HBASE – OVERVIEW

Since 1970, RDBMS is the solution for data storage and maintenance related problems. After the advent of big data, companies realized the benefit of processing big data and started opting for solutions like Hadoop.
Hadoop uses distributed file system for storing big data, and MapReduce to process it. Hadoop excels in storing and processing of huge data of various formats such as arbitrary, semi-, or even unstructured.

Limitations of Hadoop

1. **Only Batch Processing**: Hadoop is slow because it processes data in large batches, not in real-time.
2. **Sequential Data Access**: Even for simple tasks, it must scan the entire dataset (like reading a book from start to finish just to find one word).
3. **Creates More Data**: Processing big data often creates even more data, which takes more time to process.
4. **No Random Access**: Unlike databases (e.g., SQL), you can't quickly fetch specific data you always have to scan everything.

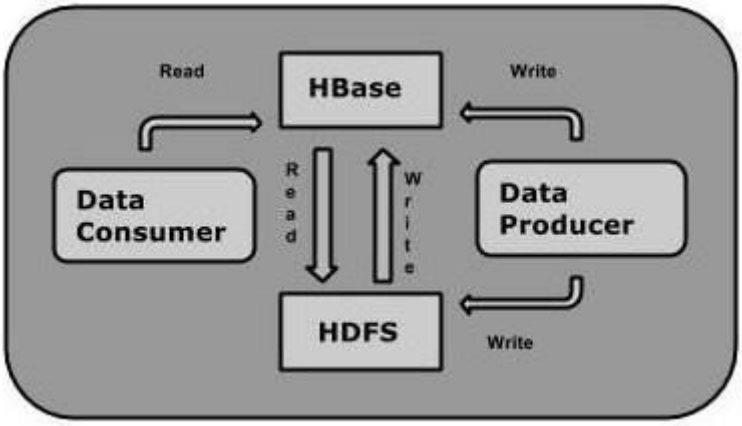
Hadoop Random Access Databases

Applications such as HBase, Cassandra, CouchDB, Dynamo, and MongoDB are some of the databases that store huge amounts of data and access the data in a random manner.

What is 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;	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.

Storage Mechanism in HBase

HBase is a **column-oriented database** and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp. In short, in an HBase:

- Table is a collection of rows.
- Row is a collection of column families.
- Column family is a collection of columns.
- Column is a collection of key value pairs.

Given below is an example schema of table in HBase.

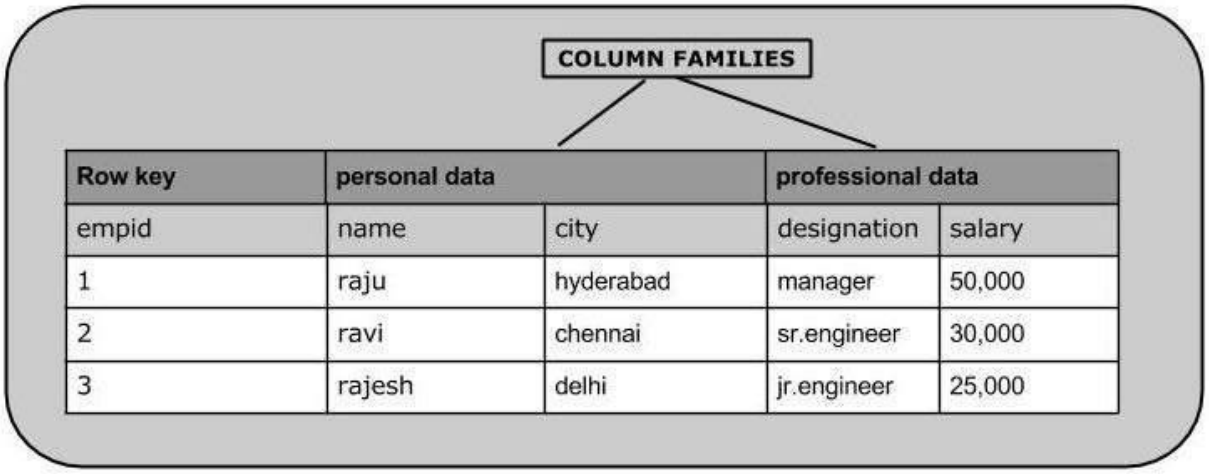
Rowid	Column Family			Column Family			Column Family			Column Family		
	col1	col2	col3	col1	col2	col3	col1	col2	col3	col1	col2	col3
1												
2												
3												

Column Oriented and Row Oriented

Column-oriented databases are those that store data tables as sections of columns of data, rather than as rows of data. Shortly, they will have column families.

Row-Oriented Database	Column-Oriented Database
It is suitable for Online Transaction Process (OLTP).	It is suitable for Online Analytical Processing (OLAP).
Such databases are designed for small number of rows and columns.	Column-oriented databases are designed for huge tables.

The following image shows column families in a column-oriented database:



HBase and RDBMS

HBase	RDBMS
HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families.	An RDBMS is governed by its schema, which describes the whole structure of tables.

It is built for wide tables. HBase is horizontally scalable.	It is thin and built for small tables. Hard to scale.
No transactions are there in HBase.	RDBMS is transactional.
It has de-normalized data.	It will have normalized data.
It is good for semi-structured as well as structured data.	It is good for structured data.

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.

Where to Use HBase

- Apache HBase is used to have random, real-time read/write access to Big Data.
- It hosts very large tables on top of clusters of commodity hardware.
- Apache HBase is a non-relational database modeled after Google's Bigtable. Bigtable acts up on Google File System, likewise Apache HBase works on top of Hadoop and HDFS.

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.

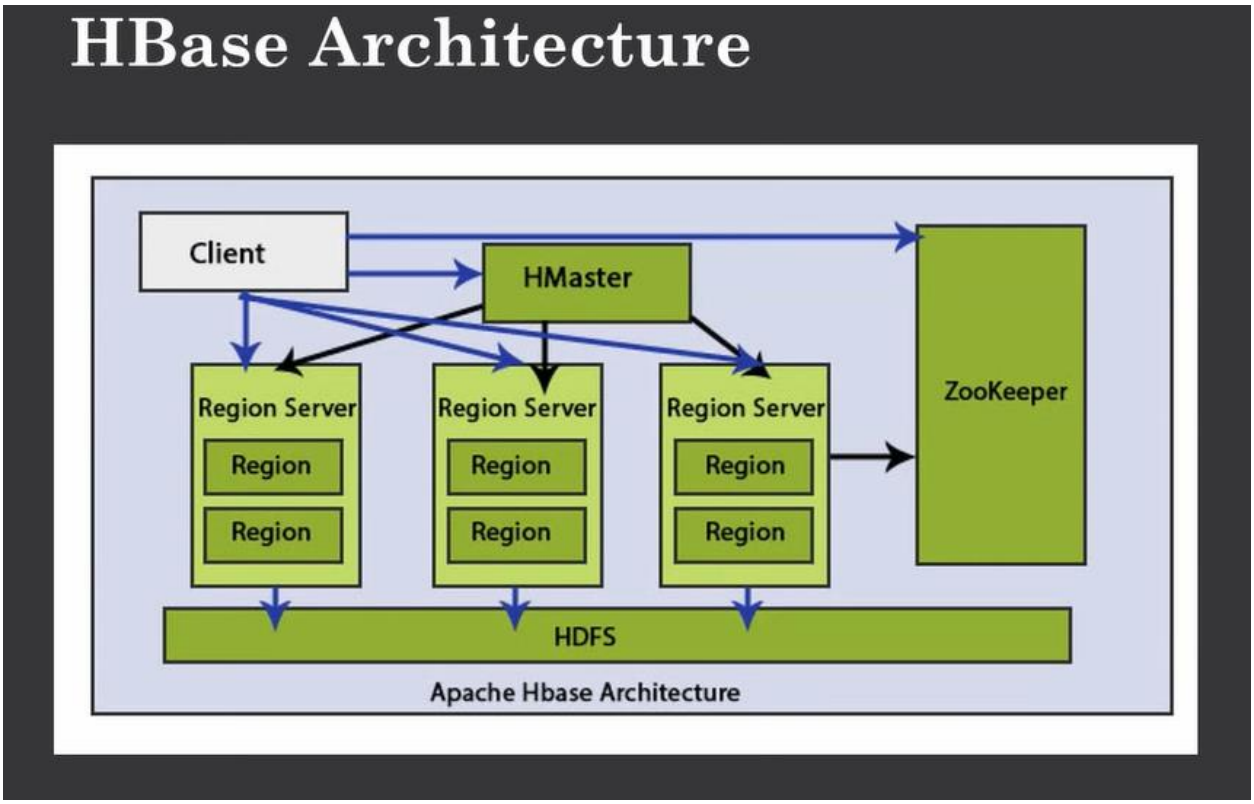
HBase History

Year	Event
Nov 2006	Google released the paper on BigTable.
Feb 2007	Initial HBase prototype was created as a Hadoop contribution.
Oct 2007	The first usable HBase along with Hadoop 0.15.0 was released.
Jan 2008	HBase became the sub project of Hadoop.
Oct 2008	HBase 0.18.1 was released.
Jan 2009	HBase 0.19.0 was released.
Sept 2009	HBase 0.20.0 was released.
May 2010	HBase became Apache top-level project.

Hbase Architecture

In HBase, tables are split into regions and are served by the region servers. Regions are vertically divided by column families into “Stores”. Stores are saved as files in HDFS. Shown below is the architecture of HBase.

Note: The term ‘store’ is used for regions to explain the storage structure.



HBase has three major components: the client library, a master server, and region servers. Region servers can be added or removed as per requirement.

MasterServer

The master server:

- Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.
- Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
- Maintains the state of the cluster by negotiating the load balancing.
- Is responsible for schema changes and other metadata operations such as creation of tables and column families.

Regions

❖ Regions are nothing but tables that are split up and spread across the region servers.

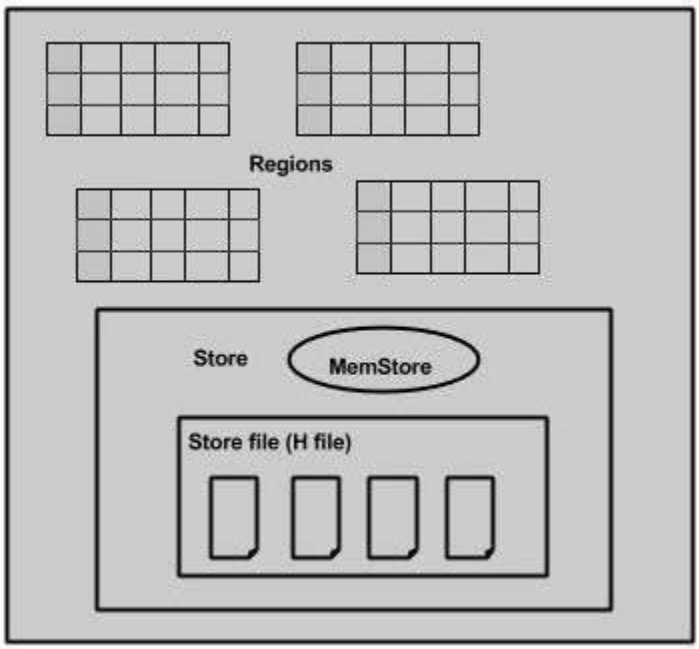
❖ Automatically done.

Region server:

The region servers have regions that:

- Communicate with the client and handle data-related operations.
- Handle read and write requests for all the regions under it.
- Decide the size of the region by following the region size thresholds.

When we take a deeper look into the region server, it contain regions and stores as shown below:



The store contains memory store and HFiles. Memstore is just like a cache memory. Anything that is entered into the HBase is stored here initially. Later, the data is transferred and saved in Hfiles as blocks and the memstore is flushed.

Zookeeper:

- Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc.
- ZooKeeper uses temporary nodes to track region servers, and master servers use these nodes to find out which servers are available.

- In addition to availability, the nodes are also used to track server failures or network partitions.
- Clients communicate with region servers via zookeeper.

Cassandra

What is Cassandra

Apache Cassandra is highly scalable, high performance, distributed NoSQL database. Cassandra is designed to handle huge amounts of data across many commodity servers, providing high availability without a single point of failure.

Cassandra has a distributed architecture which is capable to handle a huge amount of data. Data is placed on different machines with more than one replication factor to attain a high availability without a single point of failure.

Important Points of Cassandra

- ❖ Cassandra is a column-oriented database.
- ❖ Cassandra is scalable, consistent, and fault-tolerant.
- ❖ Cassandra's distribution design is based on Amazon's Dynamo and its data model on Google's Bigtable.
- ❖ Cassandra is created at Facebook. It is totally different from relational database management systems.
- ❖ Cassandra follows 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 like Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

Features of Cassandra

There are a lot of outstanding technical features which makes Cassandra very popular. Following is a list of some popular features of Cassandra:

High Scalability

Cassandra is highly scalable which facilitates you to add more hardware to attach more customers and more data as per requirement.

Rigid Architecture

Cassandra has not a 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. It increases your throughput because it facilitates you to increase the number of nodes in the cluster. Therefore it maintains a quick response time.

Fault tolerant

Cassandra is fault tolerant. Suppose, there are 4 nodes in a cluster, here each node has a copy of same data. If one node is no longer serving then other three nodes can served as per request.

Flexible Data Storage

Cassandra supports all possible data formats like structured, semi-structured, and unstructured. It facilitates you to make changes to your data structures according to your need.

Easy Data Distribution

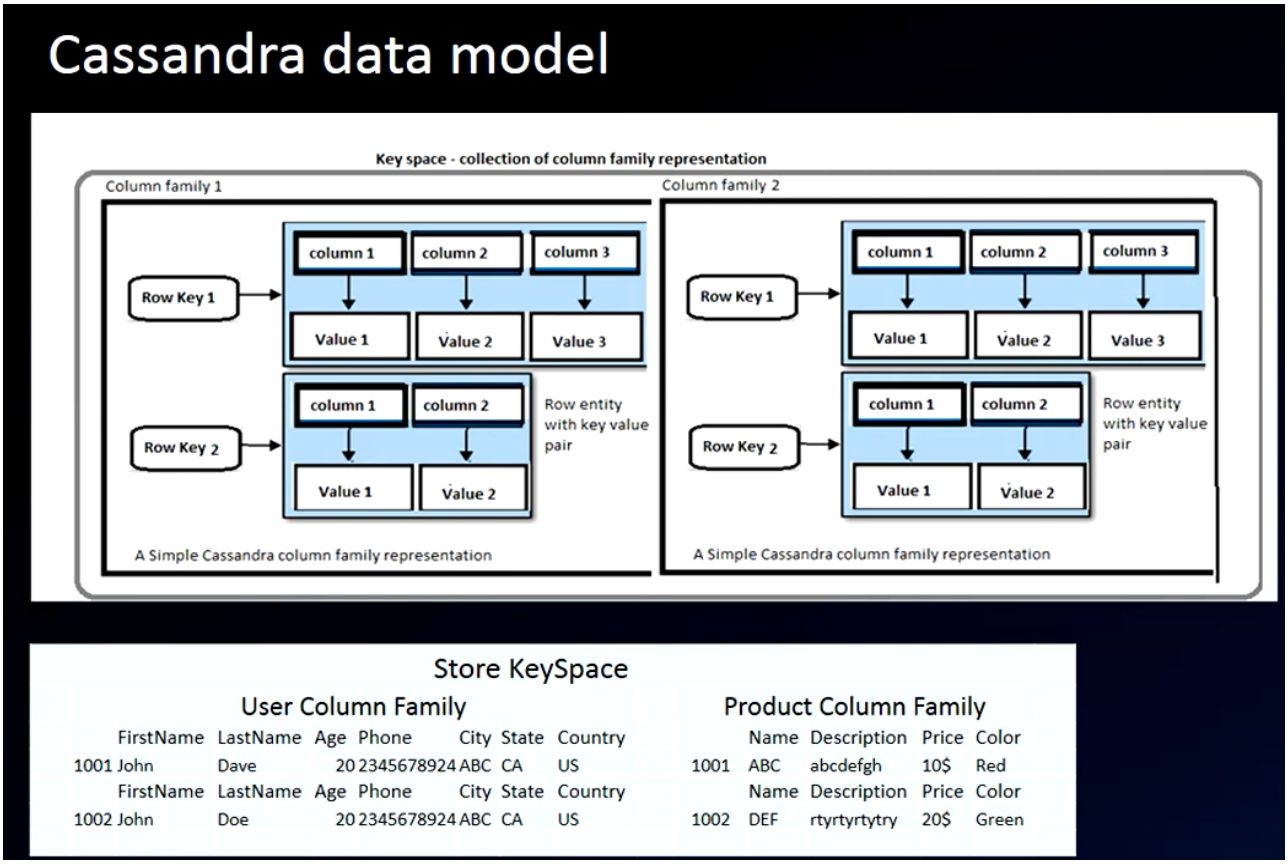
Data distribution in Cassandra is very easy because it provides the flexibility to distribute data where you need by replicating data across multiple data centers.

Fast writes

Cassandra was designed to run on cheap commodity hardware. It performs fast writes and can store hundreds of terabytes of data, without sacrificing the read efficiency.

Cassandra Data Model

The Cassandra Data Model is designed to handle large-scale, distributed data with high availability and fault tolerance. Unlike traditional relational databases, Apache Cassandra uses a NoSQL, column-family-based model that prioritizes scalability and fast write performance. Here's a breakdown of its key components:



1.Column

A column is the basic data structure of Cassandra with three values, A name (also called the column key), value, and time stamp. Given below is the structure of a column.

Column		
name : byte[]	value : byte[]	clock : clock[]

2.Row

In Cassandra, a row is the fundamental unit of data storage and is uniquely identified by a primary key. Each row belongs to a specific table and consists of one or more columns that hold the actual data. The primary key not only ensures uniqueness of the row but also determines how the data is distributed and sorted within the database. Rows with the same partition key are stored together on the same node, enabling fast access to related data. Unlike traditional relational databases, rows in Cassandra can have a

variable number of columns, allowing for a flexible and efficient data structure that supports high scalability and performance in distributed environments.

Example:

Imagine a table of orders:

User_ID (Partition Key)	Order_ID	Item
1001	A1	Laptop
1001	A2	Mouse
1002	B1	Smartphone

- All orders for User_ID 1001 (A1, A2) are stored together on the same node.
- So when you query "Give me all orders for user 1001", the system quickly fetches them from one place.

3.Column Family

A column family in Cassandra is like a table in traditional databases. It holds rows, and each row has one or more columns. Rows are organized using a primary key, and the columns inside each row are stored in a sorted order. Unlike traditional tables, each row in a column family can have different columns, making it more flexible. Although earlier versions of Cassandra used the term "column family," modern Cassandra (using CQL) simply calls it a table. This flexible structure helps Cassandra perform well in distributed systems.

4.Cluster

Cassandra database is distributed over several machines that operate together. The outermost container is known as the Cluster. For failure handling, every node contains a replica; in case of a failure, replica takes charge. Cassandra arranges the nodes in a cluster, in a ring format, and assigns data to them.

5.Key space

Key space is the outermost container for data in Cassandra. The basic attributes of Key space in Cassandra are:

Replication factor - It is the number of machines in the cluster that will receive copies of the same data.

Replica placement strategy - Replica placement strategy in Cassandra decides how copies of data (replicas) are stored across the nodes in the cluster. Common strategies include SimpleStrategy (for single data centers), and NetworkTopologyStrategy (for multiple data centers with rack awareness). These strategies help ensure fault tolerance and high availability by placing replicas on different nodes or racks.

6.SuperColumn

A super column is a special type of column in Cassandra that contains a group of related columns. Instead of holding a single value like a regular column, a super column holds a map of sub-columns, organized as a key-value structure.

Think of it like this:

- A column = (name, value)
- A super column = (name, {sub-column1, sub-column2, ...})

Given below is the structure of a super column.

```
UserData = {
  "Rohan": {
    "PersonalInfo": {
      "name": "Rohan",
      "age": "28"
    },
    "ContactInfo": {
      "email": "rohan@example.com",
      "phone": "1234567890"
    }
```

}
}
Here, PersonalInfo and ContactInfo are super columns containing related sub-columns.

What is Apache Pig?

Apache Pig is a high-level platform used for analyzing and processing large data sets in the Hadoop ecosystem. It uses a scripting language called Pig Latin, which is similar to SQL but designed specifically for parallel processing of data. Pig simplifies the development of MapReduce programs by allowing users to write data transformation tasks like filtering, joining, sorting, and grouping in a more readable and concise way. Internally, Pig scripts are converted into MapReduce jobs, which are executed on a Hadoop cluster.

One of the key strengths of Apache Pig is its ability to handle both structured and semi-structured data. It is highly flexible and supports User Defined Functions (UDFs), which can be written in Java, Python, or other languages to perform custom operations. Pig is commonly used for ETL (Extract, Transform, Load) processes, data preprocessing, and quick prototyping of big data workflows. It is especially useful for developers who prefer a scripting approach over writing complex Java-based MapReduce programs.

Why Do We Need Apache Pig?

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 20 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, filer, 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.
- UDFs: A UDF is a function written by the user (usually in Java) to add custom processing logic. You can then call this function in your Pig script just like any built-in Pig function.
- 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 MapReduce

Listed below are the major differences between Apache Pig and MapReduce.

Apache Pig	MapReduce
Apache Pig is a data flow language.	MapReduce is a data processing paradigm.
It is a high level language.	MapReduce is low level and rigid.
Performing a Join operation in Apache Pig is pretty simple.	It is quite difficult in MapReduce to perform a Join operation between datasets.
Any novice programmer with a basic knowledge of SQL can work conveniently with Apache Pig.	You need to know Java to work with MapReduce, because MapReduce programs are usually written in Java.

Apache Pig uses multi-query approach, thereby reducing the length of the codes to a great extent.	MapReduce will require almost 20 times more the number of lines to perform the same task.
There is no need for compilation. On execution, every Apache Pig operator is converted internally into a MapReduce job.	MapReduce jobs have a long compilation process.

Apache Pig Vs SQL

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

Pig	SQL
Pig Latin is a procedural language.	SQL is a declarative language.
In Apache Pig, schema 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 nested relational.(i.e Tuples:a group of fields Bags: a collection of tuples Maps: key-value pairs)	The data model used in SQL is flat relational.
Apache Pig provides limited opportunity for Query optimization.	There is more opportunity for query optimization in SQL.

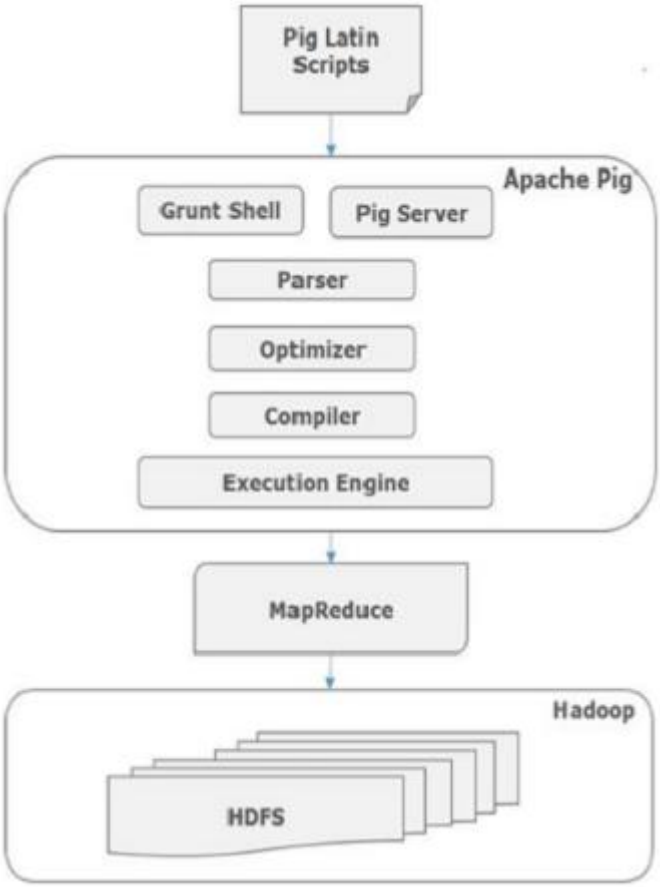
Apache Pig - Architecture

The language used to analyze data in Hadoop using Pig is known as Pig Latin. It is a high level data processing language which provides a rich set of data types and operators to perform various operations on the data.

To perform a particular task Programmers using Pig, programmers need to write a Pig script using the Pig Latin language, and execute them using any

of the execution mechanisms (Grunt Shell, UDFs, Embedded). After execution, these scripts will go through a series of transformations applied by the Pig Framework, to produce the desired output.

Internally, Apache Pig converts these scripts into a series of MapReduce jobs, and thus, it makes the programmers job easy. The architecture of Apache Pig is shown below.



Apache Pig Components

As shown in the figure, there are various components in the Apache Pig framework. Let us take a look at the major components.

Parser

Initially the Pig Scripts are handled by the Parser. It checks the syntax of the script, does type checking, and other miscellaneous checks. The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators.

In the DAG, the logical operators of the script are represented as the nodes and the data flows are represented as edges.

Optimizer

The optimizer improves the logical plan for better performance for example, by removing unnecessary steps or reordering operations.

Compiler

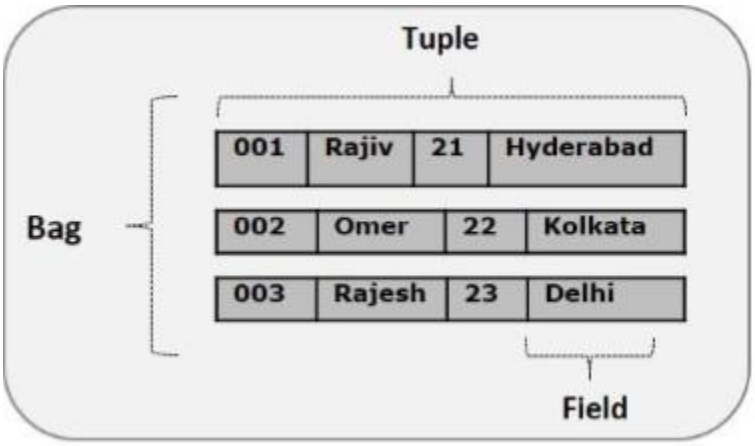
The compiler compiles the optimized logical plan into a series of MapReduce jobs.

Execution engine

Finally the MapReduce jobs are submitted to Hadoop in a sorted order. Finally, these MapReduce jobs are executed on Hadoop producing the desired results.

Pig Latin Data Model

The data model of Pig Latin is fully nested and it allows complex non-atomic datatypes such as map and tuple. Given below is the diagrammatical representation of Pig Latins data model.



Atom
Any single value in Pig Latin, irrespective of their data, type is known as an Atom. It is stored as string and can be used as string and number. int, long, float, double, chararray, and bytearray are the atomic values of Pig. A piece of data or a simple atomic value is known as a field.
Example − raja or 30

Tuple
A record that is formed by an ordered set of fields is known as a tuple, the fields can be of any type. A tuple is similar to a row in a table of RDBMS.
Example − (Raja, 30)

Bag
A bag is an unordered set of tuples. In other words, a collection of tuples (non-unique) is known as a bag. Each tuple can have any number of fields (flexible schema). A bag is represented by {}. It is similar to a table in RDBMS, but unlike a table in RDBMS, it is not necessary that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.
Example − {(Raja, 30), (Mohammad, 45)}
A bag can be a field in a relation; in that context, it is known as inner bag.
Example − {Raja, 30, **{9848022338, raja@gmail.com,}**}

Map
A map (or data map) is a set of key-value pairs. The key needs to be of type chararray and should be unique. The value might be of any type. It is represented by []
Example − [name#Raja, age#30]

Relation
A relation is a bag of tuples. The relations in Pig Latin are unordered (there is no guarantee that tuples are processed in any particular order).

Developing and Testing Pig Latin Script

Pig provides several tools and diagnostic operators to help you develop your applications. In this section we will explore these and also look at some tools others have written to make it easier to develop Pig with standard editors and integrated development environments (IDEs).

Syntax Highlighting and Checking

Syntax highlighting often helps users write code correctly, at least syntactically, the first time around. Syntax highlighting packages exist for several popular editors. Pig Latin syntax highlighting packages.

Tool	URL
Eclipse	http://code.google.com/p/pig-eclipse
Emacs	http://github.com/cloudera/piglatin mode,http://sf.net/projects/pig-mode
TextMate	http://www.github.com/kevinweil/pig.tmbundle
Vim	http://www.vim.org/scripts/script.php?script_id=218

What is Hive

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

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

Hive is not

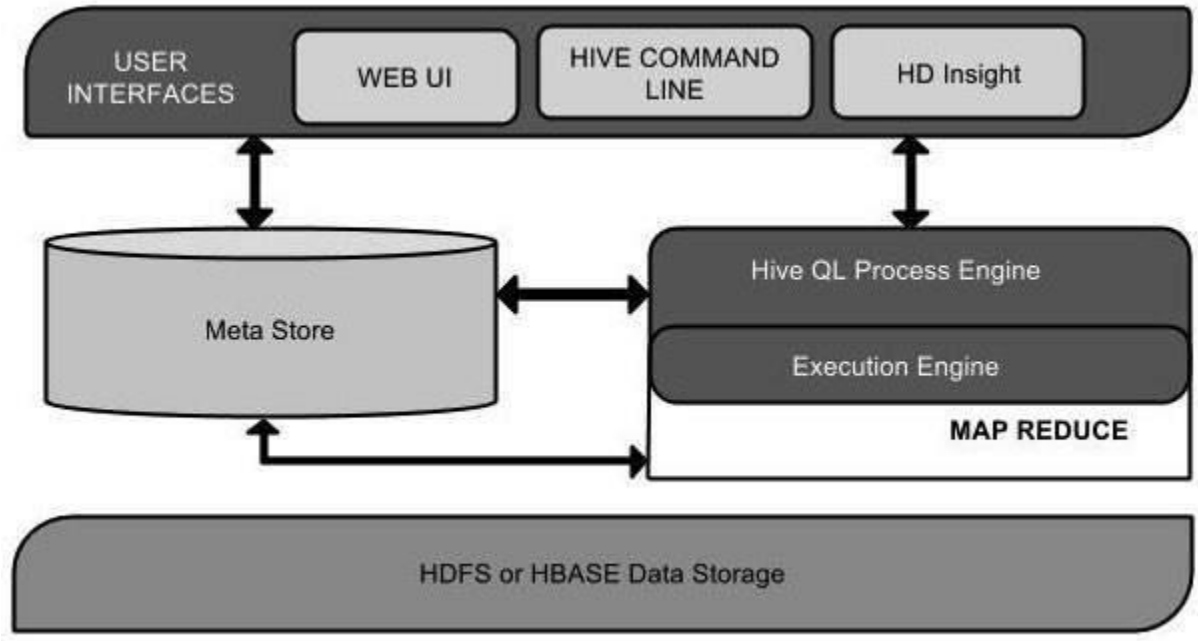
- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

Features of Hive

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

Architecture of Hive

The following component diagram depicts the architecture of Hive:



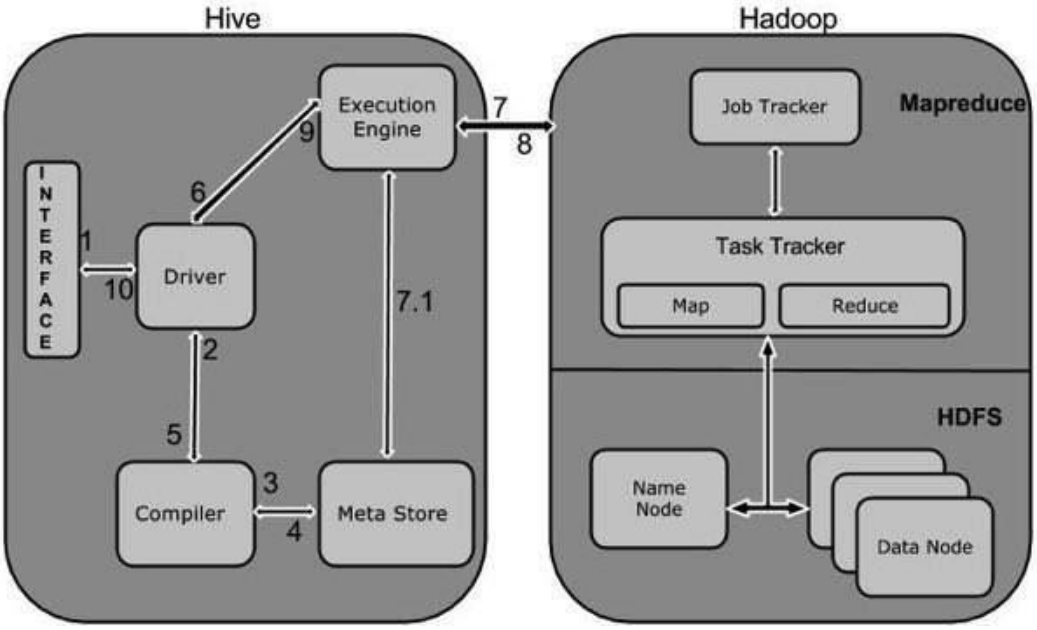
This component diagram contains different units. The following table describes each unit:

Unit Name	Operation

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

Working of Hive

The following diagram depicts the workflow between Hive and Hadoop.



The following table defines how Hive interacts with Hadoop framework:

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

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