

# H1B VISA PROJECT

By,  
A.Antony Ravinsta,  
S181165200323

## INTRODUCTION

In this document we can study the overview of new technology called Bigdata with Hadoop and its components.

Hadoop **MapReduce** is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

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

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy. Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Apache **Pig** is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with Hadoop; we can perform all the data manipulation operations in Hadoop using Apache Pig. To write data analysis programs, Pig provides a high-level language known as **Pig Latin**. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data. To analyze data using Apache Pig, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as Pig Engine that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

**Sqoop** is a tool designed to transfer data between Hadoop and relational database servers. It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases.

**Yarn** - Yet Another Resource Manager takes programming to the next level beyond Java, and makes it interactive to let another application Hbase, Spark etc. to work on it. Different Yarn applications can co-exist on the same cluster so MapReduce, Hbase, Spark all can run at the same time bringing great benefits for manageability and cluster utilization.

Apache **Flume** is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log files, events (etc...) from various sources to a centralized data store.

**HBase** is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS). It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System. One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.

**ZooKeeper** is a distributed co-ordination service to manage large set of hosts. Co-ordinating and managing a service in a distributed environment is a complicated process. ZooKeeper solves this

**Oozie** is an **Open Source Java Web-Application** available under Apache license 2.0. It is responsible for triggering the workflow actions, which in turn uses the Hadoop execution engine to actually execute the task. Hence, Oozie is able to leverage the existing Hadoop machinery for load balancing, fail-over, etc. **Apache Spark** is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application. **Apache Cassandra** is a highly scalable, high-performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. It is a type of NoSQL database. Let us first understand what a NoSQL database does.

Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks.



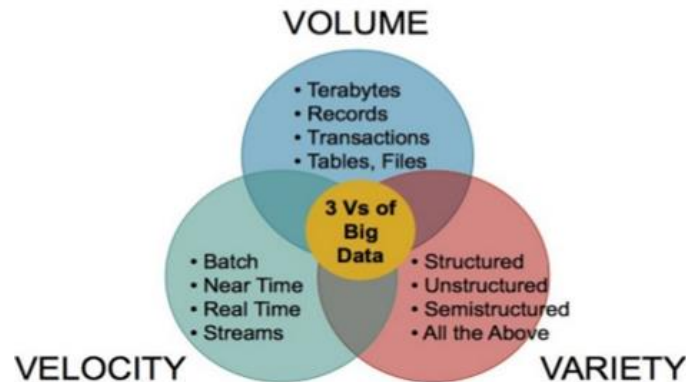
Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

- **Black Box Data** : It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
- **Social Media Data** : Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.
- **Stock Exchange Data** : The stock exchange data holds information about the ‘buy’ and ‘sell’ decisions made on a share of different companies made by the customers.

- **Power Grid Data** : The power grid data holds information consumed by a particular node with respect to a base station.
- **Transport Data** : Transport data includes model, capacity, distance and availability of a vehicle.
- **Search Engine Data** : Search engines retrieve lots of data from different databases.

### **BIGDATA CHARACTERISTICS**

Big Data characteristics that includes three main **V's**. They are **Volume**, **Velocity**, and **Variety** of data and some make it as four **V's** (i.e.), it includes **Verocity**. The data in it will be of three types.



- **Structured data** : Relational data.
- **Semi Structured data** : XML data.
- **Unstructured data** : Word, PDF, Text, Media Logs.

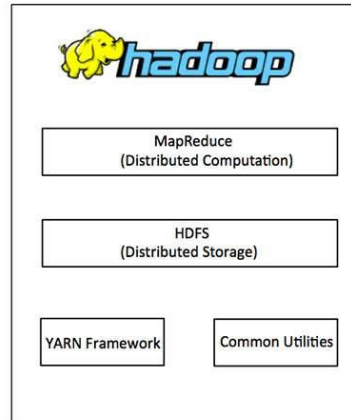
### **APACHE HADOOP**

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frameworked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

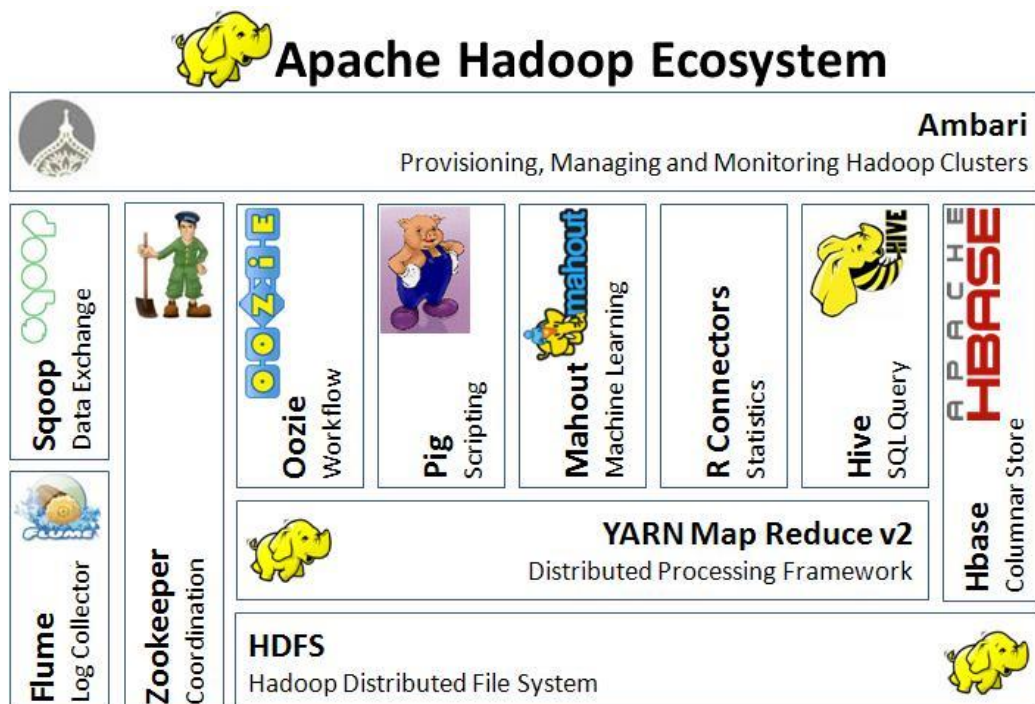
### **HADOOP ARCHITECTURE**

Hadoop framework includes following four modules:

- **Hadoop Common**: These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- **Hadoop YARN**: This is a framework for job scheduling and cluster resource management.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop MapReduce**: This is YARN-based system for parallel processing of large data sets.



## HADOOP ECOSYSTEM



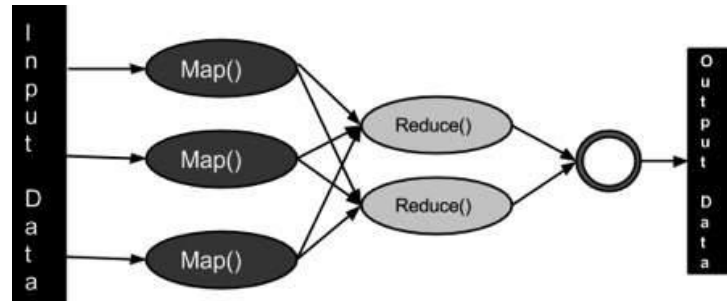
## MAP-REDUCE

Hadoop **MapReduce** is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

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

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:

- **The Map Task:** This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).
- **The Reduce Task:** This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.



## JAVA PERSPECTIVE

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

## Sort and Shuffle

The sort and shuffle occur on the output of mapper and before the reducer. When the mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. Using the input from each mapper <k2,v2>, we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2,list(v2)> is sent as input to reducer phase.

## TERMINOLOGY

- **PayLoad** - Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** - Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** - Node that manages the Hadoop Distributed File System (HDFS).
- **DataNode** - Node where data is presented in advance before any processing takes place.
- **MasterNode** - Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** - Node where Map and Reduce program runs.
- **JobTracker** - Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** - Tracks the task and reports status to JobTracker.
- **Job** - A program is an execution of a Mapper and Reducer across a dataset.
- **Task** - An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** - A particular instance of an attempt to execute a task on a SlaveNode.

## HADOOP DISTRIBUTED FILE SYSTEM – HDFS

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly faulttolerant 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.

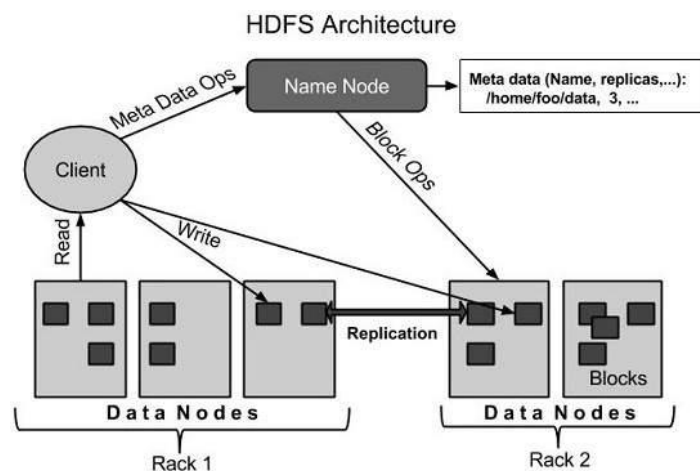
### FEATURES OF HDFS

- 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.

### GOALS OF HDFS

- Fault detection and recovery
- Huge datasets
- Hardware at data

### HDFS ARCHITECTURE



## **Namenode**

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks:

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

## **Datanode**

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- 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.

## **Block**

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

## **Goals of HDFS**

- **Fault detection and recovery** : Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.
- **Huge datasets** : HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.
- **Hardware at data** : A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

## **HIVE**

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

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



## Hive is not

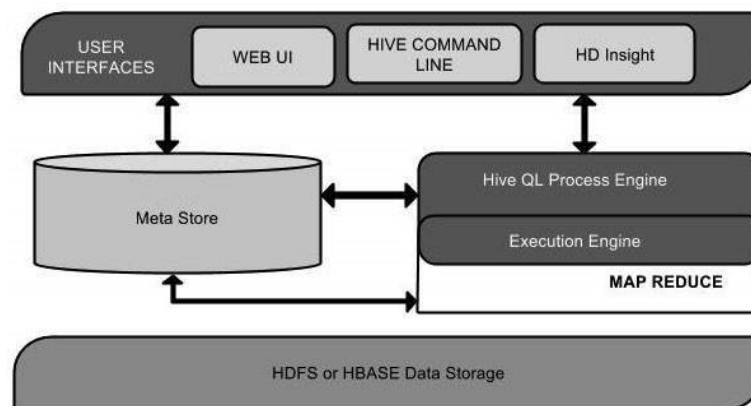
- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

## Features of Hive

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

## ARCHITECTURE OF HIVE

The following component diagram depicts the architecture of Hive:

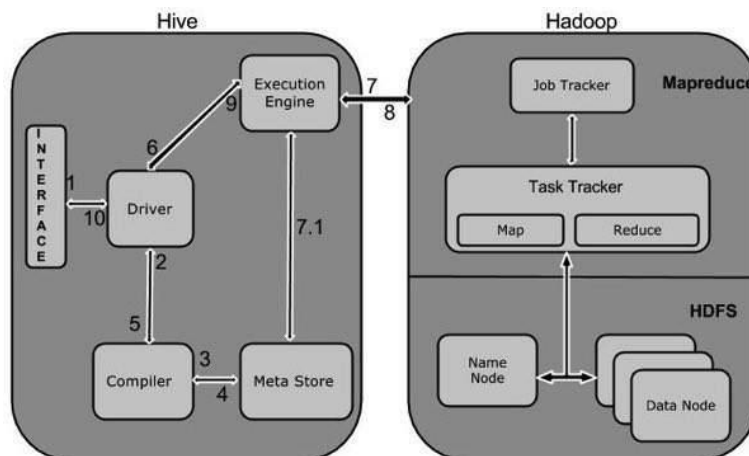


Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.

HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

### Working of Hive

The following diagram depicts the workflow between Hive and Hadoop.



The following table defines how Hive interacts with Hadoop framework:

Step No.	Operation
1	<b>Execute Query</b> <p>The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.</p>
2	<b>Get Plan</b> <p>The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.</p>
3	<b>Get Metadata</b>

	The compiler sends metadata request to Metastore (any database).
4	<b>Send Metadata</b> Metastore sends metadata as a response to the compiler.
5	<b>Send Plan</b> The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6	<b>Execute Plan</b> The driver sends the execute plan to the execution engine.
7	<b>Execute Job</b> Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
7.1	<b>Metadata Ops</b> Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8	<b>Fetch Result</b> The execution engine receives the results from Data nodes.
9	<b>Send Results</b> The execution engine sends those resultant values to the driver.
10	<b>Send Results</b> The driver sends the results to Hive Interfaces.

## **PIG**

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with **Hadoop**; we can perform all the data manipulation operations in Hadoop using Apache Pig.

To write data analysis programs, Pig provides a high-level language known as **Pig Latin**. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.

To analyze data using **Apache Pig**, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as **Pig Engine** that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

Programmers who are not so good at Java normally used to struggle working with Hadoop, especially while performing any MapReduce tasks. Apache Pig is a boon for all such programmers.

- Using **Pig Latin**, programmers can perform MapReduce tasks easily without having to type complex codes in Java.
- Apache Pig uses **multi-query approach**, thereby reducing the length of codes. For example, an operation that would require you to type 200 lines of code (LoC) in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately Apache Pig reduces the development time by almost 16 times.
- Pig Latin is **SQL-like language** and it is easy to learn Apache Pig when you are familiar with SQL.
- Apache Pig provides many built-in operators to support data operations like joins, filters, ordering, etc. In addition, it also provides nested data types like tuples, bags, and maps that are missing from MapReduce.

### Features of Pig

Apache Pig comes with the following features –

- **Rich set of operators** – It provides many operators to perform operations like join, sort, filter, etc.
- **Ease of programming** – Pig Latin is similar to SQL and it is easy to write a Pig script if you are good at SQL.
- **Optimization opportunities** – The tasks in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.
- **Extensibility** – Using the existing operators, users can develop their own functions to read, process, and write data.
- **UDF's** – Pig provides the facility to create **User-defined Functions** in other programming languages such as Java and invoke or embed them in Pig Scripts.
- **Handles all kinds of data** – Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

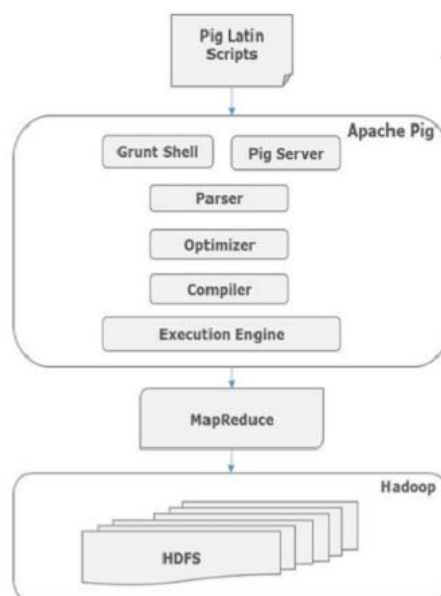
### Apache Pig Vs SQL

- Listed below are the major differences between Apache Pig and SQL.

Pig	SQL
-----	-----

Pig Latin is a <b>procedural</b> language.	SQL is a <b>declarative</b> language.
In Apache Pig, <b>schema</b> is optional. We can store data without designing a schema (values are stored as \$01, \$02 etc.)	Schema is mandatory in SQL.
The data model in Apache Pig is <b>nested relational</b> .	The data model used in SQL is <b>flat relational</b> .
Apache Pig provides limited opportunity for <b>Query optimization</b> .	There is more opportunity for query optimization in SQL.

### ARCHITECTURE OF PIG



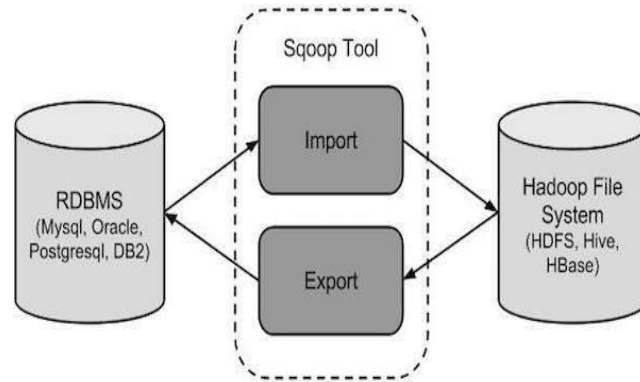
### SQOOP

Sqoop is a tool designed to transfer data between Hadoop and relational database servers. It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases.

**Sqoop:** “SQL to Hadoop and Hadoop to SQL”

### ARCHITECTURE

The following image describes the workflow of Sqoop.



### Sqoop Import

The import tool imports individual tables from RDBMS to HDFS. Each row in a table is treated as a record in HDFS. All records are stored as text data in text files or as binary data in Avro and Sequence files.

### Sqoop Export

The export tool exports a set of files from HDFS back to an RDBMS. The files given as input to Sqoop contain records, which are called as rows in table. Those are read and parsed into a set of records and delimited with user-specified delimiter.

## YARN

Yet Another Resource Manager takes programming to the next level beyond Java , and makes it interactive to let another application Hbase, Spark etc. to work on it. Different Yarn applications can co-exist on the same cluster so MapReduce, Hbase, Spark all can run at the same time bringing great benefits for manageability and cluster utilization.

### Components Of YARN

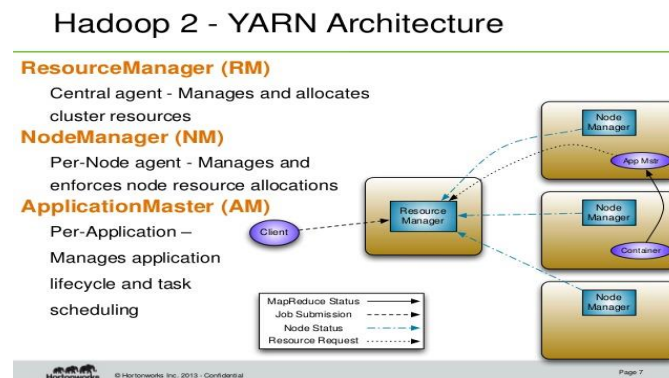
- **Client:** For submitting MapReduce jobs.
- **Resource Manager:** To manage the use of resources across the cluster
- **Node Manager:** For launching and monitoring the computer containers on machines in the cluster.
- **Map Reduce Application Master:** Checks tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager, and managed by the node managers.

Jobtracker & Tasktracker were used in previous version of Hadoop, which were responsible for handling resources and checking progress management. However, Hadoop 2.0 has Resource manager and NodeManager to overcome the shortfall of Jobtracker & Tasktracker.

### Benefits of YARN

- **Scalability:** Map Reduce 1 hits scalability bottleneck at 4000 nodes and 40000 task, but Yarn is designed for 10,000 nodes and 1 lakh tasks.
- **Utilization:** Node Manager manages a pool of resources, rather than a fixed number of the designated slots thus increasing the utilization.
- **Multitenancy:** Different version of MapReduce can run on YARN, which makes the process of upgrading MapReduce more manageable.

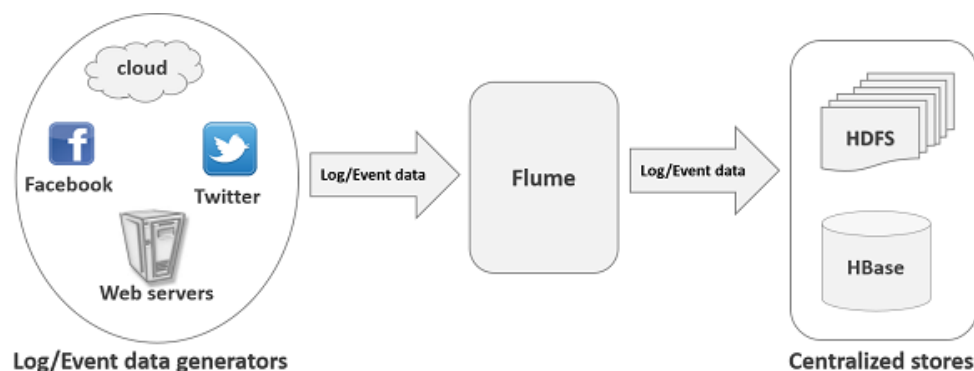
## ARCHITECTURE



## FLUME

Apache Flume is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log files, events (etc...) from various sources to a centralized data store. Flume is a highly reliable, distributed, and configurable tool. It is principally designed to copy streaming data (log data) from various web servers to HDFS.

## ARCHITECTURE



## **Applications of Flume**

Assume an e-commerce web application wants to analyze the customer behavior from a particular region. To do so, they would need to move the available log data in to Hadoop for analysis. Here, Apache Flume comes to our rescue. Flume is used to move the log data generated by application servers into HDFS at a higher speed.

## Advantages of Flume

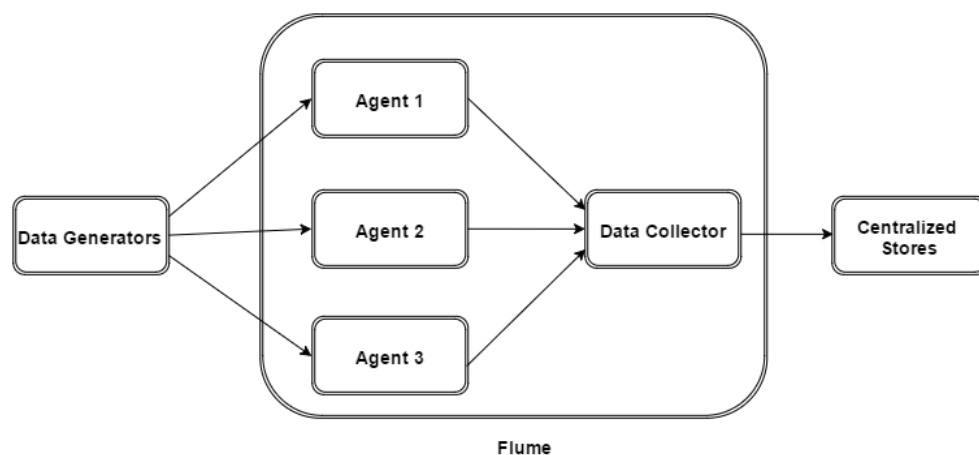
Here are the advantages of using Flume

- Using Apache Flume we can store the data in to any of the centralized stores (HBase, HDFS).
- When the rate of incoming data exceeds the rate at which data can be written to the destination, Flume acts as a mediator between data producers and the centralized stores and provides a steady flow of data between them.
- Flume provides the feature of **contextual routing**.
- The transactions in Flume are channel-based where two transactions (one sender and one receiver) are maintained for each message. It guarantees reliable message delivery.
- Flume is reliable, fault tolerant, scalable, manageable, and customizable.

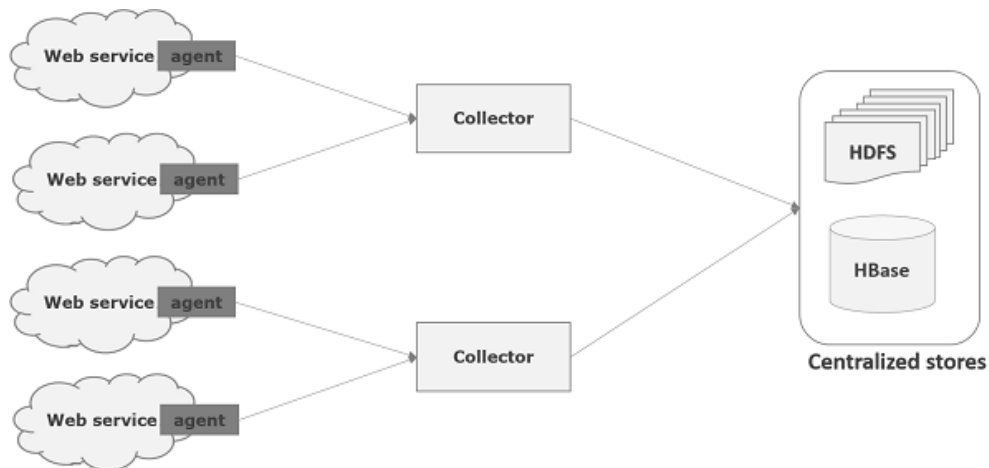
## Features of Flume

Some of the notable features of Flume are as follows

- Flume ingests log data from multiple web servers into a centralized store (HDFS, HBase) efficiently.
- Using Flume, we can get the data from multiple servers immediately into Hadoop.
- Along with the log files, Flume is also used to import huge volumes of event data produced by social networking sites like Facebook and Twitter, and e-commerce websites like Amazon and Flipkart.
- Flume supports a large set of sources and destinations types.
- Flume supports multi-hop flows, fan-in fan-out flows, contextual routing, etc.
- Flume can be scaled horizontally.

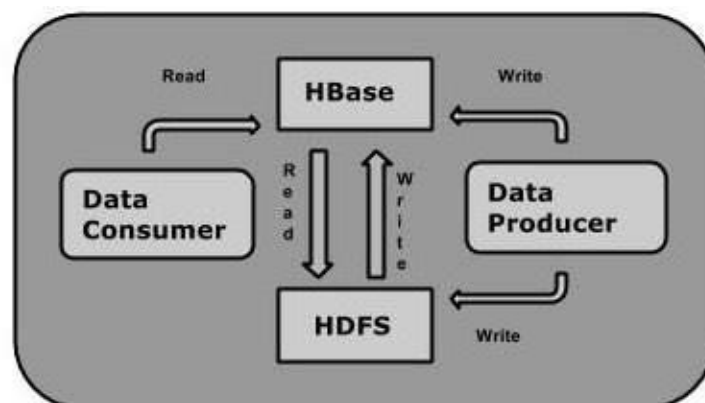






## HBASE

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS). It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System. One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.



### HBase and HDFS

HDFS	HBase
HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.

It provides high latency batch processing; no concept of batch processing.	It provides low latency access to single rows from billions of records (Random access).
It provides only sequential access of data.	HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups.

### **Features of HBase**

- HBase is linearly scalable.
- It has automatic failure support.
- It provides consistent read and writes.
- It integrates with Hadoop, both as a source and a destination.
- It has easy java API for client.
- It provides data replication across clusters.

### **Applications of HBase**

- It is used whenever there is a need to write heavy applications.
- HBase is used whenever we need to provide fast random access to available data.
- Companies such as Facebook, Twitter, Yahoo, and Adobe use HBase internally.

### **ZOOKEEPER**

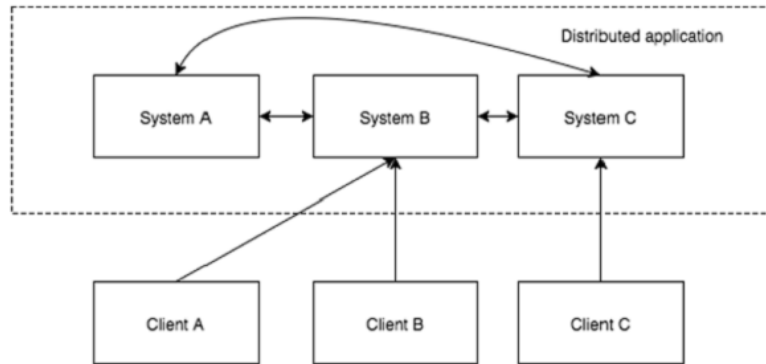
ZooKeeper is a distributed co-ordination service to manage large set of hosts. Co-ordinating and managing a service in a distributed environment is a complicated process. ZooKeeper solves this issue with its simple architecture and API. ZooKeeper allows developers to focus on core application logic without worrying about the distributed nature of the application. The ZooKeeper framework was originally built at “Yahoo!” for accessing their applications in an easy and robust manner. Later, Apache ZooKeeper became a standard for organized service used by Hadoop, HBase, and other distributed frameworks. For example, Apache HBase uses ZooKeeper to track the status of distributed data.

### **Distributed Application**

A distributed application can run on multiple systems in a network at a given time (simultaneously) by coordinating among themselves to complete a particular task in a fast and efficient manner. Normally, complex and time-consuming tasks, which will take hours to complete by a non-distributed application (running in a single system) can be done in minutes by a distributed application by using computing capabilities of all the system involved.

The time to complete the task can be further reduced by configuring the distributed application to run on more systems. A group of systems in which a distributed application is running is called a **Cluster** and each machine running in a cluster is called a **Node**.

A distributed application has two parts, **Server** and **Client** application. Server applications are actually distributed and have a common interface so that clients can connect to any server in the cluster and get the same result. Client applications are the tools to interact with a distributed application.



### Benefits of Distributed Applications

- **Reliability** – Failure of a single or a few systems does not make the whole system to fail.
- **Scalability** – Performance can be increased as and when needed by adding more machines with minor change in the configuration of the application with no downtime.
- **Transparency** – Hides the complexity of the system and shows itself as a single entity / application.

### Challenges of Distributed Applications

- **Race condition** – Two or more machines trying to perform a particular task, which actually needs to be done only by a single machine at any given time. For example, shared resources should only be modified by a single machine at any given time.
- **Deadlock** – Two or more operations waiting for each other to complete indefinitely.
- **Inconsistency** – Partial failure of data.

### Apache ZooKeeper

Apache ZooKeeper is a service used by a cluster (group of nodes) to coordinate between themselves and maintain shared data with robust synchronization techniques. ZooKeeper is itself a distributed application providing services for writing a distributed application.

The common services provided by ZooKeeper are as follows –

- **Naming service** – Identifying the nodes in a cluster by name. It is similar to DNS, but for nodes.
- **Configuration management** – Latest and up-to-date configuration information of the system for a joining node.

- **Cluster management** – Joining / leaving of a node in a cluster and node status at real time.
- **Leader election** – Electing a node as leader for coordination purpose.
- **Locking and synchronization service** – Locking the data while modifying it. This mechanism helps you in automatic fail recovery while connecting other distributed applications like Apache HBase.
- **Highly reliable data registry** – Availability of data even when one or a few nodes are down.

Distributed applications offer a lot of benefits, but they throw a few complex and hard-to-crack challenges as well. ZooKeeper framework provides a complete mechanism to overcome all the challenges. Race condition and deadlock are handled using **fail-safe synchronization approach**. Another main drawback is inconsistency of data, which ZooKeeper resolves with **atomicity**.

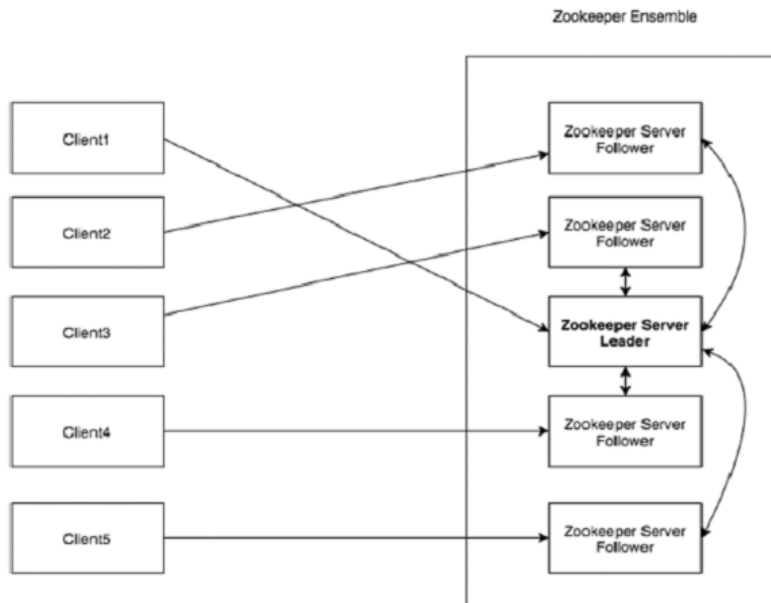
### Benefits of ZooKeeper

Here are the benefits of using ZooKeeper –

- **Simple distributed coordination process**
- **Synchronization** – Mutual exclusion and co-operation between server processes. This process helps in Apache HBase for configuration management.
- **Ordered Messages**
- **Serialization** – Encode the data according to specific rules. Ensure your application runs consistently. This approach can be used in MapReduce to coordinate queue to execute running threads.
- **Reliability**
- **Atomicity** – Data transfer either succeed or fail completely, but no transaction is partial.

### Architecture of ZooKeeper

Take a look at the following diagram. It depicts the “Client-Server Architecture” of ZooKeeper.



## OOZIE

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. Within a sequence of task, two or more jobs can also be programmed to run parallel to each other.

One of the main advantages of Oozie is that it is tightly integrated with Hadoop stack supporting various Hadoop jobs like **Hive, Pig, Sqoop** as well as system-specific jobs like **Java and Shell**.

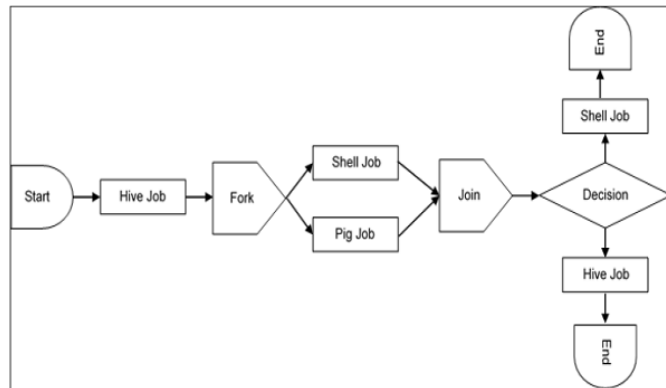
Oozie is an **Open Source Java Web-Application** available under Apache license 2.0. It is responsible for triggering the workflow actions, which in turn uses the Hadoop execution engine to actually execute the task. Hence, Oozie is able to leverage the existing Hadoop machinery for load balancing, fail-over, etc.

Oozie detects 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 it is complete. If the task fails to invoke the callback URL, Oozie can poll the task for completion.

**Following three types of jobs are common in Oozie –**

- **Oozie Workflow Jobs** – These are represented as Directed Acyclic Graphs (DAGs) to specify a sequence of actions to be executed.
- **Oozie Coordinator Jobs** – These consist of workflow jobs triggered by time and data availability.
- **Oozie Bundle** – These can be referred to as a package of multiple coordinator and workflow jobs.

A sample workflow with Controls (Start, Decision, Fork, Join and End) and Actions (Hive, Shell, Pig) will look like the following diagram



### Use-Cases of Apache Oozie

Apache Oozie is used by Hadoop system administrators to run complex log analysis on **HDFS**. Hadoop Developers use Oozie for performing ETL operations on data in a sequential order and saving the output in a specified format (Avro, ORC, etc.) in HDFS. In an enterprise, Oozie jobs are scheduled as coordinators or bundles.

### Oozie Editors

Before we dive into Oozie let's have a quick look at the available editors for Oozie. Most of the time, you won't need an editor and will write the workflows using any popular text editors (like Notepad++, Sublime or Atom) as we will be doing in this tutorial. But as a beginner it makes some sense to create a workflow by the drag and drop method using the editor and then see how the workflow gets generated. Also, to map **GUI** with the actual **workflow.xml** created by the editor. This is the only section where we will discuss about Oozie editors and won't use it in our tutorial.

### SPARK

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application.

Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

### Evolution of Apache Spark

Spark is one of Hadoop's sub project developed in 2009 in UC Berkeley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

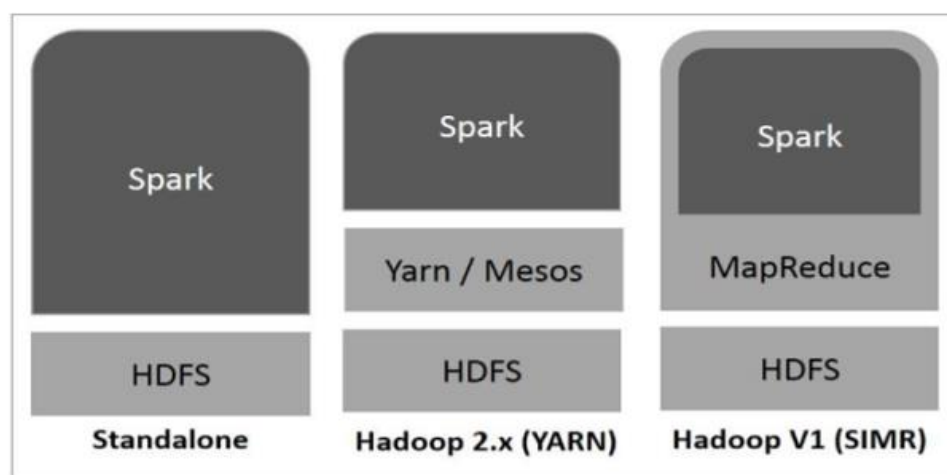
### Features of Apache Spark

Apache Spark has following features.

- **Speed** – Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
- **Supports multiple languages** – Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.
- **Advanced Analytics** – Spark not only supports ‘Map’ and ‘reduce’. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

### Spark Built on Hadoop

The following diagram shows three ways of how Spark can be built with Hadoop components.



There are three ways of Spark deployment as explained below.

- **Standalone** – Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.
- **Hadoop Yarn** – Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.
- **Spark in MapReduce (SIMR)** – Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

### CASSANDRA

Apache Cassandra is a highly scalable, high-performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. It is a type of NoSQL database. Let us first understand what a NoSQL database does.

### NoSQL Database

A NoSQL database (sometimes called as Not Only SQL) is a database that provides a mechanism to store and retrieve data other than the tabular relations used in relational databases. These databases are schema-free, support easy replication, have simple API, eventually consistent, and can handle huge amounts of data.

The primary objective of a NoSQL database is to have

- simplicity of design,
- horizontal scaling, and
- finer control over availability.

NoSql databases use different data structures compared to relational databases. It makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it must solve.

### Apache Cassandra

Apache Cassandra is an open source, distributed and decentralized/distributed storage system (database), for managing very large amounts of structured data spread out across the world. It provides highly available service with no single point of failure.

Listed below are some of the notable points of Apache Cassandra:

- It is scalable, fault-tolerant, and consistent.
- It is a column-oriented database.
- Its distribution design is based on Amazon's Dynamo and its data model on Google's Bigtable.
- Created at Facebook, it differs sharply from relational database management systems.
- Cassandra implements a Dynamo-style replication model with no single point of failure, but adds a more powerful "column family" data model.
- Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

### Features of Cassandra

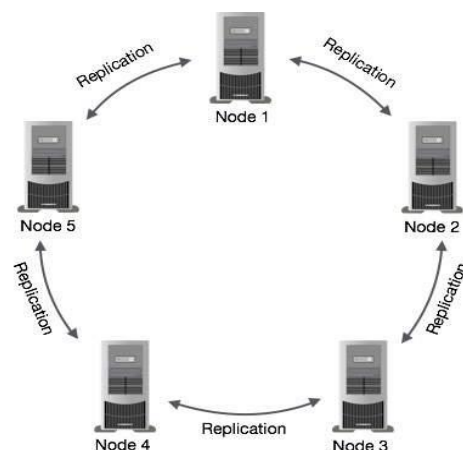
Cassandra has become so popular because of its outstanding technical features. Given below are some of the features of Cassandra:

- **Elastic scalability** - Cassandra is highly scalable; it allows to add more hardware to accommodate more customers and more data as per requirement.
- **Always on architecture** - Cassandra has no single point of failure and it is continuously available for business-critical applications that cannot afford a failure.



- **Fast linear-scale performance** - Cassandra is linearly scalable, i.e., it increases your throughput as you increase the number of nodes in the cluster. Therefore it maintains a quick response time.
- **Flexible data storage** - Cassandra accommodates all possible data formats including: structured, semi-structured, and unstructured. It can dynamically accommodate changes to your data structures according to your need.
- **Easy data distribution** - Cassandra provides the flexibility to distribute data where you need by replicating data across multiple data centers.
- **Transaction support** - Cassandra supports properties like Atomicity, Consistency, Isolation, and Durability (ACID).
- **Fast writes** - Cassandra was designed to run on cheap commodity hardware. It performs blazingly fast writes and can store hundreds of terabytes of data, without sacrificing the read efficiency.

## ARCHITECTURE



## PROJECT – BIGDATA WITH HADOOP

### H1B – VISA APPLICATION

#### COMPONENTS USED

- Hive
- Pig
- MapReduce
- Sqoop

#### CONCLUSION:

Thus the H1B-visa data has been analysed with queries and results have been obtained and saved.

