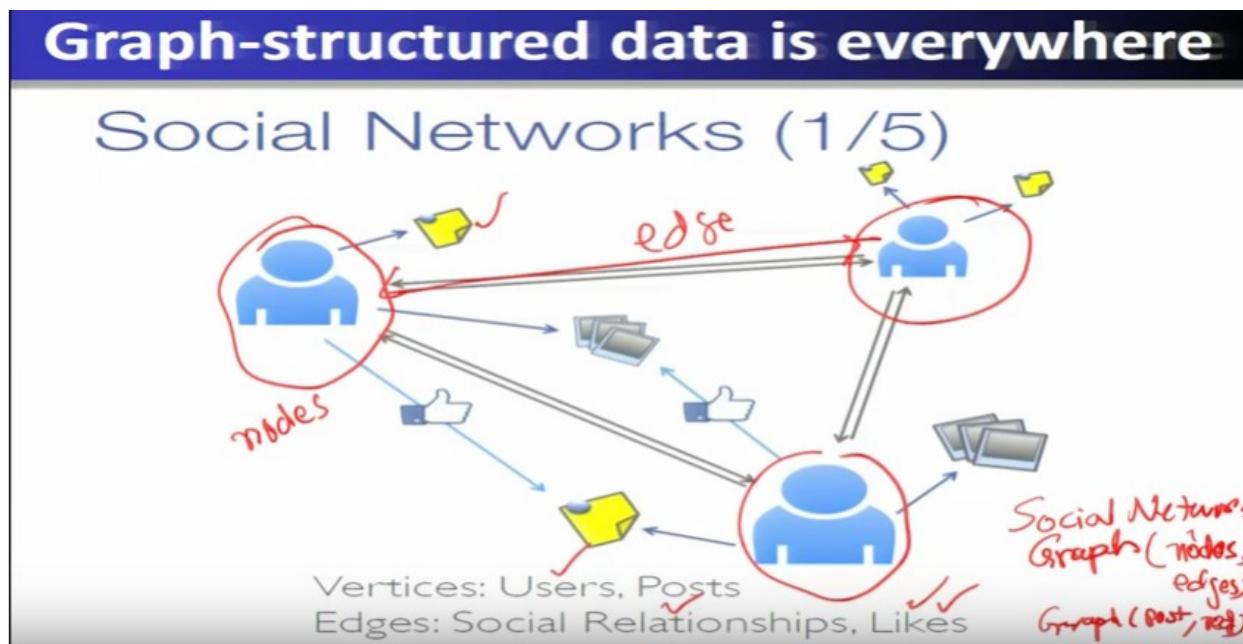
Now graphs are also used in the machine learning, landscape, for example, the models and dependencies, of the data, which can be represented in the form of a graph, whether it can be a small and or a dense graph or it is a sparse graph. So, that means once, a data is represented in a form of a graph, then machine learning, can be applied on this particular graph as a data, these particular aspects we will discuss more details, in this part of the course. So, what do we mean, by the parameter server that means, when we going to deal with, the programming of the machine learning using graph, we will basically encounter, with the large and dense graph, which has to be dealt with the parameter servers, similarly the operations, which are there in the machine learning, algorithms, which are implemented on top of the data sets, which are represented as a graph, has to be done in a graph parallel and data parallel operations that we will see, how we are going to support these, architectures. Similarly another architecture, which is going to be supported, for the big data landscape and which is there in the machine learning landscape, is about the MapReduce, application on small and dense scenarios, of a Big Data.

Refer slide time :( 08:50)



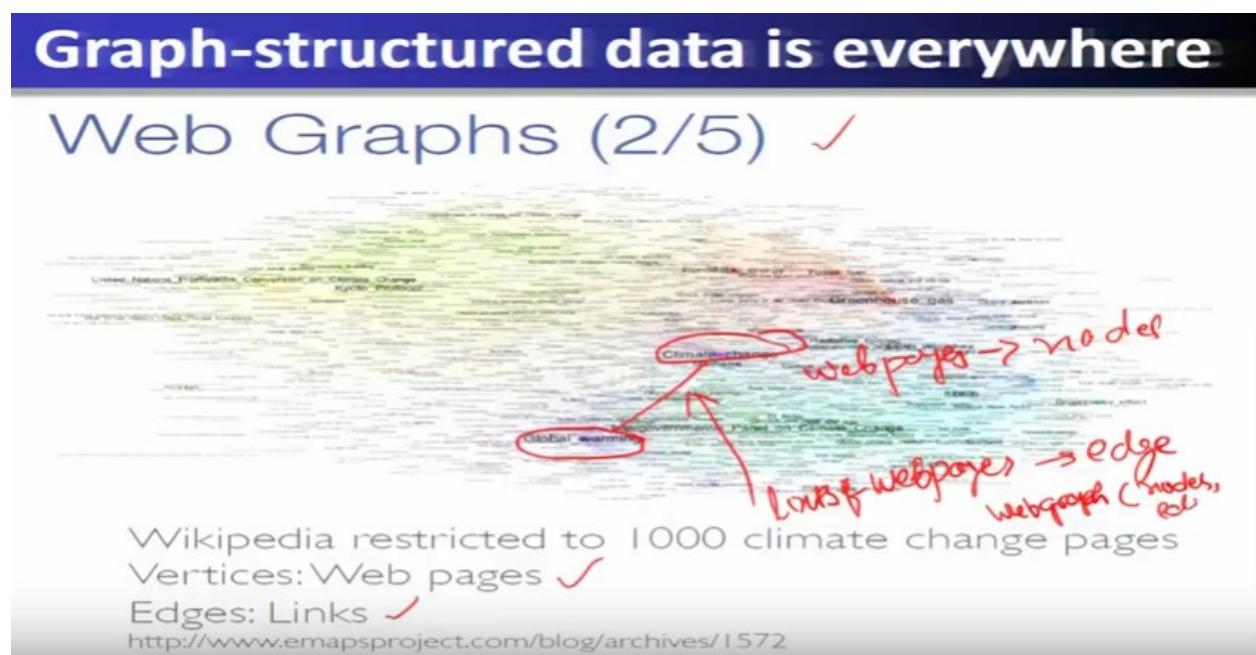
So, the basic architecture, which we will discuss here is the GraphX, how these GraphX, will be supported as the framework, which can also build the machine learning landscape, similarly all these part, we will discuss in this part of the lecture. So, graphs are there everywhere.

Refer slide time :( 09:18)



So, graphs are a structural data, which is there everywhere let us take some examples, where you can understand that the importance, of that graph, computation is very much required in today's workloads. So, for example social network, which is visualized as a social network graph, in this social network graph, the people are called as the, 'Nodes'. So, these are the people, they are called the 'Nodes', of a graph and the relationship, between these people they represent the edge. So, hence, the graph comprises of the nodes and the edges, of a social network graph, where the nodes represents the people or the users and the social relationship, is represented by the edge, similarly the people who does, the post and they also, performs the like operation is also can be represented, in, in a form of a graph that represents, the post as the nodes and the likes, is basically the relationship, as an edge. So, the graph data or a social network data, you can represent, in a different form of graphs and further do the analysis, on top of it that we will discuss. So, graphs are a structured data and it is there everywhere.

Refer slide time :( 11:10)

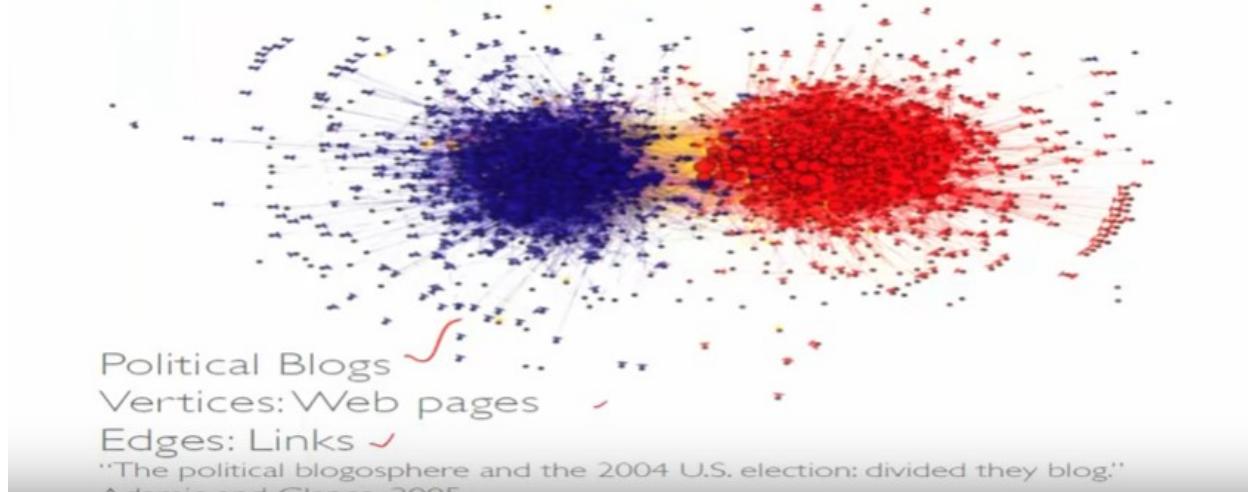


Another searching example, of the graphs is about the web graphs. So, for example, the Wikipedia has 1,000 different, climate change pages. So, if we represent the Wikipedia, as a graph as a web graph, then we can find out that all the web pages, will become the vertices and the links are among these web pages, they becomes the edges hence, this will form the web graph. So, we can see here, these are the web pages global warming and climate change and so, on they becomes the vertices and the connections between or the links between, these edges or the web pages, the links of the web pages, becomes an edge and these web pages, is are the nodes. So, this will form a web graph, which comprises of the nodes and edges.

Refer slide time :( 12:24)

## Graph-structured data is everywhere

### Web Graphs (2/5)

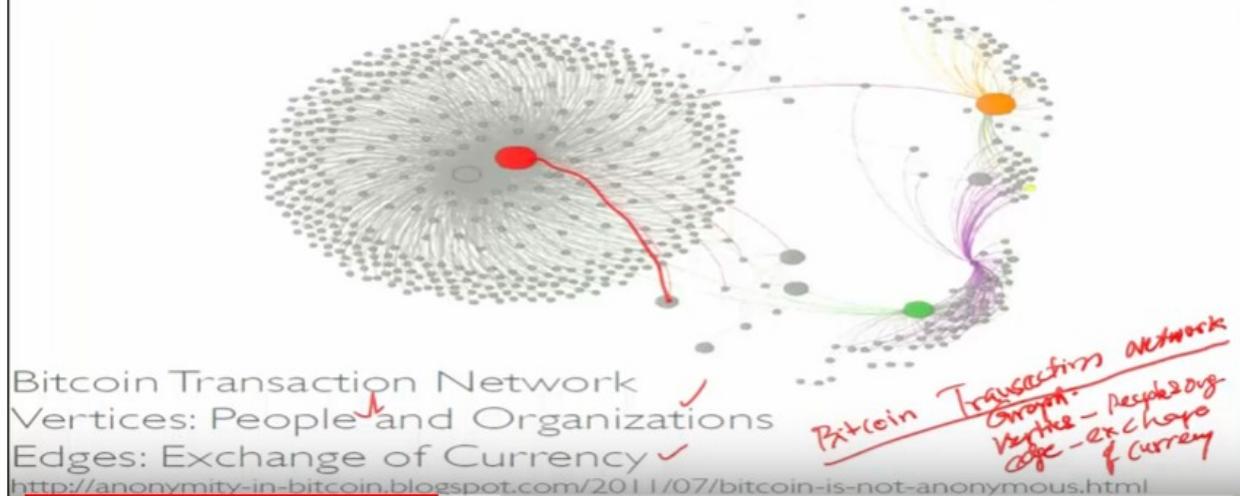


Similarly another form, of web graph is the political blogs data. So, that means the blogs, which represents the political blogs is also represented in the form of a web graph, wherein the web pages are the vertices and the links are the edges. So, the graph which is formed out of, these web pages and the links are called as the ‘Political Blog Graphs’, they are also the web graphs.

Refer slide time :( 12:55)

## Graph-structured data is everywhere

### Transaction Networks (3/5)



Similarly, the Bitcoin Transaction Network, also can be represented in a form of a transaction, graph or a transaction Network graph, we are in the vertices, represents the people and the organizations, which are executing the transactions, of the Bitcoin and the edges are representing, the interactions or exchange of currency, therefore this kind of interactions, between the people and exchange of currency, they can be captured in the form of a graph. And it is called the 'Transaction Network', Bitcoin transaction network graph.

Refer slide time :( 14:10)

# Graph-structured data is everywhere

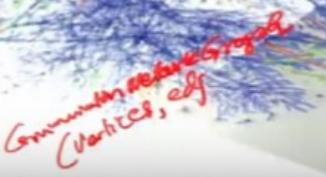
## Communication Networks (4/5)

connections such as hyperlinks, friendships, or email. In fact, graphs play a role in almost every aspect of our daily lives. This section gives a few further examples, and illustrates how many different kinds of data on a basic basis have graph embeddings.

### 2.2 Paths and Connectivity

We now turn to some of the fundamental concepts of network graph theory, perhaps because graphs are so amenable to the use of standard mathematical and theoretic notation. We begin with the concept of connectivity. This is often described in graph theory as a "terminal condition," in that it is the condition that must be met, ultimately, for our purposes. We will be able to see in a few minutes that this is indeed the case.

Internet communication network  
Vertices: Devices, Routers  
Edges: Network Flows  
<http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture2/lecture2.html>

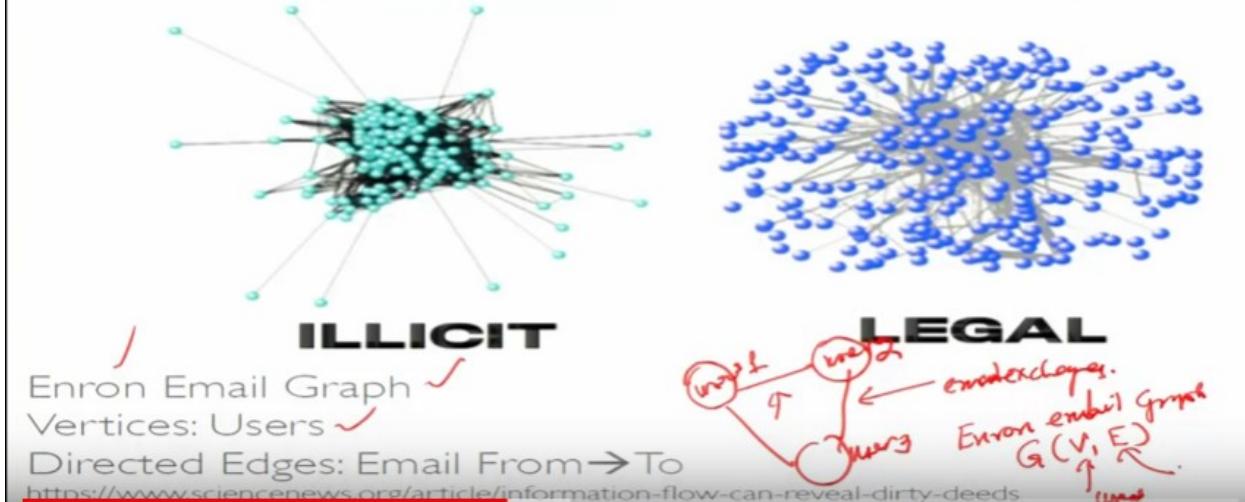


Similarly another kind of graph is, there in the internet communication network, we can also be represented in a form of a communication network graph, wherein the vertices are nothing, but the routers and internet devices. And the edges are the network, interactions or the flows of the data, between these devices is represented, as an edge therefore the communication, network can be modeled, in the form of a graph, communication network, graph, where the nodes are or the vertices are the routers and various internet devices. And the connections, between them that is also to be taken, as the network flows, is becoming an edge and this becomes a graph.

Refer slide time :( 15:11)

## Graph-structured data is everywhere

### Communication Networks (4/5)

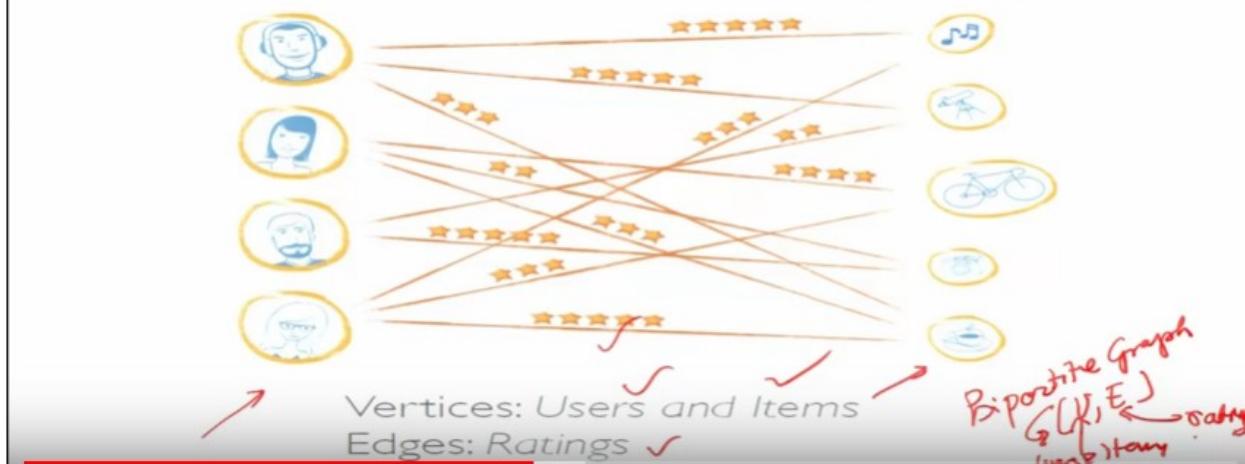


Similarly to detect the frauds among for the companies. So, for example Enron email graph can also be build up can be used. So, Enron was the company, which has committed fraud. So, their email exchanges, between different users is modeled in the form of a graph for, this kind of analysis. So, here the users will become the vertices and the email exchanges are basically the edges. So, this will form, the Enron email, email graph, wherein the vertices, when vertices are the users who have exchanged the mails. And these edges are the set of, users who have communicated, through the mail.

Refer slide time :( 16:31)

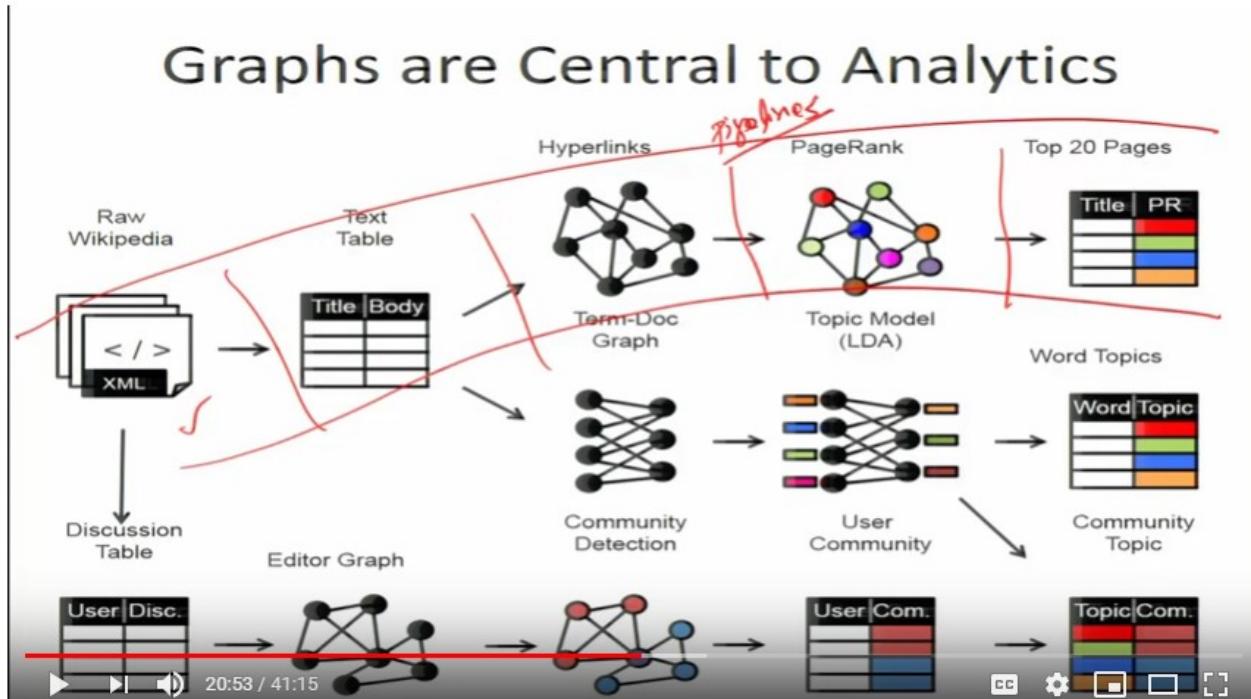
## Graph-structured data is everywhere

### User-Item Graphs (5/5)



Similarly the user item graph can also be visualized, as a bipartite graph, wherein the vertices are the users and items and their ratings that means the users will give the rating, to the items or the products that becomes an edge, with the weights given as the ratings. So, bipartite graph, user item graph, is a bipartite graph, where the vertices are the users and items, whereas the edges are the ratings, given by the user for different products, can be represented in a form of a graph.

Refer slide time :( 17:24)



Now let us see some of the analytics, which can be performed, on the graph. So, graphs are central to the analytics also, let us consider the raw Wikipedia, purpose and using this Wikipedia, account corpus we can construct a table that is called a 'Text Table', which has the title and the body within it using this particular text table. We can construct and hyperlink graph, on hyperlink graph, we can run the PageRank algorithm. And after this PageRank algorithm, we can find out the top 20 different pages. So, their title and their page ranks are extracted in a form of it of a in a particular table. Now similarly from a, text table we can construct another graph, which is called a 'Term Document Graph'.

And this particular term document graph, we can apply topic model algorithm, I and from that we can extract, the word topic table that is word and topic table, on top of it similarly, from the raw Wikipedia again, we can construct another table, which is called a 'Discussion Table', wherein the users and discussion, topics are mentioned, from this we can generate an, editor graph and we can apply a community detection algorithm on top of it. So, we can identify the user community users and different communities, which are there in the Wikipedia text. Now we can combine both of them that is the topic model and the user community, together to get the topic community, information in the form of a table.

So, these from Raw Wikipedia. We can get top 20 pages with the title and their page ranks, similarly from the Raw Wikipedia text; we can get the word and their topics, corresponding similarly from Raw Wikipedia. We can get the topics and the different communities. So, this is basically, the start point of analytics and this flow or the sequence in, which we can get this particular, output or the analysts' analysis, out of that text is basically called a 'Pipeline'. So, we will see here, in GraphX that not only it provides, the analytics, which can be performed on the raw text or in raw data. And these are the steps or the stages, to transform the data from Raw Wikipedia, into the output of that is top 20 pages and extract this insight, from the data is called the 'Analytics'. So, the graphs are also used, in the analytics and this support of making the pipeline, building the pipeline and performing, the analytics is also supported, in the GraphX that we will see in this discussion.

Refer slide time :( 20:58)

## PageRank: Identifying Leaders

$$R[i] = 0.15 + \sum_{j \in \text{Nbrs}(i)} w_{ji} R[j]$$

Rank of user  $i$

Weighted sum of neighbors' ranks

✓ Update ranks in parallel ✓  
 Iterate until convergence

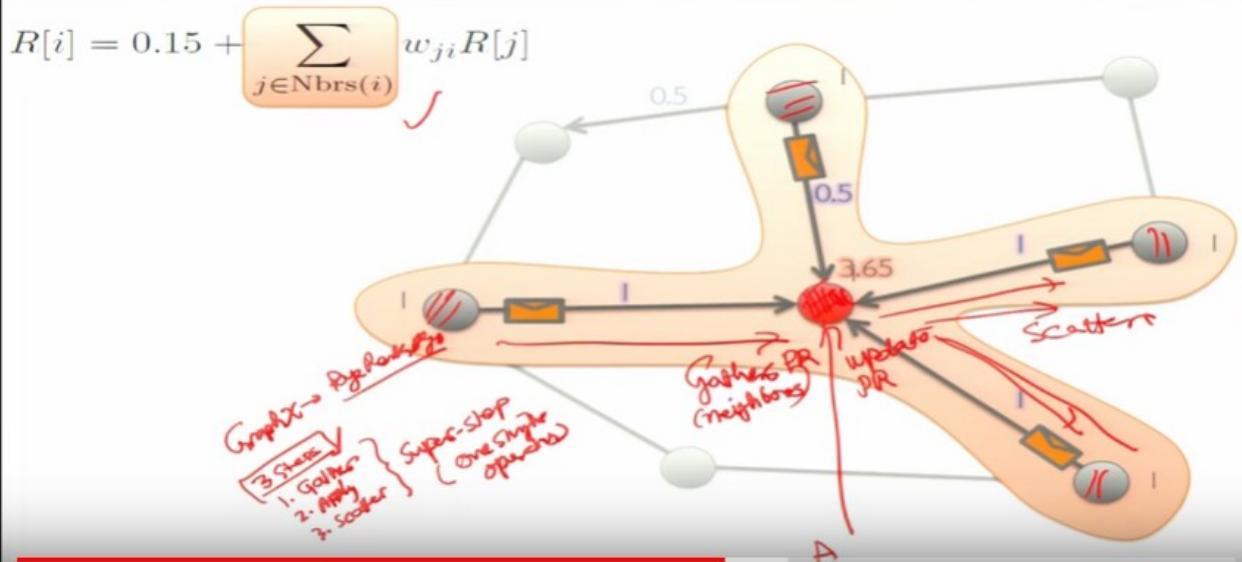
Now we will see the page rank, which is very central to the graph analytics. So, here we can understand that this page rank, algorithm is a parallel computation algorithm and it also performed, in different iterations that also, has to be done in a parallel, I and this iterations, will continue until it converges. And every iteration has to calculate,

$$R[i] = 0.15 + \sum_{j \in \text{Nbrs}(i)} w_{ji} R[j]$$

find out, with a new page rank and this kind of page ranks, are updated this update of the page rank, continues over several iterations till it converges. And then only this page rank algorithm will finish its operations.

Refer slide time :( 21:51)

# PageRank

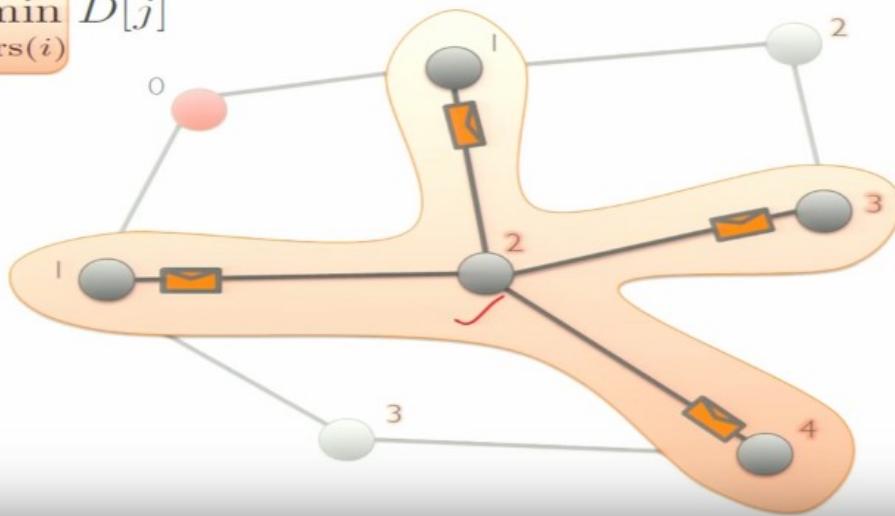


So, let us take, the internal important operation, which is required, to be supported, to run the page rank algorithm efficiently is that. So, first the neighbors or a neighbor pages, neighbor nodes, which represents the pages, they send their data, rather we can see that if you want to find out the page rank, of the node a so, it collects or it gathers, the page rank information, from the neighboring node, after gathering it, it will then compute, its update, it then it will update, its page rank and then it will again, disseminate and then it will disseminate the same information again that is called ‘Scatter’. So, all these three steps, gather, apply and scatter, is treated as the super step that is it will be treated as, one single operation. And if it is supported then writing, the PageRank algorithm will be quite easier and also will become very efficient. So, these kind of constructs, are being provided, in the graph X that we will see and using these constructs, the GraphX also, provides the PageRank algorithm as its library functions. So, all the library functions, which are provided as the graph algorithms, in GraphX are highly efficient and optimized in this particular scenario.

Refer slide time :( 23:59)

## Single-Source Shortest Path

$$D[i] = 1 + \arg \min_{j \in \text{Nbrs}(i)} D[j]$$

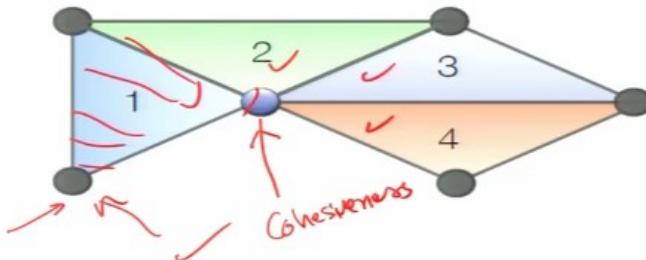


Similarly, we can also see that, shortest path algorithm also exists, as the library, in the library of GraphX, is also using this gather, apply and scatter paradigm and using this in parallel, at all the nodes this particular function, will be performed in data parallel and computation parallel operations are being done in to support these algorithms.

Refer slide time :( 24:35)

## Finding Communities

Count triangles passing through each vertex:



Measure “cohesiveness” of local community

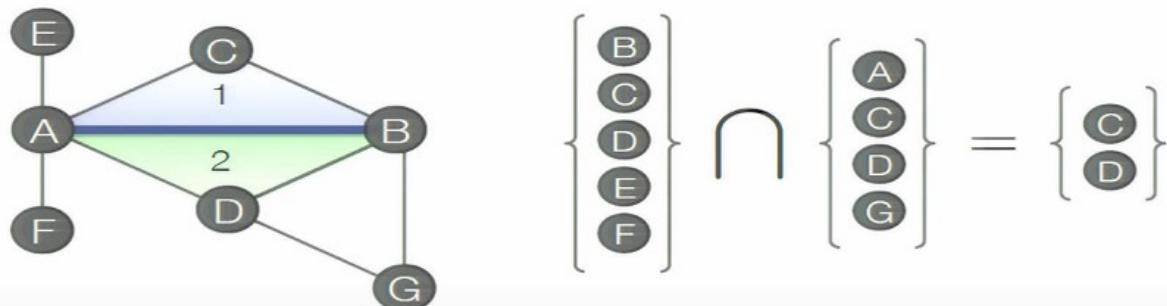
$$\text{ClusterCoeff}[i] = \frac{2 * \# \text{Triangles}[i]}{\text{Deg}[i] * (\text{Deg}[i] - 1)}$$

Another algorithm is to find, the triangles, is to count the triangles and in this manner, too we have to measure, we can measure the cohesiveness within that community. So, this particular way, by counting the triangles. So, here we can see that so, as far as this node is concerned one it has one, two, three, four, four triangles, which are passing through this node, one hence this particular, node is a part of large number of community hence, this particular node gives the cohesiveness compared to the node compared to the other nodes, which are not that conceived. So, we can count. So, this particular node has only the vicinity of one triangle, which passes through only one triangle. So, this particular node, which is shown over here, is more coercive compared, to the other nodes. So, here we are going to analyze, these particular behaviors, in any community, which we are going to deal with that. So, counting triangles is one of the important algorithms.

Refer slide time :( 25:51)

## Counting Triangles

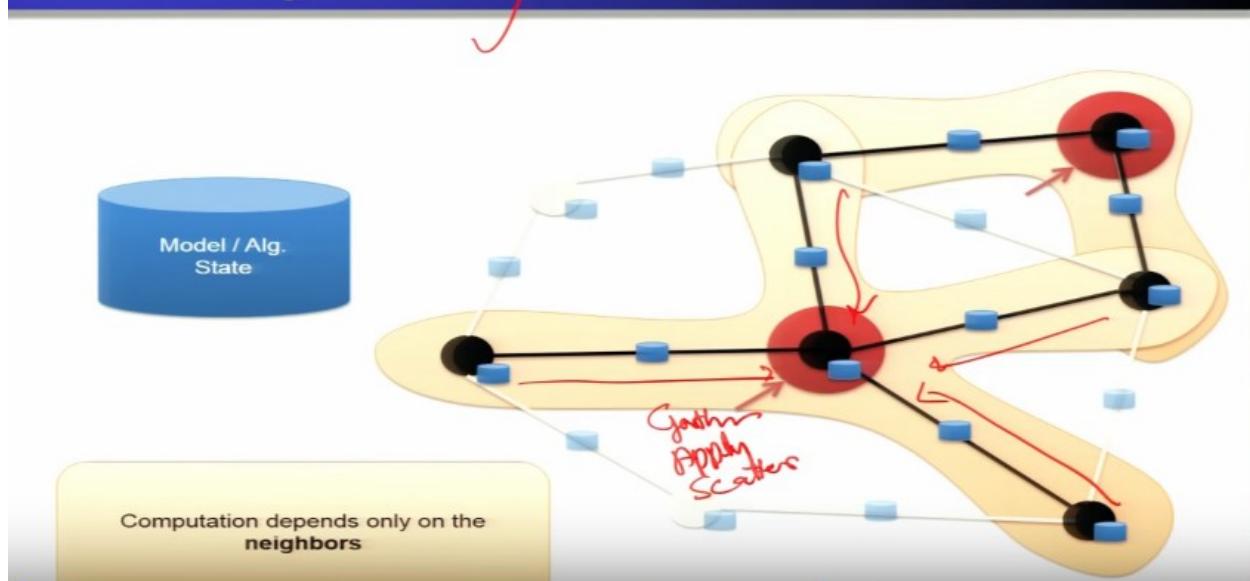
Count triangles passing through each vertex by counting triangles on each edge:



Which is supported as far as the GraphX libraries that we will see.

Refer slide time :( 25:56)

# The Graph-Parallel Pattern

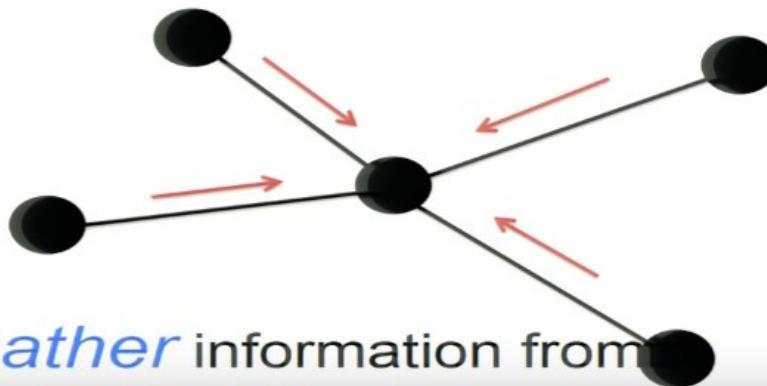


So, now the graph parallel operations, we will see the pattern, how that is all supported. So, consider this particular graph and we are going to take this particular node. And we have to run the algorithm on this node. So, if we can run this algorithm on this node, we can run in parallel at all other nodes. So, let us understand, how this particular algorithm, we can design. So, computation in most of the graph algorithm depends upon the neighbor node. So, it will gather the information from the neighbor node, perform the computation, gather apply perform, the computation and then scatter. This information, which is being computed for updation, at all the ends. So, this kind of operation, is in parallel being run at all the nodes, hence this Groff parallel, computation can be performed, in this particular manner.

Refer slide time :( 27:04)

## Graph-Parallel Pattern

- Gonzalez et al. [OSDI'12]



Therefore the graph parallel pattern is a part of GraphX implementation. So, let us see the parallel, graph parallel pattern is supported by, a gather apply and scatter.

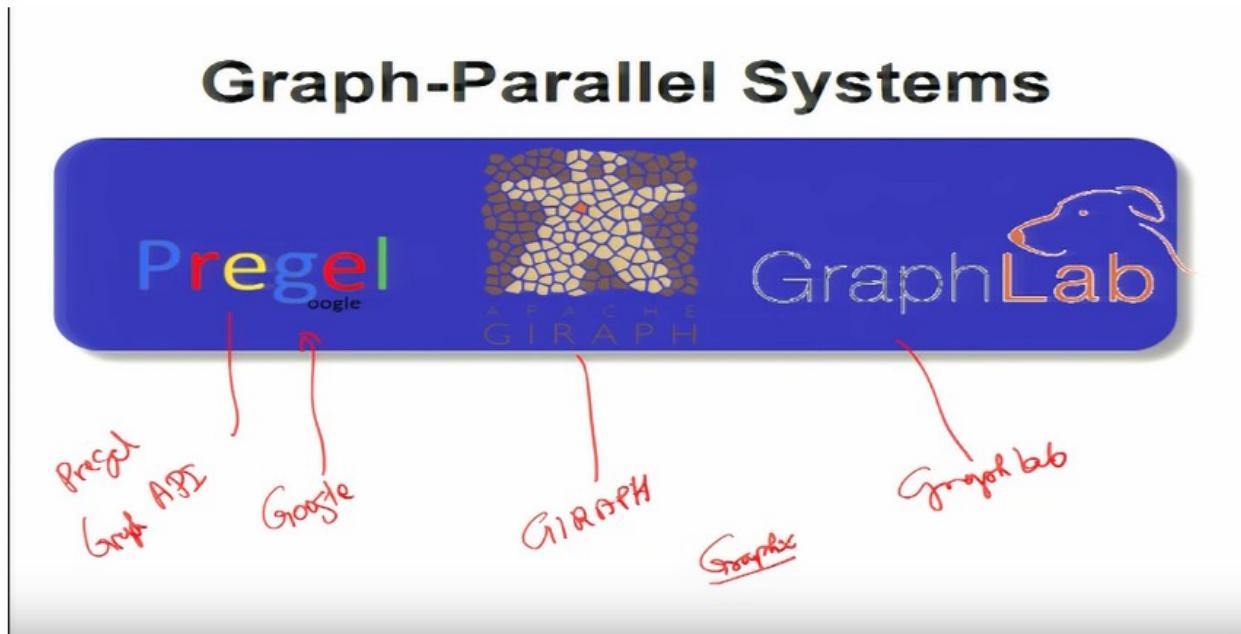
Refer slide time :( 27:19)

## Many Graph-Parallel Algorithms

- Collaborative Filtering ✓
  - Alternating Least Squares
  - Stochastic Gradient Descent
  - Tensor Factorization
- Structured Prediction
  - Loopy Belief Propagation
  - Max-Product Linear Programs
  - Gibbs Sampling
- Semi-supervised ML ✓
  - Graph SSL ✓
  - CoEM
- Community Detection ✓
  - Triangle-Counting ✓
  - K-core Decomposition
  - K-Truss
- Graph Analytics ✓
  - PageRank ✓
  - Personalized PageRank
  - Shortest Path ✓
  - Graph Coloring ✓
- Classification
  - Neural Networks

Notion using this many graph parallel algorithms, are now available with the GraphX, libraries some of them are performing, the graph analytics as, we have told that page rank algorithm, is their shortest path graph coloring, all these algorithms, are available, which uses this graph parallel framework. For community detection, triangle counting k-core decomposition, k truss all these algorithms, are available similarly for machine learning, we have graph SSL and cohere algorithms available similarly, for collaborative filtering, alternating least square, LS algorithm is also, there and stochastic gradient descent tensor factorization, these algorithms, are using the graph parallel approach for building.

Refer slide time :( 28:16)



So, graph parallel systems, are available as on date is given, by the Google. That is in the form of Peregrine API that is called a ‘Pregel’, another graph system, which is called ‘Apache Giraph’. And another one is called ‘Graph Lab’. And we will see that graph hacks is performing better, in all these scenarios. So, we will expose different API is to simplify, the graph computation and that is the graph parallel computation systems.

Refer slide time :( 29:11)

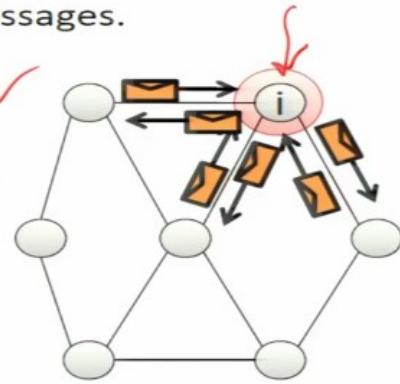
# The Pregel (Push) Abstraction

- “Think like a Vertex.” - Pregel [SIGMOD’10]
- Vertex-Programs interact by sending messages.

```
Pregel_PageRank(i, messages) :
 // Receive all the messages
 total = 0
 foreach(msg in messages) :
 total = total + msg ✓ Gather

 // Update the rank of this vertex
 R[i] = 0.15 + total Apply

 // Send new messages to neighbors
 foreach(j in out_neighbors[i]) :
 Send msg(R[i]) to vertex j Scatter
```



Now let us see that how the pregel, gives an abstraction and in so, that the graph computation, becomes quite easy and for the programmer, intricacies are hidden into, the framework. So, in the Playgirl, it will be war tech centric operations. So, that means you have to think, like a vertex and it does not require, MapReduce to be introduced, here in the program. So, only vertex centric, computations or program can be written, by the programmer hence it becomes very easy to write, the graph algorithms using pregel. Now vertex programs will interact by sending the messages. So, let us see here, in this typical example that. So, if we are going to write down, for this particular vertex ‘i’ the page rank, algorithm in pregel. So, it will perform this particular page rank on, this particular node ‘i’ by sending different messages. So, first it will receive the messages, from its neighbor and then it will do the aggregation, of it and after that after that, then it will compute or it will update, the rank that is called the, ‘Apply Operation’. And then it will scatter these two different nodes. So, this particular paradigm is being applied, at all the vertices in parallel. So, only one vertex program you have to write down a rest of the algorithm is taken care. So, writing algorithm, graph algorithms using Pregel, becomes quite easy without, without bothering about much intricacies.

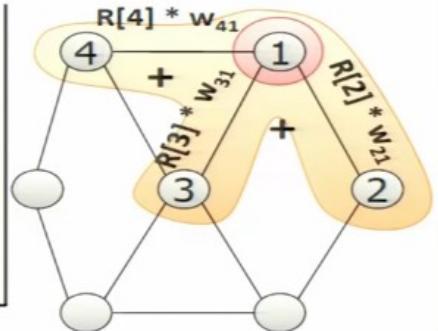
Refer slide time :( 31:09)

# The GraphLab (Pull) Abstraction

- Vertex Programs directly access adjacent vertices and edges

```
GraphLab_PageRank(i) ✓
 // Compute sum over neighbors
 total = 0
 foreach(j in neighbors(i)):
 total = total + R[j] * wji ✓

 // Update the PageRank
 R[i] = 0.15 + total ✓
```

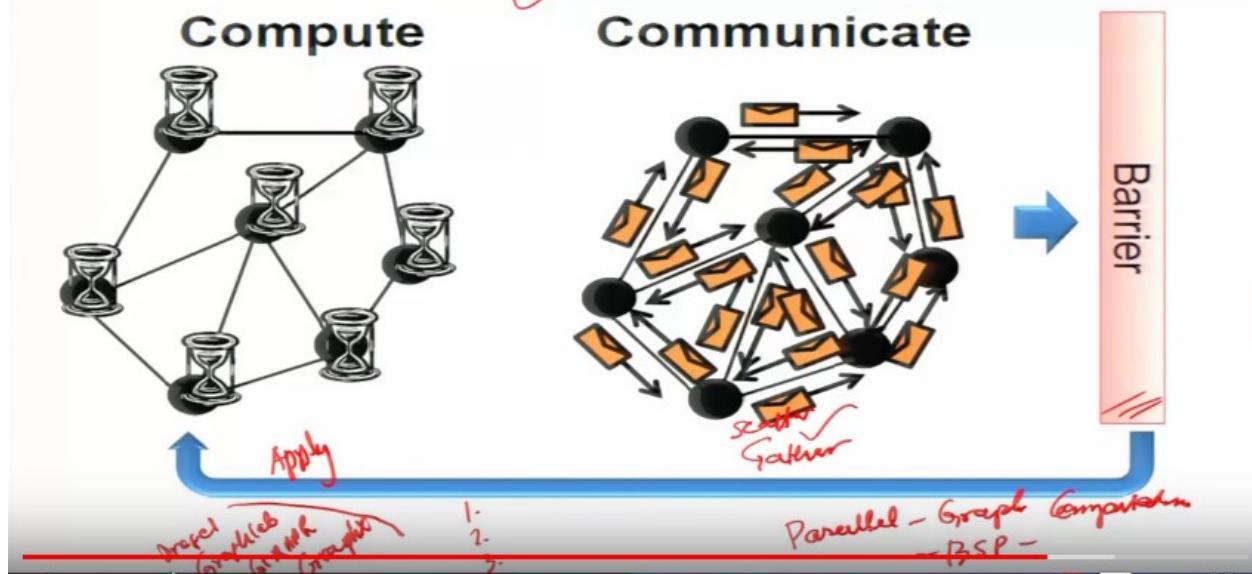


- Data movement is managed by the system and not the user.

Similarly in a graph lab, there will be a full abstraction. So, same algorithm of page rank, if we can write down in a graph or lab using pool abstraction. So, it will become a vertex programmed directly access the adjacent, vertices and edges. So, Graph Lab PageRank, for the node 'i' will can, can also be seen as it will. Now compute the sum over the neighbors, using this particular iterations and that means it will collect the information, compute the this one compute, the page ranks out of the neighboring, pages information, using this particular formula and then it will update, its PageRank and that is all. So, here the data movement is managed by the system and automatically it will be handled. So, programmer, the user has to write down for these vertex programs and restore all it will be taken care. So, that means when these, data from the neighbors are required, in the programming construct, automatically the pool abstraction, will get these values, from the neighboring node and that is the part of internal parts, of the graph lamp.

Refer slide time :( 32:39)

## Iterative Bulk Synchronous Execution



Now in short what we can what we have seen these graph frameworks they provide, the features in which you can communicate, with the neighboring nodes, neighboring vertices and after the communication is reached, which being obtained or is reached, from all the neighbors then there will be a barrier and then the computation, will be performed and then it will then again communicate. So, this is the paradigm which is called an, ‘Iterative Bulk Synchronous Execution’, which will provide the parallel, graph computations, this particular diagram, shows the BSP a bulk synchronous execution, of the graph computations. So, this you can understand that when you say communicate, you can gather and when you compute then you say gather apply and again when you communicate it will become I scatter. So, again I told you that all these operations, are have different names and paradigms, in different the graph frameworks, let us say pregel and then the graph lab and zero I GraphX. So, all these are performing the same set, of basic operations, to support the iterative bulk synchronous execution. And using that the vertex programs can be written, in a very simplified manner, for different graph algorithms.

Refer slide time :( 34:38)

# Graph-Parallel Systems

Pregel  
oozie



GraphLab

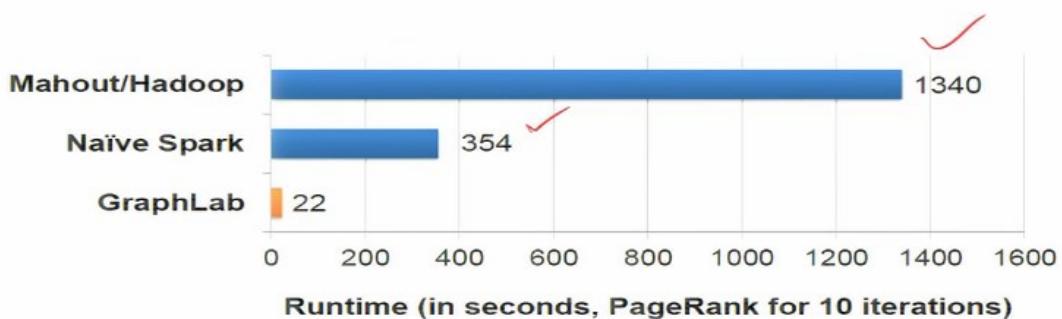
*Expose specialized APIs to simplify graph programming.*

*Exploit graph structure to achieve orders-of-magnitude performance gains over more general data-parallel systems.*

This we have already seen that using pregel or using zero or using graph lab, it will expose, the specialized API is to simplify, the graph programming. And exploit the graph structure, to achieve orders of magnitude performance grain or the more journal data parallel systems.

Refer slide time :( 34:59)

## PageRank on the Live-Journal Graph

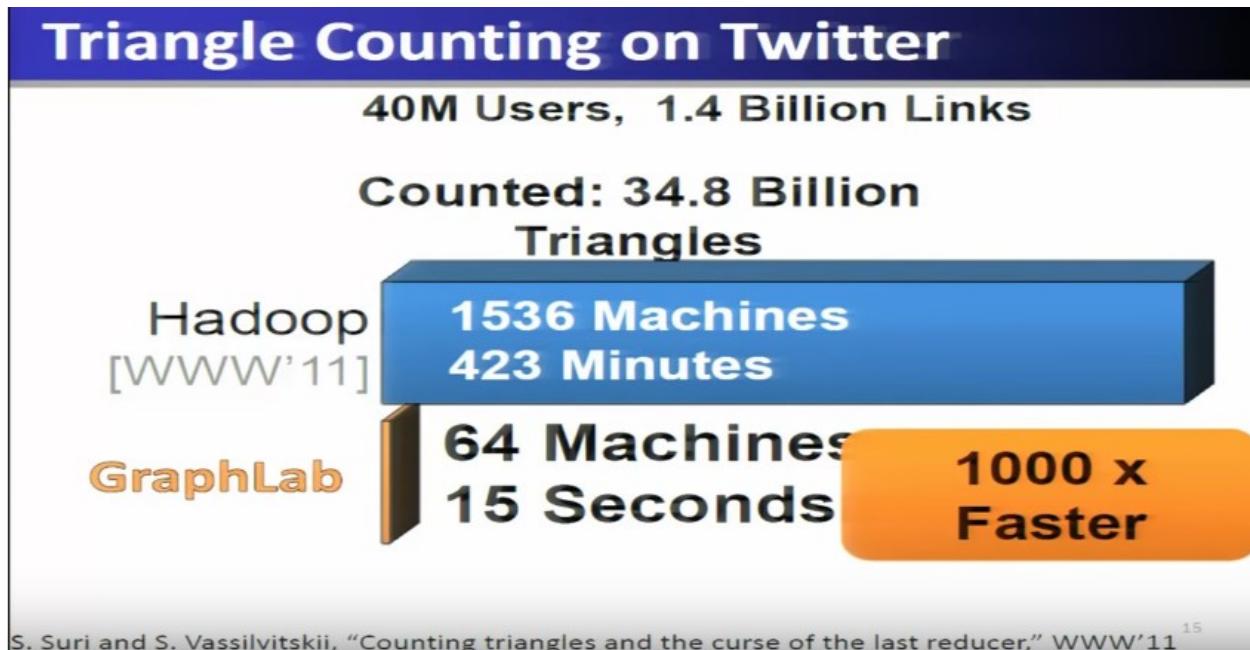


GraphLab is *60x faster* than Hadoop  
GraphLab is *16x faster* than Spark

So, for example the PageRank on Live Journal graph, if you see that, you see that if it is done through the Hadoop, it is taking 1, 3, 4, 0 of this particular time, compared to the Naive spark, if you use it will take 3

43:54 and graph lab is 22 and even the this one, even if you see the GraphX, which is even, faster than the graph lab also. So, we have seen how this particular improvement, in the performance, is gained is because, these basic operations and the algorithm, are optimized and being supported, by the core spark and optimized, by the GraphX for this algorithm. So, those algorithms, which are part of the libraries, of GraphX, are much more efficient, in the performance wise compared to any other framework.

Refer slide time :( 36:11)

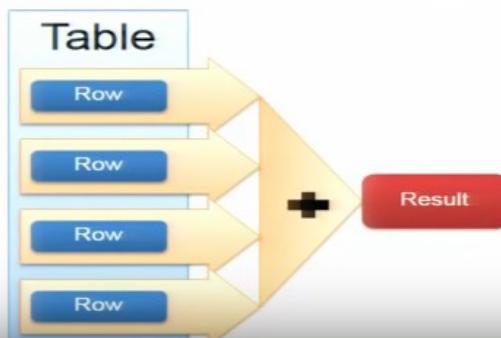


Similarly the triangle counting and if you see that it is 1000 times faster in GraphX.

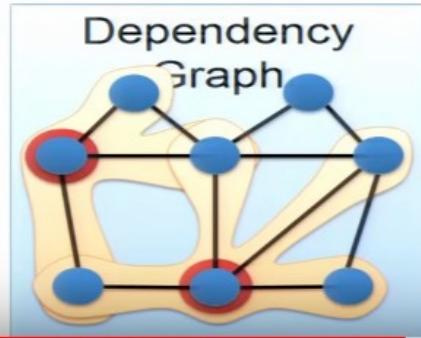
Refer slide time :( 36:20)

## Separate Systems to Support Each View

### Table View



### Graph View



▶ ▶ ⏴ 37:30 / 41:15 Big Data Computing

Graph CC HD □ □ □

Let us see this we have seen now. So, basically now, we have to go and see, how these graph systems, are being supported, for their implementation. Now as far as these data, we have seen that the data can be easily represented, in a form of a table and that particular, view is called a, 'Table View' . And the computation can be easily performed, on this view that is called 'Table View', to get the results similarly there is another, view that from this table view we can get we can convert in the form of a graph. So, it will become a graph view and then various algorithms can be applied, with the help of pregel APIs graph lab. And Giraph so, this particular graph, will be useful for applying, different graph algorithms, to get the results back. So, we will see that there will be two different views, from table view, we are converting to graph view and then again into a table view. So, there is a dependencies, between table view and a graph view. Let us see how the different framework, they are handling this kind of views.

Refer slide time :( 37:40)

Having separate systems  
for each view is  
**difficult to use** and  
**inefficient**

So, having separate views, for each view is difficult to use.

Refer slide time :( 37:46)

## Difficult to Program and Use

Users must *Learn, Deploy, and Manage*  
multiple systems



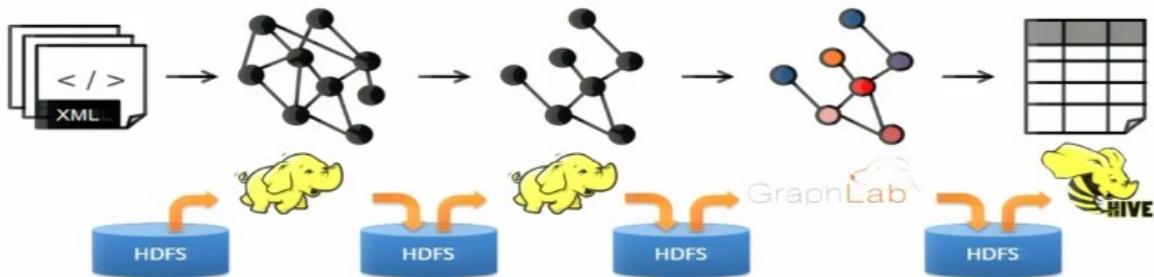
Leads to brittle and often  
complex interfaces

And also will become inefficient.

Refer slide time :( 37:47)

## Inefficient

Extensive **data movement** and **duplication** across the network and file system



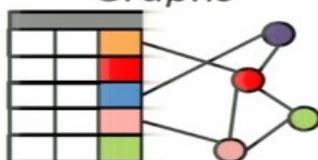
So, why because the data will be moving and also that there will be a duplication, into the network and the file system, therefore the this particular changing of views, from table view to the graph, view and again back to the table view, requires the data to be moved, through the HDFS file system hence, it will become inefficient in most of the scenarios.

Refer slide time :( 38:18)

## Solution: The GraphX Unified Approach

### New API

*Blurs the distinction  
between Tables and  
Graphs*

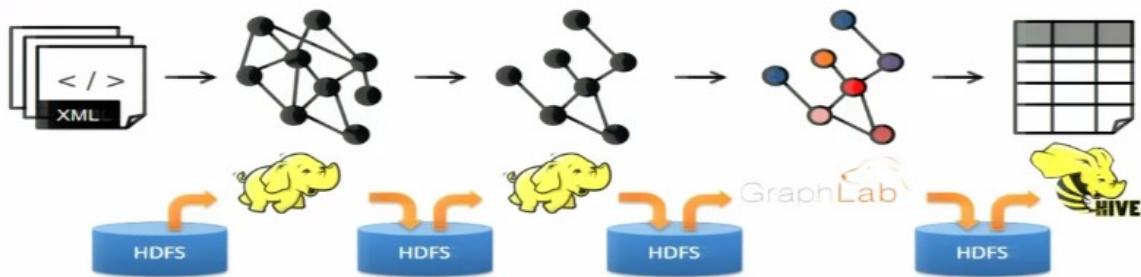


Now as far as the GraphX why we are understanding, about GraphX is that it will give a unified, approach and it will not incur these inefficiencies,

Refer slide time :( 38:31)

## Inefficient

Extensive **data movement** and **duplication** across the network and file system



Limited reuse internal data-structures across stages

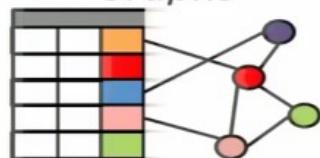
of internal are basically, the using use of file system and the network to store the data in between.

Refer slide time :( 38:38)

## Solution: The GraphX Unified Approach

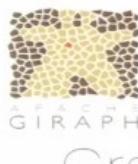
### New API ✓

*Blurs the distinction between Tables and Graphs*



### New System ✓

*Combines Data-Parallel Graph-Parallel Systems*

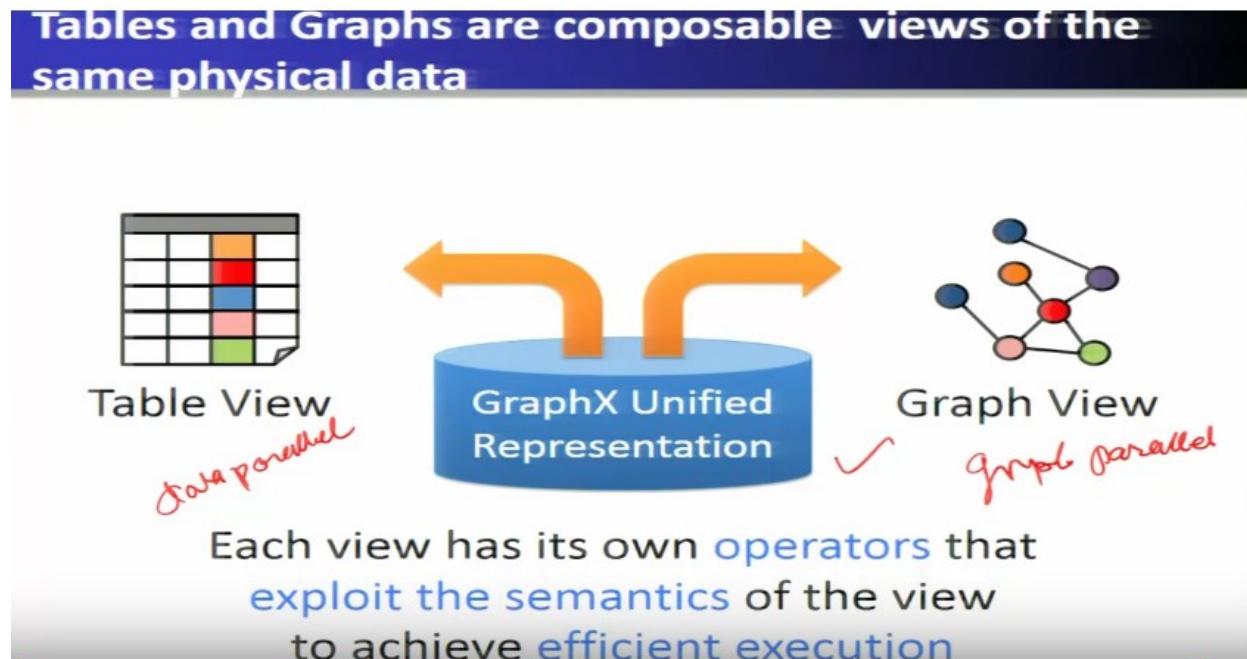


Enabling users to **easily** and **efficiently** express the entire graph analytics pipeline

Hence it is called a unified view. So, GraphX provides a unified approach, wherein the data, will not be required to store, in SDFS. So, between different views, automatically they are being integrated together and data can remain in memory. So, all the operations are supported, with full efficiencies. So, therefore this new API is which are supported in the form, of the graph X will, will blur the distinction between the

tables and the graph. So, the new system, which will now, evolve in this in the form of a graph X as the unified approach will combine the data parallel and graph parallel systems together. So, this will enable the users to easily and efficiently express the entire, graph analytics pipeline that we have that we are going to see.

Refer slide time :( 39:36)



So, therefore the graphs and the tables are composable views of the same physical data. And this GraphX will provide a unified representation. So, each view has its own operators that exploit the semantics of the view to achieve the efficient execution. So, table and the graph both views are together unified and is being supported in the GraphX. So, therefore data parallel. And the draw parallel operations, are integrated together, iron is being supported, as unified view.

Refer slide time :( 39:20)

## Graphs → Relational Algebra

1. Encode graphs as distributed tables
  2. Express graph computation in relational algebra
  3. Recast graph systems optimizations as:
    1. Distributed join optimization
    2. Incremental materialized maintenance
- 

Integrate Graph and  
Table data  
processing systems.

Achieve performance  
parity with  
specialized systems.

For better efficiency, graphs can also be represented or map in a form of relational algebra and therefore, we can encode the graph, as distributed tables and Express the graph computation, in the relational algebra recast the graph systems, optimizations as distributed joint, optimization and incremental, material maintenance.

INDIAN INSTITUTE OF TECHNOLOGY PATNA

NPTEL

NATIONAL PROGRAMME ON TECHNOLOGY ENHANCED LEARNING

COURSE TITLE  
BIG DATA COMPUTING

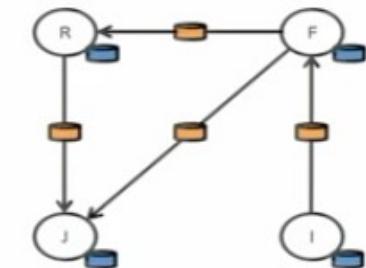
LECTURE-33  
SPARK GRAPH-X AND GRAPH ANALYTICS  
PART-II

BY  
DR. RAJIV MISRA  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY PATNA

(Refer Slide Time: 00:16)

## View a Property Graph as a Table

### Property Graph



Vertex Property Table

| Id       | Property (V)    |
|----------|-----------------|
| Rxin     | (Stu., Berk.)   |
| Jegonzal | (PstDoc, Berk.) |
| Franklin | (Prof., Berk)   |
| Istoica  | (Prof., Berk)   |

Edge Property Table

| SrcId    | DstId    | Property (E) |
|----------|----------|--------------|
| rxin     | jegonzal | Friend       |
| franklin | rxin     | Advisor      |
| istoica  | franklin | Coworker     |
| franklin | jegonzal | PI           |

### Big Data Computing

### GraphX

The graph in GraphX is represented as the property graph, so graph in a GraphX is called the property graph. Property graph has information such that every node has the vertex ID, and also it has the properties. This particular vertex has two informations, one is called vertex ID, the other is called the property and both this information are stored in a table which is called vertex property table.

Now similarly the edges have, so the edges have the ID's of both the ends, so for example in the directed edge it will be original, is the source ID and the destination ID, so this will be the edge ID's and then every edge will also contain the property with associated with that every

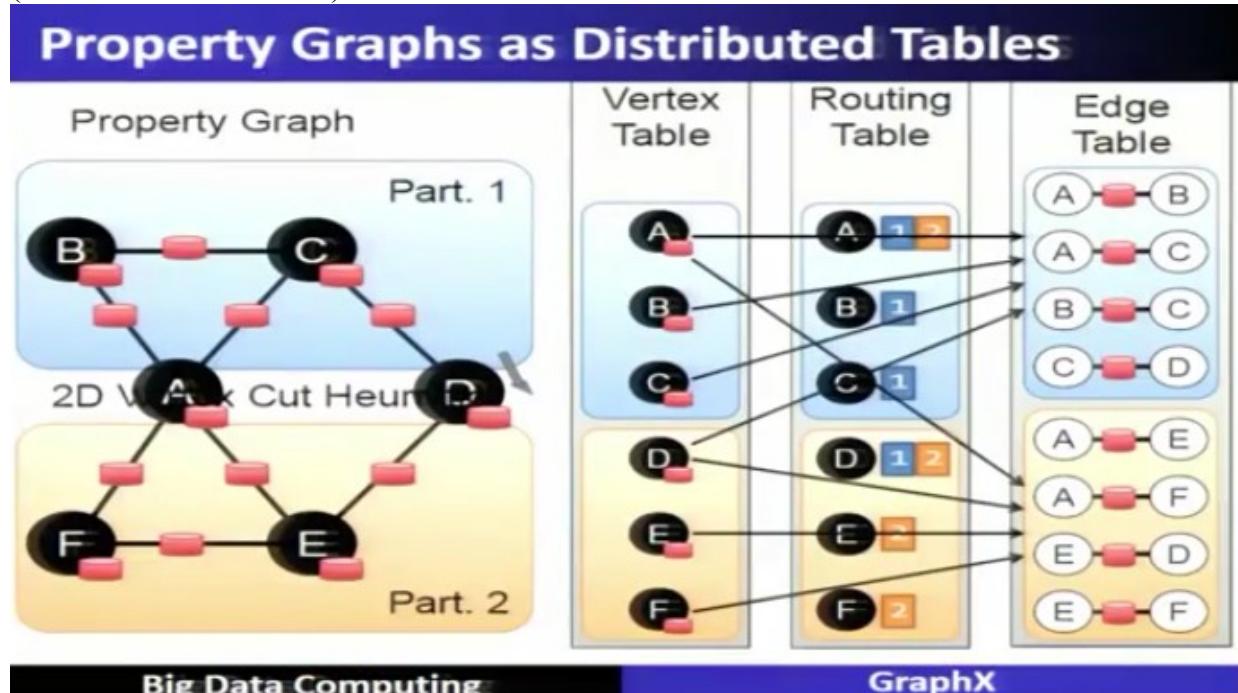
edge, so this information is stored in a table which is called edge property table, so we can see here that for example the vertex ID I and vertex ID F, so vertex ID I and vertex ID F, so vertex ID I will have this particular property that it is from, it is a professor and it is from Berkley. Similarly the vertex ID which is F, the name is Franklin or the ID is the F, so it is also a professor, but it is also working in the Berkley.

Now if you see this particular edge between I and F, so the edge between I and F is represented over here and it represents the property which is called coworker, so we have seen that the vertices have the properties, vertices have ID's and also a property and we can represent the vertices in a form of a table which is called vertex property table.

Similarly the edges are represented by source ID and definition ID and every edge also has a property and this edge property table contains the information about all the edges, so vertex property table and edge property table will form the representation of a property graph, so graph can be represented in a form of, a property graph can be represented in a form of the table property graph can be represented in a form of a table which are called as a vertex property table, and the edge property table, so the graph again I am repeating this aspect which is very important that graph in GraphX is called a property graph, and the property graph is represented in a form of a table which is nothing but vertex property table and edge property table.

So graph can be viewed, so as graph is represented as the property graph and it can be changed or it can be viewed as the table also, that is having the property table, vertex property table and edge property table.

(Refer Slide Time: 04:34)



So property graph is represented as the distributed table, so we can see here that if a graph which is called a property graph, if it is too big it cannot fit in 2 1 machine, then we can cut

across these vertex, and once it is cut across this vertex the first plot can be stored in one machine, the second part can be stored on the other machine.

So for example the corresponding vertex table and the edge table will be, can be stored in two different machines, so this is machine number 1, this is machine number 2, so A, B and C, so the upper part will be stored in, the first three node of the table will be stored in one machine, the another three nodes will be stored in other machine, similarly the edges, so all the edges which are in part 1 will be stored in the first edge table, similarly the remaining edges are stored in another table.

These vertices and edges, how they are communicating each other is communicating through each other via routing table, so routing table is also another table which is added in between, so there are three different tables which you will see here are to represent the graph as a form of a table, and this particular property graph will be represented in this way as the distributed tables, so vertex table, routing table and the edge table.

By routing table we mean that so the, for example the vertex A, vertex A is being referred by the edges in both the parts that is, it will be referred by A, it will be referred by B, it will be referred by F, so that means the vertex it will be referred in both the partitions, therefore it is represented as the routing table will contain the partitions where this vertex A will be referred.

And for example the node or a vertex E will be only referred in the second partition, hence the routing table for E will contain only 2, the second partition, this way the vertex table and edge table can be distributed and with the help of routing table together will be used as the distributed tables.

(Refer Slide Time: 07:25)

## Table Operators

- Table (RDD) operators are inherited from Spark:

|                |             |             |
|----------------|-------------|-------------|
| map            | reduce      | sample      |
| filter         | count       | take        |
| groupBy        | fold        | first       |
| sort           | reduceByKey | partitionBy |
| union          | groupByKey  | mapWith     |
| join           | cogroup     | pipe        |
| leftOuterJoin  | cross       | save        |
| rightOuterJoin | zip         | ...         |

There are different table operators which are inherited from the spark, that we can see here, (Refer Slide Time: 07:35)

## Graph Operators

```
class Graph [V, E] {
 def Graph(vertices: Table[(Id, V)],
 edges: Table[(Id, Id, E)]) {

 def vertices: Table[(Id, V)] = vertices
 def edges: Table[(Id, Id, E)] = edges
 def triplets: Table[((Id, V), (Id, V), E)] = edges

 def reverse: Graph = ...
 def subgraphByV: Table[(Id, V)] = ...
 def mapV(m: Table[(Id, V)] => Table[(Id, V)]) = ...
 def mapE(m: Table[(Id, Id, E)] => Table[(Id, Id, E)]) = ...

 def joinVbyE: Table[((Id, V), (Id, Id, E))] = ...
 def joinEbyV: Table[((Id, Id, E), (Id, V))] = ...

 def mTriplets(mapF: Table[(Id, V)] => Table[(Id, V)]) = ...
 }
}
```

**Big Data Computing**

**GraphX**

let us see the graph operators which are supported here in GraphX, so graph is represented as a class graph vertex and edges where the graph vertices is nothing but a having a table as off ID's and vertices,

(Refer Slide Time: 07:54)

## Graph Operators

```
class Graph [V, E] {
 def Graph(vertices: Table[(Id, V)],
 edges: Table[(Id, Id, E)]) {

 // Table Views -----
 def vertices: Table[(Id, V)] = ...
 def edges: Table[(Id, Id, E)] = ...
 def triplets: Table[((Id, V), (Id, V), E)] = ...

 def reverse: Graph = ...
 def subgraphByV: Table[(Id, V)] = ...
 def mapV(m: Table[(Id, V)] => Table[(Id, V)]) = ...
 def mapE(m: Table[(Id, Id, E)] => Table[(Id, Id, E)]) = ...

 def joinVbyE: Table[((Id, V), (Id, Id, E))] = ...
 def joinEbyV: Table[((Id, Id, E), (Id, V))] = ...

 def mTriplets(mapF: Table[(Id, V)] => Table[(Id, V)]) = ...
 }
}
```

**Big Data Computing**

**GraphX**

and edges are also having the table of source vertices and definition vertices and the edges, properties, so this is the representation of a graph, so graph is represented as the vertices and the edges, whereas the vertices are represented as the vertex property table, and edge is represented as the edge property table, and these are represented together as the graph, so graph here is

represented as the vertex property table, edge property table, and which is shown in this particular representation of the graph.

Now having defined this particular graph in this manner, so you can also see the graph as table view, so the vertices is represented by this table that is called a vertex property table, edge is represented by the edge property table, and also there will be a triplet view of a particular graph, that means so triplet view of a graph is also triplet view table can be represented.

(Refer Slide Time: 09:24)

## Graph Operators

```

class Graph [V, E] {
 def Graph(vertices: Table[(Id, V)], edges: Table[(Id, Id, E)]) { } // G (Vertex properties, Edge properties)
 // Table views -----
 def vertices: Table[(Id, V)] { }
 def edges: Table[(Id, Id, E)] { }
 def triplets: Table[((Id, V), (Id, V), E)] { }
 // Transformations -----
 def reverse: Graph[V, E]
 def subgraph(pV: (Id, V) => Boolean,
 pE: Edge[V, E] => Boolean): Graph[V, E]
 def mapV(m: (Id, V) => T): Graph[T, E]
 def mapE(m: Edge[V, E] => T): Graph[V, T]

 def joinV[T](t: Table[(Id, T)]) { }
 def joinE[T](t: Table[(Id, Id, T)]) { }

 def mapTriplets(m: ((Id, V), (Id, V), E) => T): Graph[V, T]
}

```

Where various transformations in the graph which are supported here in the GraphX is called the reverse of, then subgraph, finding the subgraph and then taking the map of this particular graph, the joint operations are also performed on the graph,  
(Prof. Shail Tiwari, 2012)

(Refer Slide Time: 09:42)

# Graph Operators

```
class Graph [V, E] {
 def Graph(vertices: Table[(Id, V)],
 edges: Table[(Id, Id, E)]) {
 // Table Views
 def vertices: Table[(Id, V)] = vertices
 def edges: Table[(Id, Id, E)] = edges
 def triplets: Table [((Id, V), (Id, V), E)] = vertices
 .map((v, e) => (v, v, e))
 .join(edges)
 .map((v, e) => (v, e, e))
 // Transformations
 def reverse: Graph[V, E] = edges
 .map((e, v) => (v, e, e))
 .join(vertices)
 .map((v, e) => (v, e, e))
 def subgraph(pV: (Id, V) => Boolean,
 pE: Edge[V, E] => Boolean): Graph[V, E] =
 edges
 .filter(pE)
 .join(vertices)
 .filter(pV)
 .map((v, e) => (v, e, e))
 def mapV(m: (Id, V) => T): Graph[T, E] =
 vertices
 .map(m)
 .join(edges)
 .map((v, e) => (v, e, e))
 def mapE(m: Edge[V, E] => T): Graph[V, T] =
 edges
 .map(m)
 .join(vertices)
 .map((v, e) => (v, e, e))
 // Joins
 def joinV(tbl: Table [(Id, T)]): Graph[(V, T), E] =
 vertices
 .join(tbl)
 .map((v, t) => (v, v, t))
 def joinE(tbl: Table [(Id, Id, T)]): Graph[V, (E, T)] =
 edges
 .join(tbl)
 .map((e, t) => (e, e, t))
 // Computation
 def mrTriplets(mapF: (Edge[V, E]) => List[(Id, T)],
 reduceF: (T, T) => T): Graph[T, E] =
 edges
 .map((e, v) => mapF(e))
 .reduce(reduceF)
}
```

Big Data Computing

GraphX

and the computations are performed in the graph.

(Refer Slide Time: 09:47)

## Triplets Join Vertices and Edges

- The *triplets* operator joins vertices and edges:



Big Data Computing

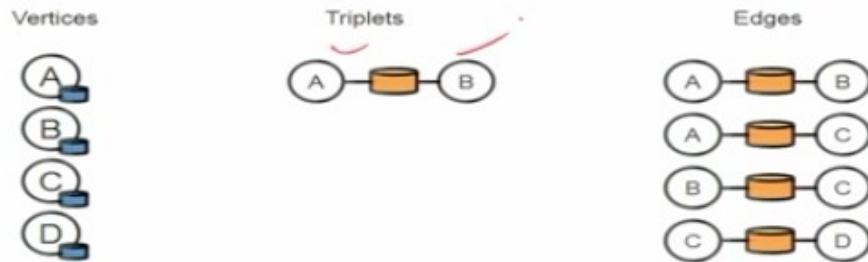
GraphX

Now the triplet join vertices and edges, so triplet operator joins the vertices and the edges, so if this is the edge between A and B,

(Refer Slide Time: 10:03)

## Triplets Join Vertices and Edges

- The *triplets* operator joins vertices and edges:



### Big Data Computing

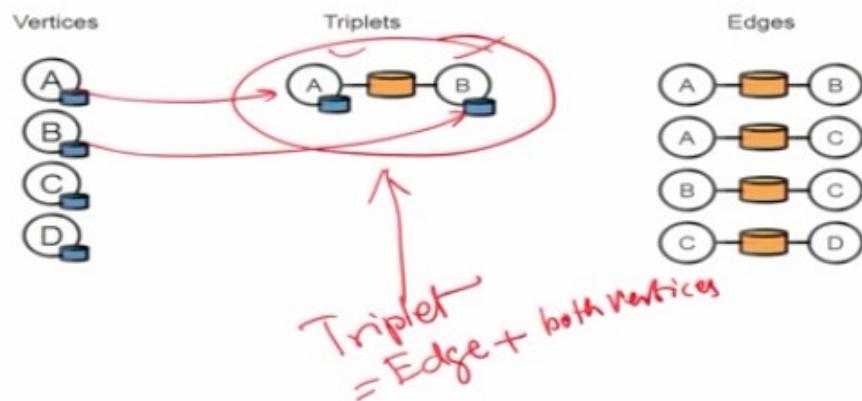
### GraphX

so it will also combine with both the vertices A and B along with this edge is called a triplet, so triplet means that, is that edge + both operates vertices is called the triplet.

So we can see here that this edge will have the property A also, property of vertex A,  
(Refer Slide Time: 10:37)

## Triplets Join Vertices and Edges

- The *triplets* operator joins vertices and edges:



### Big Data Computing

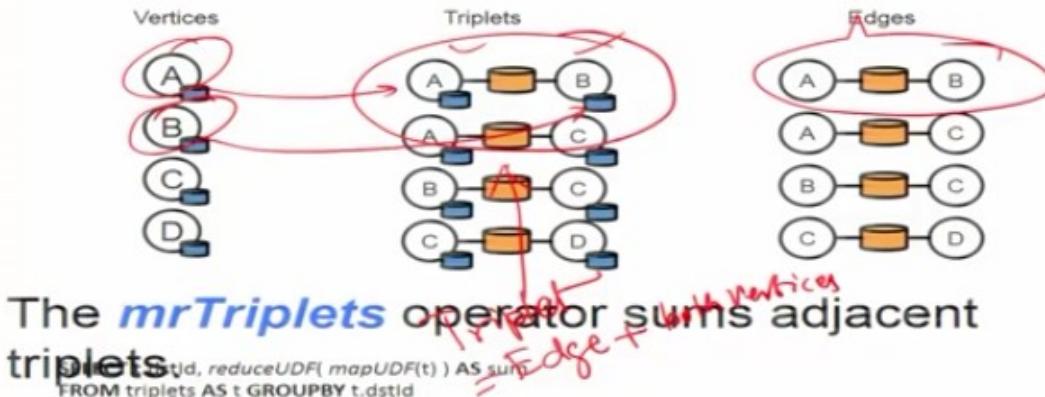
### GraphX

and this edge also has the property of vertex B as well as it has the property of the edge A, B. Together all three things is called the triplet. Triplet is supported in the GraphX and is very much useful in building the graph algorithms.

(Refer Slide Time: 10:55)

## Triplets Join Vertices and Edges

- The *triplets* operator joins vertices and edges:



### Big Data Computing

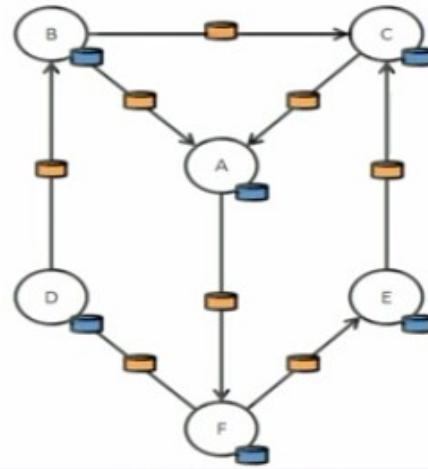
### GraphX

The map reduce triplet operator sums the adjacent triplets and which will be used in various graph algorithms,

(Refer Slide Time: 11:02)

## Map Reduce Triplets

- Map-Reduce for each vertex



### Big Data Computing

### GraphX

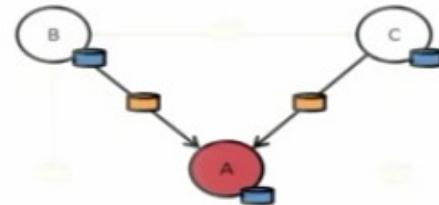
let us see that how map-reduce using triplets can solve some of the important, some of the problems of a graph analytics, so for example this vertex A

(Refer Slide Time: 11:20)

## Map Reduce Triplets

- Map-Reduce for each vertex

mapF( )  
mapF( )



### Big Data Computing

### GraphX

want to do the computation, it requires the information from the neighboring vertices and using the map function on this triplet you can output, you can emit the A1, similarly for the map function between A and C, it will output A, and then reduce will combine them together  
(Refer Slide Time: 11:48)

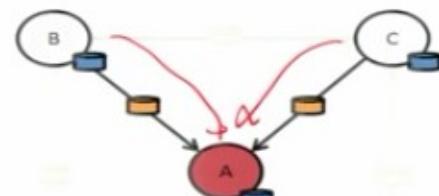
## Map Reduce Triplets

- Map-Reduce for each vertex

mapF( ) → 

mapF( ) → 

reduceF  → 



### Big Data Computing

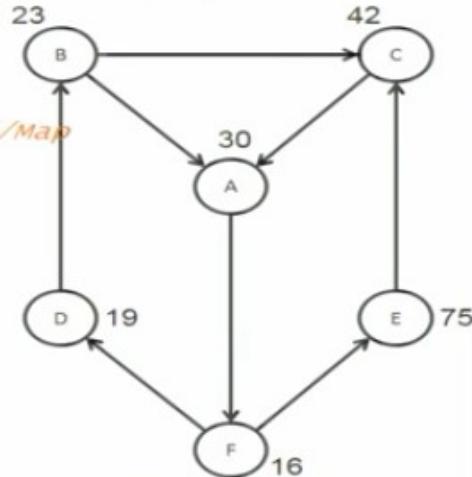
### GraphX

this particular view, so once the triplet view is there then map function and reduce function can become very efficiently programmable in building the algorithms.

(Refer Slide Time: 12:00)

## Example: Oldest Follower

- What is the age of the oldest follower for each user?
- ```
val oldestFollowerAge = graph
  .mrTriplets(
    e=> (e.dst.id, e.src.age), //Map
    (a,b)=> max(a, b) //Reduce
  )
  .vertices
```



Big Data Computing

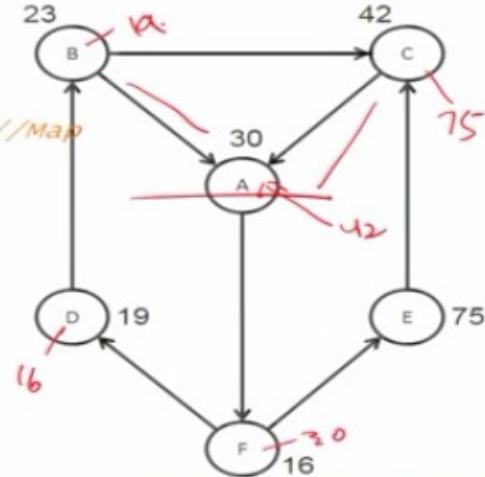
GraphX

Let us see that how using this triplet view we can write an algorithm or a program for finding the oldest follower, so the question is what is the age of the oldest follower for each of these users? So let us see a simple program which can solve this particular problem and, so you can see here that in this particular example, for example the node A is followed by C and, C and B, now let us see that it says that what is the age of the oldest follower of each user, so the follower of A, the oldest follower of A is 42, similarly you can find out the follower of C which is 75 and 23, this is 75 and then you can find out the follower of 16 is 30, and follower of 19 which is 16, and follower of D is basically 19,

(Refer Slide Time: 13:18)

Example: Oldest Follower

- What is the age of the oldest follower for each user?
- ```
val oldestFollowerAge = graph
 .mrTriplets(
 e=> (e.dst.id, e.src.age), //Map
 (a,b)=> max(a, b) //Reduce
)
 .vertices
```



Big Data Computing

GraphX

so out of these different followers you can find out the maximum that is the answer will be, the C with the values 75, now this you can program using the triplet in a quite efficient manner, so you can find out using the graph triplet, using mr triplet which will say that the edges with the destination and source with the destination id E and the source id age will be now taken up as the maximum of all the triplets which are merging at any node, so you can see that this A will have this triplet, A will have another triplet, so it will find out the maximum of all the triplets a particular edge is having, and this way that will now then emit the age value and it will take the maximum in the reduce function, and this way you can see that using triplet it will be very simple program to find out these details.

(Refer Slide Time: 14:30)

---

## GraphX System Design

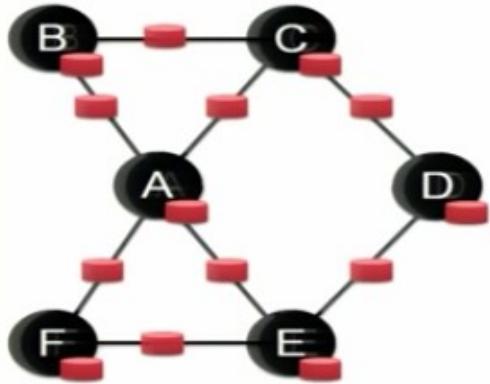
**Big Data Computing:**

**GraphX**

Let us see the GraphX system design details,  
(Refer Slide Time: 14:38)

## Distributed Graphs as Tables (RDDs)

Property Graph



### Big Data Computing

now as we have seen that the graph is represented as the property graph, and which is nothing but the distributed tables, and here these different aspects that is the vertices, edges and property tables they are represented in the form of the RDD's.

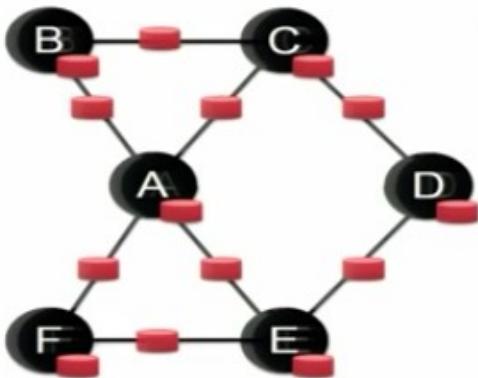
Let us see how this distributed graph is internally managed in GraphX, why distributed graph is required because the graph is very large, is called large scale graphs, now this large scale graph cannot be accommodated in one computer system, therefore it will be stored in a distributed manner, hence it is called a distributed graph.

If you say it is a distributed graph that means the graph can be cut into different pieces and being stored across the cluster of machines, how that is all done, we are going to see how the GraphX will support this distributed graph.

(Refer Slide Time: 15:54)

## Distributed Graphs as Tables (RDDs)

Property Graph



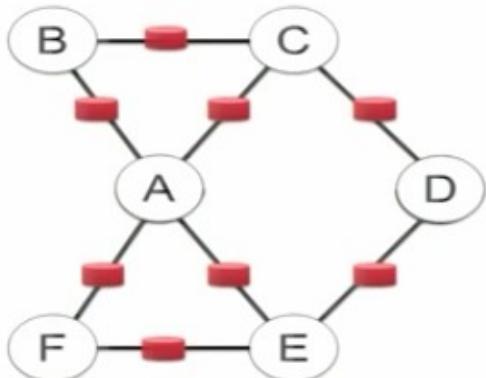
Large Scale Graph  
Can't fit in one system  
→ Distributed Graph  
GraphX

Big Data Computing

So this is the example of a small property graph, let us see how it is done in the GraphX,  
(Refer Slide Time: 16:05)

## Distributed Graphs as Tables (RDDs)

Property Graph



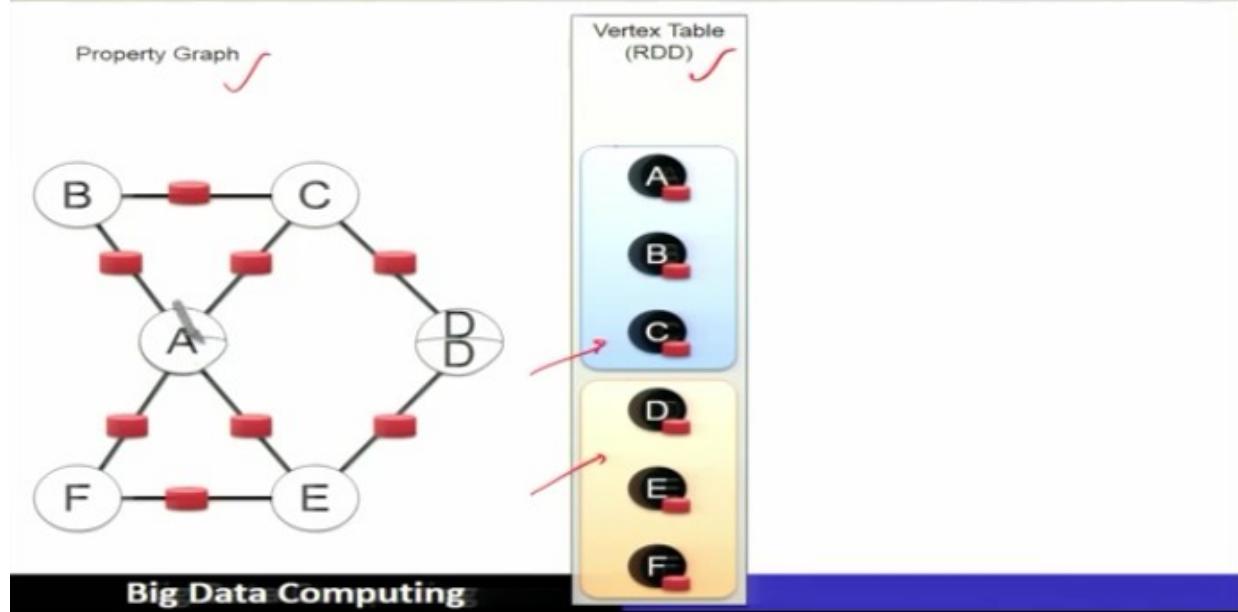
Large Scale Graph  
Can't fit in one system  
→ Distributed Graph  
GraphX

Big Data Computing

so the graph comprises of two table which is called vertex, now this particular GraphX which is this particular graph which is called a property graph has the vertex property table and the edge property table, now vertex property table will be represented as the distributed table of this vertex table, so this is called vertex table RDD's which are supported in the GraphX, so the vertices, all the vertices of a graph let us say that it's a millions of vertices so it cannot be accommodated in one table, in one computer system so obviously several computer systems are required.

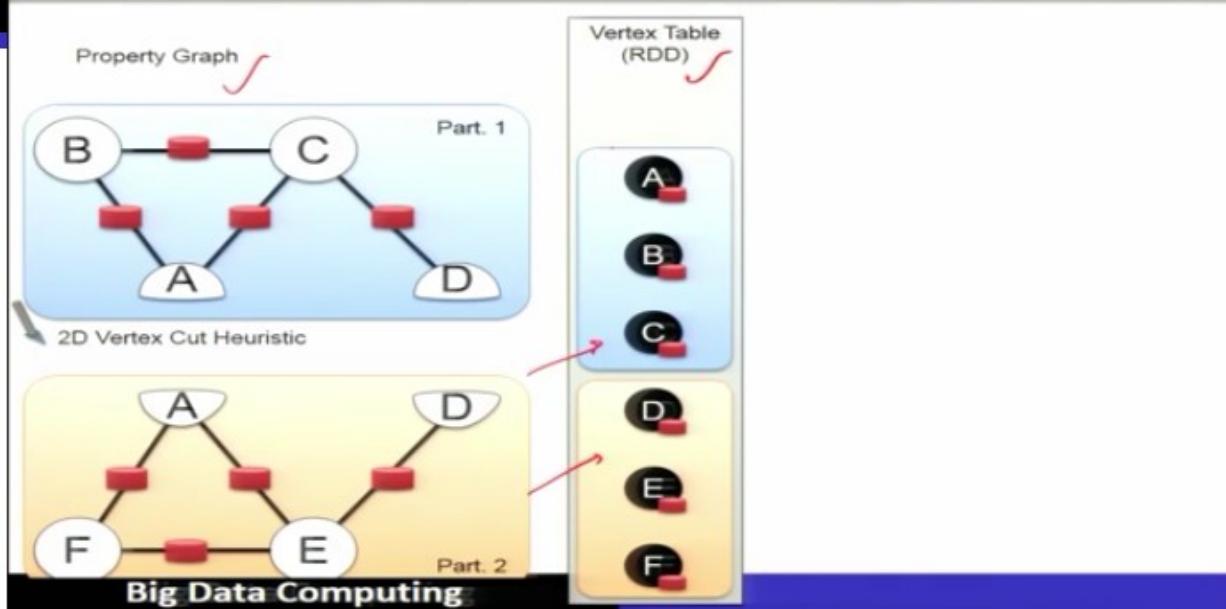
Let us understand that two computer systems is good enough for this example to store the vertex table in the form of RDDs,  
(Refer Slide Time: 17:11)

## Distributed Graphs as Tables (RDDs)



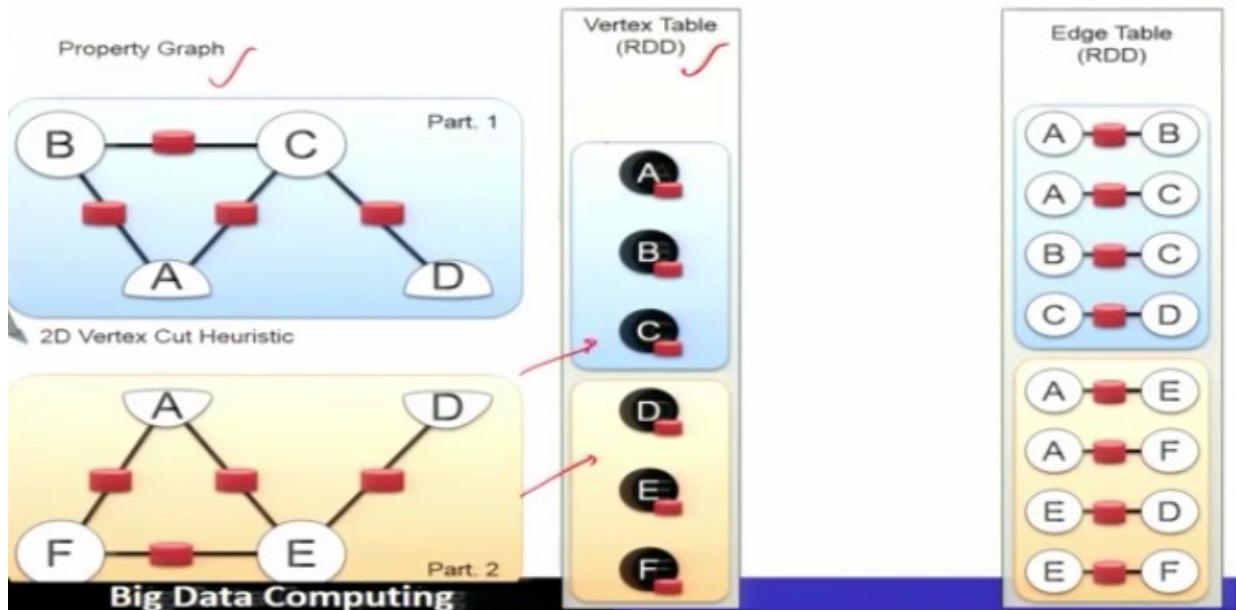
now as far as the graph is concerned we are going to cut this particular graph across the vertices  
(Refer Slide Time: 17:20)

## Distributed Graphs as Tables (RDDs)



and this particular way these part of the graphs are stored in as a distributed graphs.  
(Refer Slide Time: 17:30)

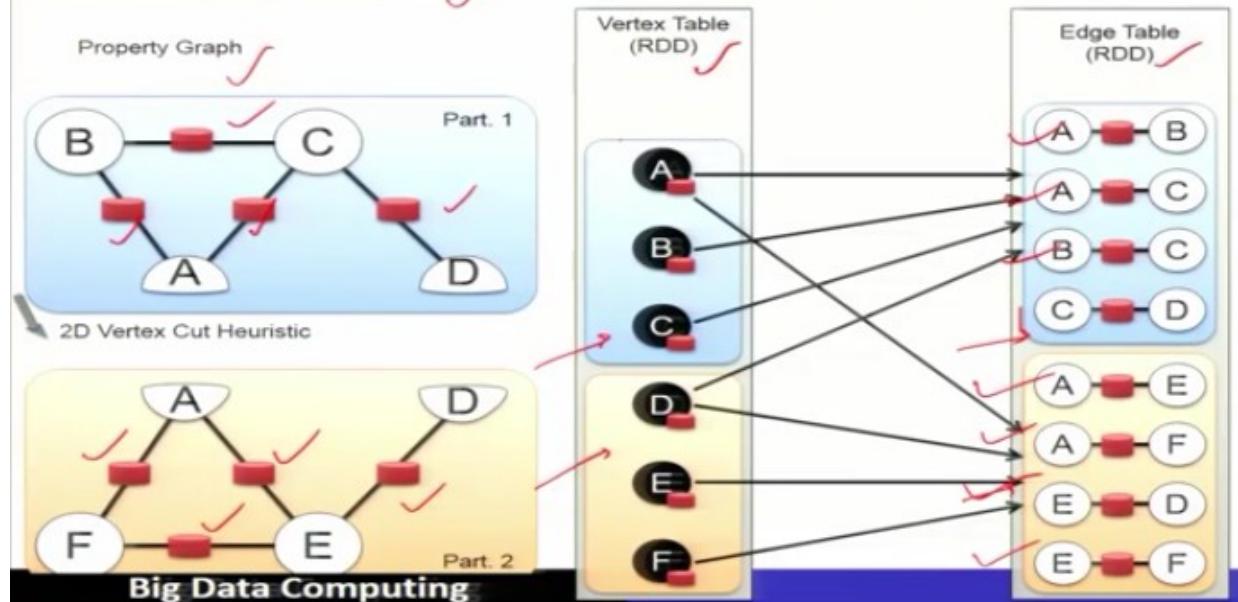
# Distributed Graphs as Tables (RDDs)



So let us see that once we cut the vertices, then there are two parts, part 1 and part 2, so part 1 and part 2 they are stored in different machines, so obviously the edge part that is called edge property table which is nothing but edge table RDD which is stored in 2 different machines like this, so the first part will have AB and then AC, the BC and then CD, so the first part is represented in the form of a edge table, similarly the second part which is called AF, then AE, the FE, and then ED, so all the edges of the second part is stored on the second node, so this way the vertices are also is split and stored in different nodes, edges are also stored on different nodes and this way it is called the distributed graphs which are stored in this way.

Now the thing is one over, there will be a communication between AB or AC or between AF and so how this is all facilitated, that we are going to see because all these vertices and edges they are stored on different nodes, and it is called distributed graph,  
 (Refer Slide Time: 18:59)

# Distributed Graphs as Tables (RDDs)

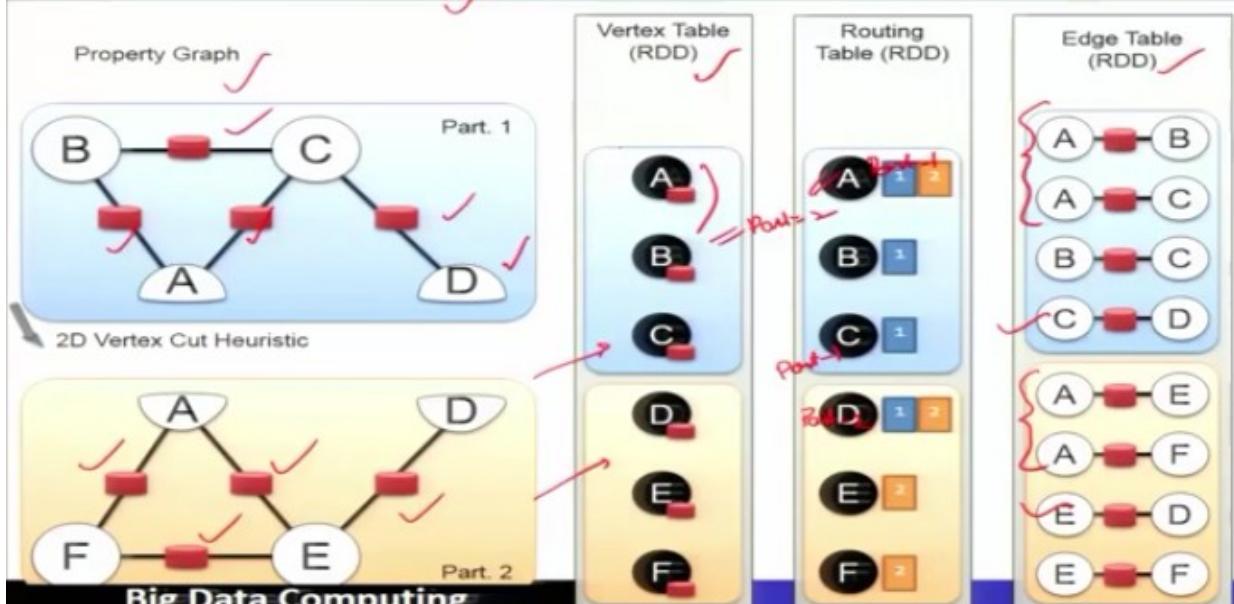


so for example A would like to communicate with both the nodes that is the part 1 and the part 2, both the parts will be stored on two different nodes, so this will be communicating with part 1 and the other communication is shown by the arrows as of the part 2, so A will be communicating to part 1 and A will be communicating to part 2, so to support these edges that is, to support these edges AE and AF, similarly A will communicate with part 1 to support the edges that is AB and AC.

Similarly as far as B is concerned, B can communicate with BC and BA, so BC and BA both are in part 1, so it does not require to be communicating with the part 2 in any of the case.

Similarly D is concerned, D will have the communication with C on part 1 and D will have communication with E on part 1, so D will have part 1, and D will have communication with part 2 also, so therefore CD will be part 1, and then DE will be on the part 2, so therefore sum of these vertices requires to communicate in all the parts where the edge table is being stored, and some of the nodes are required to be communicating with or single part,  
(Refer Slide Time: 20:53)

## Distributed Graphs as Tables (RDDs)

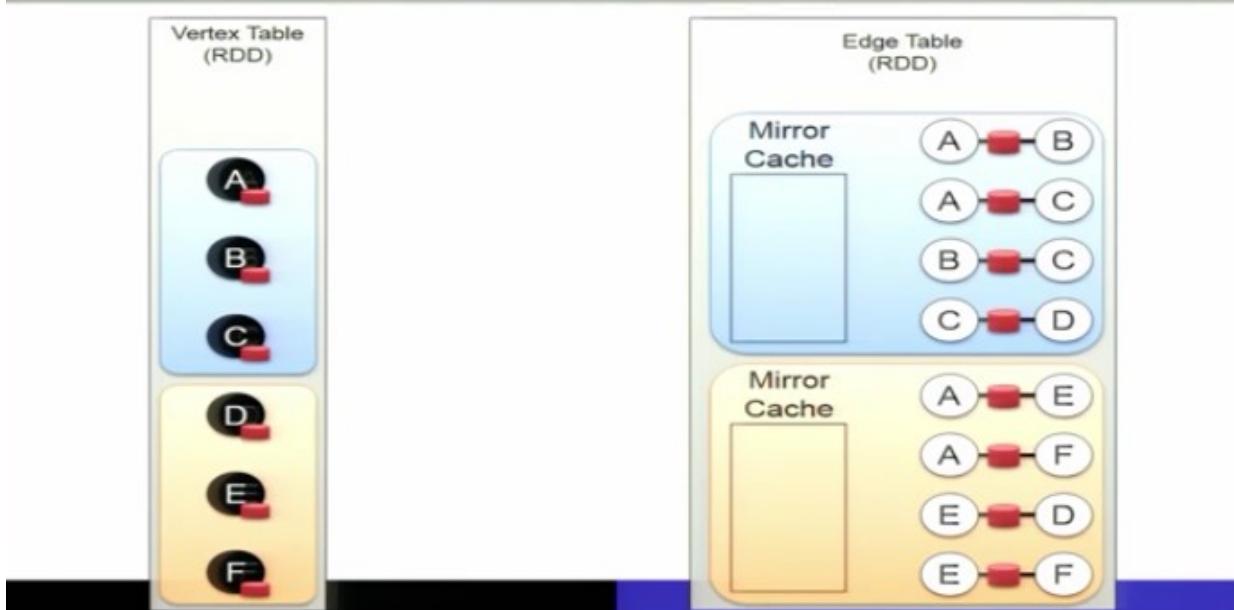


so therefore a routing table will capture this information and this is shown over here that for the node A it has to communicate with the part 1 and part 2, for B will has to communicate with part 1, C has to communicate with part 1, D has to communicate with both the parts, so also for E and F, so once the routing table capture this information therefore the interactions between the nodes and their corresponding edges can be facilitated using the routing table.

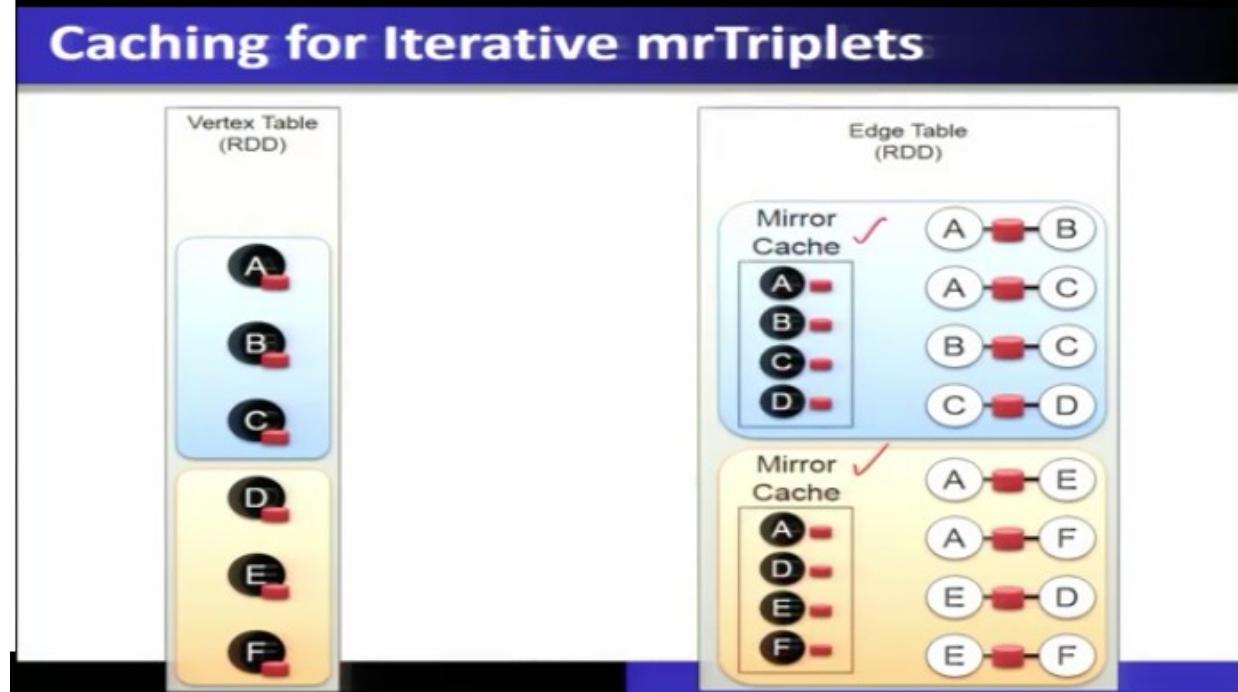
Now if we keep this information of the routing table part with the edge table also as a part of caching then this communication cost can also be reduced.

(Refer Slide Time: 21:45)

## Caching for Iterative mrTriplets

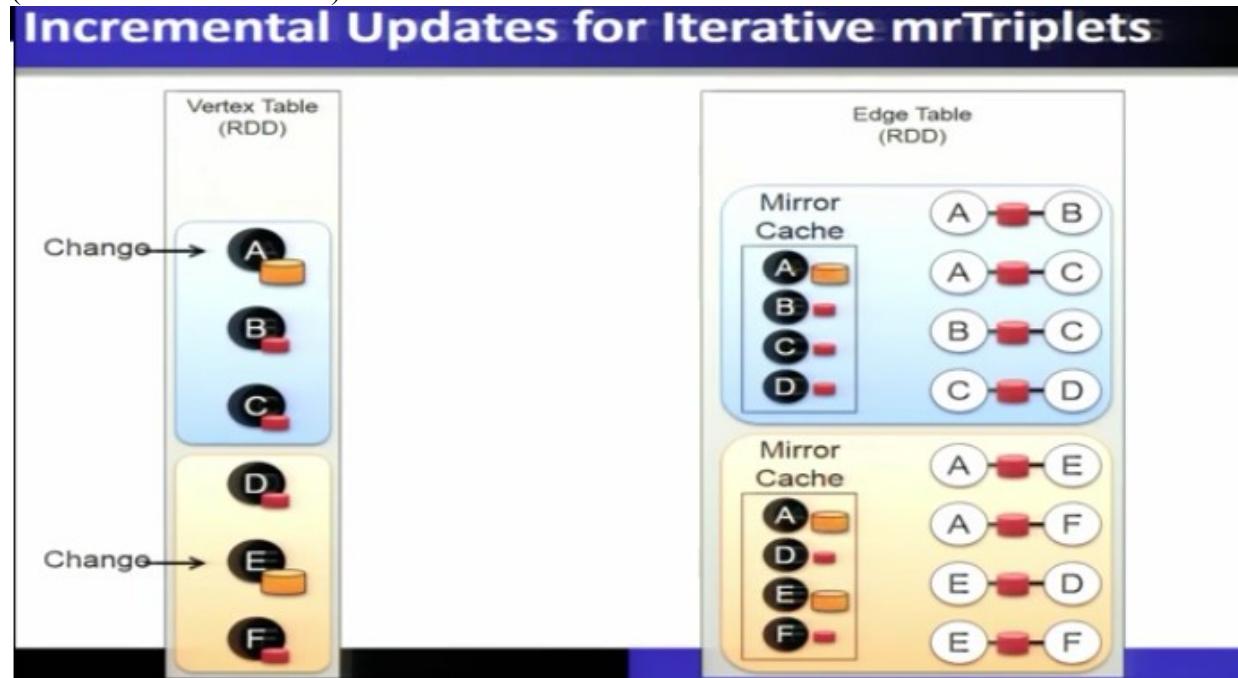


So here we can see here that the edge table will now contain the mirror cache of the routing table enter this,  
 (Refer Slide Time: 21:54)



and therefore the vertices which are required to communicate in the first portion, the first part will be stored in its mirror cache, similarly the nodes which are required to communicate on the second part will require to be stored in the mirror cache of that corresponding part.

Now whenever there is a change happening at the vertex table on those nodes,  
 (Refer Slide Time: 22:28)

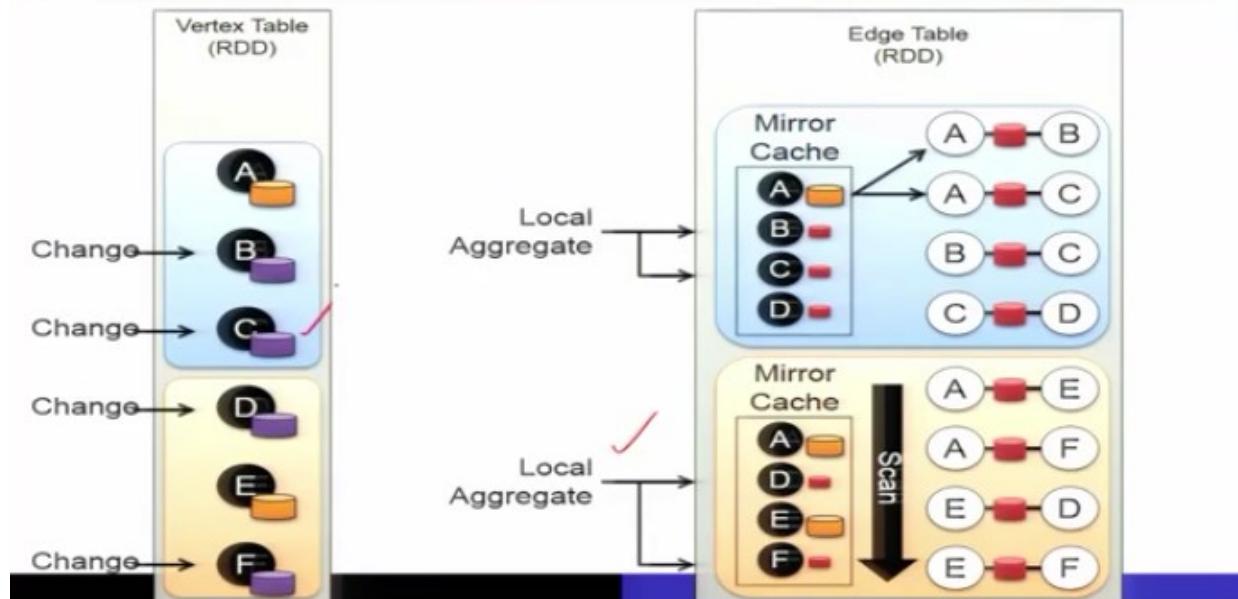


and these particular changes are to be communicated in the mirror cache, and this mirror cache in turn will propagate those changes further on the edge table.

Now as far as whenever this particular mirror cache will be now whenever the nodes are scanned and requires any changes in the mirror cache, so basically a local aggregation will be performed and this local aggregation will be communicated back to the vertex table, so therefore the number of communications are to be reduced here in this way of implementing the mirror cache and performing the local aggregates,

(Refer Slide Time: 23:17)

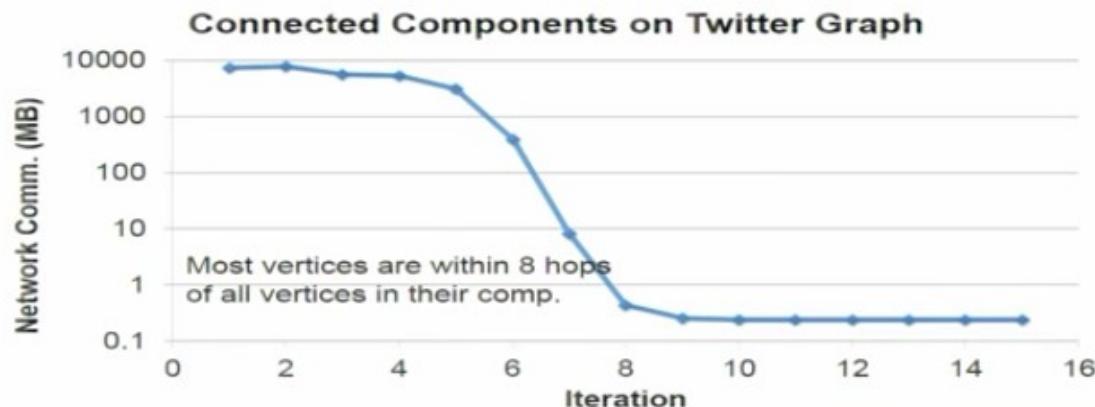
## Aggregation for Iterative mrTriplets



so whether it is in the vertex whenever some modifications are done, or at while the scanning of the edges, any computation or any changes are happening at this edges all this will be now aggregated, and the number of communications can be reduced here in this way, hence the performance can be increased.

(Refer Slide Time: 23:43)

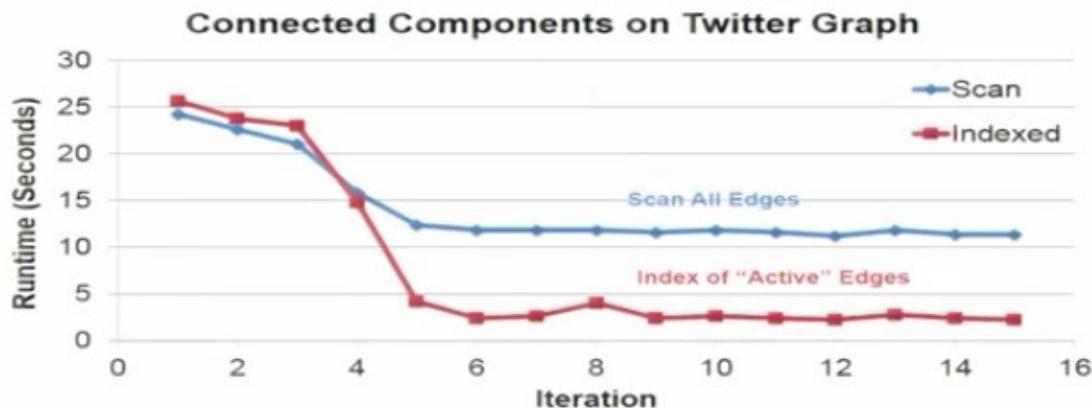
## Reduction in Communication Due to Cached Updates



Now if you measure how much reduction in the communication is done due to the cache updates, so we can see here that even for the connected components, computation on a twitter graph we can see that so many number of that means the communication is heavily dropped here in this particular case, therefore the performance as we have seen earlier in the previous slide that if it is graph or then Pregel or GraphLab, when you compare to the GraphX, so GraphX becomes a better performance in compared to any other graph framework, graph computation framework which are available due to these internal details of implementation using cache updates.

(Refer Slide Time: 24:38)

## Benefit of Indexing Active Edges



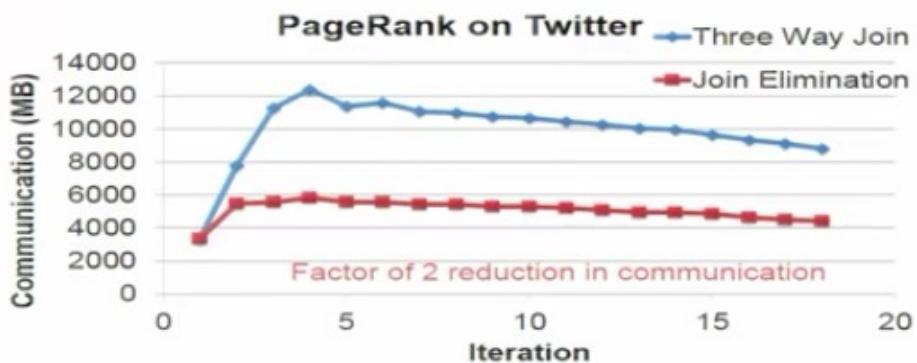
So these are, we can see that this wave of scanning the edges and aggregating the local updates and communicating with them is nothing but we are indexing, they are indexing the active edges, so the indexing of active edges again in turn will reduce the run time and, why because, these run times will be reduced because all the data is available in memory and they can be executed efficiently, and the effect is shown here in this particular graph, the run time and the execution time is heavily reduced, why because the indexes of active pages is being handled automatically internally has the time will be reduced.

(Refer Slide Time: 25:29)

## Join Elimination

Identify and bypass joins for unused triplets fields

- Example: PageRank only accesses source attribute



Similarly in this particular way we have eliminated the join, so identifying and bypass the join for unused triplet fields, so even for the PageRank only accesses the source attributes, so therefore the factor of two reduction in the communication due to the join elimination you can see the improvement in the communication is quite reduced,  
(Refer Slide Time: 26:01)

## Additional Query Optimizations

- Indexing and Bitmaps:
  - To accelerate joins across graphs
  - To efficiently construct sub-graphs
- Substantial Index and Data Reuse:
  - Reuse routing tables across graphs and sub-graphs
  - Reuse edge adjacency information and indices

### Big Data Computing

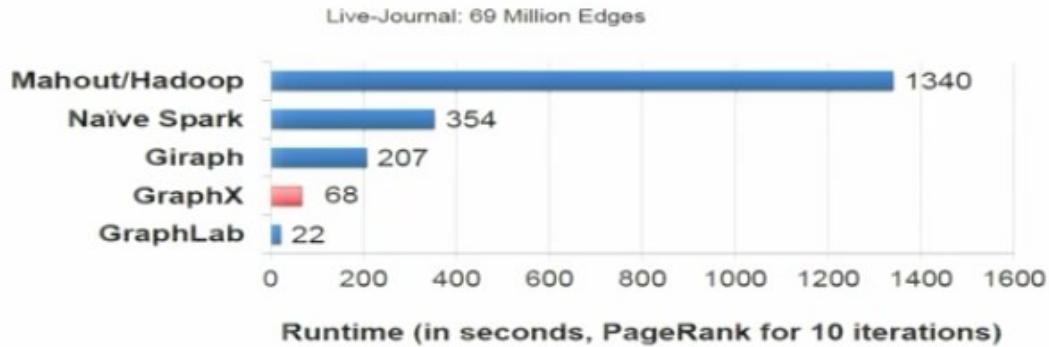
### GraphX

therefore there are additional query optimizations also available for indexing and bitmaps to accelerate joins across the graphs, to efficiently construct sub-graphs, substantial index and data reuse, so reuse routing tables across graphs and sub-graphs, reuse edge adjacency information and indices, so these are all different query optimizations which are possible so that it will accelerate either the joins across the graph or it can efficiently construct the sub-graph or using the index and data reuse, it can reuse the routing table across the graphs and also reuse the adjacency information and indices.

To summarize we can say that in this particular framework that is the GraphX, various optimizations are already in place and we can exploit using these optimizations to make more efficient algorithms on the graphs.

(Refer Slide Time: 27:18)

# Performance Comparisons



GraphX is roughly **3x slower** than GraphLab

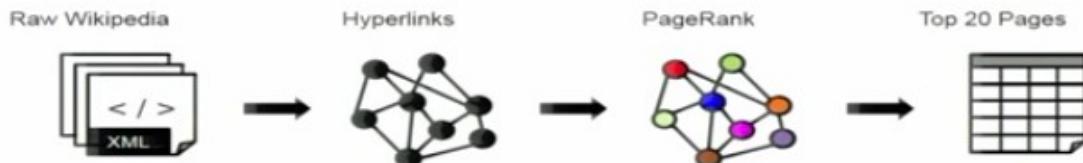
## Big Data Computing

## GraphX

So if you measure the performance we can see that the GraphX is roughly 3 x slower than the GraphLab, so therefore this large graph can be easily computable using the GraphX.

(Refer Slide Time: 27:41)

# A Small Pipeline in GraphX



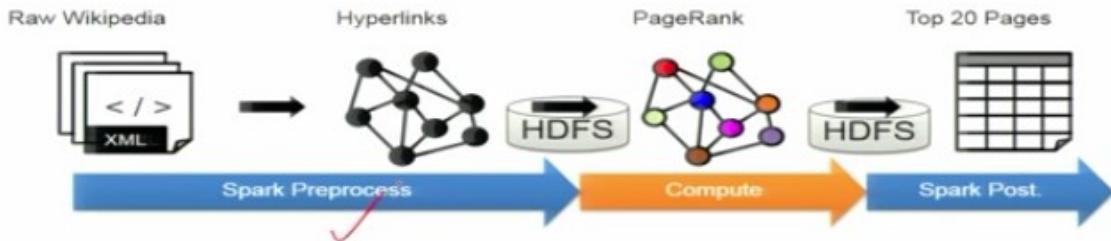
## Big Data Computing

## GraphX

So let us see that again we have to come back and see that we can also build a small pipeline in GraphX and this pipeline can be optimized so that it will be getting better performance, so pipeline intern will start with a spark processing and framework,

(Refer Slide Time: 28:06)

## A Small Pipeline in GraphX

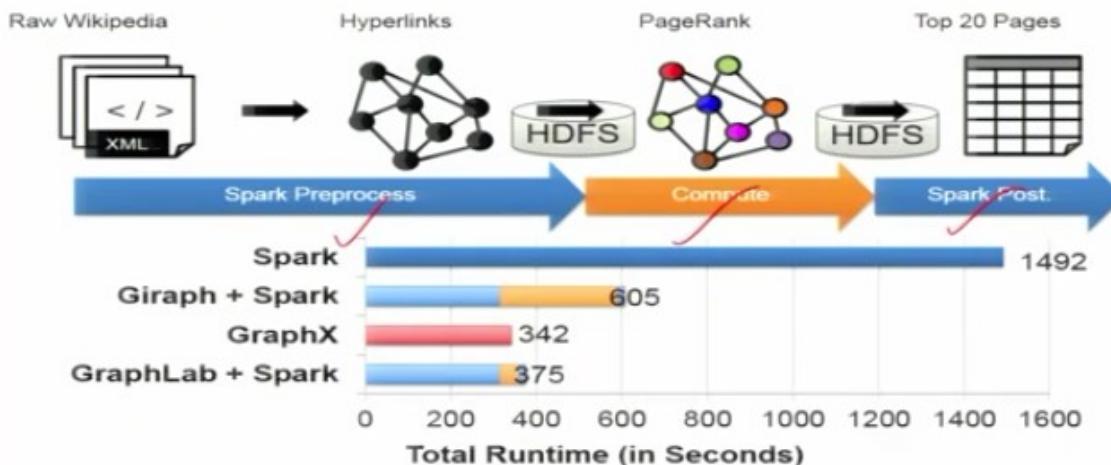


### Big Data Computing

### GraphX

so spark preprocessed will take the raw Wikipedia data and convert in the form of a hyperlink graph, and use the HDFS to compute this PageRank and then perform the spark processing to capture the top 20 pages,  
(Refer Slide Time: 28:24)

## A Small Pipeline in GraphX



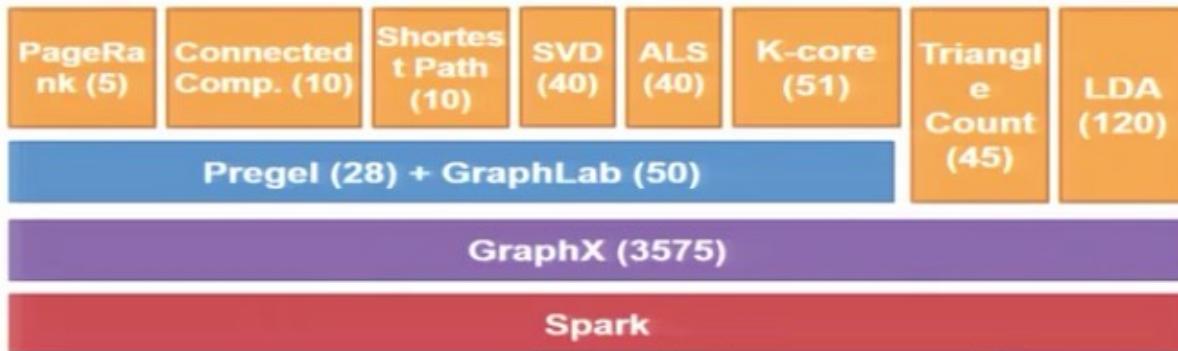
### Big Data Computing

### GraphX

so is you use this particular pipeline, so you see that the smallest time which is basically is taking, is using GraphX, why because? This view of table and the graph is integrated or basically unified in GraphX, whereas in all other framework this particular pipeline needs lot of inefficiencies due to the internal HDFS storage for handling two different viewpoints, so

therefore time end to end GraphX is faster than GraphLab also, so due to the fact that we have already seen that it has unified view of tables and graph which are supported in GraphX, (Refer Slide Time: 29:32)

## The GraphX Stack (Lines of Code)

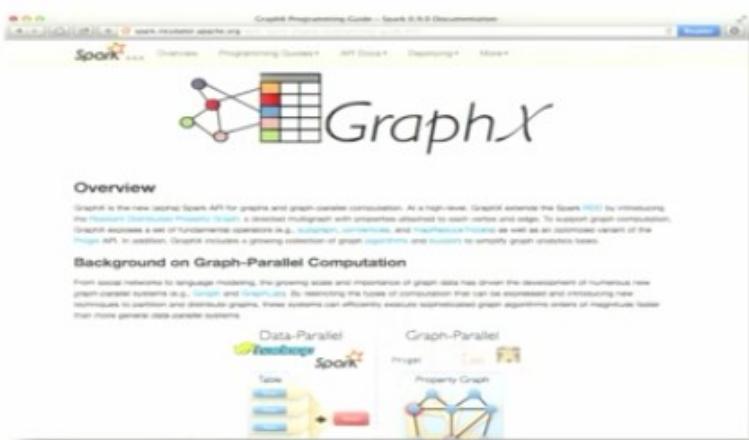


**Big Data Computing** **GraphX**  
So therefore GraphX stack has this kind of features that means it has these different lines of code which is great and above the spark, and different algorithms are available either through the Pregel or GraphLab API's or directly which is implemented as the libraries of GraphX.

(Refer Slide Time: 29:58)

## Status

- Alpha release as part of Spark 0.9



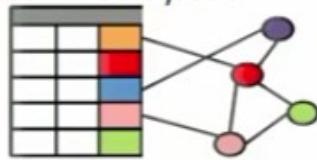
**Big Data Computing** **GraphX**  
Now here we have discussed the GraphX which is an alpha release apart of the Spark 0.9,

(Refer Slide Time: 30:07)

## GraphX: Unified Analytics

### New API

*Blurs the distinction  
between Tables and  
Graphs*



### New System

*Combines Data-Parallel  
Graph-Parallel Systems*



- Enabling users to easily and efficiently express the entire graph analytics pipeline

**Big Data Computing**

**GraphX**

so again we have to see that this new API is available and it has given a new system which will combine the data parallel and a graph parallel system.

(Refer Slide Time: 30:17)

## A Case for Algebra in Graphs

- A standard algebra is essential for graph systems:
- e.g.: SQL → proliferation of relational system
- By embedding graphs in *relational algebra*:
- Integration with tables and preprocessing
- Leverage advances in relational systems
- Graph opt. recast to relational systems

**Big Data Computing**

**GraphX**

So this particular graphs also is used as the, I mean relational algebra so the queries, sequel queries can also be operated on the graph data structure, so we have already covered the specific views that tables and graphs, the tables and graphs are composable objects and

specialized operators can exploit these semantics and also we can efficiently span or build the single pipeline which will minimize the data movement and therefore increase the efficiency, (Refer Slide Time: 31:00)

## Observations

- Domain specific views: *Tables and Graphs*
  - tables and graphs are first-class composable objects
  - specialized operators which exploit view semantics
- Single system that efficiently spans the pipeline
  - minimize data movement and duplication
  - eliminates need to learn and manage multiple systems
- Graphs through the lens of database systems
  - Graph-Parallel Pattern → Triplet joins in relational alg.
  - Graph Systems → Distributed join optimizations

### Big Data Computing

### GraphX

so graph through the lengths of database systems we can also use different relational sequel queries on top of this particular graph and various distributed join operations are also supported.

(Refer Slide Time: 31:13)

## Active Research

- Static Data → Dynamic Data
  - Apply GraphX unified approach to time evolving data
  - Model and analyze relationships over time

### Big Data Computing

### GraphX

So this has become an active area of research, how from static data we can build the dynamic data by applying the GraphX unified approach to a time evolving data model, and analyze the relationship over the time.

(Refer Slide Time: 31:28)

## Active Research

- Static Data → Dynamic Data
  - Apply GraphX unified approach to time evolving data
  - Model and analyze relationships over time
  
- Serving Graph Structured Data
  - Allow external systems to interact with GraphX
  - Unify distributed graph databases with relational database technology

### Big Data Computing

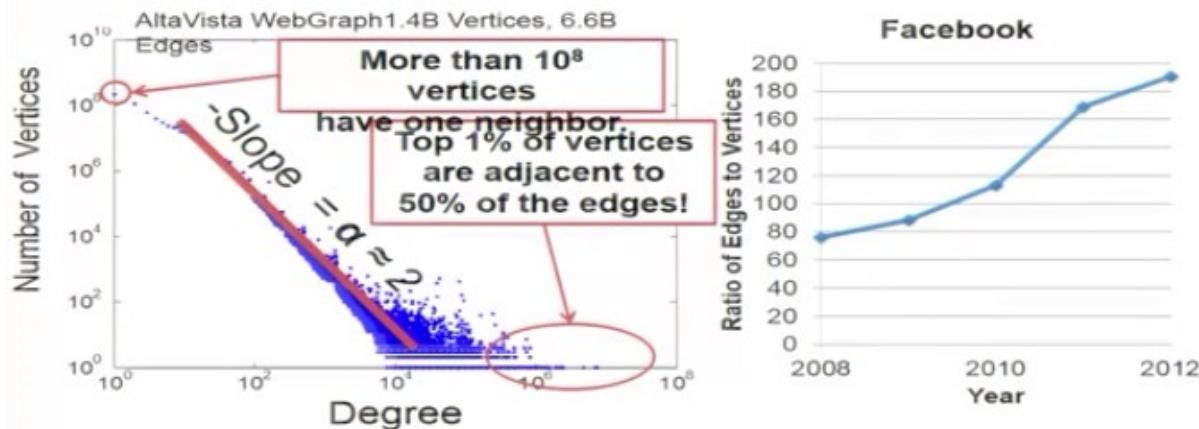
### GraphX

Now serving graph data structured data, serving graph structured data, now allow external system to interact with the graph and unify the distributed graph data based with the relational database technologies,

(Refer Slide Time: 31:41)

## Graph Property 1 Real-World Graphs

Power-Law Degree Distribution    Edges >> Vertices



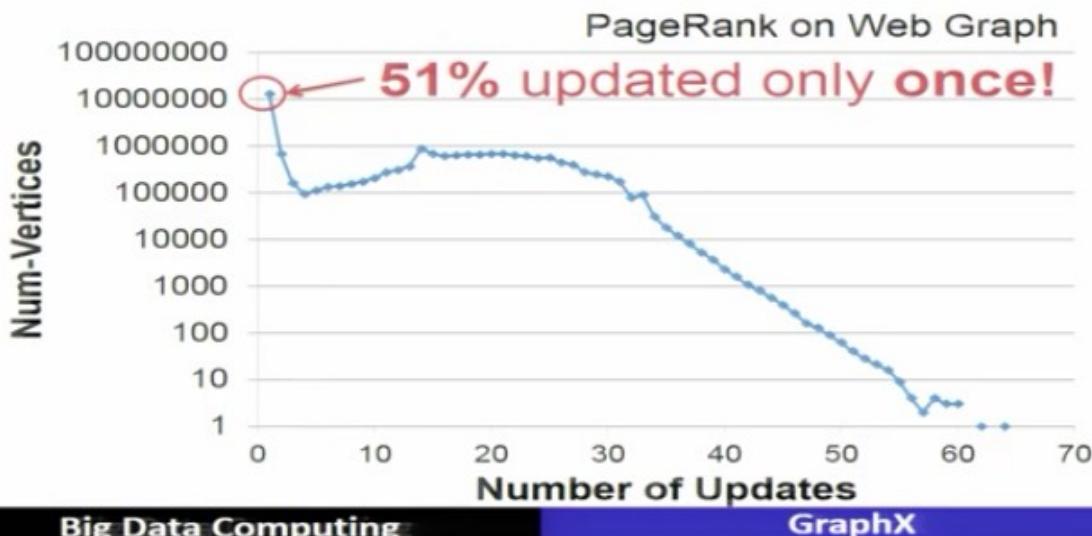
### Big Data Computing

### GraphX

so we can see here that with this support of all these internal implementations or optimizations of a GraphX even the power large distribution of the large graphs can easily be supported even for the Facebook we can see.

(Refer Slide Time: 32:06)

## Graph Property 2 Active Vertices



So here another property is about the active vertices, it's not all the time the entire graph is to be modified, only the active vertices are to be touched upon, hence the active vertices will improve the tracking of active vertices will improve the performance,

(Refer Slide Time: 32:23)

## Graphs are Essential to Data Mining and Machine Learning

- Identify influential people and information
- Find communities
- Understand people's shared interests
- Model complex data dependencies

### Big Data Computing

### GraphX

so graph are essentials to the data mining and machine learning, it will identify the influential people information, find the communities, understand people's shared interest and model the complex data dependencies.

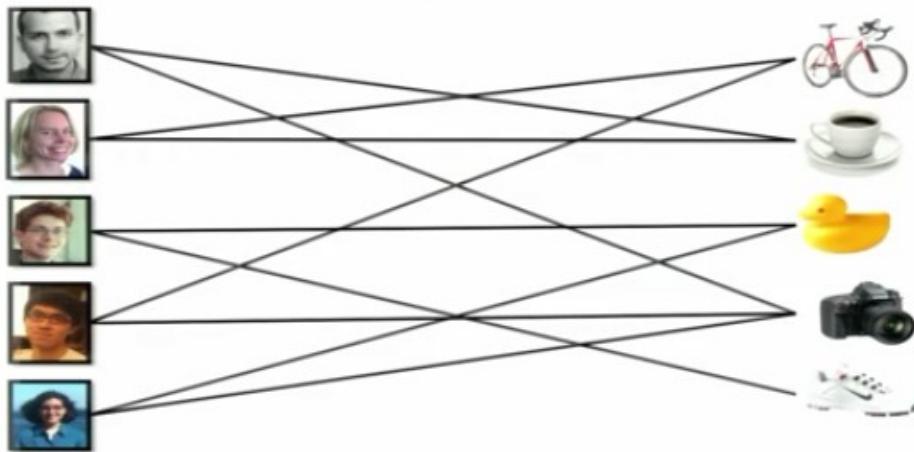
(Refer Slide Time: 32:38)

## Recommending Products

### Users

### Ratings

### Items

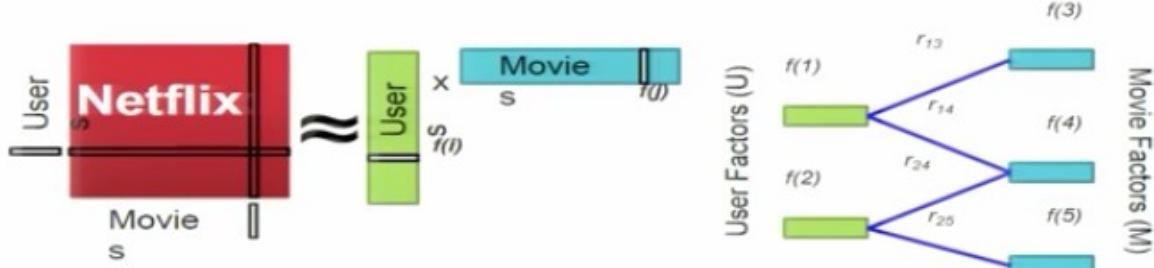


Now the graphs can be modeled as the bipartite graph or recommending the products that we have seen,

(Refer Slide Time: 32:49)

# Recommending Products

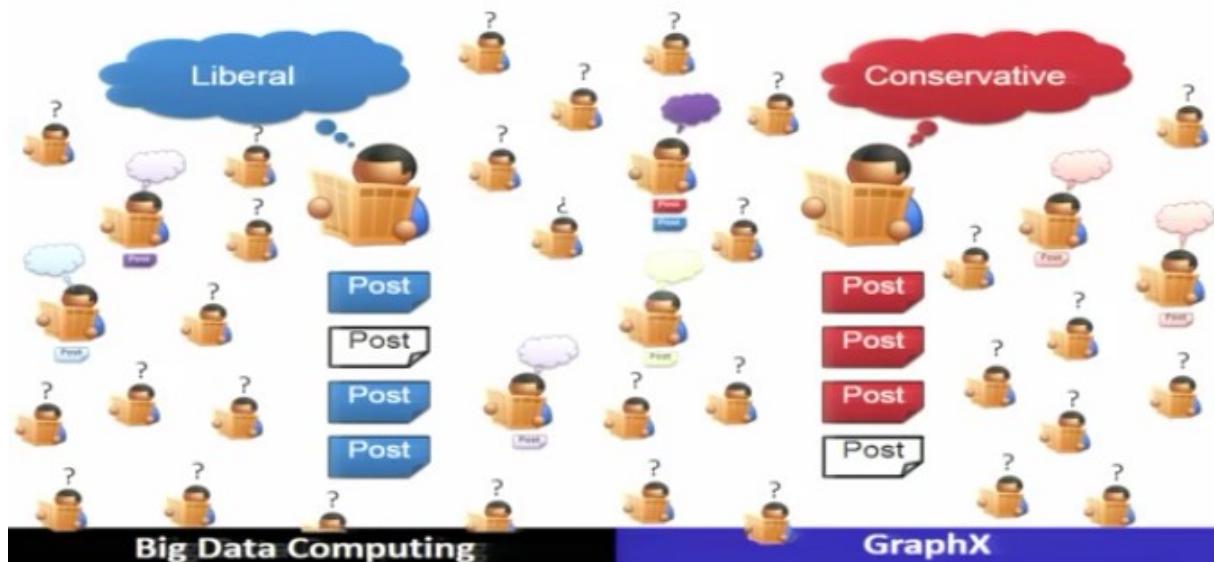
Low-Rank Matrix Factorization:



Big Data Computing:

GraphX

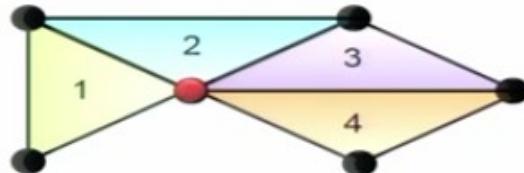
## Predicting User Behavior



it is also for recommending the products or the movies using the recommendation engines, for example in the movie case the graph can also be used, similarly to protect the user behavior we can model the problem as the graph and we can do the analysis finding out the conditional random field or belief propagation,  
(Refer Slide Time: 33:16)

## Finding Communities

- Count triangles passing through each vertex:



- Measures “cohesiveness” of local community



Fewer Triangles  
Weaker Community



More Triangles  
Stronger Community

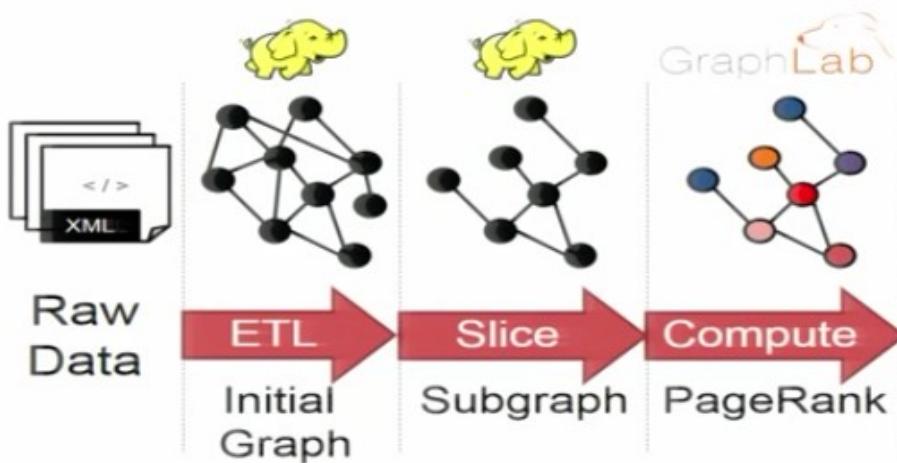
**Big Data Computing**

**GraphX**

finding the communities using triangle count and will also measure the cohesiveness of local community, for example fewer triangles means a weaker community, more triangle means a strong community,

(Refer Slide Time: 33:31)

## Example Graph Analytics Pipeline



and also we have seen the building how graph analytics can be achieved using building the pipeline and,  
(Refer Slide Time: 33:38)

## References

- Xin, R., Crankshaw, D., Dave, A., Gonzalez, J., Franklin, M.J., & Stoica, I. (2014). GraphX: Unifying Data-Parallel and Graph-Parallel Analytics. *CoRR, abs/1402.2394*.

### **GraphX: Unifying Data-Parallel and Graph-Parallel Analytics**

Reynold S. Xin  
Joseph E. Gonzalez

Daniel Crankshaw  
Michael J. Franklin

Ankur Dave  
Ion Stoica

UC Berkeley AMPLab  
(rxin, crankshaw, ankurd, jegonzal, franklin, istoica)@cs.berkeley.edu

- <http://spark.apache.org/graphx>

**Big Data Computing**

**GraphX**

so these are some of the references and this is the paper of GraphX unifying data parallel and graph parallel analytics.

### **Acknowledgement**

**Ministry of Human Resource & Development**

**Prof. Satyaki Roy**

**Co-coordinator, IIT Kanpur**

**NPTEL Team**

Sanjay Pal  
Bharat Lal  
Ashish Singh  
Badal Pradhan  
Tapobrata Das  
K. K. Mishra  
Ashutosh Gairola  
Puneet Kumar Bajpai  
Bhadro Rao  
Shikha Gupta  
Aradhana Singh Rajawat  
Sweta  
Nipa Sikdar  
Anupam Mishra  
Ram Chandra

**Manoj Shrivastava**  
**Dilip Tripathi**  
**Padam Shukla**  
**Sharwan K Verma**  
**Sanjay Mishra**  
**Shubham Rawat**  
**Santosh Nayak**  
**Pradyuman Singh Chauhan**  
**Mahendra Singh Rawat**  
**Tushar Srivastava**  
**Uzair Siddiqui**  
**Lalty Dutta**  
**Murali Krishnan**  
**Ganesh Rana**  
**Ajay Kanaujia**  
**Ashwani Srivastava**  
**M.L. Benerjee**

**an IIT Kanpur Production**

**© Copyright Reserved**

**INDIAN INSTITUTE OF TECHNOLOGY PATNA**

**NPTEL**

**NATIONAL PROGRAMME ON TECHNOLOGY ENHANCED LEARNING**

**COURSE TITLE  
BIG DATA COMPUTING**

**LECTURE-34**

**CASE STUDY: FLIGHT DATA ANALYSIS USING SPARK GRAPH-X**

**BY**

**DR. RAJIV MISRA**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY PATNA**

**(Refer Slide Time: 00:16)**

---

**Case Study: Flight Data Analysis  
using Spark GraphX**

**Big Data Computing**

**GraphX**

Cast Study: Flight Data Analysis using Spark GraphX.

(Refer Slide Time: 00:20)

## Problem Statement

- To analyze Real-Time Flight data using Spark GraphX, provide near real-time computation results and visualize the results.

### Big Data Computing

### GraphX

The problem statement, to analyze the Real-Time Flight data using Spark GraphX, and to provide near real-time computation results and visualize the results.

(Refer Slide Time: 00:35)

## Flight Data Analysis using Spark GraphX

### Dataset Description:

The data has been collected from U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics site which tracks the on-time performance of domestic flights operated by large air carriers. The Summary information on the number of on-time, delayed, canceled, and diverted flights is published in DOT's monthly Air Travel Consumer Report and in this dataset of January 2014 flight delays and cancellations.

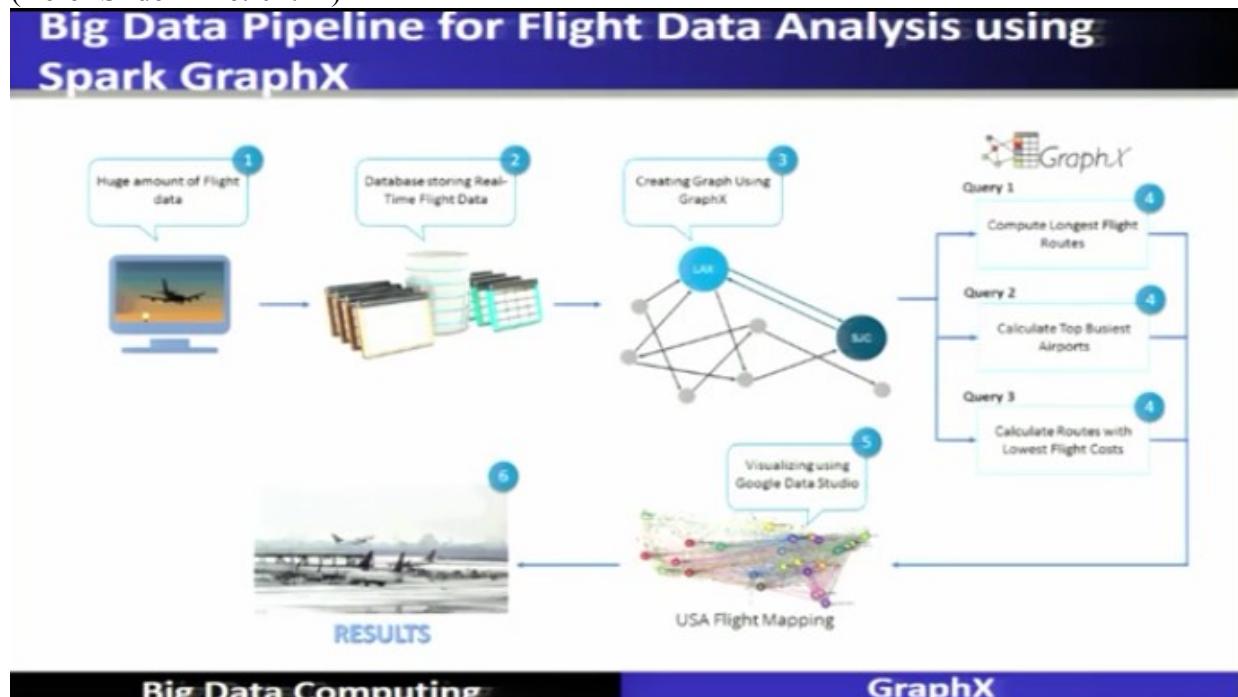
### Big Data Computing

### GraphX

So for this let us see the dataset description, so data has been collected from US Department of Transportation DOT, Bureau of Transportation Statistics site which tracks on-time performance of domestic flights operated by large carriers. The summary information on the number of on-time delayed, canceled, and diverted flights is published in DOT's monthly Air Travel

Consumer Report and in this dataset of January 2014 flight delays and cancellations are been considered.

(Refer Slide Time: 01:11)



So we are going to analyze this dataset that is called a flight dataset using the pipeline, that is called big data pipeline for Flight Data Analysis using Spark GraphX, so the first stage, first step in the pipeline is we have collected the dataset, so we have obtained the dataset which is a huge amount of flight data information.

Now this particular data base storing real-time flight data is being applied or being used into the data base, so after getting the data or a car pass or a big data set, we converted into the form of a graph, so that means the data is converted into the form of a table, and from table which will be converted into the form of a graph, the second pipeline, so creating graph using GraphX we will create a graph for the entire data.

So after creating the graph we can apply various queries based on the graph algorithms, so the queries are we have to compute the longest flights, longest flight routes or we can compute the top busiest airports or we can compute the routes with the lowest flight costs, so after doing all these queries we can also do the visualization of a flight mapping on the maps, and then we will use this particular result for this.

Let us see how this steps which is called the big data pipeline is use in doing this analyzing this flight data information.

(Refer Slide Time: 03:11)

## UseCases

- i. Monitoring Air traffic at airports
- ii. Monitoring of flight delays
- iii. Analysis overall routes and airports
- iv. Analysis of routes and airports per airline

### Big Data Computing

### GraphX

So this particular study will have the use cases such as monitoring air traffic at airports, monitoring of flight delays, analysis of overall routes and airports, analysis of routes and airports per airline and so on.

(Refer Slide Time: 03:32)

## Objectives

- Compute the total number of airports
- Compute the total number of flight routes
- Compute and sort the longest flight routes
- Display the airports with the highest incoming flights
- Display the airports with the highest outgoing flights
- List the most important airports according to PageRank
- List the routes with the lowest flight costs
- Display the airport codes with the lowest flight costs
- List the routes with airport codes which have the lowest flight costs

### Big Data Computing

### GraphX

Let us see the objective of this session about this use case or case study is to obtain for following objectives, first is that how to compute the total number of airports in this dataset, how to find out the total number of flight routes, how to compute and sort the longest flight routes, how to display the airports with the highest incoming flights, how to display the airports with the highest outgoing flights, how to list the most important airports according to the

PageRank, and list the routes with the lowest flight costs, display the airport codes and the lowest flight costs, list the routes with the airport codes which have the lowest flight costs, let us see how these queries or how these objectives are to be fulfilled using the given dataset. (Refer Slide Time: 04:23)

## Features

- 17 Attributes

| Attribute Name | Attribute Description                 |
|----------------|---------------------------------------|
| dOfM           | Day of Month                          |
| dOfW           | Day of Week                           |
| carrier        | Unique Airline Carrier Code           |
| tailNum        | Tail Number                           |
| fNum           | Flight Number Reporting Airline       |
| origin_id      | Origin Airport Id                     |
| origin         | Origin Airport                        |
| dest_id        | Destination Airport Id                |
| dest           | Destination Airport                   |
| crsdeptime     | CRS Departure Time (local time: hhmm) |
| deptime        | Actual Departure Time                 |

### Big Data Computing

### GraphX

This dataset, if we go in more detail and see the features, there are 17 different attributes available in this particular dataset, for example dOfM that is day of the month, dOfW means day of the week, carrier unique airline carrier code, tail number is the tail number, and F number is the flight number reporting airlines, and origin ID is the origin airport ID, origin and origin airport, destination ID is the destination airport ID, destination is destination airport, CRS departure time, departure time is actual departure time, and departure delay minutes difference in the minutes between the schedule and actual departure and earlier departure is set to 0.

(Refer Slide Time: 05:04)

# Features

| Attribute Name | Attribute Description                                                                         |
|----------------|-----------------------------------------------------------------------------------------------|
| depdelaymins   | Difference in minutes between scheduled and actual departure time. Early departures set to 0. |
| crsarrtime     | CRS Arrival Time (local time: hhmm)                                                           |
| arrtime        | Actual Arrival Time (local time: hhmm)                                                        |
| arrdelaymins   | Difference in minutes between scheduled and actual arrival time. Early arrivals set to 0.     |
| crselapsedtime | CRS Elapsed Time of Flight, in Minutes                                                        |
| dist           | Distance between airports (miles)                                                             |

## Big Data Computing

## GraphX

Similarly there are arrival times and actual arrival time difference between, in the minutes between the scheduled and actual arrival time that is arrival delay in the minutes, and similarly the elapse time of the flight and the distance between the airport.

(Refer Slide Time: 05:33)

# Sample Dataset

| A  | B    | C    | D       | E       | F     | G         | H      | I       | J    | K          | L       | M            | N          | O       | P            | Q              |      |
|----|------|------|---------|---------|-------|-----------|--------|---------|------|------------|---------|--------------|------------|---------|--------------|----------------|------|
| 1  | dOfM | dOfW | carrier | tailNum | flNum | origin_id | origin | dest_id | dest | crsdeptime | deptime | depdelaymins | crsarrtime | arrtime | arrdelaymins | crselapsedtime | dist |
| 2  | 1    | 3    | AA      | N338AA  | 1     | 12478     | JFK    | 12892   | LAX  | 900        | 914     | 14           | 1225       | 1238    | 13           | 385            | 2475 |
| 3  | 2    | 4    | AA      | N338AA  | 1     | 12478     | JFK    | 12892   | LAX  | 900        | 857     | 0            | 1225       | 1226    | 1            | 385            | 2475 |
| 4  | 4    | 6    | AA      | N327AA  | 1     | 12478     | JFK    | 12892   | LAX  | 900        | 1005    | 65           | 1225       | 1324    | 59           | 385            | 2475 |
| 5  | 5    | 7    | AA      | N323AA  | 1     | 12478     | JFK    | 12892   | LAX  | 900        | 1050    | 110          | 1225       | 1415    | 110          | 385            | 2475 |
| 6  | 6    | 1    | AA      | N319AA  | 1     | 12478     | JFK    | 12892   | LAX  | 900        | 917     | 17           | 1225       | 1217    | 0            | 385            | 2475 |
| 7  | 7    | 2    | AA      | N328AA  | 1     | 12478     | JFK    | 12892   | LAX  | 900        | 910     | 10           | 1225       | 1212    | 0            | 385            | 2475 |
| 8  | 8    | 3    | AA      | N323AA  | 1     | 12478     | JFK    | 12892   | LAX  | 900        | 923     | 23           | 1225       | 1215    | 0            | 385            | 2475 |
| 9  | 9    | 4    | AA      | N339AA  | 1     | 12478     | JFK    | 12892   | LAX  | 900        | 859     | 0            | 1225       | 1204    | 0            | 385            | 2475 |
| 10 | 10   | 5    | AA      | N319AA  | 1     | 12478     | JFK    | 12892   | LAX  | 900        | 929     | 29           | 1225       | 1245    | 20           | 385            | 2475 |

## Big Data Computing

## GraphX

So this is the sample dataset with all the 17 features shown over here for our analysis use case,  
(Refer Slide Time: 05:40)

# Spark Implementation

```
1 //Importing the necessary classes
2 import org.apache.spark._
3 ...
4 import java.io.File
5
6 object airport {
7
8 def main(args: Array[String]){
9
10 //Creating a Case Class Flight
11 case class Flight(dofM:String, dofW:String, ... ,dist:Int)
12
13 //Defining a Parse String function to parse input into Flight class
14 def parseFlight(str: String): Flight = {
15 val line = str.split(",")
16 Flight(line(0), line(1), ... , line(16).toInt)
17 }
18 val conf = new SparkConf().setAppName("airport").setMaster("local[2]")
19 val sc = new SparkContext(conf)
```

Big Data Computing

GraphX

so here we can see here the part of the spark code which implements this flight data analysis using GraphX, so let us understand some of the code before we go into more detail of it.

So that means first we have to create a class which is called a flight class with all these details which are provided as 17 different features, then we can pass the string function to pass the input into the flight class data, so for that these functions are there, and then we will create the application name and we have to create the spark context, so this will create the spark context.

(Refer Slide Time: 06:51)

# Spark Implementation

```
1 //Load the data into a RDD
2
3 val textRDD = sc.textFile ("/home/iitp/spark-2.2.0-bin-hadoop-2.6/flights/airportdataset.csv")
4
5 //Parse the RDD of CSV lines into an RDD of flight classes
6 val flightsRDD = Map ParseFlight to Text RDD
7
8 //Create airports RDD with ID and Name
9 val airports = Map Flight OriginID and Origin
10 airports.take(1)
11
12 //Defining a default vertex called nowhere and mapping Airport ID for printlns
13 val nowhere = "nowhere"
14 val airportMap = Use Map Function .collect.toList.toMap
15
16 //Create routes RDD with sourceID, destinationID and distance
17 val routes = flightsRDD. Use Map Function .distinct
18 routes.take(2)
19
20 //Create edges RDD with sourceID, destinationID and distance
21 val edges = routes.map{ Map OriginID and DestinationID } => Edge(org_id.toLong,
22 edges.take(1)
23
24 //Define the graph and display some vertices and edges
25 val graph = Graph(Airports, Edges and Nowhere)
26 graph.vertices.take(2)
27 graph.edges.take(2)
```

Big Data Computing

GraphX

Now the next step is to read the file that is the text file, once it is read then it will become an RDD, so the file is read and after reading it becomes an RDD into the spark system, and now we have to pass the RDD into an RDD flight class using this map function, and it will create the airport RDD's with the ID and names using this particular function.

After defining default vertex called nowhere and mapping airport ID's for print lines, so here all these details are listed over here, now we have to create the routes RDD's with the source ID and destination and the distance using the map function, and create the edge RDD's with the source ID and the distance and so on.

So as RDD is the vertex ID's routes RDD's are now created, now we can define the graph as the airports as the vertices, and the routes and edges, so routes will be the edges, and the airports will be the vertices, the vertices, edges and nowhere, so after taking them now we will convert it into an edge RDD and vertex RDD.

(Refer Slide Time: 08:50)

## Graph Operations

```
//Query 1 - Find the total number of airports
val numairports = Vertices Number

//Query 2 - Calculate the total number of routes?
val numroutes = Number Of Edges

//Query 3 - Calculate those routes with distances more than 1000 miles
graph.edges.filter { Get the edge distance }=> distance > 1000}.take(3)
```

**Big Data Computing**

**GraphX**

So using vertices and a graphs we can now, we have computed, we have form the graph of that particular dataset, now we have to perform various queries, so the query 1 is to find out how many number of airports are there in that, so just we have to calculate, we have to find out the number of vertices in the graph. If you want to find out how many number of routes are there, we have to calculate how many different edges are there in the graph.

Now if you want to find out, calculate those routes with the distance more than 1000 miles, so easily we can apply the filter function of this GraphX on this particular graph, so after applying the filter function where in the distance is greater than 1000, it will generate graph that is called as subgraph. Now this whole function will not delete the portions the graph, but it will use tombstone and will generate the subgraph out of the main graph.

(Refer Slide Time: 10:00)

## Graph Operations

- How many airports are there?
- How many unique flights from airport A to airport B are there?

```
graph.numVertices // 305
graph.numEdges // 5366
```

- What are the top 10 flights from airport to airport?

```
graph.triplets.sortBy(_.attr, ascending=false).map(triplet =>
 "There were " + triplet.attr.toString + " flights from " + triplet.srcAttr + " to "
 + triplet.dstAttr + ".").take(10)
```

```
res60: Array[String] = Array(There were 13788 flights from SFO to LAX., There were 1339
0 flights from LAX to SFO., There were 12383 flights from OGG to HNL., There were 12035
flights from LGA to BOS., There were 12029 flights from BOS to LGA., There were 12014
flights from HNL to OGG., There were 11773 flights from LAX to LAS., There were 11729
flights from LAS to LAX., There were 11257 flights from LAX to SAN., There were 11224
flights from SAN to LAX.)
```

### Big Data Computing

### GraphX

Similarly if you want to find out how many airports are there, so airports are represented as the number of vertices, so we have to run this particular function that is graph.numVertices so we can find out that number of vertices are 304, 305.

Similarly how many unique flights from airport A to B are there, so we have to find out the number of edges here and if you want to find out the top 10 flights from airport to airport, so we can find out using the graph triplets and we can sort the flights, top 10 flights and now we can take these 10 data out of this sorted top 10 flights, we can use it, so we can see here that there are so many number of flights from source destination and by taking the triplets and doing the sort and we can easily get this information out of this particular graph.

(Refer Slide Time: 11:20)

# Graph Operations

- What are the lowest 10 flights from airport to airport?

```
graph.triplets.sortBy(_.attr).map(triplet =>
 "There were " + triplet.attr.toString + " flights from " + triplet.srcAttr + " to " + triplet.dstAttr + ".").take(10)
```

```
res62: Array[String] = Array(There were 1 flights from RNO to PIH., There were 1 flights fro m PHL to ICT., There were 1 flights from FSD to PIA., There were 1 flights from RIC to JAX., There were 1 flights from MOD to BFL., There were 1 flights from ASE to MSN., There were 1 flights from JFK to HPN., There were 1 flights from MCO to LIT., There were 1 flights from ROA to BWI., There were 1 flights from OMA to ABQ.)
```

## Big Data Computing:

## GraphX

So what are the lowest 10 flights from airport to airport using some triplets and by sorting it in the descending order we can obtain this data also.

(Refer Slide Time: 11:31)

# Graph Operations

**what airport has the most in degrees or unique flights into it?**

```
graph.inDegrees.join(airportVertices).sortBy(_.._2._1, ascending=False).take(1)
```

```
Array[(org.apache.spark.graphx.VertexId, (Int, String))] = Array((2042033420, (173, ATL)))
```

And out of it?

```
graph.outDegrees.join(airportVertices).sortBy(_.._2._1, ascending=False).take(1)
```

```
Array[(org.apache.spark.graphx.VertexId, (Int, String))] = Array((2042033420, (173, ATL)))
```

## Big Data Computing:

## GraphX

Now if you want to find out which, what airport has the most in degree and or unique flights into it, then using in degrees and the joint airport vertices we can obtain this amount of information, and out of it what are the flights which are going out of the airport, most of the degrees so that also we can find out using out degree, and after applying the joint operation.

(Refer Slide Time: 12:07)

# Graph Operations

What are our most important airports ?

```
val ranksAndAirports = ranks.join(airportVertices).sortBy(_._2._1, ascending=False).map(_._2._2)
ranksAndAirports.take(10)
```

```
res81: Array[String] = Array(ATL, DFW, ORD, MSP, SLC, DEN, DTW, IAH, CVG, LAX)
```

## Big Data Computing

## GraphX

What are the most important airports? So for that using PageRank what we can ranks and, not using PageRank, but ranks and airport means it will sort and find out the airports,  
(Refer Slide Time: 12:24)

# Graph Operations

Output the routes where the distance between airports exceeds 1000 miles

```
graph.edges.filter {
 case (Edge(org_id, dest_id, distance)) => distance > 1000
}.take(5).foreach(println)
```

## Big Data Computing

## GraphX

so graph operations output the routes where the distance between the airport exceeds 1000 that we have seen, it will just apply the filter and get this output of the routes where the distance between the airport exceeds 1000 miles.

(Refer Slide Time: 12:42)

# Graph Operations

**Output the airport with maximum incoming flight**

```
// Define a reduce operation to compute the highest degree vertex
def max: (VertexId, Int), b: (VertexId, Int)): (VertexId, Int) = {
 if (a._2 > b._2) a else b
}
// Compute the max degrees
val maxInDegree: (VertexId, Int) = graph.inDegrees.reduce(max)
```

## Big Data Computing

## GraphX

Similarly if you want to output the airport with the maximum incoming flight, so here in degree we can find out with the maximum of it, and this will give the maximum incoming flight.  
(Refer Slide Time: 13:00)

# Graph Operations

**Output the airports with maximum incoming flights**

```
val maxIncoming=graph.inDegrees.collect.sortWith (_._2 >
 _._2).map(x => (airportMap(x._1),
 x._2)).take(10).foreach(println)
```

## Big Data Computing

## GraphX

Similarly maximum incoming flights we can get using this particular piece of code.  
(Refer Slide Time: 13:08)

# Graph Operations

## Output the longest routes

```
graph.triplets.sortBy(_.attr, ascending = false).map(triplet =>
 "There were " + triplet.attr.toString + " flights from " + triplet.srcAttr
 + " to " + triplet.dstAttr + ".").take(20).foreach(println)
```

**Big Data Computing**

**GraphX**

Similarly we have to output the longest routes by doing this kind of sorting, and applying on the graph triplets.

(Refer Slide Time: 13:19)

# Graph Operations

## Output the cheapest airfare routes

```
val gg = graph.mapEdges(e => 50.toDouble + e.attr.toDouble / 20)
//Call pregel on graph
val sssp = initialGraph.pregel[Double, PositiveInfinity]{
 //vertex program
 (id, distCost, newDistCost) => math.min(distCost, newDistCost), triplet => {
 //send message
 if(triplet.srcAttr + triplet.attr < triplet.dstAttr) {
 Iterator((triplet.dstId, triplet.srcAttr + triplet.attr))
 } else {
 Iterator.empty
 }
 },
 //Merge Messages
 (a,b) => math.min(a,b)
}
//print routes with lowest flight cost
print("routes with lowest flight cost")
println(sssp.edges.take(10).mkString("\n"))
```

**Big Data Computing**

**GraphX**

So if you want to find out the cheapest airline route, then this is the code and using this particular code we can obtain this.

(Refer Slide Time: 13:29)

# Graph Operations (PageRank)

**Output the most influential airports using PageRank**

```
val rank = graph.pageRank(0.001).vertices
val temp = rank.join(airports)
temp.take(10).foreach(println)
```

**Output the most influential airports from most influential to latest**

```
val temp2 = temp.sortBy(_._2._1,false)
```

**Big Data Computing**

**GraphX**

Now we can find out the most influential airport using the PageRank, so we can apply the PageRank algorithm over the vertices, and we can then join the airports and then take a temp, and top 10 and then it will become the most influential airport using PageRank, so output the most influential airport from the most influential to the latest, so further more we can sort and we can obtain this.

(Refer Slide Time: 14:05)

## Conclusion

- The growing scale and importance of graph data has driven the development of specialized graph computation engines capable of inferring complex recursive properties of graph-structured data.
- In this lecture we have discussed GraphX, a distributed graph processing framework that unifies graph-parallel and data-parallel computation in a single system and is capable of succinctly expressing and efficiently executing the entire graph analytics pipeline.

**Big Data Computing**

**GraphX**

So conclusion, the growing scale and importance of graph data has driven the development of specialized graph computation engines, capable of inferring complex recursive properties of graph-structured data.

In this lecture we have discussed GraphX, a distributed processing framework that unifies the graph-parallel and data-parallel computation in a single system and is capable of succinctly expressing and efficiently executing the entire graph data, graph analytics pipeline. Thank you.

### **Acknowledgement**

**Ministry of Human Resource & Development**

**Prof. Satyaki Roy**

**Co-coordinator, IIT Kanpur**

**NPTEL Team**

**Sanjay Pal**  
**Bharat Lal**  
**Ashish Singh**  
**Badal Pradhan**  
**Tapobrata Das**  
**K. K. Mishra**  
**Ashutosh Gairola**  
**Puneet Kumar Bajpai**  
**Bhadro Rao**  
**Shikha Gupta**  
**Aradhana Singh Rajawat**  
**Sweta**  
**Nipa Sikdar**  
**Anupam Mishra**  
**Ram Chandra**  
**Manoj Shrivastava**  
**Dilip Tripathi**  
**Padam Shukla**  
**Sharwan K Verma**  
**Sanjay Mishra**  
**Shubham Rawat**  
**Santosh Nayak**  
**Pradyuman Singh Chauhan**  
**Mahendra Singh Rawat**  
**Tushar Srivastava**  
**Uzair Siddiqui**  
**Lalty Dutta**



**THIS BOOK IS NOT FOR SALE  
NOR COMMERCIAL USE**



(044) 2257 5905/08



nptel.ac.in



swayam.gov.in