Unit – I

Introduction to Big Data: Data, Types of Data, Big Data – 3 Vs of Big Data, Analytics, Types of Analytics, Need for Big Data Analytics. Introduction to Apache Hadoop: Invention of Hadoop, Hadoop Architecture, Hadoop Components, Hadoop Eco Systems, Hadoop Distributions, Benefits of Hadoop

Data: Data is the collection of raw facts and figures. Actually data is unprocessed, that is why data is called collection of raw facts and figures. We collect data from different resources. After collection, data is entered into a machine for processing. Data may be collection of words, numbers, pictures, or sounds etc.

Examples of Data:

- *Student data on admission form*- bundle of admission forms contains name, father's name, address, photograph etc.
- *Student's examination data* - In examination system of a college/school, data about obtained marks of different subjects for all students is collected, exam schedule etc.
- *Census Report, Data of citizens*- During census, data of all citizens like number of persons living in a home, literate or illiterate, number of children, cast, religion etc.
- *Survey Data* – data can be collected by survey to know the opinion of people about their product like / unlike their products. They also collect data about their competitor companies in a particular area.

Information: Processed data is called information. When raw facts and figures are processed and arranged in some proper order then they become information. Information has proper meanings. Information is useful in decision-making. In other words, Information is data that has been processed in such a way as to be meaningful values to the person who receives it.

Examples of information:

- *Student'saddress labels*- Stored data of students can be used to print address labels of students. These address labels are used to send any intimation / information to students at their home addresses.
- *Student's examination, Results*- In examination system collected data (obtained marks in each subject) is processed to get total obtained marks of a student. Total obtained marks are Information. It is also used to prepare result card of a student.
- *Census Report, Total Population*- Census data is used to get report/information about total population of a country and literacy rate, total population of males, females, children, aged persons, persons in different categories line cast, religion, age groups etc.
- *Survey Report* – Survey data is summarized into reports/information to present to management of the company. The management will take important decisions on the basis of data collected through surveys.

Ex: The **data** collected is in a survey report is: ***'HYD20M'***

If we process the above data we understand that code is **information** about a person as follows: HYD is city name 'Hyderabad', 20 is age and M is to represent 'MALE'.

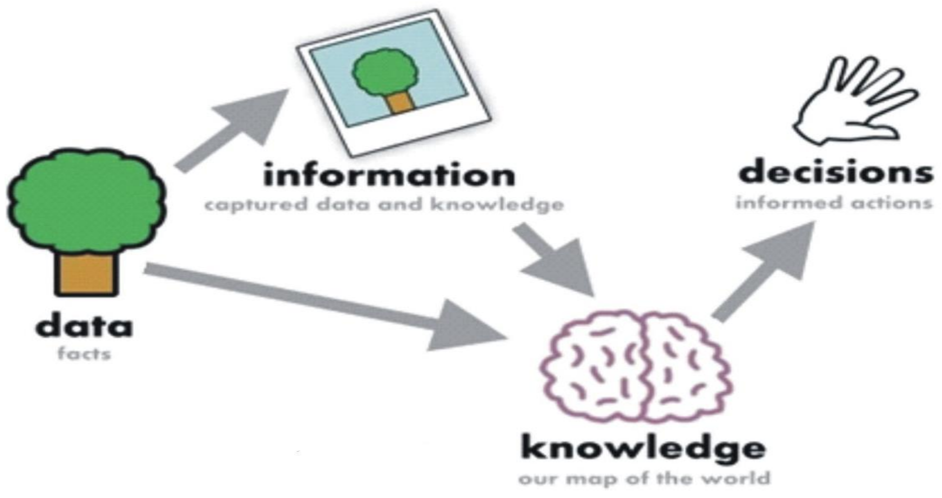


Fig: Data, information, Knowledge

Units of data: When dealing with big data, we consider numbers to represent like megabytes, gigabytes, terabytes etc. Here is the system of units to represent data.

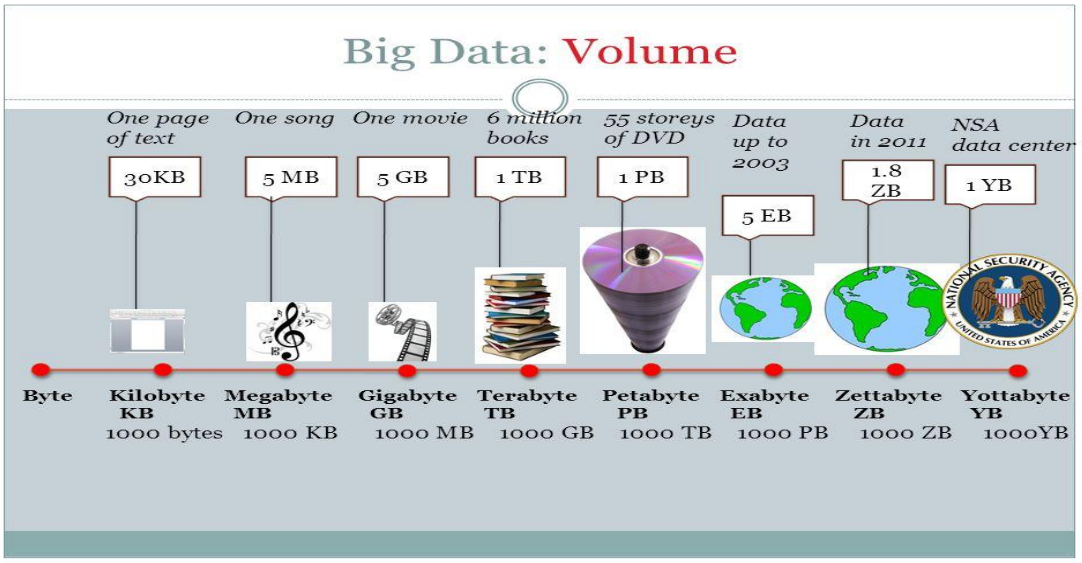
International system of Units (SI)		
Kilobyte	KB	10^3
Megabyte	MB	10^6
Gigabyte	GB	10^9
Terabyte	TB	10^{12}
Petabyte	PB	10^{15}
Exabyte	EB	10^{18}
Zettabyte	ZB	10^{21}
Yottabyte	YB	10^{24}

What is Big Data?

Big Data:

- Big Data is a term used for a collection of data sets that are large and complex, which is difficult to store and process using available database management tools or traditional data processing applications. The quantity of data on planet earth is growing exponentially for many reasons. Various sources and our day to day activities generate lots of data. With the smart objects going online, the data growth rate has increased rapidly.

- The definition of Big Data, given by Gartner is, "Big data is high-volume, and high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation".
- The major sources of Big Data: social media sites, sensor networks, digital images/videos, cell phones, purchase transaction records, web logs, medical records, archives, military surveillance, ecommerce, complex scientific research and so on.
- By 2020, the data volumes will be around 40 Zettabytes which is equivalent to adding every single grain of sand on the planet multiplied by seventy-five.
- ***Examples of Bigdata***:
    1. The **New York Stock Exchange** generates about *one terabyte* of new trade data per day.
    2. **Social Media Impact:**Statistic shows that *500+terabytes* of new data gets ingested into the databases of social media site **Facebook**, every day. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc.
    3. Walmart handles more than **1 million** customer transactions every hour.
    4. Facebook stores, accesses, and analyzes **30+ Petabytes** of user generated data.
    5. **230+ millions** of tweets are created every day.
    6. More than **5 billion** people are calling, texting, tweeting and browsing on mobile phones worldwide.
    7. YouTube users upload **48 hours** of new video every minute of the day.
    8. Amazon handles **15 million** customer click stream user data per day to recommend products.
    9. **294 billion** emails are sent every day. Services analyses this data to find the spams.
    10. Single **Jet engine** can generate *10+ terabytes* of data in *30 minutes* of a flight time. With many thousand flights per day, generation of data reaches up to many *Petabytes*.



Types of Big Data:

Big data' could be found in three forms:

1. **Structured**
2. **Unstructured**
3. **Semi-structured**

1. Structured Data:

Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data. Over the period of time, talent in computer science have achieved greater success in developing techniques for working with such kind of data (where the format is well known in advance) and also deriving value out of it. However, now days, we are foreseeing issues when size of such data grows to a huge extent, typical sizes are being in the rage of multiple zettabyte.

Eg: Data stored in a relational database management system is one example of a **'structured'** data.

'Employee' table in a database is an example of Structured Data

Employee_ID	**Employee_Name**	**Gender**	**Department**	**Salary_In_lacs**
2365	Rajesh Kulkarni	Male	Finance	650000
3398	Pratibha Joshi	Female	Admin	650000
7465	Shushil Roy	Male	Admin	500000
7500	Shubhojit Das	Male	Finance	500000
7699	Priya Sane	Female	Finance	550000

2. Unstructured Data:

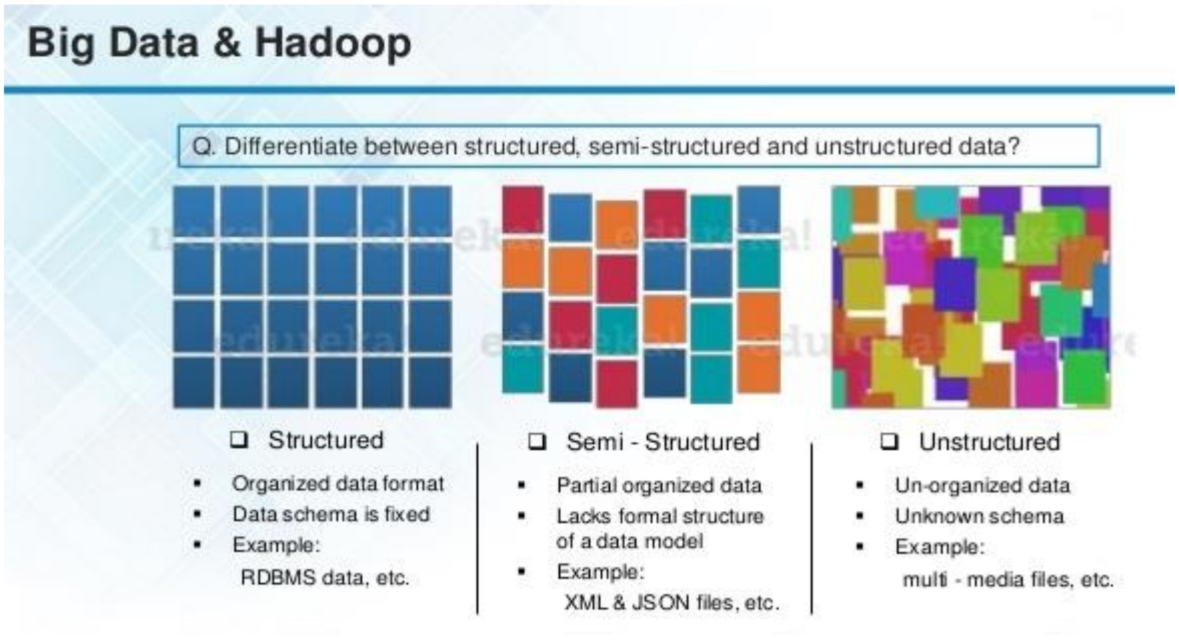
Any data with unknown form or the structure is classified as unstructured data. In addition to the size being huge, un-structured data poses multiple challenges in terms of its processing for deriving value out of it. Typical example of unstructured data is, a heterogeneous data source containing a combination of simple text files, images, videos etc. Now a day organizations have wealth of data available with them but unfortunately they don't know how to derive value out of it since this data is in its raw form or unstructured format.

Examples of Un-structured Data: Output returned by 'Google Search'

3. **Semi-Structured Data:**

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. Example of semi-structured data is a data represented in XML file.

Examples of Semi-Structured Data: XML files or JSON documents are examples of semi-structured data.



Big Data & Hadoop

Q. Differentiate between structured, semi-structured and unstructured data?

❑ Structured
- Organized data format
- Data schema is fixed
- Example: RDBMS data, etc.

❑ Semi - Structured
- Partial organized data
- Lacks formal structure of a data model
- Example: XML & JSON files, etc.

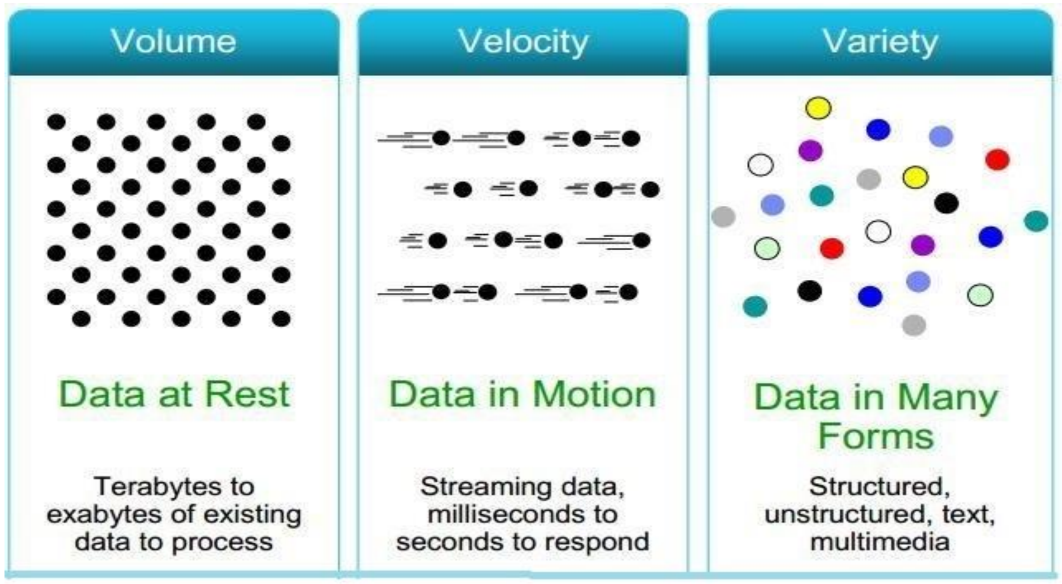
❑ Unstructured
- Un-organized data
- Unknown schema
- Example: multi - media files, etc.

3 V's of Big Data: Three V's of Big Data/ **Characteristics of 'Big Data'** are as follows:

1. Volume – Volume refers to the 'amount of data', which is growing day by day at a very fast pace. The size of data generated by humans, machines and their interactions on social media itself is massive. Size of data plays very crucial role in determining value out of data. Also, whether a particular data can actually be considered as a Big Data or not, is dependent upon volume of data. Hence, **'Volume'** is one characteristic which needs to be considered while dealing with 'Big Data'. Researchers have predicted that 40 Zettabytes (40,000 Exabytes) will be generated by 2020, which is an increase of 300 times from 2005.

2. 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. Now days, data in the form of emails, photos, videos, monitoring devices, PDFs, audio, etc. is also being considered in the analysis applications. This variety of unstructured data poses certain issues for storage, mining and analyzing data.

3. 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. If you are able to handle the velocity, you will be able to generate insights and take decisions based on real-time data.


Volume
Data at Rest
Terabytes to exabytes of existing data to process
Velocity
Data in Motion
Streaming data, milliseconds to seconds to respond
Variety
Data in Many Forms
Structured, unstructured, text, multimedia

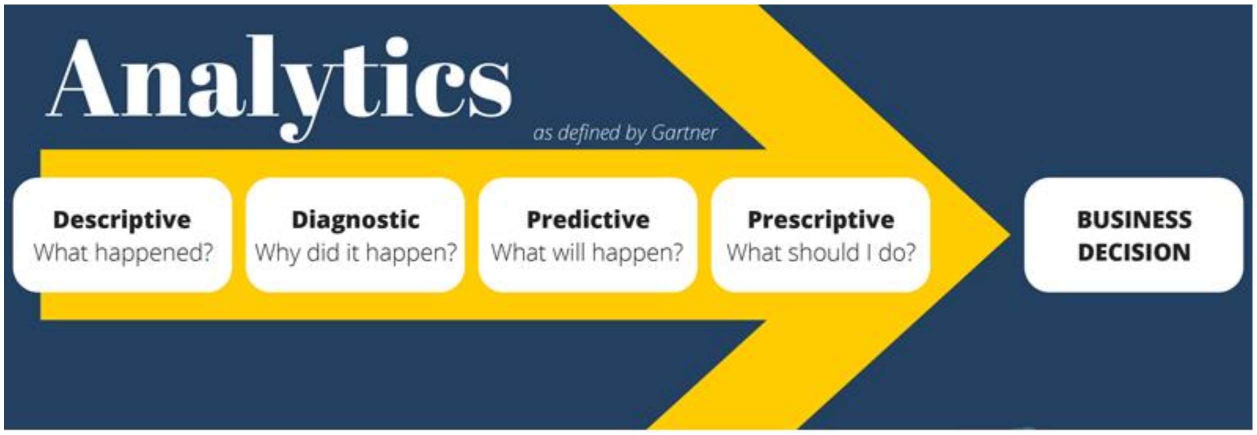
Data Analytics*:*

- Data Analytics is the process of examining raw data (data sets) with the purpose of drawing conclusions about that information, increasingly with the aid of specialized systems and software.
- Data Analytics involves applying an algorithmic or mechanical process to derive insights. For example, running through a number of data sets to look for meaningful correlations between each other.
- It is used in a number of industries to allow the organizations and companies to make better decisions as well as verify and disprove existing theories or models.
- The focus of Data Analytics lies in inference, which is the process of deriving conclusions that are solely based on what the researcher already knows.
- Data analytics initiatives can help businesses increase revenues, improve operational efficiency, optimize marketing campaigns and customer service efforts, respond more quickly to emerging market trends and gain a competitive edge -- all with the ultimate goal of boosting business performance.

Types of Analytics:

There are 4 types of analytics. Here, we start with the simplest one and go down to more sophisticated. As it happens, the more complex an analysis is, the more value it brings.

1. Descriptive Analytics
2. Diagnostic Analysis
3. Perspective Analytics
4. Predictive Analytics



1. Descriptive analytics:

- The simplest way to define descriptive analytics is that, it answers the question **“What has happened?”**
- This type of analytics, analyses the data coming in real-time and historical data for insights on how to approach the future.
- The main objective of descriptive analytics is to find out the reasons behind precious success or failure in the past.
- The ‘Past’ here, refers to any particular time in which an event had occurred and this could be a month ago or even just a minute ago.
- The vast majority of big data analytics used by organizations falls into the category of descriptive analytics. 90% of organizations today use descriptive analytics which is the most basic form of analytics.
- Descriptive analytics juggles raw data from multiple data sources to give valuable insights into the past. However, these findings simply signal that something is wrong or right, without explaining why. For this reason, highly data-driven companies do not content themselves with descriptive analytics only, and prefer combining it with other types of data analytics.
- Eg:A manufacturer was able to decide on focus product categories based on the analysis of revenue, monthly revenue per product group, income by product group, total quality of metal parts produced per month.

2. Diagnostic analytics:

- At this stage, historical data can be measured against other data to answer the question of ***why something happened*.**
- Companies go for diagnostic analytics, as it gives a deep insight into a particular problem. At the same time, a company should have detailed information at their disposal; otherwise data collection may turn out to be individual for every issue and time-consuming.

- Eg: Let's take another look at the examples from different industries: a healthcare provider compares patients' response to a promotional campaign in different regions; a retailer drills the sales down to subcategories.

3. Predictive analytics:

- Predictive analytics tells ***what is likely to be happen***. It uses the findings of descriptive and diagnostic analytics to detect tendencies, clusters and exceptions, and to predict future trends, which makes it a valuable tool for forecasting.
- Despite numerous advantages that predictive analytics brings, it is essential to understand that forecasting is just an estimate, the accuracy of which highly depends on data quality and stability of the situation, so it requires a careful treatment and continuous optimization.
- Eg: A management team can weigh the risks of investing in their company's expansion based on cash flow analysis and forecasting. Organizations like Walmart, Amazon and other retailers leverage predictive analytics to identify trends in sales based on purchase patterns of customers, forecasting customer behavior, forecasting inventory levels, predicting what products customers are likely to purchase together so that they can offer personalized recommendations, predicting the amount of sales at the end of the quarter or year.

4. Prescriptive analytics

- The purpose of prescriptive analytics is to literally prescribe ***what action to take*** to eliminate a future problem or take full advantage of a promising trend.
- Prescriptive analytics is a combination of data, mathematical models and various business rules.
- The data for prescriptive analytics can be both internal (within the organization) and external (like social media data).
- Besides, prescriptive analytics uses sophisticated tools and technologies, like machine learning, business rules and algorithms, which make it sophisticated to implement and manage. That is why, before deciding to adopt prescriptive analytics, a company should compare required efforts vs. an expected added value.
- Prescriptive analytics are comparatively complex in nature and many companies are not yet using them in day-to-day business activities, as it becomes difficult to manage. Large scale organizations use prescriptive analytics for scheduling the inventory in the supply chain, optimizing production, etc. to optimize customer experience.
- An example of prescriptive analytics: a multinational company was able to identify opportunities for repeat purchases based on customer analytics and sales history.

Descriptive and diagnostic analytics* help you construct a narrative of the **past while *predictive and prescriptive analytics* help you envision a possible **future.**

Need for Big Data Analytics:

The new benefits that big data analytics brings to the table, however, are speed and efficiency. Whereas a few years ago a business would have gathered information, run analytics and unearthed information that could be used for future decisions, today that business can identify insights for immediate decisions. The ability to work faster – and stay agile – gives organizations a competitive edge they didn't have before.

Big data analytics helps organizations harness their data and use it to identify new opportunities. That, in turn, leads to smarter business moves, more efficient operations, higher profits and happier customers in the following ways:

1. **Cost reduction:** Big data technologies such as Hadoop and cloud-based analytics bring significant cost advantages when it comes to storing large amounts of data – plus they can identify more efficient ways of doing business.
2. **Faster, better decision making:** With the speed of Hadoop and in-memory analytics, combined with the ability to analyze new sources of data, businesses are able to analyze information immediately – and make decisions based on what they've learned.
3. **New products and services:** With the ability to gauge customer needs and satisfaction through analytics comes the power to give customers what they want. Davenport points out that with big data analytics, more companies are creating new products to meet customers' needs.
4. **End Users Can Visualize Data:** While the business intelligence software market is relatively mature, a big data initiative is going to require next-level data visualization tools, which present BI data in easy-to-read charts, graphs and slideshows. Due to the vast quantities of data being examined, these applications must be able to offer processing engines that let end users query and manipulate information quickly—even in real time in some cases.

Use case: What is the need of Hadoop

Problem: An e-commerce site XYZ (having 100 million users) wants to offer a gift voucher of 100$ to its top 10 customers who have spent the most in the previous year. Moreover, they want to find the buying trend of these customers so that company can suggest more items related to them.

Issues:

Huge amount of unstructured data which needs to be stored, processed and analyzed.

Solution:

Apache Hadoop is not only a storage system but is a platform for data storage as well as processing.

- **Storage:** This huge amount of data, Hadoop uses HDFS (Hadoop Distributed File System) which uses commodity hardware to form clusters and store data in a distributed fashion. It works on Write once, read many times principle.
- **Processing:** Map Reduce paradigm is applied to data distributed over network to find the required output.
- **Analyze:** Pig, Hive can be used to analyze the data.
- **Cost:** Hadoop is open source so the cost is no more an issue.

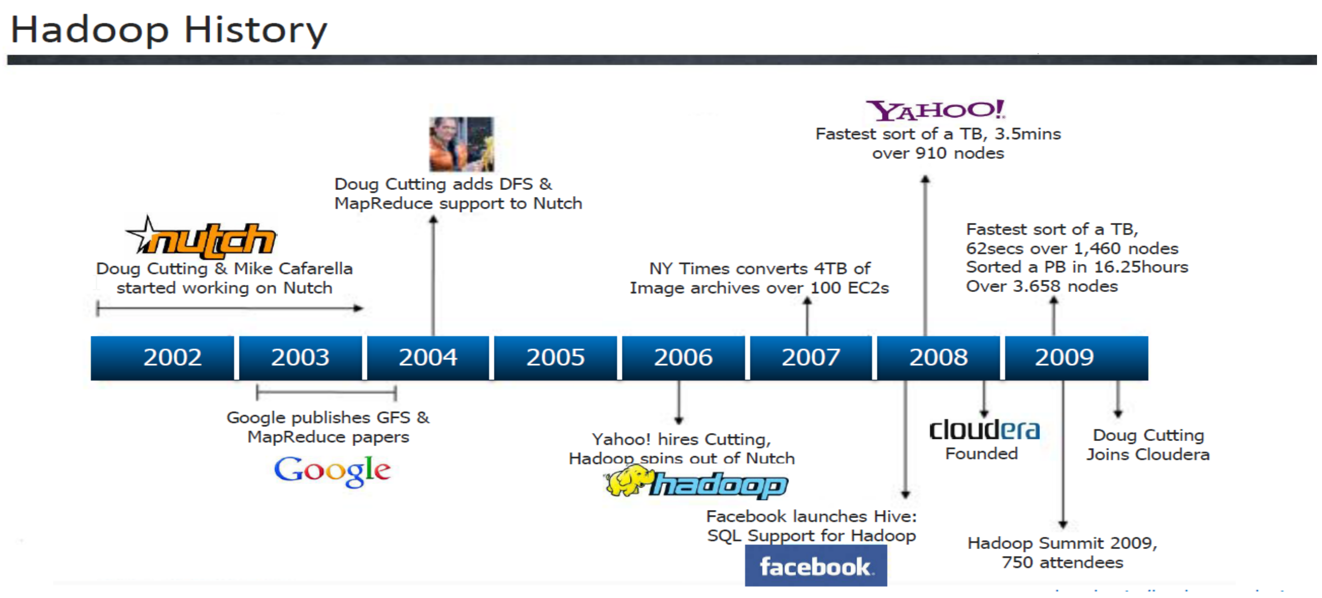
Hadoop

Introduction:

- Hadoop is an open-source software framework used for storing and processing Big Data in a distributed manner on large clusters of commodity hardware. Hadoop is licensed under Apache Software Foundation (ASF).
- Hadoop is written in the Java programming language and ranks among the highest-level Apache projects.
- **Doug Cutting** and **Mike J. Cafarella** developed Hadoop.
- By getting inspiration from **Google**, Hadoop is using technologies like **Map-Reduce** programming model as well as Google file system (**GFS**).
- It is optimized to handle massive quantities of data that could be structured, unstructured or semi-structured, using commodity hardware, that is, relatively inexpensive computers.
- It is intended to work upon from a single server to thousands of machines each offering local computation and storage. It supports the large collection of data set in a distributed computing environment.

History:

Hadoop came into existence and why it is so popular in the industry nowadays. So, it all started with two people, Mike Cafarella and Doug Cutting, who were in the process of building a search engine system that can index 1 billion pages. After their research, they estimated that such a system will cost around half a million dollars in hardware, with a monthly running cost of $30,000, which is quite expensive. However, they soon realized that their architecture would not be capable enough to work around with billions of pages on the web.

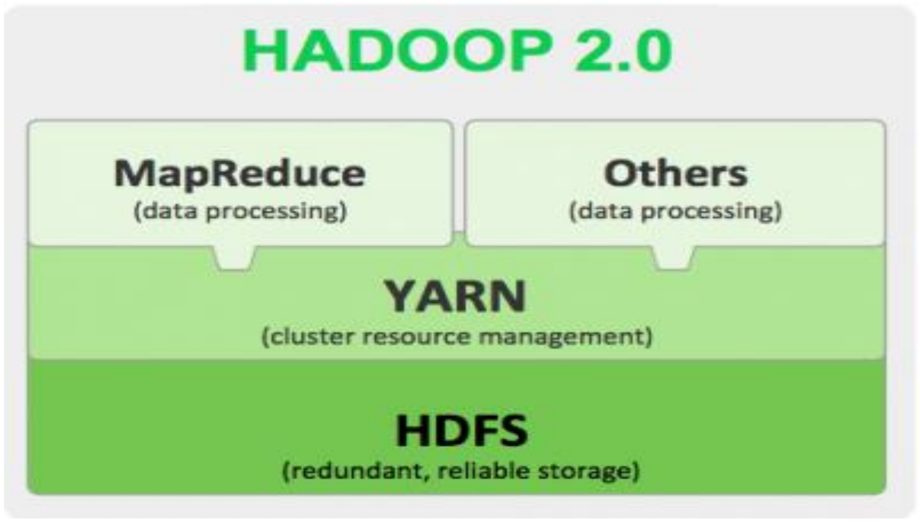

Hadoop History
Doug Cutting & Mike Cafarella started working on Nutch
Doug Cutting adds DFS & MapReduce support to Nutch
NY Times converts 4TB of Image archives over 100 EC2s
Fastest sort of a TB, 3.5mins over 910 nodes
Fastest sort of a TB, 62secs over 1,460 nodes
Sorted a PB in 16.25hours
Over 3.658 nodes
2002
2003
2004
2005
2006
2007
2008
2009
Google publishes GFS & MapReduce papers
Yahoo! hires Cutting, Hadoop spins out of Nutch
Facebook launches Hive: SQL Support for Hadoop
cloudera Founded
Doug Cutting Joins Cloudera
Hadoop Summit 2009, 750 attendees

Doug and Mike came across a paper, published in 2003, that described the architecture of Google's distributed file system, called GFS, which was being used in production at Google. Now, this paper on GFS proved to be something that they were looking for, and soon, they realized that it would solve all their problems of storing very large files that are generated as a part of the web crawl and indexing process. Later in 2004, Google published one more paper that introduced MapReduce to the world. Finally, these two papers led to the foundation of the framework called "***Hadoop***". **Doug Cutting, Mike Cafarella** and team took the solution provided by Google and started an Open Source Project called HADOOP in 2005 and Doug named it after his son's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.

Hadoop Architecture:

- Apache Hadoop offers a scalable, flexible and reliable distributed computing big data framework for a cluster of systems with storage capacity and local computing power by leveraging commodity hardware.
- Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capable enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for huge amounts of data.
- Hadoop follows a Master Slave architecture for the transformation and analysis of large datasets using Hadoop MapReduce paradigm.
- The 3 **important Hadoop** ***core components*** that play a vital role in the Hadoop architecture are -

1. Hadoop Distributed File System (HDFS)
2. Hadoop MapReduce
3. Yet Another Resource Negotiator (YARN)



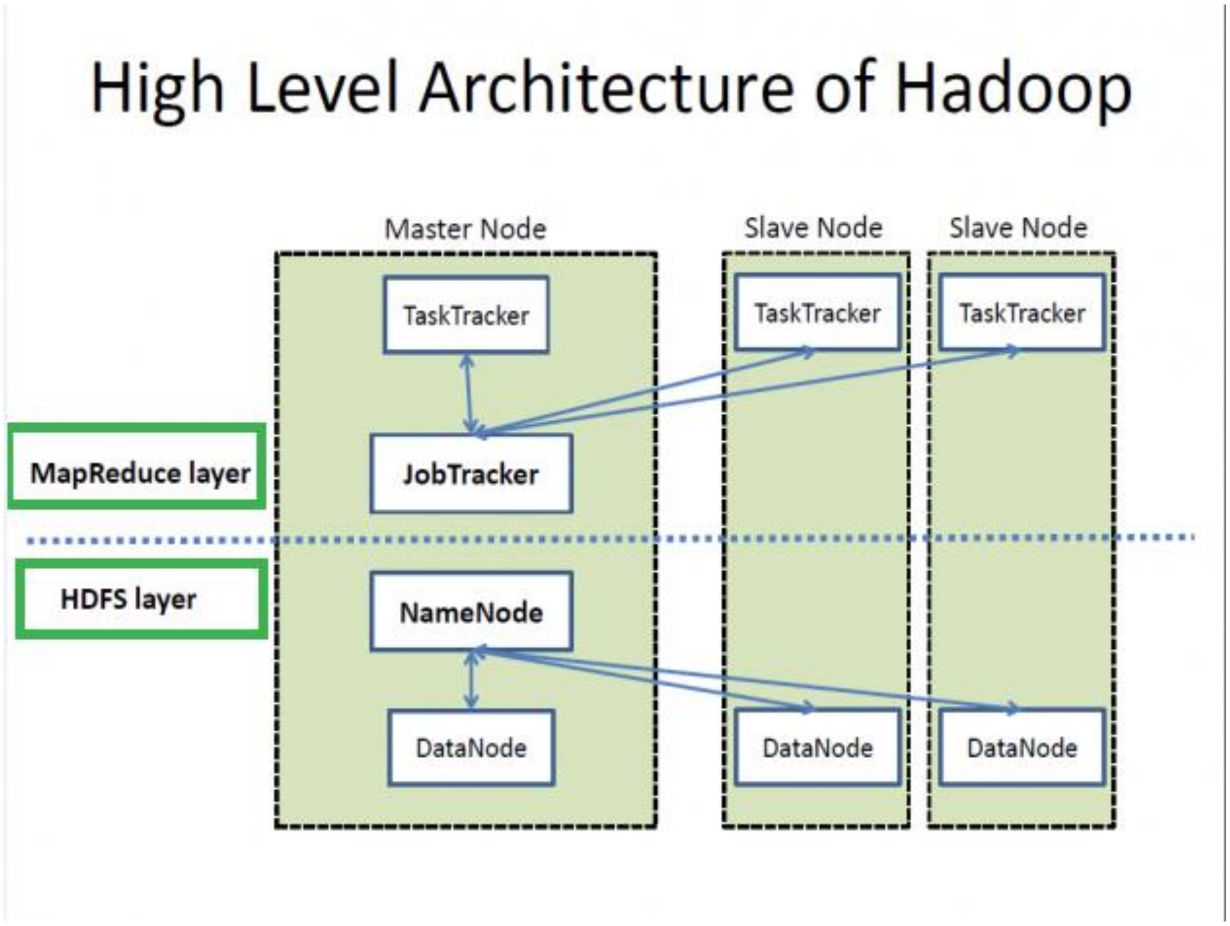
- **Hadoop Distributed File System (HDFS):**
  - Hadoop Distributed File System runs on top of the existing file systems on each node in a Hadoop cluster.
  - Hadoop Distributed File System is a block-structured file system where each file is divided into blocks of a pre-determined size.
  - Data in a Hadoop cluster is broken down into smaller units (called blocks) and distributed throughout the cluster. Each block is duplicated twice (for a total of three copies), with the two replicas stored on two nodes in a rack somewhere else in the cluster.
  - Since the data has a default replication factor of three, it is highly available and fault-tolerant.
  - If a copy is lost (because of machine failure, for example), HDFS will automatically re-replicate it elsewhere in the cluster, ensuring that the threefold replication factor is maintained.

- **Hadoop MapReduce:** This is for parallel processing of large data sets.
  - The MapReduce framework consists of a single **master node** (**JobTracker)** and n numbers of **slave nodes** (**Task Tracker**) where n can be 1000s. Master manages, maintains and monitors the slaves while slaves are the actual worker nodes.
  - Client submit a job to Hadoop. The job can be a mapper, a reducer or a list of input. The job is sent to job tracker process on master node. Each slave node runs a process through task tracker.

- The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks.
- The slaves TaskTracker execute the tasks as directed by the master and provide task-status information to the master periodically.
- Master stores the metadata (data about data) while slaves are the nodes which store the data. The client connects with master node to perform any task.

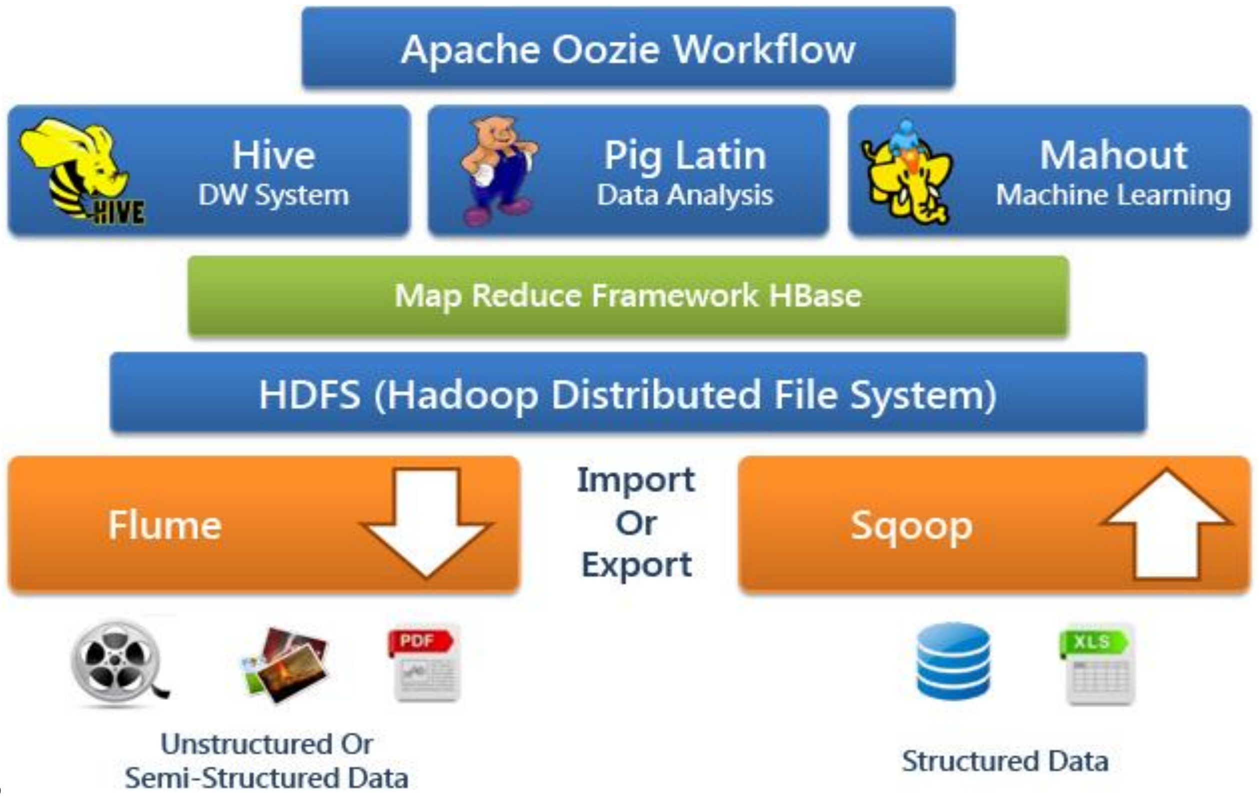
➢ **Hadoop YARN:** YARN (Yet Another Resource Negotiator) is the framework responsible for assigning computational resources for application execution and cluster management. YARN consists of three core components:

- ResourceManager (one per cluster)
- ApplicationMaster (one per application)
- NodeManagers (one per node)



Hadoop ecosystem:

- Hadoop Ecosystem is neither a programming language nor a service; it is a platform or framework which solves big data problems. You can consider it as a suite which encompasses a number of services (ingesting, storing, analyzing and maintaining) inside it. Let us discuss and get a brief idea about how the services work individually and in collaboration.
- The Hadoop ecosystem provides the furnishings that turn the framework into a comfortable home for big data activity that reflects your specific needs and tastes.
- The Hadoop ecosystem includes both official Apache open source projects and a wide range of commercial tools and solutions.
- Below are the Hadoop components, that together form a Hadoop ecosystem,
  - ✓ **HDFS** -> *Hadoop Distributed File System*
  - ✓ **YARN** -> *Yet Another Resource Negotiator*
  - ✓ **MapReduce** -> *Data processing using programming*
  - ✓ **Spark** -> In-memory Data Processing
  - ✓ **PIG, HIVE**-> *Data Processing Services using Query (SQL-like)*
  - ✓ **HBase** -> *NoSQL Database*
  - ✓ **Mahout, Spark MLlib** -> *Machine Learning*
  - ✓ **Apache Drill** -> *SQL on Hadoop*
  - ✓ **Zookeeper** -> *Managing Cluster*
  - ✓ **Oozie** -> *Job Scheduling*
  - ✓ **Flume, Sqoop** -> *Data Ingesting Services*
  - ✓ **Solr&Lucene** -> *Searching & Indexing*



S

Apache open source Hadoop ecosystem elements:

Spark, Pig, and Hive are three of the best-known Apache Hadoop projects. Each is used to create applications to process Hadoop data.

- **Spark:** Apache Spark is a framework for real time data analytics in a distributed computing environment. It executes in-memory computations to increase speed of data processing over Map-Reduce.
- **Hive:** Facebook created HIVE for people who are fluent with SQL. Basically, HIVE is a data warehousing component which performs reading, writing and managing large data sets in a distributed environment using SQL-like interface. The query language of Hive is called Hive Query Language (HQL), which is very similar like SQL. ***HIVE + SQL = HQL.*** It provides tools for ETL operations and brings some SQL-like capabilities to the environment.
- **Pig:** Pig is a procedural language for developing parallel processing applications for large data sets in the Hadoop environment. Pig is an alternative to Java programming for MapReduce, and automatically

generates MapReduce functions. Pig includes Pig Latin, which is a scripting language. Pig translates Pig Latin scripts into MapReduce, which can then run on YARN and process data in the HDFS cluster.

- **HBase:** HBase is a scalable, distributed, NoSQL database that sits atop the HFDS. It was designed to store structured data in tables that could have billions of rows and millions of columns. It has been deployed to power historical searches through large data sets, especially when the desired data is contained within a large amount of unimportant or irrelevant data (also known as sparse data sets).
- **Oozie:** Oozie is the workflow scheduler that was developed as part of the Apache Hadoop project. It manages how workflows start and execute, and also controls the execution path. Oozie is a server-based Java web application that uses workflow definitions written in hPDL, which is an XML Process Definition Language similar to JBOSS JBPM jPDL.
- **Sqoop:** Sqoop is bi-directional data injection tool. Think of Sqoop as a front-end loader for big data. Sqoop is a command-line interface that facilitates moving bulk data from Hadoop into relational databases and other structured data stores. Using Sqoop replaces the need to develop scripts to export and import data. One common use case is to move data from an enterprise data warehouse to a Hadoop cluster for ETL processing. Performing ETL on the commodity Hadoop cluster is resource efficient, while Sqoop provides a practical transfer method.

- **Ambari** – A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig, and Sqoop.
- **Flume** – A distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming event data.
- **Mahout** – A scalable machine learning and data mining library.
- **Zookeeper** – A high-performance coordination service for distributed applications.

The ecosystem elements described above are all open source Apache Hadoop projects. There are numerous commercial solutions that use or support the open source Hadoop projects.

Hadoop Distributions:

- ➢ Hadoop is an open-source, catch-all technology solution with incredible scalability, low cost storage systems and fast paced big data analytics with economical server costs.
- ➢ Hadoop Vendor distributions overcome the drawbacks and issues with the open source edition of Hadoop. These distributions have added functionalities that focus on:

- **Support:**
  Most of the Hadoop vendors provide technical guidance and assistance that makes it easy for customers to adopt Hadoop for enterprise level tasks and mission critical applications.
- **Reliability:**
  Hadoop vendors promptly act in response whenever a bug is detected. With the intent to make commercial solutions more stable, patches and fixes are deployed immediately.
- **Completeness:**
  Hadoop vendors couple their distributions with various other add-on tools which help customers customize the Hadoop application to address their specific tasks.
  - **Fault Tolerant:**
    Since the data has a default replication factor of three, it is highly available and fault-tolerant.

Here is a list of top Hadoop Vendors who play a key role in big data market growth

- ➢ Amazon Elastic MapReduce
- ➢ Cloudera CDH Hadoop Distribution
- ➢ Hortonworks Data Platform (HDP)
- ➢ MapR Hadoop Distribution
- ➢ IBM Open Platform (IBM Infosphere Big insights)
- ➢ Microsoft Azure's HDInsight -Cloud based Hadoop Distribution

Advantages of Hadoop:

The increase in the requirement of computing resources has made Hadoop a viable and extensively used programming framework. Modern day organizations can learn Hadoop and leverage their knowhow of managing processing power of their businesses.

1. Scalable: Hadoop is a highly scalable storage platform, because it can stores and distribute very large data sets across hundreds of inexpensive servers that operate in parallel. Unlike traditional relational database systems (RDBMS) that can’t scale to process large amounts of data, Hadoop enables businesses to run applications on thousands of nodes involving many thousands of terabytes of data.

2. Cost effective: Hadoop also offers a cost effective storage solution for businesses' exploding data sets. The problem with traditional relational database management systems is that it is extremely cost prohibitive to scale to such a degree in order to process such massive volumes of data. In an effort to reduce costs, many companies in the past would have had to down-sample data and classify it based on certain assumptions as to which data was the most valuable. The raw data would be deleted, as it would be too cost-prohibitive to keep. While this approach may have worked in the short term, this meant that when business priorities changed, the complete raw data set was not available, as it was too expensive to store.

3. Flexible: Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations. Hadoop can be used for a wide variety of purposes, such as log processing, recommendation systems, data warehousing, market campaign analysis and fraud detection.

4. Speed of Processing: Hadoop's unique storage method is based on a distributed file system that basically 'maps' data wherever it is located on a cluster. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. If you're dealing with large volumes of unstructured data, Hadoop is able to efficiently process terabytes of data in just minutes, and petabytes in hours.

5. Resilient to failure: A key advantage of using Hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.

Summary of UNIT – I:

- **Part – I - BigData**
    1. Data, Information, Knowledge application on data
    2. What is Bigdata, examples of Bigdata (TBs &PBs of data)
    3. Sources of Bigdata, Facts about Bigdata (ex: Aircrafts, social Media, tweets, e-commerce)
    4. Types of Bigdata (structured, unstructured, semi-structured)
    5. 3 Vs of Bigdata (properties of bigdata – volume, velocity, variety)
    6. Data Analytics, Types of Data Analytics
    7. Need of Data Analytics

- **Part – II - Hadoop**
    1. What is Hadoop
    2. History of Hadoop (Doug Cutting -2005)
    3. Evolution of Hadoop
    4. Basic Architecture of Hadoop (High level architecture also)
    5. Hadoop eco-system (various tools- HDFS, Mapreduce, spark, Hbase, pig, Hive, Zoo Keeper, Sqoop, Flume, Oozie etc).
    6. Hadoop distributions
    7. Benefits of Hadoop

Activities:

8. Sketch Noting
9. One minute Chat
10. Quescussion
11. Presentations
12. Question/Answers

Unit - II

Hadoop Distributed File System (HDFS): Introduction, Design Principles, Components of HDFS – Name Node, Secondary Name Node, Data Nodes, HDFS File Blocks, Storing File Blocks into HDFS from Client Machine, Rack Awareness, HDFS File Commands.

Map Reduce: Introduction, Architectural Components of Map Reduce, Functional Components of Map Reduce, Heartbeat Signal, Speculative Execution

HDFS:

- **Hadoop Distributed file system** is a Java based distributed file system that allows you to store large data across multiple nodes in a Hadoop cluster. So, if you install Hadoop, you get HDFS as an underlying storage system for storing the data in the distributed environment.
- Hadoop File System was developed using distributed file system design. It runs on commodity hardware. Unlike other distributed systems, HDFS is highly fault tolerant and designed using low-cost hardware.
- HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.
- HDFS runs on top of the existing file systems on each node in a Hadoop cluster. It is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines.

Features of HDFS:

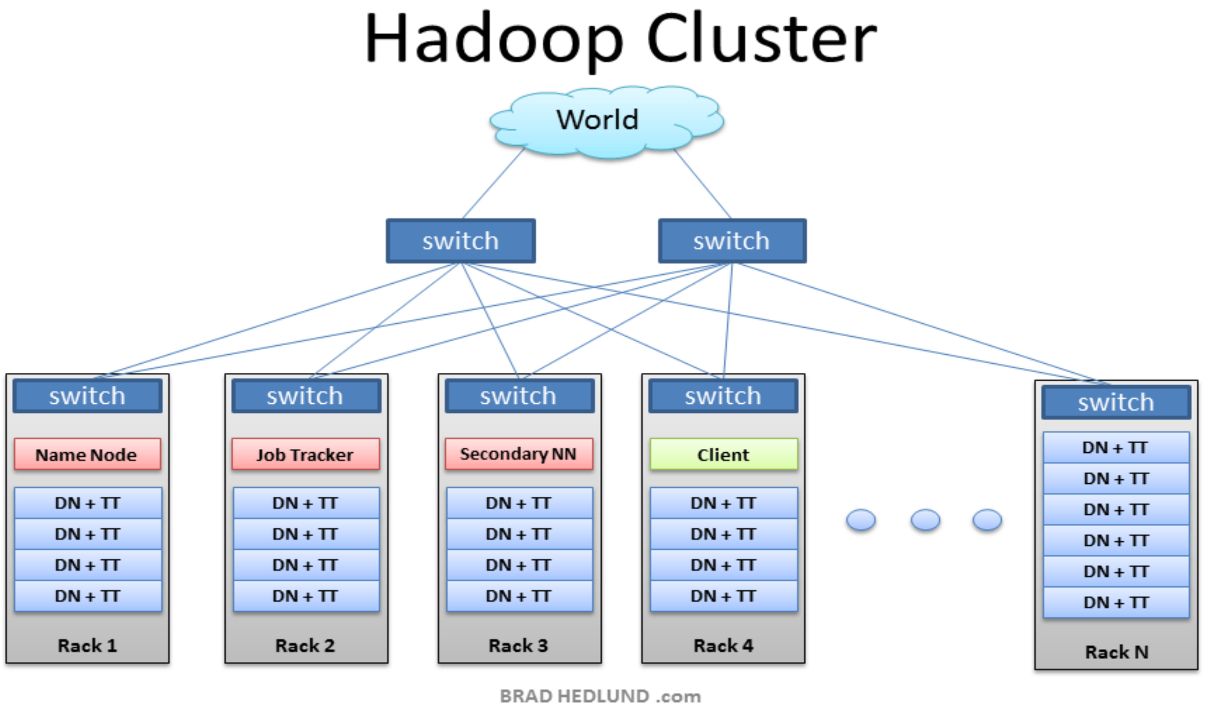
- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of namenode and datanode help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication.

Basic terminology:

Node: A node is simply a computer. This is typically non-enterprise, commodity hardware for nodes that contain data.

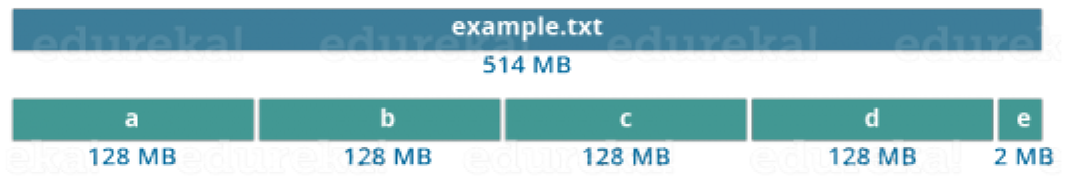
Rack: Collection of nodes is called as a rack. A rack is a collection of 30 or 40 nodes that are physically stored close together and are all connected to the same network switch.
Network bandwidth between any two nodes in the same rack is greater than bandwidth between two nodes on different racks.

Cluster: A Hadoop Cluster (or just cluster from now on) is a collection of racks.



File Blocks:

Blocks are nothing but the smallest continuous location on your hard drive where data is stored. In general, in any of the File System, you store the data as a collection of blocks. Similarly, HDFS stores each file as blocks which are scattered throughout the Apache Hadoop cluster. The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x) which you can configure as per your requirement. All blocks of the file are the same size except the last block, which can be either the same size or smaller. The files are split into 128 MB blocks and then stored into the Hadoop file system. The Hadoop application is responsible for distributing the data block across multiple nodes.



Let's take an example where we have a file "example.txt" of size 514 MB as shown in above figure. Suppose that we are using the default configuration of block size, which is 128 MB. Then, 5 blocks will be created. The first four blocks will be of 128 MB. But, the last block will be of 2 MB size only.

Components of HDFS:

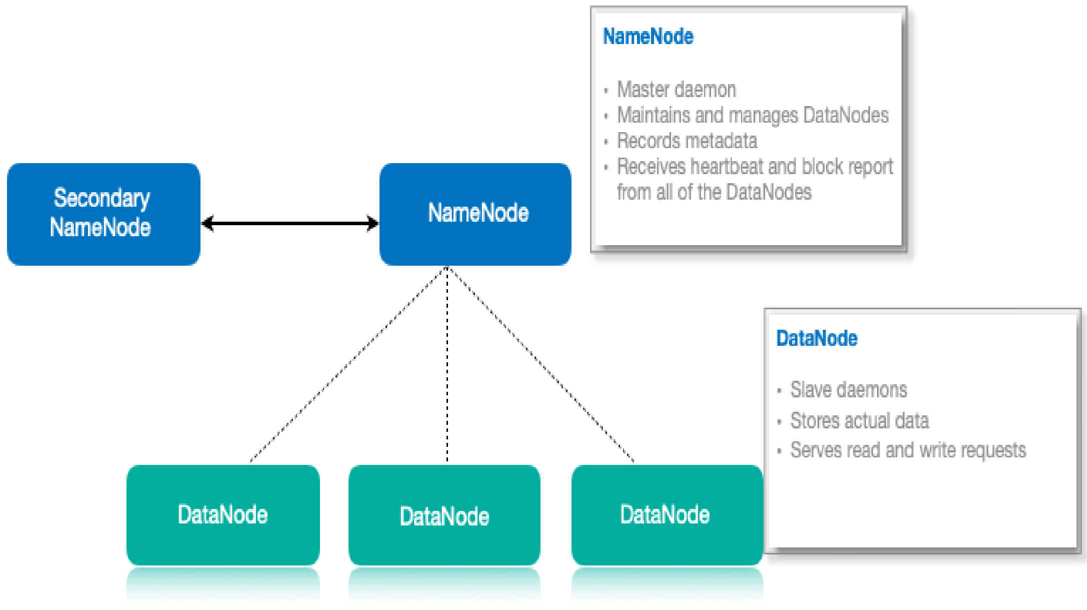
HDFS is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines. Apache Hadoop HDFS Architecture follows a *Master/Slave Architecture*, where a cluster comprises of a single **NameNode (Master node)** and all the other nodes are **DataNodes (Slave nodes).** HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.

NameNode:

NameNode is the master node in the Apache Hadoop HDFS Architecture that maintains and manages the blocks present on the DataNodes (slave nodes). NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients. The HDFS architecture is built in such a way that the user data never resides on the NameNode. Name node contains metadata and the data resides on DataNodes only.

Functions of NameNode:

- It is the master daemon that maintains and manages the DataNodes (slave nodes)
- Manages the file system namespace.
- It records the metadata of all the files stored in the cluster, e.g. the location of blocks stored, the size of the files, permissions, hierarchy, etc. There are two files associated with the metadata:
    - **FsImage:** It contains the complete state of the file system namespace since the start of the NameNode.
    - **EditLogs:** It contains all the recent modifications made to the file system with respect to the most recent FsImage.
- It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
- The NameNode is also responsible to take care of the **replication factor** of all the blocks which we will discuss in detail later in this HDFS tutorial blog.
- In **case of the DataNode failure**, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

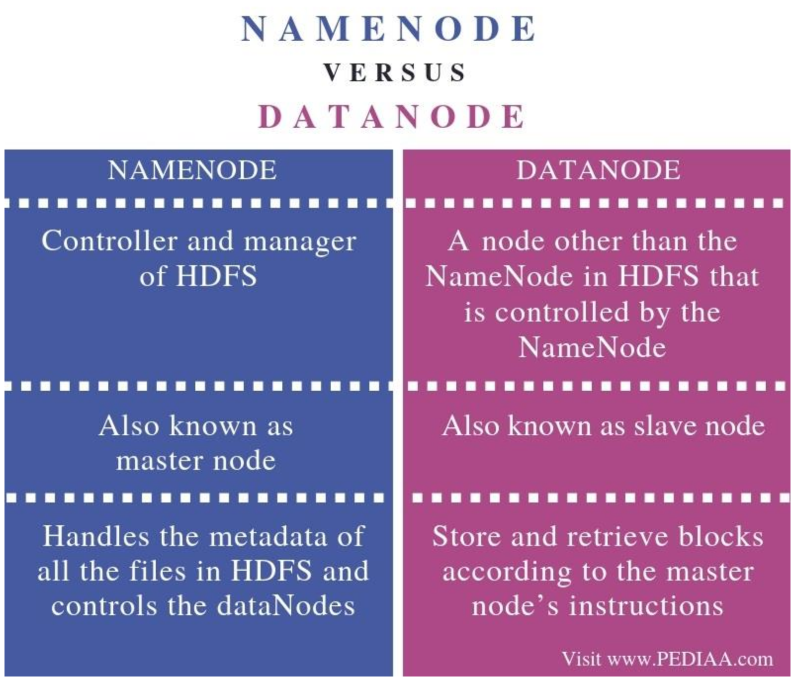


DataNode:

Data Nodes are the slave nodes in HDFS. Unlike NameNode, DataNode is commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. The Data Node is a block server that stores the data in the local file ext3 or ext4.

Functions of DataNode:

- The actual data is stored on DataNodes.
- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.
- They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default; this frequency is set to 3 seconds.



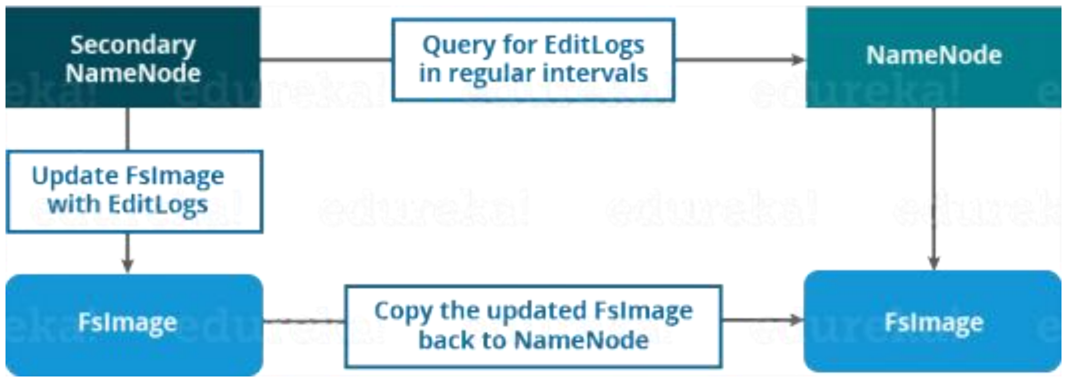
NAMENODE VERSUS DATANODE

NAMENODE	DATANODE
Controller and manager of HDFS	A node other than the NameNode in HDFS that is controlled by the NameNode
Also known as master node	Also known as slave node
Handles the metadata of all the files in HDFS and controls the dataNodes	Store and retrieve blocks according to the master node's instructions

Visit www.PEDIAA.com

Secondary NameNode:

It is a separate physical machine which acts as a helper of name node. It performs periodic check points. It communicates with the name node and take snapshot of meta data which helps minimize downtime and loss of data. The Secondary NameNode works concurrently with the primary NameNode as a **helper daemon.**

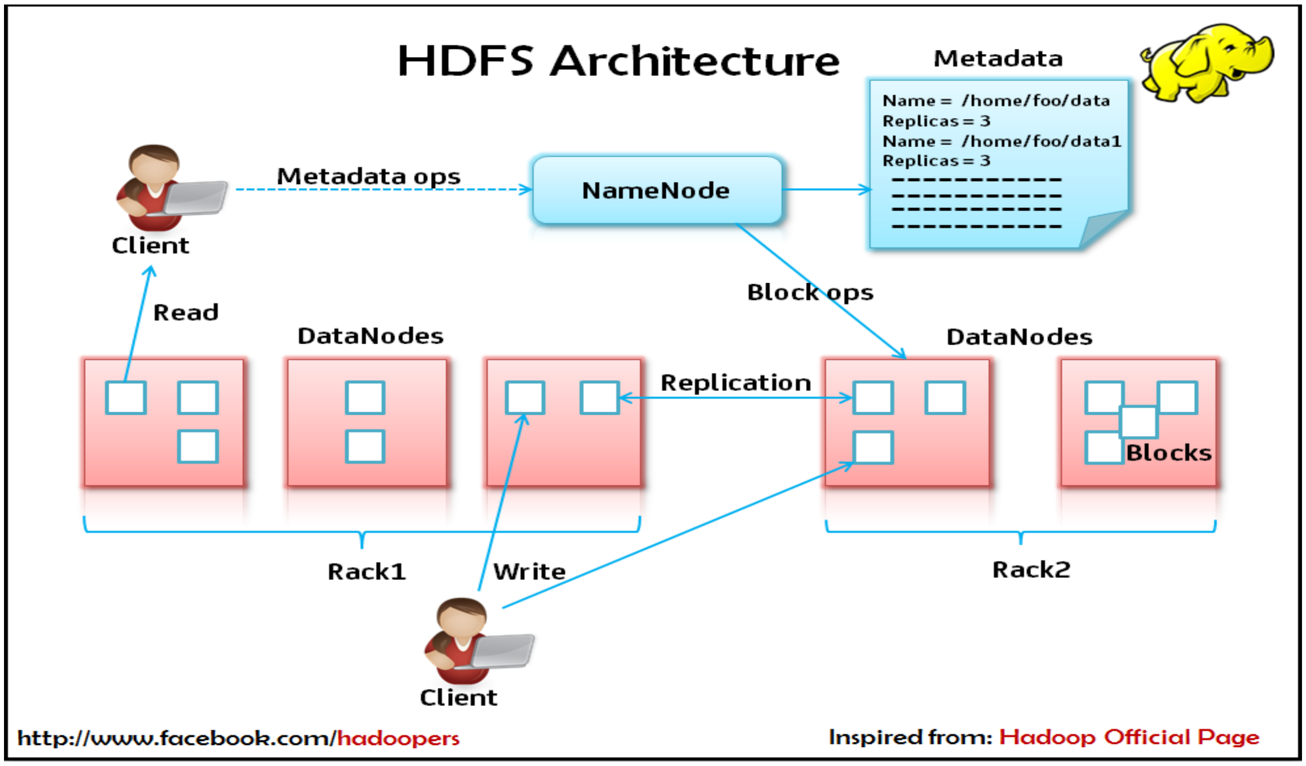

Secondary NameNode
Query for EditLogs in regular intervals
NameNode
Update FsImage with EditLogs
FsImage
Copy the updated FsImage back to NameNode
FsImage

Functions of Secondary NameNode:

- The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.
- It is responsible for combining the EditLogs with FsImage from the NameNode.

- It downloads the EditLogs from the NameNode at regular intervals and applies to FsImage. The new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time.
- Hence, Secondary NameNode performs regular checkpoints in HDFS. Therefore, it is also called CheckpointNode.

HDFS Architecture: Apache Hadoop HDFS Architecture follows a *Master/Slave Architecture*, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes). HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.



Storing data into HDFS:

HDFS stores data in a reliable fashion using **replication and distribution**. Here is the series of steps that happen when a client writes a file in hdfs:

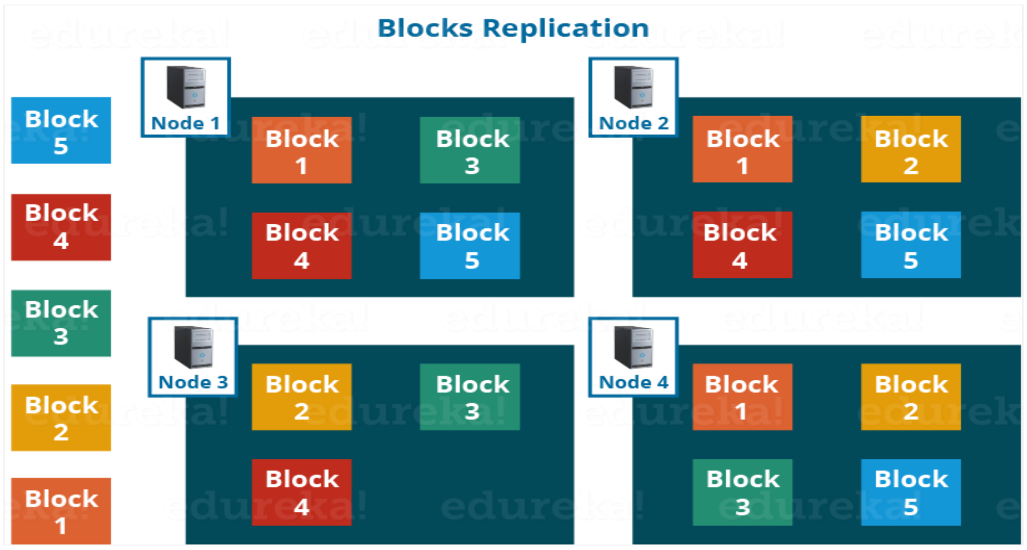
1. Client requests Namenode to create the file. It passes size of file as a parameter
2. Namenode responds with location of nodes where client can store data. By default there'll be 3 locations per block. If file size is 200mb, there'll be 2 blocks, first 128 mb, 2nd 72 mb. Similarly depending on the size, you'll have n number of blocks.
3. Client directly starts writing data to the first datanode out of three given by namenode. Please note that if there are 2 blocks to be written client can start writing them in parallel.
4. When the first datanode has stored the block, it replies to the client with success and now it passes on the same block to 2nd datanode. 2nd datanode will write this block and pass it on to 3rd datanode.
5. So basically writing of blocks from client to datanodes happens in parallel but replication happens in series.

Blocks of same file can go to different nodes, at least the replicated blocks will always be on different nodes. The first block is always on the datanode which is nearest to the client, 2nd and 3rd blocks are stored based on free capacity of the datanodes and/or rack awareness.

What is Replication Management?

- HDFS performs replication to provide Fault Tolerant and to improve data reliability.
- There could be situations where the data is lost in many ways- node is down, Node lost the network connectivity, a node is physically damaged, and a node is intentionally made unavailable for horizontal scaling.
- For any of the above-mentioned reasons, data will not be available if the replication is not made. HDFS usually maintains 3 copies of each Data Block in different nodes and different Racks. By doing this, data is made available even if one of the systems is down.
- Downtime will be reduced by making data replications. This improves the reliability and makes HDFs fault tolerant.
- Block replication provides fault tolerance. If one copy is not accessible and corrupted, we can read data from other copy.

- The number of copies or replicas of each block of a file in HDFS Architecture is replication factor. ***The default replication factor is 3*** which are again configurable. So, each block replicates three times and stored on different DataNodes.
- So, as you can see in the figure below where each block is replicated three times and stored on different DataNodes (considering the default replication factor): If we are storing a file of 128 MB in HDFS using the default configuration, we will end up occupying a space of 384 MB (3*128 MB).



Rack Awareness in HDFS Architecture:

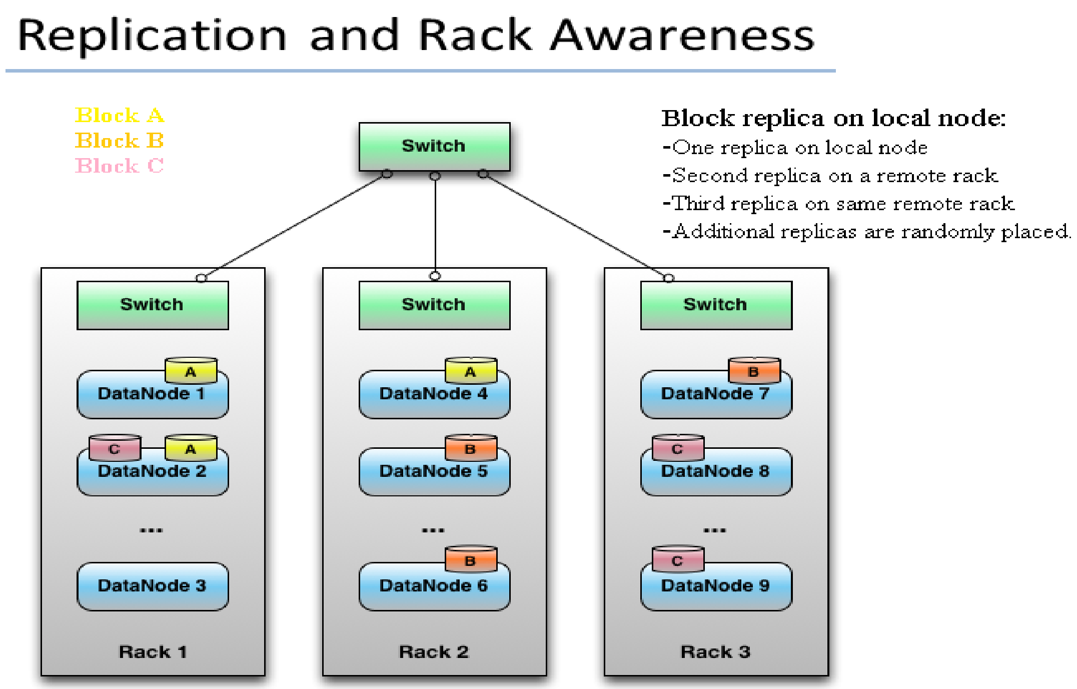
- **Rack**- It the collection of machines around 30-40. All these machines are connected using the same network switch and if that network goes down then all machines in that rack will be out of service. Thus we say rack is down.
- Rack Awareness was introduced by Apache Hadoop to overcome this issue. **Rack awareness** is the knowledge that how the data nodes are distributed across the rack of Hadoop cluster.
- In the large cluster of Hadoop, in order to improve the network traffic while reading/writing HDFS file, NameNode chooses the DataNode which is closer to the same rack or nearby rack to Read /write request. NameNode achieves rack information by maintaining the rack ids of each DataNode. This concept that chooses Datanodes based on the rack information is called **Rack Awareness** in Hadoop.

- In HDFS, NameNode makes sure that all the replicas are not stored on the same rack or single rack; it follows Rack Awareness Algorithm to reduce latency as well as fault tolerance.
- As we know the default **Replication Factor** is **3** and Client want to place a file in HDFS, then Hadoop places the replicas as follows:

  1) The first replica is written to the data node creating the file, to improve the write performance because of the write affinity.

  2) The second replica is written to another data node within the same rack, to minimize the cross-rack network traffic.

  3) The third replica is written to a data node in a different rack, ensuring that even if a switch or rack fails, the data is not lost (Rack awareness).
- This configuration is maintained to make sure that the File is never lost in case of a Node Failure or even an entire Rack Failure.


Replication and Rack Awareness
Block A
Block B
Block C
Switch
Block replica on local node:
-One replica on local node
-Second replica on a remote rack
-Third replica on same remote rack
-Additional replicas are randomly placed.
Switch
DataNode 1
A
DataNode 2
C
A
...
DataNode 3
Rack 1
Switch
DataNode 4
A
DataNode 5
B
...
DataNode 6
B
Rack 2
Switch
DataNode 7
B
DataNode 8
C
...
DataNode 9
C
Rack 3

Advantages of Rack Awareness:

- Minimize the writing cost and Maximize read speed – Rack awareness places read/write requests to replicas on the same or nearby rack. Thus minimizing writing cost and maximizing reading speed.

- Provide maximize network bandwidth and low latency – Rack awareness maximizes network bandwidth by blocks transfer within a rack. This is particularly beneficial in cases where tasks cannot be assigned to nodes where their data is stored locally.
- Data protection against rack failure – By default, the namenode assigns 2nd & 3rd replicas of a block to nodes in a rack different from the first replica. This provides data protection even against rack failure

MapReduce:

MapReduce is a programming framework that allows us to perform parallel and distributed processing on huge data sets in distributed environment.

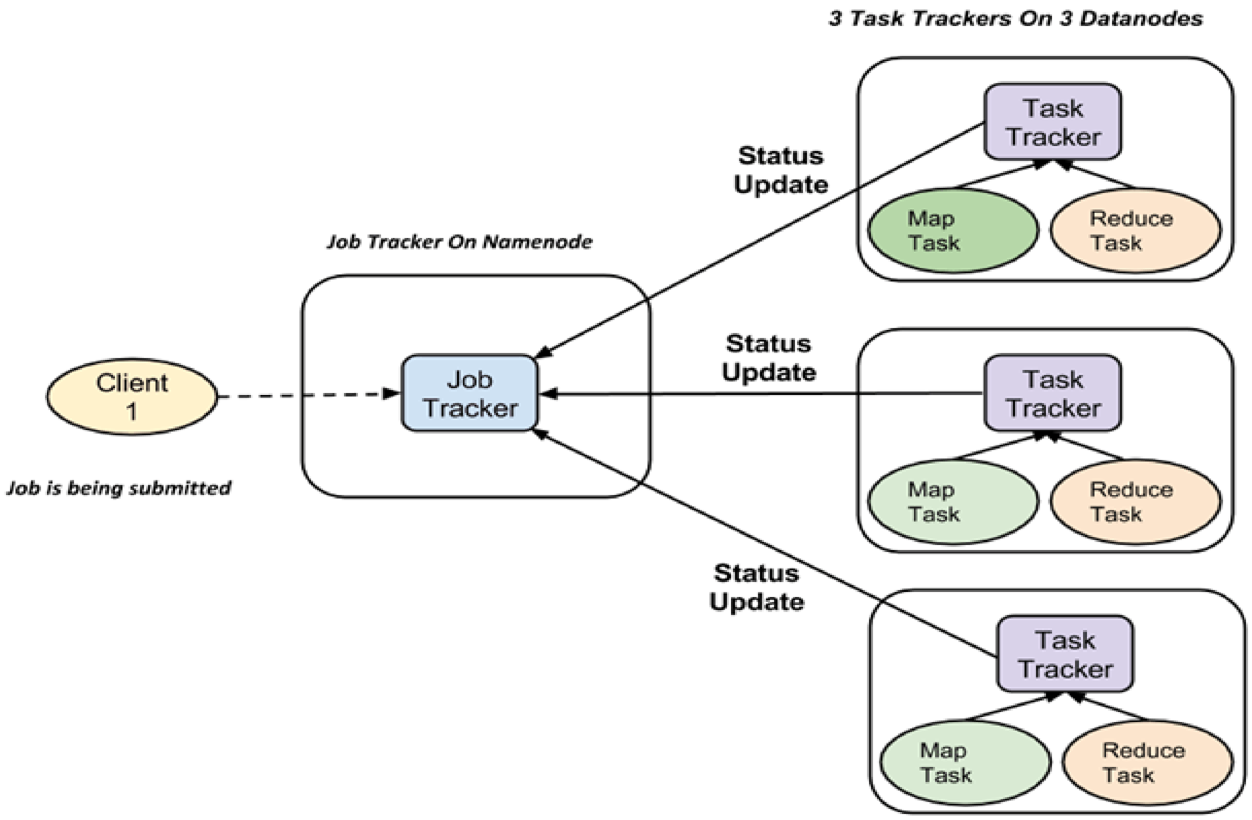
Map Reduce can be implemented by its components:

1. **Architectural components**
    a) Job Tracker
    b) Task Trackers
2. **Functional components**
    a) Mapper (map)
    b) Combiner (Shuffler)
    c) Reducer (Reduce)

Architectural Components:

- The complete execution process (execution of Map and Reduce tasks, both) is controlled by two types of entities called:
    1. A Job tracker : Acts like a master (responsible for complete execution of submitted job)
    2. Multiple Task Trackers : Acts like slaves, each of them performing the job
- For every job submitted for execution in the system, there is one Job tracker that resides on Name node and there are multiple task trackers which reside on Data node.

- A job is divided into multiple tasks which are then run onto multiple data nodes in a cluster.
- It is the responsibility of job tracker to coordinate the activity by scheduling tasks to run on different data nodes.
- Execution of individual task is then look after by task tracker, which resides on every data node executing part of the job.
- Task tracker's responsibility is to send the progress report to the job tracker.
- In addition, task tracker periodically sends 'heartbeat' signal to the Job tracker so as to notify him of current state of the system.
- Thus job tracker keeps track of overall progress of each job. In the event of task failure, the job tracker can reschedule it on a different task tracker.



Functional Components:

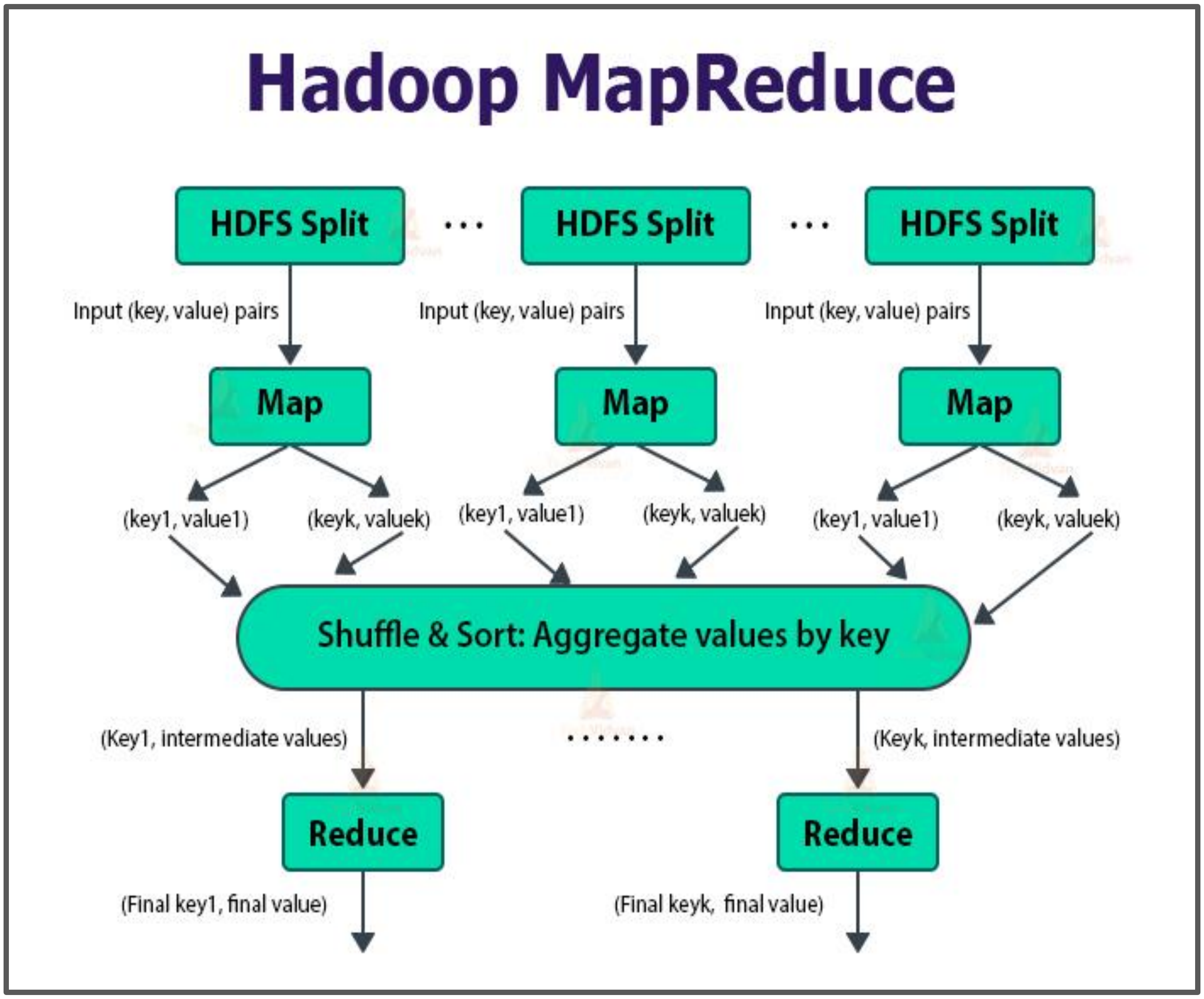
A job (complete Work) is submitted to the master, Hadoop divides the job into phases , map phase and reduce phase. In between Map and Reduce, there is small phase called shuffle & Sort in MapReduce.

1. Map tasks
2. Reduce tasks

Map Phase: This is very first phase in the execution of map-reduce program. In this phase data in each split is passed to a mapping function to produce output values. The map takes key/value pair as input. Key is a reference to the input. Value is the data set on which to operate. Map function applies the business logic to every value in input. Map produces an output is called intermediate output. An output of map is stored on the local disk from where it is shuffled to reduce nodes.

Reduce Phase: In MapReduce **Reduce** takes intermediate Key / Value pairs as input and process the output of the mapper. Key/value pairs provided to reduce are sorted by key. Usually, in the reducer, we do aggregation or summation sort of computation. A function defined by user supplies the values for a given key to the Reduce function. Reduce produces a final output as a list of key/value pairs. This final output is stored in HDFS and replication is done as usual.

Shuffling: This phase consumes output of mapping phase. Its task is to consolidate the relevant records from Mapping phase output.

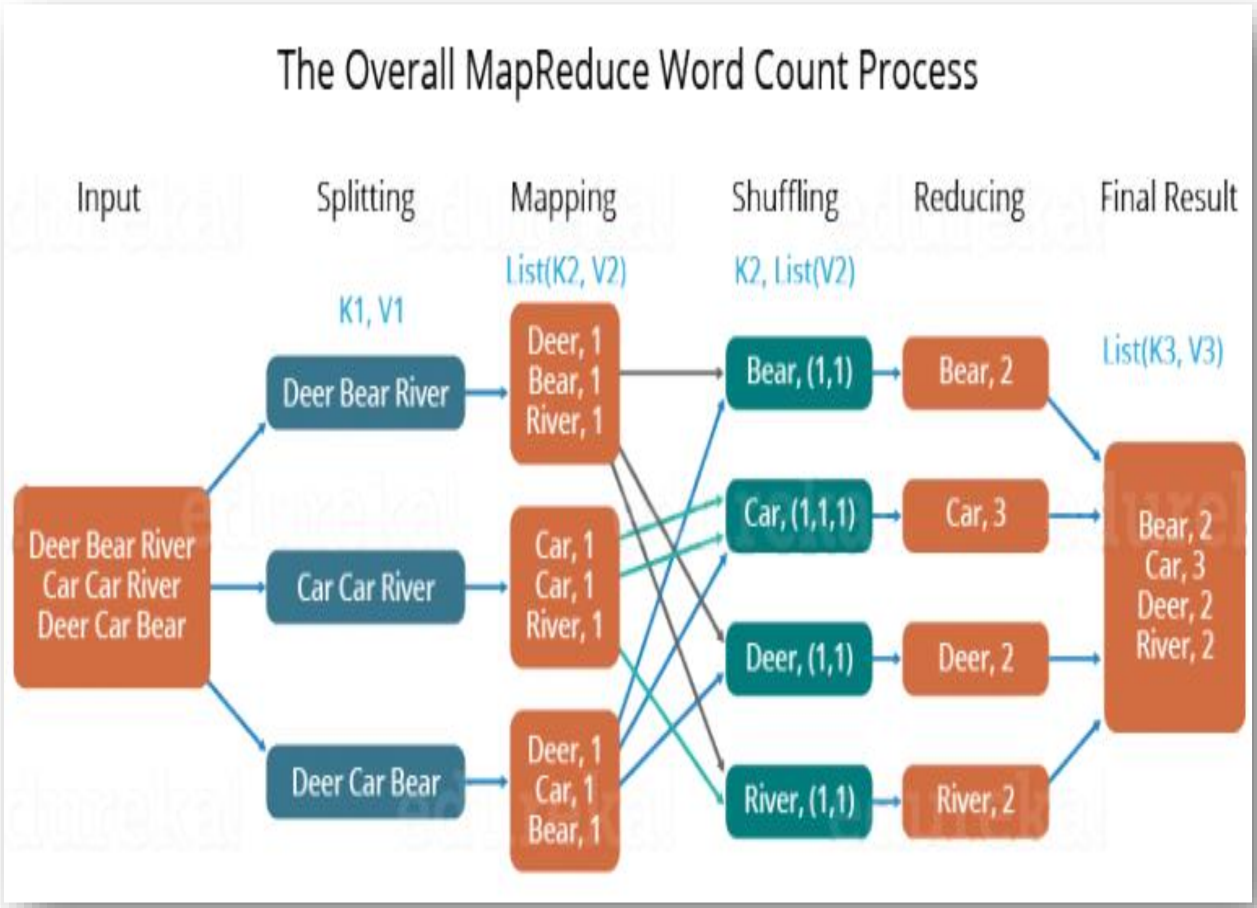


A Word Count Example of MapReduce:

Let us understand, how a MapReduce works by taking an example where I have a text file called example.txt whose contents are as follows:

Dear, Bear, River, Car, Car, River, Deer, Car and Bear

Now, suppose, we need to perform a word count on the sample.txt using MapReduce. So, we will be finding the unique words and the number of occurrences of those unique words.



- First, we divide the input in three splits as shown in the figure. This will distribute the work among all the map nodes.
- Then, we tokenize the words in each of the mapper and give a hardcoded value (1) to each of the tokens or words. The rationale behind giving a hardcoded value equal to 1 is that every word, will occur once.
- Now, a list of key-value pair will be created where the key is the individual word and value is one. So, for the first line (Dear Bear River) we have 3 key-value pairs – Dear, 1; Bear, 1; River, 1. The mapping process remains the same on all the nodes.
- After mapper phase, a partition process takes place where sorting and shuffling happens so that all the tuples with the same key are sent to the corresponding reducer.
- So, after the sorting and shuffling phase, each reducer will have a unique key and a list of values corresponding to that very key. For example, Bear, [1,1]; Car, [1,1,1].., etc.

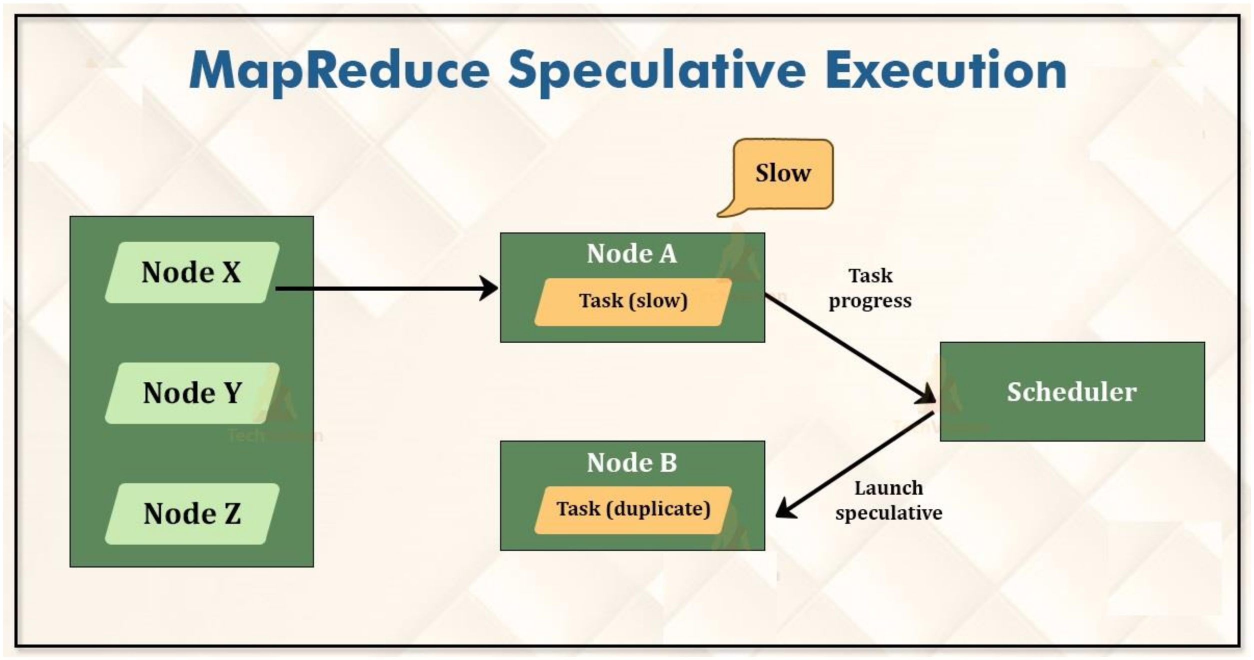
- Now, each Reducer counts the values which are present in that list of values. As shown in the figure, reducer gets a list of values which is [1,1] for the key Bear. Then, it counts the number of ones in the very list and gives the final output as – Bear, 2.
- Finally, all the output key/value pairs are then collected and written in the output file.

Heartbeat Signal:

- HDFS follows a master slave architecture. Namenode (master) stores metadata about the data and Datanodes store/process the actual data (and its replications).
- Now the namenode should know if any datanode in a cluster is down (power failure/network failure) otherwise it will continue assigning tasks or sending data/replications to that dead datanode.
- **Heartbeat** is a mechanism for *detecting datanode failure* and ensuring that the link between datanodes and namenode is intact. In Hadoop , Name node and data node do communicate using **Heartbeat**. Therefore, Heartbeat is the signal that is sent by the datanode to the namenode after the regular interval of time to indicate its presence, i.e. to indicate that it is available.
- **The default heartbeat interval is 3 seconds**. If the DataNode in HDFS does not send heartbeat to NameNode in ten minutes, then NameNode considers the DataNode to be out of service and the Blocks replicas hosted by that DataNode to be unavailable.
- Hence once the heartbeat stops sending a signal to NameNode, then NameNode perform certain tasks such as replicating the blocks present in DataNode to other DataNodes to make the data is highly available and ensuring **data reliability**.
- NameNode that receives the Heartbeats from a DataNode also carries information like total storage capacity, the fraction of storage in use, and the number of data transfers currently in progress. For the NameNode’s block allocation and load balancing decisions, we use these statistics.

Speculative Execution

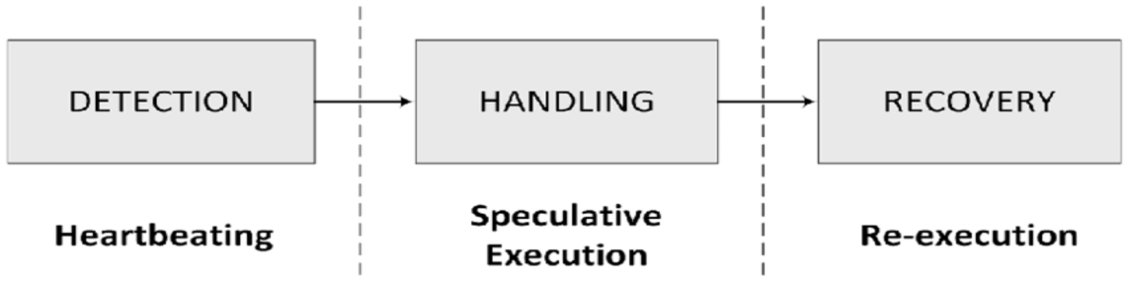
- In Hadoop, **MapReduce** breaks jobs into tasks and these tasks run parallel rather than sequential, thus reduces overall execution time. This model of execution is sensitive to slow tasks (even if they are few in numbers) as they slow down the overall execution of a job.
- There may be various reasons for the slowdown of tasks, including hardware degradation or software misconfiguration, but it may be difficult to detect causes since the tasks still complete successfully, although more time is taken than the expected time.
- The Hadoop framework does not try to diagnose or fix the slow-running tasks. The framework tries to detect the task which is running slower than the expected speed and launches another task, which is an equivalent task as a backup. The backup task is known as the speculative task, and this process is known as speculative execution in Hadoop.



- As the name suggests, Hadoop tries to speculate the slow running tasks, and runs the same tasks in the other nodes parallel. Whichever task is completed first, that output is considered for proceeding further, and the slow running tasks are killed.
- Firstly, all the tasks for the job are launched in Hadoop MapReduce. The speculative tasks are launched for those tasks that have been running for some time (at least one minute) and have

not made any much progress, on average, as compared with other tasks from the job. The speculative task is killed if the original task completes before the speculative task, on the other hand, the original task is killed if the speculative task finishes before it.

- In conclusion, we can say that Speculative execution is the key **feature of Hadoop** that improves job efficiency. Hence, it reduces the job execution time.



UNIT III: IBM Infosphere BigInsights

Introduction to Infosphere BigInsights – Open Source Ecosystems in IBM Infosphere BigInsights, IBM Proprietary Ecosystems in IBM Infosphere BigInsights, Layers of IBM Infosphere BigInsights, GPFS-FPO, Adaptive Map Reduce

Big Insights Web console security – Web Console Roles, Compression Techniques, Authentication Types

InfoSphere BigInsights 1.2 is a software platform designed to help firms discover and analyze business insights hidden in large volumes of a diverse range of data—data that's often ignored or discarded because it's too impractical or difficult to process using traditional means. Examples of such data include log records, click streams, social media data, news feeds, electronic sensor output, and even some transactional data. To help firms derive value from such data in an efficient manner, BigInsights incorporates several open source projects (including Apache™ Hadoop™) and a number of IBM-developed technologies.

BigInsights is a software platform for discovering, analyzing, and visualizing data from disparate sources. You use this software to help process and analyze the volume, variety, and velocity of data that continually enters your organization every day.

BigInsights helps your organization to understand and analyze massive volumes of unstructured information as easily as smaller volumes of information. The flexible platform is built on an Apache Hadoop open source framework that runs in parallel on commonly available, low-cost hardware. You can easily scale the platform to analyze hundreds of terabytes, petabytes, or more of raw data that is derived from various sources. As information grows, you add more hardware to support the influx of data.

By using BigInsights, users can extract new insights from this data to enhance knowledge of your business. For more information about the IBM Open Source Platform,

BigInsights incorporates tooling and value-add services for numerous users, speeding time to value and simplifying development and maintenance:

- Software developers can use the value-add services that are provided to develop custom text analytic functions to analyze loosely structured or largely unstructured text data.
- Data scientists and business analysts can use the data analysis tools within the value-add services to explore and work with unstructured data in a familiar spreadsheet-like environment.
- BigInsights provides distinct capabilities for discovering and analyzing business insights that are hidden in large volumes of data. These technologies and features combine to help your organization manage data from the moment that it enters your enterprise.
- Apache Hadoop helps enterprises harness data that was previously difficult to manage and analyze. BigInsights features Hadoop and its related technologies as a core component.

Figure 1 illustrates IBM's big data platform, which includes software for processing streaming data and persistent data. BigInsights supports the latter, while InfoSphere Streams supports the former. The two can be deployed together to support real-time and batch analytics of various forms of raw data, or they can be deployed individually to meet specific application objectives.

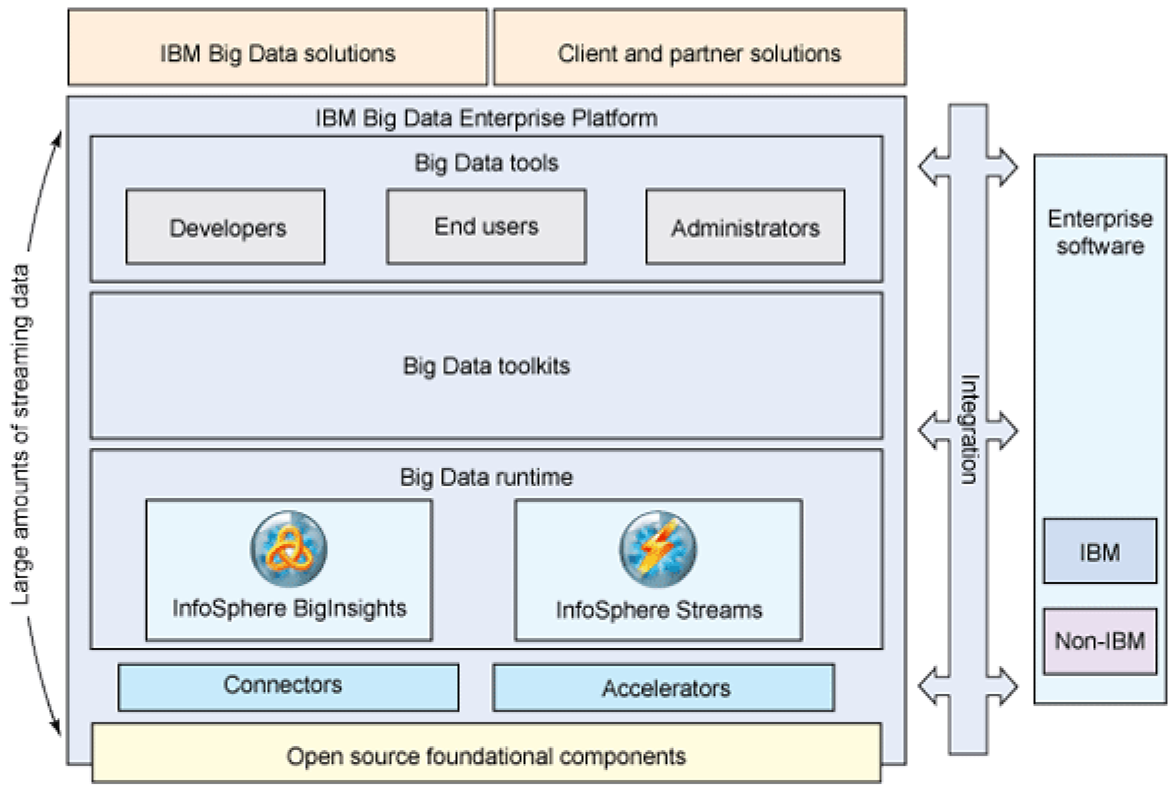


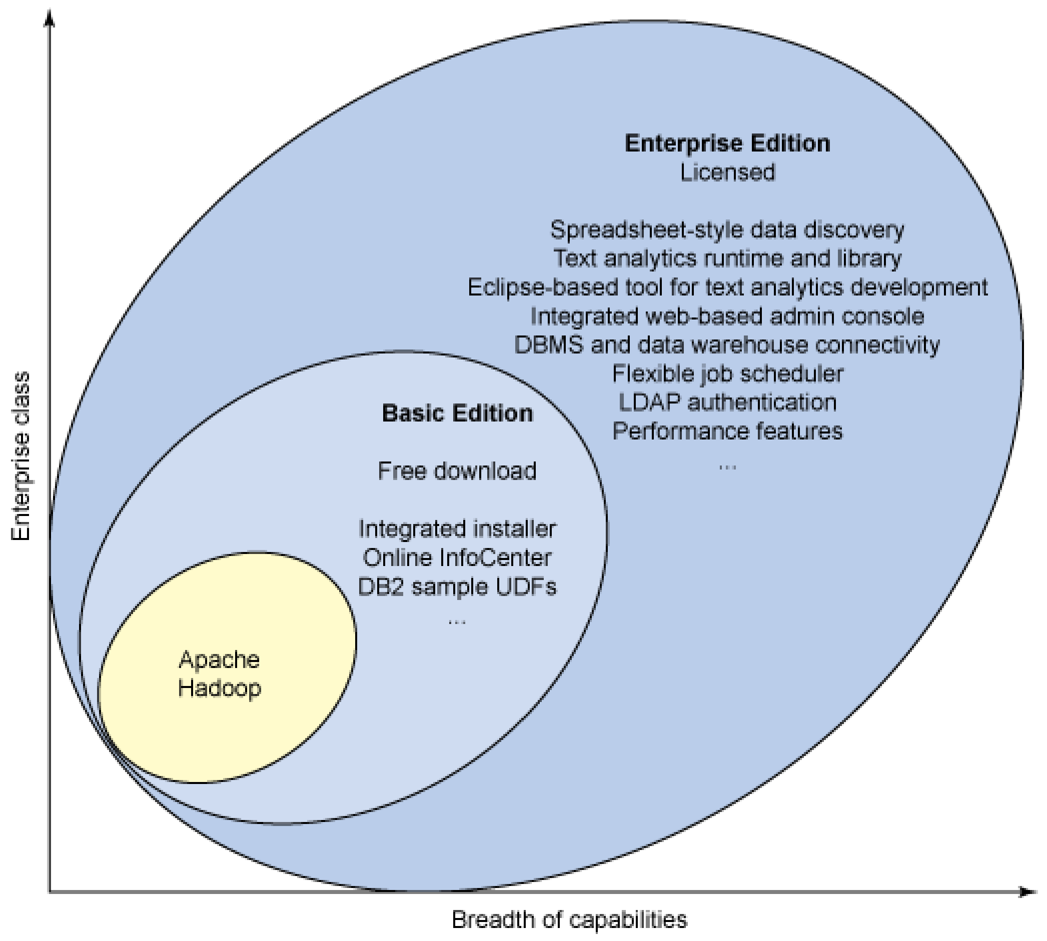
Figure 1

IBM developed BigInsights to help firms process and analyze the increasing volume, variety, and velocity of data of interest to many enterprises.

Basic and Enterprise Editions

By now, you may be wondering about the technologies that comprise the BigInsights platform. Several IBM and open source technologies are part of BigInsights, which is available in two editions: Basic and Enterprise. As shown in Figure 2, both editions include Apache Hadoop and other open source software, which are explained in more detail later.

Figure 2. InfoSphere BigInsights 1.2 Basic and Enterprise Editions



- Basic Edition is available for free download and can manage up to 10 TB of data. As such, it is suitable for pilot projects and exploratory work.

- The Enterprise Edition is a fee-based offering with no licensing restrictions on the quantity of data that can be managed. It includes all the features of the Basic Edition and offers additional analytic, administrative, and software integration capabilities. As such, Enterprise Edition is suitable for production applications.

- InfoSphere BigInsights is the IBM enterprise-grade Hadoop offering. It is based on industry-standard Apache Hadoop, but IBM provides extensive capabilities, including installation and management facilities and additional tools and utilities. Special care is taken to ensure that InfoSphere BigInsights is 100% compatible with open source Hadoop.

- The IBM capabilities provide the Hadoop developer or administrator with additional choices and flexibility without locking them into proprietary technology. Here are some of the standard open source utilities included with BigInsights 1.2 Basic and Enterprise Editions:

- **Apache Hadoop** (including the Hadoop Distributed File System (HDFS), MapReduce framework, and common utilities), a software framework for data-intensive applications that exploit distributed computing environments
- **Pig,** a high-level programming language and runtime environment for Hadoop
- **Jaql,** a high-level query language based on JavaScript Object Notation (JSON), which also supports SQL.
- **Hive,** a data warehouse infrastructure designed to support batch queries and analysis of files managed by Hadoop
- **HBase,** a column-oriented data storage environment designed to support large, sparsely populated tables in Hadoop
- **Flume,** a facility for collecting and loading data into Hadoop
- **Lucene,** text search and indexing technology
- **Avro,** data serialization technology
- **ZooKeeper,** a coordination service for distributed applications
- **Oozie,** workflow/job orchestration technology

IBM InfoSphere BigInsights provides advanced software capabilities that are not found in competing Hadoop distributions. Here are some of these capabilities:

- **Big SQL:** Big SQL is a rich, ANSI-compliant SQL implementation. Big SQL builds on 30 years of IBM experience in SQL and database engineering. Big SQL has several advantages over competing SQL on Hadoop implementations like SQL language compatibility, Support for native data sources, Performance etc. Unlike competing SQL-on-Hadoop implementations that introduce proprietary metadata or require that their own specific databases be deployed, Big SQL is open. Big SQL runs natively on existing Hadoop data sets in HDFS and uses existing Hadoop standards, such as the Hive metastore.

- **Big R:** Big R is a set of libraries that provide end-to-end integration with the popular R programming language that is included in InfoSphere BigInsights. Big R provides a familiar environment for developers and data scientists proficient with the R language. Big R provides analytic functions that mirror existing language facilities. Big R implements parallelized execution across the Hadoop cluster, avoiding the need for developers to code their own MapReduce logic. A significant feature of Big R is that it supports downloadable packages from the Comprehensive R Archive Network.

- **Big Sheets:** Big Sheets is a spreadsheet style data manipulation and visualization tool that allows business users to access and analyze data in Hadoop without the need to be knowledgeable in Hadoop scripting languages or MapReduce programming. Using built-in line readers, BigSheets can import data in multiple formats.

- **Application Accelerators:** IBM InfoSphere BigInsights extends the capabilities of open source Hadoop with accelerators that use pre-written capabilities for common big data use cases to build quickly high-quality applications. Here are some of the accelerators that are included in InfoSphere BigInsights:

  Text Analytics Accelerators: A set of facilities for developing applications that analyze text across multiple spoken languages

Machine Data Accelerators: Tools that are aimed at developers that make it easy to develop applications that process log files, including web logs, mail logs, and various specialized file formats

Social Data Accelerators: Tools to easily import and analyze social data at scale from multiple online sources, including tweets, boards, and blogs

NOTE: Not all capabilities that are available in InfoSphere BigInsights Enterprise Edition are supported on Linux for System z platforms.

- Adaptive MapReduce: An alternative, Hadoop-compatible scheduling framework that provides enhanced performance for latency sensitive Hadoop MapReduce jobs.
- IBM GPFS™ FPO: A variant of IBM GPFS that is POSIX-compliant and provides HDFS compatibility while providing Hadoop-style data locality by emulating the operation of the Hadoop NameNode.

Adaptive MapReduce:

Hadoop is now being used for files that are smaller in size. As queries are run against smaller data sets, programmers do not want to wait much time for results of a query. The overhead of Apache MapReduce becomes more and more noticeable. MapReduce does not scale well in a heterogeneous environment (Heterogeneous environments are distributed systems with nodes that vast greatly in hardware). These are some computational issues with MapReduce programming. IBM's Big Insights Adaptive MapReduce is designed to address that type of problems.

The Adaptive MapReduce component can run distributed application services on a scalable, shared, heterogeneous grid. This low-latency scheduling solution supports sophisticated workload management capabilities beyond those of standard Hadoop MapReduce.

The adaptive MapReduce component can organize distributed services on a shared grid in response to dynamically changing workloads. This component combines a service-oriented application middleware (SOAM) framework, a low-latency task scheduler, and a scalable grid management infrastructure. This design ensures application reliability while also ensuring low-latency and high-throughput communication between clients and compute services.

Features of Adaptive MapReduce:

- **Low-Latency task scheduling** – It is the goal of AMR. Reduce the task scheduling from 30 seconds to real-time. The adaptive MapReduce approach allows a single task tracker to start the processing, after once starting it is to keep working on multiple tasks.
- **Faster performance for small job workloads** – Another goal of AMR is redesigning of shuffling algorithm. Redesigned shuffling algorithm is to skip disk spill before hand-off data to reduce phase.
  (Spilling happens at least once, when the mapper finished, because the output of the mapper should be sorted and saved to the disk for reducer processes to read it.)
- **High availability MapReduce framework** – Two tiers of workload and resource scheduling. In Apache MapReduce, the workload scheduling and the resource scheduling are two different layers and are tightly coupled. Adaptive MapReduce separates these two layers, and there is a concept of central site management of workloads. Therefore, no single point of failure for job tracker, high availability is possible.
- Run MapReduce service on available hosts without fixed hostname for job tracker.

- When coupled with InfoSphere BigInsights, the adaptive MapReduce component transparently provides improved performance at a lower cost of a variety of big data workload managers.
- Adaptive MapReduce is compatible with Apache MapReduce.

Adaptive MapReduce Architecture:

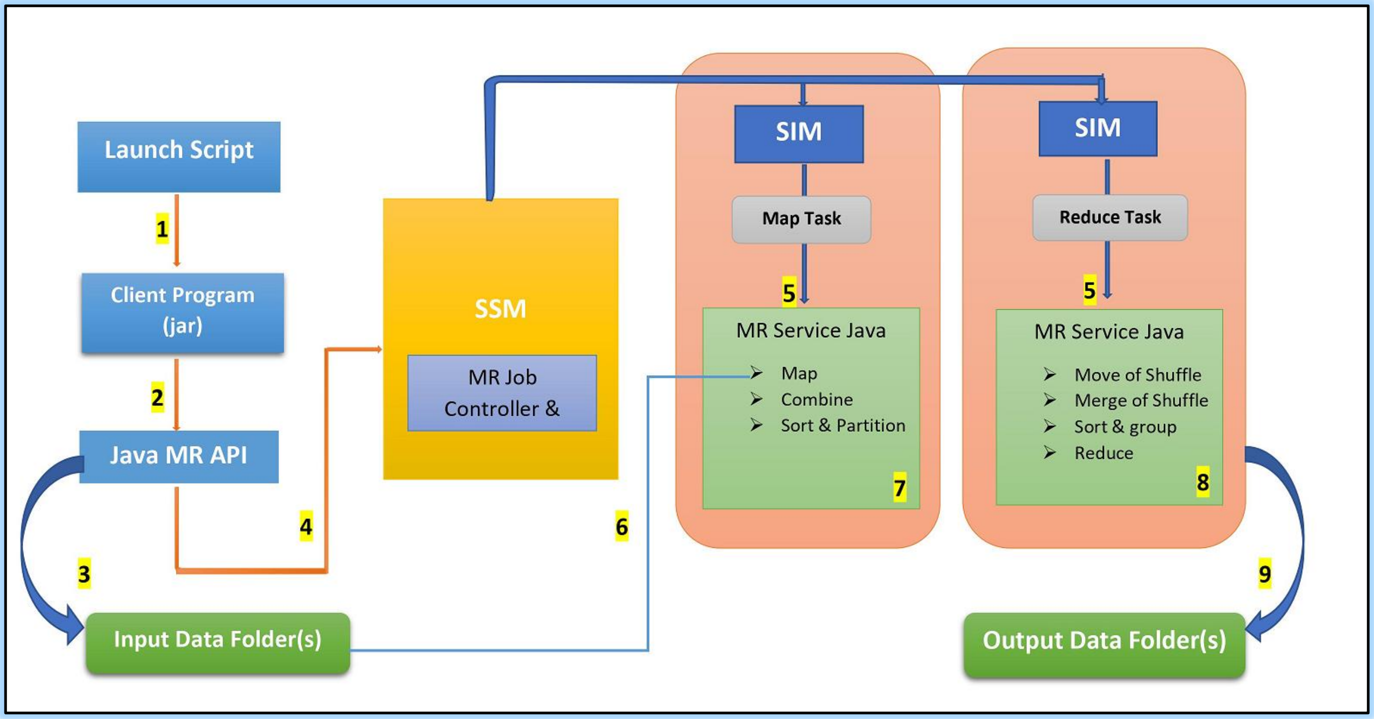
There is a resource scheduler, called EGO Master (Enterprise Grid Orchestrator). On each node in the cluster there is a small process called LIM (Load Information Manager). The LIM sends its current state back to the EGO master. Next there is a workload manager which can be dispatched to any node in the cluster pool. This sends MapReduce requests to the resource scheduler on plug-ins like SIM, J2EE etc. however with the current implementation of BigInsights, it is only taken advantage of scheduling mapreduce tasks.

With Adaptive MapReduce, the Job-Tracker is replaced with a Service Session Manager (SSM) and the Task-Trackers are replaced with Service Instance Manager (SIM). A task sent by client goes to through the java API and is sent to SSM. The task is then dispatched into the slots on the SIMs to process.

Adaptive Mapreduce architecture - Execution Details shown in the figure given below.

1. Mapreduce script is to launch for execution
2. A task sent in by the client goes through the Java Mapreduce API
3. Iterate the input files and create tasks based on file scripts
4. Create job and submit tasks with data locations and is sent to the Service Session Manager (SSM)
5. The task is then dispatched into the slots on the Service Instance Manager (SIM) Map task and reduce task will be implemented
6. Read the data in splits from input files to implement map task

7. Executes mapping task with – map function, combiner, sorting and partitioning
8. Executes reduce task with – shuffling, merging, grouping, and reducing
9. Generates the output.



Compression Techniques:

Even though Hadoop slave nodes are designed to be inexpensive, they are not free, and with large volumes of data that have a tendency to grow at increasing rates, compression is an obvious tool to control extreme data volumes. Data coming into Hadoop can be compressed. Mappers can output compressed data, reducers can read the compressed output from the map task and the reducers can be directed to write the results in compressed format. How much the size of the data decreases will depend on the chosen compression algorithm.

➢ File compression brings two major benefits:

It saves a great deal of storage space.

It also speeds up the transfer of the blocks throughout the clusters.

- A codec, which is a shortened form of compressor/decompressor, is technology (software or hardware, or both) for compressing and decompressing data; it's the implementation of a compression/decompression algorithm.
- There are many different compression algorithms and tools, and their characteristics and strengths vary. The most common trade-off is between compression ratios (the degree to which a file is compressed) and compress/decompress speeds.
- The Hadoop framework supports several codecs. The framework transparently compresses and decompresses most input and output file formats.
- The following list identifies some common codecs that are supported by the Hadoop framework. Be sure to choose the codec that most closely matches the demands of your particular use case (for example, with workloads where the speed of processing is important, chose a codec with high decompression speeds):

Gzip: A compression utility that was adopted by the GNU project, Gzip (short for GNU zip) generates compressed files that have a .gz extension. You can use the gunzip command to decompress files that were created by a few compression utilities, including Gzip.

- Provides High compression ratio.
- Uses high CPU resources to compress and decompress data.
- Good choice for Cold data which is infrequently accessed.
- Compressed data is not splittable and hence not suitable for MapReduce jobs.

Bzip2: From a usability standpoint, Bzip2 and Gzip are similar. Bzip2 generates a better compression ratio than does Gzip, but it's much slower. In fact, of all the available compression codecs in Hadoop, Bzip2 is by far the slowest. If you're setting up an archive that you'll rarely need to query and space is at a high premium, then maybe would Bzip2 be worth considering.

- Provides High compression ratio (even higher than GZIP).
- Takes long time to compress and decompress data.
- Good choice for Cold data which is infrequently accessed.
- Compressed data is splittable.
- Even though the compressed data is splittable, it is generally not suited for MR jobs because of high compression/decompression time.

Snappy: The Snappy codec from Google provides modest compression ratios, but fast compression and decompression speeds. (In fact, it has the fastest decompression speeds, which makes it highly desirable for data sets that are likely to be queried often.). The Snappy codec is integrated into Hadoop Common, a set of common utilities that supports other Hadoop subprojects. You can use Snappy as an add-on for more recent versions of Hadoop that do not yet provide Snappy codec support.

- Provides average compression ratio.
- Aimed at very fast compression and decompression time.
- Compressed data is not splittable if used with normal file like .txt
- Generally used to compress Container file formats like Avro and SequenceFile because the files inside a Compressed Container file can be split.

LZO: Like Snappy, LZO (short for Lempel-Ziv-Oberhumer, the trio of computer scientists who came up with the algorithm) provides modest compression ratios, but fast compression and decompression speeds. LZO is licensed under the GNU Public License (GPL). LZO supports splittable compression, which enables the parallel processing of compressed text file splits by your MapReduce jobs. LZO needs to create an index when it compresses a file, because with variable-length compression blocks, an index is required to tell the mapper where it can safely split the compressed file. LZO is only really desirable if you need to compress text files.

- Provides Low compression ratio.
- Very fast in compressing and decompressing data.
- Compressed data is splittable if an appropriate indexing algorithm is used.
- Best suited for MR jobs.

Hadoop Codecs

Codec	File Extension	Splittable?	Degree of Compression	Compression Speed
Gzip	.gz	No	Medium	Medium
Bzip2	.bz2	Yes	High	Slow
Snappy	.snappy	No	Medium	Fast
LZO	.lzo	No, unless indexed	Medium	Fast

Web Console Security:

The InfoSphere BigInsights security architecture includes authentication, roles and authorization levels, HTTPS support for the InfoSphere BigInsights console, and reverse proxy. By default Hadoop HTTP web-consoles (JobTracker, NameNode, TaskTrackers and DataNodes) allow access without any form of authentication.

Similarly to Hadoop RPC, Hadoop HTTP web-consoles can be configured to require Kerberos authentication using HTTP SPNEGO protocol (supported by browsers like Firefox and Internet Explorer). In addition, Hadoop HTTP web-consoles support the equivalent of Hadoop's Pseudo/Simple authentication. If this option is enabled, user must specify their user name in the first browser interaction using the user.name query string parameter.

For example: http://localhost:50030/jobtracker.jsp?user.name=babu.

Authorization schemas and role authorization are closely related. A minimal role authorization schema is created for you by the installation program that is based on the authentication schema that you select during installation. You can install InfoSphere® BigInsights™ with one of the following authentication schemas:

The three authentications are

1. PAM (Pluggable Authentication Module)
2. LDAP (Lightweight Directory Access Protocol)
3. Flat File

- PAM with flat file authentication
- PAM with LDAP authentication

Authentication:

End-User authentication: On any distributed system, authentication is the basic level of security control that validates a user's identity before allowing access to the system. InfoSphere BigInsights leverages pluggable authentication modules (PAMs) and provides support for the following authentication types through PAM: LDAP, Kerberos, or the flat-file authentication that is based on operating system security mechanisms. It is important to define users, groups, and roles according to your organization structure and job functions prior to installation.

PAM with flat file

- If you choose flat file authentication during installation, when a user accesses the InfoSphere BigInsights Console, the username and password are passed to the InfoSphere BigInsights.
- If you choose PAM with flat file authentication during installation, the password for the biadmin user must be the password that you enter in the Secure Shell panel of the InfoSphere BigInsights installation program. When you choose this option for security, you can either install sample groups and users or specify the groups and users individually. If you install the sample groups and users, the installation program generates the groups and users and maps the roles automatically. The Example section that follows the Procedure lists the default groups and user roles.
- If you specify the users and groups individually, then you must create users and groups individually, set the password for each user, and assign each user to groups.

PAM with LDAP

If you choose LDAP authentication during installation, you configure the InfoSphere BigInsights Console to communicate with an LDAP credential store for users, groups, and user to group mappings.

LDAP (Lightweight Directory Access Protocol):

- ➢ LDAP stands for **"Lightweight Directory Access Protocol"**. It is a simplification of the X.500 Directory Access Protocol (DAP) used to access directory information. In simple words, it is a hierarchical database in which data is stored in tree like structure where leaf node holds actual data.

- A directory is essentially a special-purpose database optimized to handle identity-related information. The overall structure of any particular directory is defined by its schema, much like a database schema defines the tables and columns.
- If you are using LDAP authentication method, then during the installation of BigInsights you need to specify the connection information for your LDAP server.
- The BigInsights administration user that you intend to create must already be defined in the LDAP server as well as in web-console roles.
- Some familiar products which uses LDAP are Microsoft Active Directory, IBM Tivoli Server, Oracle Directory, JNDI, Open LDAP.
- Business computer networks employ the Lightweight Directory Access Protocol to distribute lists of information organized into hierarchies. To access a network's LDAP services, your computer must first log in to a server that supports the protocol, a process called authentication. LDAP lets a network administrator assign different levels of access to its many users, keeping the information secure.
- LDAP defines a "Bind" operation that authenticates the LDAP connection and establishes a security context for subsequent operations on that connection. LDAP client send the username (as a LDAP distinguished name) and password (in clear text) to the LDAP server. The LDAP server looks up the object with that username in the directory, compares the password provided to the password(s) stored with the object, and authenticates the connection if they match. Because the password is provided in clear text, LDAP simple Binds should only be done over a secure TLS connection.

PAM (Pluggable Authentication Module):

- Pluggable authentication modules are a common framework for authentication and security. A pluggable authentication module (PAM) is a mechanism to integrate multiple low-level authentication schemes into a high-level application programming interface (API). It allows programs that rely on authentication to be written independent of the underlying authentication scheme.
- PAM uses a pluggable, modular architecture, which affords the system administrator a great deal of flexibility in setting authentication policies for the system.
- PAM authentication enables system users to be granted access and privileges transparently, so that applications do not need to contain information about where and how user data is stored.
- If you are using PAM authentication methods, it allows you to authenticate using a Linux shadow password file or by communicating with an LDAP server. If you using a Linux shadow password file, then the BigInsights administration user that you intend to create must be defined in **/etc/shadow** along with the encrypted password. The groups that associated with the web console roles must be defined in **/etc/groups** file.
- The purpose of PAM is to separate the development of privilege granting software from the development of secure and appropriate authentication schemes. This is accomplished by providing a library of functions that an application may use to request that a user be authenticated. When a program needs to authenticate a user, PAM provides a library containing the functions for the proper authentication scheme. Because this library is loaded dynamically, changing authentication schemes can be done by simply editing a configuration file.
- Flexibility is one of PAM's greatest strengths, PAM can be configured to deny certain programs the right to authenticate users, to only allow certain users to

be authenticated, to warn when certain programs attempt to authenticate, or even to deprive all users of login privileges. PAM's modular design gives you complete control over how users are authenticated.

Add users and groups to your security configuration.

Option	Description
LDAP	Add users and user groups to your LDAP configuration. Ensure that the full distinguished name for each user is mapped to the group in LDAP.
Flat file	To add users and associated passwords to the $*BIGINSIGHTS_HOME*/conf/install.xml file, run the following commands: `adduser user1` `passwd user1` Run the $*BIGINSIGHTS_HOME*/console/bin/refresh_security_config.sh script.

1. To map users or groups to roles, from the $*BIGINSIGHTS_HOME*/conf directory, edit the install.xml file. In the<security> section, each InfoSphere BigInsights security role is represented as a node. For the user roles to which you want to map users, add the <group> and <user> sections to the user roles.
2. To restart the InfoSphere BigInsights Console, run the following commands:

```
./stop.sh console
./start.sh console
```

3. If you mapped users or groups to roles by modifying the install.xml file, from the $*BIGINSIGHTS_HOME*/console/bin/directory, run the **refresh_security_config.sh** script.

4. Verify that the groups and roles are associated with each user in the InfoSphere BigInsights Console. In a secure environment, you can log in to obtain the session ID and then pass that ID to the subsequent REST URI.

Web Console Roles:

InfoSphere® BigInsights™ supports four predefined roles. During installation, you can map users and groups in your enterprise to the four InfoSphere BigInsights roles.

Roles

A *role* defines a set of user privileges and determines the actions that a user can complete, including the data that a user can see. A user can be associated with a role directly or indirectly by being a member of a group that is assigned to a role.

The InfoSphere BigInsights console uses the following default roles:

BigInsights System Administrator

Completes all system administration tasks, such as monitoring cluster health and adding, removing, starting, and stopping nodes.

BigInsights Data Administrator

Completes all data administration tasks, such as creating directories, running Hadoop file system commands, and uploading, deleting, downloading, and viewing files.

BigInsights Application Administrator

Completes all application administration tasks, such as publishing and un publishing (deleting) applications, deploying and un-deploying applications to a cluster, configuring icons, applying application descriptions, changing runtime libraries and categories of applications, and assigning application permissions to a group.

BigInsights User Administrator

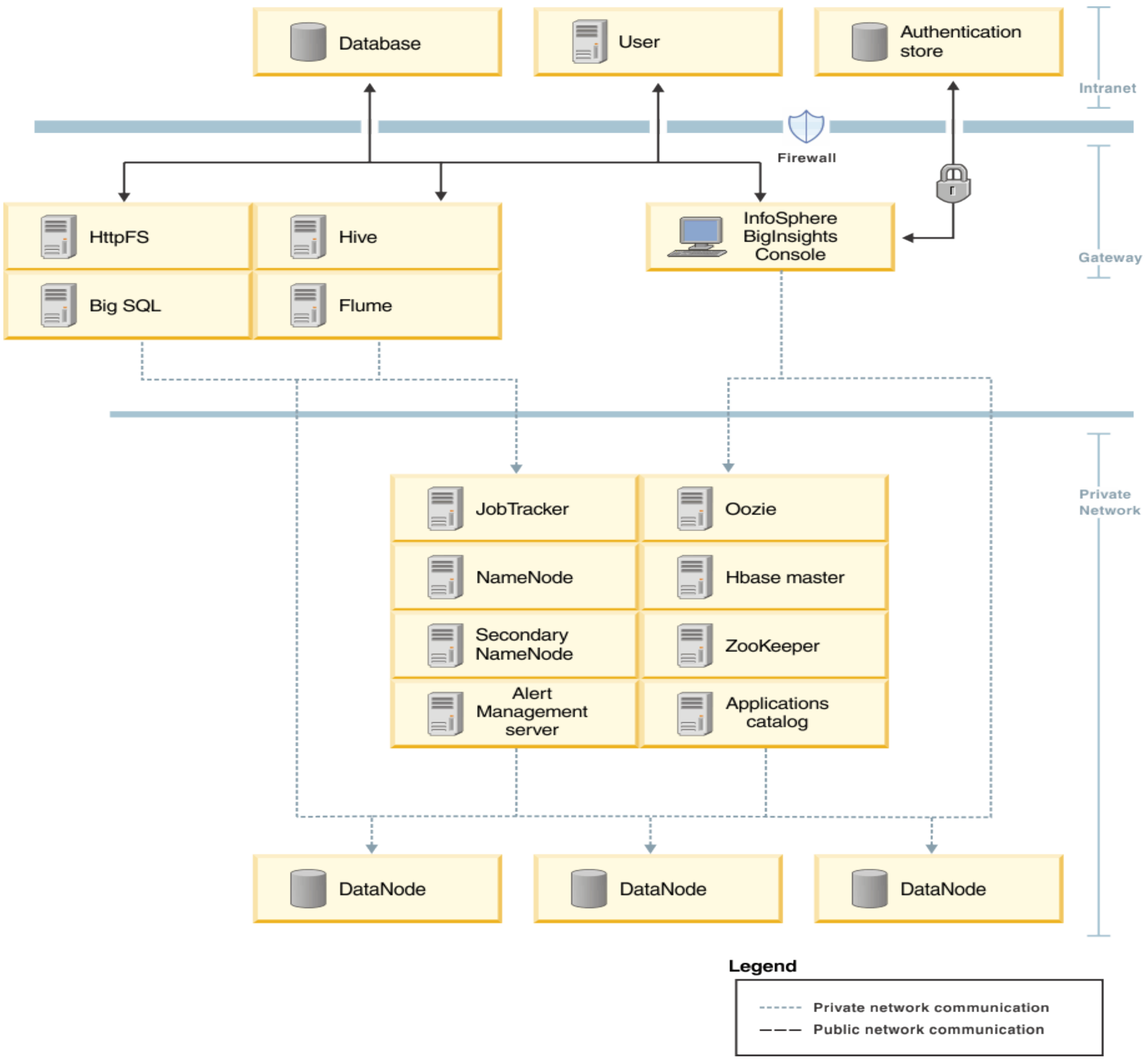
When a user who is authorized to run an application is logged into a secure InfoSphere BigInsights Console, the application workflows are submitted under the user ID and primary group ID of that user. The InfoSphere BigInsights Console verifies the roles of the current user to verify that the user is authorized to complete certain actions.
If you are using the HDFS file system, and the user is authorized to browse that file system, all existing files and directories are listed. However, file-level access control lists (ACLs) determine whether the user can read or write file contents. A data administrator can use the **hadoop fs -chmod ...** command to change HDFS ACLs.

Groups

A *group* associates a set of users who have similar business responsibilities. Groups separate users at the business level rather than the technical level. You might define one or more groups that associate users into logical sets based on business needs and constraints.

Users

A *user* is an entity that can be authenticated and typically represents an individual user. Each user authenticates to the system by using credentials, including a user name and password, and might belong to one of more groups. A user inherits the roles associated with all groups of which that user is a member.



<u>GPFS (General Parallel File System):</u>

IBM General Parallel File System (GPFS) is an IBM's parallel cluster file system. It is high scalable, high performance file management infrastructure for AIX, Linux and Windows systems.

<u>FPO (File Placement Optimization):</u>

GPFS is basically storage file system developed as a SAN (Storage Area Network) file system. Being a storage system, one cannot attach it directly with Hadoop system that makes a cluster. To do this IBM FPO (File placement optimization) comes in picture and bridge the gap. FPO is essentially emulation of key component of HDFS which is moving the workload from the application to data. Basically, it moves the job to Data instead of moving the Data to job.

GPFS Features:

- It is adaptable to many user environments by supporting a wide range of basic configurations and disk technologies.
- A highly available cluster architecture, Concurrent shared disk access to a global namespace.
- Having the capabilities for high performance parallel workload.
- GPFS provides safe, high bandwidth access using the POSIX I/O API.
- Provides good performance for large volume, I/O intensive jobs.
- It works best for large record, sequential access patterns, has optimizations for other patterns.
- Converting to GPFS does not require application code changes provided the code works in a POSIX compatible environment.
- No special nodes are required like name node in HDFS. It is easy to add/remove nodes and storage on the fly and rolling upgrades. It has integrated tiered storage, administer from any node is possible.

Features	GPFS	HDFS
Hierarchical storage management	Allows sufficient usage of disk drives with different performance characteristics	
High performance support for MapReduce applications	Stripes data across disks by using meta-blocks, which allows a MapReduce split to be spread over local disks.	Places a MapReduce split on one local disk
High performance support for traditional applications	▪ Manages metadata by using the local node when possible rather than reading metadata into memory unnecessarily ▪ Caches data on the client side to increase throughput of random reads ▪ Supports concurrent reads and writes by multiple programs ▪ Provides sequential access that enables fast sorts, improving performance for query languages such as Pig and Jaql	
High availability	Has no single point of failure because the architecture supports the following attributes: ▪ Distributed metadata ▪ Replication of both metadata and data ▪ Node quorums ▪ Automatic distributed node failure recovery and reassignment	Has a single point of failure on the NameNode, which requires it to run in a separate high availability environment

POSIX compliance	Is fully POSIX compliant, which provides the following benefits: ▪ Support for a wide range of traditional applications ▪ Support for UNIX utilities, that enable file copying by using FTP or SCP ▪ Updating and deleting data ▪ No limitations or performance issues when using a Lucene text index	Is not POSIX compliant, which creates the following limitations: ▪ Limited support of traditional applications ▪ No support of UNIX utilities, which requires using the **hadoop dfs get** command or the **put** command to copy data ▪ After the initial load, data is read-only ▪ Lucene text indexes must be built on the local file system or NFS because updating, inserting, or deleting entries is not supported
Ease of deployment	Supports a single cluster for analytics and databases	Requires separate clusters for analytics and databases

Unit – IV

Cluster Setup, Configuration and Administration of a Hadoop – Capacity Planning, Configuration Files, Configuration Management.
Oozie: Workflow, Control Flow Nodes, Expression Language Constants and Functions, Workflow EL Functions, Hadoop EL Constants and Functions, Workflow job, Oozie Coordinator System

Hadoop Cluster Set UP:

- "A hadoop cluster is a collection of independent components connected through a dedicated network to work as a single centralized data processing resource"
- "A hadoop cluster can be referred to as a computational computer cluster for storing and analyzing big data (structured, semi-structured and unstructured) in a distributed environment."
- "A computational computer cluster that distributes data analysis workload across various cluster nodes that work collectively to process the data in parallel."

- There are some parameters in setting up a BigInsights cluster.

While setting up the cluster, we need to know the below starting points into consideration:

- What is the volume of data for which the cluster is being set? (For example, 100 TB.)
- What kind of applications are we execute in cluster? – this will also consider
  - What is the amount of data transfer between the nodes?
  - Are the applications computationally intensive?
  - Will I be accessing the most of my data?
- How the cluster environment grow over time? Will it be anticipated?
- The retention policy of the data. (For example, 2 years.)

- The kinds of workloads you have — CPU intensive, i.e. query; I/O intensive, i.e. ingestion, memory intensive, i.e. Spark processing.
- The storage mechanism for the data — plain Text/AVRO/Parque/Jason/ORC/etc. or compresses GZIP, Snappy. (For example, 30% container storage 70% compressed.)

Advantages of a Hadoop Cluster Setup

- As big data grows exponentially, parallel processing capabilities of a Hadoop cluster help in increasing the speed of analysis process. However, the processing power of a hadoop cluster might become inadequate with increasing volume of data. In such scenarios, hadoop clusters can scaled out easily to keep up with speed of analysis by adding extra cluster nodes without having to make modifications to the application logic.
- Hadoop cluster setup is inexpensive as they are held down by cheap commodity hardware. Any organization can setup a powerful hadoop cluster without having to spend on expensive server hardware.
- Hadoop clusters are resilient to failure meaning whenever data is sent to a particular node for analysis, it is also replicated to other nodes on the hadoop cluster. If the node fails, then the replicated copy of the data present on the other node in the cluster can be used for analysis.

Hadoop Cluster Architecture

A hadoop cluster architecture consists of a data center, rack and the node that executes the jobs. Data center consists of the racks and racks consists of nodes. A medium to large cluster consists of a two or three level hadoop cluster architecture that is built with rack mounted servers. Every rack of servers is interconnected through 1 gigabyte of Ethernet (1 GigE). Each rack level switch in a hadoop cluster is connected to a cluster level switch which are in turn connected to other cluster level switches or they uplink to other switching infrastructure.

Single Node Hadoop Cluster vs. Multi Node Hadoop Cluster

As the name says, Single Node Hadoop Cluster has only a single machine whereas a Multi-Node Hadoop Cluster will have more than one machine.

In a single node hadoop cluster, all the daemons i.e. Data Node, Name Node, Task Tracker and Job Tracker run on the same machine/host. In a single node hadoop cluster setup everything runs on a single JVM instance. The hadoop user need not make any configuration settings except for setting the JAVA_HOME variable. For any single node hadoop cluster setup the default replication factor is 1.

In a multi-node hadoop cluster, all the essential daemons are up and run on different machines/hosts. A multi-node hadoop cluster setup has a master slave architecture where in one machine acts as a master that runs the Name Node daemon while the other machines acts as slave or worker nodes to run other hadoop daemons. Usually in a multi-node hadoop cluster there are cheaper machines (commodity computers) that run the Task-Tracker and Data Node daemons while other services are run on powerful servers. For a multi-node hadoop cluster, machines or computers can be present in any location irrespective of the location of the physical server.

Apache Hadoop can be installed in 3 different modes of execution –

1. Standalone Mode – Single node hadoop cluster setup
2. Pseudo Distributed Mode – Single node hadoop cluster setup
3. Fully Distributed Mode – Multi-node hadoop cluster setup

Capacity Planning:

Apache Hadoop is an open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license. It supports the running of applications on large clusters of commodity hardware. We need an efficient, correct approach to build a large hadoop cluster with a large set of data having accuracy, speed. Capacity planning plays important role to decide choosing right hardware configuration for hadoop components.
Capacity planning usually flows from a top-down approach of understanding.

While planning the capacity of cluster:

➢ **Start small:** Like most significant projects, it is always better to start small and get an understanding of how to administer the Hadoop environment. In this context the considerations are:
    - How many nodes you need?
    - What is the capacity of each node, on the CPU side?
    - What is the capacity of each node, on the memory side?

➢ **Other requirements:** When you decide to go big or adding nodes, you must consider few requirements-
  - Where to locate the machines – the closer that they are to each other, the less network with impact your installation. On the other hand, if all machines at a single location then there should be a proper plan for disaster recovery.
  - Power requirements
  - Cooling requirements
  - Access to machines to change failed components
  - Stock of spare machines/disks etc.

➢ Disks and File System Considerations:

- ***No RAID*** (Redundant Array Independent Disks) in Hadoop for storage, because HDFS itself it is having RAID technology.
- **Data Nodes** are planned as ***JBOD*** (Just a Bunch of Ordinary Disks) technology. i.e., low commodity hardware.
- The type of **File system** that is recommended to use is ***EXT3*** (default FS for Linux). Make sure all the data nodes are using the same file system.
- Check with ***BIOS*** (Basic Input Output Setup) I/O system is running efficiently.
- Make sure that ***SATA*** (Serial Advanced Technology for Attachment) **drives** are running as SATA drives only but not IDE (Integrated Drive Electronics) simulation.
- The **data transfer** rate for disks needs to be above ***70 MB/sec range***.
- Use ***common naming convention*** for **disks.**
- While doing **disk partition** leave ***40GB of disk for root*** partition.

➢ Hardware Considerations:

Hardware for Master Nodes:

- No commodity hardware for master nodes, enterprise hardware is required.
- Reliable power supply
- Dual internet cards
- Large amount of RAM – each block average size is less than 200 bytes.

Secondary Name Node:

- Same machine as primary name node.
- It is important to have same memory size.
- Should be on separate machine unless the cluster is small (<10 nodes)

Job Tracker node:

- Run on a separate machine
- It can run on the name node machine in small cluster.
- If it fails, all the submitted jobs are lost.

Data Nodes:

- Data nodes configuration is depends on workload
- Significant processing machines are required - 2 quad-core 2 - 2.5GHz CPUs.
- Multiple terabytes of disk drives
- Memory capacity should be in range of 16 – 24 GB RAM with 4 X 1 TB SATA disks.
- Fast network card - Gigabit Ethernet is required.

➢ **Network Considerations:**

- Network transfer rate plays a key role.
- Slave nodes at minimum 1GB network cards.
- 1Gbit switches – make sure that not exceeding the total capacity of the switch.
- HDFS cluster will perform at the speed of slowest card.

➢ **OS(Operating System) Considerations:**

- BigInsights runs on 64 bit Linux distributions (SUSE and RHEL)
- Various tools required are:
  - ✓ xCAT (Extreme Cloud Administration Tool) for deployment and management of cluster.
  - ✓ RedHat Kickstart is for doing automatic unattended OS installation and configuration.
  - ✓ Debian Fully Automatic Installation allows for centralized deployment and configuration management.

Capacity calculations:

➢ **Data Nodes Requirements:** we can plan for commodity machines required for the cluster. The nodes that will be required depends on data to be stored/analyzed. The amount of space needed on data nodes depends on the following factors:

- Replication Factor (Generally Three)
- 25% of additional space to handle shuffle files.

- Estimated data growth rates
- Plan of having data requirements are 4 times the amount of your raw data size.

- **Hardware Requirement for Slave Nodes:** In general, when considering higher-performance vs lower performance components.
  "Save the money, buy more nodes!"
- The HDFS' configuration is usually set up to replicate the data 3 ways. So, you will need 3x the actual storage capacity for your data.

Ex:

Prob: Assume that your initial amount of data is 300TB and data growth rate is about 1TB per day. Each Data Node has 4 disks with 1TB each. Calculate how many data nodes are required?

Sol:

Current data size: 300TB

Data growth Rate: 1TB/day

Replication Factor: 3

So

Total Space required for Initial data is: 300TB * 4 = 1200TB

Extra space required in a year is: 365TB * 4 = 1460TB

Total raw data requirement is: 1200TB + 1460TB = 2660TB

No.of date nodes required are: total raw data requirement/disks per node

i.e 2660/4 = 665

therefore 665 data nodes are required.

Let's take an example:

*Say we have 70TB of raw data to store on a yearly basis (i.e. moving window of 1 year). So after compression (say, with Gzip) we will get 70 – (70 * 60%) = 28Tb that will multiply by 3x = 84, but keep 70% capacity: 84Tb = x * 70% thus x = 84/70% = 120Tb is the value we need for capacity planning.*

Number of nodes: Here are the recommended specifications for DataNode/TaskTrackers in a balanced Hadoop cluster:

12-24 1-4TB hard disks in a JBOD (Just a Bunch Of Disks) configuration (no RAID, please!)

multi-core CPUs, running at least 2-2.5GHz

So let's divide up the value we have in capacity planning by the number of hard disks we need in a way that makes sense: 120Tb/12 1Tb = 10 nodes.

Memory: Now let's figure out the memory we can assign to these tasks. By default, the tasktracker and datanode take up each 1 GB of RAM per default. For each task calculate mapred.child.java.opts (200MB per default) of RAM. In addition, count 2 GB for the OS. So say, having 24 Gigs of memory available, 24-2= 22 Gig available for our 14 tasks – thus we can assign 1.5 Gig for each of our tasks (14 * 1.5 = 21 Gigs).

Configuration Management:

- Configuration Management is the practice of **handling changes systematically** so that a system maintains its integrity over time. Configuration Management (CM) ensures that the current design and build state of the system is known, good & trusted; and does not rely on the knowledge of the development team.
- Each hadoop node in cluster has its own set of configuration files, and it possible to set a single configuration file that is used for all master and worker machines, that is files on each node will be configured.

- Hadoop scripts are sufficient to manage single configuration setup. Hadoop controls the scripts for managing hadoop which may be used to configure the management tools for maintaining the cluster.
- You should have **configuration management tools** to manage your machines configuration across the entire cluster. These tools are used to install software and run scripts. Such tools are:
  - **Chef** - The chef is a configuration management tool, provides a way to define infrastructure resources as a code. The user can manage the infrastructure through code rather than using a manual process. The chef can do the following automation tasks:
    - ✓ Application Deployment
    - ✓ Infrastructure Configuration
    - ✓ Network configuration management
  - **Puppet** - Puppet is a Configuration Management tool that is used for deploying, configuring, and managing servers. Puppet uses a Master Slave architecture in which the Master and Slave communicate through a secure encrypted channel. Following functions can be performed by the Puppet DevOps tool:
    - ✓ Scaling-up and scaling-down of machines dynamically
    - ✓ To define distinct configuration for every host by continuously checking
    - ✓ To provide centralized control over all machines,

    Both chef & puppet are configuration management tools used to install software and run scripts.
- BigInsights includes **syncconf.sh** file that is used to synchronize the configuration files.
- Configuration properties are specified in:

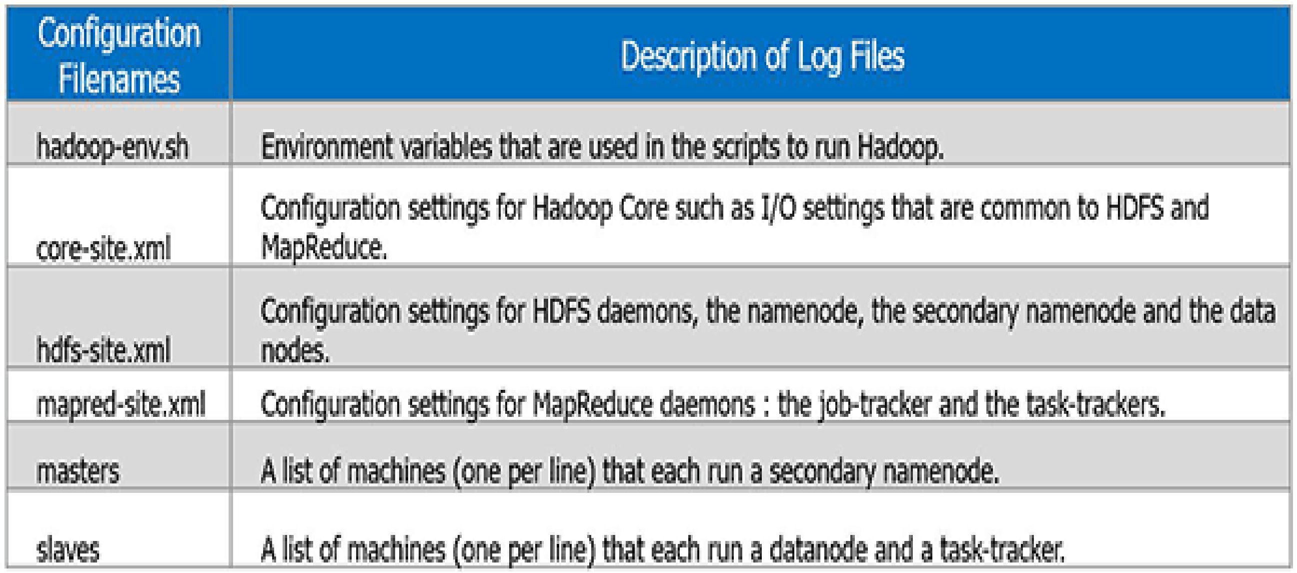
- core-site.xml
- hdfs-site.xml
- mapred-site.xml

Configuration Files:

Through Hadoop configuration files one can control how Hadoop is to run. configuration is driven by two types of important configuration files:

1. Read-only default configuration - src/core/core-default.xml, src/hdfs/hdfs-default.xml and src/mapred/mapred-default.xml.
2. Site-specific configuration - *conf/core-site.xml*, *conf/hdfs-site.xml* and *conf/mapred-site.xml*.

Hadoop Cluster Configuration Files: The following table lists the same. All these files are available under '**conf**' directory of Hadoop installation directory.



Configuration Filenames	Description of Log Files
hadoop-env.sh	Environment variables that are used in the scripts to run Hadoop.
core-site.xml	Configuration settings for Hadoop Core such as I/O settings that are common to HDFS and MapReduce.
hdfs-site.xml	Configuration settings for HDFS daemons, the namenode, the secondary namenode and the data nodes.
mapred-site.xml	Configuration settings for MapReduce daemons : the job-tracker and the task-trackers.
masters	A list of machines (one per line) that each run a secondary namenode.
slaves	A list of machines (one per line) that each run a datanode and a task-tracker.

Here is a listing of these files in the File System:

hadoop-env.sh

- This file specifies environment variables that affect the JDK used by **Hadoop Daemon** (bin/hadoop).
- As Hadoop framework is written in *Java* and uses *Java Runtime environment*, one of the important environment variables for Hadoop daemon **is $JAVA_HOME** in **hadoop-env.sh**. This variable directs Hadoop daemon to the *Java path* in the system.

The following three files are the important configuration files for the runtime environment settings of a Hadoop cluster.

core-site.xml

- This file informs Hadoop daemon where NameNode runs in the cluster.
- It contains the configuration settings for Hadoop Core such as I/O settings that are common to *HDFS* and *MapReduce* where *hostname* and *port* are the machine and port on which NameNode daemon runs and listens.
- It also informs the Name Node as to which IP and port it should bind. The commonly used port is 8020 and you can also specify IP address rather than hostname.

hdfs-site.xml

- This file contains the configuration settings for *HDFS daemons; the Name Node, the Secondary Name Node, and the data nodes.*
- You can also configure **hdfs-site.xml** to specify default block replication and permission checking on HDFS. The actual number of replications can also be specified when the file is created. The default is used if replication is not specified in create time.
- The value "true" for property **'dfs.permissions'** enables permission checking in HDFS and the value "false" turns off the permission checking.
- Typically, the parameters in this file are marked as final to ensure that they cannot be over written.

mapred-site.xml

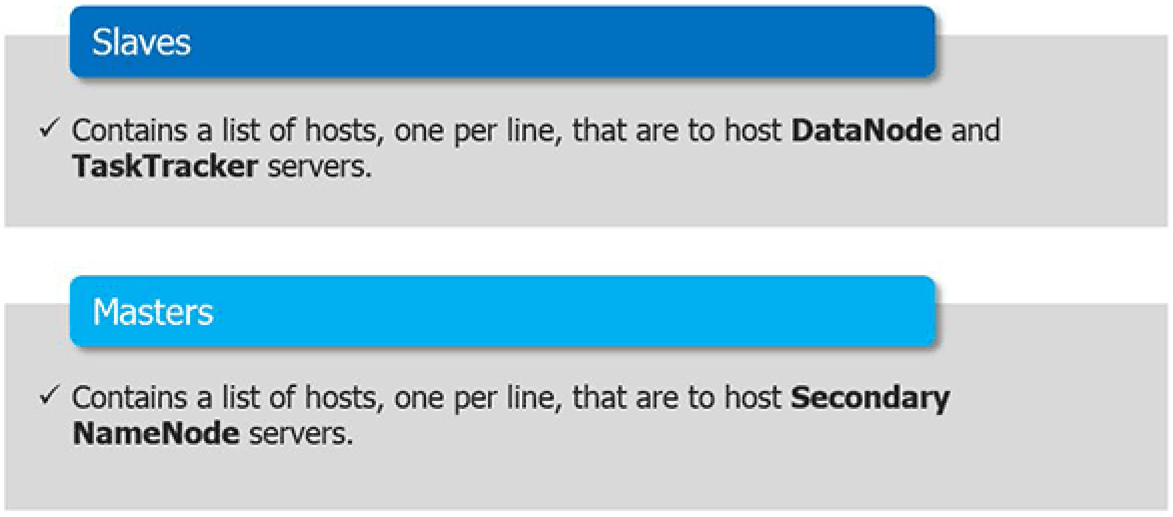
- This file contains the configuration settings for MapReduce daemons; *the job tracker and the task-trackers*.
- The **mapred.job.tracker** parameter is a *hostname* (or IP address) and *port* pair on which the Job Tracker listens for RPC communication. This parameter specifies the location of the Job Tracker to Task Trackers and MapReduce clients.

You can replicate all of the four files explained above to all the Data Nodes and Secondary Namenode. These files can then be configured for any node specific configuration e.g. in case of a different **JAVA_HOME** on one of the Datanodes.

The following two file 'masters' and 'slaves' determine the master and salve Nodes in Hadoop cluster.

Masters

This file informs about the Secondary Namenode location to hadoop daemon. The '**masters**' file at Master server contains a hostname Secondary Name Node servers.



Slaves

✓ Contains a list of hosts, one per line, that are to host **DataNode** and **TaskTracker** servers.

Masters

✓ Contains a list of hosts, one per line, that are to host **Secondary NameNode** servers.

The 'masters' file on **Slave Nodes** is blank.

Slaves

The '**slaves**' file at Master node contains a list of hosts, one per line, that are to host Data Node and Task Tracker servers. The '**slaves**' file on Slave server contains the IP address of the slave node. Notice that the 'slaves' file at Slave node contains only its own IP address and not of any other Data Nodes in the cluster.

The following two file 'hadoop-metric.properties' and 'log4j.properties' controls and determine the properties of metrics and log files.

hadoop-metric.properties: It specifies the properties for controlling metrics to be published in Hadoop.

log4j.properties: This file gives the properties for log files like name node audit log, task log etc.

Cluster Administration:

Rack Topology
Cluster status
Node Administration
Balancer,
Safe Mode at start up
Dashboards

Oozie

Glossary of Oozie terminology

Action

An execution/computation task (Map-Reduce job, Pig job, a shell command). It can also be referred as task or 'action node'.

Workflow

A collection of actions arranged in a control dependency DAG (Direct Acyclic Graph). "Control dependency" from one action to another means that the second action can't run until the first action has completed. Workflows can be defined by using workflow definition language, that is hPDL (An XML Process Definition Language) and are stored in a file called *workflow.xml.*

Workflow Definition

A programmatic description of a workflow that can be executed. Workflow definitions are written in hPDL (An XML Process Definition Language) and are stored in a file called *workflow.xml.*

Workflow Definition Language

The language used to define a Workflow Definition.

Workflow Job

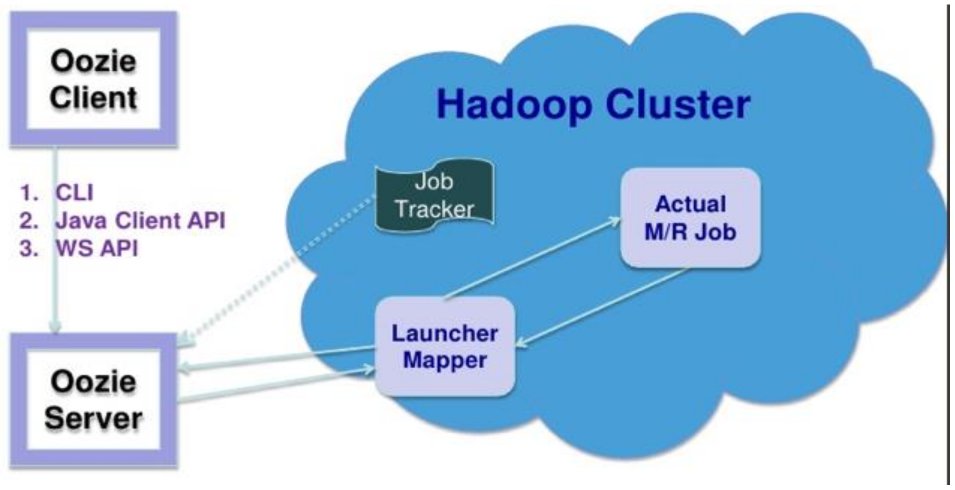
An executable instance of a workflow definition.

Workflow Engine

A system that executes workflows jobs. It can also be referred as a DAG engine.

Introduction:

- Apache Oozie is a scheduler system to run and **manage Hadoop jobs** in a distributed environment. It allows to combine multiple complex jobs to be run in a sequential order to achieve a bigger task.
- Using Apache Oozie you can also schedule your jobs in hadoop. Within a sequence of task, two or more jobs can also be programmed to run parallel to each other. It is a scalable, reliable and extensible system.
- Oozie is an open Source Java web-application, which is responsible for triggering the workflow actions. It in turn uses the Hadoop execution engine to execute the tasks.



Features of Oozie:

- Everything in Oozie is represented as a node in DAG (Direct Acyclic Graph) where a node can be visited only once.
- Oozie has client API and command line interface which can be used to launch, control and monitor job from Java application.
- Using its Web Service APIs one can control jobs from anywhere.
- Oozie has provision to execute jobs which are scheduled to run periodically.
- Oozie has provision to send email notifications upon completion of jobs.
- Apache Oozie detects the completion of tasks through callback and polling.

- When Oozie starts a task, it provides a unique callback HTTP URL to the task, and notifies that URL when the task is completed. If the task fails to invoke the callback URL, Oozie can poll the task for completion.

There are three types of jobs in Apache Oozie:

1. **Oozie Workflow Jobs**– These are Directed Acyclic Graphs (DAGs) which specifies a sequence of actions to be executed.
2. **Oozie Coordinator Jobs**– These consist of workflow jobs triggered by time and data availability.
3. **Oozie Bundles**– These can be referred as a package of multiple coordinator and workflow jobs.

1. Oozie Workflow

Workflow is a sequence of actions arranged in a Direct Acyclic Graph (DAG). The actions are dependent on one another, as the next action can only be executed after the output of current action. A workflow action can be a Pig action, Hive action, MapReduce action, Shell action, Java action etc. There can be decision trees to decide how and on which condition a job should run.

Oozie workflow consists of two nodes:

1. A**ction nodes**
2. C**ontrol-flow nodes**.

An **action node** represents a workflow task, e.g., moving files into HDFS, running a MapReduce, Pig or Hive jobs, importing data using Sqoop or running a shell script of a program written in Java. It triggers the execution of tasks running scripts or map reduce jobs. Action node specifies the type of task that is to be invoked, it could be a MapReduce job, streaming or piping script. Oozie supports two mechanisms in order to detect when a task completes.

Callback - oozie provides a unique callback URL to the task to notify that it is completed.

Polling – if for some reason the task fails to invoke the URL, it polls the task for completion.

A **control-flow node** controls the workflow execution between actions by allowing constructs like conditional logic wherein different branches may be followed depending on the result of earlier action node. **The following are the nodes consider in control nodes:**

- **Start Node** - designates start of the workflow job. The start node is the entry point for a workflow job, it indicates the first workflow node the workflow job must transitions to. When a workflow is started, it automatically transitions to the node specified in the start. A workflow definition must have one start node.

- **End Node –** It signals end of the job. The end node is the end for a workflow job, it indicates that the workflow job has completed successfully. If one or more actions started by the workflow job are executing when the end node is reached, the actions will be killed. A workflow definition must have one end node.

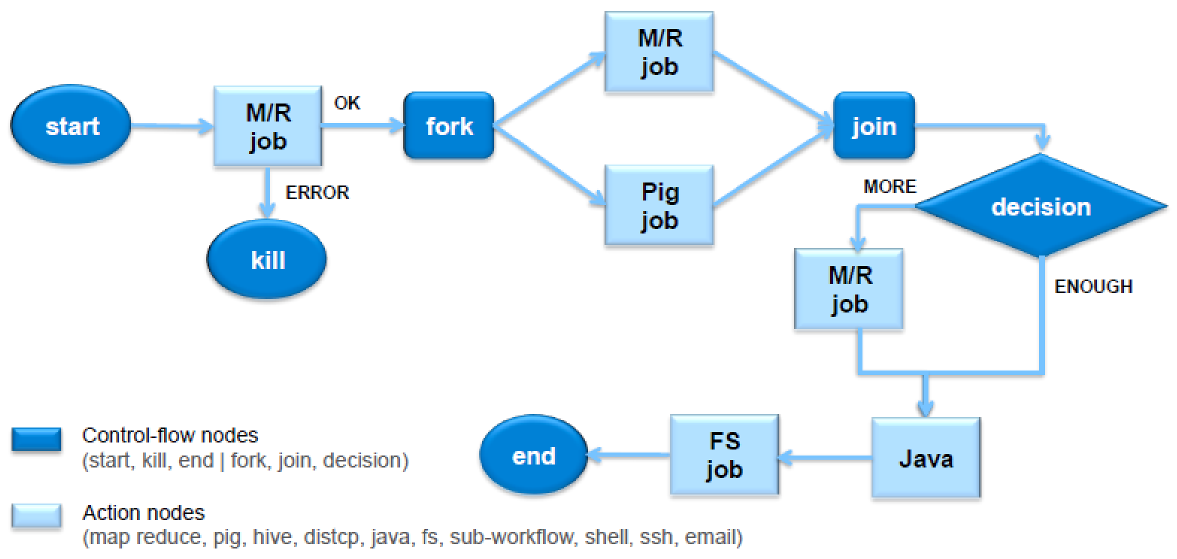
- **Kill Node** - designates an occurrence of error and corresponding error message to be printed. The kill node allows a workflow job to kill itself. When a workflow job reaches the kill it finishes in error (KILLED). A workflow definition may have zero or more kill nodes.

- **Decision node** - A decision node enables a workflow to make a selection on the execution path to follow. The behavior of a decision node can be seen as a switch-case statement. The default element in the decision node indicates the transition to take if none of the predicates evaluates to true and this is to avoid error state if none of the predicates evaluates to true.

- **fork and join** - fork splits execution path to multiple concurrent execution paths. join waits until all forks completed. If there is a fork node then there should be join node. The join node assumes concurrent execution paths are children of the same fork node.

At the end of execution of workflow, HTTP callback is used by Oozie to update client with the workflow status. Entry-to or exit-from an action node may also trigger callback.

Example Workflow Diagram


start
M/R job
OK
ERROR
kill
fork
M/R job
Pig job
join
decision
MORE
M/R job
ENOUGH
Java
FS job
end
Control-flow nodes (start, kill, end | fork, join, decision)
Action nodes (map reduce, pig, hive, distcp, java, fs, sub-workflow, shell, ssh, email)

2. Oozie Coordinator

You can schedule complex workflows as well as workflows that are scheduled regularly using Coordinator. Oozie Coordinators triggers the workflows jobs based on time, data or event predicates. The workflows inside the job coordinator starts when the given condition is satisfied.

Definitions required for the coordinator jobs are:

- **start**– Start date-time for the job.
- **end**– End date-time for the job.
- **timezone**– Time-zone of the coordinator application.
- **frequency**– The frequency, in minutes, for executing the jobs.

Some more properties are available for Control Information:

- **timeout**– The maximum time, in minutes, for which an action will wait to satisfy the additional conditions, before getting discarded. 0 indicates that if all the input events are not satisfied at the time of action materialization, the action should timeout immediately. -1 indicates no timeout, the action will wait forever. The default value is -1.
- **concurrency**– The maximum number of actions for a job that can run parallel. The default value is 1.
- **execution**– It specifies the execution order if multiple instances of the coordinator job have satisfied their execution criteria. It can be:

- FIFO (default)
- LIFO
- LAST_ONLY

3. Oozie Bundle

Oozie Bundle system allows you to define and execute a set of coordinator applications, often called a data pipeline. In aOozie bundle, there is no explicit dependency among the coordinator applications. However, you could use the data dependency of coordinator applications to create an implicit data application pipeline. You can start/stop/suspend/resume/rerun the bundle. It gives a better and easy operational control.

Kick-off-time − The time when a bundle should start and submit coordinator applications.

Expression Language:

Expression Language (EL) is mechanism that simplifies the accessibility of the data stored in Java bean component and other object like request, session and application, etc.

An EL function should be simple, fast, and robust. One of the reasons Oozie uses a MapReduce "launcher" job to run most of its actions is so these large and complicated programs are not executed directly in the Oozie server itself. However, always keep in mind that EL functions are executed in the Oozie server, so you don't want to overtax or threaten its stability with a poorly conceived one.

EL Functions and Constants:

Oozie provides a set of built-in EL functions and constants.

- **Basic EL Constants:**
  - KB: 1024 bytes, one Kilo Byte
  - MB: 1024 * KB, one Mega Byte
  - GB: 1024 * MB, one Giga Byte
  - TB: 1024 * GB, one Tera Byte
  - PB: 1024 * TB, one Peta Byte

➢ Basic EL Functions:

Some examples of “good” EL functions are ones that perform a simple string operation, arithmetic operation, or return some useful internal value from Oozie. A few of them are:

concat(String s1, String s2): It returns the concatenation of 2 strings. A string with null value is considered as an empty string.

replaceAll(String src, String regex, String replacement): Replace each occurrence of regular expression match in the first string with the replacement string and return the replaced string. A 'regex' string with null value is considered as no change. A 'replacement' string with null value is consider as an empty string.

trim(String s): It returns the trimmed value of the given string. A string with null value is considered as an empty string.

urlEncode(String s): It returns the URL UTF-8 encoded value of the given string. A string with null value is considered as an empty string.

timestamp( ): It returns the UTC current date and time in W3C format down to the second (YYYY-MM-DDThh:mm:ss.sZ). I.e.: 1997-07-16T19:20:30.45Z

toJsonStr(Map): It returns an XML encoded JSON representation of a Map. This function is useful to encode as a single property the complete action-data of an action, **wf:actionData(String actionName)** , in order to pass it in full to another action.

Hadoop EL Constants

- **RECORDS:** Hadoop record counters group name.
- **MAP_IN:** Hadoop mapper input records counter name.
- **MAP_OUT:** Hadoop mapper output records counter name.
- **REDUCE_IN:** Hadoop reducer input records counter name.
- **REDUCE_OUT:** Hadoop reducer input record counter name.
- **GROUPS:** 1024 * Hadoop mapper/reducer record groups counter name.

Workflow EL Functions:

wf:id( )- It returns the workflow job ID for the current workflow job.

wf:name( ) - It returns the workflow application name for the current workflow job.

wf:appPath( ) - It returns the workflow application path for the current workflow job.

wf:conf(String name) - It returns the value of the workflow job configuration property for the current workflow job, or an empty string if undefined.

wf:user( )- It returns the user name that started the current workflow job.

wf:group( ) - It returns the group/ACL for the current workflow job.

wf:errorCode(String node): It returns the error code for the specified action node, or an empty string if the action node has not exited with ERROR state.

Each type of action node must define its complete error code list.

wf:errorMessage(String message): It returns the error message for the specified action node, or an empty string if no action node has not exited
with ERROR state.

The error message can be useful for debugging and notification purposes.

wf:run( )- It returns the run number for the current workflow job,
normally 0 unless the workflow job is re-run, in which case indicates the current run.

Workflow Job:

Workflow jobs can be configured to make an HTTP GET notification upon start and end of a workflow action node and upon the completion of a workflow job. Oozie will make a best effort to deliver the notifications, in case of failure it will retry the notification a pre-configured number of times at a pre-configured interval before giving up.

Workflow Job Status Notification

If the oozie.wf.workflow.notification.url property is present in the workflow job properties when submitting the job, Oozie will make a notification to the provided URL when the workflow job changes its status.

If the URL contains any of the following tokens, they will be replaced with the actual values by Oozie before making the notification:

- $jobId : The workflow job ID
- $status : the workflow current state

A workflow job can have be in any of the following states:

PREP: When a workflow job is first create it will be in PREP state. The workflow job is defined but it is not running.

RUNNING: When a CREATED workflow job is started it goes into RUNNING state, it will remain in RUNNING state while it does not reach its end state, ends in error or it is suspended.

SUSPENDED: A RUNNING workflow job can be suspended, it will remain in SUSPENDED state until the workflow job is resumed or it is killed.

SUCCEEDED: When a RUNNING workflow job reaches the end node it ends reaching the SUCCEEDED final state.

KILLED: When a CREATED , RUNNING or SUSPENDED workflow job is killed by an administrator or the owner via a request to Oozie the workflow job ends reaching the KILLED final state.

FAILED: When a RUNNING workflow job fails due to an unexpected error it ends reaching the FAILED final state.

Oozie coordinator System

Commonly, workflow jobs are run based on regular time intervals and/or data availability. And, in some cases, they can be triggered by an external event.

The oozie coordinator system allows for recurring and dependent workflow jobs. This means that you can - using coordinator define triggers to invoke workflows on regular time intervals, data availability and for some limited external events. These definitions are defined in coordinateor.xml file.

It is also necessary to connect workflow jobs that run regularly, but at different time intervals. The outputs of multiple subsequent runs of a workflow become the input to the next workflow. For example, the outputs of last 4 runs of a workflow that runs every 15 minutes become the input of another workflow that runs every 60 minutes. Chaining together these workflows result it is referred as a data application pipeline.

The Oozie **Coordinator** system allows the user to define and execute recurrent and interdependent workflow jobs (data application pipelines).

Real world data application pipelines have to account for reprocessing, late processing, catchup, partial processing, monitoring, notification and SLAs.

Coordinator Application

A coordinator application is a program that triggers actions (commonly workflow jobs) when a set of conditions are met. Conditions can be a time frequency, the availability of new dataset instances or other external events.

Types of coordinator applications:

- **Synchronous:** Its coordinator actions are created at specified time intervals.

Coordinator applications are normally parameterized.

Coordinator Job

To create a coordinator job, a job configuration that resolves all coordinator application parameters must be provided to the coordinator engine.

A coordinator job is a running instance of a coordinator application running from a start time to an end time.

At any time, a coordinator job is in one of the following status:

PREP, RUNNING, PREPSUSPENDED, SUSPENDED, PREPPAUSED, PAUSED, SUCCEEDED, DONWITHERROR, KILLED, FAILED.

Valid coordinator job status transitions are:

- **PREP --> PREPSUSPENDED | PREPPAUSED | RUNNING | KILLED**
- **RUNNING --> SUSPENDED | PAUSED | SUCCEEDED | DONWITHERROR | KILLED | FAILED**
- **PREPSUSPENDED --> PREP | KILLED**
- **SUSPENDED --> RUNNING | KILLED**

- **PREPPAUSED --> PREP | KILLED**
- **PAUSED --> SUSPENDED | RUNNING | KILLED**

When a coordinator job is submitted, oozie parses the coordinator job XML. Oozie then creates a record for the coordinator with status **PREP** and returns a unique ID. The coordinator is also started immediately if pause time is not set.

When a user requests to suspend a coordinator job that is in **PREP** state, oozie puts the job in status **PREPSUSPEND** . Similarly, when pause time reaches for a coordinator job with **PREP** status, oozie puts the job in status **PREPPAUSED** .

Conversely, when a user requests to resume a **PREPSUSPEND** coordinator job, oozie puts the job in status **PREP** . And when pause time is reset for a coordinator job and job status is **PREPPAUSED** ,oozie puts the job in status **PREP** .

When a coordinator job starts, oozie puts the job in status **RUNNING** and start materializing workflow jobs based on job frequency.

When a user requests to kill a coordinator job, oozie puts the job in status **KILLED** and it sends kill to all submitted workflow jobs. If any coordinator action finishes with not **KILLED** ,oozie puts the coordinator job into **DONEWITHERROR** .

When a user requests to suspend a coordinator job that is in **RUNNING** status, oozie puts the job in status **SUSPEND** and it suspends all submitted workflow jobs.

When pause time reaches for a coordinator job that is in **RUNNING** status, oozie puts the job in status **PAUSED** .

Conversely, when a user requests to resume a **SUSPEND** coordinator job, oozie puts the job in status **RUNNING** . And when pause time is reset for a coordinator job and job status is **PAUSED** ,oozie puts the job in status **RUNNING** .

A coordinator job creates workflow jobs (commonly coordinator actions) only for the duration of the coordinator job and only if the coordinator job is in **RUNNING** status. If the coordinator job has been suspended, when resumed it will create all the coordinator actions that should have been created during the time it was suspended, actions will not be lost, they will delayed.

When the coordinator job materialization finished and all workflow jobs finish, oozie updates the coordinator status accordingly. For example, if all workflows are **SUCCEEDED**, oozie put the coordinator job into **SUCCEEDED** status.

However, if any workflow job finishes with not **SUCCEEDED** (e.g. **KILLED** or **FAILED** or **TIMEOUT**), oozie puts the coordinator job into **DONEWITHERROR** . If all coordinator actions are **TIMEDOUT** ,oozie puts the coordinator job into **DONEWITHERROR** .

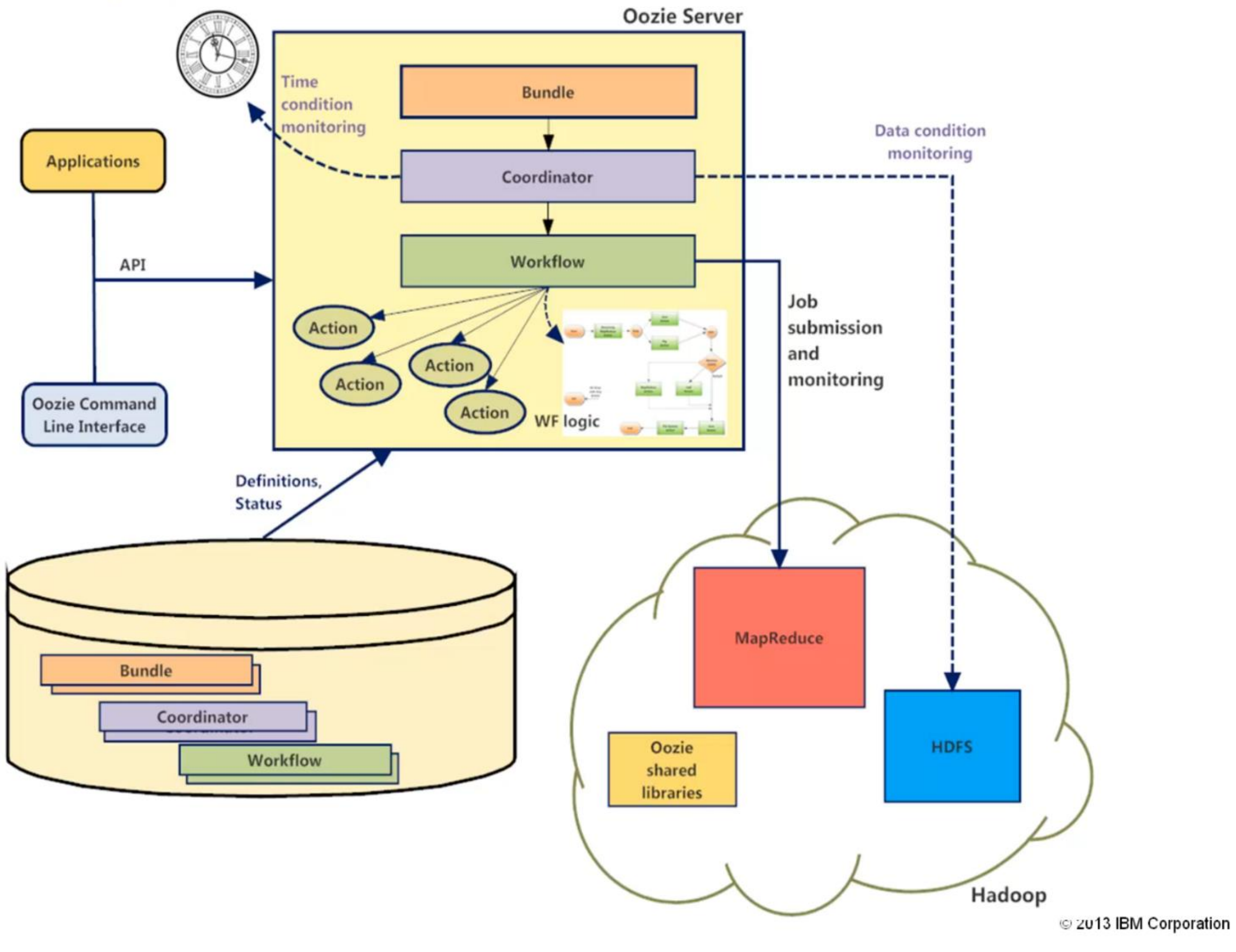
Coordinator Action

A coordinator job creates and executes coordinator actions.

A coordinator action is normally a workflow job that consumes and produces dataset instances.

Once an coordinator action is created (this is also referred as the action being materialized), the coordinator action will be in waiting until all required inputs for execution are satisfied or until the waiting times out.

Oozie components



Unit V: Schedulers & Moving Data into Hadoop

Managing Job Execution – Schedulers – FIFO, FAIR - equal share, minimum share, minimum share-no demand , minimum share exceeds available slots, minimum share less than fair share and weights

Moving Data into Hadoop – Data Load Scenarios – Data at Rest, Streaming Data, and Data from Data Warehouse, Sqoop , Data in Motion, Apache Flume, Flume Components.

Job Execution:

There are two types of nodes that control job execution.

- There is a single Job Tracker node which may run on the NameNode or may run on a node by itself.
- There are Task Tracker nodes run on the DataNodes.

Each job is broken up into a list of Map functions and Reduce functions. It is the job of the Task Tracker to schedule the jobs. Tasks are handed out to the Task Tracker, which are responsible for scheduling those tasks. Each Task Tracker is assigned with a number of slots which may vary from task tracker to task tracker. Each map function or reduce function consumes a slot when executing. Each task tracker informs to the job tracker the number of slots that are available. If a slot is available on a task tracker, then task tracker is able to accept a task for execution.

Schedulers:

Schedulers are used to control the number of task slots that can be consumed by a user group. Users submit jobs, these jobs invoke tasks on the data nodes and tasks can make use of task slots to execute. The job of scheduler is to see if any task slots are available on a datanode before sending a task to that datanode. Also scheduler can ensure that not all slots are consumed by a particular user of a group.

FIFO (First In First Out) scheduler:

The original scheduling algorithm that was integrated within the JobTracker was called *FIFO.* In FIFO scheduling, a JobTracker pulled jobs from a work queue, oldest job first. This schedule had no concept of the priority or size of the job, but the approach was simple to implement and efficient.

The default scheduler for Hadoop is FIFO scheduler. Jobs are executed based on priority in the sequence in which they were received.

Priorities in FIFO: There is a possibility in job execution that is a single job can execute long time and consumes the cluster. This leads to long waiting for the next jobs in sequence to execute. To avoid this, FIFO scheduler is assigned with five priority queues and job priority can be added via ***mapred.job.proiority*** file by using ***setJobPriority()*** method on job client.

1. VERY HIGH
2. HIGH
3. NORMAL
4. LOW
5. VERY LOW

Jobs in very high priority queue are executed before jobs in the high priority queue. Job scheduler can choose the next job to run based on highest priority. The problem with this approach is preemption is not supported.

FAIR scheduler

- The fair scheduler was developed by Facebook.
- It allows multiple users to run jobs on the cluster at the same time.

- The idea with this scheduler is that all jobs get on the average an equal share of resources over time. If there is just a single job running, then it can use the entire cluster.
- When other jobs are submitted, task jobs that are freed up are assigned to the new jobs. This concept of scheduling allows short jobs to finish in a reasonable amount of time without starving for long running jobs. The FAIR scheduler can also work with priorities.
- Jobs are assigned to pools and resources are shared fairly between the pools. By default there is a separate pool for each user so that each user gets some share of the cluster.

  - FAIR Scheduler with equal share
  - FAIR Scheduler with minimum share
  - FAIR Scheduler with minimum share, no demand
  - FAIR Scheduler with minimum share exceeds available slots
  - FAIR Scheduler with minimum share less than fair share
  - FAIR Scheduler with weights

Moving Data into Hadoop:

Data Load Scenarios:

There are different data load scenarios –

- Data at Rest
- Data in Motion
- Streaming Data
- Data from a web server
- Data from a Data Warehouse

Data at Rest:

In this scenario, the data is already generated into a file in some directories. Once the data is loaded into a file no additional updates will be done to the file and will be transferred over as it is saved.

HDFS commands for loading Data at Rest scenario:

cp

copyFromLocal

Data in Motion:

This is the data that is continuously updated. New data might be regularly added to these data sources and data will appended to a file or logs can be merged into a single log file. We need to have the capability of merging the files before copying them into Hadoop cluster.

Ex: Data from a web server such as Apache server etc.

Data in server logs or application logs like site browsing history

Twitter data, blogs, forums etc.

For basic Hadoop the open source product Flume could be used to load this type of scenario. Also there are several sample applications that come with BigInsights that can be used to load data in motion.

Flume is a great tool for collecting data from a web server and storing it into HDFS. Another option is to use Java Management Extension (JMX) commands.

Ex: Web Crawler, Board Reader

Streaming Data:

InfoSphere Streams process continuous data on the fly. This type of data is different from logging data. Although logging data might be continuously updated, it is also being stored on disk but there is not a sense of urgency when processing that data.

InfoSphere Streams is designed to work with data that continuously flows and theoretically may not have an end. So waiting for this data to write on to the disk might not make a sense. Once the appropriate analysis has been completed on the streaming data, we may decide some results should be kept for further analysis. For this reason **InfoSphere streams** is able to write its results into HDFS files.

Ex: Data from a security camera or patient monitoring system.

In case of patient monitoring data, data must be analyzed instantly that it gets generated. To wait for the data to be accumulated and then written to disk before starting analysis could be the difference between life and death.

Data from a Data warehouse:

This type of data loading scenario is dealing with the data that is from a data warehouse or any RDBMS. We have different methods load this type of data. One such technique is to export the data and then use Hadoop commands to import the data. A second technique that is having better performance is to use sqoop. **Sqoop** is a set of high performance open source connectors that can be customized for specific external connections. Even we can use BigSQL load to load the RDBMS data from DB2, Netizza and Teradata.

Sqoop:

Sqoop is an open source bidirectional data injection tool designed to transfer data between relational database systems and Hadoop. It uses JDBC to access the relational systems. Sqoop access the database in order to understand the schema of the database. It then generates a MapReduce application to import and export the data. When you use sqoop to import the data into Hadoop, sqoop generates a java class that encapsulates one row of the imported table. You have access to the source code for the generated class. This can allow you to quickly develop any other MapReduce applications that use the records that sqoop stored into HDFS.

Sqoop connection:

- ➢ Database connection requirements are same for import and export of data in sqoop.
- ➢ JDBC (Java Database Connectivity) connection string is required with the specifications – username. Password for sqoop import/export, connection path, username of DB2 user and password.

  Ex: sqoop import --connect jdbc:db2//your.db2.com:50000/yourDB \--username db2user –password db2password

Sqoop import

- ➢ The Sqoop import command is used to extract data from a RDBMS table and load the data into Hadoop.
- ➢ Each row in the table corresponds to a separate record in HDFS. The resulting data in HDFS can be stored as text files or binary files as well as imported directly into Hbase or Hive.

- Sqoop allows you to specify particular columns or a condition to limit the rows. You can write your own queries to access the relational data.

Ex: -sqoopimport –connect jdbc:db2://your.db2.com:50000/yourDB \ --username db2user –password db2password –table db2table\ --target-dir sqoopdta.

The above command will extract all rows and columns from a table db2table that is a DB2 database called yourDB. The results are stored in a directory in HDFs called sqoopdata. It is to connect to a DB2 system that listens to port 50000 and is running on a system with a host name as "your.db2.com". The connection is made with a userid and password.

Sqoop export

- The sqoop export command reads data in Hadoop and places into relational tables. The target table name must already exist to store the exported data.
- By default, sqoop ***inserts*** rows into the relational table. If there are any errors when doing the insert, the export process will fail.
- The other export mode to load the data is ***update mode***. It causes sqoop to generate update statements. To do updates you must specify the "update-key" argument.
- The third mode is ***call mode.*** With this mode, sqoop calls and passes the record to a stored procedure.

Ex: -sqoopexport –connect jdbc:db2://your.db2.com:50000/yourDB \ --username db2user –password db2password –table employee \ --export-dir /employeedata/processed.

Flume:

Flume is a distributed, reliable and available service for efficiently collecting, aggregating and moving large amounts of streaming data into HDFS. It is a great tool for collecting data from a web server and storing it into HDFS.
It can be used to collect data from sources and transfer to other agents.

Flume architecture:
Data generators (such as facebook, twitter) generate data which gets collected by individual flume agents running on them. Thereafter, a data collector (which is an agent) collects the data from the agents which is aggregated and stored into a centralized store such as HBase, HDFS.

An agent is an independent daemon process (JVM) in flume. It receives the data (events) from clients or other agents and forwards it to its next destination (sink/agent). Flume agent contains **three components namely source, channel and sink.**
Flume agents are installed at the sources of the data and at the target location. These agents then either externalize the data or pass the data to another flume agent as an event.

A source is a component which receives data from data generators and transfers it to one or more channels in the form of flume events. Flume supports several types of sources to receive events from a specified data generators.
Ex: Avro source, Thrift source

A channel is a transient store which receives the events from the source and buffers them till they are consumed by sinks. It acts as a bridge between the source and the sink.
Ex: JDBC channel. Memory channel, file system channel etc.

A sink stores the data into centralized stores like HDFS and HBase. It consumes the data (events) from the destination. The destination might be another agent or the central store.
Ex: HDFS sink