SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRONICS

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

UNIT – I - OVERVIEW OF BIG DATA – SECA7018

I. OVERVIEW OF BIG DATA

Fundamentals of Big-data analytics, Overview & analytics life cycle, Need, Structured and multi-structured data analysis, Big- data analytics major components, Analytical models and approaches,

Fundamentals of Big Data Analytics:

Big Data Analytics is the process of transforming, inspecting and modeling the **data** with the aim of finding the applicable information. The phrase "big data" implies more than just storing more data. It also means doing more with data.

The various techniques into three big groups:

1. **Predictive analytics**, which are the class of algorithms that use data from the past to predict the future
2. **Collective intelligence**, which uses the inputs from large groups to create seemingly intelligent behavior
3. **Machine learning**, in which programs "learn from experience" and refine their algorithms-based on new information

These are clearly intersecting techniques - collective intelligence often is predictive, while predictive and collective techniques both involve machine learning.

Overview of Big Data Analytics:

Big Data Analytics largely involves collecting data from different sources, munge it in a way that it becomes available to be consumed by analysts and finally deliver data products useful to the organization business.

The process of converting large amounts of unstructured raw data, retrieved from different sources to a data product useful for organizations forms the core of Big Data Analytics.

Types of Big Data:

Big Data could be found in three forms:

1. **Structured** - Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data.
2. **Unstructured** - Any data with unknown form or the structure is classified as unstructured data.
3. **Semi-structured -** Semi-structured data can contain both the forms of data. We 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.

Characteristics of Big Data:

There are **Five V's** of Big Data **- Volume, Velocity, Variety, Veracity and Value**

***(i) Volume* –** Size of data plays a very crucial role in determining value out of data. A particular data can actually be considered as a Big Data or not, is dependent upon the volume of data. Hence, **'Volume'** is one characteristic which needs to be considered while dealing with Big Data.

***(ii) Variety* –** Variety refers to heterogeneous sources and the nature of data, both structured and unstructured. During earlier days, spreadsheets and databases were the only sources of data considered by most of the applications. Nowadays, data in the form of emails, photos, videos, monitoring devices, PDFs, audio, etc. are also being considered in the analysis applications. This variety of unstructured data poses certain issues for storage, mining and analyzing data.

***(iii) Velocity* –** The term **'velocity'** refers to the speed of generation of data. How fast the data is generated and processed to meet the demands, determines real potential in the data.

Big Data Velocity deals with the speed at which 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.

***(iv) Variability* –** This refers to the inconsistency which can be shown by the data at times, thus hampering the process of being able to handle and manage the data effectively.

Importance of Big Data Analytics:

It is an essential revolution in the sector of IT, and this technique is enlarging every year. It is the process of inspecting the huge data sets to emphasize both the patterns and insights.

Cost Reduction: The analytics technique like a Cloud Computing, Hadoop which it is important to cost benefits storing into the huge sets of information and data. In reality, they will recognize efficient ways of running the business.

Faster, Best Decision Making: Speed of Hadoop and that combination of able to analyzing the latest sources of information, business.

Services and Products: The ability to measure client satisfaction and needs through an analytics. They are so many companies are developing the new services and products to meet their client needs.

Real Time Benefits in Big Data Analytics:

It has been massive growth in this sector, and it led to the usability of big data in numerous industries ranging. For the purpose of, this tool helps Apache Hadoop to minimize the cost of storage.

1. Banking
2. Healthcare
3. Energy
4. Technology
5. Consumer
6. Manufacturing

Most of the banking sectors are using this big data technique. To put it another way, the education field may apply the big data concepts. As well as, a possibility for both the analysis and research utilize the data.

Advantages of Big Data Analytics:

- **Data procurement:** Particularly, it is a large amount of data for developing the store. They are several websites are accumulating into the data, secondary, and primary.

- **Data Integration and Data Quality:** The data and information may store in the high changes in data sets. The big data analytics are a lot of repetition which it is creates the expenses and confusion.
- **Data Segmentation:** It may use to distribute the data in various parameters for example location, age, gender, budget, product segmentation and so on.
- **Business Intelligence:** Fundamentals of Big Data Analytics is driven which it is consist the decision making, and it enables the scientists to visual data, aggregate, generate helping into the management decisions.
- **Prescriptive and Predictive Analytics:** It allows the various possible activity towards the solutions. In general, the mixture of historical data are found into the CRM, POS, ERP and HR systems may identify the patterns. Applying the algorithms and statistical models capturing the different datasets.

Big Data Analytics Life Cycle:

A big data analytics cycle can be described by the following stage:

- Business Problem Definition
- Research
- Human Resources Assessment
- Data Acquisition
- Data Munging
- Data Storage
- Exploratory Data Analysis
- Data Preparation for Modeling and Assessment
- Modeling
- Implementation

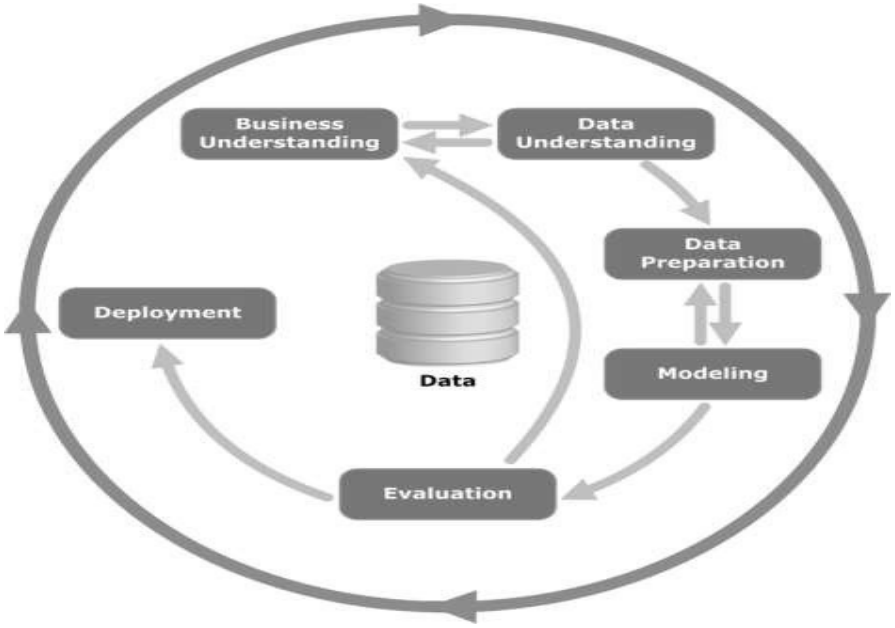


Fig.1 A big data analytics cycle

Business Problem Definition

This is a point common in traditional BI and big data analytics life cycle. Normally it is a non-trivial stage of a big data project to define the problem and evaluate correctly how much potential gain it may have for an organization. It seems obvious to mention this, but it has to be evaluated what are the expected gains and costs of the project.

Research

Analyze what other companies have done in the same situation. This involves looking for solutions that are reasonable for your company, even though it involves adapting other solutions to the resources and requirements that your company has. In this stage, a methodology for the future stages should be defined.

Human Resources Assessment

Once the problem is defined, it's reasonable to continue analyzing if the current staff is able to complete the project successfully. Traditional BI teams might not be capable to deliver an optimal solution to all the stages, so it should be considered before starting the project if there is a need to outsource a part of the project or hire more people.

Data Acquisition

This section is key in a big data life cycle; it defines which type of profiles would be needed to deliver the resultant data product. Data gathering is a non-trivial step of the process; it normally involves gathering unstructured data from different sources. To give an example, it could involve writing a crawler to retrieve reviews from a website. This involves dealing with text, perhaps in different languages normally requiring a significant amount of time to be completed.

Data Munging

Once the data is retrieved, for example, from the web, it needs to be stored in an easy to-use format. To continue with the reviews examples, let's assume the data is retrieved from different sites where each has a different display of the data.

Suppose one data source gives reviews in terms of rating in stars, therefore it is possible to read this as a mapping for the response variable $y \in \{1, 2, 3, 4, 5\}$. Another data source gives reviews using two arrows system, one for up voting and the other for down voting. This would imply a response variable of the form $y \in \{\text{positive, negative}\}$.

In order to combine both the data sources, a decision has to be made in order to make these two response representations equivalent. This can involve converting the first data source response representation to the second form, considering one star as negative and five stars as positive. This process often requires a large time allocation to be delivered with good quality.

Data Storage

Once the data is processed, it sometimes needs to be stored in a database. Big data technologies offer plenty of alternatives regarding this point. The most common alternative is using the Hadoop File System for storage that provides users a limited version of SQL, known as HIVE Query Language. This allows most analytics task to be done in similar ways as would be done in traditional BI data warehouses, from the user perspective. Other storage options to be considered are MongoDB, Redis, and SPARK.

This stage of the cycle is related to the human resources knowledge in terms of their abilities to implement different architectures. Modified versions of traditional data warehouses are still being used in large scale applications. For example, teradata and IBM offer SQL databases that can handle terabytes of data; open source solutions such as PostgreSQL and MySQL are still being used for large scale applications.

Even though there are differences in how the different storages work in the background, from the client side, most solutions provide a SQL API. Hence having a good understanding of SQL is still a key skill to have for big data analytics.

This stage priori seems to be the most important topic, in practice, this is not true. It is not even an essential stage. It is possible to implement a big data solution that would be working with real-time data, so in this case, we only need to gather data to develop the model and then implement it in real time. So there would not be a need to formally store the data at all.

Exploratory Data Analysis

Once the data has been cleaned and stored in a way that insights can be retrieved from it, the data exploration phase is mandatory. The objective of this stage is to understand the data, this is normally done with statistical techniques and also plotting the data. This is a good stage to evaluate whether the problem definition makes sense or is feasible.

Data Preparation for Modeling and Assessment

This stage involves reshaping the cleaned data retrieved previously and using statistical preprocessing for missing values imputation, outlier detection, normalization, feature extraction and feature selection.

Modelling

The prior stage should have produced several datasets for training and testing, for example, a predictive model. This stage involves trying different models and looking forward to solving the business problem at hand. In practice, it is normally desired that the model would give some insight into the business. Finally, the best model or combination of models is selected evaluating its performance on a left-out dataset.

Implementation

In this stage, the data product developed is implemented in the data pipeline of the company. This involves setting up a validation scheme while the data product is working, in order to track its performance. For example, in the case of implementing a predictive model, this stage would involve applying the model to new data and once the response is available, evaluate the model.

Need for Big Data Analytics:

- Big data analytics efficiently helps operations to become more effective. This helps in improving the profits of the company.
- Big data analytics tools like Hadoop helps in reducing the cost of storage. This further increases the efficiency of the business.
- Improved customer service and better operational efficiency.
- Big data dramatically increases both the number of data sources and the variety and volume of data that is useful for analysis. In most organizations, multi-structured data is growing at a considerably faster rate than structured data.
- Early identification of risk to the product/services. Businesses can utilize outside intelligence while taking decisions.

Structured and Multi structured Data Analysis:

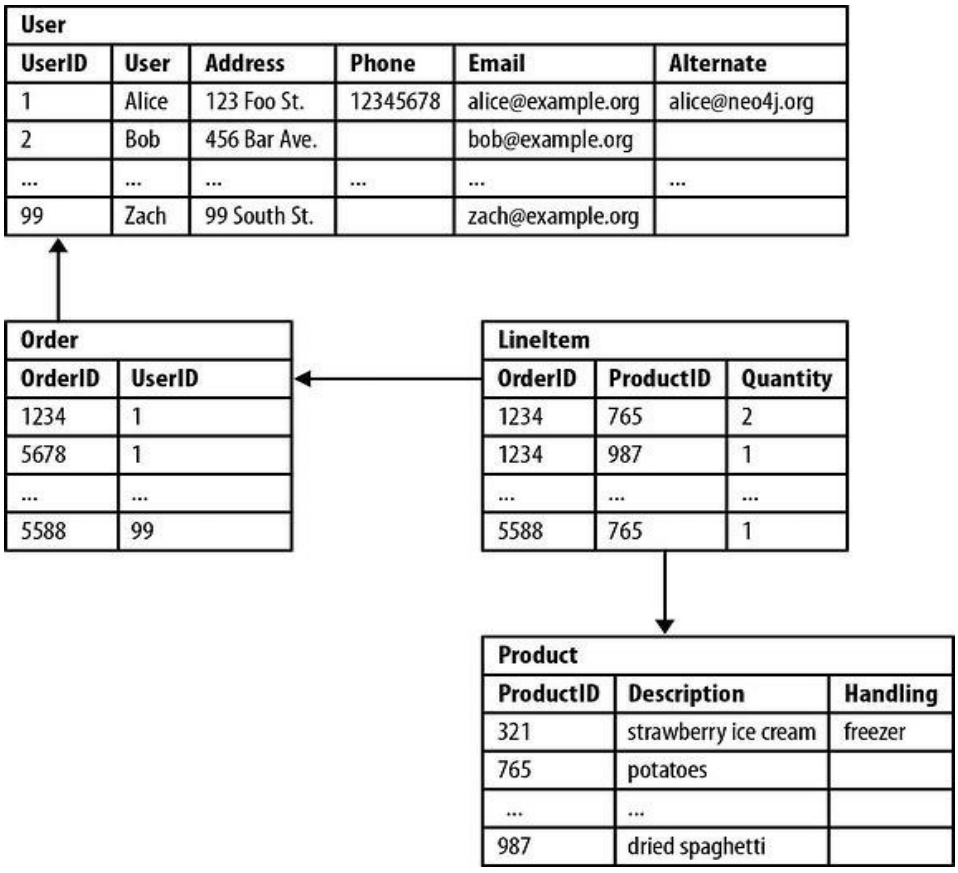
Structured Data Analysis:

The term *structured data* generally refers to data that has a defined length and format for big data. Examples of structured data include numbers, dates, and groups of words and numbers called strings.

Structured data is highly-organized and formatted in a way so it's easily searchable in relational databases and understood by machine language. Those working within relational databases can input, search, and manipulate structured data relatively quickly. This is the most attractive feature of structured data.

The programming language used for managing structured data is called structured query language, also known as SQL. This language was developed by IBM in the early 1970s and is particularly useful for handling relationships in databases.

Examples of **structured data** include names, dates, addresses, credit card numbers, stock information, geolocation etc. The picture below should help visualize how structured data relates to each other within a database.



User

UserID	User	Address	Phone	Email	Alternate
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	...
99	Zach	99 South St.		zach@example.org	

Order

OrderID	UserID
1234	1
5678	1
...	...
5588	99

LineItem

OrderID	ProductID	Quantity
1234	765	2
1234	987	1
...	...	...
5588	765	1

Product

ProductID	Description	Handling
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

Fig.2 Example of structured data

- From the top-down, we can see that **UserID 1** refers to the customer Alice, who had two **OrderIDs** of '1234' and '5678'.
- Next, Alice had two **ProductIDs** of '765' and '987'. Finally, we can see Alice purchased two packages of potatoes and one package of dried spaghetti.

Sources of Structured Big Data:

The sources of data are divided into two categories:

1. **Computer- or machine-generated:** Machine-generated data generally refers to data that is created by a machine without human intervention.

2. **Human-generated:** This is data that humans, in interaction with computers, supply.

Machine-generated structured data can include the following:

- Sensor Data
- Web Log Data
- Point of Sale Data
- Financial Data
- Input Data

- Click stream Data
- Gaming Related Data

Multi-Structured Data Analysis:

Multi-structured data refers to a variety of **data** formats and types and can be derived from interactions between people and machines, such as web applications or social networks.

Two significant breakthroughs are quickly evolving the capabilities of multi-structured data:

1. Less expensive, centralized storage and processing of multi-structured data in Hadoop.
2. Exponentially more powerful analytics applications that improve analyst productivity.
   Typical human-generated multi-structured 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 multi-structured data includes:

- **Satellite imagery:** Weather data, land forms, 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.

Difference between Structured and Unstructured Data:

	Structured Data	Unstructured Data
Characteristics	• Pre-defined data models • Usually text only • Easy to search	• No pre-defined data model • May be text, images, sound, video or other formats • Difficult to search
Resides in	• Relational databases • Data warehouses	• Applications • NoSQL databases • Data warehouses • Data lakes
Generated by	Humans or machines	Humans or machines
Typical applications	• Airline reservation systems • Inventory control • CRM systems • ERP systems	• Word processing • Presentation software • Email clients • Tools for viewing or editing media
Examples	• Dates • Phone numbers • Social security numbers • Credit card numbers • Customer names • Addresses • Product names and numbers • Transaction information	• Text files • Reports • Email messages • Audio files • Video files • Images • Surveillance imagery

Fig.3 Difference between Structured and Unstructured Data

BIG DATA ANALYTICS MAJOR COMPONENTS:

The main components of Big Data are as follows:

- Machine Learning
- Natural Language processing
- Business Intelligence
- Cloud Computing

1. **Machine Learning**

It is the science of making computers learn stuff by themselves. In machine learning, a computer is expected to use algorithms and statistical models to perform specific tasks without any explicit instructions. Machine learning applications provide results based on past experience. For example, these days there are some mobile applications that will give you a summary of your finances, bills, will remind you on your bill payments, and also may give you suggestions to go for some saving plans. These functions are done by reading your emails and text messages.

2. Natural Language processing

It is the ability of a computer to understand human language as spoken. The most obvious examples that people can relate to these days is google home and Amazon Alexa. Both use NLP and other technologies to give us a virtual assistant experience. NLP is all around us without us even realizing it. When writing a mail, while making any mistakes, it automatically corrects itself and these days it gives auto-suggests for completing the mails and automatically intimidates us when we try to send an email without the attachment that we referenced in the text of the email, this is part of Natural Language Processing Applications which are running at the backend.

3. Business Intelligence

Business Intelligence (BI) is a method or process that is technology-driven to gain insights by analyzing data and presenting it in a way that the end-users (usually high-level executives) like managers and corporate leaders can gain some actionable insights from it and make informed business decisions on it.

4. Cloud Computing

If we go by the name, it should be computing done on clouds, well, it is true, just here we are not talking about real clouds, Cloud here is a reference for the Internet. So we can define cloud computing as the delivery of computing services—servers, storage, databases, networking, software, analytics, intelligence and more over the Internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale.

ANALYTICAL MODELS AND APPROACHES:

MODELS:

The four different types of analytics:

- Descriptive Analytics
- Diagnostic Analytics
- Predictive Analytics
- Prescriptive Analytics

We need to understand what each **type of analytics** delivers to improve on, an organization's operational capabilities. Below, we will introduce each type and give examples of how they are utilized in business.

Descriptive Analytics

It is the simplest and most common use of data in business today. Descriptive analysis answers the "what happened" by summarizing past data, usually in the form of dashboards. The biggest use of descriptive analysis in business is to track Key Performance Indicators (KPIs). KPIs describe how a business is performing based on chosen benchmarks.

Business applications of descriptive analysis include:

- KPI dashboards
- Monthly revenue reports
- Sales leads overview

Diagnostic Analytics

After asking the main question of "what happened", the next step is to dive deeper and ask why did it happen? This is where diagnostic analysis comes in. Diagnostic analysis takes the insights found from descriptive analytics and drills down to find the causes of those outcomes. Organizations make use of this type of analytics as it creates more connections between data and identifies patterns of behavior. A critical aspect of diagnostic analysis is creating detailed information. When new problems arise, it is possible you have already collected certain data pertaining to the issue. By already having the data at your disposal, it ends having to repeat work and makes all problems interconnected.

Business applications of diagnostic analysis include:

- A freight company investigating the cause of slow shipments in a certain region
- A SaaS company drilling down to determine which marketing activities increased trials

Predictive Analytics

Predictive analysis attempts to answer the question "what is likely to happen". This type of analytics utilizes previous data to make predictions about future outcomes. This type of analysis is

another step up from the descriptive and diagnostic analyses. Predictive analysis uses the data we have summarized to make logical predictions of the outcomes of events. This analysis relies on statistical modeling, which requires added technology and manpower to forecast. It is also important to understand that forecasting is only an estimate; the accuracy of predictions relies on quality and detailed data. While descriptive and diagnostic analysis are common practices in business, predictive analysis is where many organizations begin show signs of difficulty. Some companies do not have the manpower to implement predictive analysis in every place they desire. Others are not yet willing to invest in analysis teams across every department or not prepared to educate current teams.

Business applications of predictive analysis include:

- Risk Assessment
- Sales Forecasting
- Using customer segmentation to determine which leads have the best chance of converting
- Predictive analytics in customer success teams

Prescriptive Analysis

The final type of data analysis is the most sought after, but few organizations are truly equipped to perform it. Prescriptive analysis is the frontier of data analysis, combining the insight from all previous analyses to determine the course of action to take in a current problem or decision. Prescriptive analysis utilizes state of the art technology and data practices. It is a huge organizational commitment and companies must be sure that they are ready and willing to put forth the effort and resources.

Artificial Intelligence (AI) is a perfect example of prescriptive analytics. AI systems consume a large amount of data to continuously learn and use this information to make informed decisions. Well-designed AI systems are capable of communicating these decisions and even putting those decisions into action. Business processes can be performed and optimized daily without a human doing anything with artificial intelligence. Currently, most of the big data-driven companies (Apple, Facebook, Netflix, etc.) are utilizing prescriptive analytics and AI to improve decision making.

The models which are explained below are trend in Data analytics:

1. PREDICTIVE CHURN PREVENTION MODELS

Retain Your Customer Base

Leverage your ERP data to build models of customer behavior that can identify who is likely to switch to a competitor and why. These valuable models can be used to prevent customer churn and help implement highly effective retention campaigns to save your enterprise substantial revenue.

2. PREDICTIVE CUSTOMER LIFETIME VALUE MODELS

Identify Your High-Value Customers

Put your big data to work and calculate customer lifetime value. Pinpoint individuals with a propensity to invest more in your products and services so that relationships can be cultivated and nurtured to ensure a continuous revenue stream.

3. CUSTOMER SEGMENTATION MODELS

Refine Your Messaging

Group customers based on similar characteristics and buying behaviors in order to align your company's marketing strategy and develop targeted outreach programs to these groups. Your big data mining could also uncover new insights that alter your marketing tactics!

4. ADAPTIVE & PREDICTIVE NEXT-BEST-ACTION MODELS

Get to Know Your Customers

These predictive analytics models foresee the next best action by observing, learning and responding to life-event patterns, purchasing behaviors, social media interactions, and additional aspects. This allows your company to determine which customers need to be approached and the best channel to contact them.

5. PREDICTIVE MAINTENANCE MODELS

Don't Fall Prey to Unforeseen Expenses

Predictive maintenance models can help forecast previously unpredictable machine breakdown, thereby helping companies to calculate and improve maintenance planning, leading to decreases in costly downtime of critical equipment.

6. PREDICTIVE PRODUCT PROPENSITY MODELS

Know What Customers Will Purchase Before They Do

Integrate your customer's online behavior from social networks like Facebook, Twitter and Instagram with their historical purchasing data to identify and understand factors that will influence future purchasing decisions. Models can be used to identify which products a customer is likely to buy and automatically provide recommendations, thereby increasing sales and driving revenue growth.

7. QUALITY ASSURANCE MODELS

Evoke Confidence in Your Products

Quality assurance models prevent defects in your products and avoid headaches when delivering solutions to your customers. Use historical data to detect and solve problems in production and ensure that equipment, machinery and processes are delivering proper output and quality. These models will provide you with the peace of mind and keep your quality management on point.

8. PREDICTIVE RISK MODELS

Identify and Mitigate Your Risks

Banking, insurance and telecommunications organizations are capable of mining big data with models designed to deliver faster insights into fraud and score liabilities. These models are designed to help organizations spot and abate risk exposure. Industry recommends standard process called CRISP-DM, which is an acronym for Cross Industry Practices for Data Mining.

9. SENTIMENT ANALYSIS MODELS

Protect Your Reputation

Sentiment analysis, or "opinion mining" models identify, extract and categorize information from publicly available data sources, such as online reviews, blogs and social media posts. Their purpose is to analyze and determine sentiments towards an organization and its products and services.

Assess the polarity of product reviews and discussions around the web and quickly adopt strategies designed to counter negative opinions and enhance positive sentiment.

10. PREDICTIVE UPSELL & CROSS-SELL MODELS

Sell More, Sell Smarter

Alleviate the depletion of resources and increase selling power to support year-over-year growth. Predictive upsell and cross-sell models combine buying behaviors and market basket analyses to reveal insights into which products and services customers have the propensity to purchase and actively cross-sell and upsell them.

BIG DATA ANALYTICS APPROACHES:

Big data is a field that treats ways to analyze, systematically extract information from, or otherwise deal with **data** sets that are too large or complex to be dealt with by traditional **data**-processing application software. **Big data** was originally associated with three key concepts: volume, variety, and velocity.

- **Volume** refers to the vast amounts of data generated every second. It is not about Terabytes but Zettabytes or Brontobytes. If we take all the data generated in the world between the beginning of time, the same amount of data will soon be generated every minute. This makes most data sets too large to store and analyse using traditional database technology. New big data tools use distributed systems so that we can store and analyse data across databases that are dotted around anywhere in the world.

- **Velocity** refers to the speed at which new data is generated and the speed at which data moves around. Think of social media messages going viral in seconds. Technology allows us now to analyse the data while it is being generated (sometimes referred to as in-memory analytics), without ever putting it into databases.

- **Variety** refers to the different types of data . In the past we only focused on structured data that neatly fitted into tables or relational databases, such as financial data. In fact, 80% of the world's data is unstructured (text, images, video, voice, etc.) With big data technology we can now analyse and bring together data of different types such as messages, social media conversations, photos, sensor data, video or voice recordings.

- **Veracity** refers to the messiness or trustworthiness of the data. With many forms of big data quality and accuracy are less controllable (just think of Twitter posts with hash tags, abbreviations, typos and colloquial speech as well as the reliability and accuracy of content) but technology now allows us to work with this type of data.

- **Value**- Having access to big data is no good unless we can turn it into value. Companies are starting to generate amazing value from their big data.

STEPS FOR DATA ANALYSIS:

● Pose a problem

● Collect data – raw and dirty data

● Pre-process data (like extract feature) – clean data

● Design mathematical model (formulation)

● Find a solution

● Evaluation

BIG DATA CHALLENGES:

1. Insufficient Understanding and acceptance of big data.
2. Confusing variety of big data technologies.
3. Paying loads of money.
4. Complexity of managing data quality.
5. Dangerous Big data security holes.
6. Tricky process of converting big data into valuable insights.
7. Troubles of upscaling.
8. Dealing with data growth.
9. Generating insights in a timely manner.
10. Recruiting and retaining big data talent.
11. Integrating disparate data sources.

12. Validating data.
13. Securing big data.
14. Organizational resistance.

Advantages of Big Data

Following are the **advantages of Big Data**:

- Big data analysis derives innovative solutions. Big data analysis helps in understanding and targeting customers. It helps in optimizing business processes.
- It helps in improving science and research.
- It improves healthcare and public health with availability of record of patients.
- It helps in financial trading, sports, polling, security/law enforcement etc.
- Anyone can access vast information via surveys and deliver answer of any query.
- Every second additions are made.
- One platform carry unlimited information.

Disadvantages of Big Data:

Following are the **disadvantages of Big Data**:

- Traditional storage can cost lot of money to store big data.
- Lots of big data is unstructured.
- Big data analysis violates principles of privacy.
- It can be used for manipulation of customer records.
- It may increase social stratification.
- Big data analysis is not useful in short run. It needs to be analyzed for longer duration to leverage its benefits.
- Big data analysis results are misleading sometimes.
- Speedy updates in big data can mismatch real figures.
-

- **PART A**
-

Q.No.	QUESTION
1	Interpret big data is different from normal data with example
2	Identify the need of big data solution?
3	Identify a big data architecture suitable for bank data?
4	Demonstrate how big data solution work?

5	Identify any five big data application in the real world.

-

- **PART B**

Q.No.	QUESTION
1	Customize a Big Data Architecture for identifying no of tweets for one hashtag in a day?
2	Discover the need of big data and how is big data architecture designed?
3	Attain the reason for the need of big data solution.
4	Classify is different than your Big data?
5	Determine how the big data can be efficiently utilized in various sectors?

UNIT – II - ARCHITECTURES IN BIG DATA – SECA7018

Designing and building big data applications, Big data architecture, Distributed Computing platforms and Data Storage, Security and Data Privacy, Application Areas, Application Tools and Platforms.

Designing and building big data applications:

Design Principles for Big Data Performance:

The main principles to keep in mind when you design and implement a data-intensive process of large data volume, which could be a data preparation for your machine learning applications, or pulling data from multiple sources and generating reports or dashboards for customers.

The goal of performance optimization is to either reduce resource usage or make it more efficient to fully utilize the available resources, so that it takes less time to read, write, or process the data. The ultimate objectives of any optimization should include:

1. Maximized usage of memory that is available
2. Reduced disk I/O
3. Minimized data transfer over the network
4. Parallel processing to fully leverage multi-processors

Therefore, when working on big data performance, a good architect is not only a programmer, but also possess good knowledge of server architecture and database systems.

Principle 1. Design based on your data volume

Before you start to build any data processes, you need to know the data volume you are working with: what will be the data volume to start with, and what the data volume will be growing into. If the data size is always small, design and implementation can be much more straightforward and faster. If the data start with being large, or start with being small but will grow fast, the design needs to take performance optimization into consideration. On the other hand, an application designed for small data would take too long for big data to complete. In other words, an application or process should be designed differently for small data vs. big data. Below lists the reasons in detail:

1. Because it is time-consuming to process large datasets from end to end, more breakdowns and checkpoints are required in the middle. The goal is 2-folds: first to allow one to check the immediate results or raise the exception earlier in the process, before the whole process ends; second, in the case that a job fails, to allow restarting from the last successful checkpoint, avoiding re-starting from the beginning which is more expensive. For small data, on the contrary, it is usually more efficient to execute all steps in 1 shot because of its short running time.

2. When working with small data, the impact of any inefficiencies in the process also tends to be small, but the same inefficiencies could become a major resource issue for large data sets.

3. Paralleling processing and data partitioning (see below) not only require extra design and development time to implement but also takes more resources during running time, which, therefore, should be skipped for small data.

4. When working with large data, performance testing should be included in the unit testing; this is usually not a concern for small data.

5. Processing for small data can complete fast with the available hardware, while the same process can fail when processing a large amount of data due to running out of memory or disk space.

The bottom line is that the same process design cannot be used for both small data and large data processing. Large data processing requires a different mindset, prior experience of working with large data volume, and additional effort in the initial design, implementation, and testing. On the other hand, do not assume "one-size-fit-all" for the processes designed for the big data, which could hurt the performance of small data.

Principle 2: Reduce data volume earlier in the process.

When working with large data sets, reducing the data size early in the process is always the most effective way to achieve good performance. There is no silver bullet to solving the big data issue no matter how much resources and hardware you put in. So always try to reduce the data size before starting the real work. There are many ways to achieve this, depending on different use cases. Below lists some common techniques, among many others:

1. Do not take storage (e.g., space or fixed-length field) when a field has NULL value.
2. Choose the data type economically. For example, if a number is never negative, use integer type, but not unsigned integer; If there is no decimal, do not use float.
3. Code text data with unique identifiers in integer, because the text field can take much more space and should be avoided in processing.
4. Data aggregation is always an effective method to reduce data volume when the lower granularity of the data is not needed.
5. Compress data whenever possible.
6. Reduce the number of fields: read and carry over only those fields that are truly needed.
7. Leverage complex data structures to reduce data duplication. One example is to use the array structure to store a field in the same record instead of having each on a separate record when the field shares many other common key fields.

The above list gives some ideas as to how to reduce the data volume. In fact, the same techniques have been used in many database softwares and IoT edge computing. The better you understand the data and business logic, the more creative you can be when trying to reduce the size of the data before working with it. The end result would work much more efficiently with the available memory, disk and processors.

Principle 3: Partition the data properly based on processing logic

Enabling data parallelism is the most effective way of fast data processing. As the data volume grows, the number of parallel processes grows, hence, adding more hardware will scale the overall data process without the need to change the code. For data engineers, a common method is data partitioning. There are many details regarding data partitioning techniques, which is beyond the scope of this article. Generally speaking, an effective partitioning should lead to the following results:

1. Allow the downstream data processing steps, such as join and aggregation, to happen in the same partition. For example, partitioning by time periods is usually a good idea if the data processing logic is self-contained within a month.

2. The size of each partition should be even, in order to ensure the same amount of time taken to process each partition.

3. As the data volume grows, the number of partitions should increase, while the processing programs and logic stay the same.

Also, changing the partition strategy at different stages of processing should be considered to improve performance, depending on the operations that need to be done against the data. For example, when processing user data, the hash partition of the User ID is an effective way of partitioning. Then when processing users' transactions, partitioning by time periods, such as month or week, can make the aggregation process a lot faster and more scalable.

Hadoop and Spark store the data into data blocks as the default operation, which enables parallel processing natively without needing programmers to manage themselves. However, because their framework is very generic in that it treats all the data blocks in the same way, it prevents finer controls that an experienced data engineer could do in his or her own program. Therefore, knowing the principles stated in this article will help you optimize process performance based on what's available and what tools or software you are using.

Principle 4: Avoid unnecessary resource-expensive processing steps whenever possible

As stated in Principle 1, designing a process for big data is very different from designing for small data. An important aspect of designing is to avoid unnecessary resource-expensive operations whenever possible. This requires highly skilled data engineers with not just a good understanding of how the software works with the operating system and the available hardware resources, but also comprehensive knowledge of the data and business use cases. In this article, I only focus on the top two processes that we should avoid to make a data process more efficient: data sorting and disk I/O.

Putting the data records in a certain order is often needed when 1) joining with another dataset; 2) aggregation; 3) scan; 4) deduplication, among other things. However, sorting is one of the most

expensive operations that require memory and processors, as well as disks when the input dataset is much larger than the memory available. To get good performance, it is important to be very frugal about sorting, with the following principles:

1. Do not sort again if the data is already sorted in the upstream or the source system.
2. Usually, a join of two datasets requires both datasets to be sorted and then merged. When joining a large dataset with a small dataset, change the small dataset to a hash lookup. This allows one to avoid sorting the large dataset.
3. Sort only after the data size has been reduced (Principle 2) and within a partition (Principle 3).
4. Design the process such that the steps requiring the same sort are together in one place to avoid re-sorting.
5. Use the best sorting algorithm (e.g., merge sort or quick sort).

Another commonly considered factor is to reduce the disk I/O. There are many techniques in this area, which is beyond the scope of this article. Below lists 3 common reasons that need to be considered in this aspect:

1. Data compression
2. Data indexing
3. Performing multiple processing steps in memory before writing to disk

Data compression is a must when working with big data, for which it allows faster read and write, as well as faster network transfer. Data file indexing is needed for fast data accessing but at the expense of making writing to disk longer. Index a table or file only when it is necessary while keeping in mind its impact on the writing performance. Lastly, perform multiple processing steps in memory whenever possible before writing the output to disk. This technique is not only used in Spark but also used in many database technologies.

BUILDING BIG DATA APPLICATIONS

Software applications have traditionally been perceived as a unit of computation designed and used to solve a problem. Whether an application is a CRM tool that helps manage customer information or a complex supply-chain management system. Applications are also frequently designed with a relatively static set of input and output interfaces, and communication to and from the application uses specially designed (or chosen) protocols

Applications are also designed around data. The data that an application uses to solve a problem is stored using a data platform. This underlying data platform has historically been designed to enable optimal data storage and retrieval. Somewhere in the process of storage and retrieval of data, an application applies computation is to produce results in the application.

Define objectives with an open mind

Define the tangible results of the application. Will the application merely provide the ability to join datasets across technologies? Find high-value customers earlier than existing systems? Regardless of the defined goal, keep in mind that joining data may yield insights beyond the scope of the defined goal. Don't throw away data because you may find nuggets of gold in unexpected places.

Understand volumes, sources and integration points

Define the existing applications you want to stream data to and from. Document the volumes of data and how frequently the data changes. Involve the teams that manage those datasets early on to understand the best method(s) of exchanging data from various sources. Missing key customer information can make a world of difference.

Determine the platform

Understanding the types and volume of data your Big Data Application will operate on is going to dictate the platform required. Traditionally, Hadoop is the de-facto platform since it allows for all forms of ingest and analysis, ranging from real-time, to batch, to low-latency analytics. Choose the platform that is the most versatile and fits your needs. Partner with a vendor who can support you throughout your applications' lifespan.

Start with batch; graduate slowly

Big data platforms today offer a variety of ways to analyze your data whether it be batch, real-time, or in-stream processing. Begin initially with offline, or batch processing. This allows you to process and analyze your data in a manner unobtrusive to existing applications. As phases of the

application mature, transition to more real-time integration points between your big data application and existing applications.

Create a process for data collection

Batch loading and processing is the right way to start a Big Data Application project. Once you perform your initial analysis, work on creating a process that allows for incremental refresh of data. You should be able to copy just the updated changes from your source systems into your big data platform. A continuous refresh of business data, that can provide up-to-date analysis.

Create a 360-degree feedback loop

Once a process for data collection and updates has been established, begin creating datasets that can feed the existing applications and systems to make them smarter. Establish a 360-degree process that will schedule data and feed it into your big data application for analysis. Create incremental and smaller datasets to be consumed back by the originating application. Using this method, all the existing applications benefit from having the intelligence, analysis and up to date information from systems that they need not know exist, or were unable to benefit from previously.

Use data mining and predictive analytics

You're well on your way to a big data application once you create a large feedback loop that now bridges all the various data sources within the business. Go further by doing deep data mining and behavioral analysis on your data to help predict where best to optimize resources. As the Big Data technologies mature, newer higher level open source frameworks, such as Kiji, can lower the barrier to entry for advanced machine learning and deep data mining analytics.

Evaluate real-time, predictive models

Given previous data mining, or other analytics insights, augment your applications further by providing a real-time interface that can dynamically rescore your predictive models based on up to the second changes to data. Use knowledge learned from batch/iterative research and analysis to build better predictive models that can be executed in real-time, on a per-user level. Open source frameworks such as Kiji resT and Kiji Scoring allow this to be done nearly at the speed of thought. Being able to iterate quickly and deploy new models as trends shift allow you to capitalize on trends as they occur instead of lagging customer trends by weeks or months.

Continually refine the process

A successful big data application requires continuous evaluation and refinement. Is the right data being collected? Are there new data sources that need to be integrated? Are there any data

sources that should removed or are stale? Are there better predictive models we can use to provide a more accurate real-time experience? These are questions that should be continually evaluated so that the quality of data streaming into your big data application remains constantly relevant to the business.

Measure results and reap the rewards

The way to know the impact and performance of your big data application is by measuring the results. It might be as simple as knowing how much data is being collected daily verse the number of records being sent back to existing traditional applications or implementing advanced A/B testing on behavioral models. The best way to know whether your big data application is achieving its objectives is to measure the results against expectations.

BIG DATA ARCHITECTURE:

Big data architecture is the overarching system used to ingest and process enormous amounts of data (often referred to as "big data") so that it can be analyzed for business purposes. The architecture can be considered the blueprint for a big data solution based on the business needs of an organization. Big data architecture is designed to handle the following types of work:

- Batch processing of big data sources.
- Real-time processing of big data.
- Predictive analytics and machine learning.

The Big Data Architecture

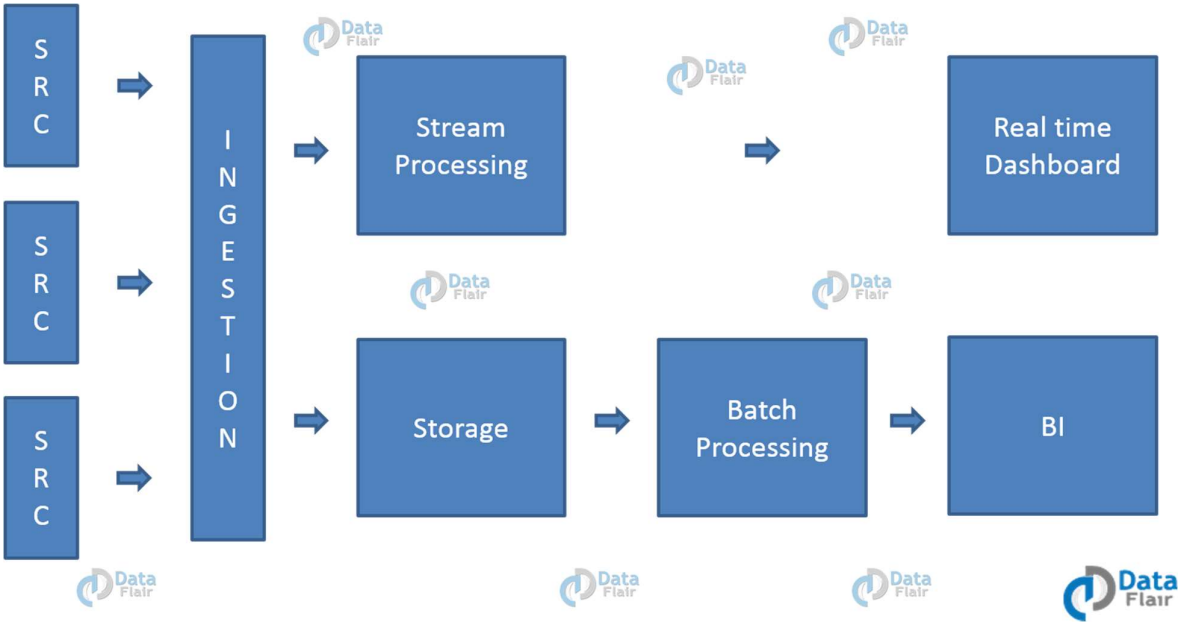


Fig .4. Big Data Architecture

- ***Sources Layer***

  The Big Data sources are the ones that govern the Big Data architecture. The designing of the architecture depends heavily on the data sources. The data is arriving from numerous sources that too in different formats. These include relational databases, company servers and sensors such as IoT devices, third-party data providers, etc. This data can be both batch data as well as real-time data. These sources pile up a huge amount of data in no time. The Big Data architecture is designed such that it is capable of handling this data.

- ***Data Ingestion***

  This is the first layer from which the journey of Big Data arriving from numerous sources begins. This layer takes care of categorizing the data for the smooth flow of data into the further layers of the architecture. The primary goal of this layer is to furnish trouble-free transportation of data into the further layers of data architecture. Generally, Kafka Streams or REST APIs are used for Ingestion.

- ***Storage Layer***

  This layer is at the receiving end for the Big Data. It receives data from the various data sources and stores it in the most appropriate manner. This layer can even change the format of the data according to the requirements of the system. For example, batch processing data is generally stored in a distributed file storage system such as HDFS that are capable of storing high volume

data that too in different formats. On the other hand, structured data can be stored using RDBMS only. It all depends on the format of the data and the purpose we need it for.

- ***Analysis Layer***

  The only goal of companies employing Big Data is to gain insights from it and thus make data-driven decisions. To empower users to analyze Big Data, the most important layer in the Big Data architecture is the analysis layer. This analysis layer interacts with the storage layer to gain valuable insights. The architecture requires multiple tools to analyze Big Data. The structured data is easy to handle whereas some advanced tools are needed to analyze the unstructured data.

1. **Batch Processing**

   Since the data is so huge in size, the architecture needs a batch processing system to filter, aggregate, and process data for advanced analytics. These are long-running batch jobs. This involves reading the data from the storage layer, processing it, and finally writing the outputs to the new files. ***Hadoop*** is the most commonly used solution for it.

2. **Real-Time Processing**

   Processing the data arriving in real-time is the hottest trend in the Big Data world. The Big Data architecture, therefore, must include a system to capture and store real-time data. This can be done by simply ingesting the real-time data into a data store for processing. The architecture needs to have a robust system for dealing with real-time data.

- ***BI Layer***

  This layer receives the final analysis output and replicates it to the appropriate output system. The different types of outputs are for human viewers, applications, and business processes. The whole process of gaining Big Data solutions includes ingesting data from multiple sources, repeated data processing operations, and drawing the results into a report or a dashboard. These reports are then used for making data-driven decisions by the companies.

Benefits of Big Data Architecture:

- **Reducing costs**. Big data technologies such as Hadoop and cloud-based analytics can significantly reduce costs when it comes to storing large amounts of data.
- **Making faster, better decisions**. Using the streaming component of big data architecture, you can make decisions in real-time.
- **Predicting future needs and creating new products**. Big data can help you to gauge customer needs and predict future trends using analytics.

Challenges

Data Quality, Scaling, Security. Big Data technologies are evolving new changes that help in building optimized systems. While the Haddop technologies such as Hive and Pig have stabilized, emerging technologies such as Spark are continuously introducing extensive changes and enhancements with each new release. To choose the right technology according to your business requirements is the key to Big Data architecture.

Implementing Big Data architecture brings a lot of security challenges. The insights depend on centrally stored static data. But accessing this data is a challenging task as the data could be ingested and consumed by multiple applications and platforms. In the era where data breaching is commonplace, implementing a robust security system becomes a necessity to safeguard the data from various thefts. A service-level agreement must be signed with the service provider at the beginning itself to ensure the safety of your data.

Distributed computing platforms and data storage and Distributed Computing:

Distribute Computing is a field of computer science studying distributed system. A **distributed system** consists of multiple nodes that communicate via network. The computers in each node interact each other to accomplish a shared goal. In other words, distributed computing uses distributed systems to solve problems.

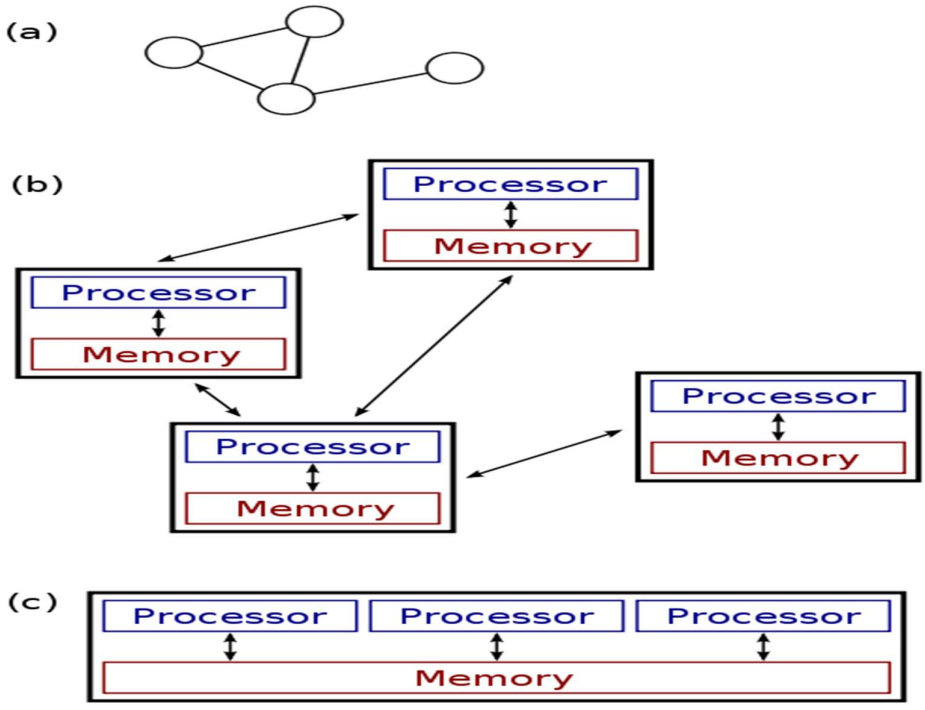


Fig 5. A distributed system

Three-tier Architecture

Three-tier architecture is a client-server architecture. The presentation, the application processing, and the data management are logically separate processes.

For example, an application that uses middleware to service data requests between a user and a database employs multi-tier architecture. The most widespread use of multi-tier architecture is the three-tier architecture.

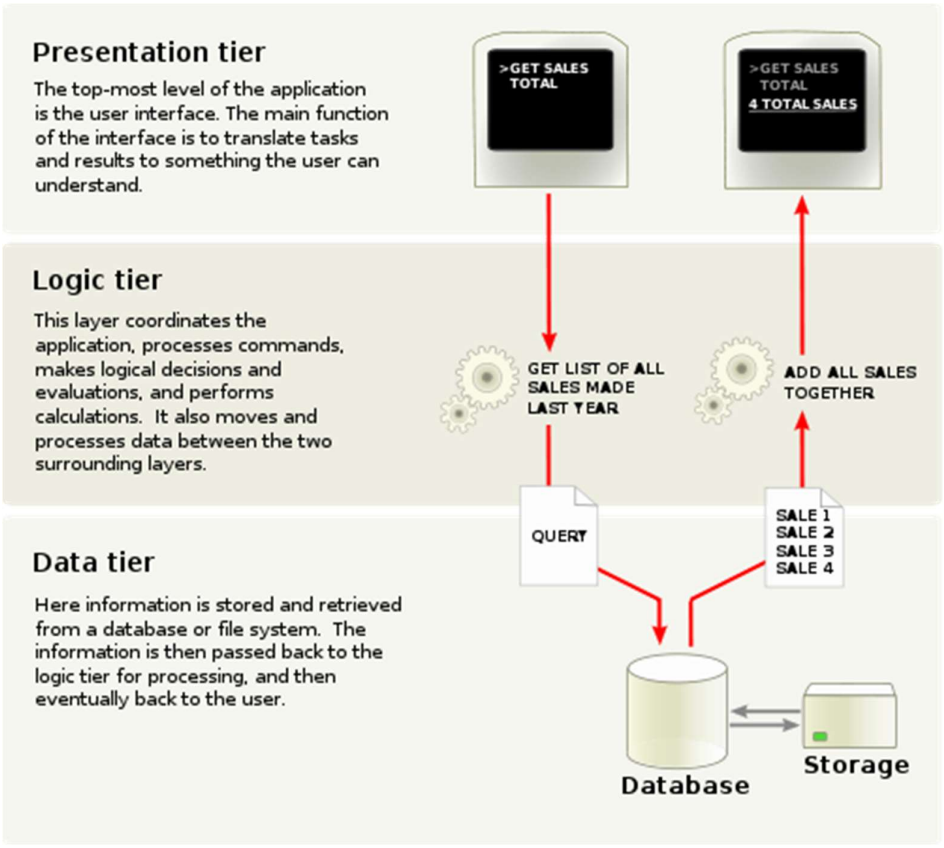


Fig 6. Three-tier architecture

EXAMPLE:

Hadoop is a framework for running applications on large cluster built of commodity hardware. It has two main components:

1. **Map/Reduce**

   It is a computational paradigm, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster.

2. **Hadoop distributed file system (HDFS)**

   It stores data on the compute nodes, providing very high aggregate bandwidth across the cluster. Both **Map/Reduce** and the **Hadoop Distributed File System** are designed so that node failures are automatically handled by the framework.

   There are other components of Hadoop:

1. HBase:

   HBase is the Hadoop database. Think of it as a distributed, scalable, big data store.
2. Cassandra:

   Apache Cassandra is an open source distributed database management system. It is a NoSQL solution that was initially developed by Facebook and powered their Inbox Search feature until late 2010.
3. Hive:

   Hive is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems. So, actually, it converts SQL-like language to Map-Reduce programs

Data storage in Big Data:

Big data storage is a compute-and-**storage** architecture that collects and manages large **data** sets and enables real-time **data** analytics. Although a specific volume size or capacity is not formally defined, **big data storage** usually refers to volumes that grow exponentially to terabyte or petabyte scale.

People automatically associate HDFS, or Hadoop Distributed File System, with Hadoop **data** warehouses. HDFS stores information in clusters that are made up of smaller blocks. These blocks are **stored** in onsite physical **storage** units, such as internal disk drives.

The key requirements of big data storage are that it can handle very large amounts of data and keep scaling to keep up with growth, and that it can provide the input/output operations per second (IOPS) necessary to deliver data to analytics tools.

The largest big data practitioners – Google, Facebook, Apple, etc – run what are known as hyperscale computing environments.

These comprise vast amounts of commodity servers with direct-attached storage (DAS). Redundancy is at the level of the entire compute/storage unit, and if a unit suffers an outage of any component it is replaced wholesale, having already failed over to its mirror.

Such environments run the likes of Hadoop, NoSQL and Cassandra as analytics engines, and typically have PCIe flash storage alone in the server or in addition to disk to cut storage latency to a minimum. There's no shared storage in this type of configuration.

The other storage format that is built for very large numbers of files is object storage. This tackles the same challenge as scale-out NAS (Network Associated Storage) – that traditional tree-like file systems become unwieldy when they contain large numbers of files. Object-based storage gets around this by giving each file a unique identifier and indexing the data and its location. It's more like the DNS way of doing things on the internet than the kind of file system we're used to.

Object storage systems can scale to very high capacity and large numbers of files in the billions, so are another option for enterprises that want to take advantage of big data. Having said that, object storage is a less mature technology than scale-out NAS.Big data storage needs to be able to handle capacity and provide low latency for analytics work.

Security and Data privacy:

Big data security is the collective term for all the measures and tools used to guard both the **data** and analytics processes from attacks, theft, or other malicious activities that could harm or negatively affect them

Some of the major challenges in securing Big data are:

- **Secure Computations**: Big data technologies use distributed programming frameworks to process large amounts of data. These distributed frameworks like MapReduce don't have good

security protections. In MapReduce, the data is split, then processed by a mapper and allocated storage. If someone can change the mapper settings as it doesn't have any additional security layer, it can manipulate the data being processed. Also, it is very difficult to detect these untrusted mappers. It is very important to secure the computations being handled in these distributed programming frameworks so as to ensure that the integrity of data is maintained.

- **Protecting Data and Transaction logs:** Due to the size of data and transaction logs, these are stored in multi-tiered storage environments with auto-tiering functionality. Auto-tiering does not keep track of the data location. Auto-tiering systems can expose new vulnerabilities because of unknown physical data locations, untrusted storage devices which can result in organizations losing control over data. Data transmission between tiers can also provide information regarding user activities and data properties which can be used by attackers. Data and transaction logs need to be protected to maintain the confidentiality, integrity, and availability of data.

- **Validation of Inputs from Endpoints:** Big data collects data from a variety of input devices including endpoints. It may be collecting logs from a large number of devices and applications. The data which Big data is receiving might contain rogue data being sent by an untrusted endpoint. This can affect the organization's analytical outputs. A challenge here is to validate all the inputs the Big data is receiving to ensure that it came from a trusted source.

- **Secure Non-Relational Data Stores:** Non-Relational data stores like NoSQL are rapidly being used in Big data technologies. These data stores are not mature and secure enough, as of today. They have many security issues like no encryption support for the data files, weak authentication between client and server, data at rest is unencrypted which can cause privacy threats.

- **Privacy-preserving data analytics:** Privacy is an important issue in applying Big data technologies for analytics. As more and more data is being collected, this data aggregation along with data analytics could result in user privacy violation. If the data analytics is outsourced, an untrusted third party employee can infer personal information of users. The organizations want to use Big data analytics tools to enhance customer satisfaction, but they need to ensure protecting user privacy while doing so.

- **Access control:** Big data handles a variety of data including sensitive data such as Personally Identifiable Information of users. There are many legal and compliance requirements to protect

those data. Granular access control policies should be implemented so that only authorized users to have access to sensitive user data and analytics done on those data sets. This is needed to ensure the confidentiality of data.

- **Real-time security monitoring:** Real-time security monitoring is needed for Big data infrastructure and the analytics it is handling. It has always been a difficult task because of the number of alerts generated by devices. These alerts have a large number of false positives as well. Due to this reason, companies often struggle to monitor real-time data.

Encryption can help in handling data protection in Big data technologies at multiple stages to ensure confidentiality, integrity, and availability of data is maintained.

Big data privacy:

While big data offers a variety of benefits to organizations of all shapes and sizes, it also comes with several noteworthy privacy risks including:

1. **Data breaches:** Data breaches occur when information is accessed without authorization. Data breaches are the result of out-of-date software, weak passwords, and targeted malware attacks. Keeping software up-to-date, changing passwords often, and educating employees on best security practices can all help prevent data breaches.
2. **Data brokerage:** The sale of unprotected and incorrect data is considered data brokerage. Some companies gather and sell customer profiles, which contain false information that leads to flawed algorithms. Before buying data, organizations should do their research and make sure they are receiving data from a reputable provider that offers accurate data.

3. **Data discrimination:** Since data can consist of customer demographic information, organizations may develop algorithms that penalize individuals based on age, gender, or ethnicity. Organizations should always have a thorough and accurate representation of customers, account for biases, and put fairness above analytics.

Data privacy best practices for big data

There are certain strategies organizations can use to protect big data. Several of the best practices for maintaining the privacy of big data include:

Employ real-time monitoring

Since a privacy issue can happen at any moment, organizations should find a solution that monitors data in real-time. This way, they'll be aware of a problem as soon as it happens and can take appropriate steps to resolve it right away.

Implement homomorphic encryption

Homomorphic encryption is a form of encryption that allows users to compute data without decrypting it first. This form of encryption should be implemented to store and process information in the cloud to prevent organizations from revealing private information to outside vendors.

Avoid collecting too much data

Only the data that is absolutely necessary should be collected. An organization may not need the Social Security numbers of their customers; customer login usernames and passwords may only be necessary. Organizations should consider deleting any personal information that is not needed to best protect customer data privacy.

Prevent internal threats

Organizations are also exposed to internal privacy risks from angry or simply uninformed employees. Therefore, it's essential to educate all employees on best practices for ensuring data privacy like changing passwords frequently and logging off unused computers.

Big data privacy tools:

When shopping around for big data privacy tool, here are some features and capabilities organizations should look for:

- **Cloud-compatible:** It's essential for a big data privacy tool to be compatible with the cloud. If it only works on a physical server or computer, it's likely an out-of-date solution that cannot keep up with today's big data privacy challenges.
- **User-friendly design:** If an organization has to spend a lot of time learning how to use a tool, it's probably not a great fit. A tool should offer an intuitive design that various employees can use with confidence.
- **Automation:** Most organizations don't have the time to manually protect their data from privacy threats. Therefore, organizations should opt for a tool that allows them to automate the process. Automation can free up companies so they can focus on running their business and meeting goals.

Application Areas:

Big Data is a powerful tool that makes things ease in various fields as said above. Big data used in so many applications they are **banking**, agriculture, chemistry, data **mining**, cloud computing, finance, marketing, stocks, BDA, health care etc

1.Banking and Securities

Industry-specific Big Data Challenges

The challenges in this industry include: securities fraud early warning, tick analytics, card fraud detection, archival of audit trails, enterprise credit risk reporting, trade visibility, customer data transformation, social analytics for trading, IT operations analytics, and IT policy compliance analytics, among others.

Applications of Big Data in the Banking and Securities Industry

- The Securities Exchange Commission (SEC) is using big data to monitor financial market activity. They are currently using network analytics and natural language processors to catch illegal trading activity in the financial markets.
- Retail traders, Big banks, hedge funds, and other so-called 'big boys' in the financial markets use big data for trade analytics used in high-frequency trading, pre-trade decision-support analytics, sentiment measurement, Predictive Analytics, etc.
- This industry also heavily relies on big data for risk analytics, including; anti-money laundering, demand enterprise risk management, "Know Your Customer," and fraud mitigation.
- Big Data providers are specific to this industry includes 1010data, Panopticon Software, Streambase Systems, Nice Actimize, and Quartet FS.

2.**Communications, Media and Entertainment**

Industry-specific Big Data Challenges

Since consumers expect rich media on-demand in different formats and a variety of devices, some big data challenges in the communications, media, and entertainment industry include:

- Collecting, analyzing, and utilizing consumer insights
- Leveraging mobile and social media content
- Understanding patterns of real-time, media content usage

Applications of Big Data in the Communications, Media and Entertainment Industry

Organizations in this industry simultaneously analyze customer data along with behavioral data to create detailed customer profiles that can be used to:

- Create content for different target audiences
- Recommend content on demand
- Measure content performance
    - A case in point is the Wimbledon Championships (YouTube Video) that leverages big data to deliver detailed sentiment analysis on the tennis matches to TV, mobile, and web users in real-time.
    - Spotify, an on-demand music service, uses Hadoop big data analytics, to collect data from its millions of users worldwide and then uses the analyzed data to give informed music recommendations to individual users.
    - Amazon Prime, which is driven to provide a great customer experience by offering video, music, and Kindle books in a one-stop-shop, also heavily utilizes big data.
    - Big Data Providers in this industry include Infochimps, Splunk, Pervasive Software, and Visible Measures.

3. Healthcare Providers

Industry-specific Big Data Challenges

The healthcare sector has access to huge amounts of data but has been plagued by failures in utilizing the data to curb the cost of rising healthcare and by inefficient systems that stifle faster and better healthcare benefits across the board. This is mainly because electronic data is unavailable,

inadequate, or unusable. Additionally, the healthcare databases that hold health-related information have made it difficult to link data that can show patterns useful in the medical field.

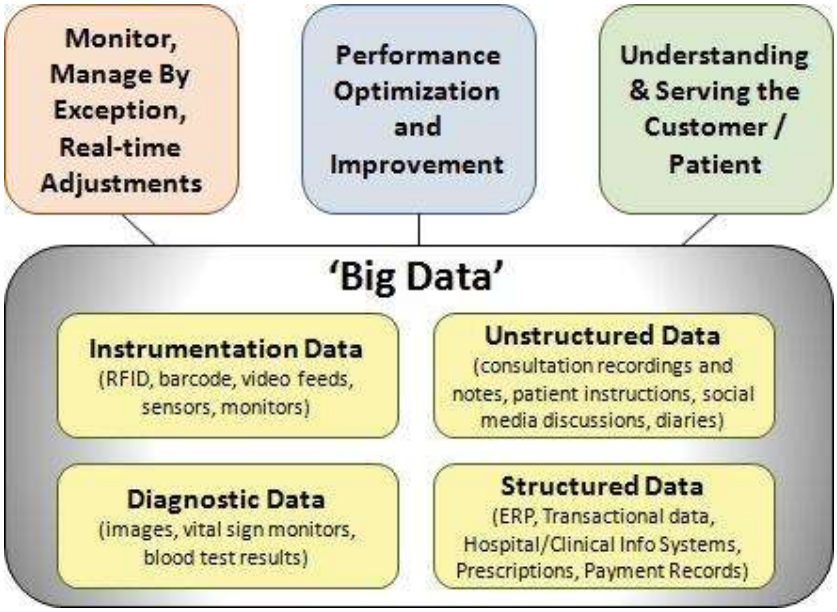


Fig 7. Big Data in the Healthcare Sector

Other challenges related to big data include the exclusion of patients from the decision-making process and the use of data from different readily available sensors.

Applications of Big Data in the Healthcare Sector

Some hospitals, like Beth Israel, are using data collected from a cell phone app, from millions of patients, to allow doctors to use evidence-based medicine as opposed to administering several medical/lab tests to all patients who go to the hospital. A battery of tests can be efficient, but it can also be expensive and usually ineffective. Free public health data and Google Maps have been used by the University of Florida to create visual data that allows for faster identification and efficient analysis of healthcare information, used in tracking the spread of chronic disease.

Obamacare has also utilized big data in a variety of ways.

Big Data Providers in this industry include Recombinant Data, Humedica, Explorys, and Cerner.

4. **Education**

Industry specific Big Data Challenges

From a technical point of view, a significant challenge in the education industry is to incorporate big data from different sources and vendors and to utilize it on platforms that were not

designed for the varying data. From a practical point of view, staff and institutions have to learn new data management and analysis tools. On the technical side, there are challenges to integrating data from different sources on different platforms and from different vendors that were not designed to work with one another. Politically, issues of privacy and personal data protection associated with big data used for educational purposes is a challenge.

Applications of Big Data in Education

Big data is used quite significantly in higher education. For example, The University of Tasmania. An Australian university with over 26000 students has deployed a Learning and Management System that tracks, among other things, when a student logs onto the system, how much time is spent on different pages in the system, as well as the overall progress of a student over time.

In a different use case of the use of big data in education, it is also used to measure teacher's effectiveness to ensure a pleasant experience for both students and teachers. Teacher's performance can be fine-tuned and measured against student numbers, subject matter, student demographics, student aspirations, behavioral classification, and several other variables.

On a governmental level, the Office of Educational Technology in the U. S. Department of Education is using big data to develop analytics to help correct course students who are going astray while using online big data courses. Click patterns are also being used to detect boredom. Big Data Providers in this industry include Knewton and Carnegie Learning and MyFit/Naviance.

5. Manufacturing and Natural Resources

Industry-specific Big Data Challenges

Increasing demand for natural resources, including oil, agricultural products, minerals, gas, metals, and so on, has led to an increase in the volume, complexity, and velocity of data that is a challenge to handle. Similarly, large volumes of data from the manufacturing industry are untapped. The underutilization of this information prevents the improved quality of products, energy efficiency, reliability, and better profit margins.

Applications of Big Data in Manufacturing and Natural Resources

In the natural resources industry, big data allows for predictive modeling to support decision making that has been utilized for ingesting and integrating large amounts of data from geospatial data, graphical data, text, and temporal data. Areas of interest where this has been used include; seismic interpretation and reservoir characterization.

Big data has also been used in solving today's manufacturing challenges and to gain a competitive advantage, among other benefits.

6. Government

Industry-specific Big Data Challenges

In governments, the most significant challenges are the integration and interoperability of big data across different government departments and affiliated organizations.

Applications of Big Data in Government

In public services, big data has an extensive range of applications, including energy exploration, financial market analysis, fraud detection, health-related research, and environmental protection.

Some more specific examples are as follows:

Big data is being used in the analysis of large amounts of social disability claims made to the Social Security Administration (SSA) that arrive in the form of unstructured data. The analytics are used to process medical information rapidly and efficiently for faster decision making and to detect suspicious or fraudulent claims.

The Food and Drug Administration (FDA) is using big data to detect and study patterns of food-related illnesses and diseases. This allows for a faster response, which has led to more rapid treatment and less death.

The Department of Homeland Security uses big data for several different use cases. Big data is analyzed from various government agencies and is used to protect the country. Big Data Providers in this industry include Digital Reasoning, Socrata, and HP.

7. Insurance

Industry-specific Big Data Challenges

Lack of personalized services, lack of personalized pricing, and the lack of targeted services to new segments and specific market segments are some of the main challenges. In a survey conducted by Market force challenges identified by professionals in the insurance industry include underutilization of data gathered by loss adjusters and a hunger for better insight.

Applications of Big Data in the Insurance Industry

Big data has been used in the industry to provide customer insights for transparent and simpler products, by analyzing and predicting customer behavior through data derived from social media, GPS-enabled devices, and CCTV footage. The big data also allows for better customer retention from insurance companies.

When it comes to claims management, predictive analytics from big data has been used to offer faster service since massive amounts of data can be analyzed mainly in the underwriting stage. Fraud detection has also been enhanced. Through massive data from digital channels and social media, real-time monitoring of claims throughout the claims cycle has been used to provide insights. Big Data Providers in this industry include Sprint, Qualcomm, Octo Telematics, The Climate Corp.

8. Retail and Wholesale trade

Industry-specific Big Data Challenges

From traditional brick and mortar retailers and wholesalers to current day e-commerce traders, the industry has gathered a lot of data over time. This data, derived from customer loyalty cards, POS scanners, RFID, etc. are not being used enough to improve customer experiences on the whole. Any changes and improvements made have been quite slow.

Applications of Big Data in the Retail and Wholesale Industry

Big data from customer loyalty data, POS, store inventory, local demographics data continues to be gathered by retail and wholesale stores. In New York's Big Show retail trade conference in 2014, companies like Microsoft, Cisco, and IBM pitched the need for the retail industry to utilize big data for analytics and other uses, including:

- Optimized staffing through data from shopping patterns, local events, and so on
- Reduced fraud
- Timely analysis of inventory

Social media use also has a lot of potential use and continues to be slowly but surely adopted, especially by brick and mortar stores. Social media is used for customer prospecting, customer retention, promotion of products, and more. Big Data Providers in this industry include First Retail, First Insight, Fujitsu, Infor, Epicor, and Vistex.

8.Transportation

Industry-specific Big Data Challenges

In recent times, huge amounts of data from location-based social networks and high-speed data from telecoms have affected travel behavior. Regrettably, research to understand travel behavior has not progressed as quickly. In most places, transport demand models are still based on poorly understood new social media structures.

Applications of Big Data in the Transportation Industry

Some applications of big data by governments, private organizations, and individuals include:

- Governments use of big data: traffic control, route planning, intelligent transport systems, congestion management (by predicting traffic conditions)
- Private-sector use of big data in transport: revenue management, technological enhancements, logistics and for competitive advantage (by consolidating shipments and optimizing freight movement)

- Individual use of big data includes route planning to save on fuel and time, for travel arrangements in tourism, etc.

- Big Data Providers in this industry include Qualcomm and Manhattan Associates.

10. Energy and Utilities

Industry-specific Big Data Challenges

Applications of Big Data in the Energy and Utility Industry

Smart meter readers allow data to be collected almost every 15 minutes as opposed to once a day with the old meter readers. This granular data is being used to analyze the consumption of utilities better, which allows for improved customer feedback and better control of utilities use.

In utility companies, the use of big data also allows for better asset and workforce management, which is useful for recognizing errors and correcting them as soon as possible before complete failure is experienced. Big Data Providers in this industry include Alstom Siemens ABB and Cloudera.

APPLICATION TOOLS AND PLATFORMS:

Tools:

1. Hadoop

Apache Hadoop is the most prominent and used tool in big data industry with its enormous capability of large-scale processing data. This is 100% open source framework and runs on commodity hardware in an existing data center. Furthermore, it can run on a cloud infrastructure. Hadoop consists of four parts:

- **Hadoop Distributed File System:** Commonly known as HDFS, it is a distributed file system compatible with very high scale bandwidth.
- **MapReduce:** A programming model for processing big data.
- **YARN:** It is a platform used for managing and scheduling Hadoop's resources in Hadoop infrastructure.
- **Libraries:** To help other modules to work with Hadoop.

2. Apache Spark

Apache Spark is the next hype in the industry among the big data tools. The key point of this open source big data tool is it fills the gaps of Apache Hadoop concerning data processing. Interestingly, Spark can handle both batch data and real-time data. As Spark does in-memory data processing, it processes data much faster than traditional disk processing. This is indeed a plus point for data analysts handling certain types of data to achieve the faster outcome.

Apache Spark is flexible to work with HDFS as well as with other data stores, for example with OpenStack Swift or Apache Cassandra. It's also quite easy to run Spark on a single local system to make development and testing easier.

Spark Core is the heart of the project, and it facilitates many things like

- distributed task transmission
- scheduling
- I/O functionality

Spark is an alternative to Hadoop's MapReduce. Spark can run jobs 100 times faster than Hadoop's MapReduce.

3. Apache Storm

Apache Storm is a distributed real-time framework for reliably processing the unbounded data stream. The framework supports any programming language. The unique features of Apache Storm are:

- Massive scalability
- Fault-tolerance
- "fail fast, auto restart" approach
- The guaranteed process of every tuple
- Written in Clojure
- Runs on the JVM
- Supports direct acrylic graph(DAG) topology
- Supports multiple languages
- Supports protocols like JSON

Storm topologies can be considered similar to MapReduce job. However, in case of Storm, it is real-time stream data processing instead of batch data processing. Based on the topology configuration, Storm scheduler distributes the workloads to nodes. Storm can interoperate with Hadoop's HDFS through adapters if needed which is another point that makes it useful as an open source big data tool.

4. Cassandra

Apache Cassandra is a distributed type database to manage a large set of data across the servers. This is one of the best big data tools that mainly processes structured data sets. It provides highly available service with no single point of failure. Additionally, it has certain capabilities which no other relational database and any NoSQL database can provide. These capabilities are:

- Continuous availability as a data source
- Linear scalable performance
- Simple operations
- Across the data centers easy distribution of data
- Cloud availability points
- Scalability
- Performance

Apache Cassandra architecture does not follow master-slave architecture, and all nodes play the same role. It can handle numerous concurrent users across data centers. Hence, adding a new node is no matter in the existing cluster even at its up time.

5. RapidMiner

RapidMiner is a software platform for data science activities and provides an integrated environment for:

- Preparing data
- Machine learning
- Text mining
- Predictive analytics
- Deep learning
- Application development
- Prototyping

This is one of the useful big data tools that support different steps of machine learning, such as:

- Data preparation
- Visualization
- Predictive analytics
- Model validation
- Optimization
- Statistical modeling
- Evaluation
- Deployment
-

RapidMiner follows a client/server model where the server could be located on-premise, or in a cloud infrastructure. It is written in Java and provides a GUI to design and execute workflows. It can provide 99% of an advanced analytical solution.

6. MongoDB

MongoDB is an open source NoSQL database which is cross-platform compatible with many built-in features. It is ideal for the business that needs fast and real-time data for instant decisions. It is ideal for the users who want data-driven experiences. It runs on MEAN software stack, NET applications and, Java platform.

Some notable features of MongoDB are:

- It can store any type of data like integer, string, array, object, boolean, date etc.
- It provides flexibility in cloud-based infrastructure.
- It is flexible and easily partitions data across the servers in a cloud structure.
- MongoDB uses dynamic schemas. Hence, you can prepare data on the fly and quickly. This is another way of cost saving.

7. R Programming Tool

This is one of the widely used open source big data tools in big data industry for statistical analysis of data. The most positive part of this big data tool is – although used for statistical analysis, as a user you don't have to be a statistical expert. R has its own public library CRAN (Comprehensive

R Archive Network) which consists of more than 9000 modules and algorithms for statistical analysis of data.

R can run on Windows and Linux server as well inside SQL server. It also supports Hadoop and Spark. Using R tool one can work on discrete data and try out a new analytical algorithm for analysis. It is a portable language. Hence, an R model built and tested on a local data source can be easily implemented in other servers or even against a Hadoop data lake.

8. Neo4j

Hadoop may not be a wise choice for all big data related problems. For example, when you need to deal with large volume of network data or graph related issue like social networking or demographic pattern, a graph database may be a perfect choice. Neo4j is one of the big data tools that is widely used graph database in big data industry. It follows the fundamental structure of graph database which is interconnected node-relationship of data. It maintains a key-value pattern in data storing.

Notable features of Neo4j are:

- It supports ACID transaction
- High availability
- Scalable and reliable
- Flexible as it does not need a schema or data type to store data
- It can integrate with other databases
- Supports query language for graphs which is commonly known as Cypher.

9. Apache SAMOA

Apache SAMOA is among well known big data tools used for distributed streaming algorithms for big data mining. Not only data mining it is also used for other machine learning tasks such as:

- Classification
- Clustering
- Regression

- Programming abstractions for new algorithms

It runs on the top of distributed stream processing engines (DSPEs). Apache Samoa is a pluggable architecture and allows it to run on multiple DSPEs which include

- Apache Storm
- Apache S4
- Apache Samza
- Apache Flink

Due to below reasons, Samoa has got immense importance as the open source big data tool in the industry:

- You can program once and run it everywhere
- Its existing infrastructure is reusable. Hence, you can avoid deploying cycles.
- No system downtime
- No need for complex backup or update process

10. HPCC

High-Performance Computing Cluster (HPCC) is another among best big data tools. It is the competitor of Hadoop in big data market. It is one of the open source big data tools under the Apache 2.0 license. Some of the core features of HPCC are:

- Helps in parallel data processing
- Open Source distributed data computing platform
- Follows shared nothing architecture
- Runs on commodity hardware
- Comes with binary packages supported for Linux distributions
- Supports end-to-end big data workflow management
- The platform includes:

Thor: for batch-oriented data manipulation, their linking, and analytics

Roxie: for real-time data delivery and analytics

- Implicitly a parallel engine
- Maintains code and data encapsulation

- Extensible
- Highly optimized
- Helps to build graphical execution plans
- It compiles into C++ and native machine code

BIG DATA PLATFORM:

Big data platform is a type of IT solution that combines the features and capabilities of several **big data** application and utilities within a single solution. It is an enterprise class IT **platform** that enables organization in developing, deploying, operating and managing a **big data** infrastructure /environment.

Some of Big Data Platforms are:

- 1010data.
- Cloudera.
- Pivotal **Big Data** Suite.
- Microsoft Azure HDInsight.
- SAP HANA.
- Vertica.
- Teradata Aster Discovery **Platform**.
- Oracle **Big Data** SQL

Q.No.	QUESTION
1.	Analyze the differences between RDBMS and Hadoop?
2.	Explain Map reduce with example
3.	Outline the main components of Map reduce job?
4.	Relate namenode and datanode in Hadoop?
5.	Explore the basic characteristics of Hadoop?

PART B

Q.No.	QUESTION
1.	Illustrate the YARN architecture.
2.	With neat block diagram explain Mapreduce
3.	Explain all the daemons of hadoop.
4.	Figure out how HDFS is able to solve big data problem.
5.	Investigate the read - write anatomy with neat diagram.

UNIT – III - HADOOP& HDFS– SECA7018

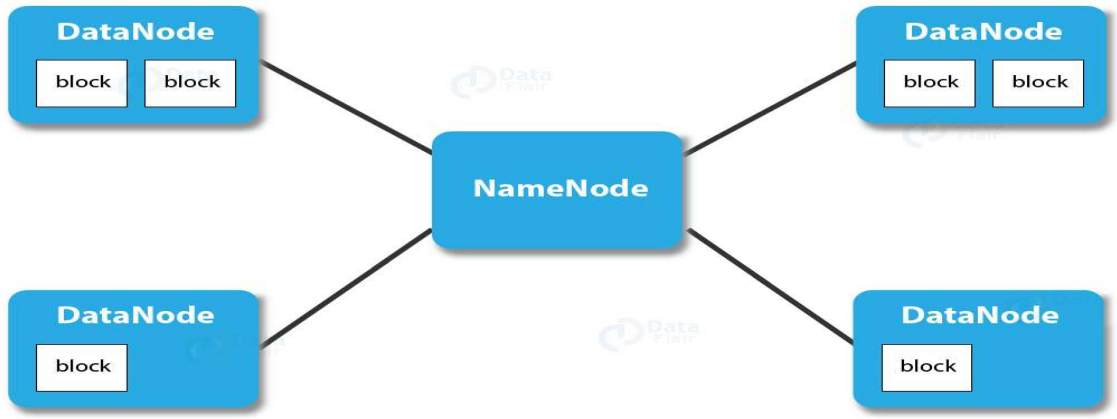
Clustered Hadoop environment, HDFS and data managements using HDFS, Analytics Using Map Reduce and programming, Map Reduce design patterns.

A Hadoop cluster is nothing but a group of computers connected together via LAN. We use it for storing and processing large data sets. Hadoop clusters have a number of commodity hardware connected together. They communicate with a high-end machine which acts as a master. These master and slaves implement distributed computing over distributed data storage. It runs open source software for providing distributed functionality.

Hadoop cluster has master-slave architecture.

i. Master in Hadoop Cluster

It is a machine with a good configuration of memory and CPU. There are two daemons running on the master and they are Name Node and Resource Manager.

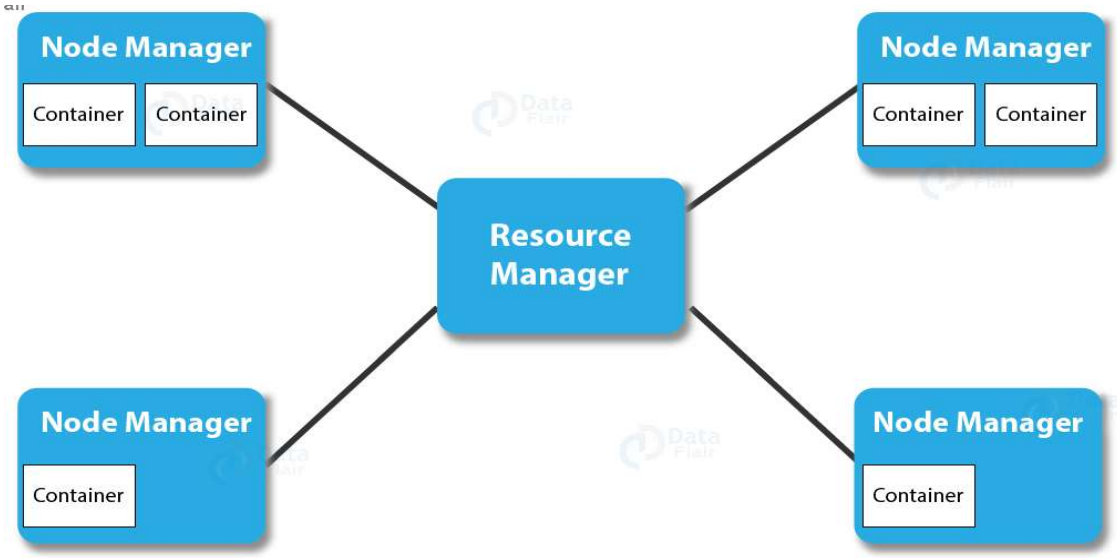


a. Functions of Name Node

- Manages file system namespace
- Regulates access to files by clients
- Stores metadata of actual data Foe example – file path, number of blocks, block id, the location of blocks etc.

b. Functions of Resource Manager

- Executes file system namespace operations like opening, closing, renaming files and directories
- The Name Node stores the metadata in the memory for fast retrieval. Hence we should configure it on a high-end machine.



- It arbitrates resources among competing nodes
- Keeps track of live and dead nodes

ii. Slaves in the Hadoop Cluster

It is a machine with a normal configuration. There are two daemons running on Slave machines and they are – DataNode and Node Manager

a. Functions of DataNode

- It stores the business data
- It does read, write and data processing operations
- Upon instruction from a master, it does creation, deletion, and replication of data blocks.

b. Functions of NodeManager

- It runs services on the node to check its health and reports the same to ResourceManager.
- We can easily scale Hadoop cluster by adding more nodes to it. Hence we call it a linearly scaled cluster. Each node added increases the throughput of the cluster.

c. Functions of the client node

- To load the data on the Hadoop cluster.
- Tells how to process the data by submitting MapReduce job.
- Collects the output from a specified location.

Single Node Cluster VS Multi-Node Cluster

As the name suggests, single node cluster gets deployed over a single machine. And multi-node clusters gets deployed on several machines.

In **single-node** Hadoop clusters, all the daemons like NameNode, DataNode run on the same machine. In a single node Hadoop cluster, all the processes run on one JVM instance. The user need not make any configuration setting. The Hadoop user only needs to set JAVA_HOME variable. The default factor for single node Hadoop cluster is one.
In **multi-node** Hadoop clusters, the daemons run on separate host or machine. A multi-node Hadoop cluster has master-slave architecture. In this NameNode daemon run on the master machine. And DataNode daemon runs on the slave machines. In multi-node Hadoop cluster, the slave daemons like DataNode and NodeManager run on cheap machines. On the other hand, master daemons like NameNode and ResourceManager run on powerful servers. Ina multi-node Hadoop cluster, slave machines can be present in any location irrespective of the physical location of the master server.

COMMUNICATION PROTOCOLS USED IN HADOOP CLUSTERS

The HDFS communication protocol works on the top of TCP/IP protocol. The client establishes a connection with NameNode using configurable TCP port. Hadoop cluster establishes the connection to the client using client protocol. DataNode talks to NameNode using the DataNode Protocol. A Remote Procedure Call (RPC) abstraction wraps both Client protocol and DataNode protocol. NameNode does not initiate any RPC instead it responds to RPC from the DataNode.

How to Build a Cluster in Hadoop

Building a Hadoop cluster is a non- trivial job. Ultimately the performance of our system will depend upon how we have configured our cluster. The various parameters one should take into consideration while setting up a Hadoop cluster.

- Understand the kind of workloads, the cluster will be dealing with. The volume of data which cluster need to handle. And kind of processing required like CPU bound, I/O bound etc.
- Data storage methodology like data compression technique used if any.
- Data retention policy like how frequently we need to flush.

Sizing the Hadoop Cluster

For determining the size of Hadoop clusters we need to look at how much data is in hand. We should also examine the daily data generation. Based on these factors we can decide the requirements of a number of machines and their configuration. There should be a balance between performance and cost of the hardware approved.

Configuring Hadoop Cluster

For deciding the configuration of Hadoop cluster, run typical Hadoop jobs on the default configuration to get the baseline. We can analyze job history log files to check if a job takes more time than expected. If so then change the configuration. After that repeat the same process to fine tune the Hadoop cluster configuration so that it meets the business requirement. Performance of the cluster greatly depends upon resources allocated to the daemons. The Hadoop cluster allocates one

CPU core for small to medium data volume to each DataNode. And for large data sets, it allocates two CPU cores to the HDFS daemons.

HADOOP CLUSTER MANAGEMENT

When you deploy your Hadoop cluster in production it is apparent that it would scale along all dimensions. They are volume, velocity, and variety. Various features that it should have to become production-ready are – robust, round the clock availability, performance and manageability. Hadoop cluster management is the main aspect of your big data initiative.

A good cluster management tool should have the following features: -

- It should provide diverse work-load management, security, resource provisioning, performance optimization, health monitoring. Also, it needs to provide policy management, job scheduling, back up and recovery across one or more nodes.
- Implement NameNode high availability with load balancing, auto-failover, and hot standbys
- Enabling policy-based controls that prevent any application from gulping more resources than others.
- Managing the deployment of any layers of software over Hadoop clusters by performing regression testing. This is to make sure that any jobs or data won't crash or encounter any bottlenecks in daily operations.

BENEFITS OF HADOOP CLUSTERS

Here is a list of benefits provided by Clusters in Hadoop –

- Robustness
- Data disks failures, heartbeats and re-replication
- Cluster Rebalancing
- Data integrity
- Metadata disk failure
- Snapshot

i. Robustness

The main objective of Hadoop is to store data reliably even in the event of failures. Various kind of failure is NameNode failure, DataNode failure, and network partition. DataNode periodically sends a heartbeat signal to NameNode. In network partition, a set of DataNodes gets disconnected with the NameNode. Thus NameNode does not receive any heartbeat from these DataNodes. It marks these DataNodes as dead. Also, Namenode does not forward any I/O request to them. The replication factor of the blocks stored in these DataNodes falls below their specified value. As a result, NameNode initiates replication of these blocks. In this way, NameNode recovers from the failure.

ii. Data Disks Failure, Heartbeats, and Re-replication

NameNode receives a heartbeat from each DataNode. NameNode may fail to receive heartbeat because of certain reasons like network partition. In this case, it marks these nodes as dead. This

decreases the replication factor of the data present in the dead nodes. Hence NameNode initiates replication for these blocks thereby making the cluster fault tolerant.

iii. Cluster Rebalancing

The HDFS architecture automatically does cluster rebalancing. Suppose the free space in a DataNode falls below a threshold level. Then it automatically moves some data to another DataNode where enough space is available.

iv. Data Integrity

Hadoop cluster implements checksum on each block of the file. It does so to see if there is any corruption due to buggy software, faults in storage device etc. If it finds the block corrupted it seeks it from another DataNode that has a replica of the block.

v. Metadata Disk Failure

FSImage and Editlog are the central data structures of HDFS. Corruption of these files can stop the functioning of HDFS. For this reason, we can configure NameNode to maintain multiple copies of FSImage and EditLog. Updation of multiple copies of FSImage and EditLog can degrade the performance of Namespace operations. But it is fine as Hadoop deals more with the data-intensive application rather than metadata intensive operation.

vi. Snapshot

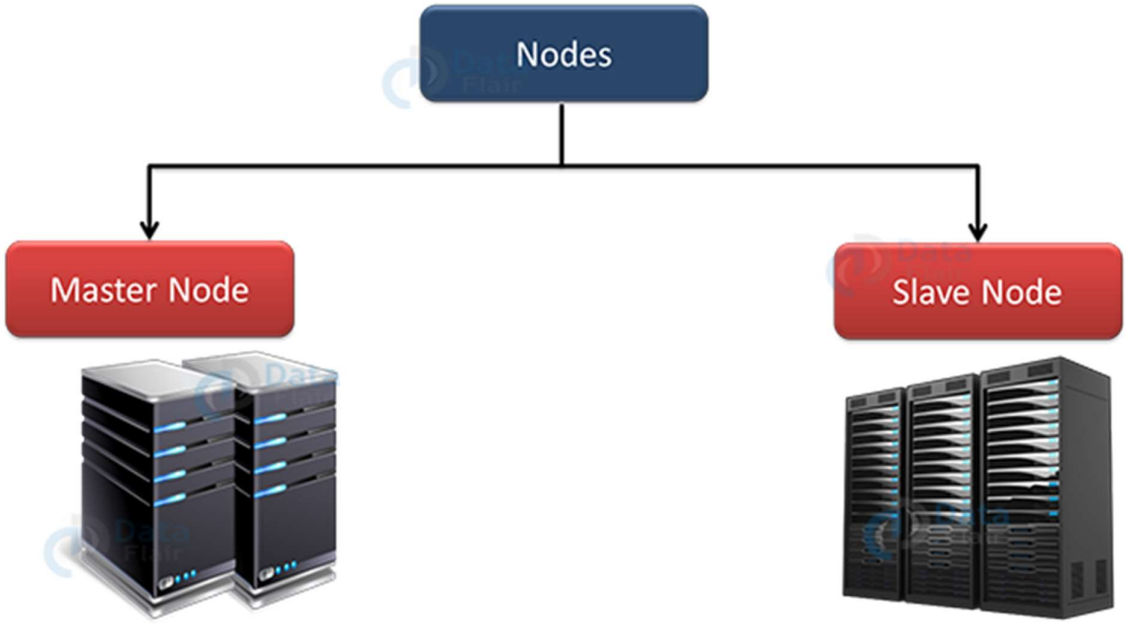
Snapshot is nothing but storing a copy of data at a particular instance of time. One of the usages of the snapshot is to rollback a failed HDFS instance to a good point in time. We can take Snapshots of the sub-tree of the file system or entire file system. Some of the uses of snapshots are disaster recovery, data backup, and protection against user error. We can take snapshots of any directory. Only the particular directory should be set as Snapshottable. The administrators can set any directory as snapshottable. We cannot rename or delete a snapshottable directory if there are snapshots in it. After removing all the snapshots from the directory, we can rename or delete it.

HDFS Tutorial – Introduction

Hadoop Distributed File system – HDFS is the world's most reliable storage system. HDFS is a Filesystem of Hadoop designed for storing very large files running on a cluster of commodity hardware. It is designed on the principle of storage of less number of large files rather than the huge number of small files. Hadoop HDFS provides a fault-tolerant storage layer for Hadoop and its other components. HDFS Replication of data helps us to attain this feature. It stores data reliably, even in the case of hardware failure. It provides high throughput access to application data by providing the data access in parallel.

HDFS Nodes

As we know, Hadoop works in master-slave fashion, HDFS also has two types of nodes that work in the same manner. These are the NameNode(s) and the DataNodes.



HDFS Master (Namenode)

NameNode regulates file access to the clients. It maintains and manages the slave nodes and assigns tasks to them. NameNode executes file system namespace operations like opening, closing, and renaming files and directories. NameNode runs on the high configuration hardware.

HDFS Slave (Datanode)

There are n number of slaves (where n can be up to 1000) or DataNodes in the Hadoop Distributed File System that manages storage of data. These slave nodes are the actual worker nodes that do the tasks and serve read and write requests from the file system's clients.They perform block creation, deletion, and replication upon instruction from the NameNode. Once a block is written on a DataNode, it replicates it to other DataNode, and the process continues until creating the required number of replicas. DataNodes runs on commodity hardware having an average configuration.

HADOOP HDFS DAEMONS

There are two daemons which run on HDFS for data storage:

Namenode: This is the daemon that runs on all the masters. NameNode stores metadata like filename, the number of blocks, number of replicas, a location of blocks, block IDs, etc.

This metadata is available in memory in the master for faster retrieval of data. In the local disk, a copy of the metadata is available for persistence. So NameNode memory should be high as per the requirement.

Datanode: This is the daemon that runs on the slave. These are actual worker nodes that store the data.

Data storage in HDFS

Hadoop HDFS broke the files into small pieces of data known as blocks. The default block size in HDFS is 128 MB. We can configure the size of the block as per the requirements. These blocks are stored in the cluster in a distributed manner on different nodes. This provides a mechanism for MapReduce to process the data in parallel in the cluster.

HDFS stores multiple copies of each block across the cluster on different nodes. This is a replication of data. By default, the HDFS replication factor is 3. Hadoop HDFS provides high availability, fault tolerance, and reliability. HDFS splits a large file into n number of small blocks and stores them on different DataNodes in the cluster in a distributed manner. It replicates each block and stored them across different DataNodes in the cluster.

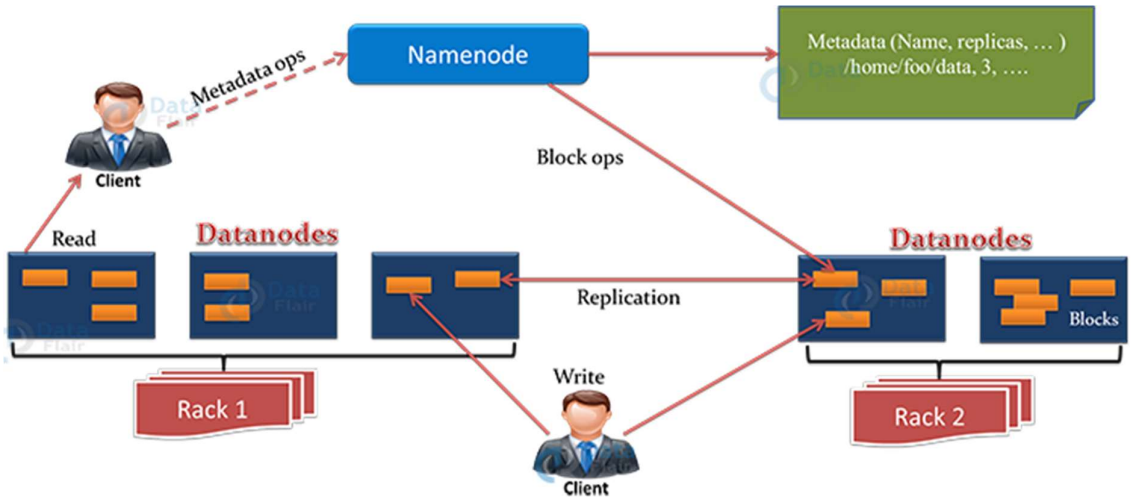
Rack Awareness in Hadoop HDFS

Hadoop runs on a cluster of computers spread commonly across many racks.

NameNode places replicas of a block on multiple racks for improved fault tolerance. NameNode tries to place at least one replica of a block in a different rack so that if a complete rack goes down, then also the system will be highly available. Optimize replica placement distinguishes HDFS from other distributed file systems. The purpose of a rack-aware replica placement policy is to improve data reliability, availability, and network bandwidth utilization.

HDFS Architecture

This architecture gives you a complete picture of the Hadoop Distributed File System. There is a single NameNode that stores metadata, and there are multiple DataNodes that do actual storage work. Nodes are arranged in racks, and replicas of data blocks are stored on different racks in the cluster to provide fault tolerance.



To read or write a file in HDFS, the client needs to interact with NameNode. HDFS applications need a write-once-read-many access model for files. A file, once created and written, cannot be edited.

Hadoop HDFS Architecture Tutorial

NameNode stores metadata, and DataNode stores actual data. The client interacts with NameNode for performing any tasks, as NameNode is the centerpiece in the cluster. There are several DataNodes in the cluster which store HDFS data in the local disk. DataNode sends a heartbeat message to NameNode periodically to indicate that it is alive. Also, it replicates data to other DataNode as per the replication factor.

Hadoop HDFS Features

In the HDFS tutorial, let us now see various features of Hadoop Distributed File System

a**. Distributed Storage**

HDFS stores data in a distributed manner. It divides the data into small pieces and stores it on different DataNodes in the cluster. In this manner, the Hadoop Distributed File System provides a way to MapReduce to process a subset of large data sets broken into blocks, parallelly on several nodes. MapReduce is the heart of Hadoop, but HDFS is the one who provides it all these capabilities.

b**. Blocks**

HDFS splits huge files into small chunks known as blocks. Block is the smallest unit of data in a filesystem. We (client and admin) do not have any control on the block like block location. NameNode decides all such things. HDFS default block size is 128 MB. We can increase or decrease the block size as per our need. This is unlike the OS filesystem, where the block size is 4 KB.

If the data size is less than the block size of HDFS, then block size will be equal to the data size.For example, if the file size is 129 MB, then 2 blocks will be created for it. One block will be of default size 128 MB, and the other will be 1 MB only and not 128 MB as it will waste the space (here block size is equal to data size). Hadoop is intelligent enough not to waste the rest of 127 MB. So it is allocating 1 MB block only for 1 MB data. The major advantage of storing data in such block size is

that it saves disk seek time and another advantage is in the case of processing as mapper processes 1 block at a time. So 1 mapper processes large data at a time.

c. Replication

Hadoop HDFS creates duplicate copies of each block. This is known as replication. All blocks are replicated and stored on different DataNodes across the cluster. It tries to put at least 1 replica in a different rack.

DataNodes are arranged in racks. All the nodes in a rack are connected by a single switch, so if a switch or complete rack is down, data can be accessed from another rack. We will see it further in the rack awareness section.

As seen earlier in this Hadoop HDFS tutorial, the default replication factor is 3, and this can be changed to the required values according to the requirement by editing the configuration files (hdfs-site.xml).

d. High Availability

Replication of data blocks and storing them on multiple nodes across the cluster provides high availability of data. As seen earlier in this Hadoop HDFS tutorial, the default replication factor is 3, and we can change it to the required values according to the requirement by editing the configuration files (hdfs-site.xml).

Learn more about high availability in Hadoop.

e. Data Reliability

As we have seen in high availability in this HDFS tutorial, data is replicated in HDFS; It is stored reliably as well. Due to replication, blocks are highly available even if some node crashes or some hardware fails. If the DataNode fails, then the block is accessible from other DataNode containing a replica of the block. Also, if the rack goes down, the block is still available on the different rack. This is how data is stored reliably in HDFS and provides fault-tolerant and high availability.

f. Fault Tolerance

HDFS provides a fault-tolerant storage layer for Hadoop and other components in the ecosystem.

HDFS works with commodity hardware (systems with average configurations) that has high chances of getting crashed at any time. Thus, to make the entire system highly fault-tolerant, HDFS replicates and stores data in different places.

g. Scalability

Scalability means expanding or contracting the cluster. We can scale Hadoop HDFS in 2 ways.

1. **Vertical Scaling:** We can add more disks on nodes of the cluster. For doing this, we need to edit the configuration files and make corresponding entries of newly added disks. Here we need to provide downtime though it is very less. So people generally prefer the second way of scaling, which is horizontal scaling.

2. **Horizontal Scaling:** Another option of scalability is of adding more nodes to the cluster on the fly without any downtime. This is known as horizontal scaling. We can add as many nodes as we want in the cluster on the fly in real-time without any downtime. This is a unique feature provided by Hadoop.

h. High throughput access to application data

Hadoop Distributed File System provides high throughput access to application data. Throughput is the amount of work done in a unit time. It describes how fast the data is getting accessed from the system, and it is usually used to measure the performance of the system. In HDFS, when we want to perform a task or an action, then the work is divided and shared among different systems. So all the systems will be executing the tasks assigned to them independently and in parallel. So the work will be completed in a very short period of time.

HADOOP HDFS OPERATIONS

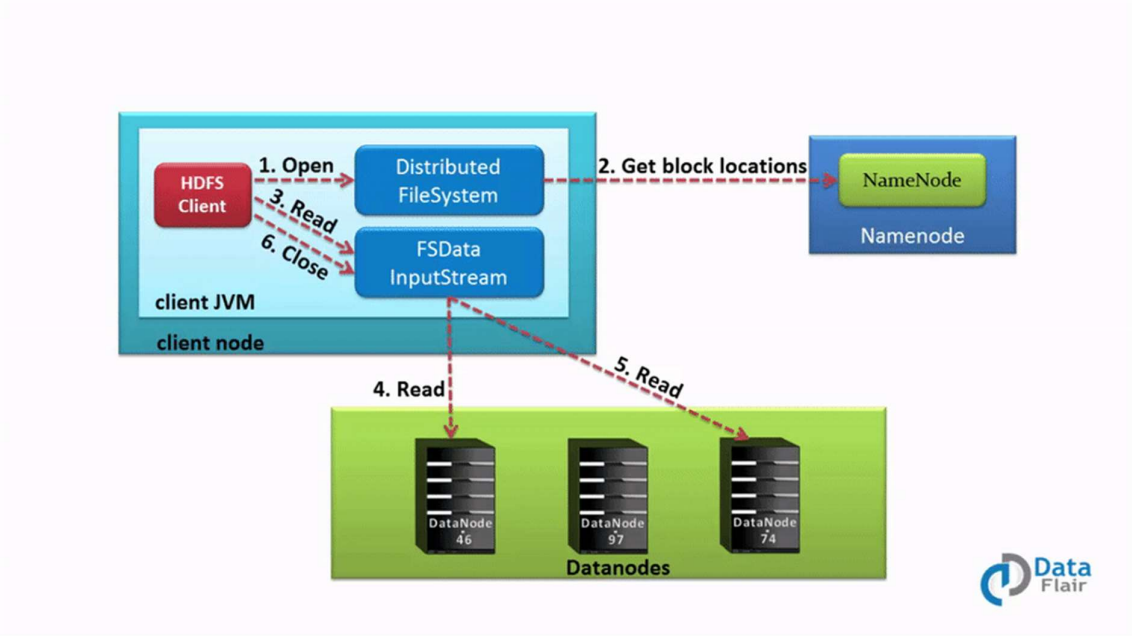
In Hadoop, we need to interact with the file system either by programming or by the command-line interface (CLI). Learn how to interact with HDFS using CLI from this commands manual. Hadoop Distributed File System has many similarities with the Linux file system. So we can do almost all the operations on the HDFS File System that we can do on a local file system like create a directory, copy the file, change permissions, etc.

It also provides different access rights like read, write, and execute to users, groups, and others.

We can browse the file system here by the browser that would be like http://master-IP:50070. By pointing the browser to this URL, you can get the cluster information like space used / available, the number of live nodes, the number of dead nodes, etc.

1. HDFS Read Operation

Whenever a client wants to read any file from HDFS, the client needs to interact with NameNode as NameNode is the only place that stores metadata about DataNodes. NameNode specifies the address or the location of the slaves where data is stored. The client will interact with the specified DataNodes and read the data from there. For security/authentication purposes, NameNode provides a token to the client, which it shows to the DataNode for reading the file.



In the Hadoop HDFS read operation, if the client wants to read data that is stored in HDFS, it needs to interact with NameNode first. So the client interacts with distributed file system API and sends a request to NameNode to send block location. Thus, NameNode checks if the client has sufficient privileges to access the data or not. If the client have sufficient privileges, then NameNode will share the address at which data is stored in the DataNode. With the address, NameNode also shares a security token with the client, which it needs to show to DataNode before accessing the data for authentication purposes. When a client goes to DataNode for reading the file, after checking the token, DataNode allows the client to read that particular block. A client then opens the input stream and starts reading data from the specified DataNodes. Hence, In this manner, the client reads data directly from DataNode. During the reading of a file, if the DataNode goes down suddenly, then a client will again go to the NameNode, and the NameNode will share another location where that block is present.

2. **HDFS Write Operation**

As seen while reading a file, the client needs to interact with NameNode. Similarly, for writing a file, the client needs to interact with the NameNode.

NameNode provides the address of the slaves on which data has to be written by the client.

Once the client finishes writing the block, the slave starts replicating the block into another slave, which then copies the block to the third slave. This is the case when the default replication factor of 3 is used. After the required replicas are created, it sends a final acknowledgment to the client. The authentication process is similar, as seen in the read section.Data Write Mechanism - HDFS Tutorial Whenever a client needs to write any data, it needs to interact with the NameNode. So the client interacts with distributed file system API and sends a request to NameNode to send a slave location. NameNode shares the location at which data has to be written. Then the client interacts with the DataNode at which data has to be written and starts writing the data through the FS data output

stream. Once the data is written and replicated, the DataNode sends an acknowledgment to the client informing that the data is written completely.

As soon as the client finishes writing the first block, the first DataNode will copy the same block to other Datanode. Thus, this Datanode after receiving the block starts copying this block to the third Datanode. Third sends an acknowledgment to second, the second DataNode sends an acknowledgment to the first DataNode, and then the first DataNode sends the final acknowledgment (in the case of default replication factor). The client is sending just 1 copy of data irrespective of our replication factor, while DataNodes replicate the blocks. Hence, writing of file in Hadoop HDFS is not costly as parallelly multiple blocks are getting written on several DataNodes.

Map Reduce

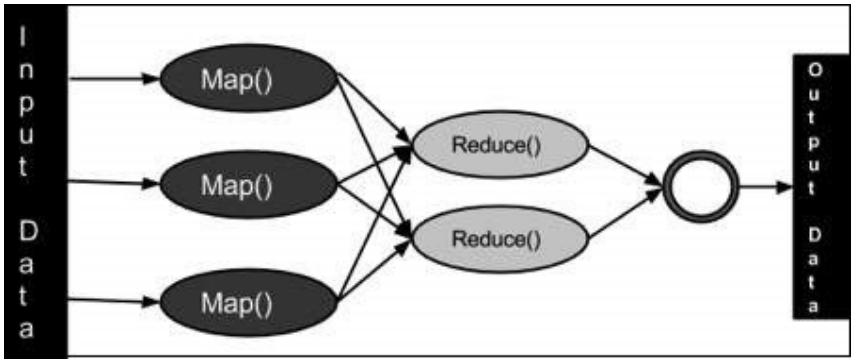
MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

- Generally, MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
    - **Map stage** − The map or mapper's job is to process the input data. Generally, the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
    - **Reduce stage** − This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a **MapReduce job** − (Input) <k1, v1> → map → <k2, v2> → reduce → <k3, v3>(Output).

	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

Terminology

- **PayLoad** − Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** − Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** − Node that manages the Hadoop Distributed File System (HDFS).
- **DataNode** − Node where data is presented in advance before any processing takes place.
- **MasterNode** − Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** − Node where Map and Reduce program runs.

- **JobTracker** − Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** − Tracks the task and reports status to JobTracker.
- **Job** − A program is an execution of a Mapper and Reducer across a dataset.
- **Task** − An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** − A particular instance of an attempt to execute a task on a SlaveNode.

Example Scenario

Given below is the data regarding the electrical consumption of an organization. It contains the monthly electrical consumption and the annual average for various years.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

If the above data is given as input, we have to write applications to process it and produce results such as finding the year of maximum usage, year of minimum usage, and so on. This is a walkover for the programmers with finite number of records. They will simply write the logic to produce the required output, and pass the data to the application written. But, think of the data representing the electrical consumption of all the largescale industries of a particular state, since its formation.

When we write applications to process such bulk data,

- They will take a lot of time to execute.
- There will be a heavy network traffic when we move data from source to network server and so on.

To solve these problems, we have the MapReduce framework.

Input Data

The above data is saved as **sample.txt** and given as input. The input file looks as shown below.

```
1979   23   23   2   43   24   25   26   26   26   26   25   26  25
1980   26   27   28  28   28   30   31   31   31   30   30   30  29
1981   31   32   32  32   33   34   35   36   36   34   34   34  34
1984   39   38   39  39   39   41   42   43   40   39   38   38  40
1985   38   39   39  39   39   41   41   41   00   40   39   39  45
```

The result of the above data

```
1981    34
1984    40
1985    45
```

MapReduce design patterns

There are four primary MapReduce design patterns:

1. Input-Map-Reduce-Output

2. Input-Map-Output

3. Input-Multiple Maps-Reduce-Output

4. Input-Map-Combiner-Reduce-Output

1. Input-Map-Reduce-Output

If we want to perform an aggregation operation, this pattern is used:

Scenario	Counting the total salary of employees based on gender.
Map (Key, Value)	Key : Gender Value : Their Salary
Reduce	Group by Key (Gender), then take the sum of the value (salary) for each group.

To count the total salary by gender, we need to make the key Gender and the value Salary. The output for the Map function is:

```
(Male, 80), (Female, 75), (Male, 100), (Male, 40), (Male, 250),
(Male, 55), (Female, 90), (Female, 300), (Female, 40)
```

Intermediate splitting gives the input for the Reduce function:

```
(Male <80, 100, 40, 250, 55>), (Female<75, 90, 300, 40>)
```

And the Reduce function output is:

```
(Male, 525), (Female, 505)
```

2. **Input-Map-Output**

The Reduce function is mostly used for aggregation and calculation. However, if we only want to change the format of the data, then the Input-Map-Output pattern is used:

Scenario	The data source has inconsistent entries for Gender, such as "Female", "F", "f", or 0. We want to make the column consistent, i.e., for male, "M" should be used, and for female, "F" should be used.
Map (Key, Value)	Key : Employee ID Value : Gender -> If Gender is Female/female/ F/ f/ 0, then return F; else if Gender is Male/male/M/m/1, then return M.

3. **Input-Multiple Maps-Reduce**

In the Input-Multiple Maps-Reduce-Output design pattern, our input is taken from two files, each of which has a different schema. (Note that if two or more files have the same schema, then there is no need for two mappers. We can simply write the same logic in one mapper class and provide multiple input files.)

Scenario 1	We have to find the total salary by gender. But we have 2 files with different schemas. **Input File 1** Gender is given as a prefix to the name. *E.g. Ms. Shital Katkar* *Mr. Krishna Katkar* **Input File 2** There is a separate, dedicated column for gender. However, the format is inconsistent. *E.g. Female/Male, 0/1, F/M*
Map **(Key, Value)**	**Map 1 (for input 1)** We need to write program to split each prefix from its name and determine the gender according to the prefix. Then we prepare the key-value pair (Gender, Salary). **Map 2 (for input 2)** Here, the program will be straightforward. Resolve the mixed formats (as seen earlier) and make a key-value pair (Gender, Salary).
Reduce	Group by Gender and take the total salary for each group.

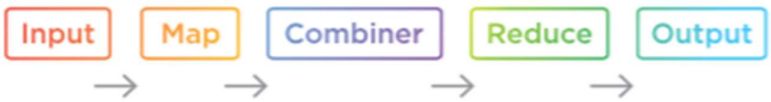
This pattern is also used in Reduce-Side Join:

<table>
<tr><td>Scenario 2</td><td>There are two large files, 'Books and 'Rating'.
<table><tr><th>Book ID</th><th>Book Name</th></tr><tr><td>101</td><td>Classical Mythology</td></tr><tr><td>102</td><td>The Kitchen God's Wife</td></tr><tr><td>103</td><td>Haveli</td></tr></table>
<table><tr><th>Book ID</th><th>Rating</th><th>User ID</th></tr><tr><td>101</td><td>8</td><td>X134</td></tr><tr><td>102</td><td>7</td><td>D454</td></tr><tr><td>101</td><td>7</td><td>C455</td></tr><tr><td>103</td><td>6</td><td>B455</td></tr></table>
We want to display each book and its average rating.</td></tr>
<tr><td>Map
(Key, Value)</td><td>In this case, we have to join two tables, Books and Rating, on the Book ID column, so we open the two files in two different Map functions.

<u>Map 1 (for the Books table)</u>
Since we want to join on Book ID, it should be sent as the key in both maps.
Key-value pair: (Book ID, Book Name)

<u>Map 2 (for the Rating table)</u>
Key-value pair: (Book ID, Rating)</td></tr>
<tr><td>Splitting
Output</td><td>(101< Classical Mythology, 8, 7, 9, 6,...,...,...,...,...>)
(102<The Kitchen God's Wife,7,8,5,6,.....>)
(103<Haveli,6,7,5,6,.....>)</td></tr>
<tr><td>Reduce</td><td>Group By Book ID, take the average of Rating, and return (Book Name, Average Rating).
(Classical Mythology, 7.5)
(The Kitchen God's Wife, 6.5)
(Haveli, 6)</td></tr>
</table>

4. Input-Map-Combiner-Reduce-Output



Apache Spark is highly effective for big and small data processing tasks not because it best reinvents the wheel, but because it best amplifies the existing tools needed to perform effective analysis. Coupled with its highly scalable nature on commodity grade hardware, and incredible performance capabilities compared to other well-known Big Data processing engines, Spark may finally let software finish eating the world.

A Combiner, also known as a semi-reducer, is an optional class that operates by accepting the inputs from the Map class and then passing the output key-value pairs to the Reducer class. The purpose of the Combiner function is to reduce the workload of Reducer.

In a MapReduce program, 20% of the work is done in the Map stage, which is also known as the data preparation stage. This stage does work in parallel. 80% of the work is done in the Reduce stage, which is known as the calculation stage. This work is not done in parallel, so it is slower than the Map phase. To reduce computation time, some work of the Reduce phase can be done in a Combiner phase.

Example

There are five departments, and we have to calculate the total salary by department, then by gender. However, there are additional rules for calculating those totals. After calculating the total for each department by gender:

- If the total department salary is greater than 200K, add 25K to the total.
- If the total department salary is greater than 100K, add 10K to the total.

Input Files (for each dept.)	**Map** (Parallel) (Key = Gender, Value = Salary)	**Combiner** (Parallel)	**Reducer** (Not parallel)	**Output**
Dept. 1	Male <10,20,25,45,15,45,25,20> Female <10,30,20,25,35>	Male <250,25> Female <120,10>	Male < 250,25,155, 10,90,90,30> Female <120,10,175,10,135, 10,110,10,130,10>	Male <650> Female <720>
Dept. 2	Male<15,30,40,25,45> Female <20,35,25,35,40>	Male <155,10> Female <175,10>		
Dept. 3	Male<10,20,20,40> Female <10,30,25,70>	Male <90,00> Female <135,10>		
Dept. 4	Male<45,25,20> Female <30,20,25,35>	Male <90,00> Female <110,10>		
Dept. 5	Male<10,20> Female <10,30,20,25,35>	Male <30,00> Female <130,10>		

PART A

Q.No.	QUESTION
6.	Analyze the differences between RDBMS and Hadoop?
7.	Explain Map reduce with example
8.	Outline the main components of Map reduce job?
9.	Relate namenode and datanode in Hadoop?
10.	Explore the basic characteristics of Hadoop?

PART B

Q.No.	QUESTION
6.	Illustrate the YARN architecture.
7.	With neat block diagram explain Mapreduce
8.	Explain all the daemons of hadoop.
9.	Figure out how HDFS is able to solve big data problem.
10.	Investigate the read - write anatomy with neat diagram.

UNIT – IV - INTRODUCTION TO MODERN DATABASES– SECA7018

Introduction to Modern databases-No SQL, New SQL, No SQLVs RDBMS databases Tradeoffs, Working with MongoDb, Data warehouse system for Hadoop.

NO SQL

NoSQL can be defined as a database which is employed for managing the massive collection of unstructured data and when your data is not piled up in a tabular format or relations like that of relational databases. The term NoSQL came from the word non SQL or nonrelational. There are a wide variety of existing Relational Databases that have been unsuccessful in solving several complex modern problems such as:

- A dynamic change in the nature of data - i.e., nowadays data are in structured, semi-structured, nonstructured as well as polymorphic in type.
- The variety of applications and the type of data feed into them for analysis has now become more diverse and distributed and is approaching cloud-oriented.
- Also, modern applications and services are serving tens of thousands of users in diverse geo-locations, having diverse time zones. So data integrity needs to be there at all the time.

Data residing in multiple virtual servers and other cloud storage (remote-based) in the cloud infrastructure can be easily analyzed using the NoSQL database management techniques and largely when the data set is in a non-structured manner. So, it can be said that the NoSQL database is intended to overcome the diversity of data, increase performance, modeling of data, scalability, and distribution, which is usually encountered in the Relational Databases.

A **NoSQL database** includes simplicity of design, simpler horizontal scaling to clusters of machines and finer control over availability. The data structures used by **NoSQL databases** are different from those used by default in relational **databases** which makes some operations faster in **NoSQL**.

When should NoSQL be used:

1. When huge amount of data need to be stored and retrieved .
2. The relationship between the data you store is not that important
3. The data changing over time and is not structured.
4. Support of Constraints and Joins is not required at database level
5. The data is growing continuously and you need to scale the database regular to handle the data.

ADVANTAGES OF NOSQL:

There are many advantages of working with NoSQL databases such as MongoDB and Cassandra. The main advantages are high scalability and high availability.

1. **High scalability –**

NoSQL database use sharding for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharding. Vertical scaling means adding more resources to the existing machine whereas horizontal scaling means adding more machines to handle the data. Vertical scaling is not that easy to implement but horizontal scaling is easy to implement. Examples of horizontal scaling databases are MongoDB, Cassandra etc. NoSQL can handle huge amount of data because of scalability, as the data grows NoSQL scale itself to handle that data in efficient manner.

2. **High availability –**

   Auto replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.

DISADVANTAGES OF NOSQL:

NoSQL has the following disadvantages.

1. **Narrow focus –** NoSQL databases have very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.

2. **Open-source–**NoSQL is open-source database. There is no reliable standard for NoSQL yet. In other words two database systems are likely to be unequal.

3. **Management challenge –**The purpose of big data tools is to make management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.

4. **GUI is not available –**GUI mode tools to access the database is not flexibly available in the market.

5. **Backup –**Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.

6. **Large document size –**Some database systems like MongoDB and CouchDB store data in JSON format. Which means that documents are quite large (BigData, network bandwidth, speed), and having descriptive key names actually hurts, since they increase the document size.

TYPES OF NOSQL DATABASE:

Types of NoSQL databases and the name of the databases system that falls in that category are:

1. **MongoDB** falls in the category of NoSQL document based database.

2. **Key value store:** Memcached, Redis, Coherence
3. **Tabular:** Hbase, Big Table, Accumulo
4. **Document based:** MongoDB, CouchDB, Cloudant

- **Document databases** store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects, and their structures typically align with objects developers are working with in code. Because of their variety of field value types and powerful query languages, document databases are great for a wide variety of use cases and can be used as a general purpose database. They can horizontally scale-out to accomodate large data volumes. MongoDB is consistently ranked as the world's most popular NoSQL database according to DB-engines and is an example of a document database.

- **Key-value databases** are a simpler type of database where each item contains keys and values. A value can typically only be retrieved by referencing its value, so learning how to query for a specific key-value pair is typically simple. Key-value databases are great for use cases where you need to store large amounts of data but you don't need to perform complex queries to retrieve it. Common use cases include storing user preferences or caching. Redis and DynanoDB are popular key-value databases.

- **Wide-column stores** store data in tables, rows, and dynamic columns. Wide-column stores provide a lot of flexibility over relational databases because each row is not required to have the same columns. Many consider wide-column stores to be two-dimensional key-value databases. Wide-column stores are great for when you need to store large amounts of data and you can predict what your query patterns will be. Wide-column stores are commonly used for storing Internet of Things data and user profile data. Cassandra and HBase are two of the most popular wide-column stores.

- **Graph databases** store data in nodes and edges. Nodes typically store information about people, places, and things while edges store information about the relationships between the nodes. Graph databases excel in use cases where you need to traverse relationships to look for patterns such as social networks, fraud detection, and recommendation engines. Neo4j and JanusGraph are examples of graph databases.

BENEFITS:

- It allows developers to create large volumes of structured, semi-structured as well as unstructured data for making the application diverse and not restricting its use because of the type of data being used within the application.

- It also allows agile development; rapid iteration along with frequent code pushes, which makes it more popular.
- It can be used with object-oriented programming (OOP), which makes it easy to use with flexibility.
- Data can be stored more efficiently, making it less expensive, providing massive architecture.

NEW SQL:

NewSQL is a set of **new SQL databases** engines with high-performance and scalability. The term **NewSQL** was first used by analyst Matthew Aslett in 2011 in this "NoSQL, **NewSQL** and Beyond" (Aslett, 2011) business analysis report, which discussed the emergence of new **databases** systems

NewSQL databases solve database problems without giving up the advantages of traditional databases. NewSQL relational database management systems provide the same scalable performance of NoSQL systems for OLTP - online transaction processing read-write workloads while maintaining the ACID guarantees of a traditional database system. The data structures used by NoSQL databases tier from those used in relational databases, and that makes some operations faster in NoSQL databases..

Top NewSQL Databases: ClustrixDB, NuoDB, CockroachDB, Pivotal GemFire XD, Altibase, MemSQL, VoltDB, c-treeACE, Percona TokuDB, Apache Trafodion, TIBCO ActiveSpaces, ActorDB are some of the Top NewSQL Databases.

NewSQL databases are relational database systems that combine the OLTP, high performance, and scalability of NoSQL. They maintain the ACID (Atomicity, Consistency, Isolation, and Durability) guarantees of traditional DBMS. ACID transactions ensure complete business processes, concurrent transactions, data survival in case of system failures or errors, and consistency before and after a transaction.

NewSQL databases tier in terms of their internal design, but all of them are RDBMSs that run on SQL. They use SQL to ingest new information, execute many transactions at the same time, and modify the contents of the database. The main categories of NewSQL systems include new architectures, transparent sharding middleware, SQL engines, and Database-asa-Service (DBaaS).

Partitioning/Sharding: Almost all NewSQL database management systems scale out by dividing the database into separate subsets known as partitions or shards. The tables are horizontally split into several fragments with boundaries based on column values. Related fragments from dierent tables are joined to create a partition.

Replication: This feature allows database users to create and maintain copies of a database or a part of a database. Copies of the database are stored in a remote site next to the main site or in a distant site. Users can update the replicas simultaneously or update one node and transfer the resultant state to other replicas.

Secondary Indexes: Secondary indexes allow database users to eciently access database records by using a dierent value other than the primary key.

Concurrency Control: This feature addresses the problems that might occur in a multiuser system when many user access or modify data simultaneously. NewSQL systems use this feature to ensure simultaneous transactions while maintaining data integrity.

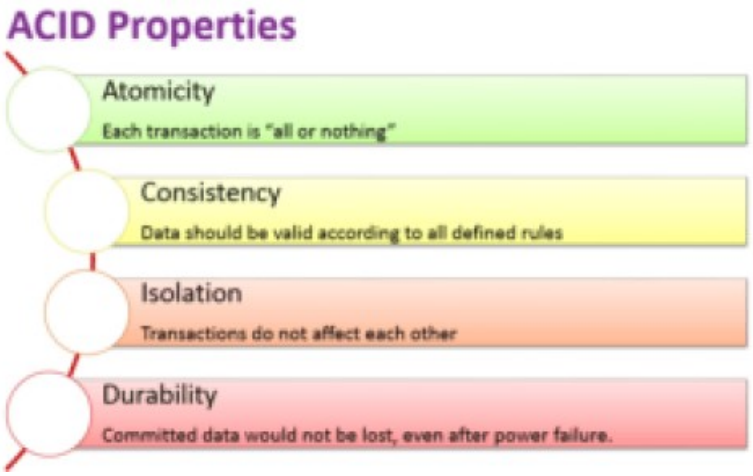
Crash Recovery: NewSQL databases have a mechanism that enables them to recover data and move to a consistent state when the system crashes.

Some of the benefits include:

Database partitioning reduces the system's communication overhead making it possible to access data with ease.

ACID transactions ensure data integrity even if there is a system failure or error. NewSQL databases can handle complex data. NewSQL systems are highly scalable.

NewSQL is a new approach to relational databases that wants to combine transactional ACID (atomicity, consistency, isolation, durability).



1. Atomicity means that a transaction must exhibit "all or nothing" behavior. Either all of the instructions within the transaction happen, or none of them happen. Atomicity preserves the "completeness" of the business process.
2. Consistency refers to the state of the data both before and after the transaction is executed. A transaction maintains the consistency of the state of the data. In other words, after a transaction is run, all data in the database is "correct."
3. Isolation means that transactions can run at the same time. Any transactions running in parallel have the illusion that there is no concurrency. In other words, it appears that the system is running only a single transaction at a time. No other concurrent transaction has visibility to the uncommitted database modifications made by any other transactions. To achieve isolation, a locking mechanism is required.

4. Durability refers to the impact of an outage or failure on a running transaction. A durable transaction will not impact the state of data if the transaction ends abnormally. The data will survive any failures.

NOSQLVS RDBMS DATABASES TRADEOFFS

RDBMS

- **Stands for Relational Database Management System**
- It is completely a structured way of storing data.
- The amount of data stored in RDBMS depends on physical memory of the system or in other words it is vertically scalable.
- In RDBMS schema represents logical view in which data is organized and tells how the relation are associates.
- It is a mixture of open and closed development models. like oracle, apache and so on.
- RDBMS databases are table based databases This means that SQL databases represent data in form of tables which consists of n number of rows of data
- RDBMS have predefined schemas.
- For defining and manipulating the data RDBMS use structured query language i.e. SQL which is very powerful.
- **RDBMS database examples:** MySql, Oracle, Sqlite, Postgres and MS-SQL.
- RDBMS database is well suited for the complex queries as compared to NoSql.
- If we talk about the type of data then RDBMS are not best fir for hierarchical data storage
- **Scalability:** RDBMS database is vertically scalable so to manage the increasing load by increase in CPU, RAM, SSD on a single server.
- RDBMS is best suited for high transactional based application and its more stable and promise for the atomicity and integrity of the data.
- RDBMS support large scale deployment and get support from there vendors.
- **Properties:** ACID properties(Atomicity, Consistency, Isolation, Durability).

NOSQL

- **Stands for Not Only SQL**
- It is completely a unstructured way of storing data.
- While in Nosql there is no limit you can scale it horizontally.
- Work on only open source development models.

- NoSQL databases are document based, key-value pairs, graph databases or wide-column stores.whereas NoSQL databases are the collection of key-value pair, documents, graph databases or wide-column stores which do not have standard schema definitions which it needs to adhered to.
- NoSql have dynamic schema with the unstructured data.
- It uses UnQL i.e. unstructured query language and focused on collection of documents and vary from database to database.
- **NoSQL database examples:** MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb
- NoSql is note well suited for complex queries on high level it dose not have standard interfaces to perform that queries.
- NoSql is best bit for hierarchical data storage because it follows the key-value pair way of data similar to JSON. Hbase is the example for the same.
- **Scalability:** as we know Nosql database is horizontally scalable so to handle the large traffic you can add few servers to support that.
- NoSql is still rely on community support and for large scale NoSql deployment only limited experts are available.
- **Properties:** Follow Brewers CAP theorem(Consistency, Availability and Partition tolerance).

RDBMS vs NoSql

Data Storage

RDBMS: Structured Data

NoSql: Unstructured Data

History

RDBMS: 1980-present Era of relational database and Database Management System (DBMS):

NoSql:

1970: NoSQL = We have no SQL
1980: NoSQL = Know SQL
2000: NoSQL = No SQL!
2005: NoSQL = Not only SQL
2013: NoSQL = No, SQL!

Schema

RDBMS: When you define the database tables using a specific database management software you must write a relation shema for each table.

NoSql: Records can add new information on the fly , unlike SQL table rows, disimilar data can be store together.

Development Model

RDBMS: Mix of open(postgresql) and closed(oracle) database

NoSql: Open Source

Data Manipulation

RDBMS: Specific languages using select,insert and update statments.eg: SELECT fields FROM table WHERE..

NoSql: Through Object-Oriented API's

Examples

RDBMS:

NoSql:

RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

RDBMS Database is a relational database. It is the standard language for relational database management systems.Data is stored in the form of rows and columns in RDBMS. The relations among tables are also stored in the form of the table SQL (Structured query Language) is a programming language used to perform tasks such as update data on a database, or to retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, etc.

FEATURES OF RDBMS

1. SQL databases are table based databases
2. Data store in rows and columns
3. Each row contains a unique instance of data for the categories defined by the columns.
4. Provide facility primary key, to uniquely identify the rows

LIMITATIONS FOR SQL DATABASE

Scalability: Users have to scale relational database on powerful servers that are expensive and difficult to handle. To scale relational database it has to be distributed on to multiple servers. Handling tables across different servers is difficult .

Complexity: In SQL server's data has to fit into tables anyhow. If your data doesn't fit into tables, then you need to design your database structure that will be complex and again difficult to handle.

NOSQL

NoSQL commonly referred to as "Not Only SQL". With NoSQL, unstructured ,schema less data can be stored in multiple collections and nodes and it does not require fixed table sachems, it supports limited join queries , and we scale it horizontally.

BENEFITS OF NOSQL

Highly And Easily Scalable

Relational database or RDBMS databases are vertically Scalable When load increase on RDBMS database then we scale database by increasing server hardware power ,need to by expensive and bigger servers and NoSQL databases are designed to expand horizontally and in Horizontal scaling means that you scale by adding more machines into your pool of resources.

Maintaining NoSQL Servers is Less Expensive

Maintaining high-end RDBMS systems is expensive and need trained manpower for database management but NoSQL databases require less management. it support many Features like automatic repair, easier data distribution, and simpler data models make administration and tuning requirements lesser in NoSQL.

Lesser Server Cost and open-Source

NoSQL databases are cheap and open source. NoSql database implementation is easy and typically uses cheap servers to manage the exploding data and transaction while RDBMS databases are expensive and it uses big servers and storage systems. So the storing and processing data cost per gigabyte in the case of NoSQL can be many times lesser than the cost of RDBMS.

No Schema or Fixed Data model

NoSQL database is schema less so Data can be inserted in a NoSQL database without any predefined schema. So the format or data model can be changed any time, without application disruption.and change management is a big headache in SQL.

Support Integrated Caching

NoSQL database support caching in system memory so it increase data output performance and SQL database where this has to be done using separate infrastructure.

Limitations & disadvantage of NoSQL

1. NoSQL database is Open Source and Open Source at its greatest strength but at the same time its greatest weakness because there are not many defined standards for NoSQL databases, so no two NoSQL databases are equal

2. No Stored Procedures in mongodb (NoSql database).

3. GUI mode tools to access the database is not flexibly available in market

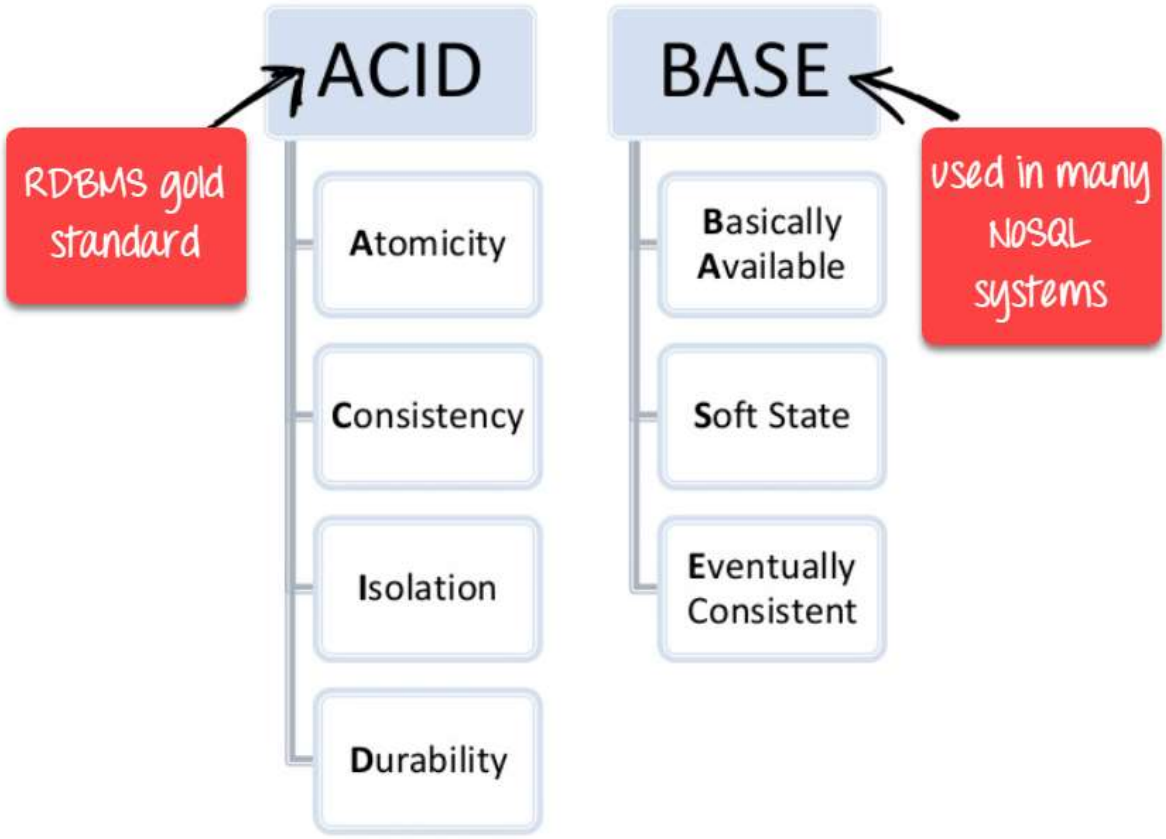
4. too difficult for finding nosql experts because it is latest technology and NoSQL developer are in learning mode

Parameter	SQL	NOSQL
Definition	SQL databases are primarily called RDBMS or Relational Databases	NoSQL databases are primarily called as Non-relational or distributed database
Design for	Traditional RDBMS uses SQL syntax and queries to analyze and get the data for further insights. They are used for OLAP systems.	NoSQL database system consists of various kind of database technologies. These databases were developed in response to the demands presented for the development of the modern application.
Query Language	Structured query language (SQL)	No declarative query language
Type	SQL databases are table based databases	NoSQL databases can be document based, key-value pairs, graph databases
Schema	SQL databases have a predefined schema	NoSQL databases use dynamic schema for unstructured data.
Ability to scale	SQL databases are vertically scalable	NoSQL databases are horizontally scalable
Examples	Oracle, Postgres, and MS-SQL.	MongoDB, Redis, , Neo4j, Cassandra, Hbase.
Best suited for	An ideal choice for the complex query intensive environment.	It is not good fit complex queries.
Hierarchical data storage	SQL databases are not suitable for hierarchical data storage.	More suitable for the hierarchical data store as it supports key-value pair method.
Variations	One type with minor variations.	Many different types which include key-value stores, document databases, and graph databases.
Development Year	It was developed in the 1970s to deal with issues with flat file storage	Developed in the late 2000s to overcome issues and limitations of SQL databases.

Open-source	A mix of open-source like Postgres & MySQL, and commercial like Oracle Database.	Open-source
Consistency	It should be configured for strong consistency.	It depends on DBMS as some offers strong consistency like MongoDB, whereas others offer only offers eventual consistency, like Cassandra.
Best Used for	RDBMS database is the right option for solving ACID problems.	NoSQL is a best used for solving data availability problems
Importance	It should be used when data validity is super important	Use when it's more important to have fast data than correct data
Best option	When you need to support dynamic queries	Use when you need to scale based on changing requirements
Hardware	Specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Network	Highly available network (Infiniband, Fabric Path, etc.)	Commodity network (Ethernet, etc.)
Storage Type	Highly Available Storage (SAN, RAID, etc.)	Commodity drives storage (standard HDDs, JBOD)
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool.
Top Companies Using	Hootsuite, CircleCI, Gauges	Airbnb, Uber, Kickstarter
Average salary	The average salary for any professional SQL Developer is $84,328 per year in the U.S.A.	The average salary for "NoSQL developer" ranges from approximately $72,174 per year
ACID vs. BASE Model	ACID(Atomicity, Consistency, Isolation, and Durability) is a standard for RDBMS	Base (Basically Available, Soft state, Eventually Consistent) is a model of many NoSQL systems

Conclusion

RDBMS and NoSQL both dbs are great in data management and both are used to keep data storage and retrieval optimized and smooth. It's hard to say which technology is better so developer take decision according requirement and situations



ACID VS BASE

MONGODB

MongoDB is a highly flexible and scalable NoSQL database management platform that is document-based, can accommodate different data models, and stores data in key-value sets. It was developed as a solution for working with large volumes of distributed data that cannot be processed effectively in relational models, which typically accommodate rows and tables. Like Hadoop, MongoDB is free and open-source.

Some Key Features of MongoDB Include:

1. It's a query language that is rich and supports text search, aggregation features, and CRUD operations.

2. It requires lesser input and output operations due to embedded data models, unlike relational databases. MongoDB indexes also support faster queries.
3. It provides fault tolerance by creating replica datasets. Replication ensures data is stored on multiple servers, creating redundancy, and ensuring high availability.
4. It features sharding, which makes horizontal scalability possible. This supports increasing data needs at a cost that is lower than vertical methods of handling system growth.
5. It employs multiple storage engines, thereby ensuring the right engine is used for the right workload, which in turn enhances performance.

The storage engines include:

- WiredTiger

  This is the default engine used in new deployments for versions 3.2 or higher. It can handle most workloads. Its features include checkpointing, compression, and document-level concurrency for write operations. The latter feature allows multiple users to use and edit documents concurrently.

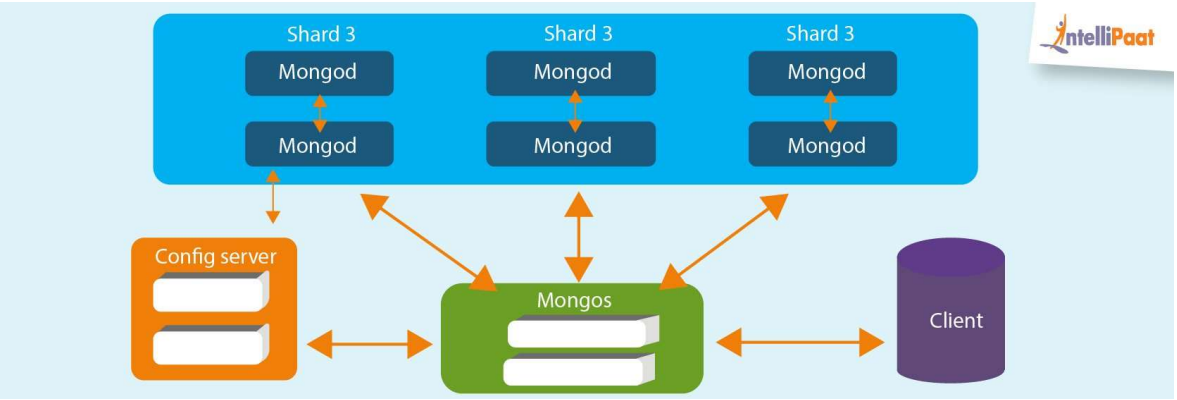
- In-Memory Storage Engine

  This engine stores documents in-memory instead of on-disk. This increases the predictability of data latencies.

- MMAPv1 Storage Engine

  This is the earliest storage for MongoDB and only works on V3.0 or earlier. It works well for workloads involving bulk in-place updates, reads, and inserts.

Architecture of MongoDB NoSQL Database



Database: In simple words, it can be called the physical container for data. Each of the databases has its own set of files on the file system with multiple databases existing on a single MongoDB server.

Collection: A group of database documents can be called a collection. The RDBMS equivalent to a collection is a table. The entire collection exists within a single database. There are no schemas when it comes to collections. Inside the collection, various documents can have varied fields, but mostly the documents within a collection are meant for the same purpose or for serving the same end goal.

Document: A set of key–value pairs can be designated as a document. Documents are associated with dynamic schemas. The benefit of having dynamic schemas is that a document in a single collection does not have to possess the same structure or fields. Also, the common fields in a collection's document can have varied types of data.

Why to Use MongoDB

1. When using relational databases, you need several tables for a construct. With Mongo's document-based model, you can represent a construct in a single entity, especially for immutable data.
2. The query language used by MongoDB supports dynamic querying.
3. The schema in MongoDB is implicit, meaning you do not have to enforce it. This makes it easier to represent inheritance in the database in addition to improving polymorphism data storage.
4. Horizontal storage makes it easy to scale.

   The feature makes MongoDB a better option than traditional RDBMS and the preferred database for processing Big Data are mentioned below:.

Flexibility – MongoDB stores data in 'Json' documents, where it provides a rich data model that flawlessly maps to native programming language types. And the dynamic schema makes it easier to evolve the data model than with a system with enforced schemas such as a RDBMS.

Power – MongoDB has lots of features like secondary indexes, dynamic queries, sorting, rich updates, upserts and easy aggregation that are available in a traditional RDBMS. This gives you functionalities similar to RDBMS and also provides added advantage of flexibility and scaling capability.

Speed – By keeping related data together in documents, queries can be much faster than in a relational database, where related data is separated into multiple tables and then needs to be joined later.

Limitations of MongoDB

While MongoDB incorporates great features to deal with many of the challenges in big data, it comes with some limitations, such as:

1. To use joins, you have to manually add code, which may cause slower execution and less-than-optimum performance.
2. Lack of joins also means that MongoDB requires a lot of memory as all files have to be mapped from disk to memory.
3. Document sizes cannot be bigger than 16MB.
4. The nesting functionality is limited and cannot exceed 100 levels.

Advantages of Mongo DB:

- **Schema-less** – This is perfect for flexible data model altering. In MongoDB, It is easy to declare, extend and alter extra fields to the data model, and optional nulled fields. Using RDBMS databases one must run scripts primarily in order to update the model. In this case it can be done through coding and no scripting is needed.
- **Clear structure of a single object** – The structure of the model is in 'Json' and the structure is clear instead of deriving it from a table structure.
- **No SQL or hibernate queries** – The good thing about MongoDB is that the operations are not complex to use (without SQL) and are key / value based. Easy expression language operators like '$gt', '$lt' can be used and practise of indexes & cursors is possible.
- **Tuning** – The level of consistency can be chosen depending on the value of the data.
- **Effortlessness scale-out** – Scale reads using replica sets and writes using sharding (auto balancing). Just start up another machine and you are good to go. Here, adding more machines distributes your work.
- **Scaling** – MongoDB also makes it easy to scale out your database. Auto-sharding lets you to scale your cluster linearly by adding more machines, making it possible to increase capacity without any downtime.
- Conversion or mapping of application objects to database objects are not needed.
- **Fast Access** – Utilizes internal memory for storing working set, allowing faster access to data.
- **Ease of use** – MongoDB concentrates on being easy to install, configure, maintain and use. For this, MongoDB provides a couple of configuration options and automatically tries to do the correct thing. This allows the user to work right away instead of spending a lot of time in fine tuning obscure database configurations.

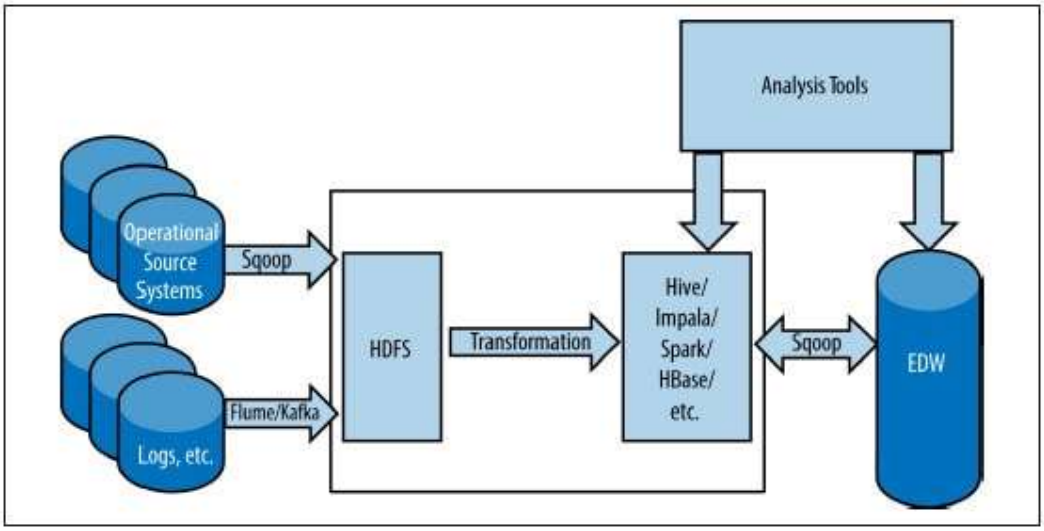
Data warehouse system for Hadoop

A data warehouse, also known as an **enterprise data warehouse (EDW)**, is a large collective store of data that is used to make such data-driven decisions, thereby becoming one of the centrepiece of an organization's data infrastructure. **Hadoop Data Warehouse** was challenge in initial days when Hadoop was evolving but now with lots of improvement, it is very easy to develop **Hadoop data warehouse Architecture**. This article will server as a guide to Hadoop data warehouse system design.

Hadoop data warehouse integration is now a days become very much popular and many companies are working on the migration tools. In this article, we will check the Hadoop data warehouse example with an architecture design.

High-Level Hadoop Data Warehouse Architecture

Below given the high-level design Hadoop data warehouse architecture:



Hadoop Data Warehouse Architecture Explanation

Extract Data From Sources

We have the operational source system such as traditional OLTP database systems. Since it is Hadoop ecosystem, you may also introduce the multi-structured data such as weblogs, machine log data, social media feeds including Facebook, twitter, linkedIn etc.

You can use Sqoop as an ingestion mechanism if you are importing data from the tradition OLTP database systems. You can use the Sqoop to export data back to OLTP systems once the processing is done in Hadoop ecosystem.

Transformation

Source data will be ingested directly into HDFS before being transformed and loaded into target systems in designated directories. Transformations will occur through one of the processing frameworks supported on Hadoop, such as MapReduce, Spa rk, Hive, Impala or Pig etc.

Load Data to Target Systems

Once transformed, data will be moved into target systems. In our example, this includes hosting the data in Hadoop via a SQL-like interface such as Hive or Impala. Additionally, you can export processed data into other data warehouse for further analysis. You can use the Sqoop export to transfer the data back to OLTP data warehouse systems. You can use sqoop to build **Hadoop data warehouse ETL process** with help of shell or Python scripting.

Additionally, you can access the data directly from the HDFS schema using various analytical tools such as BI, analytics, or visualisation tools.

Hadoop Data Warehouse Advantages

Hadoop can help to overcome some of challenges that traditional data warehouse systems are facing now:

- It can process the large volume and complex data. It can complete the ETL processing within required time constraints.
- Hadoop can process the large volume and complex data. Its distributed workload system can reduce the excessive load on the systems.
- Hadoop is flexible
- It is cheap compared to traditional data warehouse systems.

Hadoop Data Warehouse Challenges

There are few Hadoop data warehouse challenges:

- If you put data on Hadoop ecosystem that can provide an access to potentially valuable data that might otherwise never be available in traditional data warehouse ecosystem.
- Flexibility of Hadoop allows for evolving schemas and handling semi-structured and unstructured data, which enables fast turnaround time when changes to downstream reports or schemas happen.

- Using Hadoop as an online archive can free up valuable space and resources in the data warehouse, and avoid the need for expensive scaling of the data warehouse architecture.

Where you can build Hadoop Data Warehouse?

You can build **Hadoop Data warehouse in Hive or Impala**. Being MPP, Impala gives you better performance compared to Hive.

PART A

Q.No.	QUESTION
1.	Explain NoSQL? List down its vendors.
2.	Contrast RDBMS with NoSQL.
3.	Summarize the needs of data warehouse?
4.	Compare RDBMS with MongoDB.
5.	Summarize the advantages of MongoDB.

PART B

Q.No.	QUESTION
1.	Judge when and where to use MongoDB?
2.	Justify all the considerations that are made while designing Schema in MongoDB?
3.	Evaluate the concept of data warehouse.
4.	Explain the features of NoSQL and explain Vertical and horizontal scaling?
5.	Criticize column-oriented data base.

UNIT – V - PIG AND HIVE IN BIG DATA– SECA7018

Introduction to Pig and HIVE- Programming Pig: Engine for executing data flows in parallel on Hadoop, Programming with Hive.

APACHE PIG

The Apache Pig is a platform for managing large sets of data which consists of high-level programming to analyze the data. Pig also consists of the infrastructure to evaluate the programs. The advantages of pig programming is that it can easily handle parallel processes for managing very large amounts of data. The programming on this platform is basically done using the textual language Pig Latin.

Apache Pig enables people to focus more on **analyzing bulk data sets and to spend less time writing Map-Reduce programs.** Similar to Pigs, who eat anything, the Pig programming language is designed to work upon any kind of data.

Pig Latin comes with the following features:

- Simple programming: it is easy to code, execute and manage
- Better optimization: system can automatically optimize the execution
- Extensive nature: it can be used to achieve highly specific processing tasks

FEATURES OF PIG:

Pig can be used for following purposes:

- Research on raw data
- Iterative processing
- ETL (Extract, Transform, Load) Data pipeline

The scalar data types in pig are int, float, double, long, chararray, and bytearray. The complex data types in Pig are map, tuple, and bag.

Map: The data element with the data type chararray where element has pig data type include complex data type
Example- [city'#'bang','pin'#560001]

In this city and pin are data element mapping to values.

Tuple: It is a collection of data types and it has fixed length. Tuple is having multiple fields and these are ordered.

Bag: It is a collection of tuples, but it is unordered, tuples in the bag are separated by comma
Example: {('Bangalore', 560001),('Mysore',570001),('Mumbai',400001)

LOAD function:

Load function helps to load data from the file system. It is a relational operator. In the first step in data-flow language we need to mention the input, which is completed by using 'load' keyword.

The LOAD syntax is
LOAD 'mydata' [USING function] [AS schema];

Example- A = LOAD 'intellipaat.txt';

A = LOAD 'intellipaat.txt' USINGPigStorage('\t');

The relational operations in Pig:

foreach, order by, filters, group, distinct, join, limit.

foreach: It takes a set of expressions and applies them to all records in the data pipeline to the next operator.

A =LOAD 'input' as (emp_name: charrarray, emp_id : long, emp_add : chararray, phone : chararray, preferences : map []);

B = foreach A generate emp_name, emp_id;

Filters: It contains a predicate and it allows us to select which records will be retained in our data pipeline.
Syntax: alias = FILTER alias BY expression;

Alias indicates the name of the relation, By indicates required keyword and the expression has Boolean.
Example: M = FILTER N BY F5 == 4;

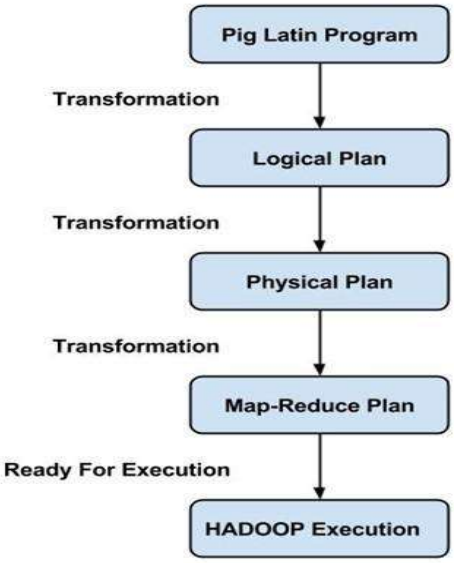
PIG ARCHITECTURE

Pig consists of two components:

1. **Pig Latin,** which is a language
2. **A runtime environment,** for running PigLatin programs.

A Pig Latin program consists of a series of operations or transformations which are applied to the input data to produce output. These operations describe a data flow which is translated into an executable representation, by Pig execution environment. Underneath, results of these transformations are series of MapReduce jobs which a programmer is unaware of. So, in a way, Pig allows the programmer to focus on data rather than the nature of execution.

PigLatin is a relatively stiffened language which uses familiar keywords from data processing e.g., Join, Group and Filter.



PIG ARCHITECTURE

Execution modes:

Pig has two execution modes:

1. Local mode: In this mode, Pig runs in a single JVM and makes use of local file system. This mode is suitable only for analysis of small datasets using Pig
2. Map Reduce mode: In this mode, queries written in Pig Latin are translated into MapReduce jobs and are run on a Hadoop cluster (cluster may be pseudo or fully distributed). MapReduce mode with the fully distributed cluster is useful of running Pig on large datasets.

HIVE:

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

HIVE IS NOT

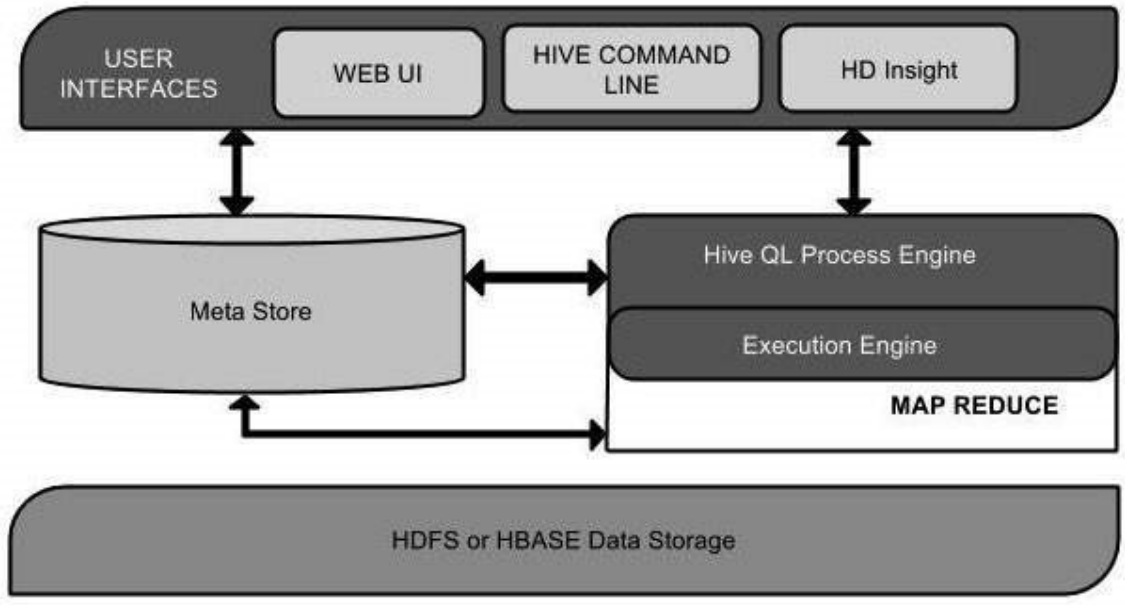
- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

FEATURES OF HIVE

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

ARCHITECTURE OF HIVE

The following component diagram depicts the architecture of Hive:



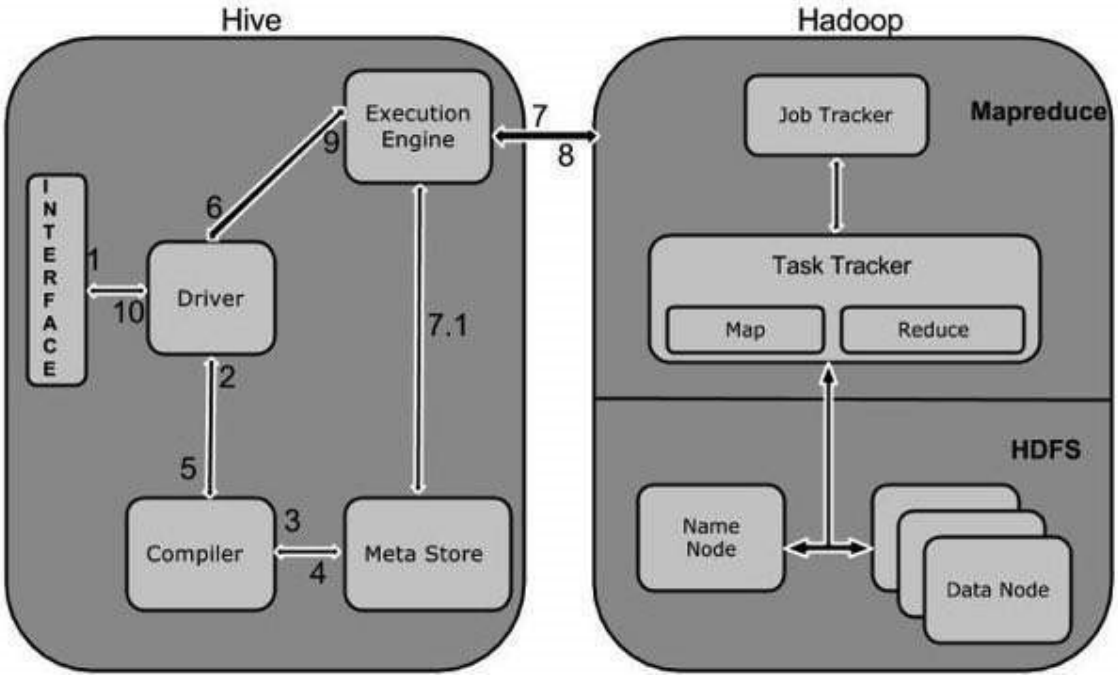
This component diagram contains different units. The following table describes each unit:

Unit Name	Operation

User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

WORKING OF HIVE

The following diagram depicts the workflow between Hive and Hadoop.



The following table defines how Hive interacts with Hadoop framework:

Step No.	Operation
1	**Execute Query** The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
2	**Get Plan** The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
3	**Get Metadata** The compiler sends metadata request to Metastore (any database).
4	**Send Metadata** Metastore sends metadata as a response to the compiler.
5	**Send Plan** The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6	**Execute Plan** The driver sends the execute plan to the execution engine.
7	**Execute Job**

	Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
7.1	**Metadata Ops** Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8	**Fetch Result** The execution engine receives the results from Data nodes.
9	**Send Results** The execution engine sends those resultant values to the driver.
10	**Send Results** The driver sends the results to Hive Interfaces.

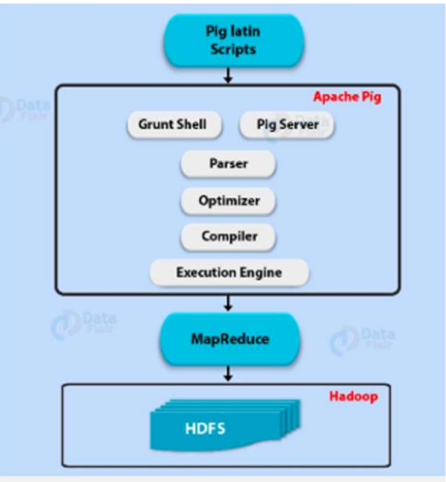
PROGRAMMING PIG: Engine for executing dataflows in parallel on Hadoop

Hadoop Pig is nothing but an abstraction over MapReduce. While it comes to analyze large sets of data, as well as to represent them as data flows, we use Apache Pig. Generally, we use it with **Hadoop**. By using Pig, we can perform all the data manipulation operations in Hadoop.

PROPERTIES:

- Ease of programming
- Optimization opportunities
- Extensibility

ARCHITECTURE:



i. Parser

At first, all the Pig Scripts are handled by the Parser. Basically, Parser checks the syntax of the script, does type checking, and other miscellaneous checks. Afterward, Parser's output will be a DAG (directed acyclic graph). That represents the Pig Latin statements as well as logical operators. Basically, the logical operators of the script are represented as the nodes and the data flows are represented as edges, in the DAG (the logical plan).

ii. Optimizer

Further, DAG is passed to the logical optimizer. That carries out the logical optimizations. Like projection and push down.

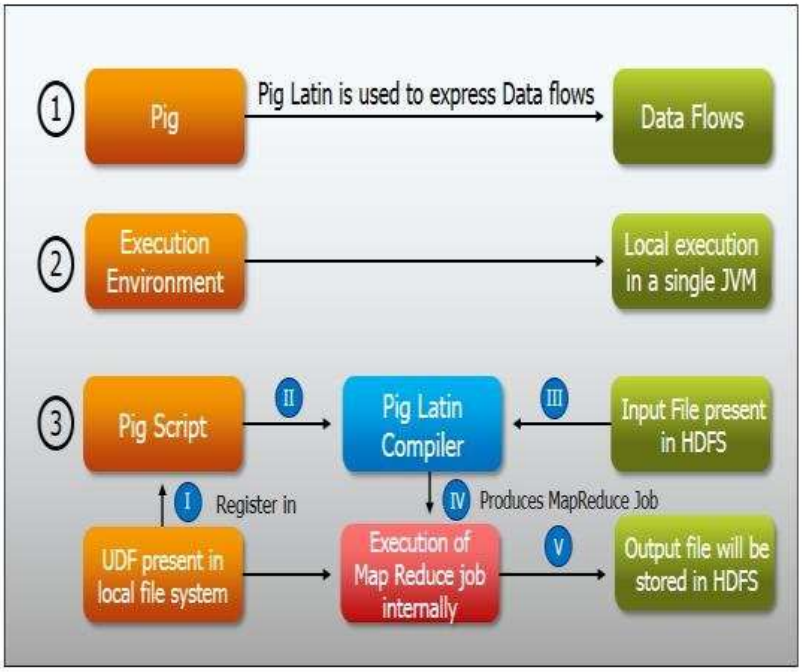
iii. Compiler

It compiles the optimized logical plan into a series of MapReduce jobs.

iv. Execution Engine

At last, MapReduce jobs are submitted to Hadoop in a sorted order. Hence, these MapReduce jobs are executed finally on Hadoop, that produces the desired results

DATA FLOW FRAMEWORK ON HADOOP:



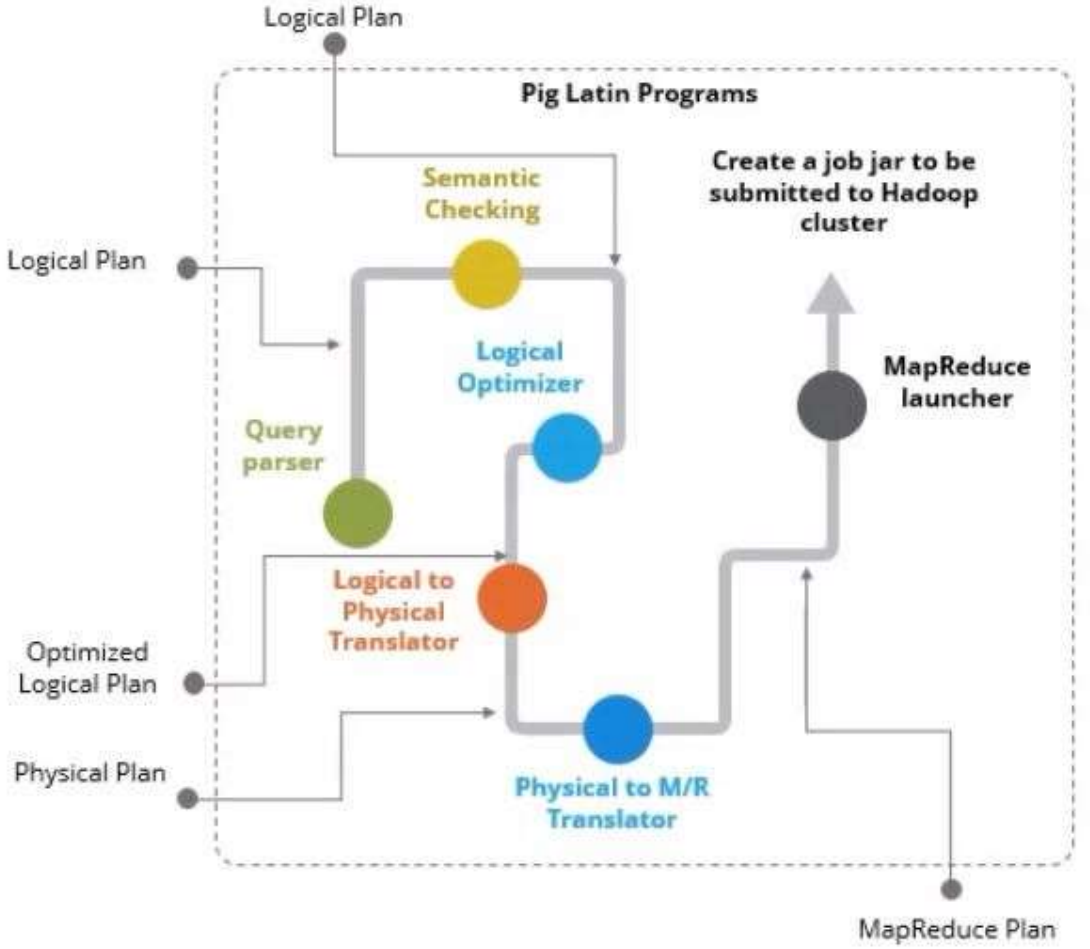
PIG SCRIPT INTERPRETATION

Pig processes Pig Latin statements in the following manner:

- Pig validates the syntax and semantics of all statements.
- It type checks with the schema.
- It verifies references. Pig performs limited optimization before execution.

- If Pig encounters a DUMP or STORE, it will execute the statements.

A Pig Latin script execution plan consists of logical, optimized logical, physical, and MapReduce plans as shown in the below diagram.



In the next section of this Pig Tutorial, we will learn some of the relations that Big Data and Hadoop Developers execute.

Various Relations Performed by Developers

Some of the relations performed by Big Data and Hadoop Developers are:

- Filtering: Filtering refers to filtering of data based on a conditional clause, such as grade and pay.
- Transforming: Transforming refers to making data presentable to extract logical data.
- Grouping:Grouping refers to generating a group of meaningful data.
- Sorting:Sorting refers to arranging the data in ascending or descending order.
- Combining: Combining refers to performing a union operation of data stored in the variable.
- Splitting: Splitting refers to separating the data with a logical meaning.

In next the section of this Pig tutorial, we will see some Pig commands which are frequently used by analysts.

PIG COMMANDS

Given below in table are some frequently used Pig Commands.

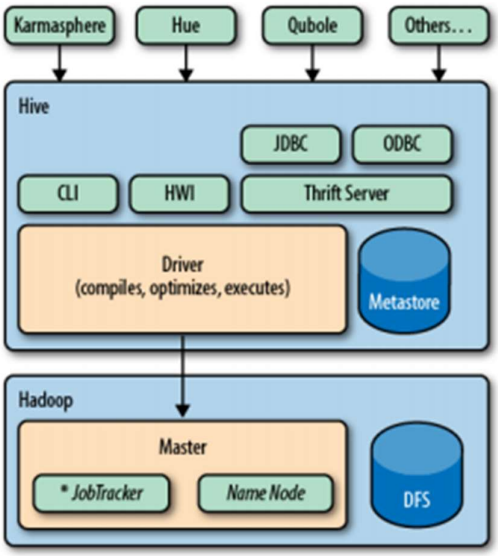
Command	Function
load	Reads data from system
Store	Writes data to file system
foreach	Applies expressions to each record and outputs one or more records
filter	Applies predicate and removes records that do not return true
Group/cogroup	Collects records with the same key from one or more inputs
join	Joins two or more inputs based on a key
order	Sorts records based on a key
distinct	Removes duplicate records
union	Merges data sets
split	Splits data into two or more sets based on filter conditions
stream	Sends all records through a user-provided binary
dump	Writes output to stdout
limit	Limits the number of records

PROGRAMMING WITHH HIVE:

HIVE HADOOP ECOSYSTEM:

- Use Hive to create, alter, and drop databases, tables, views, functions, and indexes
- Customize data formats and storage options, from files to external databases
- Load and extract data from tables—and use queries, grouping, filtering, joining, and other conventional query methods
- Gain best practices for creating user defined functions (UDFs)
- Learn Hive patterns you should use and anti-patterns you should avoid
- Integrate Hive with other data processing programs
- Use storage handlers for NoSQL databases and other datastores
- Learn the pros and cons of running Hive on Amazon’s Elastic MapReduce

HIVE HYBRID ECOSYSTEM:



Bundled with the Hive distribution is the Command Line Interface (CLI), a simple web interface called Hive web interface (HWI), and programmatic access through JDBC, ODBC, and a Thrift server. All commands and queries go to the Driver, which compiles the input, optimizes the computation required, and executes the required steps, usually with MapReduce jobs. When MapReduce jobs are required, Hive doesn't generate Java MapReduce programs. Instead, it uses built-in, generic Mapper and Reducer modules that are driven by an XML file representing the "job plan." In other words, these generic modules function like mini language interpreters and the "language" to drive the computation is encoded in XML.

Hive communicates with the JobTracker to initiate the MapReduce job. Hive does not have to be running on the same master node with the JobTracker. In larger clusters, it's common to have edge nodes where tools like Hive run. They communicate remotely with the JobTracker on the master node to execute jobs. Usually, the data files to be processed are in HDFS, which is managed by the NameNode. The Metastore is a separate relational database (usually a MySQL instance) where Hive persists table schemas and other system metadata. It is worth mentioning other higher-level tools that you should consider for your needs. Hive is best suited for data warehouse applications, where real-time responsiveness to queries and record-level inserts, updates, and deletes.

PART A

Q.No.	QUESTION
1.	Compare Apache Pig with map reduce.
2.	Compare Apache Pig with SQL.

3.	Compare Apache Pig with Apache Hive.
4.	Outline the data types in Apache Pig? Explain with examples.
5.	Compare Apache Hive with RDBMS

PART B

Q.No.	QUESTION
1.	Create the architecture of Pig with neat diagram.
2.	Develop the architecture of Hive with neat diagram.
3.	Explain relational operators in Apache Pig
4.	Explain data types of Apache Hive with examples.
5.	What are managed tables and external tables in Apache Hive?

TEXT / REFERENCE BOOKS

1. Dean Wampler, Jason Rutherglen, Edward Capriolo,"Programming Hive" O'Reilly Media, 1st Edition, 2012.
2. Sawant, Nitin, Shah, Himanshu, "Big Data Application Architecture Q & A: A Problem-Solution Approach", Apress, 1st Edition, 2013.
3. Tom White, "Hadoop the Definitive Guide", Yahoo Press, 3st Edition, 2012.
4. Alex Holmes, "Hadoop In Practice ", Manning, 1st Edition, 2012.
5. JasonVenner, "Pro Hadoop", Apress, 1st Edition, 2009.
6. Donald Miner, Zach Radtka, "Hadoop with python", O'Reilly Media, 1st Edition, 2015.
7. David Loshin, "Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph", Morgan Kaufmann, 1st Edition, 2013.
8. Jonathan R. Owens, Jon Lentz, Brian Femiano, "Hadoop Real, World Solutions Cookbook", Packt, 2nd Edition, 2016.