UNIT I

CHAPTER 1

Introduction to Big Data

University Prescribed Syllabus

Introduction to Big Data, Big Data characteristics, types of Big Data, Traditional vs. Big Data business approach, Big Data Challenges, Examples of Big Data in Real Life, Big Data Applications.

Self-learning Topics : Identification of Big Data applications and its solutions.

1.1 INTRODUCTION TO BIG DATA AND HADOOP

GQ. Firstly, We need to know "what is data" ?

(1) Now a day the amount of data created by various advanced technologies like Social networking sites, E-commerce etc. is very large. It is really difficult to store such huge data by using the traditional data storage facilities.

(2) Until 2003, the size of data produced was 5 billion gigabytes. If this data is stored in the form of disks it may fill an entire football field. In 2011, the same amount of data was created in every two days and in 2013 it was created in every ten minutes. This is really tremendous rate.

(3) In this topic, we will discuss about big data on a fundamental level and define common concepts related to big data. We will also see in deep about some of the processes and technologies currently being used in this field.

1.1.1 What is Big Data ?

GQ. What is Big Data ?

1. Big Data is a massive collection of data that continues to grow dramatically over time.
2. It is a data set that is so huge and complicated that no typical data management technologies can effectively store or process it.
3. Big Data is like regular data, but it is much larger. A data which are very large in size.
4. Normally we work on data of size MB(WordDoc ,Excel) or maximum GB(Movies, Codes) but data in Peta bytes i.e. 1015 byte size is called Big Data.
5. It is stated that almost 90% of today's data has been generated in the past 3 years.

1.1.2 Sources of Big Data

There are various sources of big data. Now a days in number of fields such huge data get created. Following are the some of fields.

1. **Stock Exchange :** The data in the share market regarding information about prices and status details of shares of thousands of companies is very huge.
2. **Social Media Data :** The data of social networking sites contains information about all the account holders, their posts, chat history, advertisements etc. On topmost sites like facebook and whatsapp, there are literally billions of users.

3. **Video sharing portals :** Video sharing portals like youtube, Vimeo etc. contains millions of videos each of which requires lots of memory to store.

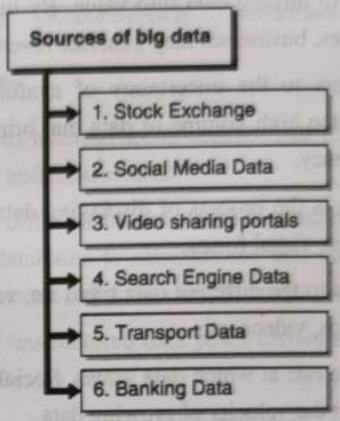


Fig. 1.1.1 : Sources of big data

4. **Search Engine Data :** The search engines like Google and Yahoo holds lot much of metadata regarding various sites.
5. **Transport Data :** Transport data contains information about model, capacity, distance and availability of various vehicles.
6. **Banking Data :** The big giants in banking domain like SBI or ICICI hold large amount of data regarding huge transactions of account holders.

1.2 BIG DATA CHARACTERISTICS

GQ. What are Characteristics of Big Data ?

UQ. Describe any five characteristics of Big Data. (MU - Dec. 17, 5 Marks)

UQ. Explain what characteristic of Social Networks make it Big Data. (MU - May 18, 5 Marks)

GQ. Explain Big data along with 5V's.

(1) **Volume** represents the volume i.e. amount of data that is growing at a high rate i.e. data volume in Petabytes.

(2) **Value** refers to turning data into value. By turning accessed big data into values, businesses may generate revenue.

(3) **Veracity** refers to the uncertainty of available data. Veracity arises due to the high volume of data that brings incompleteness and inconsistency.

(4) **Visualization** is the process of displaying data in charts, graphs, maps, and other visual forms.

(5) **Variety** refers to the different data types i.e. various data formats like text, audios, videos, etc.

(6) **Velocity** is the rate at which data grows. Social media contributes a major role in the velocity of growing data.

(7) **Virality** describes how quickly information gets spread across people to people (P2P) networks.

1.2.1 Volume

- As it follows from the name, big data is used to refer to enormous amounts of information.
- We are talking about not gigabytes but terabytes and petabytes of data.
- The IoT (Internet of Things) is creating exponential growth in data.
- The volume of data is projected to change significantly in the coming years.
- Hence, 'Volume' is one characteristic which needs to be considered while dealing with Big Data.

Volume

[Data at Rest]

Terabytes, Petabytes Records/Arch Table/Files Distributed

1.2.2 Variety

- Variety refers to heterogeneous sources and the nature of data, both structured and unstructured.
- Data comes in different formats – from structured, numeric data in traditional databases to unstructured text documents, emails, videos, audios, stock ticker data and financial transactions.
- This variety of unstructured data poses certain issues for storage, mining and analysing data.
- Organizing the data in a meaningful way is no simple task, especially when the data itself changes rapidly.
- Another challenge of Big Data processing goes beyond the massive volumes and increasing velocities of data but also in manipulating the enormous variety of these data.

Variety

[Data in many Forms]

Structured Unstructured Text Multimedia

1.2.3 Veracity

- Veracity describes whether the data can be trusted. Veracity refers to the uncertainty of available data.
- Veracity arises due to the high volume of data that brings incompleteness and inconsistency.
- Hygiene of data in analytics is important because otherwise, you cannot guarantee the accuracy of your results.

- Because data comes from so many different sources, it's difficult to link, match, cleanse and transform data across systems.
- However, it is useless if the data being analysed are inaccurate or incomplete.
- Veracity is all about making sure the data is accurate, which requires processes to keep the bad data from accumulating in your systems.

☞ **Veracity**

[Data in Doubt]

Trustworthiness Authenticity Accurate Availability

1.2.4 Velocity

- Velocity is the speed in which data is grows, process and becomes accessible.
- A data flows in from sources like business processes, application logs, networks, and social media sites, sensors, Mobile devices, etc.
- The flow of data is massive and continuous.
- Most data are warehoused before analysis, there is an increasing need for real-time processing of these enormous volumes.
- Real-time processing reduces storage requirements while providing more responsive, accurate and profitable responses.
- It should be processed fast by batch, in a stream-like manner because it just keeps growing every years.

☞ **Velocity**

[Data in Motion]

Streaming Batch Real / Near Time Processes

1.2.5 Value

- It refers to turning data into value. By turning accessed big data into values, businesses may generate revenue.
- Value is the end game. After addressing volume, velocity, variety, variability, veracity, and visualization – which takes a lot of time, effort and resources – you want to be sure your organization is getting value from the data.
- For example, data that can be used to analyze consumer behavior is valuable for your company because you can use the research results to make individualized offers.

☞ **Value**

[Data into Money]

Statistical Events Correlations

1.2.6 Visualization

- Big data visualization is the process of displaying data in charts, graphs, maps, and other visual forms.
- It is used to help people easily understand and interpret their data at a glance, and to clearly show trends and patterns that arise from this data.
- Raw data comes in a different formats, so creating data visualizations is process of gathering, managing, and transforming data into a format that's most usable and meaningful.
- Big Data Visualization makes your data as accessible as possible to everyone within your organization, whether they have technical data skills or not.

☞ **Visualization**

[Data Readable]

Readable Accessible Presentation Visual Forma

1.2.7 Virality

- Virality describes how quickly information gets spread across people to people (P2P) networks.
- It is measures how quickly data is spread and shared to each unique node.
- Time is a determinant factor along with rate of spread.

☞ **Virality**

[Data Spread]

- P2P
- Shared
- Rate of Spread

1.3 TYPES OF BIG DATA

GQ. What are different Types of Big Data ?

There are three types of Big Data Analytics :

1. Unstructured 2. Structured 3. Semi-structured

1.3.1 Type #1 : Unstructured

- Any data with unknown form or the structure is classified as unstructured data. In addition to the size being huge, un-structured data poses multiple challenges in terms of its processing for deriving value out of it.

- Typical example of unstructured data is, a heterogeneous data source containing a combination of simple text files, images, videos like search in Google Engine.
- Now a day organizations have wealth of data available with them but unfortunately they don't know how to derive value out of it since this data is in its raw form or unstructured format.
- Human Generated Data Machine Generated Data.
- Unstructured – Example : The output returned by 'Google Search'

1.3.1(A) Characteristics of Unstructured Data

(1) Data neither conforms to a data model nor has any structure.

(2) Data can not be stored in the form of rows and columns as in Databases.

(3) Data does not follows any semantic or rules.

(4) Data lacks any particular format or sequence.

(5) Data has no easily identifiable structure.

(6) Due to lack of identifiable structure, it can not used by computer programs easily.

1.3.1(B) Sources of Unstructured Data

(1) Web pages (2) Images (JPEG, GIF, PNG, etc.)

(3) Videos (4) Memos

(5) Reports

(6) Word documents and PowerPoint presentations

(7) Surveys

1.3.1(C) Advantages and Disadvantages of Unstructured Data

Advantages

1. Its supports the data which lacks a proper format or sequence.
2. The data is not constrained by a fixed schema.
3. Very Flexible due to absence of schema.
4. Data is portable.
5. It is very scalable.
6. It can deal easily with the heterogeneity of sources.
7. These type of data have a variety of business intelligence and analytics applications.

Disadvantages

1. It is difficult to store and manage unstructured data due to lack of schema and structure.
2. Indexing the data is difficult and error prone due to unclear structure and not having pre-defined attributes. Due to which search results are not very accurate.
3. Ensuring security to data is difficult task.

1.3.2 Type #2 : Structured

- Any data that can be stored, accessed and processed in the form of fixed format is termed as a "Structured" data.
- Over the period of time, talent in computer science have achieved greater success in developing techniques for working with such kind of data (where the format is well known in advance) and also determining value out of it.
- When size of such data grows to a huge extent, typical sizes are being in the range of multiple zettabyte. Data stored in a relational database management system in one example of a structured data.

- **Structured data** is the data which conforms to a data model, has a well define structure, follows a consistent order and can be easily accessed and used by a person or a computer program.
- Structured data is usually stored in well-defined schemas such as Databases. It is generally tabular with column and rows that clearly define its attributes.
- SQL (Structured Query language) is often used to manage structured data stored in databases.

1.3.2(A) Characteristics of Structured Data

- Data conforms to a data model and has easily identifiable structure.
- Data is stored in the form of rows and columns.

Example : Database

- Data is well organised so, Definition, Format and Meaning of data is explicitly known.
- Data resides in fixed fields within a record or file.
- Similar entities are grouped together to form relations or classes.
- Entities in the same group have same attributes.
- Easy to access and query, So data can be easily used by other programs.
- Data elements are addressable, so efficient to analyse and process.

1.3.2(B) Sources of Structured Data

(1) SQL Databases (2) Spreadsheets such as Excel
(3) OLTP Systems (4) Online forms
(5) Sensors such as GPS or RFID tags
(6) Network and Web server logs
(7) Medical devices

1.3.2(C) Advantages of Structured Data

1. Structured data have a well defined structure that helps in easy storage and access of data.
2. Data can be indexed based on text string as well as attributes. This makes search operation hassle-free.
3. Data mining is easy i.e. knowledge can be easily extracted from data.
4. Operations such as Updating and deleting is easy due to well structured form of data.
5. Business Intelligence operations such as Data warehousing can be easily undertaken.
6. Easily scalable in case there is an increment of data.
7. Ensuring security to data is easy.

Structured - Example

Employee_Table

Employee_ID	Employee_Name	Gender	Department	Salary_In_lacs
1	XYX	MALE	FINANCE	850000
2	ABC	MALE	ADMIN	250000
3	PQR	FEMALE	SALES	350000
4	MNR	FEMALE	FINANCE	600000

1.3.3 Type #3 : Semi Structured

- Semi structured is the third type of big data. Semi-structured data can contain both the forms of data.
- Semi-structured data pertains to the data containing both the formats mentioned above, that is, structured and unstructured data.

- To be precise, it refers to the data that although has not been classified under a particular repository (database), yet contains vital information or tags that segregate individual elements within the data.
- Web application data, which is unstructured, consists of log files, transaction history files etc.
- Online transaction processing systems are built to work with structured data wherein data is stored in relations (tables).
- Semi-structured data is data that does not conform to a data model but has some structure. It lacks a fixed or rigid schema. It is the data that does not reside in a rational database but that have some organizational properties that make it easier to analyze. With some processes, we can store them in the relational database.

1.3.3(A) Characteristics of Semi-structured Data

1. Data does not conform to a data model but has some structure. Data can not be stored in the form of rows and columns as in Databases
2. Semi-structured data contains tags and elements (Metadata) which is used to group data and describe how the data is stored.
3. Similar entities are grouped together and organized in a hierarchy. Entities in the same group may or may not have the same attributes or properties.
4. Does not contain sufficient metadata which makes automation and management of data difficult.
5. Size and type of the same attributes in a group may differ.
6. Due to lack of a well-defined structure, it can not used by computer programs easily.

1.3.3(B) Sources of semi-structured Data

(1) E-mails
(2) XML and other markup languages
(3) Binary executables
(4) TCP/IP packets
(5) Zipped files
(6) Integration of data from different sources
(7) Web pages

1.3.3(C) Advantages and Disadvantages of Semi-structured Data

Advantages

1. The data is not constrained by a fixed schema.
2. Flexible i.e. Schema can be easily changed.
3. Data is portable.
4. It is possible to view structured data as semi-structured data.
5. Its supports users who can not express their need in SQL.
6. It can deal easily with the heterogeneity of sources.

Disadvantages

1. Lack of fixed, rigid schema make it difficult in storage of the data.
2. Interpreting the relationship between data is difficult as there is no separation of the schema and the data.
3. Queries are less efficient as compared to structured data.

Semi-structured - Example

- User can see semi-structured data as a structured in form but it is actually not defined with e.g. a table definition in relational DBMS.

- Personal data stored in a XML file :

```
<rec><name>Prashant
Rao</name><sex>Male</sex><age>35</age></rec><rec><
name>Seema
R.</name><sex>Female</sex><age>41</age></rec><rec>
<name>Satish
Mane</name><sex>Male</sex><age>29</age></rec>
```

1.4 DIFFERENCE BETWEEN STRUCTURED, SEMI-STRUCTURED AND UN-STRUCTURED DATA

GQ. What is difference between structured, semi-structured and Un-Structured Data ?

Properties	Structured data	Semi-structured data	Unstructured data
Technology	It is based on Relational database table	It is based on XML/RDF(Resource Description Framework).	It is based on character and binary data
Transaction management	Matured transaction and various concurrency techniques	Transaction is adapted from DBMS not matured	No transaction management and no concurrency
Version management	Versioning over tuples, row, tables	Versioning over tuples or graph is possible	Versioned as a whole
Flexibility	It is schema dependent and less flexible	It is more flexible than structured data but less flexible than unstructured data	It is more flexible and there is absence of schema

Properties	Structured data	Semi-structured data	Unstructured data
Scalability	It is very difficult to scale DB schema	It's scaling is simpler than structured data	It is more scalable.
Robustness	Very robust	New technology, not very spread	–
Query performance	Structured query allow complex joining	Queries over anonymous nodes are possible	Only textual queries are possible

1.5 TRADITIONAL VS. BIG DATA BUSINESS APPROACH

GQ. What is Traditional Data & Big Data ?

GQ. Explain in detail Traditional vs. Big Data Business Approach. **(5 Marks)**

1. Traditional Data

- Traditional data is the structured data which is being majorly maintained by all types of businesses starting from very small to big organizations.
- In traditional database system a centralized database architecture used to store and maintain the data in a fixed format or fields in a file. For managing and accessing the data Structured Query Language (SQL) is used.

2. Bigdata

- We can consider big data an upper version of traditional data. Big data deal with too large or complex data sets which is difficult to manage in traditional data-processing application software.

- It deals with large volume of both structured, semi structured and unstructured data. Volume, Velocity and Variety, Veracity and Value refer to the 5'V characteristics of big data.
- Big data not only refers to large amount of data it refers to extracting meaningful data by analyzing the huge amount of complex data sets.

UQ. Compare big data analytics with traditional data mining. **(MU - Dec. 18, 5 Marks)**

Sr. No.	Traditional Data	Big Data
1.	Traditional data is generated in enterprise level.	Big data is generated in outside and enterprise level.
2.	Its volume ranges from Gigabytes to Terabytes.	Its volume ranges from Petabytes to Zettabytes or Exabytes.
3.	Traditional database system deals with structured data.	Big data system deals with structured, semi structured and unstructured data.
4.	Traditional data is generated per hour or per day or more.	But big data is generated more frequently mainly per seconds.
5.	Traditional data source is centralized and it is managed in centralized form.	Big data source is distributed and it is managed in distributed form.
6.	Data integration is very easy.	Data integration is very difficult.
7.	Normal system configuration is capable to process traditional data.	High system configuration is required to process big data.

Sr. No.	Traditional Data	Big Data
8.	The size of the data is very small.	The size is more than the traditional data size.
9.	Traditional data base tools are required to perform any data base operation.	Special kind of data base tools are required to perform any data base operation.
10.	Normal functions can manipulate data.	Special kind of functions can manipulate data.
11.	Its data model is strict schema based and it is static.	Its data model is flat schema based and it is dynamic.
12.	Traditional data is stable and inter relationship.	Big data is not stable and unknown relationship.
13.	Traditional data is in manageable volume.	Big data is in huge volume which becomes unmanageable.
14.	It is easy to manage and manipulate the data.	It is difficult to manage and manipulate the data.
15.	Its data sources includes ERP transaction data, CRM transaction data, financial data, organizational data, web transaction data etc.	Its data sources includes social media, device data, sensor data, video, images, audio etc.

1.6 EXAMPLES OF BIG DATA APPLICATIONS

GQ. List the examples of big data. **(2 Marks)**

GQ. Explain the examples of big data. **(6 Marks)**

There are various big data applications as shown in Fig 1.6.1

1. Fraud detection

- Fraud detection is a Big Data application example for businesses which has operations like any type of claims or transaction processing.
- Number of times the detection of fraud is concluded long after the fact. At this point the damage has been already done all that's left is to decrease the harm and revise policies to prevent it in future.

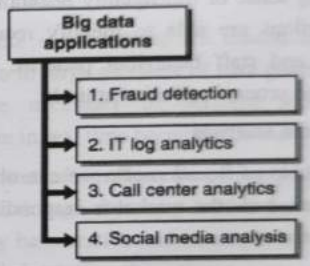


Fig. 1.6.1 : Big data applications

- The Big Data platforms can analyze claims and transactions of businesses. They identify large-scale patterns across many transactions or detect anomalous behaviour of a some user. This helps to avoid the fraud.

2. IT log analytics

- An enormous quantity of logs and trace data is generated in IT solutions and IT departments. Many times such data go unexamined: organizations simply don't have the manpower or resource to go through all such information.

- Big data has the ability to quickly identify large-scale patterns to help in diagnosing and preventing problems. It helps the organization with a large IT department.

3. Call center analytics

- Now we turn to the customer-facing Big Data application examples, of which call center analytics are particularly powerful. Without a Big Data solution, much of the insight that a call center can provide will be ignored or exposed later.
- By making sense of time/quality resolution metrics, the Big Data solutions are able to identify recurring problems or customer and staff behaviour patterns. Big data can also capture and process call content itself.

4. Social media analysis

- With the help of Social media we can observe the real-time insights into how the market is responding to products and campaigns.
- With the help of these insights, it is possible for companies to adjust their pricing, promotion, and campaign placement to get optimal results.

1.7 BIG DATA CHALLENGES

1. Sharing and Accessing Data

- Perhaps the most frequent challenge in big data efforts is the inaccessibility of data sets from external sources.
- Sharing data can cause substantial challenges.
- It include the need for inter and intra- institutional legal documents.
- Accessing data from public repositories leads to multiple difficulties.

- It is necessary for the data to be available in an accurate, complete and timely manner because if data in the companies information system is to be used to make accurate decisions in time then it becomes necessary for data to be available in this manner.

2. Privacy and Security

- It is another most important challenge with Big Data. This challenge includes sensitive, conceptual, technical as well as legal significance.
- Most of the organizations are unable to maintain regular checks due to large amounts of data generation. However, it should be necessary to perform security checks and observation in real time because it is most beneficial.
- There is some information of a person which when combined with external large data may lead to some facts of a person which may be secretive and he might not want the owner to know this information about that person.
- Some of the organization collects information of the people in order to add value to their business. This is done by making insights into their lives that they're unaware of.

3. Analytical Challenges

- There are some huge analytical challenges in big data which arise some main challenges questions like how to deal with a problem if data volume gets too large?
- Or how to find out the important data points?
- Or how to use data to the best advantage?
- These large amount of data on which these type of analysis is to be done can be structured (organized data), semi-structured (Semi-organized data) or unstructured (unorganized data).

There are two techniques through which decision making can be done :

1. Either incorporate massive data volumes in the analysis.
2. Or determine upfront which Big data is relevant.

4. Technical challenges

Quality of data

1. When there is a collection of a large amount of data and storage of this data, it comes at a cost. Big companies, business leaders and IT leaders always want large data storage.
2. For better results and conclusions, Big data rather than having irrelevant data, focuses on quality data storage.
3. This further arise a question that how it can be ensured that data is relevant, how much data would be enough for decision making and whether the stored data is accurate or not.

☞ **Fault tolerance**

1. Fault tolerance is another technical challenge and fault tolerance computing is extremely hard, involving intricate algorithms.
2. Nowadays some of the new technologies like cloud computing and big data always intended that whenever the failure occurs the damage done should be within the acceptable threshold that is the whole task should not begin from the scratch.

☞ **Scalability**

1. Big data projects can grow and evolve rapidly. The scalability issue of Big Data has lead towards cloud computing.
2. It leads to various challenges like how to run and execute various jobs so that goal of each workload can be achieved cost-effectively.

3. It also requires dealing with the system failures in an efficient manner. This leads to a big question again that what kinds of storage devices are to be used.

1.8 EXAMPLES OF BIG DATA IN REAL LIFE

(1) In the Education Industry

The University of Alabama has more than 38,000 students and an ocean of data. In the past when there were no real solutions to analyze that much data, some of them seemed useless. Now, administrators can use analytics and data visualizations for this data to draw out patterns of students revolutionizing the university's operations, recruitment, and retention efforts.

(2) In the Healthcare

Wearable devices and sensors have been introduced in the healthcare industry which can provide real-time feed to the electronic health record of a patient. One such technology is Apple.

Apple has come up with Apple HealthKit, CareKit, and ResearchKit. The main goal is to empower iPhone users to store and access their real-time health records on their phones.

(3) In Government Sector

Food and Drug Administration (FDA) which runs under the jurisdiction of the Federal Government of the USA leverages the analysis of big data to discover patterns and associations to identify and examine the expected or unexpected occurrences of food-based infections.

(4) In Media and Entertainment Industry

Spotify, on-demand music-providing platform, uses Big Data Analytics, collects data from all its users around the globe, and then uses the analyzed data to give informed music recommendations and suggestions to every individual user.

Amazon Prime which offers, videos, music, and Kindle books in a one-stop shop is also big on using big data.

(5) In Weather Patterns

IBM Deep Thunder, which is a research project by IBM, provides weather forecasting through high-performance computing of big data. IBM is also assisting Tokyo with improved weather forecasting for natural disasters or predicting the probability of damaged power lines.

(6) In Transportation Industry

Uber generates and uses a huge amount of data regarding drivers, their vehicles, lo1cations, every trip from every vehicle, etc. All this data is analyzed and then used to predict supply, demand, location of drivers, and fares that will be set for every trip.

(7) In Banking Sector

Various anti-money laundering software such as SAS AML uses Data Analytics in Banking to detect suspicious transactions and analyze customer data. Bank of America has been a SAS AML customer for more than 25 years.

(8) In Marketing

Amazon collected data about the purchase done by millions of people around the world. They analyzed the purchase patterns and payment methods used by the customers and used the results to design new offers and advertisements.

(9) In Business Sights

Netflix is using Big Data to understand the user behavior, the type of content they like, popular movies on the website, similar content that can suggest to the user, and which series or movies should they invest in.

(10) In Space Sector

NASA is collecting data from different satellites and rovers about the geography, atmospheric conditions, and other factors of mars for their upcoming mission. It uses big data to manage all that data and analyzes that to run simulations.

Chapter Ends...

❑❑❑

UNIT II

CHAPTER 2

Introduction to Big Data Frameworks

University Prescribed Syllabus

What is Hadoop? Core Hadoop Components; Hadoop Ecosystem; Working with Apache Spark What is NoSQL? NoSQL data architecture patterns : Keyvalue stores, Graph stores, Column family (Bigtable) stores, Document stores, MongoDB

Self-learning Topics : HDFS vs GFS, MongoDB vs other NoSQL system, Implementation of Apache Spark.

2.1 CONCEPT OF HADOOP

2.1.1 What is Hadoop ?

Hadoop is an open-source software Platform for storing massive volumes of data and running applications on clusters (groups) of commodity software. It gives us the massive data storage capability, massive computational power and the ability to handle different virtually limitless jobs that can be a running job, waiting jobs or tasks. Its main essential component is to support growing big data technologies, thereby support forward-thinking analytics like Predictive analytics, Machine learning and data mining. Hadoop has the capability to handle different modes of data such as structured,

unstructured and semi-structured data. It gives us the elasticity to collect, process, and investigate data that the old data warehouses concept failed to do.

2.1.2 History of Hadoop

- The Hadoop was introduced by Doug Cutting and Mike Cafarella in 2002. Its beginning was the Google File System paper, printed by Google.
- In the year 2002, Doug Cutting and Mike Cafarella started to work on a project of Apache Nutch. It is an open source i.e. free web crawler software project.
- While working on Apache Nutch, they were facing some issue with big data. To store that data, they have invested lot of money which becomes the challenging of that project for completion.
- Due to this problem appearance of Hadoop came into existence.
- In 2003, Google presented a file system known as GFS (Google file system). It is a registered distributed file system developed to provide effective access to data.
- In year 2004, Google released the concept of a white paper on Map Reduce.
- This technique makes simpler the data processing on large clusters (groups).
- In 2005, Doug Cutting and Mike Cafarella presented a new file system known as NDFS (Nutch Distributed File System), this file system also contains Map reduce. In 2006, Doug Cutting resign Google and joined Yahoo. Based on the Nutch project, Dough Cutting announce a new project Hadoop with a file system known as HDFS (Hadoop Distributed File System).

- Hadoop first version 0.1.0 was released and Doug Cutting gave named his project Hadoop after his son's toy elephant. In 2007, Yahoo successfully run two clusters of 1000 machines.
- In 2008, Hadoop became the quickest system to sort 1 terabyte of data on a 900-node cluster in 209 seconds. In 2013, Hadoop 2.2 was released. And Currently In 2017, Hadoop 3.0 was released.

2.1.3 Features of Hadoop

1. **Suitable for Big Data Analysis :** As Big Data manages to be distributed and unstructured in nature, Hadoop clusters are well-matched for analysis of Big Data. Meanwhile it is processing logic (not the actual data) that flows to the computing nodes and less network bandwidth is spent. This concept is called as data locality which helps to increase the productivity of Hadoop based applications.
2. **Scalability :** Hadoop clusters can easily be scaled to any amount by adding extra cluster nodes and thus allows for the growth of Big Data. Also, scaling does not require adjustments to application logic.
3. **Fault Tolerance :** Hadoop network has a facility to duplicate the input data on to other cluster nodes. So, in the event of a cluster node failure the data processing can still process data by using data stored on another cluster node.

2.1.4 Advantages of Hadoop

1. **Fast :** In HDFS the data distributed over the cluster and mapped such a way which helps in faster recovery. Even the tools to process the data are often on the same servers, thus reducing the processing time can be efficient way to manage the data. It also processes terabytes of data in minutes and Peta bytes in hours.

2. **Scalable :** Hadoop cluster can be extended by just adding nodes in the cluster so failure chance can be less.
3. **Cost Effective :** Hadoop is open source and uses commodity hardware to store data, it is cheaper as compared to traditional RDMS.
4. **Tough to failure :** HDFS has the property with which it can duplicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the backup data and use it. Normally, data are replicated thrice but the replication factor is configurable.

2.1.5 Challenges of Hadoop

1. Hadoop is a complex distributed system with low-level Application programming interface.
2. Specialized skills are required for using Hadoop and prevent most developers from efficiently bringing solutions.
3. Business logic and infrastructure APIs have no clear separation therefore burdening come on app developers.
4. Automated testing of end-to-end solutions is unfeasible or terrible.
5. Common data patterns often require but does not support data steadiness and accuracy.
6. Hadoop is more than just disconnected storage.
7. Hadoop is a various collection of many open source projects.
8. Understanding multiple technologies and hand-coding combination between them is difficult.
9. Significant effort is wasted on simple tasks like data absorptions and ETL (Extract, Transform, Load).

10. Real-time and batch ingestion requires extremely integration numerous components.
11. Different processing models require data to be stored in specific ways so that data can be handle easily.

2.1.6 Architecture of Hadoop

UQ. Explain the physical architecture of Hadoop.
(MU - Dec. 18, 4 Marks)

Hadoop basically has Master-Slave Architecture for storing data and distributed processing of data by using MapReduce and HDFS methods. In Hadoop, master or slave system can be set up on the cloud or on premise.

1. **NameNode :** NameNode represents all files and directory which is used in the namespace.
2. **DataNode :** DataNode helps you to manage the states of an HDFS node and allows you to cooperate with its blocks.
3. **Master Node :** The master node allows you to conduct parallel processing of data by use Hadoop MapReduce.
4. **Slave node :** The slave nodes are the supplementary machines in the Hadoop cluster which permits you to store data to conduct complex calculations. And all these slave node derives with Task Tracker and DataNode which allows to synchronize the processes with the NameNode and Job Tracker correspondingly.

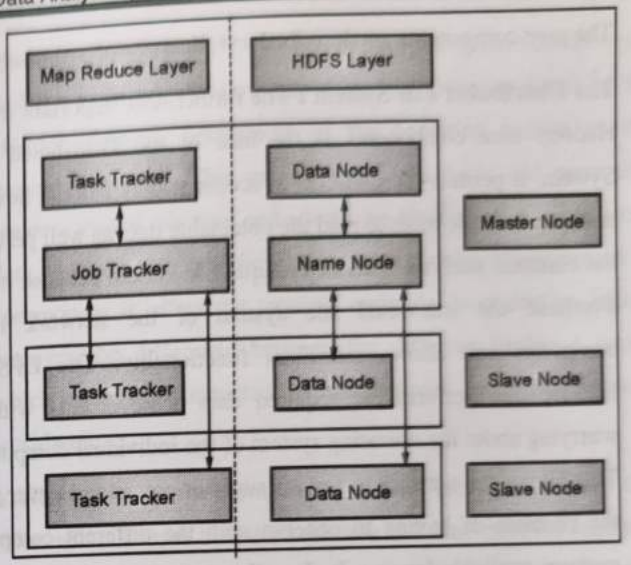


Fig. 2.1.1

2.2 CORE HADOOP COMPONENTS, HADOOP ECOSYSTEM

2.2.1 Core Hadoop Components

UQ. Explain Hadoop ecosystem with core components.
(MU - Dec. 18, 4 Marks)

Hadoop concept is an open-source big data technology platform which allows computer networks to perform complex processing and gives results that are forever available even when a condition arises when a few nodes are in unavailable state for functional processing.

There are a few important Hadoop core components that it can perform through several cloud-based platforms.

The core components are described as follows :

1. **The Distributed File System :** The furthermost important of the Hadoop core components is the idea of the Distributed File System. It permits the platform to access widely storage devices and use the basic tools to read the obtainable data as well perform the essential analysis. This is a unique file system because it lies overhead the individual file system of the network node computers and allows matchless functionality. The DFS of Hadoop can perform the required data achievements without worrying about the operating system of the individual computers. This allows the network to service superior power and never face the problem of having to observe with the different computer systems available for use. It also allows the connection to other central components, such as MapReduce.

2. **MapReduce :** MapReduce is another of Hadoop core components that trusts two separate functions, which are required for execution of smart big data operations. The first operation is to read the data from a database and doing inserting operation of data it in a suitable format for performing the required analysis. This is the function which is known as a mapping activity. It basically allows a platform to make the data for the analytical requirements in a common format to allow any computer to do further procedures as per requirements. The next method carried out is mathematical operation. This operation is named as reduction (decrease), because it usually reduces the available map to a set of proper values. Together, these functions are existing in a single module and perform the whole operation which delivers information from available different data sources.

3. **Hadoop Common :** It is also one of the Hadoop core components and the tools which allow any computer to become part of the Hadoop network unrelatedly of the operating system or the present hardware. This module uses Java tools and parts that create a virtual machine and allows the Hadoop platform to store data under its path specific file system. This component named as common as it offers the required common functionality, which removes the difference between the different hardware nodes which may be connected to the network at any time and worldwide.

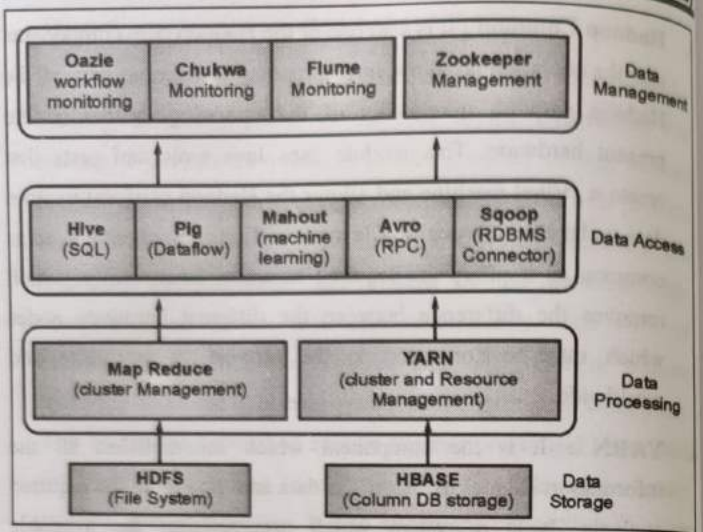
4. **YARN :** It is the component which accomplishes all the information sources that store the data and then route the required analysis. It is a system which accomplishes the available resources in a network group, as well as schedule the processing tasks to come up with a clever solution for every big data want on the system.

2.2.2 Hadoop Ecosystem Overview

UQ. How big data problems are handled by Hadoop system ?
(MU - May 19, 5 Marks)

Hadoop ecosystem is a platform or framework which assistances in solving the big data problems. It includes different components and services (ingesting, storing, analysing, and maintaining) inside of it.

Most of the services available in the Hadoop ecosystem which contains the main four core components of Hadoop which include HDFS, YARN, MapReduce and Common. Hadoop ecosystem contains both Apache Open Source projects and other wide variation of marketable tools and solutions.



(1A9)**Fig. 2.2.1 : Hadoop Ecosystem**

Some of the open source examples are Spark, Hive, Pig, Sqoop and Oozie. As we have got some idea about what Hadoop ecosystem is, what it does, and what are its components, let's discuss each concept in detail.

☞ Components of Hadoop Ecosystem

As we have seen an overview of Hadoop Ecosystem and well-known open-source examples, now we are going to discuss deeply the list of Hadoop Components individually and their specific roles in the big data processing. The components of Hadoop ecosystems are :

1. **HDFS :** Hadoop Distributed File System is the spine of Hadoop which runs on java language and allows to stores data in Hadoop applications. They act as a command interface to relate with Hadoop. There are two components of HDFS that are Data node and Name Node. Name node is the main node which manages file systems and activates all data nodes and maintains records of

metadata updating. In some case of deletion of data, they automatically record in its Edit Log file. Data Node requires massive storage space due to the operation of reading and writing. They work according to the orders of the Name Node. The data nodes are hardware in the distributed system.

2. **HBASE :** It is an open-source framework storing all types of data and doesn't support the SQL database. They run on top of HDFS and they are written in java language. Most of the companies use them for capturing the features like supporting all types of data, high security, use of HBase tables etc. They play a dynamic role in analytical processing. The two major components of HBase are HBase master and Regional Server. The HBase master is answerable for load balancing in a Hadoop cluster and controls the failure occurred. They are answerable for performing management role. The role of the regional server would be a worker node and responsible for reading, writing data in the cache.
3. **YARN :** It is an important component in the ecosystem and named as operating system in Hadoop which delivers resource management and job scheduling task. The components are Resource and Node manager, Application manager and container. They also act as protectors across Hadoop clusters. They help in the dynamic allocation of cluster resources, increase in data centre process and allows multiple access engines.
4. **Sqoop :** It is a tool that helps in data transfer between HDFS and MySQL and gives hand-on to import and export of data and as well they have a connector for fetching and linking a data.
5. **Apache Spark :** It is an open-source cluster computing framework for data analytics and an important data processing engine. It is written in Scala and comes with packaged standard libraries. They are also used by many companies for their high processing speed and stream processing.

6. **Apache Flume :** It is a distributed service collecting a huge quantity of data from the source (web server) and moves back to its source and relocated to HDFS. It has its own three components are Source, sink, and channel.
7. **Apache Pig :** Data Handling of Hadoop is performed by Apache Pig and by using of Pig Latin Language. It helps in the reuse of code and easy to read and write code.
8. **Hive :** It is an open-source Platform software for execution of data warehousing ideas, it accomplishes to query for large data sets stored in HDFS. It is built on upper of the Hadoop Ecosystem, the language used is Hive Query language. The user submits the hive queries with metadata which converts SQL into Map-reduce jobs and directs to the Hadoop cluster which consists of one master and many slaves.
9. **Apache Drill :** Apache Drill is an open-source SQL engine which procedures on non-relational databases and File system. They are intended to support Semi-structured databases originate in Cloud storage. They have good Memory management abilities to maintain junk collection
10. **Apache Zookeeper :** It is an API that supports in distributed Organisation. Here a node called Z node which is created by an application in the Hadoop cluster. They do services like Synchronization, Configuration etc. Its categories out the time-consuming coordination in the Hadoop Ecosystem.
11. **Oozie :** Oozie is a java web application which maintains several workflows in a Hadoop cluster. Having Web service APIs controls over a job is done anywhere. It is widespread for handling Multiple jobs successfully.

2.2.3 Examples of Hadoop Ecosystem

Regarding map-reduce, we can see an example and use case. One such case is Skybox which uses Hadoop to analyse a huge volume of data. Hive can find simplicity on Facebook. Frequency of word count in a sentence using map-reduce. MAP performs by taking the count as input and perform functions such as Filtering and sorting and the reduce () consolidates the result. Hive example on taking students from different states from student databases using various DML commands.

2.2.4 Limitations

UQ. State limitations of Hadoop ecosystem. **(MU - Dec. 18, 2 Marks)**

1. Issue with Small Files
2. Slow Processing Speed
3. Support for Batch Processing only
4. No Real-time Data Processing
5. No Delta Iteration
6. Latency
7. Security
8. No Abstraction
9. No Caching
10. Lengthy Line of Code

1. Issue with Small Files

Hadoop is not suitable for the small data. **(HDFS) Hadoop distributed file system** wants the capability to professionally support the arbitrary reading of the small files since of it is high volume design. Small files are the main problematic in HDFS. A small file is expressively minor than the HDFS block size (default 128MB). If we are storing these vast numbers of small files, then HDFS cannot handle these files while HDFS is working accurately with a small unit of large files by storing large data

sets rather than storing several small files. If there are several small files, then the **NameNode** will get burden meanwhile it stores the namespace of HDFS.

2. Slow Processing Speed

In Hadoop, with a support of the parallel and distributed algorithm the MapReduce procedure the large data sets. There are some tasks that we need to perform like Map and Reduce and thus the MapReduce needs a lot of time to complete these tasks thus by increasing the time interval. The Data is spread and handled over the cluster in MapReduce which increases the period and decreases the handling and execution speed.

3. Support for Batch Processing only

Hadoop supports the batch processing only and it does not procedure the streamed data and later complete performance is slower. The MapReduce framework of the Hadoop does not influence the memory of the **Hadoop cluster** to the extreme level.

4. No Real-time Data Processing

Apache Hadoop is for the operation of batch processing, which allow it to take a vast amount of data in input and execute it and generate the outcome. Even though batch processing is very well-organized for processing a data of high volume dependent on the size of the data that is being processes and the computational power of the system but basically an output can be delay so the Hadoop is not appropriate for Real-time data processing.

5. No Delta Iteration

Hadoop isn't well-organized for the constant processing basically the Hadoop doesn't support the repeated data flow i.e. sequence of phases during which the respective output of the earlier phase is the input to the succeeding phase.

6. Latency

The Hadoop MapReduce framework is that the comparatively slower so meanwhile its supporting the various format, structure and vast capacity of information or data. In **MapReduce**, Map takes the collection of the data and decodes it into the alternative set of data where the separate elements are fragmented down into **key-value pairs** and reduce the output from the map as input and process extra and MapReduce requires plenty of the time to accomplish these tasks thus by increasing the latency.

7. Security

Hadoop is challenges in handling the compound application. If the user doesn't know the way to enable a platform who is managing the platform that the data can be in the danger.

At storage and network levels, Hadoop is missing the encryption part, which can leads to the major point of concern. Hadoop supports **Kerberos authentication**, which is tough to manage. HDFS **supports access control lists** (ACLs) and a standard file permissions model. However, third-party vendors have enabled an association to influence the **Active Directory Kerberos** and **LDAP** for verification.

8. No Abstraction

Hadoop doesn't have any kind of abstraction; thus, MapReduce creators need to the hand code for each process which makes it difficult to work.

9. No Caching

Hadoop is not well-organized for caching. In Hadoop, MapReduce cannot store the intermediate data in the memory for a additional condition which reduces the performance of the Hadoop.

10. Lengthy Line of Code

Hadoop has approximately about 1,20,000 codes of line, the number of lines generated by the bugs and it will take more period of the time to execute the program.

UQ. Explain how Hadoop goals are covered in Hadoop distributed file system. **(MU - May 19, 10 Marks)**

HDFS and MapReduce are two important major components of Hadoop, whereas HDFS is useful for infrastructure point of view and MapReduce is useful for programming concept. To understand the concept behind scalability of Hadoop from single-node to a thousand-nodes cluster, HDFS is very useful. It covered the goals of Hadoop as follows :

1. **Handling of large dataset :** As Hadoop supports distributed storage and processing of large data set, HDFS architecture is designed as it most useful to store and retrieve large data.
2. **Fault tolerance and data replication :** In HDFS, the data files are divided into big blocks of data and for fault tolerance each one block is stored on three nodes from which two nodes are from same rack and one is from a different rack. A block is considered as the amount of data stored on each data node. The redundancy of data leads to robustness, fault detection, quick recovery of data and scalability.
3. **Commodity Hardware :** HDFS consider that the cluster will run on common hardware such as less expensive or ordinary machines. And an important feature of Hadoop is that Hadoop can be installed on any average commodity hardware. For installation and execution of Hadoop It does not require any super computers or high-end hardware. This reduces the overall cost.
4. **Data Locality concept :** Data locality means locating computation logic near to the data, instead of moving data to the

computation logic or application space . This reduces the bandwidth utilization in a system. HDFS provide interfaces for applications to relocate themselves closer to the location where data.

UQ. How big data analytics can be useful in the development of Digital India ? **(MU - Dec. 18, 5 Marks)**

- "Digital India" program is get introduce in in our country with the vision to make over India into a digitally empowered society and build knowledge economy.

  Its vision is mostly focused on three main areas :

  (i) Digital Infrastructure as a basic utility to every citizen,
  (ii) Governance and services on demand and
  (iii) Digital empowerment of citizens of India.
- Subsequently, to fulfil this vision, availability of different Data resources has got increased.
- In Big data huge amount of data collected and stored whether it is in structured or unstructured form or semi-structured. This data may contains various business related transactions, email, images, audio, surveillance camera videos, logs and unstructured data from blogs or messages from social media, medical data, banks related transaction data, e-Governance data, media data, defence related data, IT sectors.
- If this data get efficiently cleaned and then get analyzed, it can helpful in data visualization for business trade for various enterprises or organizations.
- This digital technology has make the progress enterprises or organizations more easier. Data from collected from tweets, various blogs and other social networks sites can useful to an enterprise or organization to analyze consumer's view. It will helpful for them to understand needs and choices of their customers.

- The big data supports five V"s attributes : Volume, Velocity, Variety, Veracity and Values.

1. **Volume :** Means the amount of data it contains. Now in this digital world , organization may have huge amount of data and to deal with this huge data big data uses the concept of distributed systems so that they can store their data and analyse it through-out databases that are scattered around the world and linked via internet
2. **Velocity :** In big data refers to the data transfer from various digital platform such as online systems, various sensors, social media, live web capture, etc. As we all known that social media messages get viral in seconds of time. Such scenario of big data represents to its velocity. In some cases, it needs to be analysed the data without storing it.
3. **Variety :** Refers to different types of data from many sources, it may be in structured, unstructured or semi-structured. Big data analytics provides the facility to integrate this data.
4. **Veracity :** Means Complexity which indicate that the data must be able to transfer via different multiple data centers such as the cloud and geographical zones. Managing or analyzing this huge data is very complex task.
5. **Value :** Is measure of visible or invisible benefits gained by organisation by the use of Big Data. It is more useful when the data from Big Data is converted into value then it gets used.

2.3 WORKING WITH APACHE SPARK

- First, Spark is intended to enhance, not replace, the Hadoop stack. From day one, Spark was designed to read and write data from and to HDFS, as well as other storage systems, such as HBase and Amazon's S3. As such, Hadoop users can enrich their processing capabilities by combining Spark with Hadoop MapReduce, HBase, and other big data frameworks.
- Second, we have constantly focused on making it as easy as possible for every Hadoop user to take advantage of Spark's capabilities. No matter whether you run Hadoop 1.x or Hadoop 2.0 (YARN), and no matter whether you have administrative privileges to configure the Hadoop cluster or not, there is a way for you to run Spark! In particular, there are three ways to deploy Spark in a Hadoop cluster: standalone, YARN, and SIMR.

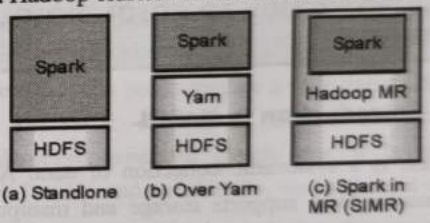


Fig. 2.3.1

- **Standalone deployment :** With the standalone deployment one can statically allocate resources on all or a subset of machines in a Hadoop cluster and run Spark side by side with Hadoop MR. The user can then run arbitrary Spark jobs on her HDFS data. Its simplicity makes this the deployment of choice for many Hadoop 1.x users.

- **Hadoop Yarn deployment :** Hadoop users who have already deployed or are planning to deploy Hadoop Yarn can simply run Spark on YARN without any pre-installation or administrative access required. This allows users to easily integrate Spark in their Hadoop stack and take advantage of the full power of Spark, as well as of other components running on top of Spark.
- Spark In MapReduce (SIMR): For the Hadoop users that are not running YARN yet, another option, in addition to the standalone deployment, is to use SIMR to launch Spark jobs inside MapReduce. With SIMR, users can start experimenting with Spark and use its shell within a couple of minutes after downloading it! This tremendously lowers the barrier of deployment, and lets virtually everyone play with Spark.

2.4 INTRODUCTION TO NOSQL, NOSQL BUSINESS DRIVERS

2.4.1 Introduction to NoSQL

- A database is a systematic collection of data. And a database management system supports storage and manipulation of data which makes data management easy. For example, an online telephone directory uses a database to store data of people like phone numbers and other contact details that can be used by service provider to manage billing client related issues and handle fall data etc. That means A database management system provides the mechanism to store and retrieve the data.

There are different kinds of management systems :

RDBMS	OLAP	NoSQL
(Relational Database Management System)	(Online Analytical Processing)	(Not only SQL)

- NoSQL refers to all databases and data stores that are not based on the relational database management system or RDBMS principles. NoSQL are the new set of databases that has emerged recent past as an alternative solution to relational databases.

Carl Strozzi introduced the term NoSQL to name his file-based database in 1998.

- NoSQL does not represent single product or technology but it represents a group products and various related data concepts for storage and management. NoSQL is an approach to database management that can accommodate a wide variety of data models including key-value, document, column and graph formats. NoSQL database generally means that it is non-relational, distributed, flexible and scalable. So, we can bind it up as :
- NoSQL an approach to database design that provides flexible schemas for the storage and retrieval of data beyond the traditional table structures found in relational databases.
- It relates to large data sets accessed and manipulated on a Web scale.

2.4.2 Brief History of NoSQL Databases

- 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database.
- 2000- Graph database Neo4j is launched.

- 2004- Google BigTable is launched.
- 2005- CouchDB is launched.
- 2007- The research paper on Amazon Dynamo is released.
- 2008- Facebooks open sources the Cassandra project.
- 2009- The term NoSQL was reintroduced.

2.4.3 Why NoSQL?

The concept of NoSQL databases became popular with internet giants like Google, Facebook, Amazon etc. Who deals with huge volumes of data the system response time becomes slow when we use RDBMS for massive volumes of data so to resolve this problem, we could scale up our system by upgrading our existing hardware but this process is an expensive. So alternative for this issue is to distribute database load on multiple hosts whenever the load increases this method known as scaling out.

NoSQL databases are non-relational so they scale-out better than relational databases. As they designed with the web applications in mind. Now NoSQL database is exactly type pf database that can handle all sorts of semi structured data, unstructured data, rapidly changing data and bigdata. So, to resolve the problems related to large volume and semi structured data. NoSQL databases have emerged.

2.4.4 CAP Theorem

GQ. What is CAP Theorem? How it is applicable to NOSQL systems ? **(4 Marks)**

It plays important role in NoSQL databases. CAP theorem is also called brewer's theorem which states that it is impossible for a distributed data store to offer more than two out of three guarantees :

So basically, some NoSQL databases offer consistency and partition tolerance. While some offer availability and partition tolerance. But partition tolerance is common as NoSQL databases are distributed in nature so based on requirement, we can choose NoSQL database has to be used. Different types of NoSQL databases are available based on data models.

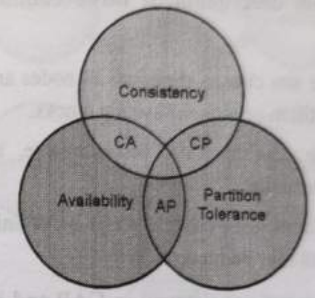


Fig. 2.4.1 : CAP Property

Consistency

- This means that the data in the database remains consistent after the execution of an operation.
- For example, after an update operation all clients see the same data.

Availability

This means that the system is always on (service guarantee availability), no downtime.

Partition Tolerance

- This means that the system continues to function even the communication among the servers is unreliable, i.e., the servers may be partitioned into multiple groups that cannot communicate with one another.

- In theoretically it is impossible to fulfil all 3 requirements. CAP provides the basic requirements for a distributed system to follow 2 of the 3 requirements. Therefore, all the current NoSQL database follow the different combinations of the C, A, P from the CAP theorem.
- Here is the brief description of **three combinations CA, CP, AP :**
  - **CA** - Single site cluster, therefore all nodes are always in contact. When a partition occurs, the system blocks.
  - **CP** - Some data may not be accessible, but the rest is still consistent/accurate.
  - **AP** - System is still available under partitioning, but some of the data returned may be inaccurate.

The use of the word consistency in CAP and its use in ACID do not refer to the same identical concept.

In CAP, the term consistency refers to the consistency of the values in different copies of the same data item in a replicated distributed system. In ACID, it refers to the fact that a transaction will not violate the integrity constraints specified on the database schema

2.4.5 Characteristics / Features of NoSQL

UQ. Describe characteristics of a NoSQL database.
MU- Dec 17, 10 Marks

1. Non-relational

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization

- No complex features like query languages, query planners, referential integrity joins, ACID

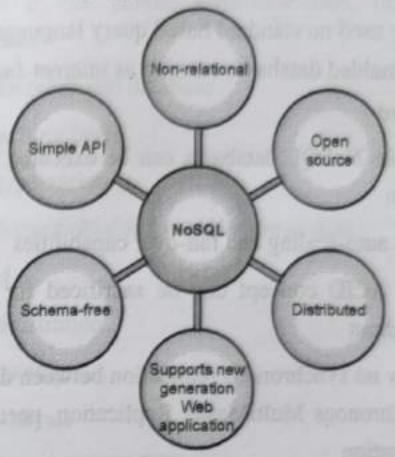


Fig. 2.4.2

2. Open-source

NoSQL databases don't require expensive licensing fees and can run on inexpensive hardware, rendering their deployment cost-effective.

3. Schema-free

- NoSQL databases are either schema-free or have relaxed schemas
- Do not require any sort of definition of the schema of the data
- Offers heterogeneous structures of data in the same domain

4. Simple API

- Offers easy to use interfaces for storage and querying data provided
- APIs allow low-level data manipulation & selection methods

- Text-based protocols mostly used with HTTP REST with JSON
- Mostly used no standard based query language
- Web-enabled databases running as internet-facing services.

5. Distributed

- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes Asynchronous MultiMaster Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher distribution.

2.4.6 Advantages and Disadvantages of NoSQL

Advantages of NoSQL

1. Scale(horizontal)
2. SQL databases are vertically scalable. This means that you can increase the load on a single server by increasing things like RAM, CPU or SSD. But on the other hand, NoSQL databases are horizontally scalable. This means that you handle more traffic by sharding, or adding more servers in your NoSQL database
3. Simple data model (fewer joins)
4. Streaming/ volume
5. Reliability
6. Schema-less (no modelling or prototyping)

7. Rapid development
8. Flexible as it can handle semi-structured, unstructured and structured data.
9. Cheaper than relational database
10. Creates a caching layer
11. Wide data type variety
12. Uses large binary objects for storing large data
13. Bulk upload
14. Lower administration
15. Distributed storage
16. Real-time analysis

Disadvantages of NoSQL

1. ACID transactions
2. Cannot use SQL
3. Cannot perform searches
4. Data loss
5. No referential integrity
6. Lack of availability of expertise

2.4.7 Difference between RDBMS and NoSQL

UQ. Differentiate between a RDBMS and NoSQL database.
MU- Dec-18, 10 Marks

Sr. No.	RDBMS	NoSQL
1.	Have fixed or static or predefined schema	Have dynamic schema
2.	Not suited for hierarchical data storage	Best suited for hierarchical data storage
3.	Vertically scalable	Horizontally scalable

Sr. No.	RDBMS	NoSQL
4.	Follow ACID property	Follows CAP (consistency, availability, partition tolerance)
5.	Relational Database supports transactions (also complex transactions with joins).	NoSQL databases don't support transactions (support only simple transactions).
6.	Relational database manages only structured data.	NoSQL database can manage structured, unstructured and semi-structured data.
7.	Relational databases have a single point of failure with failover.	NoSQL databases have no single point of failure.
8.	Relational Database supports a powerful query language.	NoSQL Database supports a very simple query language.
9.	It gives only read scalability.	It gives both read and write scalability.
10.	Transactions written in one location.	Transactions written in many locations.
11.	It supports complex transactions.	It supports simple transactions.
12.	It is used to handle data coming in low velocity.	It is used to handle data coming in high velocity.
13.	**Examples** : MySQL, Oracle, Sqlite, PostgreSQL and MS-SQL etc.	**Examples** : MongoDB, BigTable, Redis, RavenDB, Cassandra, Hbase, Neo4j, CouchDB etc.

2.4.8 NoSQL Business Drivers

The scientist-philosopher Thomas Kuhn coined the term paradigm shift to identify a recurring process. He observed that in science, where innovative ideas came in bursts and impacted the world in nonlinear ways. We'll use Kuhn's concept of the paradigm shift as a way to think about and explain the NoSQL movement and the changes in thought patterns, architectures, and methods emerging today.

Many organizations are supporting to the single-CPU relational systems that have fulfilled the needs of their organizations as per the requirements.

Businesses have initiated the value in fast catching and examining huge quantity of adjustable data and making direct changes in their businesses based on the data that they obtain. As all of these drivers applies burden on single-processor relational model, its basis suits less steady and in time no extended encounters the organization's needs.

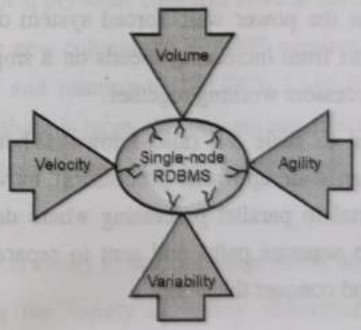


Fig. 2.4.3

Fig. 2.4.3 In this, we see how the business drivers volume, speed, variability, and agility create pressure on a single CPU system, resulting in cracks. Volume and velocity state to the capability to

handle and manage the big datasets that appears early. Variability states to how various data types do not fit within the structured tables, and agility states to how much fast an organization replies to the business modification.

There are 4 major business drivers for NoSQL as :

(1)	Volume	(2)	Velocity
(3)	Variability	(4)	Agility

1. Volume

- Undoubtedly, the main factor forcing organizations to look at alternatives to their existing RDBMS is to investigate big data using clusters of commodity processors.
- Until around 2005, performance problems were eliminated by purchasing faster processors. Over time, the ability to speed up the process is no longer an option.
- As the chip density increases, the heat can no longer dissolve rapidly enough without chip overheating. This phenomenon, known as the power wall, forced system designers to shift their focus from increasing speeds on a single chip to using more processors working together.
- The need to scale out (also known as horizontal scaling), rather than scale up (faster processors), moved organizations from serial to parallel processing where data problems are split into separate paths and sent to separate processors to divide and conquer the work.

2. Velocity

- While large data issues are considered for many organizations moving away from RDBMSs, the ability of a single processor system to read and write data quickly is also important.

- Many single-processor RDBMS cannot meet the demand for online queries on databases created by real-time insert and public-facing websites.
- RDBMSs frequently index multiple columns of each new row, a process that reduces system performance.
- When single-processor RDBMSs are used as a back-end in front of a web store, random outbursts in web traffic reduce the response for everyone, and tuning the system can be expensive when both high read and write throughput is required.

3. Variability

- Companies that seek to capture and report exceptional data conflicts when attempting to use the rigorous database schema structure imposed by RDBMSs. For example, if a business unit wants to capture some custom fields for a specific customer, it must store this information even if it does not apply to all customer rows in the database.
- Adding new columns to RDBMS requires shutting down the system and running the ALTER TABLE command. When the database is large, this process can affect the availability of the system, costing time and money.

4. Agility

- Agility is ability to accept change easily and quickly.
- Among the variety of agility dimensions such as model agility (ease and speed of changing data modules), operational agility (ease models), operational agility (ease and speed of changing operational aspects), and programming agility (ease and speed of application development) – one of the most important is the ability to

quickly and seamlessly scale an application to accommodate large amounts of data, users and connections.

- The most complex part of building applications using RDBMS is the process of putting data into and getting data out of the database.
- If your data has nested and repeated subgroups of data structures, you need to include an object-relational mapping layer.
- The responsibility of this layer is to generate the correct combination of Insert, update, delete and select SQL statement to move object data to and from the RDBMS persistence layer.
- This process is not simple and is associated with the largest barrier to rapid change when developing new or modifying existing applications.
- Generally, object relational mapping requires experienced software developers.
- Even with experienced staff, small change requests can cause slowdowns in development and testing schedules.
- All these hurdles are best overcome by NOSQL database.
- These databases are schematic and can be scaled down easily.
- They can accommodate application changes easily and can handle any volume of data efficiently.
- This agility has become business driver for NOSQL databases.

2.5 NOSQL DATA ARCHITECTURE PATTERNS

UQ. What are the different architectural patterns in NoSQL? Explain Graph data store and Column Family Store patterns with relevant examples.

NoSQL databases were born out of the rigidity of traditional relational or SQL databases, which use tables, columns, and rows to establish relationships across data. Developers welcomed NoSQL databases because they didn't require an upfront schema design; they were able to go straight to development. And it's this flexibility, this "ad-hoc" approach to organizing data, that has arguably been NoSQL's greatest selling point, which continues to appeal to organizations that need to store, retrieve, and analyze either unstructured or rapidly changing data.

The data stored in NoSQL follows any of the four data architecture patterns.

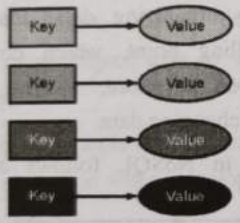
(A) Key-Value Stores
(B) Column family (Bigtable) Stores
(C) Document Stores
(D) Graph Stores

2.5.1 Key-Value Stores

UQ. Explain in detail key-value store NoSQL architectural pattern. Identify two applications that can use thispattern. **(MU - May 18, 5 Marks)**

- One of the most basic NoSQL database models is this model. The data is collected in the pattern of Key-Value Pairs, as the name implies. A series of strings, integers or characters is typically the key, but it can also be a more advanced form of data.

- Usually, the value is connected or co-related to the key. The databases for key-value pair storage typically store information as a hash table where each key is unique.
- The value may be of any form (JavaScript Object Notation (JSON), Binary Large Object (BLOB), strings, etc.).
- **Application :** This style of architecture is commonly used in shopping websites or e-commerce applications and its important assets is its ability for wide management of data volumes, heavy loads and the ease with which keys are used to retrieve data.



Key	Value
user-123	"John Doe"
image-123.jpg	<binary image file>
http://webpage-123.html	<web page html>
file-123.pdf	<pdf document>

Fig. 2.5.1 : An example of Key-Value

Keys and values are flexible. Keys can be image names, web page URLs, or file path names that point to values like binary images, HTML web pages, and PDF documents.

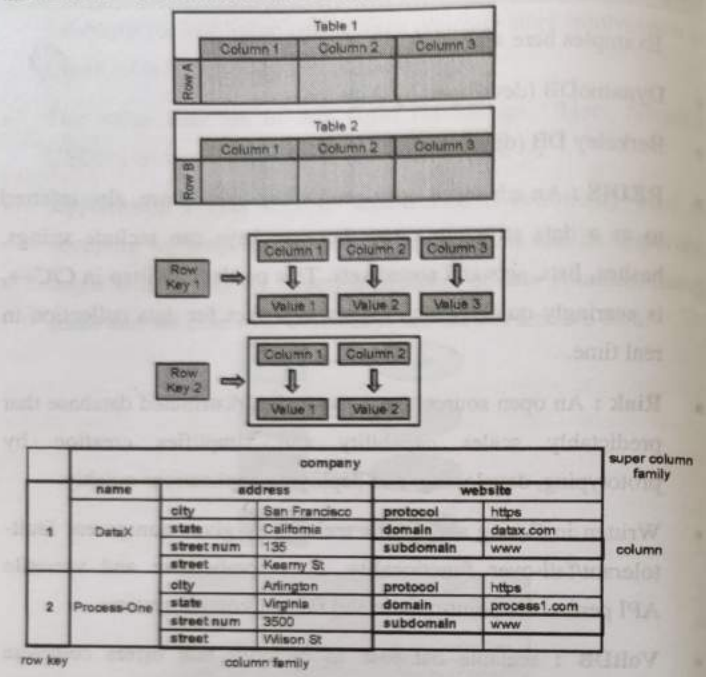
Constraints associated with the key-value store databases is its complexity in handling queries which will attempt to include many key-value pairs that may delay output and may cause data to clash with many-to-many relationships.

GQ. State example of any two key value databases **(2 Marks)**

Examples here are :

- DynamoDB (developed by Amazon)
- Berkeley DB (developed by Oracle)
- **REDIS :** An advanced open-source key-value store, also referred to as a data structure server because keys can include strings, hashes, lists, sets and sorted sets. This product, written in C/C++, is searingly quick, which makes it perfect for data collection in real time.
- **Riak :** An open source that is powerful, distributed database that predictably scales capability and simplifies creation by prototyping, developing, and deploying applications quickly.
- Written in Erlang and C this technology gives transparent fault-tolerant/fail-over functionality, a comprehensive and versatile API perfect for point-of-sale and factory control systems.
- **VoltDB :** scalable database in memory that offers complete transactional ACID consistency and ultra-high throughput, self-referred to as the NewSQL.
- This technology relies on segmentation and replication to achieve high-availability data snapshots and durable command logging using Java stored processes (for crash recovery), making it ideal for capital markets, digital networks, network services, and for online gaming.

2.5.2 Column Store Database



	company				super column family
	name	address		website	
1	DataX	city	San Francisco	protocol	https
		state	California	domain	datax.com
		street num	135	subdomain	www
		street	Kearny St		
2	Process-One	city	Arlington	protocol	https
		state	Virginia	domain	process1.com
		street num	3500	subdomain	www
		street	Wilson St		

column

row key

column family

Fig. 2.5.2 : An Example of Column Store

- This pattern employs data storage in individual cells that is further divided into columns, rather than storing data in relational tuples.
- Databases that are column-oriented operate only on columns. They together store vast quantities of data in columns. The column format and titles will diverge from one row to another.
- Each column is handled differently, but still, like conventional databases, each individual column will contain several other columns. (Niharika, 2020)

- Basically, columns are in this sort of storage mode. Data is readily available and it is possible to perform queries such as Number, AVERAGE, COUNT on columns easily.
- The setbacks for this system includes: transactions should be avoided or not supported, queries can decrease high performance with table joins, record updates and deletes reduce storage efficiency, and it can be difficult to design efficient partitioning/indexing schemes.

GQ. State example of any two column store databases. **(2 Marks)**

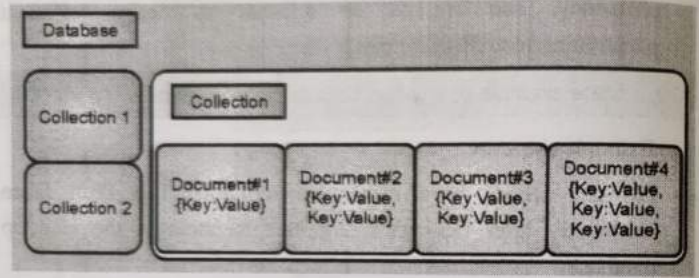
Examples here are :

- **HBase :** HBase is a distributed, portable, Big Data Store modelled after Google's BigTable technology, the Hadoop database.
- Google's BigTable
- **Cassandra :** An open-source distributed database management system built to manage very large volumes of data scattered over several servers without a single point of failure while delivering a highly accessible service.
- Written in Java, this product is best for non-transactional real-time data analysis with linear scalability and proven fault-tolerance combined with column indexes.

2.5.3 Document Database

- In the form of key-value pairs, the record database fetches and accumulates information, but here the values are called documents. A complicated data structure can be represented as a text.
- It is hierarchical version of key-value databases.

- The document can be in text form, arrays, strings, JSON (JavaScript Object Notation), XML (Extensible Markup Language) or any other format.
- The use of nested documents is immensely popular. It is highly efficient since most of the generated information is generally in the form of JSONs and is unstructured.



Key	Document
document-1	{ "id":"1", "name":"John Smith", "isactive":"true", "birthdate":"08/30/1984" }
document-2	{ "id":"2", "fullname":"Sara Walker", "isactive":"false", "birthdate":"02/15/1971" }
document-3	{ "id":"3", "fullname": { "firstname":"Max", "lastname":"Johnson" "middleinitial":"B" } "isactive":"true", "birthdate":"04/02/1984" }

Fig. 2.5.3 : An Example of Document

This format is extremely useful and appropriate for semi-structured data, and it is simple to retrieve and handle documents from storage. The drawbacks associated with this system includes the challenging factor of handling multiple documents and the inaccurate working of aggregation operations.

GQ. State example of any two document store databases. **(2 Marks)**

Examples of such databases are :

- **MongoDB :** This scalable, high-performance, open-source NoSQL database features document-oriented (JSON-like) storage, full index support, replication, and fast on-site updates from "humongous". This product is suitable for dynamic queries, dynamic data structures, written in C/C++, and if you favour indexes over Map/Reduce.
- **CouchDB :** Also an open-source database that focuses on the ease of data storage in aseries of JSON documents, each with its own definitions of the schema. Eventual consistency is enforced by ACID semantics that prevents locking database files during writing. This product, written in Java, is suitable for web-based applications that manage large quantities of data that are loosely organized.

2.5.4 Graph Database

GQ. Write a short note on Graph Databases. **(4 Marks)**

- This pattern of architecture clearly deals with information storage and management in graphs.
- Graphs are essentially structures that represent relations between two or more objects in some data.

- Objects or entities are referred to as nodes and are connected with relationships known as edges. There is a unique identifier on each edge.

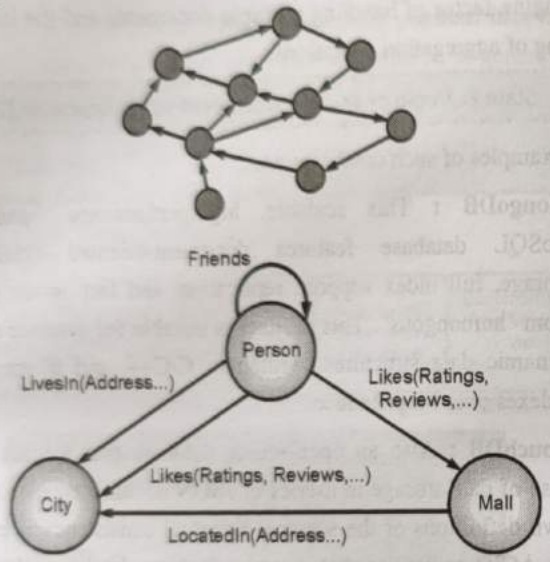


Fig. 2.5.4 : An Example of Graph

- For the graph, each node serves as a point of touch. In social networks where there are many and large numbers of entities, this pattern is very widely used and each entity has one or many characteristics that are linked by edges.
- There are loosely connected tables in the relational database pattern, whereas graphs are often strong and rigid in nature, have a faster traversal due to connections, and allow spatial data to be easily handled, but incorrect connections can lead to infinite loops. (Ian, 2016).

GQ. State example of any two graph databases. **(2 Marks)**

Examples of such databases are :

- **Neo4J :** The leading native graph database and graph platform is Neo4J : Neo4j. For enterprise levels of security and high performance and reliability by clustering, it is available both as open source and through a commercial license. Cypher, the graph query language of Neo4j, is very simple to learn and can use newly released open source toolkits, "Cypher on Apache Spark (CApS) and Cypher for Gremlin to operate across Neo4j, Apache Spark and Gremlin-based products."
- **FlockDB (Used by Twitter) :** FlockDB is easier than other graph databases since it attempts to solve less problems. It fits horizontally and is optimized for on-line, low-latency, high throughput environments such as websites.
- **ArangoDB :** this type of graph database requires one database, One Query language, Three models for data. The Limitless Possibilities. ArangoDB is a fast-growing native multi-model NoSQL database, with more than one million downloads.
- **OrientDB :** OrientDB is the first Distributed DBMS multi-model with a True Graph Driver. Multi-Model means NoSQL 2nd generation that is capable of managing complex domains with amazing efficiency.
- **Titan :** Titan is a scalable graph database designed for storing and querying graphs spread over a multi-machine cluster comprising hundreds of billions of vertices and edges. Titan is a transactional database that can facilitate the real-time execution of complex graph traversals by thousands of concurrent users.
- **DataStax :** In a rapidly changing environment where aspirations are strong, DataStax helps businesses thrive and new technologies occur daily.

- **Amazon Neptune :** Amazon Neptune with a highly connected datasets to create and run applications is secure, fast and has a fully managed graph database service that is easy. (Niharika, 2020).

NoSQL databases features :

Types Features	Key-value store	Document store	Column-oriented store	Graph store
Characteristics	A simple hash tableindexed by key	Multiple key/value pairsform a document. Document stored generally in JSON format.	Store data in columnar format. Each key is associated with multiple attributes.	Focused on modelling the structure of the data.
Pros	Very fast and scalable. Simple model.	Schema free: Unstructured data can be stored easily. Simple, powerful data model. Horizontal scalability.	Better for complex read queries. Fast querying of data. Storage of very large quantities of data. Better analytic performance. Improved data compression.	Powerful data model. Locally connected data. Indexed data. Easy to query. Handling complex relational information.
Cons	Stored data have no schema. Poor for complex data. All joins must be done in code. No foreign key constraints. No triggers.	Query model limited to keys and indexes. No standard query syntax. Map Reduce for larger queries. Poor for interconnected data.	Very low-level API. Undefined data usage pattern. Increased disk seek time. Increased cost of inserts. Poor for interconnected data.	Travers entire graph to give correct results. Sharding.
Suitable for	Storing session's information, user profiles,	Content management systems, blogging	Content management systems, blogging platforms,	Space problem and connected data, such as

Types Features	Key-value store	Document store	Column-oriented store	Graph store
	preferences, shopping cart data.	platforms, web analytics, real-time analytic's, e-commerce applications.	maintaining counters, expiring usage, heavy write volume such as log aggregation	social networks, spatial data, routing information for goods and money, recommendation engines.
Examples	Riak Redis MemcacheDB Dynamo Voldemort	MongoDB CouchDB ArangoDB MarkLogic RzthinkDB	BigTable Habse Casandra Accumulo Hypertable	Neo4j OrientDB Allegro Virtuoso InfiniteGraph

2.6 MONGODB

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables. MongoDB is a database which came into light around the mid-2000s.

MongoDB Features

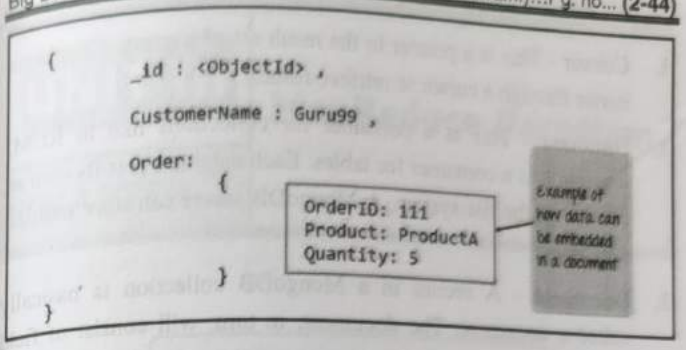
1. Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.

2. The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
3. The rows (or documents as called in MongoDB) doesn't need to have a schema defined beforehand. Instead, the fields can be created on the fly.
4. The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.
5. Scalability : The MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with around millions of documents within the database

MongoDB Example

The below example shows how a document can be modeled in MongoDB :

1. The _id field is added by MongoDB to uniquely identify the document in the collection.
2. What you can note is that the Order Data (OrderID, Product, and Quantity) which in RDBMS will normally be stored in a separate table, while in MongoDB it is actually stored as an embedded document in the collection itself. This is one of the key differences in how data is modeled in MongoDB.



Key Components of MongoDB Architecture

Below are a few of the common terms used in MongoDB :

1. **_id** – This is a field required in every MongoDB document. The _id field represents a unique value in the MongoDB document. The _id field is like the document's primary key. If you create a new document without an _id field, MongoDB will automatically create the field. So for example, if we see the example of the above customer table, Mongo DB will add a 24 digit unique identifier to each document in the collection.

_Id	CustomerID	CustomerName	OrderID
563479cc8a8a4246bd27d784	11	Guru99	111
563479cc7a8a4246bd47d784	22	Trevor smith	222
563479cc9a8a4246bds57d784	33	Nicole	333

2. **Collection** - This is a grouping of MongoDB documents. A collection is the equivalent of a table which is created in any other RDMS such as Oracle or MS SQL. A collection exists within a single database. As seen from the introduction collections don't enforce any sort of structure.

3. **Cursor** - This is a pointer to the result set of a query. Clients can iterate through a cursor to retrieve results.
4. **Database** - This is a container for collections like in RDMS wherein it is a container for tables. Each database gets its own set of files on the file system. A MongoDB server can store multiple databases.
5. **Document** - A record in a MongoDB collection is basically called a document. The document, in turn, will consist of field name and values.
6. **Field** - A name-value pair in a document. A document haszero or more fields. Fields are analogous to columns in relational databases.

Chapter Ends...

❑❑❑

UNIT III

CHAPTER 3

MapReduce Paradigm

University Prescribed Syllabus

MapReduce : The Map Tasks, Grouping by Key, The Reduce Tasks, Combiners, Details of MapReduce Execution, Coping With Node Failures. Algorithms Using MapReduce: Matrix-Vector Multiplication by MapReduce , Relational-Algebra Operations, Computing Selections by MapReduce, Computing Projections by MapReduce, Union, Intersection, and Difference by MapReduce, Computing Natural Join by MapReduce, Grouping and Aggregation by MapReduce, Matrix Multiplication, Matrix Multiplication with One MapReduce Step. Illustrating use of MapReduce with use of real life databases and applications.

Self-learning Topics : Implementation of MapReduce algorithms like Word count, Matrix-Vector and Matrix-Matrix algorithm.

3.1 INTRODUCTION TO MAPREDUCE

UQ. What is MapReduce ? Explain the role of Combiner with help of an example. **(MU - Dec. 18, 10 Marks)**

UQ. Explain Concept of MapReduce using an example. **(MU - Dec. 17, 5 Marks)**

MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. As the processing component, MapReduce is the heart of **Apache Hadoop.** The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

3.1.1 How MapReduce Works

- At a high level, MapReduce breaks input data into fragments and distributes them across different machines.
- The input fragments consist of key-value pairs. Parallel map tasks process the chunked data on machines in a cluster. The mapping output then serves as input for the reduce stage. The reduce task combines the result into a particular key-value pair output and writes the data to HDFS.
- The Hadoop Distributed File System usually runs on the same set of machines as the MapReduce software. When the framework executes a job on the nodes that also store the data, the time to complete the tasks is reduced significantly.

3.1.2 Basic Terminology of Hadoop MapReduce

As we mentioned above, MapReduce is a processing layer in a Hadoop environment. MapReduce works on tasks related to a job. The idea is to tackle one large request by slicing it into smaller units.

JobTracker and TaskTracker

- In the early days of Hadoop (version 1), JobTracker and TaskTracker daemons ran operations in MapReduce. At the time, a Hadoop cluster could only support MapReduce applications.
- A JobTracker controlled the distribution of application requests to the compute resources in a cluster. Since it monitored the execution and the status of MapReduce, it resided on a master node.
- A TaskTracker processed the requests that came from the JobTracker.

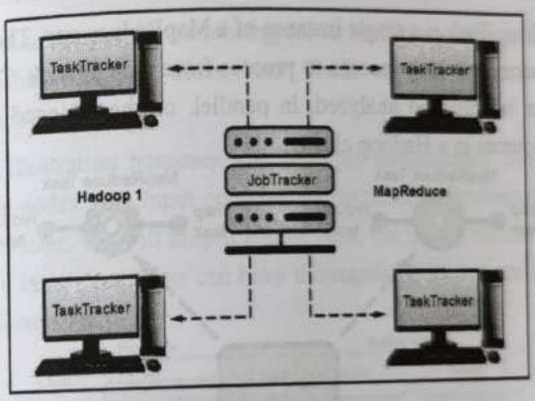


Fig. 3.1.1

- All task trackers were distributed across the slave nodes in a Hadoop cluster.

YARN

Later in Hadoop version 2 and above, YARN became the main resource and scheduling manager. Hence the name Yet Another Resource Manager. Yarn also worked with other frameworks for the distributed processing in a Hadoop cluster.

MapReduce Job

- A MapReduce job is the top unit of work in the MapReduce process. It is an assignment that Map and Reduce processes need to complete. A job is divided into smaller tasks over a cluster of machines for faster execution.
- The tasks should be big enough to justify the task handling time. If you divide a job into unusually small segments, the total time to prepare the splits and create tasks may outweigh the time needed to produce the actual job output.

☞ **MapReduce Task**

MapReduce jobs have two types of tasks.

(1) A Map Task is a single instance of a MapReduce app. These tasks determine which records to process from a data block. The input data is split and analyzed, in parallel, on the assigned compute resources in a Hadoop cluster.

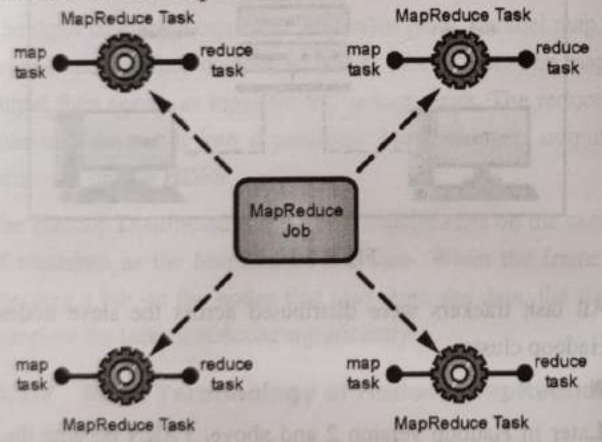


Fig. 3.1.2

This step of a MapReduce job prepares the <key, value> pair output for the reduce step.

(2) A Reduce Task processes an output of a map task. Similar to the map stage, all reduce tasks occur at the same time, and they work independently. The data is aggregated and combined to deliver the desired output. The final result is a reduced set of <key, value> pairs which MapReduce, by default, stores in HDFS.

3.1.3 How Hadoop Map and Reduce Work Together

- As the name suggests, MapReduce works by processing input data in two stages – Map and Reduce. To demonstrate this, we will use a simple example with counting the number of occurrences of words in each document.
- The final output we are looking for is: How many times the words Apache, Hadoop, Class, and Track appear in total in all documents.
- For illustration purposes, the example environment consists of three nodes. The input contains six documents distributed across the cluster. We will keep it simple here, but in real circumstances, there is no limit. You can have thousands of servers and billions of documents.

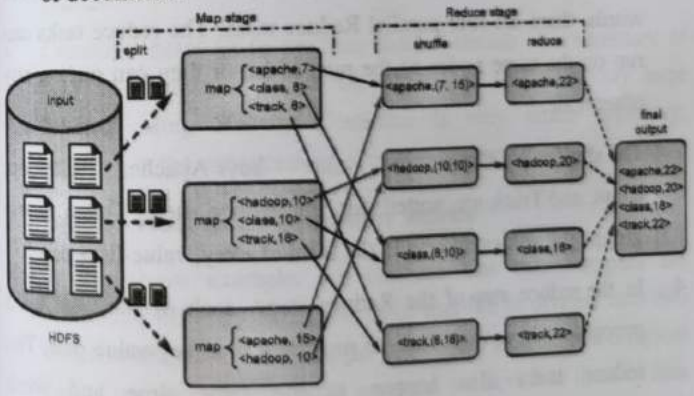


Fig. 3.1.3

1. First, in the map stage, the input data (the six documents) is split and distributed across the cluster (the three servers). In this case, each map task works on a split containing two documents. During mapping, there is no communication between the nodes. They perform independently.

2. Then, map tasks create a <key, value> pair for every word. These pairs show how many times a word occurs. A word is a key, and a value is its count. For example, one document contains three of four words we are looking for: Apache 7 times, Class 8 times, and Track 6 times. The key-value pairs in one map task output look like this:

   - <apache, 7>
   - <class, 8>
   - <track, 6>

   This process is done in parallel tasks on all nodes for all documents and gives a unique output.

3. After input splitting and mapping completes, the outputs of every map task are shuffled. This is the first step of the Reduce stage. Since we are looking for the frequency of occurrence for four words, there are four parallel Reduce tasks. The reduce tasks can run on the same nodes as the map tasks, or they can run on any other node.

   The shuffle step ensures the keys Apache, Hadoop, Class, and Track are sorted for the reduce step. This process groups the values by keys in the form of <key, value-list> pairs.

4. In the reduce step of the Reduce stage, each of the four tasks process a <key, value-list> to provide a final key-value pair. The reduce tasks also happen at the same time and work independently.

In our example from the diagram, the reduce tasks get the following individual results :

- <apache, 22>
- <hadoop, 20>
- <class, 18>
- <track, 22>

3.2 MAPREDUCE – COMBINERS

- Combiner always works in between Mapper and Reducer. The output produced by the Mapper is the intermediate output in terms of key-value pairs which is massive in size.
- If we directly feed this huge output to the Reducer, then that will result in increasing the Network Congestion. So to minimize this Network congestion we have to put combiner in between Mapper and Reducer.
- These combiners are also known as semi-reducer. It is not necessary to add a combiner to your Map-Reduce program, it is optional.
- Combiner is also a class in our java program like Map and Reduce class that is used in between this Map and Reduce classes.
- Combiner helps us to produce abstract details or a summary of very large datasets. When we process or deal with very large datasets using Hadoop Combiner is very much necessary, resulting in the enhancement of overall performance.

3.2.1 How does combiner works

- In the above example, we can see that two Mappers are containing different data. The main text file is divided into two different Mappers. Each mapper is assigned to process a different line of our data. In our above example, we have two lines of data so we have two Mappers to handle each line.

- Mappers are producing the intermediate key-value pairs, where the name of the particular word is key and its count is its value. For example, for the data Geeks For Geeks For the key-value pairs are shown below.

  // Key Value pairs generated for data

  (Geeks,1)

  (For,1)

  (Geeks,1)

  (For,1)

- The key-value pairs generated by the Mapper are known as the intermediate key-value pairs or intermediate output of the Mapper.
- Now we can minimize the number of these key-value pairs by introducing a combiner for each Mapper in our program. In our case, we have 4 key-value pairs generated by each of the Mapper. Since these intermediate key-value pairs are not ready to directly feed to Reducer because that can increase Network congestion so Combiner will combine these intermediate key-value pairs before sending them to Reducer.
- The combiner combines these intermediate key-value pairs as per their key. For the above example for data for the combiner will partially reduce them by merging the same pairs according to their key value and generate new key-value pairs as shown below.

  // Partially reduced key-value pairs with combiner

  (Geeks,2)

  (For,2)

- With the help of Combiner, the Mapper output got partially reduced in terms of size(key-value pairs) which now can be made available to the Reducer for better performance.

- Now the Reducer will again Reduce the output obtained from combiners and produces the final output that is stored on **HDFS (Hadoop Distributed File System)**.

3.2.2 Advantage of Combiners

1. Reduces the time taken for transferring the data from Mapper to Reducer.
2. Reduces the size of the intermediate output generated by the Mapper.
3. Improves performance by minimizing Network congestion.

3.2.3 Disadvantage of combiners

- The intermediate key-value pairs generated by Mappers are stored on Local Disk and combiners will run later on to partially reduce the output which results in expensive Disk Input-Output.
- The map-Reduce job cannot depend on the function of the combiner because there is no such guarantee in its execution.

3.3 MATRIX VECTOR MULTIPLICATIONBY MAPREDUCE

UQ. Write pseudo code for Matrix vector Multiplication by MapReduce. Illustrate with an example showing all thesteps. **(MU - May 19, 10 Marks)**

GQ. What happen when vector does not fit in memory in matrix vector multiplication ?

UQ. Write MapReduce pseudocode to multiply to matrices. Illustrate the procedure on the following matrices.Clearly show all the steps. **(MU - May 18, 10 Marks)**

- MapReduce is a technique in which a huge program is subdivided into small tasks and run parallelly to make computation faster, save time, and mostly used in distributed systems. It has 2 important parts :

- **Mapper :** It takes raw data input and organizes into key, value pairs. For example, In a dictionary, you search for the word "Data" and its associated meaning is "facts and statistics collected together for reference or analysis". Here the Key is Data and the Value associated with is facts and statistics collected together for reference or analysis.
- **Reducer :** It is responsible for processing data in parallel and produce final output.

☞ **Algorithm 1 : The map function**

1. **For** each element m_{ij} of M **do**
2. product (key value) pair as ((i, k), (M, j, m_{ij})), for k = 1, 2,3,...up to the number of columns of N
3. for each element n_{jk} of N do
4. product (key value) pair as ((i, k), (N, j, n_{jk})), for i = 1, 2,3,...up to the number of rows of M.
5. **return** set of (key value) pairs that each key, (i, k), has a list with values (M, j, m_{ij}) and (N, j, n_{jk}) for all possible values of j

☞ **Algorithm 2 : The reduce function**

1. For each key (i, k) **do**
2. sort value begin with M by j in $list_M$
3. sort value begin with N by j in $list_N$
4. multiply m_{ij} and n_{jk} for j_{th} value of each list
5. sum up $m_{ij} * m_{jk}$
6. **return** (i, k), $\sum_{j=1} m_{ij} * n_{jk}$

Let us consider the matrix multiplication example to visualize MapReduce

Consider the following matrix

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Here matrix A is a 2 × 2 matrix which means the number of rows(i) = 2 and the number of columns(j) = 2. Matrix B is also a 2 × 2 matrix where number of rows(j) = 2 and number of columns(k)=2. Each cell of the matrix is labelled as A_{ij} and B_{jk}. Ex. element 3 in matrix A is called A_{21} i.e. 2nd- row 1stcolumn. Now one step matrix multiplication has 1 mapper and 1 reducer. The Formula is:

Mapper for Matrix A (k, v) = ((i, k), (A, j, A_{ij})) for all k

Mapper for Matrix B (k, v) = ((i, k), (B, j, B_{jk})) for all i

Therefore, ecomputing the mapper for Matrix A:

k, i, j computes the number of times it occurs .

Here all are 2, therefore when k = l, i can have

2 values 1 & 2, each case can have 2 further

values of j = 1 and j = 2. Substituting all values

in formula

k = 1 i = 1 j = 1 ((1, 1), (A, 1, 1))

j = 2 ((1, 1), (A, 2, 2))

i = 2 j = 1 ((2, 1), (A, 1, 3))

j = 2 ((2, 1), (A, 2, 4))

k = 2 i = 1 j = 1 ((1, 2), (A, 1, 1))

j = 2 ((1, 2), (A, 2, 2))

i = 2 j = 1 ((2, 2), (A, 1, 3))

j = 2 ((2, 2), (A, 2, 4))

Matrix-Vector Multiplication by Map-Reduce

Computing the mapper for matrix B

i = 1 j = 1 k = 1 ((1, 1), (B, 1, 5))

k = 2 ((1, 2), (B, 1, 6))

j = 2 k = 1 ((1, 1), (B, 2, 7))

j = 2 ((1, 2), (B, 2, 8))

i = 2 j = 1 k = 1 ((2, 1), (B, 1, 5))

k = 2 ((2, 2), (B, 1, 6))

j = 2 k = 1 ((2, 1), (B, 2, 7))

k = 2 ((2, 2), (B, 2, 8))

- Matrix-Vector Multiplication by Map-Reduce

Reducer (k, v) = (i, k)=>Make sorted A_{list} and B_{list}

(i,k) => Summation (A_{ij} * B_{jk})) for J

Output =>((i k),sum)

Therefore, computing the reducer:

We can observe from Mapper computation

that 4 pairs are common (1, 1), (1, 2),

(2, 1) and (2, 2)

Make a list separate for Matrix A &

B with adjoining values taken from

Mapper step above :

Matrix-Vector Multiplication by Map-Reduce

(1, 1) =>A_{list}={(A, 1, 1), (A, 2, 2)}

B_{list}={(B, 1, 5), (B, 2, 7)}

Now A_{ij}×B_{jk}: [(1 * 5) + (2*7)] = 19 ...(i)

(1, 2) =>A_{list}={(A, 1, 1), (A, 2, 2)}

B_{list}={(B, 1, 6), (B, 2, 8)}

Now A_{ij}×B_{jk}: [(1*6) + (2*8)] :22 ...(ii)

(2, 1) =>A_{list}={(A, 1, 3), (A, 2, 4) }

B_{list}={(B, 1, 5), {B, 2, 7)}

Now A_{ij}×B_{jk}: [(3* 5) + (4*7)] = 43 ...(iii)

(2, 2) =>A_{list}={(A, 1, 3), (A, 2, 4)}

B_{list}={(B, 1, 6), (B, 2, 8)}

Now A_{ij}×B_{jk}: [(3*6) + (4*8)] = 50 ...(iv)

From (i), (ii), (iii) and (iv) we conclude that

((1, 1), 19)

((1, 2), 22)

((2, 1), 43)

((2, 2), 50)

Therefore, the final matrix is,

$$\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

3.4 RELATIONAL ALGEBRA OPERATIONS

1. Selection
2. Projection
3. Union & Intersection
4. Natural Join
5. Grouping & Aggregation.

1. Selection

- Apply a condition c to each taple in the relation and produce as output only those tuples that satisfy c.
- The result of this selection is denoted by 6c(R)6c(R)
- Selection really does not need the full power of MapReduce.

- They can be done most conveniently in the map portion alone, although they could also be done in the reduce portion also.
- The pseudo code is as follows :

```
Map (key, valve)
for tuple in valve :
if tuple satisfies C :
emit (tuple, tuple)
Reduce (key, valves)
emit (key, key)
```

2. Projection

- For some subset s of the attribute of the relation, produce from each tuple only the components for the attributes in S.
- The result of this projection is denoted TTs (R)
- Projection is performed similarly to selection.
- As projection may cause the same tuple to appear several times, the reduce function eliminate duplicates.
- The pseudo code for projection is as follows :

```
Map (key, valve)
for tuple in valve :
ts = tuple with only the components forthe attributes in S.
emit (ts, ts)
Reduce (key, values)
emit (key, key)
```

3. Union Using Map Reduce

- Both selection and projection are operations that are applied on a single table, whereas Union, intersection and difference are among the operations that are applied on two or more tables.

- Let's consider that schemas of the two tables are the same, and columns are also ordered in same order.
- **Map Function :** For each row r generate key-value pair (r, r).
  - **Reduce Function :** With each key there can be one or two values (As we don't have duplicate rows), in either case just output first value.

This operations has the **map function of the selection and reduce function of projection**. Let's see the working using an example. Here yellow colour represents one table and green colour is used to represent the other one stored at two map workers.

Map worker 1

Table 1	
A	**B**
1	2
2	3
5	6

Table 2	
A	**B**
2	3
4	4
6	1

Map worker 2

Table 1	
A	**B**
6	1
6	3
7	6

Table 2	
A	**B**
9	8
3	3
0	1

Initial data at map workers

After applying the map function and grouping the keys we will get output as :

Map worker 1

Key	Value
(1,2)	[(1,2)]
(2,3)	[(2,3), (2, 3)]
(5,6)	[(5, 6)]
(4, 4)	[(4, 4)]
(6, 1)	[(6, 1)]

Map worker 2

Key	Value
(6, 1)	[(6, 1)]
(6, 3)	[(6, 3)]
(7, 6)	[(7, 6)]
(9, 8)	[(9, 8)]
(3, 3)	[(3, 3)]
(0, 1)	[(0, 1)]

Map and grouping the keys

The data to be sent to reduce workers will look like:

Map worker 1

RW 1	
Key	value
(1,2)	[(1,2)]
(2,3)	[(2,3), (2, 3)]
(5,6)	[(5, 6)]

RW 2	
Key	value
(4, 4)	[(4, 4)]
(6, 1)	[(6, 1)]

Map worker 2

RW1	
Key	value
(6, 3)	[(6, 3)]
(3, 3)	[(3, 3)]
(0, 1)	[(0, 1)]

RW2	
Key	Value
(6, 1)	[(6, 1)]
(7, 8)	[(7, 8)]
(9, 8)	[(9, 8)]

Files to be sent to reduce workersData at reduce workers after will be :

Reduce worker 1

RW 1	
Key	value
(1,2)	[(1,2)]
(2,3)	[(2,3), (2, 3)]
(5,6)	[(5, 6)]

RW 1	
Key	value
(6, 3)	[(6, 3)]
(3, 3)	[(3, 3)]
(0, 1)	[(0, 1)]

Reduce worker 2

Key	Value
(4, 4)	[(4, 4)]
(6, 1)	[(6, 1)]

Key	Value
(6, 1)	[(6, 1)]
(7, 8)	[(7, 8)]
(9, 8)	[(9, 8)]

Files At reduce workers

At reduce workers aggregation on keys will be done.

Reduce worker 1

Key	Value
(1,2)	[(1,2)]
(2,3)	[(2,3), (2, 3)]
(5,6)	[(5, 6)]
(6, 3)	[(6, 3)]
(3, 3)	[(3, 3)]
(0, 1)	[(0, 1)]

Reduce worker 2

Key	Value
(4, 4)	[(4, 4)]
(6, 1)	[(6, 1), (6, 1)]
(7, 6)	[(7, 6)]
(9, 8)	[(9, 8)]

Aggregated data at reduce workers

The final output after applying the reduce function which takes only the first value and ignores everything else is as follows:

Reduce worker 1

A	B
1	2
2	3
5	6
6	3
3	3
0	1

Reduce worker 2

A	B
4	4
6	1
7	6
9	8

Final table after union

Here we note that in this case same as projection we can this done without moving data around in case we are not interested in removing duplicates. And hence this operation is also efficient it terms of data shuffle across machines.

☞ Intersection Using Map Reduce

For intersection, let's consider the same data we considered for union and just change the map and reduce functions

- **Map Function** : For each row r generate key-value pair (r, r) (Same as union).
- **Reduce Function** : With each key there can be one or two values (As we don't have duplicate rows), in case we have length of list as 2 we output first value else we output nothing.

As the map function is same as union and we are considering the same data lets skip to the part before reduce function is applied.

Reduce worker 1

Key	Value
(1,2)	[(1,2)]
(2,3)	[(2,3), (2, 3)]
(5,6)	[(5, 6)]
(6, 3)	[(6, 3)]
(3, 3)	[(3, 3)]
(0, 1)	[(0, 1)]

Reduce worker 2

Key	Value
(4, 4)	[(4, 4)]
(6, 1)	[(6, 1), (6, 1)]
(7, 6)	[(7, 6)]
(9, 8)	[(9, 8)]

Data at reduce workers

Now we just apply the reduce operation which will output only rows if list has a length of 2.

Reduce worker 1

File 1	
A	B
2	3

Reduce worker 2

File 1	
A	B
6	1

Output of intersection

☞ Difference Using Map Reduce

Let's again consider the same data. The difficulty with difference arises with the fact that we want to output a row only if it exists in the first table but not the second one. So the reduce function needs to keep track on which tuple belongs to which relation. To visualize that easier, we will keep those rows green which come from 2nd table and yellow for which come from 1st table and **purple** which comes from both tables.

- **Map Function :** For each row r create a key-value pair (r, T1) if row is from table 1 else product key-value pair (r, T2).
- **Reduce Function :** Output the row if and only if the value in the list is T1 , otherwise output nothing.

The data taken initially is the same as it was for union

Map worker 1

Table 1	
A	B
1	2
2	3
5	6

Table 2	
A	B
2	3
4	4
6	1

Map worker 2

Table 1	
A	B
6	1
6	3
7	6

Table 2	
A	B
9	8
3	3
0	1

Initial Data

After applying the map function and grouping the keys the data looks like the following figure

Reduce worker 1

Key	Value
(1, 2)	[T1]
(2, 3)	[T1, T2]
(5, 6)	[T1]
(4, 4)	[T2]
(6, 1)	[T2]

Reduce worker 2

Key	Value
(6, 3)	[T1]
(7, 6)	[T1]
(9, 8)	[T2]
(6, 1)	[T1]
(3, 3)	[T2]
(0, 1)	[T2]

Data after applying map function and grouping keys

After applying map function files for reduce workers will be created based on hashing keys as has been the case so far.

Reduce worker 1

RW1	
Key	Value
(1, 2)	[T1]
(2, 3)	[T1, T2]
(5, 6)	[T1]

RW2	
Key	Value
(4, 4)	[T2]
(6, 1)	[T2]

Reduce worker 2

RW1	
Key	Value
(6, 3)	[T1]
(7, 6)	[T1]
(9, 8)	[T2]

RW2	
Key	Value
(6, 1)	[T1]
(3, 3)	[T2]
(0, 1)	[T2]

Files for reduce workers

The data at the reduce workers will look like

Reduce worker 1

RW1	
Key	Value
(1, 2)	[T1]
(2, 3)	[T1, T2]
(5, 6)	[T1]

RW1	
Key	Value
(6, 3)	[T1]
(7, 6)	[T1]
(9, 8)	[T2]

Reduce worker 2

RW1	
Key	Value
(4, 4)	[T2]
(6, 1)	[T2]

RW1	
Key	Value
(6, 1)	[T1]
(3, 3)	[T2]
(0, 1)	[T2]

Files at reduce workers

After aggregation of the keys at reduce workers the data looks like:

Reduce worker 1

Key	Value
(1, 2)	[T1]
(2, 3)	[T1, T2]
(5, 6)	[T1]
(6, 3)	[T1]
(9, 8)	[T2]

Key	Value
(4, 4)	[T2]
(6, 1)	[T1, T2]
(3, 3)	[T2]
(0, 1)	[T2]
(7, 6)	[T1]

Data after aggregation of keys at reduce workers

The final output is generated after applying the reduce function over the output.

File 1	
A	B
1	2
5	6
6	3

File 1	
A	B
7	6

Output of difference of the tables

For the difference operation we notice that we cannot get rid of the reduce part and hence have to send data across the workers as the context of from which table the value came is needed. Hence it will be **more expensive** operation as compared to selection, projection, union and intersection.

Grouping and Aggregation Using Map Reduce

Usually understanding grouping and aggregation takes a bit of time when we learn SQL, but not in case when we understand these operations using map reduce. The logic is already there in the working of the map. Map workers implicitly group keys and the reduce function acts upon the aggregated values to generate output.

- **Map Function :** For each row in the table, take the attributes using which grouping is to be done as the key, and value will be the ones on which aggregation is to be performed. For example, If a relation has 4 columns A, B, C, D and we want to group by A, B and do an aggregation on C we will make (A, B) as the key and C as the value.
- **Reduce Function :** Apply the aggregation operation (sum, max, min, avg, ...) on the list of values and output the result.

For our example lets group by (A, B) and apply sum as the aggregation.

Map worker1							
File 1				File 2			
A	B	C	D	A	B	C	D
1	2	3	1	4	2	1	3
2	2	3	2	6	8	4	4
1	2	1	3	3	2	2	4

Map worker 2							
File 1				File 2			
A	B	C	D	A	B	C	D
1	2	5	2	3	2	5	2
2	3	2	4	2	3	9	2
1	3	1	3	3	4	2	1

Initial data at the map workers

The data after application of map function and grouping keys will creates (A, B) as key and C as value and D is discarded as if it doesn't exist.

Reduce worker 1	
Key	Value
(1, 2)	[3, 1]
(2, 2)	[3]
(4, 2)	[1]
(6, 8)	[4]
(3, 2)	[2]

Reduce worker 2	
Key	Value
(1, 2)	[5]
(2, 3)	[2, 9]
(1, 3)	[1]
(3, 2)	[1]
(3, 4)	[2]

Data at map workers

Applying partitioning using hash functions, we get

Files for the reduce workers

Map worker 1			
RW1		RW2	
Key	Value	Key	Value
(1, 2)	[3, 1]	(6, 8)	[4]
(2, 2)	[3]	(3, 2)	[2]
(4, 2)	[1]		

Map worker 2			
RW1		RW2	
Key	Value	Key	Value
(1, 2)	[5]	(3, 2)	[1]
(2, 3)	[2, 9]	(3, 4)	[2]
		(1, 3)	[1]

Files for reduce workers

The data at the reduce workers will look like

Reduce worker 1			
RW 1		RW1	
Key	Value	Key	Value
(1, 2)	(3, 1)	(1, 2)	[5]
(2, 2)	[3]	(2, 3)	[2, 9]
(4, 2)	[1]		

Reduce worker 1			
RW 2		RW 2	
Key	Value	Key	Value
(6, 8)	[4]	(3, 2)	[1]
(3, 2)	[2]	(3, 4)	[2]
		(1, 3)	[1]

☞ **Data at reduce workers**

The data is aggregated based on keys before applying the aggregation function (sum in this case).

Reduce worker 1	
Key	Value
(1, 2)	[3, 1, 5]
(2, 2)	[3]
(4, 2)	[1]
(2, 3)	[(2, 9)]

Reduce worker 2	
Key	Value
(6, 8)	[4]
(3, 2)	[1, 2]
(3, 4)	[2]
(1, 3)	[1]

☞ **Aggregated data based on keys**

After applying the sum over the value lists we get the final output

Reduce worker 1		
A	B	Sum
1	2	9
2	2	3
4	2	1
2	3	11

Reduce worker 2		
A	B	Sum
6	8	4
3	2	3
3	4	2
1	3	1

☞ **Output of group by (A, B) sum(C)**

Here also like difference operation we can't get rid of the reduce stage. The context of tables isn't wanted here but the aggregation function makes it necessary for the values to be in one place for a single key. This operation is also inefficient as compared to selection, projection, union, and intersection. The column that is not in aggregation or grouping clause is ignored and isn't required. So, if the data be stored in a columnar format, we can save cost of loading a lot of data. Usually there are only a few columns involved in grouping and aggregation it does save up a lot of cost both in terms of data that is sent over the network and the data that needs to be loaded to main memory for execution.

3.5 COMPUTING NATURAL JOIN USING MAP REDUCE

The natural join will keep the rows that matches the values in the common column for both tables. To perform natural join, we will have to keep track of from which table the value came from. If the values for the same key are from different tables we need to form pairs of those values along with key to get a single row of the output. Join can explode the number of rows as we have to form each and every possible combination of the values for both tables.

- **Map Function :** For two relations Table 1(A, B) and Table 2(B, C) the map function will create key-value pairs of form b: [(T1, a)] for table 1 where T1 represents the fact that the value a came from table 1, for table 2 key-value pairs will be of the form b: [(T2, c)].
- **Reduce Function :** For a given key b construct all possible combinations for the values where one value is from table T1 and the other value is from table T2. The output will consist of key-value pairs of form b : [(a, c)] which represent one row a, b, c for the output table.

☞ **For an example let's consider joining Table 1 and Table 2, where B is the common column**

Map worker 1				Map worker 2			
Table 1		Table 2		Table 1		Table 2	
A	B	B	C	A	B	B	C
1	2	2	3	6	1	9	8
2	3	4	4	6	3	3	4
5	6	6	1	7	6	2	1

☞ **Initial data at map workers**

The data after applying the map function and grouping at the map workers will look like :

Map worker 1	
Key	Value
2	[T1, 1), (T2, 3)]
3	[T(1, 2)]
6	[(T1, 5), (T2, 1)]
4	[(T1, 4)]

Map worker 2	
Key	Value
1	[T1, 6)
3	[T1, 6), (T2, 4)]
6	[(T1, 7)
9	[(T2, 8)]
2	[(T2, 1)]

☞ **Data at map workers after applying map function and grouping the keys**

As has been the case so far files for reduce workers will be created at the map workers :

Map worker 1			
RW 1		RW 2	
Key	Value	Key	Value
2	[(T1, 1), (T2, 3), (T2, 1)]	6	[(T1, 5), (T2, 1)]
3	[(T1, 2)]	4	[(T1, 4)]

Map worker 2			
RW 1		RW 2	
Key	Value	Key	Value
1	[(T1, 6)	6	[(T1, 7)]
3	[(T1, 6), (T2, 4)]	9	[(T2, 8)]

☞ **Files constructed for reduce workers**

The data at the reduce workers will be :

Reduce worker 1			
RW1		RW1	
Key	Value	Key	Value
2	[(T1, 1), (T2, 3), (T2, 1)]	1	[(T1, 6)]
3	[(T1, 2)]	3	[(T1, 6), (T2, 4)]

Reduce worker 1			
RW2		RW2	
Key	Value	Key	Value
6	[(T1, 5), (T2, 1)]	6	[(T1, 7)]
4	[(T1, 4)]	9	[(T2, 8)]

☞ **Data at reduce workers**

Applying aggregation of keys at the reduce workers we get :

Reduce workers 1	
Key	value
2	[(T1, 1), (T2, 3), (T2, 1)]
3	[(T1, 2), (T1, 6), (T2, 4)]
1	[(T1, 6)]

	Reduce workers 2
Key	value
6	[(T1, 5), (T1, 7), (T2, 1)]
4	[(T1, 4)
9	[(T2, 8)]

☞ **Data after aggregation of keys at the reduce workers**

After applying the reduce function which will create a row by taking one value from table T1 and other one from T2. If there are only values from T1 or T2 in the values list that won't constitute a row in output.

Reduce worker 1		
B	A	C
2	1	3
2	1	1
3	2	4
3	6	4

Reduce worker 1		
B	A	C
6	5	1
6	7	1

☞ **Output of the join**

- As we need to keep context from which table a value came from, we can't get rid of the data that needs to be sent across the workers for application of reduce task, this operation also becomes costly as compared to others we discussed so far.
- The fact that for each list of values we need to create pairs also plays a major factor in the computation cost associated with this operation.

3.6 ILLUSTRATING USE OF MAPREDUCE WITH USE OF REAL LIFE DATABASES AND APPLICATIONS

MapReduce is a programming model used to perform distributed processing in parallel in a Hadoop cluster, which Makes Hadoop working so fast. When you are dealing with Big Data, serial processing is no more of any use. MapReduce has mainly two tasks which are divided phase-wise:

- Map Task
- Reduce Task

Let us understand it with a real-time example, and the example helps you understand Mapreduce Programming Model in a story manner :

- Suppose the Indian government has assigned you the task to count the population of India. You can demand all the resources you want, but you have to do this task in 4 months. Calculating the population of such a large country is not an easy task for a single person(you). So what will be your approach?.
- One of the ways to solve this problem is to divide the country by states and assign individual in-charge to each state to count the population of that state.

- Task Of Each Individual: Each Individual has to visit every home present in the state and need to keep a record of each house members as :

  State_Name Member_House1

  State_Name Member_House2

  State_Name Member_House3

  State_NameMember_House n

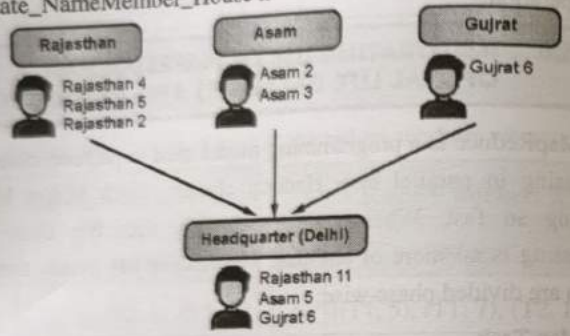


Fig. 3.6.1

For Simplicity, we have taken only three states.

- This is a simple Divide and Conquer approach and will be followed by each individualto count people in his/her state.
- Once they have counted each house member in their respective state. Now they need to sum up their results and need to send it to the Head-quarter at New Delhi.
- We have a trained officer at the Head-quarter to receive all the results from each state and aggregate them by each state to get the population of that entire state. And now, with this approach, you are easily able to count the population of India by summing up the results obtained at Head-quarter.

- The Indian Govt. is happy with your work and the next year they asked you to do the same job in 2 months instead of 4 months. Again you will be provided with all the resources you want.
- Since the Govt. has provided you with all the resources, you will simply double the number of assigned individual in-charge for each state from one to two. For that divide each state in 2 division and assigned different in-charge for these two divisions as:
  - State_Name_Incharge _division1
  - State_Name_Incharge_division2
- Similarly, each individual in charge of its division will gather the information about members from each house and keep its record.
- We can also do the same thing at the Head-quarters, so let's also divide the Head-quarter in two division as:
  - Head-qurter_Division1
  - Head-qurter_Division2

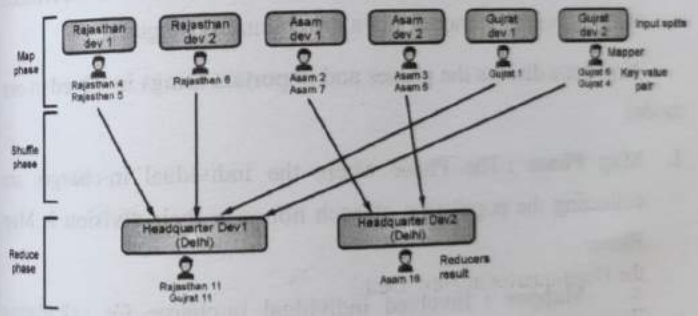


Fig. 3.6.2

- Now with this approach, you can find the population of India in two months.
- But there is a small problem with this, we never want the divisions of the same state to send their result at different Head-

quarters then, in that case, we have the partial population of that state in Head-quarter_Division1 and Head-quarter_Division2 which is inconsistent because we want consolidated population by the state, not the partial counting.

- One easy way to solve is that we can instruct all individuals of a state to either send there result to Head-quarter_Division1 or Head-quarter_Division2. Similarly, for all the states.
- Our problem has been solved, and you successfully did it in two months.
- Now, if they ask you to do this process in a month, you know how to approach the solution.
- Great, now we have a good scalable model that works so well. The model we have seen in this example is like the MapReduce Programming model. So now you must be aware that MapReduce is a programming model, not a programming language.

Now let's discuss the phases and important things involved in our model.

1. **Map Phase :** The Phase where the individual in-charges are collecting the population of each house in their division is Map Phase.
    - **Mapper :** Involved individual in-charge for calculating population.
    - **Input Splits :** The state or the division of the state.
    - **Key-Value Pair :** Output from each individual Mapper like the key is Rajasthan and value is 2.

2. **Reduce Phase :** The Phase where you are aggregating your result.
    - **Reducers :** Individuals who are aggregating the actual result. Here in our example, the trained-officers. Each Reducer produce the output as a key-value pair.
3. **Shuffle Phase :** The Phase where the data is copied from Mappers to Reducers is Shuffler's Phase. It comes in between Map and Reduces phase. Now the Map Phase, Reduce Phase, and Shuffler Phase out the three main Phases of our Mapreduce.

Chapter Ends...

❑❑❑