Unit-1: Introduction to Big Data

Introduction to Big Data

What is Big Data?

- Big Data is a collection of large datasets that cannot be adequately processed using traditional processing techniques.
- Big data is not only data it has become a complete subject, which involves various tools, techniques and frameworks.
- Big data term describes the volume amount of data both structured and unstructured manner that adapted in day-to-day business environment. It's important that what organizations utilize with these with the data that matters.
- Big data helps to analyze the in-depth concepts for the better decisions and strategic taken for the development of the organization.
- Big Data includes huge volume, high velocity, and extensible variety of data.
- The data in it will be of three types.
  1. **Structured data** − Relational data.
  2. **Semi Structured data** − XML data.
  3. **Unstructured data** − Word, PDF, Text, Media Logs.

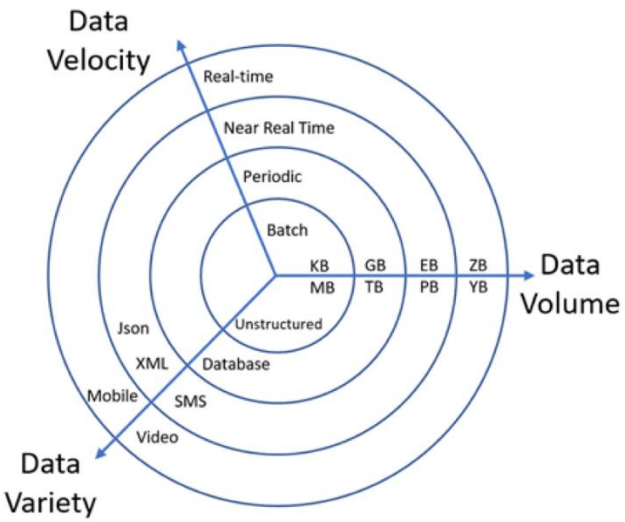


Figure 1: Data: Big in volume, variety, and velocity

Benefits of Big Data

- Using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions, and other advertising mediums.

- Using the information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production.
- Using the data regarding the previous medical history of patients, hospitals are providing better and quick service.

Categories of 'Big Data'

1. Structured Data

- 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 deriving value out of it.
- When size of such data grows to a huge extent, typical sizes are being in the rage of multiple zettabyte.

Examples of structured Data

Employee_ID	Employee_Name	Gender	Department	Salary_In_lacs
1	AAA	M	Sales	750000
2	BBB	F	Admin	800000
3	CCC	M	Account	500000

2. Unstructured Data

- 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 etc.
- Now a days 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.

Examples of Unstructured Data

- Typical human-generated unstructured data includes:
    - **Text files**: Word processing, spreadsheets, presentations, email, logs.
    - **Email**: Email has some internal structure thanks to its metadata, and we sometimes refer to it as semi-structured. However, its message field is unstructured and traditional analytics tools cannot parse it.
    - **Social Media**: Data from Facebook, Twitter, LinkedIn.
    - **Website**: YouTube, Instagram, photo sharing sites.
    - **Mobile data**: Text messages, locations.

- **Communications**: Chat, IM, phone recordings, collaboration software.
- **Media**: MP3, digital photos, audio and video files.
- **Business applications**: MS Office documents, productivity applications.

- Typical machine-generated unstructured data includes:
  - **Satellite imagery**: Weather data, landforms, military movements.
  - **Scientific data**: Oil and gas exploration, space exploration, seismic imagery, atmospheric data.
  - **Digital surveillance**: Surveillance photos and video.
  - **Sensor data**: Traffic, weather, oceanographic sensors.

3. Semi-Structured Data

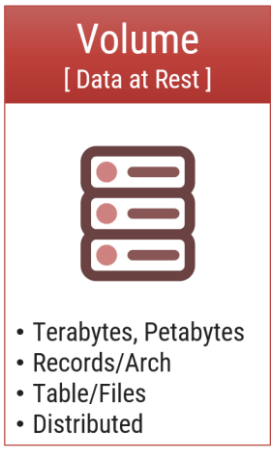
- Semi-structured data can contain both the forms of data.
- User can see semi-structured data as a structured in form, but it is actually not defined with e.g. a table definition in relational DBMS.

Examples of Semi-structured Data

- Personal data stored in a XML file:
  <rec><name>Rahul Mehta</name><sex>Male</sex><age>25</age></rec>
  <rec><name>Reena Shah</name><sex>Female</sex><age>34</age></rec>
  <rec><name>Karan Joshi</name><sex>Male</sex><age>30</age></rec>

Characteristic of Big Data

Volume



- Volume represents the volume i.e., amount of data that is growing at a high rate i.e., data volume in Petabytes.
- Bits → Bytes → Kilobytes → Megabytes → Gigabytes → Terabytes → Petabytes → Exabytes → Zettabytes → Yottabytes
- The name big data itself is associated with its huge size.
- Big data is a huge amount of data generated daily from a variety of sources, including business processes, machines, social media platforms, networks, and human interactions.
- **Typical internal data sources**: Data present within an organization's firewall. It is as follows:
  - Data storage: File systems, SQL (RDBMSs – Oracle, MS SQL Server, DB2, MySQL, PostgreSQL, etc.), NoSQL (MongoDB, Cassandra, etc.), and so on.
  - Archives: Archives of scanned documents, paper archives, customer correspondence records, patients' health records, students' admission records, students' assessment records, and so on.
- **External data sources**: Data residing outside an organization's firewall. It is as follows:

- Public Web: Wikipedia, weather, regulatory, compliance, census, etc.

- **Both (internal + external data sources):**
  - Sensor data: Car sensors, smart electric meters, office buildings, air conditioning units, refrigerators, and so on.
  - Machine log data: Event logs, application logs, Business process logs, audit logs, clickstream data, etc.
  - Social media: Twitter, blogs, Facebook, LinkedIn, YouTube, Instagram, etc.
  - Business apps: ERP, CRM, HR, Google Docs, and so on.
  - Media: Audio, Video, Image, Podcast, etc.
  - Docs: Comma separated value (CSV), Word Documents, PDF, XLS, PPT, and so on.

Value

- Value refers to turning data into value. By turning accessed big data into values, businesses may generate revenue.
- 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.

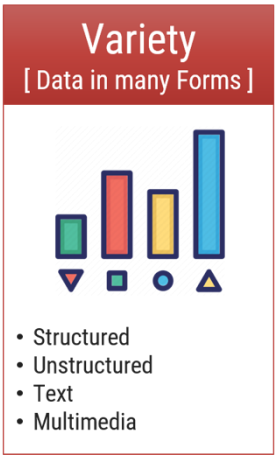
Veracity

- Veracity refers to the uncertainty of available data. Veracity arises due to the high volume of data that brings incompleteness and inconsistency.
- 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.

Visualization

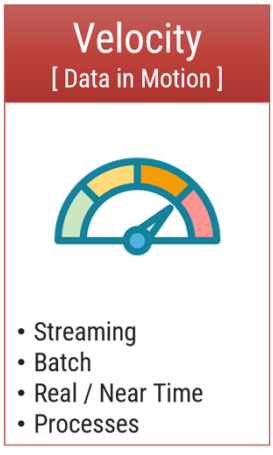
- Visualization is the process of displaying data in charts, graphs, maps, and other visual forms.
- Big data visualization is the process of displaying data in charts, graphs, maps, and other visual forms.
- It is used to help people easily understand and interpret their data briefly, and to clearly show trends and patterns that arise from this data.
- Raw data comes in a different format, 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.

Variety



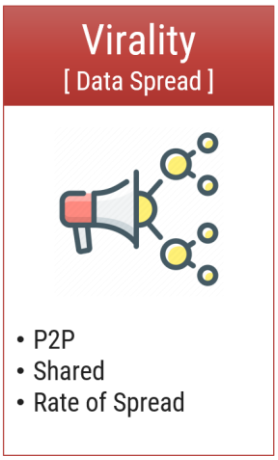
- Variety refers to the different data types i.e., various data formats like text, audios, videos, etc.
- Variety refers to heterogeneous sources and the nature of data, both structured and unstructured.
- Data comes in different formats – from structured, numeric data in traditional databases to unstructured 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.

Velocity



- Velocity is the rate at which data grows. Social media contributes a major role in the velocity of growing data.
- 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.



Virality

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

Who is using Big Data? Different applications of big data

Healthcare

- Big Data has already started to create a huge difference in the healthcare sector.
- With the help of predictive analytics, medical professionals and HCPs are now able to provide personalized healthcare services to individual patients.
- Apart from that, fitness wearable's, telemedicine, remote monitoring – all powered by Big Data and AI – are helping change lives for the better.

Academia

- Big Data is also helping enhance education today.
- Education is no more limited to the physical bounds of the classroom – there are numerous online educational courses to learn from.
- Academic institutions are investing in digital courses powered by Big Data technologies to aid the all-round development of budding learners.

Banking

- The banking sector relies on Big Data for fraud detection.
- Big Data tools can efficiently detect fraudulent acts in real-time such as misuse of credit/debit cards, archival of inspection tracks, faulty alteration in customer stats, etc.

Manufacturing

- According to TCS Global Trend Study, the most significant benefit of Big Data in manufacturing is improving the supply strategies and product quality.
- In the manufacturing sector, Big data helps create a transparent infrastructure, thereby, predicting uncertainties and incompetencies that can affect the business adversely.

IT

- One of the largest users of Big Data, IT companies around the world are using Big Data to optimize their functioning, enhance employee productivity, and minimize risks in business operations.
- By combining Big Data technologies with ML and AI, the IT sector is continually powering innovation to find solutions even for the most complex of problems.

Retail

- Big Data has changed the way of working in traditional brick and mortar retail stores.
- Over the years, retailers have collected vast amounts of data from local demographic surveys, POS scanners, RFID, customer loyalty cards, store inventory, and so on.
- Now, they've started to leverage this data to create personalized customer experiences, boost sales, increase revenue, and deliver outstanding customer service.
- Retailers are even using smart sensors and Wi-Fi to track the movement of customers, the most frequented aisles, for how long customers linger in the aisles, among other things.
- They also gather social media data to understand what customers are saying about their brand, their services, and tweak their product design and marketing strategies accordingly.

Transportation

- Big Data Analytics holds immense value for the transportation industry.
- In countries across the world, both private and government-run transportation companies use Big Data technologies to optimize route planning, control traffic, manage road congestion, and improve services.
- Additionally, transportation services even use Big Data to revenue management, drive technological innovation, enhance logistics, and of course, to gain the upper hand in the market.

Traditional Data vs. Big Data

Sr.	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.
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.

Big Data Case Study

1. Walmart

- Walmart uses big data and data mining to make personalized product recommendations for its customers.
- With the help of these two new technologies, Walmart offers the most frequently purchased products, the most popular products, and even the most popular product packages (complementary and usually purchased together).

- You can find valuable samples to show. Based on this knowledge, Walmart makes attractive and customized recommendations for individual users.
- By effectively implementing data mining technology, retail giants have improved conversion rates and significantly improved customer service.
- In addition, Walmart uses Hadoop and NoSQL technologies to give customers access to real-time data from a variety of sources.

American Express

- Credit card giants use vast amounts of customer data to identify indicators of potential customer loyalty.
- We also use big data to create advanced predictive models for analyzing historical transactions with 115 different variables to predict potential customer attrition.
- Due to big data solutions and tools, American Express can identify 24% of accounts that are likely to be closed in the next 4-5 months.

General Electric

- In the words of General Electric Chairman Jeff Immelt, GE has succeeded in integrating the strengths of both the "physical and analytical worlds" over the past few years.
- GE makes extensive use of big data. Every machine that runs under General Electric produces data about how it works.
- The GE analytics team then processes this vast amount of data to gain relevant insights and redesign the machine and its operations accordingly.
- Today, companies recognize that even small improvements can play an important role in the company's infrastructure.
- According to GE statistics, big data could increase US productivity by 1.5% and increase average national income by a staggering 30% in 20 years.

Uber

- Uber is one of the largest taxi service providers in the world. Use customer data to track and identify the most popular and most frequently used services for your users.
- Once this data is collected, Uber uses data analytics to analyze customer usage behavior and decide which services should be given more weight and importance.
- Uber controls big data in another unique way. Uber carefully researches the supply and demand of services and changes taxi fares accordingly.
- This is a surge pricing mechanism that works this way. If you're in a hurry and need to book a taxi from a crowded location, Uber will charge you twice as much as usual.

Netflix

- Netflix is one of the most popular on-demand online video content streaming platform used by people around the world.

- Netflix is a major proponent of the recommendation engine. It collects customer data to understand the specific needs, preferences, and taste patterns of users. Then it uses this data to predict what individual users will like and create personalized content recommendation lists for them.
- Today, Netflix has become so vast that it is even creating unique content for users.
- Data is the secret ingredient that fuels both its recommendation engines and new content decisions.
- The most pivotal data points used by Netflix include titles that users watch, user ratings, genres preferred, and how often users stop the playback, to name a few.
- Hadoop, Hive, and Pig are the three core components of the data structure used by Netflix.

Procter & Gamble (P&G)

- Procter & Gamble has been around us for ages now. However, despite being an "old" company, P&G is nowhere close to old in its ways.
- Recognizing the potential of Big Data, P&G started implementing Big Data tools and technologies in each of its business units all over the world.
- The company's primary focus behind using Big Data was to utilize real-time insights to drive smarter decision making.
- To accomplish this goal, P&G started collecting vast amounts of structured and unstructured data across R&D, supply chain, customer-facing operations, and customer interactions, both from company repositories and online sources.
- The global brand has even developed Big Data systems and processes to allow managers to access the latest industry data and analytics.

Unit-3: NoSQL

Introduction to NoSQL

What is NoSQL (Not Only SQL)?

- The term NoSQL was first coined by Carlo Strozzi in 1998 to name his lightweight, open-source, relational database that did not expose the standard SQL interface. Johan Oskarsson, who was then a developer at last.
- In 2009 reintroduced the term NoSQL at an event called to discuss open-source distributed network.
- The NoSQL was coined by Eric Evans and few other databases people at the event found it suitable to describe these non-relational databases.
- Few features of NoSQL databases are as follows:
    1. They are open source.
    2. They are non-relational.
    3. They are distributed.
    4. They are schema-less.
    5. They are cluster friendly.
    6. They are born out of 21st century web applications.

Where is it Used?

- NoSQL databases are widely used in big data and other real-time web applications. Refer Figure - 1. NoSQL databases is used to stock log data which can then be pulled for analysis.
- Likewise, it is used to store social media data and all such data which cannot be stored and analyzed comfortably in RDBMS.

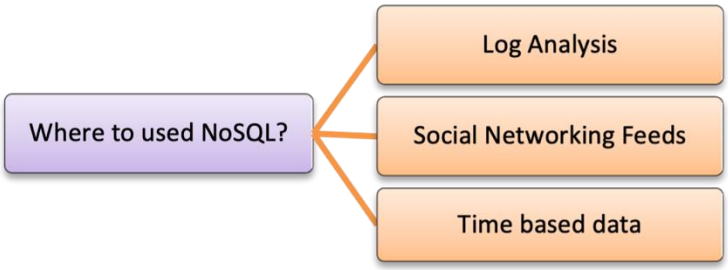


Figure 1 - Where to use NoSQL?

What is it?

- NoSQL stands for Not Only SQL. These are non-relational, open source, distributed databases.

- They are hugely popular today owing to their ability to scale out or scale horizontally and the adeptness at dealing with a rich variety of data: structured, semi-structured and unstructured data. Refer Figure - 2 for additional features of NoSQL.
- **NoSQL databases are non-relational:**
  - They do not adhere to relational data model. In fact, they are either key–value pairs or document-oriented or column-oriented or graph-based databases.
- **NoSQL databases are distributed:**
  - They are distributed meaning the data is distributed across several nodes in a cluster constituted of low-cost commodity hardware.
- **NoSQL databases offer no support for ACID properties (Atomicity, Consistency, Isolation, and Durability):**
  - They do not offer support for ACID properties of transactions.
  - On the contrary, they have adherence to Brewer's CAP (Consistency, Availability, and Partition tolerance) theorem and are often seen compromising on consistency in favor of availability and partition tolerance.
- **NoSQL databases provide no fixed table schema:**
  - NoSQL databases are becoming increasing popular owing to their support for flexibility to the schema.
  - They do not mandate for the data to strictly adhere to any schema structure at the time of storage.

NoSQL Business Drivers

- The demands of volume, velocity, variability, and agility play a key role in the emergence of NoSQL solutions.
- As each of these drivers applies pressure to the single-processor relational model, its foundation becomes less stable and in time no longer meets the organization's needs.
- The business driver's volume, velocity, variability, and agility apply pressure to the single CPU system, resulting in the failures.

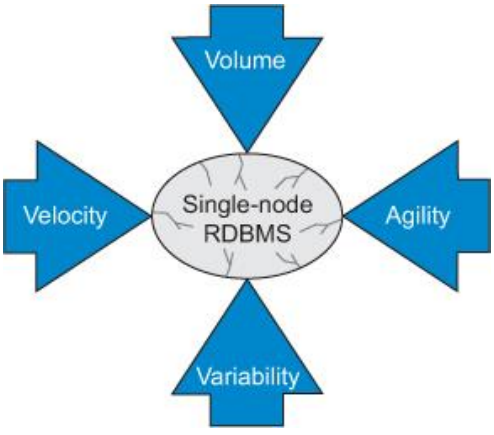


Figure 2 - NoSQL Business Drivers

- Volume and velocity refer to the ability to handle large datasets that arrive quickly.
- Variability refers to how diverse data types don't fit into structured tables.
- Agility refers to how quickly an organization responds to business change.

NoSQL Business Driver – Volume

- The key factor that led organizations to seek alternatives to their current RDBMS was the need to use commodity processor clusters to query big data.
- Early 2005 that performance problems were solved by buying faster processors.
- Over time, the ability to increase processing speed is no longer an option. As chip density increases, heat cannot be quickly dissipated if chips are overheated.
- This phenomenon, known as the power wall, forces system designers to shift their attention from increasing the speed of a single chip to using more processors to work together.

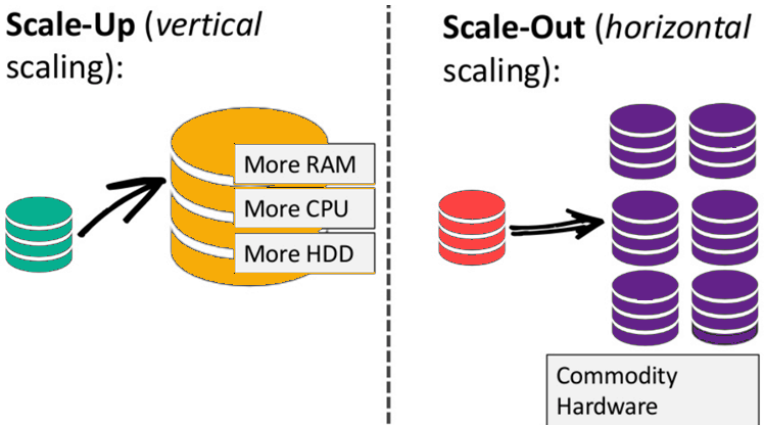


Figure 3 - Business Driver - Volume

- The need for horizontal scaling instead of vertical scaling (faster processors) shifts the organization from serial processing to parallel processing, where data problems are broken down into separate paths and sent to separate processors to divide and conquer.

NoSQL Business Driver – Velocity

- A big data issues are a consideration for many organizations far from RDBMS, the ability of uniprocessor systems to quickly read and write data is also critical.
- Many single-processor RDBMSs cannot meet the real-time insertion and online database query needs of public websites.
- RDBMS often indexes many columns in each new row, a process that reduces system performance.
- When a single-processor RDBMS is used as the back end of a web store front end, random bursts in web traffic will slow down everyone's response speed, and the cost of adjusting these systems when high read and write performance is required can be high.

NoSQL Business Driver – Variability

- An organization that want to capture and report abnormal data will encounter difficulties when trying to use the strict database schema structure enforced by the RDBMS.
- For example, if a business unit wants to capture some custom fields for a specific customer, all customer rows in the database must store this information, even if it is not applicable.
- Adding a new column to the RDBMS requires shutting down the system and executing the ALTER TABLE command.
- When the database is large, this process affects the availability of the system, which consumes time and money.

NoSQL Business Driver – Agility

- The most complex part of creating an application with RDBMS is the process of entering and extracting data from the database.
- If your data has nested and repeated data structure subgroups, you must include an object-relational mapping layer.
- The responsibility of this layer is to generate the correct combination of SQL INSERT, UPDATE, DELETE, and SELECT statements to move object data in and out of the RDBMS persistence layer.
- This process is not simple. When developing new applications or modifying existing applications, it is the biggest obstacle to rapid change.
- Object-relational mapping usually requires object-relational frameworks such as Java Hibernate (or NHibernate for .Net systems).

Types of NoSQL

- Traditional RDBMS uses SQL syntax to store and retrieve data from SQL databases.
- They all use a data model that has a different structure than the traditional row-and-column table model used with relational database management systems (RDBMSs).
- Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.
- They can be broadly classified into the following:
  1. **Key-Value Pair Oriented**
     - Key-value stores are the simplest type of NoSQL database.
     - Data is stored in key/value pairs.
     - It uses keys and values to store the data. The attribute name is stored in ‘key’, whereas the values corresponding to that key will be held in ‘value’.

Key	Value
First Name	Rahul
Last Name	Mehta

- In Key-value store databases, the key can only be string, whereas the value can store string, JSON, XML, Blob, etc. Due to its behavior, it is capable of handling massive data and loads.
- The use case of key-value stores mainly stores user preferences, user profiles, shopping carts, etc.
- DynamoDB, Riak, Redis are a few famous examples of Key-value store NoSQL databases.
- **Use cases:**
  - For storing user session data
  - Maintaining schema-less user profiles
  - Storing user preferences
  - Storing shopping cart data

2. Document Oriented

- Document Databases use key-value pairs to store and retrieve data from the documents.
- Documents can contain many different key-value pairs, or key-array pairs, or even nested documents. MongoDB is the most popular of these databases.
- A document is stored in the form of XML and JSON.
- Data is stored as a value. Its associated key is the unique identifier for that value.
- The difference is that, in a document database, the value contains structured or semi-structured data.
- Example:

```
{
"Book Name": "Fundamentals of Business Analytics",
"Publisher":
"Wiley India",
"Year of Publication": "2011"
}
```

- This structured/semi-structured value is referred to as a document and can be in XML, JSON or BSON format.
- Examples of Document databases are – MongoDB, OrientDB, Apache CouchDB, IBM Cloudant, CrateDB, BaseX, and many more.
- **Use cases:**
  - E-commerce platforms
  - Content management systems
  - Analytics platforms
  - Blogging platforms

3. Column Oriented

- Column based database store data together as columns instead of rows and are optimized for queries over large datasets.
- It works on columns and are based on BigTable paper by Google.

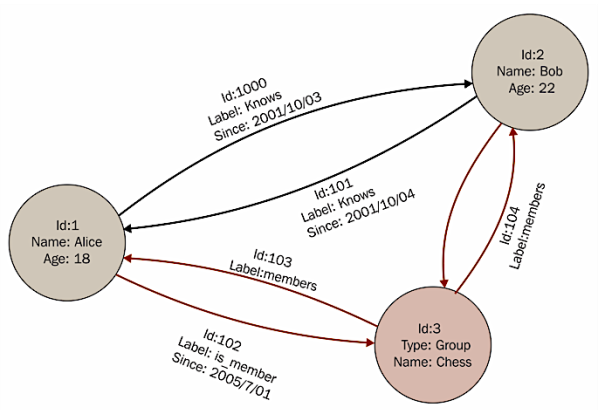
- Every column is treated separately. Values of single column databases are stored contiguously.

Column Family			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

- They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.
- HBase, Cassandra, HBase, Hypertable are NoSQL query examples of column-based database.
- **Use cases:**
  - Content management systems
  - Blogging platforms
  - Systems that maintain counters
  - Services that have expiring usage
  - Systems that require heavy write requests (like log aggregators)

4. Graph Oriented

- Graph databases form and store the relationship of the data.
- Each element/data is stored in a node, and that node is linked to another data/element.
- A typical example for Graph database use cases is Facebook.
- It holds the relationship between each user and their further connections.
- Graph databases help search the connections between data elements and link one part to various parts directly or indirectly.



- The Graph database can be used in social media, fraud detection, and knowledge graphs. Examples of Graph Databases are – Neo4J, Infinite Graph, OrientDB, FlockDB, etc.

- **Use cases:**
  - Fraud detection
  - Graph based search
  - Network and IT operations
  - Social networks, etc

Why NoSQL in Big Data?

1. It has scale out architecture instead of the monolithic architecture of relational databases.
2. It can house large volumes of structured, semi-structured, and unstructured data.
3. **Dynamic schema:** NoSQL database allows insertion of data without a pre-defined schema. In other words, it facilitates application changes in real time, which thus supports faster development, easy code integration, and requires less database administration.
4. **Auto-sharding:** It automatically spreads data across an arbitrary number of servers. The application in question is more often not even aware of the composition of the server pool. It balances the load of data and query on the available servers; and if and when a server goes down, it is quickly replaced without any major activity disruptions.
5. **Replication:** It offers good support for replication which in turn guarantees high availability, fault tolerance, and disaster recovery.

Using NoSQL to Manage Big Data

- The main reason behind organization moving towards a NoSQL solution and leaving the RDBMS system behind is the requirement to analyze a large volume of data.
- It is any business problem which could be so large and single processor cannot manage it.
- We need to move single processor environment to distributed computing environment due to big data problem. It has own problems and challenges while solving big data problems.

Advantages of NoSQL

1. **It can easily scale up and down:** NoSQL database supports scaling rapidly and elastically and even allows to scale to the cloud.
   - Cluster scale: It allows distribution of database across 100+ nodes often in multiple data centers.
   - Performance scale: It sustains over 100,000+ database reads and writes per second.
   - Data scale: It supports housing of 1 billion+ documents in the database.
2. **Doesn't require a pre-defined schema:** NoSQL does not require any adherence to pre-defined schema.
   1. It is pretty flexible. For example, if we look at MongoDB, the documents (equivalent of records in RDBMS) in a collection (equivalent of table in RDBMS) can have different sets of key–value pairs.

{_id: 101,"BookName": "Fundamentals of Business Analytics", "AuthorName": "Seema Acharya", "Publisher": "Wiley India"} {_id:102, "BookName":"Big Data and Analytics"}

3. **Cheap, easy to implement:** Deploying NoSQL properly allows for all of the benefits of scale, high availability, fault tolerance, etc. while also lowering operational costs.
4. **Relaxes the data consistency requirement**: NoSQL databases have adherence to CAP theorem (Consistency, Availability, and Partition tolerance). Most of the NoSQL databases compromise on consistency in favour of availability and partition tolerance. However, they do go for eventual consistency.
5. **Data can be replicated to multiple nodes and can be partitioned**: There are two terms that we will discuss here:
    - **Sharding**: Sharding is when different pieces of data are distributed across multiple servers.
    - NoSQL databases support auto-sharding; this means that they can natively and automatically spread data across an arbitrary number of servers, without requiring the application to even be aware of the composition of the server pool.
    - Servers can be added or removed from the data layer without application downtime.
    - This would mean that data and query load are automatically balanced across servers, and when a server goes down, it can be quickly and transparently replaced with no application disruption.
    - **Replication**: Replication is when multiple copies of data are stored across the cluster and even across data centers. This promises high availability and fault tolerance.

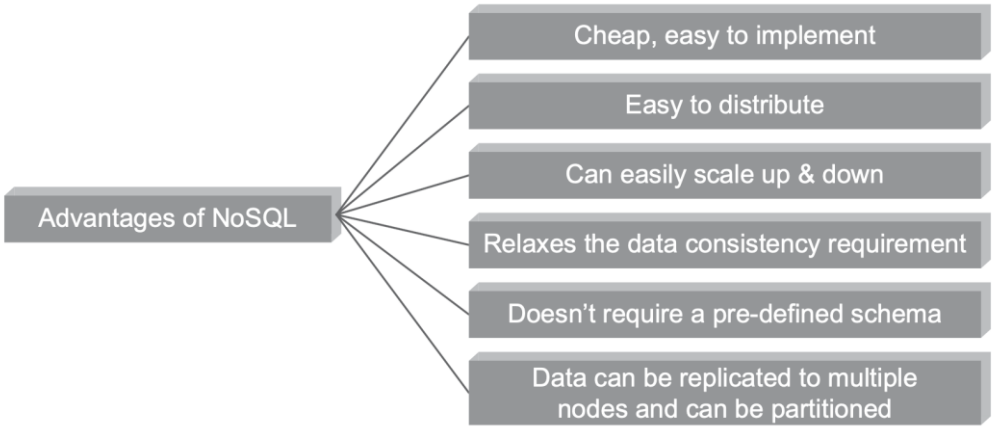


Figure 4 - Advantages of NoSQL

Four ways that NoSQL System handle Big Data Problems

1. Moving Queries to the data, Not Data to the Queries
2. Using Hash Rings to Evenly Distribute Data on a Cluster
3. Using Replication to Scale Reads

4. Letting the Database Distribute Queries Evenly to Data Nodes

SQL Vs. NoSQL

SQL	**NoSQL**
Relational database	Non-relational, distributed database
Relational model	Model-less approach
Pre-defined schema	Dynamic schema for unstructured data
Table based databases	Document-based or graph-based or wide column store or key–value pairs databases
Vertically scalable (by increasing system resources)	Horizontally scalable (by creating a cluster of commodity machines)
Uses SQL	Uses UnSQL (Unstructured Query Language)
Not preferred for large datasets	Largely preferred for large datasets
Not a best ft for hierarchical data	Best ft for hierarchical storage as it follows the key–value pair of storing data like JSON (Java Script Object Notation)
Excellent support from vendors	Relies heavily on community support
Supports complex querying and data keeping needs	Does not have good support for complex querying
Can be configured for strong consistency	Few supports strong consistency (e.g., MongoDB), some others can be configured for eventual consistency (e.g., Cassandra)
Examples: Oracle, DB2, MySQL, MS SQL, PostgreSQL, etc.	Examples: MongoDB, HBase, Cassandra, Redis, Neo4j, CouchDB, Couchbase, Riak, etc.

NoSQL Vendors

- Refer Table for few popular NoSQL vendors.

Company	**Product**	**Most Widely Used by**
Amazon	DynamoDB	LinkedIn, Mozilla
Facebook	Cassandra	Netfix, Twitter, eBay
Google	BigTable	Adobe Photoshop

Unit-4: Mining Data Stream

Introduction Data Stream Mining

What is Data Stream Mining?

- It is an activity of collecting insights from continuous high-speed data records which comes to the system in a stream.

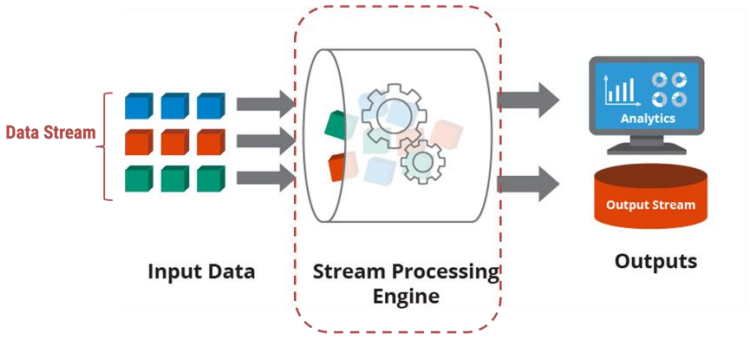


Figure 1 - Data Stream Mining

Characteristics of Data Stream Mining

1. Continuous Stream of Data:
    - High amount of data in an infinite stream.
    - We do not know entire database.
2. Concept Drifting
    - The data keep growing or changing over time.
3. Volatility of Data
    - The system does not store the data received – limited resource. When data is analyzed, it is discarded or summarized.

Stream Data & Processing

Stream Data

- Streaming data is becoming a core component of enterprise data architecture due to the explosive growth of data from non-traditional sources such as IoT sensors, security logs and web applications.
- Streaming data refers to data that is continuously generated, usually in high volumes and at high velocity.
- A streaming data source would typically consist of a stream of logs that record events as they happen – such as a user clicking on a link in a web page, or a sensor reporting the current temperature.
- Examples of streaming data include:

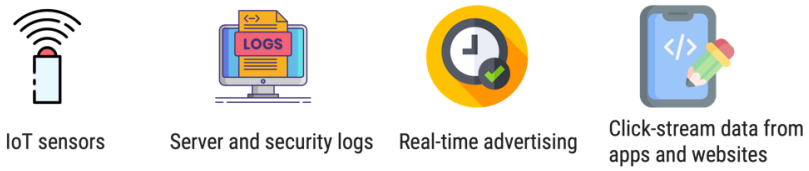


Figure 2 - Example of Stream Data

Stream Processing

- Stream processing used to be a 'niche' technology used only by a small subset of companies.
- However, with the rapid growth of SaaS, IoT and machine learning, organizations across industries are now reducing their feet into streaming analytics.
- It's difficult to find a modern company that doesn't have an app or a website; as traffic to these digital assets grows, and with increasing craving for complex and real-time analytics, the need to adopt modern data infrastructure is quickly becoming mainstream.
- While traditional batch architectures can be sufficient at smaller scales, stream processing provides several benefits that other data platforms cannot given.

Benefit of Stream Processing

1. ***Able to deal with never-ending streams of events***
   - Several kinds of data are naturally structured.
   - Traditional batch processing tools require stopping the stream of events.
   - It captures batches of data and combining the batches to draw overall conclusions.
   - It provides immediate insights from large volumes of streaming data.

2. ***Real-time or Near-Real-time Processing***
   - Most organizations adopt stream processing to enable real time data analytics.
   - Even though with high performance database systems in real time analytics is possible, and data is moved to stream processing model.

3. ***Detecting Patterns in time-series data***
   - Identify patterns over time, for example trends in website traffic data.
   - It requires data to be continuously processed and analyzed.
   - Batch processing makes this more difficult because it breaks data into batches, meaning some events are broken across two or more batches.

4. ***Easy Data Scalability***
   - Increasing data volumes can break a batch processing system which required additional resources or modify the architecture.
   - Modern stream processing infrastructure is hyper-scalable, able to deal with Gigabytes of data per second with a single stream processor which deal easily with growing data volumes without infrastructure changes.

Stream Data Model and Architecture

- A streaming data architecture is a framework of software components construct to absorb and process large amount of streaming data from numerous sources.
- While traditional data solutions focused on writing and reading data in batches.
- A streaming data architecture ingest data immediately as it is produced, is persistence in its storage, and may include various additional components per use case - such as tools for real-time processing, data manipulation and analytics.
- Streaming architectures must be able to address for the unique feature of data streams, which tend to make huge amounts of data (terabytes to petabytes) that it is at best semi-structured and requires significant pre-processing.

Streaming Architecture Components

The Message Broker / Stream Processor

- This is the element that takes data from a source, called a producer, translates it into a standard message format, and streams it on an ongoing basis.
- Other components can then listen in and consume the messages passed on by the broker.
- The first generation of message brokers, such as RabbitMQ and Apache ActiveMQ which relied on the Message Oriented Middleware (MOM) paradigm.
- Two popular stream processing tools are Apache Kafka and Amazon Kinesis Data Streams.

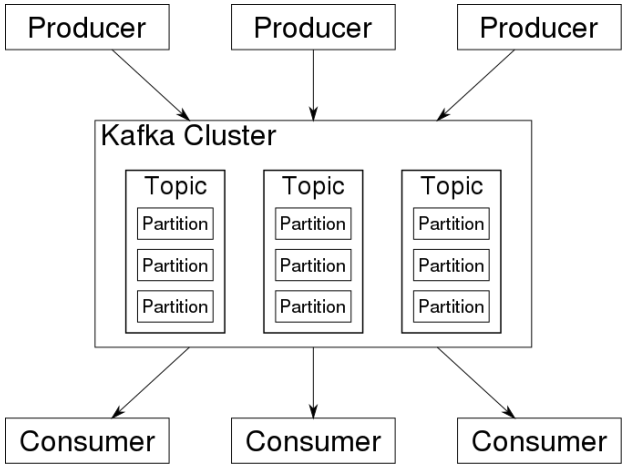


Figure 3 - Message Broker / Stream Processor

Batch and Real-time ETL tools

- Data streams from one or more message brokers need to be accumulated, reconstruct and restructured before data can be analyzed with SQL-based analytics tools.
- This should be possible by an ETL tool or platform receives queries from users, fetches events from message queues and applies the query, to generate a result – often performing additional joins, transformations on aggregations on the data.

- The result may be an API call, an action, a visualization, an alert, or in some cases a new data stream.

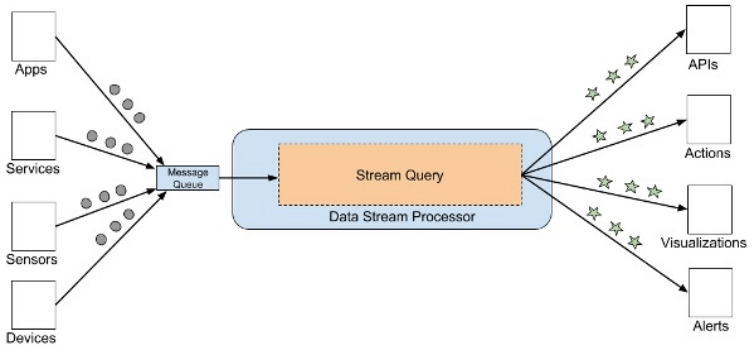


Figure 4 - Batch and Real-time ETL tools

Data Analytics / Serverless Query Engine

- After streaming data is prepared for consumption by the stream processor, it must be visualizing providing value.
- There are many numerous approaches to streaming data analytics.

Streaming Data Storage

- As every organization has a massive data to store, they opt for cheap storage options. So they choose storing data in streaming way.

Stream Computing

- A high-performance computer system that analyzes multiple data streams from many sources live.
- The word stream in stream computing is used to mean pulling in streams of data, processing the data and streaming it back out as a single flow.

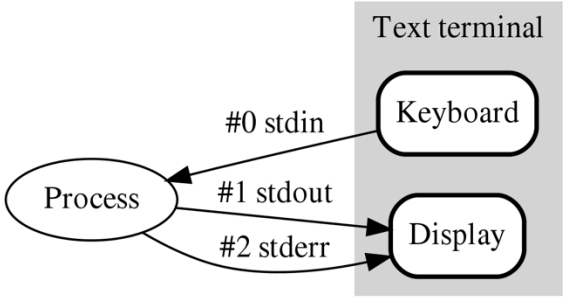


Figure 5 - Stream Computing

- Stream computing uses software algorithms that analyzes the data in real time as it streams in to increase speed and accuracy when dealing with data handling and analysis.

Sampling Data in Stream

- The sample in data stream is taken much smaller than the whole stream.
- This can be designed to retain many relevant features of the stream. The same can be used to calculate many relevant combinations on the stream.
- Unlike sampling from a stored data set, stream sampling should be performed on the web, when data arrives. Any component that is not stored inside the sample is lost everlastingly and cannot be recovered.

Filtering Stream

- Due to the nature of data streams, stream filtering is one out of the major useful and practical approaches to efficient stream evaluation.
- Two filtering approaches are in Data Stream Management Systems:
  1. Implicit Filtering
  2. Explicit Filtering

Implicit Filtering

- Data stream management system cope with the high rates and the bursty nature of streams in several ways in the order to guarantee stability under heavy loads.
- Some of them employ various load shedding techniques which reduce the load by processing only a fraction of the items from the stream and discarding others without any processing.
- The Aurora DSMS employs random and semantic load shedding techniques to deal with the unpredictable nature of data streams, where semantic load shedding makes use of tuple utility computed based on quality of service – QoS parameters.
- Automatically, the system drops tuples that are assumed to be less important for stream evaluation.
- QoS of the system is captured by several functions:
- Latency graph, which specifies the utility of a tuple as a function of tuple propagation through the query plan.
- Value-based graph, which specifies which values of the output are more important than others.
- Loss tolerant graph, which describes how sensitive the application is to approximate answer.
- A strategy of dropping tuples at the early stages of the query plan makes the process of query evaluation more efficient for subsequent operators in the plan.

Explicit Filtering

- As per implicit filtering techniques may have negative impacts on a variety of data steam analyze problems, such as computation of sketches and samples of distinct items and other properties of stream.

- Also, problem in this category include estimation of IP network flow sizes, detection of worm signature generation in the network.
- Fine-grained estimation of network traffic (flows) volume is very important in various network analysis tasks.
- It offers threshold sampling algorithm that generates sample of stream items with guarantees on estimated flow sizes.
- The procedure of sample generation maintains a value of threshold, which is compared to the size of item, and make decision whether item of stream filtered out or retained in sample.
- Due to the nature of this sampling algorithm, with a number of various parameters, such as few items in the final sample, item size, threshold value, count of items larger than the threshold value.

Counting Distinct Elements in Stream

- Let a = a1,... ,an be a sequence of n elements from the domain [m] = {1, . . . , m}.
- The zeroth-frequency moment of this sequence is the number of distinct elements that occur in the sequence and is denoted F0 = F0(a).
- In the data stream model, an algorithm is considered efficient if it makes one (or a small number of) passes over the input sequence, uses very little space, and processes each element of the input very quickly.
- In our context, a data stream algorithm to approximate F0 is considered efficient if it uses only poly(1/ε, log n, log m) bits of memory, where 1 ± ε is the factor within which F0 must be approximated.
- Let ε, δ > 0 be given.
- An algorithm A is said to (ε, δ)-approximate F0 if for any sequence a = a ,... ,a , with each ai ∈ [m], it outputs a number F˜0 such that Pr [! ! ! F0 − F˜0 ! ! ! ≤ ε F0] ≥ 1−δ, where the probability is taken over the internal coin tosses of A.
- Two main parameters of A are of interest: the workspace and the time to process each item. We study these quantities as functions of the domain size m, the number n of elements in the stream, the approximation parameter €, and the confidence parameter δ.

Estimating Moments

- Alon-Matias-Szegedy (AMS) Algorithm
  - Ex Stream : a, b, c, b, d, a, c, d, a, b, d, c, a, a, b
  - n=15; a(x5), b(x4), c(×3), d(x3); 2nd Moment = 59
  - Estimate = n * (2 * X.value - 1)
  - Pick random positions X1(3rd), X2(8th), X3(13th)
  - X1.element = "c", X1.value = 3 (# of "c" in the set from 3rd place)
  - Estimate for X1= 15*(2*3-1) = 75
  - Estimates for X2 = 15* (2*2-1) = 45, (# "d"beyond 8^{th} place = 2)

- Estimate for X3 = 45, (# "a" beyond 13th place = 2)
- Average for X1, X2, X3 = 165/3 = 55 (close to 59)

- In case of infinite streams
  - As we store one variable per randomly chosen position, so the challenge is not selecting 'n'.
  - Selecting the position is challenge.
- Strategy for position selection assuming we have space to store "s" variables, and we have seen "n' elements.
  - First "s" positions are chosen.
  - When $(n+1)^{th}$ token arrives, random selection takes place.
  - If (n+1) is selected discard from existing n position randomly and insert new element with value 1.

Counting Oneness in a Window

- Let's suppose a window of length N on a binary stream. We want at all times to be able to answer queries of the form "how many 1's are there in the last k bits?" for any k≤ N. For this purpose we use the DGIM algorithm.
- The basic version of the algorithm uses O(log2 N) bits to represent a window of N bits, and allows us to estimate the number of 1's in the window with an error of no more than 50%.
- To begin, each bit of the stream has a timestamp, the position in which it arrives. The first bit has timestamp 1, the second has timestamp 2, and so on.
- Since we only need to distinguish positions within the window of length N, we shall represent timestamps modulo N, so they can be represented by log2 N bits.
- If we also store the total number of bits ever seen in the stream (i.e., the most recent timestamp) modulo N, then we can determine from a timestamp modulo N where in the current window the bit with that timestamp is.
- We divide the window into buckets, 5 consisting of:
  - The timestamp of its right (most recent) end.
  - The number of 1's in the bucket. This number must be a power of 2, and we refer to the number of 1's as the size of the bucket.
- To represent a bucket, we need log2 N bits to represent the timestamp (modulo N) of its right end.
- To represent the number of 1's we only need log2 log2 N bits. The reason is that we know this number i is a power of 2, say 2j , so we can represent i by coding j in binary.
- Since j is at most log2 N, it requires log2 log2 N bits. Thus, O(logN) bits suffice to represent a bucket.

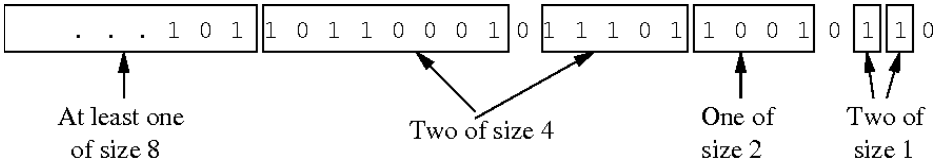


Figure 6 - A bit stream divided into bucket following DGIM rules

- There are six rules that must be followed when representing a stream by buckets.
    1. The right end of a bucket is always a position with a 1.
    2. Every position with a 1 is in some bucket.
    3. No position is in more than one bucket.
    4. There are one or two buckets of any given size, up to some maximum size.
    5. All sizes must be a power of 2.
    6. Buckets cannot decrease in size as we move to the left (back in time).

Decaying Window

- This algorithm allows you to identify the most popular elements (trending, in other words) in an incoming data stream.
- The decaying window algorithm not only tracks the most recurring elements in an incoming data stream, but also discounts any random spikes or spam requests that might have boosted an element's frequency.
- In a decaying window, you assign a score or weight to every element of the incoming data stream. Further, you need to calculate the aggregate sum for each distinct element by adding all the weights assigned to that element.
- The element with the highest total score is listed as trending or the most popular.
    1. Assign each element with a weight/score.
    2. Calculate aggregate sum for each distinct element by adding all the weights assigned to that element.
- In a decaying window algorithm, you assign more weight to newer elements.
- For a new element, you first reduce the weight of all the existing elements by a constant factor k and then assign the new element with a specific weight.
- The aggregate sum of the decaying exponential weights can be calculated using the following formula:

$$\sum t - 1i = 0\ at - i(1 - c)i$$

- Here, c is usually a small constant of the order.
- Whenever a new element, say $at + 1$, arrives in the data stream you perform the following steps to achieve an updated sum:
    - Multiply the current sum/score by the value (1−c).
    - Add the weight corresponding to the new element.

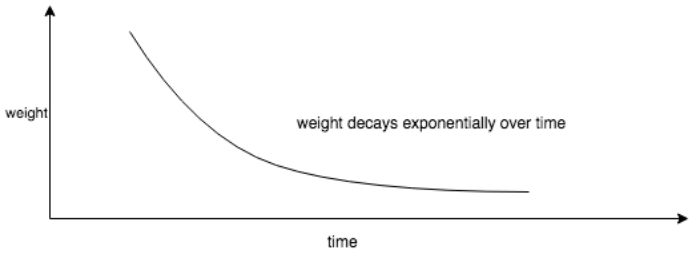


Figure 7 - Weight decays exponentially over time

- In a data stream consisting of various elements, you maintain a separate sum for each distinct element.
- For every incoming element, you multiply the sum of all the existing elements by a value of (1−c). Further, you add the weight of the incoming element to its corresponding aggregate sum.
- A threshold can be kept to, ignore elements of weight lesser than that.
- Finally, the element with the highest aggregate score is listed as the most popular element.

Example

- For example, consider a sequence of twitter tags below:
  - fifa, ipl, fifa, ipl, ipl, ipl, fifa
- Also, let's say each element in sequence has weight of 1.
  - Let's c be 0.1
- The aggregate sum of each tag in the end of above stream will be calculated as below:
- **fifa:**
  - fifa - 1 * (1-0.1) = 0.9
  - ipl - 0.9 * (1-0.1) + 0 = 0.81 (adding 0 because current tag is different than fifa)
  - fifa - 0.81 * (1-0.1) + 1 = 1.729 (adding 1 because current tag is fifa only)
  - ipl - 1.729 * (1-0.1) + 0 = 1.5561
  - ipl - 1.5561 * (1-0.1) + 0 = 1.4005
  - ipl - 1.4005 * (1-0.1) + 0 = 1.2605
  - fifa - 1.2605 * (1-0.1) + 1 = 2.135
- **ipl:**
  - fifa - 0 * (1-0.1) = 0
  - ipl - 0 * (1-0.1) + 1 = 1
  - fifa - 1 * (1-0.1) + 0 = 0.9 (adding 0 because current tag is different than ipl)
  - ipl - 0.9 * (1-0.01) + 1 = 1.81
  - ipl - 1.81 * (1-0.01) + 1 = 2.7919
  - ipl - 2.7919 * (1-0.01) + 1 = 3.764
  - fifa - 3.764 * (1-0.01) + 0 = 3.7264
- In the end of the sequence, we can see the score of fifa is 2.135 but ipl is 3.7264
- So, ipl is more trending then fifa
- Even though both occurred same number of times in input their score is still different.

Advantages of Decaying Window Algorithm:

- Sudden spikes or spam data is taken care.
- New element is given more weight by this mechanism, to achieve right trending output.

Real Time Analytics Platform Application

- Fraud detection system for online transactions
- Log analysis for understanding usage pattern
- Click analysis for online recommendations
- Social media analytics
- Push notifications to the customers for location-based advertisements for retails
- Action for emergency services such as fires and accidents in an industry
- Any abnormal measurements require immediate reaction in healthcare monitoring

Real-time RTAP Application

1. ***Apache Samza***

- Apache Samza is an open source, near-real time, asynchronous computational framework for stream processing developed by the Apache Software Foundation in Scala and Java.
- Samza allows users to build stateful applications that process data in real-time from multiple sources including Apache Kafka.
- Samza provides fault tolerance, isolation and stateful processing. Samza is used by multiple companies. The biggest installation is in LinkedIn.

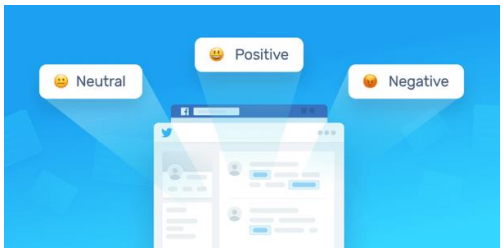
2. ***Apache Flink***

- Apache Flink is an open-source, unified stream-processing and batch-processing framework developed by the Apache Software Foundation.
- The core of Apache Flink is a distributed streaming data-flow engine written in Java and Scala.
- Fink provides a high-throughput, low-latency streaming engine as well as support for event-time processing and state management.
- Flink does not provide its own data-storage system, but provides data-source and sink connectors to systems such as Amazon Kinesis, Apache Kafka, Alluxio, HDFS, Apache Cassandra, and Elastic Search.

Real Time Sentiment Analysis

- Sentiment analysis is a text analysis tool that use of natural language processing (NLP), machine learning, and other data analysis techniques to read, analyze and derive objective quantitative results from raw text.
- These texts classify as positive, negative, neural and everywhere in between.
- It can read all manner of text for opinion and emotion – to understand the thoughts and feelings of writer.

- Sentiment analysis is also known as opinion mining.



- Real-time sentiment analysis is an AI-powered solution to track mentions of your brand and products, wherever they may appear, and automatically analyze them with almost no human input needed.

Figure 8 - Sentiment Analysis Process

- Benefits of performing real time sentiment analysis:
  - Marketing campaign success analysis
  - Prevention of business disasters
  - Instant product feedback
  - Stock market predictions
- Basic components of an opinion
  - Opinion Holder
  - Object
  - Opinion
- Opinion Mining Tasks
  - At Documents or review Level
    - Task: Sentiment classification of reviews
    - Classes: Positive, Negative and Neutral
  - At the Sentence Level
    - Task-1: Identifying subjective/opinionated sentences
      - Classes: Objective and subjective
    - Task-2: Sentiment classification of sentences
      - Classes: Positive, Negative, and Neutral
- Opinion Mining Tasks
  - At feature level
    - Task-1: Identify and extract object features that have been commented on by an opinion holder – e.g. reviewer
    - Task-2: Determine whether the opinions on the features are positive, negative or neutral

        - Task-3: Group Feature Synonyms – it produces featured based opinion summary of multiple review.
- Two types of evaluations:
    - Regular Opinions
        - Sentiment/opinion expression on some target entities, e.g. products, events, topics, persons.
        - Direct Opinion and Indirect Opinion
    - Comparative Opinion
        - Comparisons of more than one entity.
        - Example: Buy an iPhone and put a review.

Case Study – Stock Market Predictions

- The necessity of Data Mining
    - Data
    - Information
    - Business Decisions
- Principal of Stock market data mining algorithm.
- Graph Analytics and Big Data.
- Graph analytics is an analytics alternative that uses an abstraction called a graph model.
- Graph analytics is an alternative to the traditional data warehouse model as a framework for absorbing both structured and unstructured data.
- It employs the graph abstraction for representing connectivity, consisting of a collection of vertices – nodes.
- Predictable interactive performance graph.

Unit-6: Spark

Introduction of Spark

What is Spark?

- Spark extends the popular MapReduce model to efficiently support more types of computations, such as interactive queries and stream processing.
- Speed is important when working with large datasets. This is because it's the difference between browsing data interactively and waiting minutes or hours.
- One of the key speed features of Spark is the ability to perform calculations in memory, but the system is more efficient than MapReduce for complex applications running on disk.
- Spark is designed for high accessibility, Spark offers a simple API for Python, Java, Scala, SQL, and an extensive integrated library. It is also tightly integrated with other big data tools.
- Specifically, Spark can run on Hadoop clusters and can access any Hadoop data source, including Cassandra.

Spark Stack

Spark Core

- Spark Core contains the basic functionality of Spark, including components for task scheduling, memory management, fault recovery, interacting with storage systems, and more.

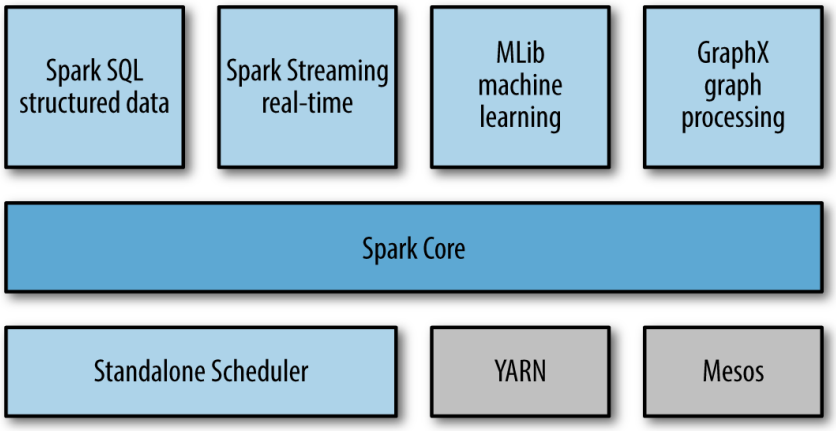


Figure 1 - Spark Stack

- Spark Core also hosts an API that defines Resilient Distributed Datasets (RDDs), the main programming abstraction for Spark.
- RDD represents a collection of elements that are distributed across many compute nodes and can be processed in parallel.
- Spark Core provides many APIs for creating and manipulating these collections.

Spark SQL

- Spark SQL is Spark's package for working with structured data.
- It allows querying data via SQL as well as the Apache Hive variant of SQL—called the Hive Query Language (HQL)—and it supports many sources of data, including Hive tables, Parquet, and JSON.
- Beyond providing a SQL interface to Spark, Spark SQL allows developers to intermix SQL queries with the programmatic data manipulations supported by RDDs in Python, Java, and Scala, all within a single application, thus combining SQL with complex analytics.
- This tight integration with the rich computing environment provided by Spark makes Spark SQL unlike any other open-source data warehouse tool.
- Spark SQL was added to Spark in version 1.0.
- Spark was an older SQL-on-Spark project out of the University of California, Berkeley, that modified Apache Hive to run on Spark.
- It has now been replaced by Spark SQL to provide better integration with the Spark engine and language APIs.

Spark Streaming

- Spark Streaming is a Spark component that enables processing of live streams of data.
- Examples of data streams include logfiles generated by production web servers, or queues of messages containing status updates posted by users of a web service.
- Spark Streaming provides an API for manipulating data streams that closely matches the Spark Core's RDD API, making it easy for programmers to learn the project and move between applications that manipulate data stored in memory, on disk, or arriving in real time.
- Underneath its API, Spark Streaming was designed to provide the same degree of fault tolerance, throughput, and scalability as Spark Core.

MLlib

- Spark comes with a library containing common machine learning (ML) functionality, called MLlib.
- MLlib provides multiple types of machine learning algorithms, including classification, regression, clustering, and collaborative filtering, as well as supporting functionality such as model evaluation and data import.
- It also provides some lower-level ML primitives, including a generic gradient descent optimization algorithm. All these methods are designed to scale out across a cluster.

GraphX

- GraphX is a library for manipulating graphs (e.g., a social network's friend graph) and performing graph-parallel computations, like Spark Streaming and Spark SQL.

- GraphX extends the Spark RDD API, allowing us to create a directed graph with arbitrary properties attached to each vertex and edge.
- GraphX also provides various operators for manipulating graphs (e.g., *subgraph* and *mapVertices*) and a library of common graph algorithms (e.g., PageRank and triangle counting).

Data Analysis with Spark

Data Science Tasks

- Data science, a discipline that has been emerging over the past few years, centers on analyzing data.
- While there is no standard definition, for our purposes a data scientist is somebody whose main task is to analyze and model data.
- Data scientists may have experience with SQL, statistics, predictive modelling (machine learning), and programming, usually in Python, MATLAB, or R.
- Data scientists also have experience with techniques necessary to transform data into formats that can be analyzed for insights (sometimes referred to as data wrangling).
- Data scientists use their skills to analyze data with the goal of answering a question or discovering insights.
- Oftentimes, their workflow involves ad hoc analysis, so they use interactive shells (versus building complex applications) that let them see results of queries and snippets of code in the least amount of time.
- Spark's speed and simple APIs shine for this purpose, and its built-in libraries mean that many
- algorithms are available out of the box.
- Spark supports the different tasks of data science with several components. The Spark shell makes it easy to do interactive data analysis using Python or Scala.
- Spark SQL also has a separate SQL shell that can be used to do data exploration using SQL, or Spark SQL can be used as part of a regular Spark program or in the Spark shell.
- Machine learning and data analysis is supported through the MLLib libraries. In addition, there is support for calling out to external programs in MATLAB or R.
- Spark enables data scientists to tackle problems with larger data sizes than they could before with tools like R or Pandas.
- Sometimes, after the initial exploration phase, the work of a data scientist will be "productized," or extended, hardened (i.e., made fault-tolerant), and tuned to become a production data processing application, which itself is a component of a business application.
- For example, the initial investigation of a data scientist might lead to the creation of a production recommender system that is integrated into a web application and used to generate product suggestions to users.

- Often it is a different person or team that leads the process of productizing the work of the data scientists, and that person is often an engineer.

Data Processing Applications

- The other major use cases for Spark can be explained in the context of an engineer's personality. For the purposes of here, we consider engineers to be a large class of software developers who use Spark to build applications for production computing.
- These developers usually understand the principles of software engineering, such as encapsulation, interface design, and object-oriented programming. They often have a degree in computer science.

Resilient Distributed Datasets (RDDs)

- RDDs are the main logical data unit in Spark.
- They are a distributed collection of objects, which are stored in memory or on disks of different machines of a cluster.
- A single RDD can be divided into multiple logical partitions so that these partitions can be stored and processed on different machines of a cluster.
- RDDs are immutable (read-only) in nature. You cannot change an original RDD, but you can create new RDDs by performing coarse-grain operations, like transformations, on an existing RDD.

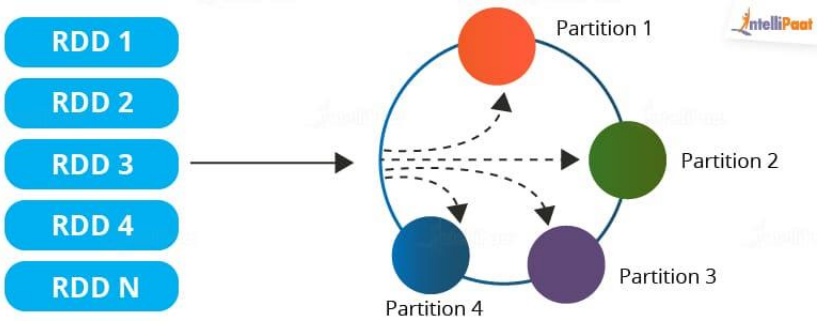


Figure 2 - RDDs

Features of RDD

Resilient

- RDDs track data lineage information to recover the lost data, automatically on failure. It is also called Fault tolerance.

Distributed

- Data present in the RDD resides on multiple nodes. It is distributed across different nodes of a cluster.

Lazy Evaluation

- Data does not get loaded in the RDD even if we define it. Transformations are computed when you call an action, like count or collect, or save the output to a file system.

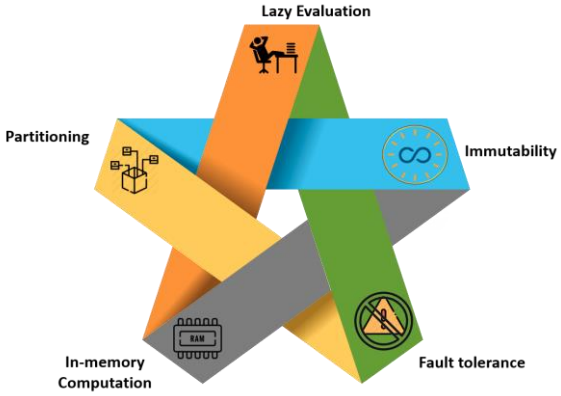


Figure 3 - Features of RDD

Immutability

- Data stored in the RDD is in a read-only mode you cannot edit the data which is present in the RDD. But you can create new RDDs by performing transformations on the existing RDDs.

In-memory Computation:

- RDD stores any immediate data that is generated in the memory (RAM) than on the disk so that it provides faster access.

Partitioning:

- Partitions can be done on any existing RDDs to create logical parts that are mutable. You can achieve this by applying transformations on existing partitions.

Operations of RDD

- There are two basic operations which can be done on RDDs. They are:
    1. Transformations
    2. Actions

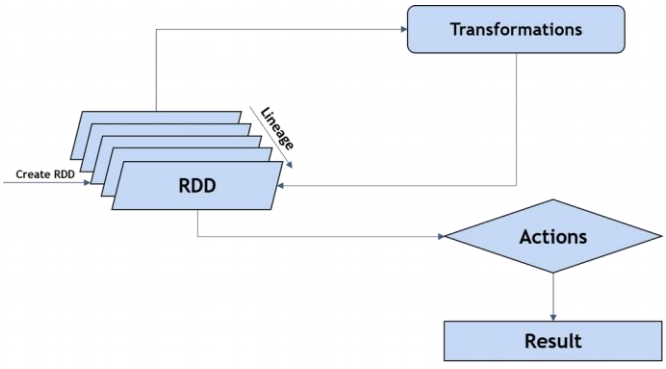


Figure 4 - Operations of RDD

Transformations

- These are functions which accept existing RDDs as the input and outputs one or more RDDs. The data in the existing RDDs does not change as it is immutable. Some of the transformation operations are shown in the table given below:

Functions	Description
map()	Returns a new RDD by applying the function on each data element
filter()	Returns a new RDD formed by selecting those elements of the source on which the function returns true
reduceByKey()	Used to aggregate values of a key using a function
groupByKey()	Used to convert a (key, value) pair to (key, <iterable value>) pair
union()	Returns a new RDD that contains all elements and arguments from the source RDD
intersection()	Returns a new RDD that contains an intersection of elements in the datasets

- These transformations are executed when they are invoked or called. Every time transformations are applied, a new RDD is created.

Actions:

- Actions in Spark are functions which return the result of RDD computations. It uses a lineage graph to load the data onto the RDD in a particular order.
- After all transformations are done, actions return the final result to the Spark Driver. Actions are operations which provide non-RDD values.
- Some of the common actions used in Spark are:

Functions	Description
count()	Gets the number of data elements in an RDD
collect()	Gets all data elements in the RDD as an array
reduce()	Aggregates data elements into the RDD by taking two arguments and returning one
take(n)	Used to fetch the first n elements of the RDD
foreach(operation)	Used to execute operation for each data element in the RDD
first()	Retrieves the first data element of the RDD

Creating Pair RDDs

- Spark provides special type of operations on RDDs containing key or value pairs. These RDDs are called pair RDDs operations.
- Pair RDDs are a useful building block in many programming language, as they expose operations that allow you to act on each key operations in parallel or regroup data across the network.
- Pair RDDs can be created by running a map() function that returns key or value pairs.
- The procedure to build the key-value RDDs differs by language. In Python language, for the functions on keyed data to work we need to return an RDD composed of tuples
- Creating a pair RDD using the first word as the key in Python programming language.

pairs = lines.map(lambda x: (x.split(" ")[0], x))

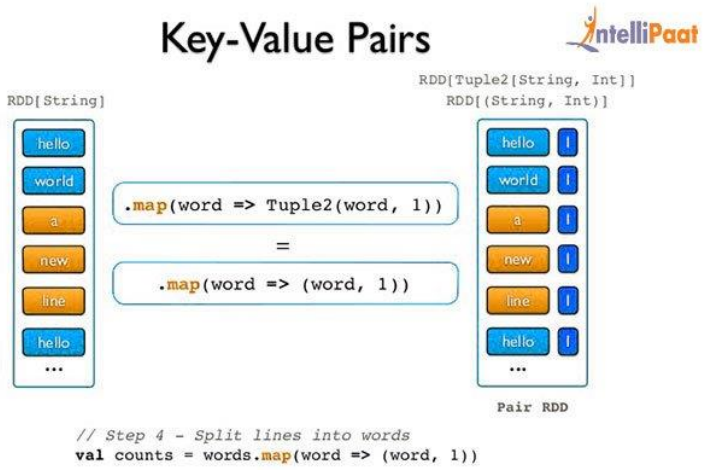


Figure 5 - Key-Value Pairs

- Java users also need to call special versions of Spark's functions when you are creating pair of RDDs.
- For instance, the mapToPair () function should be used in place of the basic map() function.
- Creating a pair RDD using the first word as the key word in Java program.

```
PairFunction<String, String, String> keyData =
new PairFunction<String, String, String>() {
public Tuple2<String, String> call(String x) {
return new Tuple2(x.split(" ")[0], x);}};
JavaPairRDD<String, String> pairs =
lines.mapToPair(keyData);
```

Transformations on Pair RDDs

Aggregations

- When datasets are described in terms of key or value pairs, it is common feature that is required to aggregate statistics across all elements with the same key value. Spark has a set of operations that combines values that own the same key value. These operations return RDDs and thus are transformations rather than actions i.e. reduceByKey(), foldByKey(), combineByKey().

Grouping Data

- With key data is a common type of use case in grouping our data sets is used with respect to predefined key value for example, viewing all a customer's orders together in one file.
- If our data is already keyed in the way we want to implement, groupByKey() will group our data using the key value using our RDD.
- On an RDD consisting of keys of type K and values of type V, we get back an RDD operation of type [K, Iterable[V]].
- A groupBy() works on unpaired data or data where we want to use a different terms of condition besides equality on the current key been specified.

- It requires a function that it allows to apply the same to every element in the source of RDD and uses the result to determine the key value obtained.

Joins

- The most useful and effective operations we get with keyed data values comes from using it together with other keyed data.
- Joining datasets together is probably one of the most common type of operations you can find out on a pair RDD.
- **Inner Join** : Only keys that are present in both pair RDDs are known as the output.
- **leftOuterJoin() :** The resulting pair RDD has entries for each key in the source RDD. The value which is been associated with each key in the result is a tuple of the value from the source RDD and an Option for the value from the other pair of RDD.
- **rightOuterJoin() :** is almost identical functioning to leftOuterJoin () except the key must be present in the other RDD and the tuple has an option for the source rather than the other RDD functions.

Sorting Data

- We can sort an RDD with key or value pairs if there is an ordering defined on the key set. Once we have sorted our data elements, any subsequent call on the sorted data to collect() or save() will result in ordered dataset.

Machine learning with MLib

- Apache Spark comes with a library named MLlib to perform machine learning tasks using spark framework.
- Since we have a Python API for Apache spark, that is, as you already know, PySpark, we can also use this library in PySpark.
- MLlib contains many algorithms and machine learning utilities.
- Machine learning is one of the many applications of Artificial intelligence (AI) where the primary aim is to enable the computers to learn automatically without any human assistance.
- With the help of machine learning, computers can tackle the tasks that were, until now, only handled and carried out by people.
- It's basically a process of teaching a system, how to make accurate predictions when fed the right data.
- It provides the ability to learn and improve from experience without being specifically programmed for that task.
- Machine learning mainly focuses on developing the computer programs and algorithms that make predictions and learn from the provided data.

What are dataframes?

- A dataframe is the new API for Apache Spark. It is basically a distributed, Strongly-typed collection of data, that is, a dataset which is organised into named columns. Dataframe is equivalent to what a table is for relational database, only, it has richer optimization options.

How to create dataframes

- There are multiple ways to create dataframes in Apache Spark:
    - Dataframes can be created using an existing RDD.
    - You can create a dataframe by loading a CSV file directly.
    - You can programmatically specify a schema to create a dataframe as well.

Basic of MLlib

- MLlib is short for machine learning library. Machine learning in PySpark is easy to use and scalable. It works on distributed systems.
- We use machine learning in PySpark for data analysis.
- We get the benefit of various machine learning algorithms such as Regression, classification etc, because of the MLlib in Apache Spark.

PySpark

Real-time Computation

- PySpark provides real-time computation on a large amount of data because it focuses on in-memory processing. It shows the low latency.

Support Multiple Language

- PySpark framework is suited with various programming languages like Scala, Java, Python, and R. Its compatibility makes it the preferable frameworks for processing huge datasets.

Caching and disk constancy

- PySpark framework provides powerful caching and good disk constancy.

Swift Processing

- PySpark allows us to achieve a high data processing speed, which is about 100 times faster in memory and 10 times faster on the disk.

Works well with RDD

- Python programming language is dynamically typed, which helps when working with RDD. We will learn more about RDD using Python in the further tutorial.

Parameters in PySpark MLlib

- Some of the main parameters of PySpark MLlib are listed below:
    - **Ratings**: This parameter is used to create an RDD of ratings, rows or tuples.
    - **Rank**: It shows the number of features computed and ranks them.

- **Lambda**: Lambda is a regularization parameter.
- **Blocks**: Blocks are used to parallel the number of computations. The default value for this is -1.

Performing Linear Regression on a real-world Dataset

- Let's understand machine learning better by implementing a full-fledged code to perform linear Regression on the dataset of top 5 Fortune 500 Companies in year 2017.

Loading the data:

- As mentioned above, we are going to use a dataframe that we have created directly from a CSV file.
- Following are the commands to load the data into a dataframe and to view the loaded data.

Input: In [1]:

```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SQLContext
Sc = SparkContext()
sqlContext = SQLContext(sc)
```

Input: In [2]:

```
company-df=
sqlContext.read.format('com.databricks.spark.csv').options(header='true',inferschema='true').load('
C:/Users/intellipaat/Downloads/spark-2.3.2-binhadoop2.7/Fortune5002017.csv')
company-df.take(1)
```

- You can choose the number of rows you want to view while displaying the data of a dataframe.
- I have displayed the first row only.

Output: Out[2]:

```
[Row (Rank=1, Title= 'Walmart', Website= 'http:/www.walmart.com', Employees-
2300000, Sector= 'retailing')]
```

Data exploration:

- To check the datatype of every column of a dataframe and print the schema of the dataframe in a tree format, you can use the following commands respectively.

Input: In[3]:

```
company-df.cache()
company-df.printSchema()
```

Output: Out [3]:

```
DataFrame[Rank: int, Title: string, Website: string, Employees: Int, Sector: string]
root
|– Rank: integer (nullable = true)
|– Title: string (nullable = true)
|– Website:string (nullable = true)
```

|– Employees: integer (nullable = true)
|– Sector: string (nullable = true)

Performing Descriptive Analysis:

Input: In [4]:

company-df.describe().toPandas().transpose()

Output: Out [4]:

	0	1	2	3	4
Summary	count	mean	stddev	min	max
Rank	5	3.0	1.581138830084	1	5
Title	5	None	None	Apple	Walmart
Website	5	None	None	www.apple.com	www.walmart.com
Employees	5	584880.0	966714.2168190142	68000	2300000
Sector	5	None	None	Energy	Wholesalers

Machine learning in Industry

- Computer systems with the ability to predict and learn from a given data and improve themselves without having to be reprogrammed used to be a dream only but in the recent years it has been made possible using machine learning.
- Now machine learning is a most used branch of artificial intelligence that is being accepted by big industries in order to benefit their businesses.
- Following are some of the organisations where machine learning has various use cases:
    - **PayPal**: PayPal uses machine learning to detect suspicious activity.
    - **IBM**: There is a machine learning technology patented by IBM which helps to decide when to handover the control of self-driving vehicle between a vehicle control processor and a human driver
    - **Google**: Machine learning is used to gather information from the users which further is used to improve their search engine results
    - **Walmart**: Machine learning in Walmart is used to benefit their efficiency
    - **Amazon**: Machine learning is used to design and implement personalised product recommendations
    - **Facebook**: Machine learning is used to filter out poor quality content.