* **Why are the companies using big data this day?**
* Big Data helps companies to generate valuable insights with the various technologies it holds, Big Data helps almost every company or sector that aspires to grow.
* Analyzing large datasets that are associated with the events of the company can give them insights to increase their customer satisfaction.
* Companies use big data in their systems to improve operations, provide better customer service, create personalized marketing campaigns, and take other actions that, ultimately, can increase revenue and profits.
* **Explain real time big data project?**

Real-time data processing is the execution of data in a short time period, providing near-instantaneous output. The processing is done as the data is inputted, so it needs a continuous stream of input data in order to provide a continuous output. Good examples of real-time data processing systems are bank ATMs, traffic control systems and modern computer systems such as the PC and mobile devices. In contrast, a batch data processing system collects data and then processes all the data in bulk in a later time, which also means output is received at a later time. Real-time data processing is also known as stream processing.

Data will be ingested in to Object store using Apache NI FI in prquet format once the data is loaded in Object store will read the data and processing it using the pyspark. We implement pyspark application as per the mapping document will do joins, transformation and aggregations and generate the text file. This is the daily run. As per the requirement we maintain history table in ObjectStore to avoid sending duplicate data. This file will be sent to the destination through Airflow.

Daily we have a stand-up meeting in that meeting we discuss what we worked yesterday and what we will work today and any blockers or issues we have. My team leader will give the day-to-day task according that we will work. I developed the logic as per the business requirement using pyspark. Once I developed the logic will test it and run the job. Before running the job will check the data is present or not in our source database. Once the data is present with today's date will run the job and generate the output File. Once my development is done QA will be doing system testing. And then business will validate the file in staging. If any Issues/ Observation they find we need to work on it. Once all the observation are done will move this code to prod

* **What challenges the industry uses to face before big data?**
* Data silos and poor data quality.
* Lack of coordination to steer big
* Skills shortage.
* Solving the wrong problem.
* Dated data and inability to operationalize insights.
* **What is batch data?**

An efficient way of processing high/large volumes of data is what you call Batch Processing. It is processed, especially where a group of transactions is collected over a period of time In this process, At first, data is collected, entered and processed. Afterward, it produces batch results. We can say [**Hadoop**](https://data-flair.training/blogs/hadoop-tutorial-for-beginners/) works on batch data processing. For input, process, and output, batch processing requires separate programs. Payroll and billing systems are beautiful examples of batch processing.  
Let’s understand batch processing with some scenario. While sales team/employees would gather information throughout a specified period of time. Afterward, all that information would be entered into the system all at once. This whole procedure is known as Batch Processing. Generally, it works for printing shipping labels, packing slips and payment processing. In other words, this method also means waiting to do everything at once. Also, it means relying on the ability of your system to handle it all.  
We can say, the batch processing system

* Batch processing access to all data.
* It might compute something big and complex.
* Generally, it is very concerned with throughput. Rather than the latency of individual components of the computation.
* Batch processing has latency measured in minutes or more.

i. Advantages of Batch Processing

* Batch Processing is Ideal for processing large volumes of data/transaction. It also increases efficiency rather than processing each individually.
* Here, we can do processing independently. Even during less-busy times or at a desired designated time.
* For the organization by carrying out the process, it also offers cost efficiency.
* Also, allows a good audit trail.

ii. Disadvantages of Batch Processing

* The time delay between the collection of data and getting the result after the batch process.
* In the batch processing master file is not always kept up to date.
* Here, a one-time process can be very slow.
* **What is streaming/live data?**

A data stream is an unbounded sequence of data arriving continuously. Streaming divides continuously flowing input data into discrete units for further processing. Stream processing is low latency processing and analysing of streaming data.

**Spark Streaming** was added to Apache Spark in 2013, an extension of the core Spark API that provides scalable, high-throughput and fault-tolerant stream processing of live data streams. Data ingestion can be done from many sources like Kafka, [Apache Flume](https://data-flair.training/blogs/introduction-apache-flume-tutorial-beginners-guide/), Amazon Kinesis or TCP sockets and processing can be done using complex algorithms that are expressed with high-level functions like map, reduce, join and window. Finally, processed data can be pushed out to filesystems, databases and live dashboards.

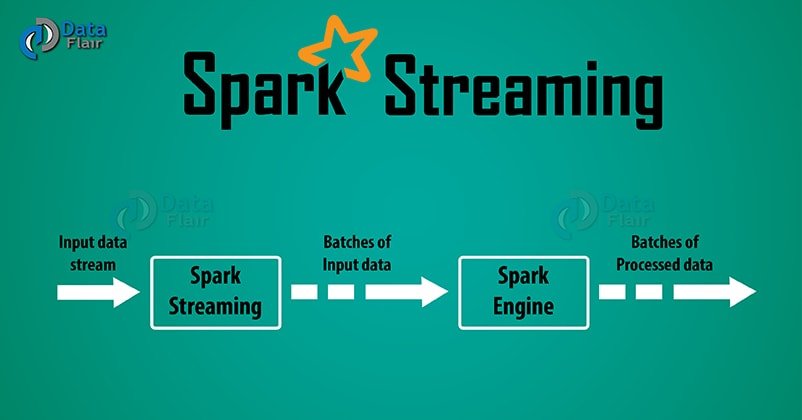
Its internal working is as follows. Live input data streams is received and divided into batches by Spark streaming, these batches are then processed by the Spark engine to generate the final stream of results in batches.

Its key abstraction is Apache Spark **Discretized Stream** or, in short, a **Spark**[DStream](https://data-flair.training/blogs/apache-spark-dstream-discretized-streams/), which represents a stream of data divided into small batches. DStreams are built on Spark [RDDs](https://data-flair.training/blogs/apache-spark-rdd-tutorial/), Spark’s core data abstraction. This allows Streaming in Spark to seamlessly integrate with any other Apache Spark components like Spark MLlib and [**Spark SQL**](https://data-flair.training/blogs/introduction-to-apache-spark-sql-tutorial/).

## **Need for Streaming in Apache Spark**

To process the data, most traditional stream processing systems are designed with a continuous operator model, which works as follows:

* Streaming data is received from data sources (e.g. live logs, system telemetry data, IoT device data, etc.) into some data ingestion system like Apache Kafka, Amazon Kinesis, etc.
* The data is then processed in parallel on a cluster.
* Results are given to downstream systems like [HBase](https://data-flair.training/blogs/hbase-tutorial-beginners-guide/), Cassandra, Kafka, etc.

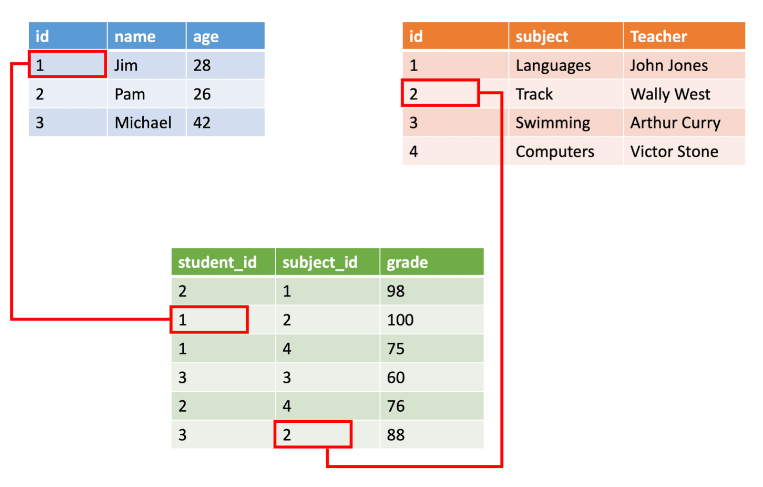
[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2017/03/apache-spark-streaming.jpg)

*Spark Streaming Tutorial for Beginners*

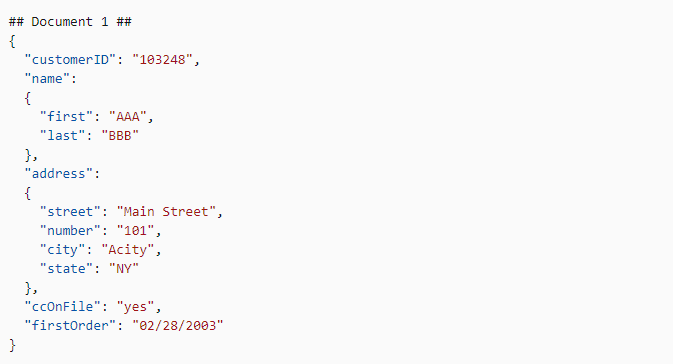
There is a set of worker nodes, each of which runs one or more continuous operators. Each continuous operator processes the streaming data one record at a time and forwards the records to other operators in the pipeline.

Data is received from ingestion systems via Source operators and given as output to downstream systems via sink operators.

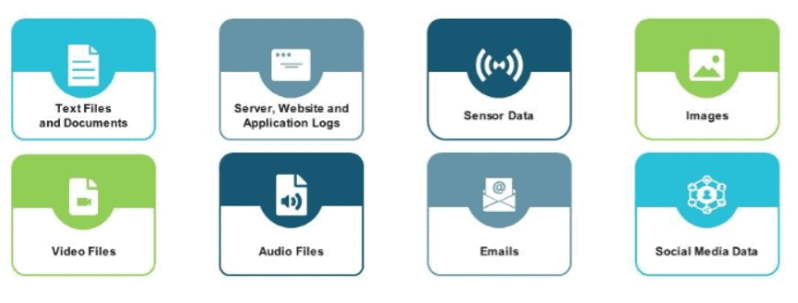
* **Structured Data**
* *Structured data* is generally tabular data that is represented by columns and rows in a database.
* Databases that hold tables in this form are called relational databases.
* The mathematical term “relation” specify to a formed set of data held as a table.
* In structured data, all row in a table has the same set of columns.
* SQL (Structured Query Language) programming language used for structured data.

****

* **Semi-structured Data**
* ***Semi-structured*** data is information that doesn’t consist of Structured data (relational database) but still has some structure to it.
* Semi-structured data consist of documents held in ***JavaScript Object Notation (JSON) format****.* It also includes ***key-value*** stores and ***graph*** databases.



* **Unstructured Data**
* **Unstructured data**is information that either does not organize in a pre-defined manner or not have a pre-defined data model.
* Unstructured information is a set of text-heavy but may contain data such as numbers, dates, and facts as well.
* **Videos, audio, and binary** data files might not have a specific structure. They’re assigned to as **unstructured** data.

****

In short, Structured data is stored is predefined format and is highly specific; whereas unstructured data is a collection of many varied data types which are stored in their native formats; while semi structured data that does not follow the tabular data structure models associated with relational databases or other data table forms.

* **What is ETL?**

**ETL** is a process that extracts the data from different source systems, then transforms the data (like applying calculations, concatenations, etc.) and finally loads the data into the Data Warehouse system. Full form of ETL is Extract, Transform and Load.

It’s tempting to think a creating a Data warehouse is simply extracting data from multiple sources and loading into database of a Data warehouse. This is far from the truth and requires a complex ETL process. The ETL process requires active inputs from various stakeholders including developers, analysts, testers, top executives and is technically challenging.

In order to maintain its value as a tool for decision-makers, Data warehouse system needs to change with business changes. ETL is a recurring activity (daily, weekly, monthly) of a Data warehouse system and needs to be agile, automated, and well documented.

* **What is Hadoop?**

Apache Hadoop is an open-source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyse massive datasets in parallel more quickly.

Hadoop consists of four main modules:

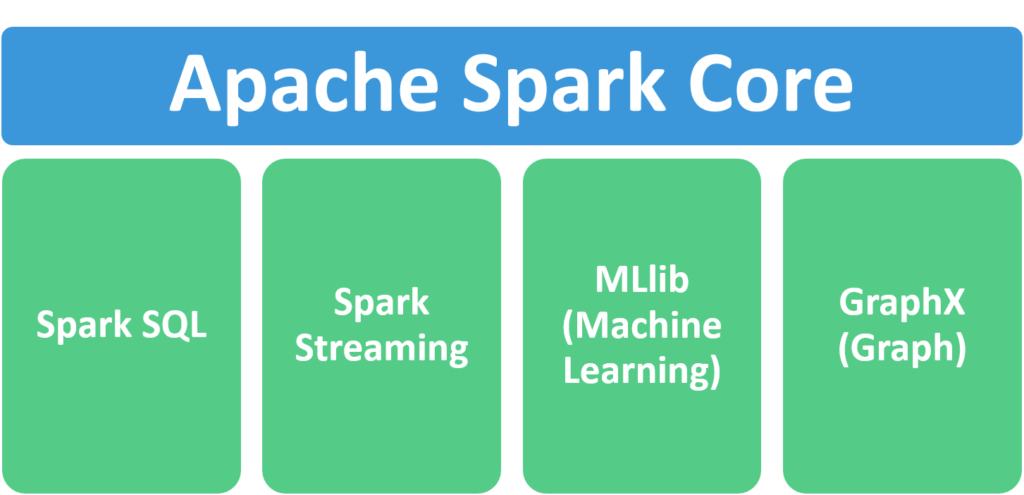
* Hadoop Distributed File System (HDFS) – A distributed file system that runs on standard or low-end hardware. HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets.
* Yet Another Resource Negotiator (YARN) – Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.
* MapReduce – A framework that helps programs do the parallel computation on data. The map task takes input data and converts it into a dataset that can be computed in key value pairs. The output of the map task is consumed by reduce tasks to aggregate output and provide the desired result.
* Hadoop Common – Provides common Java libraries that can be used across all modules.
* **What is Spark?**

Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching and optimized query execution for fast queries against data of any size. Simply put, Spark is a **fast and general engine for large-scale data processing**.

The **fast** part means that it’s faster than previous approaches to work with Big Data like classical [MapReduce](https://www.ibm.com/analytics/hadoop/mapreduce). The secret for being faster is that Spark runs on memory (RAM), and that makes the processing much faster than on disk drives.

The **general** part means that it can be used for multiple things like running distributed SQL, creating data pipelines, ingesting data into a database, running Machine Learning algorithms, working with graphs or data streams, and much more.

## [Components](https://chartio.com/learn/data-analytics/what-is-spark/#components)



1. **Apache Spark Core** – Spark Core is the underlying general execution engine for the Spark platform that all other functionality is built upon. It provides in-memory computing and referencing datasets in external storage systems.
2. **Spark SQL** – Spark SQL is Apache Spark’s module for working with structured data. The interfaces offered by Spark SQL provides Spark with more information about the structure of both the data and the computation being performed.
3. **Spark Streaming** – This component allows Spark to process real-time streaming data. Data can be ingested from many sources like Kafka, Flume, and HDFS (Hadoop Distributed File System). Then the data can be processed using complex algorithms and pushed out to file systems, databases, and live dashboards.
4. **MLlib (Machine Learning Library)** – Apache Spark is equipped with a rich library known as MLlib. This library contains a wide array of machine learning algorithms- classification, regression, clustering, and collaborative filtering. It also includes other tools for constructing, evaluating, and tuning ML Pipelines. All these functionalities help Spark scale out across a cluster.
5. **GraphX** – Spark also comes with a library to manipulate graph databases and perform computations called GraphX. GraphX unifies ETL (Extract, Transform, and Load) process, exploratory analysis, and iterative graph computation within a single system.

## [**Features**](https://chartio.com/learn/data-analytics/what-is-spark/#features)

1. **Fast processing** – The most important feature of Apache Spark that has made the big data world choose this technology over others is its speed. Big data is characterized by volume, variety, velocity, and veracity which needs to be processed at a higher speed. Spark contains [Resilient Distributed Dataset (RDD)](https://intellipaat.com/blog/tutorial/spark-tutorial/programming-with-rdds/) which saves time in reading and writing operations, allowing it to run almost **ten to one hundred times faster than Hadoop**.
2. **Flexibility** – Apache Spark supports multiple languages and allows the developers to write applications in Java, Scala, R, or Python.
3. **In-memory computing** – Spark stores the data in the RAM of servers which allows quick access and in turn accelerates the speed of analytics.
4. **Real-time processing** – Spark can process real-time streaming data. Unlike MapReduce which processes only stored data, Spark is able to process real-time data and is, therefore, able to produce instant outcomes.
5. **Better analytics** – In contrast to [MapReduce](https://www.ibm.com/analytics/hadoop/mapreduce) that includes Map and Reduce functions, Spark includes much more than that. Apache Spark consists of a rich set of SQL queries, machine learning algorithms, complex analytics, etc. With all these functionalities, analytics can be performed in a better fashion with the help of Spark.

* **What is Hive?**

Apache Hive is a distributed, fault-tolerant data warehouse system that enables analytics at a massive scale. A data warehouse provides a central store of information that can easily be analysed to make informed, data driven decisions. Hive allows users to read, write, and manage petabytes of data using SQL.

Hive is built on top of Apache Hadoop, which is an open-source framework used to efficiently store and process large datasets. As a result, Hive is closely integrated with Hadoop, and is designed to work quickly on petabytes of data. What makes Hive unique is the ability to query large datasets, leveraging Apache Tez or MapReduce, with a SQL-like interface.

* **What are NoSQL databases?**

NoSQL is an approach to database management that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats. A NoSQL database generally means that it is non-relational, distributed, flexible and scalable. Additional common NoSQL database features include the lack of a database schema, data clustering, replication support and eventual consistency, as opposed to the typical ACID (atomicity, consistency, isolation and durability) transaction consistency of relational and SQL databases. Many NoSQL database systems are also open source.

The term NoSQL originally could be taken at its word -- that is, SQL was not used as the API to access data. However, the ubiquity and usefulness of SQL caused many NoSQL databases to add support for SQL. Today it is commonly accepted that NoSQL stands for "Not Only SQL."

**MongoDB, CouchDB, CouchBase, Cassandra, HBase, Redis, Riak, Neo4J** are the popular NoSQL databases examples. MongoDB, CouchDB, CouchBase , Amazon SimpleDB, Riak, Lotus Notes are Document-oriented NoSQL databases.

* **What is HBase**

Hbase is an open source and sorted map data built on Hadoop. It is column oriented and horizontally scalable.

It is based on Google's Big Table. It has set of tables which keep data in key value format. Hbase is well suited for sparse data sets which are very common in big data use cases. Hbase provides APIs enabling development in practically any programming language. It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

* **What's difference between SQL and NoSQL?**

## **What is SQL?**

Structured Query language ([SQL](https://www.guru99.com/what-is-sql.html)) **pronounced as “S-Q-L” or sometimes as “See-Quel**” is the standard language for dealing with Relational Databases. A relational database defines relationships in the form of tables.

SQL programming can be effectively used to insert, search, update, delete database records.

That doesn’t mean SQL cannot do things beyond that. It can do a lot of things including, but not limited to, optimizing and maintenance of databases.

Relational databases like MySQL Database, Oracle, Ms SQL Server, Sybase, etc. use SQL.

## **What is NoSQL?**

[NoSQL](https://www.guru99.com/nosql-tutorial.html) is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale. NoSQL database is used for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example companies like Twitter, Facebook, Google that collect terabytes of user data every single day.

NoSQL database stands for “Not Only SQL” or “Not SQL.” Though a better term would NoREL NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

Next, we will discuss the key diff between SQL and NoSQL.

***The five critical differences between SQL vs NoSQL are:***

1. SQL databases are relational, NoSQL databases are non-relational.
2. SQL databases use structured query language and have a predefined schema. NoSQL databases have dynamic schemas for unstructured data.
3. SQL databases are vertically scalable, while NoSQL databases are horizontally scalable.
4. SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores.
5. SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like documents or JSON.

* **What is Data warehousing?**

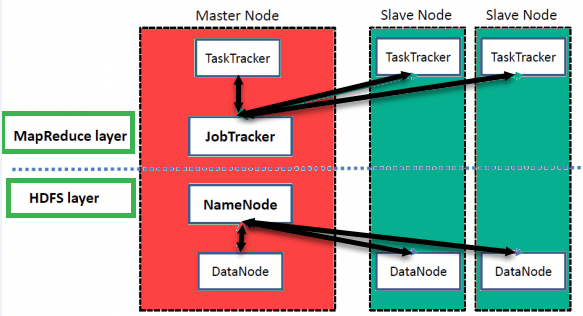
Data Warehouse is mainly an architecture not a technology. It extracting data from varieties SQL based data source (mainly relational data base) and help for generating analytic reports. In terms of definition, data repository, which using for any analytic reports, has been generated from one process, which is nothing but the data warehouse.

Diagram

Description automatically generated with medium confidence

* **What is Hadoop Architecture?**

The Hadoop architecture is a package of the file system, MapReduce engine and the HDFS (Hadoop Distributed File System). The MapReduce engine can be MapReduce/MR1 or YARN/MR2



High Level Hadoop Architecture

Hadoop has a Master-Slave Architecture for data storage and distributed data processing using MapReduce and HDFS methods.

**NameNode:**

NameNode represented every files and directory which is used in the namespace

**DataNode:**

DataNode helps you to manage the state of an HDFS node and allows you to interacts with the blocks

**MasterNode:**

The master node allows you to conduct parallel processing of data using Hadoop MapReduce.

**Slave node:**

The slave nodes are the additional machines in the Hadoop cluster which allows you to store data to conduct complex calculations. Moreover, all the slave node comes with Task Tracker and a DataNode. This allows you to synchronize the processes with the NameNode and Job Tracker respectively.

* **What is name node and data node?**

**NameNode:**

NameNode represented every files and directory which is used in the namespace

**DataNode:**

DataNode helps you to manage the state of an HDFS node and allows you to interacts with the blocks.

* **File system and Resource manager?**

## **What is HDFS?**

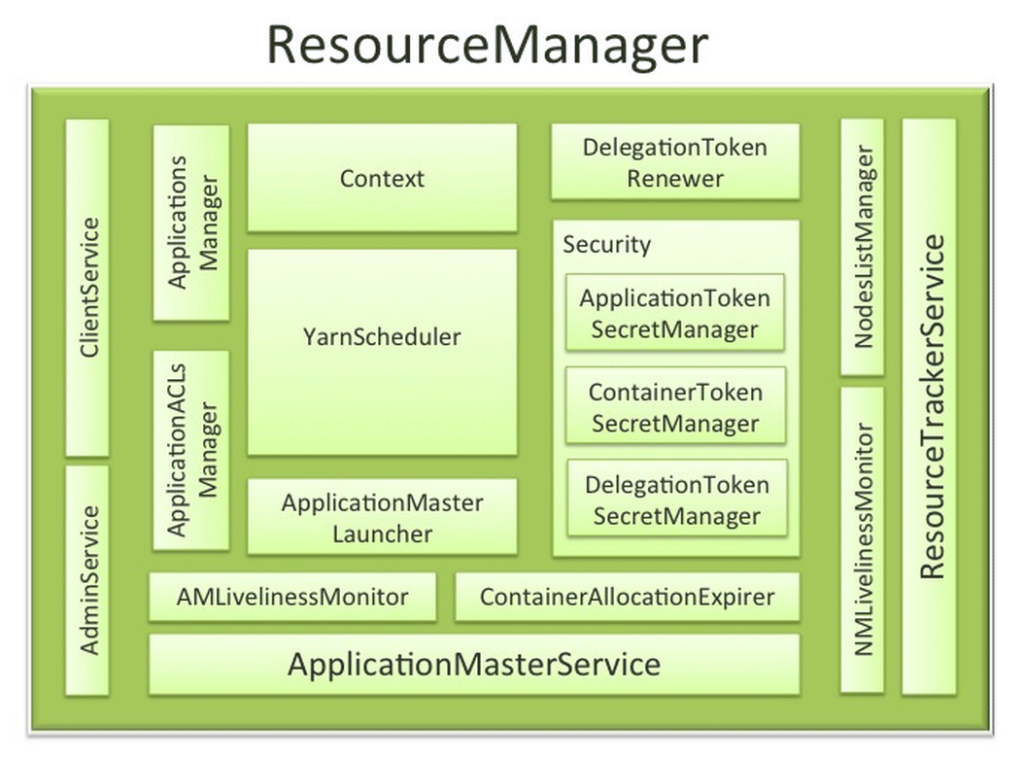
HDFS stands for Hadoop Distributed File System. HDFS operates as a distributed file system designed to run on commodity hardware.

HDFS is fault-tolerant and designed to be deployed on low-cost, commodity hardware. HDFS provides high throughput data access to application data and is suitable for applications that have large data sets and enables streaming access to file system data in [Apache Hadoop](https://databricks.com/glossary/hadoop).

So, [what is Hadoop?](https://databricks.com/glossary/hadoop) And how does it vary from HDFS? A core difference between Hadoop and HDFS is that Hadoop is the open source framework that can store, process and analyze data, while HDFS is the file system of Hadoop that provides access to data. This essentially means that HDFS is a module of Hadoop.

**ResourceManager (RM)** is the master that arbitrates all the available cluster resources and thus helps manage the distributed applications running on the YARN system. It works together with the per-node **NodeManagers (NMs)** and the per-application **ApplicationMasters (AMs)**.

1. **NodeManagers** take instructions from the ResourceManager and manage resources available on a single node.
2. **ApplicationMasters** are responsible for negotiating resources with the ResourceManager and for working with the NodeManagers to start the containers.

[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2016/06/Resource-Manager.png)

*Hadoop YARN Resource Manager*

The **Scheduler** API is specifically designed to negotiate resources and not schedule tasks. The scheduler does not perform monitoring or tracking of status for the Applications.

The Scheduler performs its scheduling function based the resource requirements of the applications; it does so base on the abstract notion of a resource Container which incorporates elements such as memory, CPU, disk, network etc.

Hadoop Yarn Resource Manager does not guarantee about restarting failed tasks either due to application failure or hardware failures. Applications can request resources at different layers of the cluster topology such as nodes, racks etc.

Hence, the scheduler determines how much and where to allocate based on resource availability and the configured sharing policy.

* **Cluster resource planning?**

Many customers ask what kind of machine to purchase to be used in a Hadoop environment, and what configuration to use. The answer to this can be essentially derived from some simple calculations that I want to write about and demonstrate.

Sizing a Hadoop cluster is important, as the right resources will allow you to optimize the environment for your purpose.  However, this is no simple task as optimizing a distributed environment and its related software can have its complexities.

The number of machines, and specs of the machines, depends on a few factors:

* The volume of data (obviously)
* The data retention policy (how much can you afford to keep before throwing away)
* The type of workload you have (data science/CPU driven vs “vanilla” use case/IO-bound)
* Also the data storage mechanism (data container, type of compression used if any)

We have to make some assumptions from the beginning; otherwise there are just too many parameters to deal with. These assumptions drive the data nodes configuration. The other types of machines (Name Node/Job tracker, in Hadoop ) will need different specs, and are generally more straightforward.

* **Important processes and daemons?**
* NameNode
* DataNode
* Secondary Name Node
* Resource Manager
* Node Manager

Namenode, Secondary NameNode, and Resource Manager work on a Master System while the Node Manager and DataNode work on the Slave machine.

For Name Node start : hadoop-daemon.sh start namenode

For Name Node stop: hadoop-daemon.sh stop namenode

* **File formats**
* Text files
* Sequence File
* Avro data files
* Parquet file format
* **Block size and it’s significant**

**By default, HDFS block size is 128MB which you can change as per your requirement**. All HDFS blocks are the same size except the last block, which can be either the same size or smaller. Hadoop framework break files into 128 MB blocks and then stores into the Hadoop file system.

* **Replication factor**

By default, the Replication Factor for Hadoop is set to 3 which can be configured means you can change it Manually as per your requirement like in above example we have made 4 file blocks which means that 3 Replica or copy of each file block is made means total of 4×3 = 12 blocks are made for the backup purpose.

* **High availability and fault tolerance**

**Hadoop provides high reliability and scalability features.** **Along with it also afford faults tolerance mechanism by which the system functions properly even after a node in the cluster fails**. Faults tolerance is mainly achieved using data replication and Heartbeat messages.

* **What is ETL, what does its use in big data**

ETL (Extract, Transform, Load) is **the process of extracting data from disparate sources, transforming it into a clean and analysis-ready format, and loading it into a data warehouse for analysis**.

* **Important ETL concepts**

**Source System :**

The source system is any system from which we are extracting the data.

Example : It should be any database, Application, File with data.

**Target System :**

Target System is any system where we are loading the data.

Example : It should be any database, Application, File with data.

**Data Mapping :**

The Data mapping is nothing but the relationships between database objects.

**Staging Area :**

The Staging area is nothing but the database area where all processing of the data will be done.

**Metadata :**

Metadata is data within a data.

These are some important terms to learn ETL Concepts.

* **Important ETL tools**
* [Hevo Data](https://hevodata.com/learn/best-big-data-etl-tools/#b1)
* [Talend (Talend Open Studio For Data Integration)](https://hevodata.com/learn/best-big-data-etl-tools/#b2)
* [Informatica – PowerCenter](https://hevodata.com/learn/best-big-data-etl-tools/#b3)
* [IBM Infosphere Information Server](https://hevodata.com/learn/best-big-data-etl-tools/#b4)
* [Pentaho Data Integration](https://hevodata.com/learn/best-big-data-etl-tools/#b5)
* [CloverDX](https://hevodata.com/learn/best-big-data-etl-tools/#b6)
* [Oracle Data Integrator](https://hevodata.com/learn/best-big-data-etl-tools/#b7)
* [StreamSets](https://hevodata.com/learn/best-big-data-etl-tools/#b8)
* [Matillion](https://hevodata.com/learn/best-big-data-etl-tools/#b9)
* **Why spark required in big data world**
* Iterative Algorithm: Generally MapReduce is not good to process iterative algorithms like Machine Learning and Graph processing. Graph and Machine Learning algorithms are iterative by nature and less saves to disk, this type of algorithm needs data în memory to run algorithm steps again and again or less transfers over network means better performance.
* In Memory Processing: MapReduce uses disk storage for storing processed intermediate data and also read from disks which is not good for fast processing.. Because Spark keeps data in Memory (Configurable), which saves lot of time, by not reading and writing data to disk as it happens in case of Hadoop.
* Near real-time data processing: Spark also supports near real-time streaming workloads via Spark Streaming application framework.
* **Hadoop vs Spark**

|  |
| --- |
| * Hadoop is an open-source framework which uses a MapReduce algorithm |
| * Hadoop’s MapReduce model reads and writes from a disk, thus slow down the processing speed |
| * Hadoop is designed to handle batch processing efficiently |
| * Hadoop is a high latency computing framework, which does not have an interactive mode |
| * With Hadoop MapReduce, a developer can only process data in batch mode only |
| ***Spark*** |
| * Spark is lightning-fast cluster computing technology, which extends the MapReduce model to efficiently use with more type of computations. |
| * Spark reduces the number of read/write cycles to disk and store intermediate data in-memory, hence faster-processing speed. |
| * Spark is designed to handle real-time data efficiently. |
| * Spark is a low latency computing and can process data interactively. |
| * Spark can process real-time data, from real time events like twitter, Facebook |
| * Spark requires a lot of RAM to run in-memory, thus increasing the cluster and hence cost. |

* **Spark data structure and their importance**

**RDD, DataFrame, and Dataset** are the three most common data structures in Spark, and they make processing very large data easy and convenient. Because of the lazy evaluation algorithm of Spark, these data structures are not executed right way during creations, transformations, and functions etc.

* **Important spark libraries**

Spark includes libraries for **SQL and structured data (Spark SQL), machine learning (MLlib), stream processing (Spark Streaming and the newer Structured Streaming), and graph analytics (GraphX)**.

* **Spark components-Drive, worker, executor etc**

As we can see that Spark follows Master-Slave architecture where we have one central coordinator and multiple distributed worker nodes. The central coordinator is called Spark Driver and it communicates with all the Workers.

Each Worker node consists of one or more Executor(s) who are responsible for running the Task. Executors register themselves with Driver. The Driver has all the information about the Executors at all the time.

This working combination of Driver and Workers is known as Spark Application.

* **Important spark functions**

#### Custom Transformation

#### SPARK SQL FUNCTIONS

#### COLUMNS FUNCTIONS

#### USER DEFINED FUNCTIONS

* **Spark program execution lifecycle**
* The user submits a spark application using the spark -submit command.
* Spark-submit launches the driver program on the same node in (client mode) or on the cluster (cluster mode) and invokes the main method specified by the user.
* The driver program contacts the cluster manager to ask for resources to launch executor JVMs based on the configuration parameters supplied.
* The cluster manager launches executor JVMs on worker nodes.
* The driver process scans through the user application. Based on the RDD actions and transformations in the program, Spark creates an operator.
* **Why is hive required in big data world?**

Hive **big data through the lens of data analytics can help us get more insights into the working of Apache Hive**. By using a batch processing sequence, Hive generates data analytics in a much easier and organized form that also requires less time as compared to traditional tools.

* **SQL vs HQL?**

**SQL is based on a relational database model whereas HQL is a combination of object-oriented programming with relational database concepts**. SQL manipulates data stored in tables and modifies its rows and columns. HQL is concerned about objects and its properties.

* **Basic hive commands**
* create database
* drop database
* create table
* drop table
* alter table
* create index
* create views
* Select
* Where
* Group By
* Order By
* Load Data

***Join:***

* Inner Join
* Left Outer Join
* Right Outer Join
* Full Outer Join
* **Performance tuning concepts/Partition/Bucketing**

Bucketing decomposes data into more manageable or equal parts. With partitioning, there is a possibility that you can create multiple small partitions based on column values. If you go for bucketing, you are restricting number of buckets to store the data. This number is defined during table creation scripts.

* **Input/Output file formats**
* FileInputFormat. It is the base class for all file-based InputFormats.
* TextInputFormat. It is the default InputFormat.
* KeyValueTextInputFormat.
* SequenceFileInputFormat.
* SequenceFileAsTextInputFormat.
* SequenceFileAsBinaryInputFormat.
* NlineInputFormat.
* DBInputFormat.
* **What does it mean to debug a script?**

**Debugging, in computer programming and engineering, is a multistep process that involves identifying a problem, isolating the source of the problem, and then either correcting the problem or determining a way to work around it. The final step of debugging is to test the correction or workaround and make sure it works.**