10-2016

POC on Credit Card "e-Statement" Details Generation for ANZ Bank

Kasi Naga Venkata Chaitanya Konatalapalli
koka1401@stcloudstate.edu

POC on Credit Card "e-Statement" Details Generation for ANZ Bank

by

Kasi Naga Venkata Chaitanya Konatalapalli

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfilment of the Requirements

for the Degree of

Master of Science

in Information Assurance

October, 2016

Starred Paper Committee:
Dr. Dennis Guster, Chairperson
Dr. Susantha Herath
Dr. Balasubramanian Kasi

Abstract

The storage and processing of data are major issues in information technology today. Every organization has been rapidly growing data day by day, and it becomes tough for the information systems to process and respond to the various queries required of them. Banking is one such industry which needs to handle millions of data records each time. Utilizing Hadoop as a solution is one way to handle these records more effectively and in less time. From this Proof of Concept (POC), the time difference between executing queries will take much less compared to the existing database system. The growth of data challenges cutting-edge companies like Google, Yahoo, Amazon, Microsoft and many more like them. They need to go through the terabytes and even petabytes of data to figure out issues regarding these websites which are popular among people. The tools they had at the time were not equipped to cope with this issue. Then Google presented MapReduce, a system they had used to cope with this issue. The majority of companies were facing the same issue as Google, so they did not want to develop another system like Google developed, and this system was suitable for all of them. After some time, this system became open source for all of them, and many companies appreciated this effort. That system was named as Hadoop, and today it is major part of the computing world. Due to its efficiency, many more companies are going to rely on Hadoop, and they are going to establish this system in their companies. Hadoop is used for running huge distributed programs so its simplicity and accessibility give it an edge over writing and running distributed programs. Any good programmer can create his own Hadoop instance in minutes, and it is also very cheap to create. Hadoop is moreover, very scalable and robust. Due to Hadoop's features, it is getting very popular in the academic and industrial world. MapReduce is a model of data processing and in this model, data can easily be scalable over multiple systems. In this model, two terms are used for data processing, and those are mappers and reducers. Sometimes it is nontrivial to decompose the data application into mappers and into reducers. However, once you write an application in the MapReduce format then scaling of that application to run over many hundreds of systems is not a big issue. Some minor changes may still be required to take place, however due to its efficiency and scalability, programmers are attracted towards MapReduce like a bear towards honey. According to experts, this era is an era of development of unbelievable things, and these developments require large systems with larger data storage in them to cope with the immense storage issues. Hadoop plays an effective role to cope with this issue with its scalability and many more striking features. Hadoop is also an astonishing development. There is a challenge that must be fulfilled and that is how the existing data will move to the Hadoop infrastructure, when the existing data infrastructure is based on traditional relational database and Structured Query Language (SQL). Meanwhile there is the concept of Hive. Hive provides a dialect of SQL named as Hive Query Language to fulfil the query of data storage in the cluster of Hadoop instances. Hive does not work as a database, instead it is bound to the limitations imposed by the constraints of Hadoop. The most surprising limitation is that it cannot provide record level updates, such as insert and delete. You can only make new tables, or you can perform queries to output results to files. Hive also does not provide transactional data.

Acknowledgement

The successful completion of this paper could not have been possible without the guidance of my beloved professors, Dr. Dennis Guster and Dr. Susantha Herath. I also would like to thank Dr. Balasubramanian Kasi for being part of the committee and finding the time to read my paper.

Table of Contents

Chapter Page

Chapter Page

List of Tables

List of Figures

Figure Page

Chapter I: Introduction

The storage and processing of data are major issues today. Every organization has been growing data day by day, and it has become increasingly difficult for the systems to process and respond to the queries. Banking is one such industry where there is a need to handle millions of queries and data records.

Problem Statement Description

I will look at the generation of an e-Statement through an email service for Business Credit Card Accounts. The bank's existing system collects the account data, account details, transaction details, and processes them. It then generates a file, which is utilized by the email sending system to send out e-statements. Approximately five million e-statements are needed to be treated by the existing system. This process of collecting and processing the data will take around 18 hours of batch production runtime, which is very long.

Proposed Solution

The solution proposed for batch processing of unstructured data is Hadoop because it can handle processing large volumes of unstructured data and can do so fast by its use of a parallelism technique. For this type of raw and confidential data (bank transactions), the backup would be crucial, and thus Hadoop deals with the backup as three replicas of the data. This adds to the security of the backups rather than just creating a secondary name node. The Hadoop approach of a map and reduce makes the processing of data fast. This MapReduce processing can be achieved through JAVA, PIG and HIVE. And some of the process can be done with SQOOP, which is a map only processor, and is the easiest way of importing the data from SQL (Structured Query Language). So, Hadoop is selected as a solution for this project because, it is

simpler to join, filter and sort the data sets and is also good at processing and maintaining the backup of the data (White, 2012).

Figure 1 explains the data generated from different files should be integrated as a CNE file to send to the customer as e-mail.

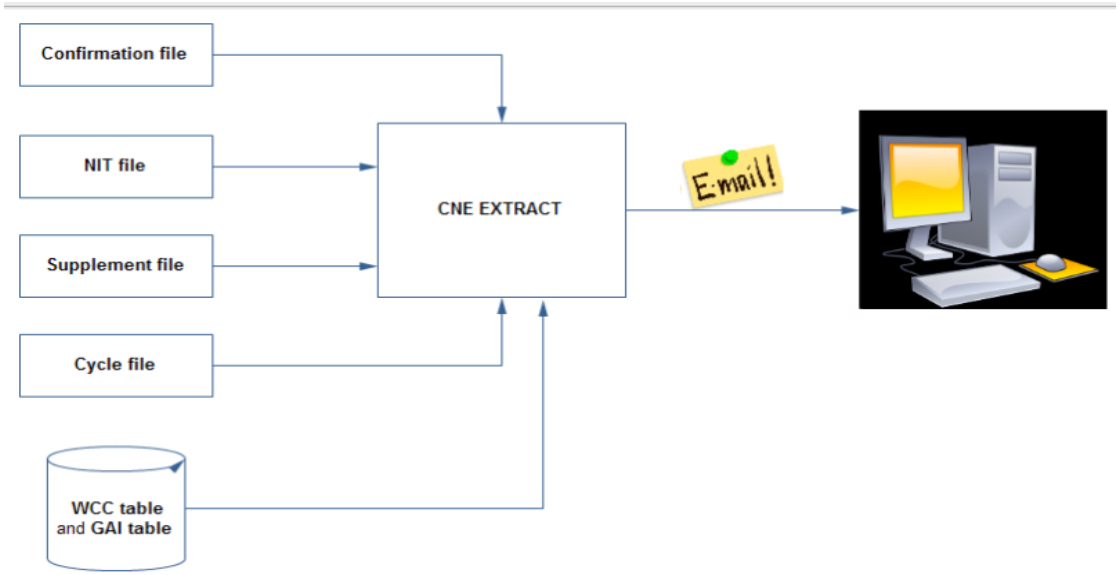


Figure 1. Illustration of Basic Date Flow for CNE File

Conclusion

According to Chapter I, the discussion concludes that Hadoop is useful in dealing with large data sets and also admirably performs handsome processing with MapReduce. So, it is worth selecting it as a solution for this project.

Chapter II: Background and Review of Literature

Purpose of Hadoop

It is difficult for large amounts of data on a single drive to be accessed through reading and writing and causes it to be slower for a system in-charge. Hadoop writes the data sets into chunks and stores them as three data blocks with 64 MB (64 MB not mandatory as per Admin settings) as a default. The loss of data may also happen for data in a single drive. However, Hadoop has three data blocks for each data set at the time of loss making a built-in redundant backup. The other problem with earlier systems is combining two data sets together becomes more challenging. However, Hadoop can manage two different data sets when combining them through a key-value pair's technique. Hadoop is a versatile tool that allows a new user to access the power of distributed computing. Hadoop transfers the code instead of data in distributed computing, so by this way, Hadoop avoids the costly transmission of data when working with massive data sets. A user does not have to worry about assigning the responsibilities to the nodes. It is Hadoop's duty to assign the task to the nodes and the user can manage his most important work at that time. Hadoop provides extensive storage, not only for a single type of data, but for all kinds of data that can be stored. Hadoop has enormous processing power and the ability to manage virtually unlimited concurrent tasks or jobs.

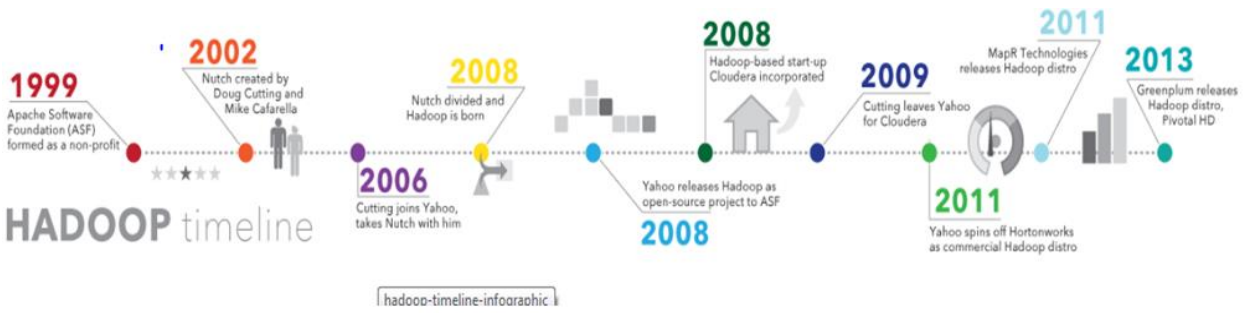


Figure 2. Evolution of Hadoop (Tutorials Point, n.d.a)

Scope of Hadoop

A main benefit of using Hadoop is that it can handle large volumes of data and is more efficient in handling data losses, which are relevant for bank transactions and even fair enough for processing the data faster, as per the current project.

With the constant increase in volumes and varieties of data, especially from social media and the Internet of Things (IoT), Hadoop can cope with these scenarios making it a vital technology.

The computing power of Hadoop is very fast. The more nodes that are used, the more processing power it will provide to the nodes. If any hardware failure has occurred, then the data and application will remain safe and protected by Hadoop. If one node goes down, another node will automatically take its responsibilities and perform operations instead of that node. These nodes automatically communicate with one another and make sure communication is not broken, and distributed computing does not fail. There is not only one original copy of the data that is saved, but there are multiple copies of data collected in Hadoop.

Unlike other traditional relational databases, you do not have to pre-process the data before it is stored. You can store unlimited data as you want and later decide about the processing of the data. The data stored can be unstructured like videos, images, and text.

Hadoop is an open-source framework and totally free of cost and uses hardware to store massive sets of data.

A little administration is required while growing your system to handle more data by adding more nodes to your distributed computing system. As the increase in distributed computing and variety of data has grown, Hadoop is now serving many companies that are managing and storing massive sets of data to manage their business. Hadoop is based on Google's technologies, and it seems that Google will further influence its future.

Difference between the Hadoop and Other Systems

Hadoop is a MapReduce processor, which deals with large amounts of data and processes within very little time. The Relational Database Management System (RDBMS) can also handle large data sets by using more disks, but it is slower because of its seek times and also troubles with the transfer rate too. Table 1 illustrates the difference between the existing technique of an RDBMS way of problem solving and the Hadoop MapReduce way of solving the problem.

Table 1

Difference between RDBMS and Hadoop-MapReduce

RDBMS	Hadoop-MapReduce
Can deal with date in GB efficiently	Can deal with PB of data
Both Interactive and Batch processing	Batch processing is accepted
Only structural data can be processed	Structural, semi-structural and even unstructured data can be processed
Reading and writing of data can be done many times	All write once and read many
Seek times lead to more for large data	Reduction of seek time can be reduced by setting data block size
B-Tree is used for updating the database	MapReduce uses sort/merge for rebuilding the database
RDBMS is good for continuous updating of datasets	Hadoop is good for write once datasets

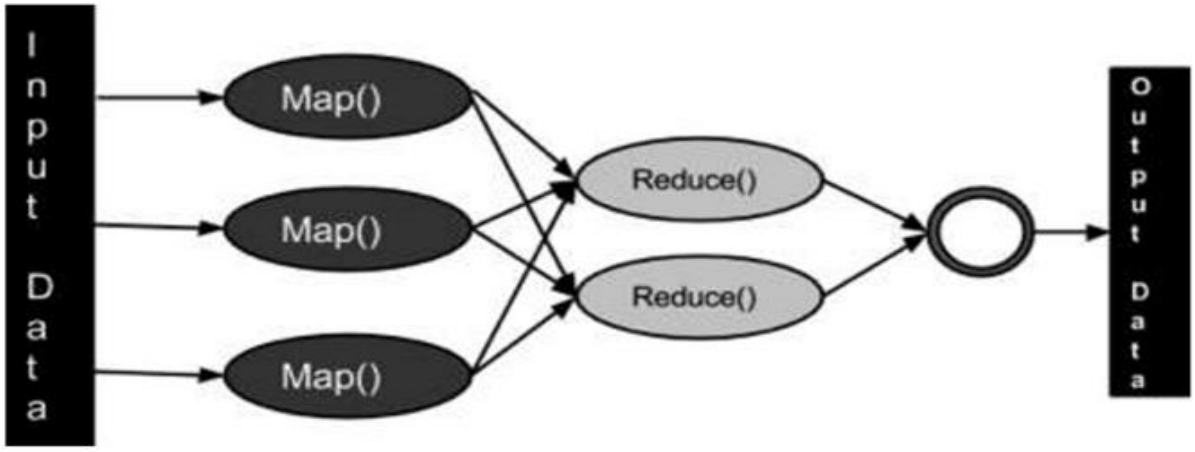


Figure 3. Map Reduce Pictorial Representation (Tutorials Point, n.d.a)

Difference between Hadoop and Other Distributed Systems

There are several differences between Hadoop and other storage systems listed in Table 2. The other systems are using different strategies for the processing and storing of data. The

storing of data is easily handled but while accessing the data the complexity rises. In the case of Hadoop, it is very useful in obtaining the data, because it knows the data locality through Namenode. The other systems cannot handle these failures even remotely as well. However, Hadoop can process to some extent by re-promoting and re-running the failed tasks for better results.

Table 2

Difference between Hadoop and Other Distributed Systems

Hadoop	Other File Systems (SAN)
Hadoop is an open-source distributed system	The cost of storing relies on
Uses less network bandwidth for MapReduce workloads	Uses more network bandwidth for workloads
Only structural data can be processed	Structural, semi-structural and even unstructured data can be processed
Data is reliable	Data is safe too
Data is immutable	Data can be modified
Node accessing is smooth through data locality as a feature	Node assessing is the place where complexity rises
Programmer thinks only about key-value generations for data flow	Developer needs to analyse the higher level algorithms for data flow

Difference between Hadoop MapReduce and Other MapReduces

There are different types of MapReduce available other than the Hadoop MapReduce. Among them are Google MapReduce, which follows all the rules as Hadoop MapReduce does, but there are some differences, which are listed in Table 3.

Table 3

Differences between Hadoop and Google MapReduce

Hadoop	Other File Systems (SAN)
Hadoop MapReduce is stored in HDFS	Goodle MapReduce stored in Google file system
Hadoop is an open source	Not mentioned
Hadoop MapReduce can be written in Java, Pig, and Hive	Google MapReduce can be written in C++

Big Data

The term big data is used here for a huge set of data that are enormous and cannot be processed by traditional computing mechanisms. Big data is not only data, but it can be a tool, a framework and maybe even some other technique.

Different devices can produce big data. Some are listed below.

- Black Box Data: It is part of helicopters, airplanes, and jet planes (difference between airplane and Jet plane?). It is used to capture and store voices of the flight crew, recording of microphones and earphones.
- Social Media Data: Social media like Facebook and Twitter holds data of millions of people from distinct locations across the globe.
- Stock Exchange Data: The stock exchange data is about buy and sell decisions made on shares of different companies by traders.
- Power Grid Data: It is data of the power grid and consists of information consumed by a node on the base station.
- Transport Data: It includes data about models, distance, and availability of a vehicle.
- Search Engine Data: Search engines gathers a lot of data from different databases.

Figure 4. Big Data (Tutorials Point, n.d.a)

So big data includes massive volume, high velocity, and extensible types of data. Three types of data are found in it:

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

Technologies of Big Data

Big data technologies are mostly used for accuracy in results and decision-making. It provides more accurate analysis, which results in greater operational efficiencies, cost reductions, and minimized risks for business.

To manage the power and depth of big data, you need an infrastructure that can manage and process huge volumes of structured and unstructured data in real time and can protect the privacy of data and security.

- Operational Big Data: Data is primarily captured and stored. This includes systems like MongoDB that provides operational capabilities for real-time workloads.
- Analytical Big Data: It consists of Massive Parallel Processing database systems and MapReduce that provide analytical capabilities for complex data.

Table 4

Operational vs. Analytical Systems

	Operational	Analytical
Latency	1 ms = 100 ms	1 minj – 100 min
Concurency	1,000 – 100,000	1 – 10
Access Pattern	Writes and Reads	Reads
Queries	Selective	Unselective
Data Scope	Operational	Retrospective
End User	Customer	Data Scientist
Technology	NoSQL	Map Reduce, MPP Database

Big Data Challenges

Following are the challenges with big data management:

- Capturing of data
- Curating of data

- Storage of data
- Searching of data
- Sharing of data
- Transferring of data
- Analysis of data
- Presentation of data.

Traditional Approach:

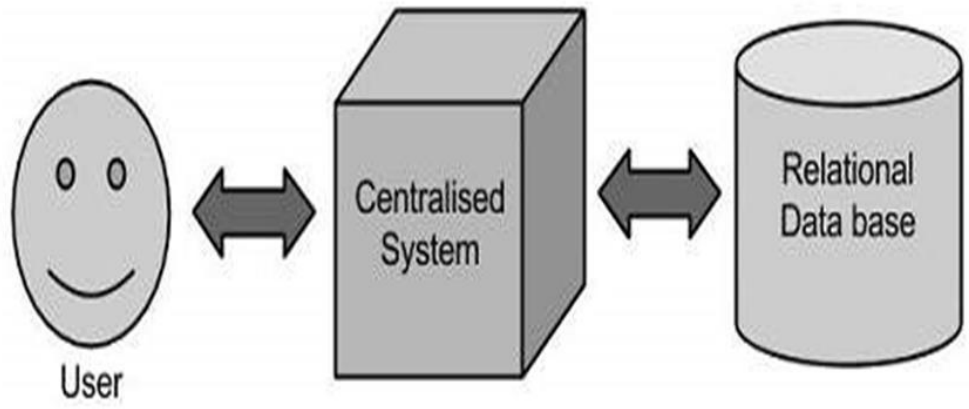


Figure 5. Data Passing Through Traditional Approach (Tutorials Point, n.b.a)

Limitation: This approach is only suitable for lesser amounts of data and can be used by standard database servers. But in the case of big data, the need for the use of Hadoop comes into play.

Solution by Google

Google solved this problem by introducing an algorithm called MapReduce. This algorithm divides the task into several parts and distributes it among several nodes called

computers or any other communicating devices connected to a network, and collects the results to make the final solutions.

Figure 6 diagram shows various commodity hardware which could be single CPU machines or servers with higher capacity processing capability.

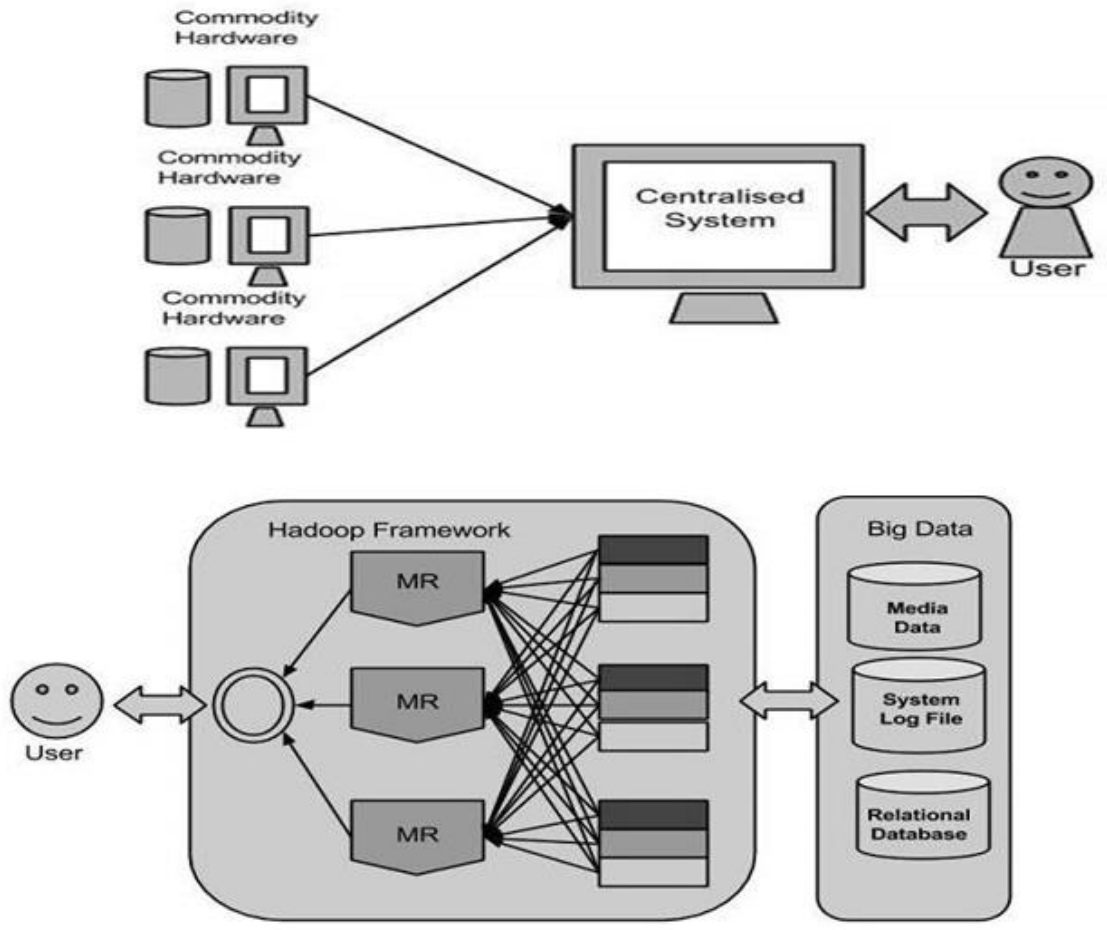


Figure 6. Hadoop Framework (Tutorials Point, n.b.a)

Hadoop Architecture

Hadoop architecture consists of four modules:

- Hadoop Common: Hadoop Common is the libraries and utilities for Java that are required by other modules. These files provide the operating system and file system level abstraction and contain the necessary Java files and other scripts that are needed to start Hadoop.
- Hadoop YARN: It is a framework for job scheduling and cluster resource handling.
- Hadoop Distributed File System (HDFS): Some applications require high-throughput access, and this module provides that access to these requests.
- Hadoop MapReduce: It also works like YARN and is used for parallel processing of large data sets.

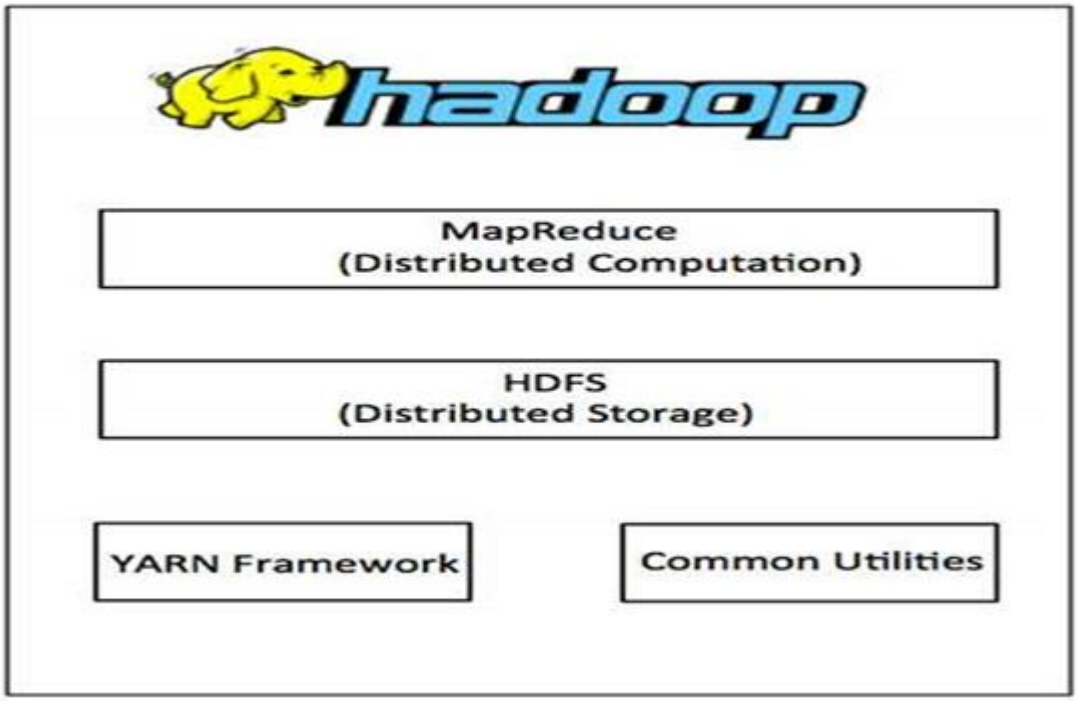


Figure 7. Hadoop Architecture (Tutorials Point, n.b.a)

Hadoop is not on the above base modules, but also contains several more modules named Apache Pig, Apache Hive, Apache HBase, Apache Spark, etc.

MapReduce

MapReduce software of the Hadoop framework for smooth writing of applications, which can process a large amount of data in parallel ways on large clusters of commodity hardware in a reliable and fault tolerant manner. MapReduce performs two types of tasks:

- Map Task: It is the first step to take input and after that break information into a set of data, where a single element is divided into tuples.
- Reduce Task: In this step, the output is taken from Map task as input and after that combines those data tuples into a smaller set of tuples. Reducer task is always performed after the map task.

Both inputs and outputs are stored in a file system, and the framework manages the scheduling of these tasks, their monitoring, and re-execution of any failed task. Every cluster node has a single master JobTracker, and one slave, TaskTracker, and the MapReduce framework relies upon them. The master can manage resource management, tracking of resources, scheduling of job's basic tasks on the slaves, monitoring them and re-execution of any failed task. The slave TaskTracker executes the tasks as directed by the master and provides the status of the task to the master. If the JobTracker goes down due to some issue or defect than Hadoop, MapReduce service will fail it as a single point of failure for Hadoop MapReduce.

Hadoop Distributed File System (HDFS)

Hadoop can perform work directly with any mountable distributed file system like a local file system, HFTP FS, S3 FS and others but the most common file system that is used by

Hadoop is Hadoop Distributed File System (HDFS). Hadoop Distributed File System is based on the Google File System (GFS), and it provides a distributed file system that is designed to run on large clusters (thousands of nodes) of small computer machines in a reliable and fault tolerant way. HDFS uses a master/slave architecture where the master consists of a single **NameNode** that manages the file system metadata and one or more slave **DataNodes** that store the actual data. A file in Hadoop Distributed File System namespace is divided into some blocks, and those blocks are stored in a set of DataNodes. Then NameNode chooses the mapping of blocks to the DataNodes. Then DataNodes takes care of read and writes operations with the file system. They also manage the creation of blocks, deletion of blocks and replications of blocks based on commands given by NameNode. HDFS provides a shell like any other file system, and a list of commands is available to interact with the file system.

Architecture of HDFS

Figure 8 shows the architecture of Hadoop Distributed File System.

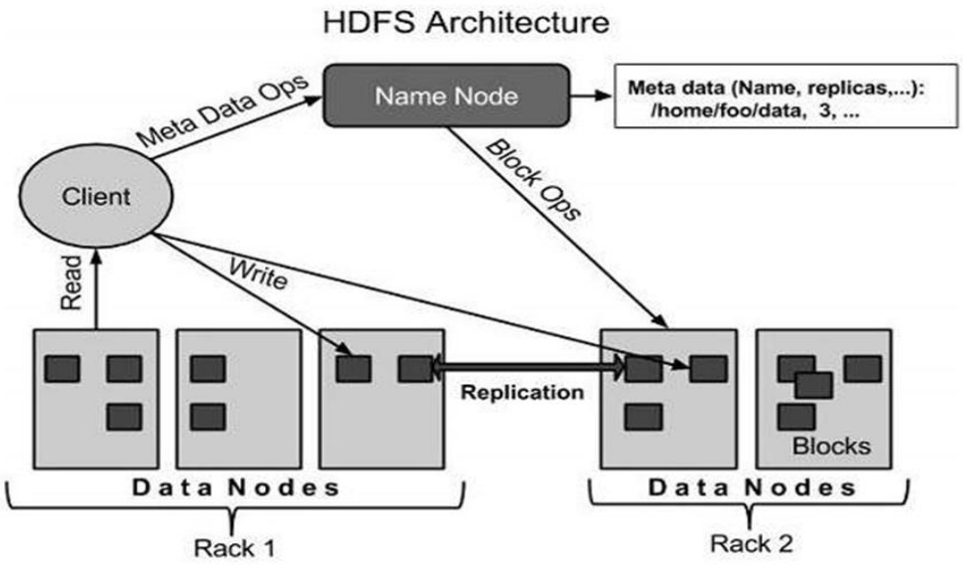


Figure 8. HDFS Architecture (Tutorials Point, n.b.a)

HDFS is based on Master-Slave mechanism and has the following components:

NameNode

It is typically built on commodity hardware that contains the Linux operating system and the Namenode software. This Namenode software can be run on commodity hardware. Any system having the Namenode can act as a master server and can perform several tasks.

- Management of namespace of a file system.
- Regulation of client access to the files.
- Execution of file system operations like closing, opening, renaming, editing, etc.

DataNode

It is also housed on commodity hardware that has a Linux operating system and DataNode software. Every node in a cluster has a DataNode for it, and these nodes handle the storage of data in their systems.

- As the request from the client, DataNodes perform read and write tasks on the file systems.
- They also perform the task of blocking of any creation, deletion, and replication according to the instructions from the NameNode.

Block

User data is stored in files in HDFS and that file in the system is further divided into one or more segments and then stored in a single data node. These segments of the file are called a block. The default size of this block is 64MB, and it can be increased according to the needs to change in the Hadoop Distributed File System configuration.

Goals of HDFS

- Fault detection and recovery: HDFS consists of a large number of commodity hardware, so the failure of components is very common. Therefore, HDFS should have a mechanism for automatic fault detection and correction.
- Huge Datasets: HDFS should have a large number of nodes per cluster to manage the applications servicing massive datasets.
- Hardware at data: A requested task can be done efficiently when the computation takes place near the data. Typically, where massive datasets are involved, it reduces the network traffic and increases the data throughput which speeds processing.

Files and path names to the objects inside HDFS:

1	**ls <path>** Lists the contents of the directory specified by path, showing the names, permissions, owner, size and modification date for each entry.
2	**ls <path>** Behaves like -ls, but recursively displays entries in all subdirectories of a path.
3	**du <path>** Shows disk usage, in bytes, for all the files that match the path; filenames are reported with the full HDFS protocol prefix.
4	**dus <path>** Like -du, but prints a summary of disk usage of all files/directories in the path.
5	**mv <src><dest>** Moves the file or directory indicated by src to dest, within HDFS.
6	**cp <src> <dest>** Copies the file or directory identified by src to dest, within HDFS.
7	**rm <path>** Removes the file or empty directory identified by path.
8	**rmr <path>** Removes the file or directory identified by path. Recursively deletes any child entries (i.e., files or subdirectories of the path).
9	**put <localSrc> <dest>**

	Copies the file or directory from the local file system identified by localSrc to dest within the DFS.
10	**copyFromLocal <localSrc> <dest>** Identical to -put
11	**moveFromLocal <localSrc> <dest>** Copies the file or directory from the local file system identified by localSrc to dest within HDFS, and then deletes the local copy on success.
12	**get [-crc] <src> <localDest>** Copies the file or directory in HDFS identified by src to the local file system path identified by localDest.
13	**getmerge <src> <localDest>** Retrieves all files that match the path src in HDFS, and copies them to a single, merged file in the local file system identified by localDest.
14	**cat <filen-ame>** Displays the contents of the filename on stdout.
15	**copyToLocal <src> <localDest>** Identical to -get
16	**moveToLocal <src> <localDest>** Works like -get, but deletes the HDFS copy on success.
17	**mkdir <path>** Creates a directory named path in HDFS. Creates any parent directories in path that are missing (e.g., mkdir -p in Linux).
18	**setrep [-R] [-w] rep <path>** Sets the target replication factor for files identified by the path to rep. (The actual replication factor will move toward the target over time)
19	**touchz <path>** Creates a file at path containing the current time as a timestamp. Fails if a file already exists at the path, unless the file is already size 0.
20	**test -[ezd] <path>** Returns 1 if the path exists; has zero length; or is a directory or 0 otherwise.
21	**stat [format] <path>** Prints information about the path. The format is a string, which accepts file size in blocks (%b), filename (%n), block size (%o), replication (%r), and modification date (%y, %Y).
22	**tail [-f] <file2name>** Shows the last 1KB of the file on stdout.
23	**chmod [-R] mode,mode,... <path>...**

	Changes the file permissions associated with one or more objects identified by path.... Performs changes recursively with R. mode is a 3-digit octal mode, or {augo}+/-{rwxX}. Assumes if no scope is specified and does not apply an umask.
24	**chown [-R] [owner][:[group]] <path>...** Sets the owning user and/or group for files or directories identified by path.... Sets owner recursively if -R is specified.
25	**chgrp [-R] group <path>...** Sets the owning group for files or directories identified by path.... Sets group recursively if -R is specified.
26	**help <cmd-name>** Returns usage information for one of the commands listed above. You must omit the leading '-' character in cmd.

Sqoop

Introduction

Structured data in traditional databases cannot be easily combined with unstructured data stored in HDFS.

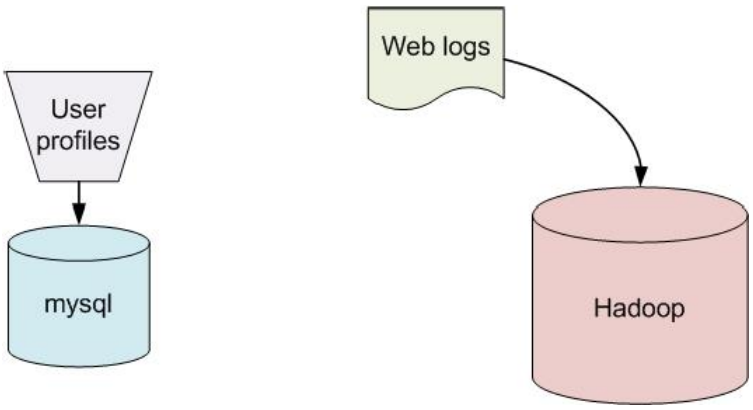


Figure 9. Storage of Data into Database and Hadoop (Tutorials Point, n.d.d)

Sqoop is also known as SQL to Hadoop. Sqoop is used for the easy import of data from many databases to HDFS. Sqoop generates code for use in MapReduce applications, and it

integrates with Hive. Sqoop is a tool designed to transfer the data among Hadoop and relational database servers. It is used to import data from relational databases like MySQL, Oracle to Hadoop HDFS, and then export from the Hadoop file system to relational databases. It performs the task of a communication bridge between these databases. Apache Software Foundation initially introduced it (Apache Software Foundation, n.d.).

How Sqoop Works

Figure 10 describes the working of Sqoop.

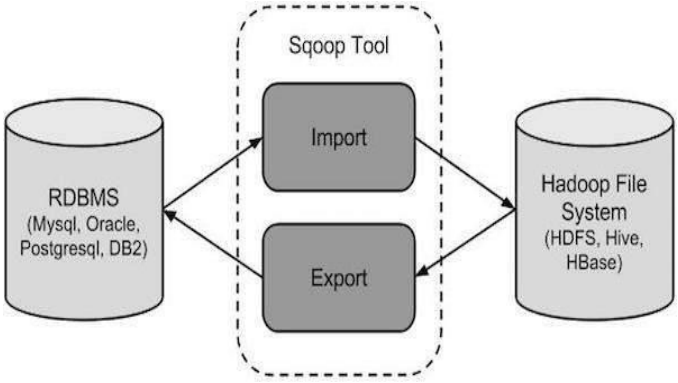


Figure 10. Working of Sqoop (Tutorials Point, n.d.d)

Sqoop Import

This import tool imports a single table from RDBMS to HDFS. Every row in that table is treated as a record in HDFS. And all files are stored as text data in text files.

Syntax. The following commands are used to import data into HDFS.

```
$ sqoop import (generic-args) (import-args)

$ sqoop-import (generic-args) (import-args)
```

Example work. I am going to make three tables named emp, emp_add, and emp_contact and these are stored in a database named as userdb in a MySQL database server.

Table emp:

id	name	deg	salary	Dept
1201	Gopal	Manager	50,000	TP
1202	Manisha	Proof reader	50,000	TP
1203	Khalil	Php dev	30,000	AC
1204	Prasanth	Php dev	30,000	AC

Table emp_add:

id	hno	street	City
1201	288A	Vgiri	Jublee
1202	108I	Aoc	Sec bad
1203	144Z	Pgutta	Hyd
1204	78B	Old city	Sec bad

Table emp_contact

id	phno	Email
1201	2356742	gopal@tp.com
1202	1661663	manisha@tp.com
1203	8887776	khalil@ac.com
1204	9988774	prasanth@ac.com

Sqoop tool “import” is used for import table data from the existing table to the Hadoop file system as a text file or a binary file. The following command is used to import the emp table from MySQL database server to HDFS.

```
$ sqoop import \
--connect jdbc:mysql://localhost/userdb \
--username root \
--table emp --m 1
```

After execution, it will give the following output.

```
14/12/22 15:24:54 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5
14/12/22 15:24:56 INFO manager.MySQLManager: Preparing to use a MySQL
streaming resultset.
14/12/22 15:24:56 INFO tool.CodeGenTool: Beginning code generation
14/12/22 15:24:58 INFO manager.SqlManager: Executing SQL statement: SELECT
t.* FROM `emp` AS t LIMIT 1
14/12/22 15:24:58 INFO manager.SqlManager: Executing SQL statement: SELECT
t.* FROM `emp` AS t LIMIT 1
14/12/22 15:24:58 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is
/usr/local/hadoop
14/12/22 15:25:11 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-
hadoop/compile/cebe706d23ebb1fd99c1f063ad51ebd7/emp.jar
---------------------------------------------------
---------------------------------------------------
14/12/22 15:25:40 INFO mapreduce.Job: The url to track the job:
http://localhost:8088/proxy/application_1419242001831_0001/
14/12/22 15:26:45 INFO mapreduce.Job: Job job_1419242001831_0001 running in
uber mode : false
14/12/22 15:26:45 INFO mapreduce.Job: map 0% reduce 0%
14/12/22 15:28:08 INFO mapreduce.Job: map 100% reduce 0%
14/12/22 15:28:16 INFO mapreduce.Job: Job job_1419242001831_0001 completed
successfully
---------------------------------------------------
```

```
----------------------------------------------------
14/12/22 15:28:17 INFO mapreduce.ImportJobBase: Transferred 145 bytes in
177.5849 seconds (0.8165 bytes/sec)
14/12/22 15:28:17 INFO mapreduce.ImportJobBase: Retrieved 5 records.
```

After execution for verification of output data in HDFS, insert the command below.

```
$ $HADOOP_HOME/bin/hadoop fs -cat /emp/part-m-*
```

It will show the emp table data and fields.

```
1201, gopal,    manager, 50000, TP
1202, manisha,  preader, 50000, TP
1203, kalil,    php dev, 30000, AC
1204, prasanth, php dev, 30000, AC
```

By using import tool, we can import data into a specific directory by using the following command.

```
--target-dir <new or exist directory in HDFS>
```

The following command is then used to import table emp_add data into the resulting query directory.

```
$ sqoop import \
--connect jdbc:mysql://localhost/userdb \
--username root \
--table emp_add \
--m 1 \
--target-dir /queryresult
```

The following command is then used for the verification of imported data in query the result directory.

```
$ $HADOOP_HOME/bin/hadoop fs -cat /queryresult/part-m-*
```

It will give the following output:

```
1201, 288A, vgiri,   jublee
1202, 108I, aoc,     sec-bad
1203, 144Z, pgutta,  hyd
1204, 78B,  oldcity, sec-bad
```

Sqoop Export

This export tool exports a set of files back from HDFS to RDBMS. Sqoop receives files as input contains records, which are called as rows in a table. Then those tables are read and parsed into a set of records and delimited with the user-specified delimiter. Sqoop is a sub-project of Hadoop, and it can only work on a Linux based operating system.

Syntax. The following commands are used to export the data.

```
$ sqoop export (generic-args) (export-args)
$ sqoop-export (generic-args) (export-args)
```

Example work. Now there is an employee data in HDFS file and employee data is available in emp_data file in “table emp” directory in HDFS and as follows:

```
1201, gopal,     manager, 50000, TP
1202, manisha,   preader, 50000, TP
1203, kalil,     php dev, 30000, AC
```

```
1204, prasanth,  php dev, 30000, AC
```

It is this table that is to be exported and is created manually and is present in the database from where it has to be exported. The command for creating a table in MYSQL and is named as "employee." The following commands are used to export the table data from "table emp_data of HDFS" to the "employee" table in the MySQL database server.

```
$ sqoop export \
--connect jdbc:mysql://localhost/db \
--username root \
--table employee \
--export-dir /emp/emp_data
```

For verification of output use the following commands:

```
mysql>select * from employee;
```

After that, the below listed output will be shown.

```
+------+-------------+------------+------------------+--------+
| Id   | Name        | Designation | Salary          | Dept   |
+------+-------------+------------+------------------+--------+
| 1201 | gopal       | manager     | 50000           | TP     |
| 1202 | manisha     | preader     | 50000           | TP     |
| 1203 | kalil       | php dev     | 30000           | AC     |
+------+-------------+------------+------------------+--------+
```

Figure 11 illustrates how Sqoop connects MySQL to Hadoop.

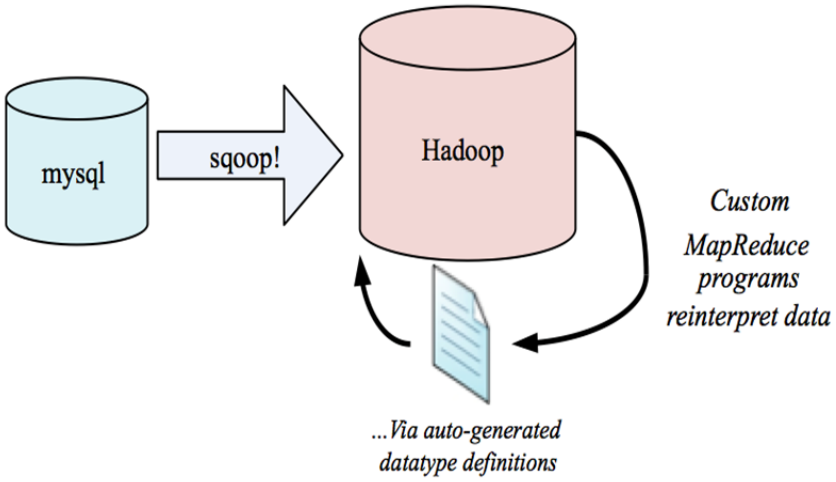


Figure 11. How Sqoop Connects to MySQL to Hadoop (Tutorials Point, n.d.d)

Key features of Sqoop are JDBC-based (Define JDBC?) implementation it works with many popular database vendors. Auto-generation of tedious user-side code it to write MapReduce applications to work with your data, faster. Integration with Hive, allows you to stay in an SQL-based environment through Extensible Backend Database-specific code paths for better performance (Cloudera, 2009).

Important features of Sqoop are:

- Full Load
- Incremental Load
- Parallel import and export
- Import results of SQL query
- Compression
- Connectors for all the key RDBMS Databases

- Kerberos Security integration
- Load data directly into Hive
- Support for Accumulo

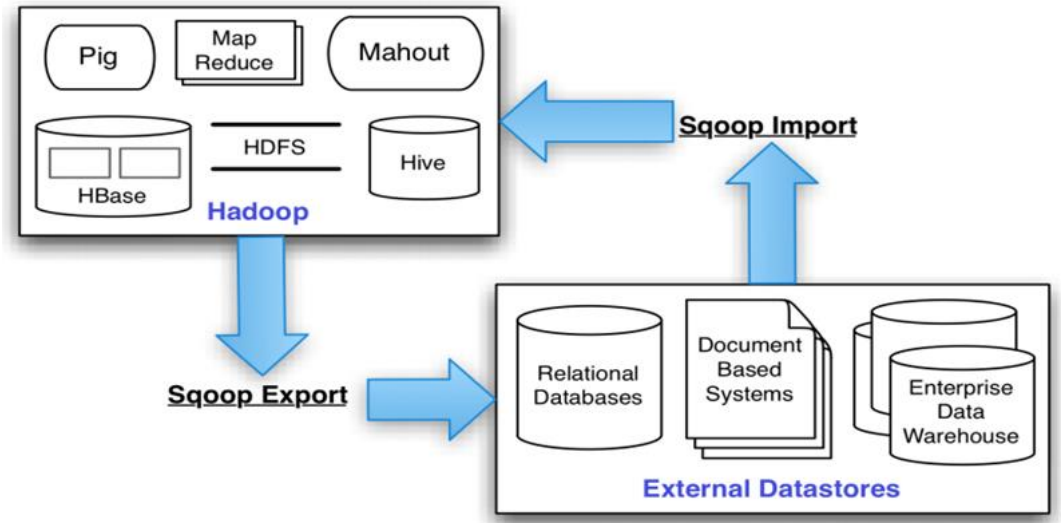


Figure 12. Sqoop Architecture (Tutorials Point, n.d.d)

PIG

Pig is one of the high-level abstractions that work on the top of the MapReduce programming. Pig uses a data flow language called Pig-Latin. Pig runs on the client machine and turns the Pig-Latin programs into Map Reduce (Hadoop, n.d.).

Pig is a high-level scripting language, and it is used with Apache Hadoop. As data flows, Pig excels at describing data analysis problems. In Apache Hadoop, users can do every possible required data manipulation by using Pig. There is a user defined function facility available in Pig and, by using this service, users can invoke code in many languages like JRuby, Python, and Java. Also, conversely the user can execute Pig scripts in many other languages. You can build larger and more complex applications to tackle real business problems by using Pig.

The best example of Pig applications is the ETL transaction mode that describes how any process will take data from a source and after extracting data from the source how to transform it according to the rules set and after transformation of data load it into the data stores.

Using user defined functions, Pig can ingest data from files, streams, and other sources. After receiving the data, it can perform operations with data like selecting, iteration, and other transformations of the data. By using the UDF feature, data can be passed to the more sophisticated algorithms for the transformation. After this, Pig can store all the results in the Hadoop Data File System. MapReduce jobs are taken by the translation of Pig scripts, and these jobs can be run on the Apache Hadoop cluster.

Programmers with the lack of knowledge in Java and other languages used to struggle to work with Hadoop, especially while they were performing any MapReduce task. Apache Pig is a pain reliever for all such programmers.

- Using **Pig Latin,** developers can perform MapReduce tasks efficiently without having to type complex codes in Java.
- Apache Pig uses a multi-query approach, thereby reducing the length of code. For example, an operation that would require you to type 200 lines of code (LoC) in Java would require less than perhaps 10 LoC in Apache Pig. Ultimately, Apache Pig reduces the development time by almost 16 times.
- Pig Latin is a **SQL-like language, (reason for bolding?)** 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. Also, it provides nested data types like tuples, bags, and maps that are missing from MapReduce.

Pig has two major components and two modes in which it can work: A high-level data processing language called Pig Latin that runs your Pig Latin script in a choice of various evaluation mechanisms. The primary evaluation mechanism is Hadoop. Pig also supports a local mode for development purposes. Pig simplifies programming because of the ease of expressing your code in Pig Latin. The compiler helps to exploit optimization opportunities in your script automatically. This frees you from having to tune your program manually. As the Pig compiler improves, your Pig Latin program will also get an automatic speed-up.

At the UNIX command prompt, give the following and you should get a Pig prompt called a grunt after few lines of debugging code (Apache Hive, 2016).

Type quit coming out of local Pig grunt and type pig to enter MapReduce mode.
If you want to run Pig in local mode, you can do it by giving the pig -x local at the UNIX prompt. Utility commands help, quit, kill, jobid, set debug [on|off]

Set job.name ‘jobname’
File commands are (cat, cd, copyFromLocal, copyToLocal, cp, ls, mkdir,mv, pwd, rm, rmf, exec, run) two new commands are exec and run. They run Pig scripts while inside the Grunt shell and can be useful in debugging Pig scripts. The exec command executes a Pig script in a separate space from the Grunt shell. Aliases defined in the script are not visible to the shell and vice versa. The command run executes a Pig script in the same space as Grunt (also known as

interactive mode). It has the same effect as manually typing in each line of the script into the Grunt shell.

Features of Pig

- Set of operators: There are many operators in Pig to perform operations like join, sort, filter, etc.
- Ease of programming: If you are good at SQL, then Pig Latin is also similar to SQL, and afterward it will be very easy to write a Pig script.
- Optimization opportunities: Apache Pig tasks are automatically executed due to their optimization in Apache Pig. So the only care that needs to be taken is care of semantics of the language.
- Extensibility: By using existing operators, users can develop their functions reading the data, processing the data, and writing the data for their ease of use.
- UDF's: Pig provides the facility to its user to create User-defined Functions in other programming languages like Java and embeds them in Pig scripts.
- Handles any data: Apache Pig handles all kinds of data in the structured and unstructured form and stores the results in HDFS.

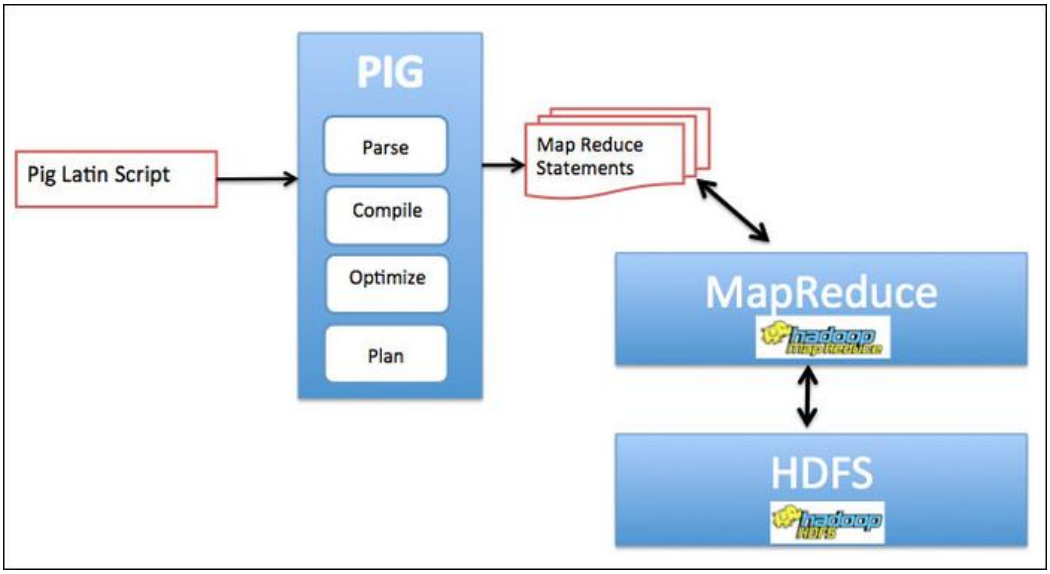


Figure 13. Pig Ecosystem Architecture (Tutorials Point, n.d.a)

Apache Pig vs. MapReduce

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

Apache Pig	MapReduce
It is data flow language.	It is data processing paradigm.
It is high-level language.	It is low level and rigid.
Join operations are very simple to perform in Apache Pig.	Performing a join operation in MapReduce is difficult.
Any new and novice programmer with basic knowledge of SQL can work very quickly with Apache Pig.	Knowledge about Java is a must to work with MapReduce.
It uses the multi-query approach, reducing the length of code to a great extent.	It takes 20 times the number of lines of code to perform the same task.
On execution Apache Pig operator is converted into a MapReduce job internally, there is no need for compilation.	MapReduce jobs have a long process of compilation.

Apache Pig vs. SQL

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

Apache Pig	SQL
It is a procedural language.	It is a declarative language.
Schema is optional, and data can be stored without designing of the schema.	The schema is a must in SQL.
There is nested relational data model used in Apache Pig.	There is flat relational data model is used in SQL.
There is very limited optimization of a query.	More optimization for a query in SQL.

- Pig Latin also declares execution plan.
- Pig Latin also provides operators to perform ETL (Extract, Transform, and Load) functions.

Apache Pig vs. Hive

Both of these languages are used to create MapReduce jobs. And, in some cases, both languages operate in similar ways. The difference between these languages is listed below.

Apache Pig	Hive
Apache Pig uses a language called Pig Latin created at Yahoo.	It uses a language named HiveQL and was created at Facebook.
It is data flow language.	It is query-processing language.
It is procedural language and fits in a pipeline paradigm.	It is declarative language.
It can handle structured, unstructured and semi-structural data.	It can mostly handle structured data.

Applications of Apache Pig

It is mainly used by data scientists to perform ad-hoc processing tasks and quick prototyping. Apache Pig is used to (a) perform processing with weblogs, (b) perform processing with search platforms, and (c) perform processing with time sensitive data.

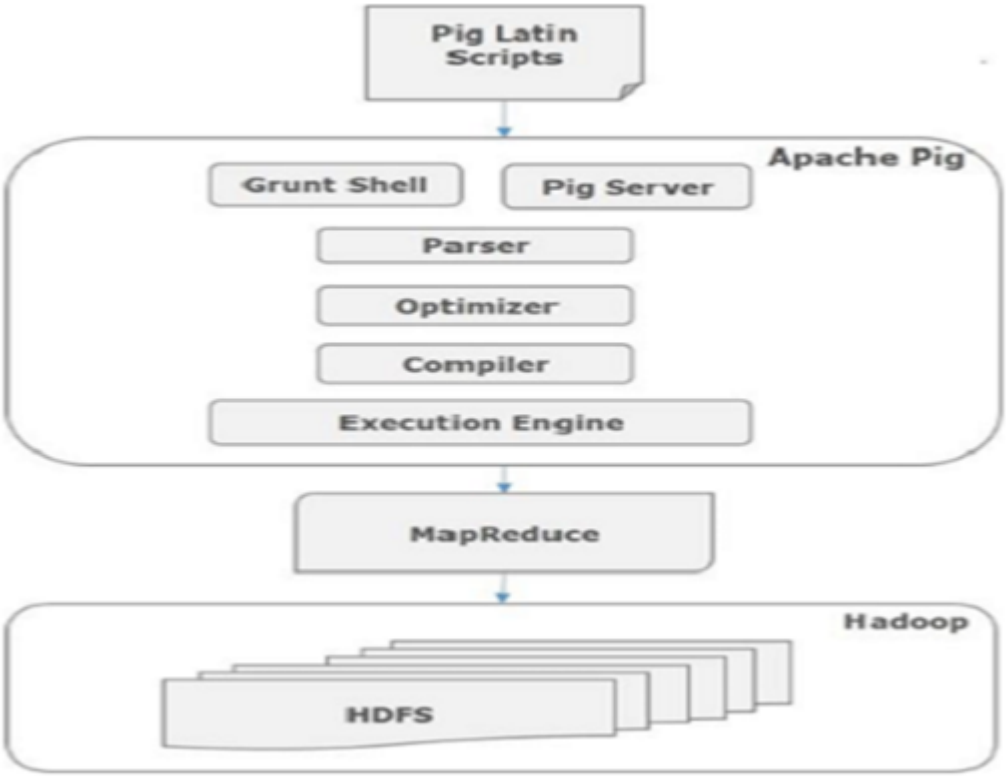


Figure 14. Applications of Pig (Tutorials Point, n.d.a)

Components of Apache Pig

Let us take a look at the major components:

- Parser: In early stages, Apache Pig scripts are managed by a parser. It checks the syntax of the script to detect syntax errors in the script. Then it shows an output that is a directed acyclic graph, which represents the Pig statements and logical operators. In

DAG (define?), logical operators are represented by nodes and data flows are represented by edges.

- Optimizer: After parser, a logical plan is passed to the logical optimizer, which performs the logical optimizations like projections and pushes down to next component.
- Compiler: Then the compiler compiles all the logical plan data into a series of jobs called MapReduce jobs.
- Execution Engine: Finally, these MapReduce jobs are stored in Hadoop in sorted order and after that, these Map Reduced jobs are executed on Hadoop to obtain the desired result.

Pig Latin data model. Figure 15 depicts the diagrammatic model of Pig Latin language.

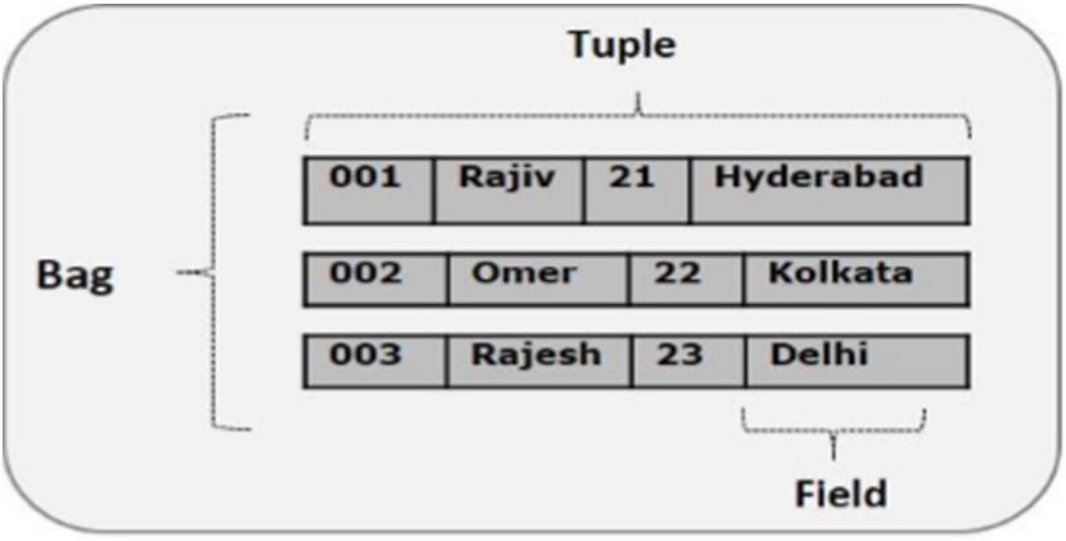


Figure 15. Data Model of Pig (Tutorials Point, n.d.a)

- Atom: Any single value in Pig Latin is called Atom irrespective of data of those values. Their data types are a string and can be stored as a string and used as string and number. Int, float, double, long, are all atomic values of Pig Latin.
- Field: Any simple atomic value or piece of data is called a field.
- Tuple: When ordered set of fields forms any record, it is known as a tuple. A field can be of a single type. A tuple is same as a row in a table of RDBMS.
- Bag: It is an unordered set of tuples. A collection of tuples is called a bag. Each tuple can have many numbers of fields in it. The symbol of a bag is represented as '{}.' It is same as the table in RDBMS.
- Map: It is a set of pairs of key-values. The key should be unique and of type char-array. It is represented by '[]' and it can store any value.
- Relation: It consists a bag of tuples. The relations are unordered in Pig Latin.

Apache Pig execution modes. There are two Apache Pig executive modes: Local Mode and MapReduce Mode.

- Local Mode: In this mode, all the files are installed and run on your local host and local file system. There is no need of Hadoop or HDFS. This method is used for testing purpose.
- MapReduce Mode: MapReduce mode is where we load or process the data that exists in the Hadoop File System (HDFS) using Apache Pig. In this mode, whenever we execute the Pig Latin statements to process the data, a MapReduce job is invoked in the back-end to perform a particular operation on the data that exists in the HDFS.

Execution mechanism. Apache Pig script can be executed in three different ways as listed below.

- Interactive Mode (Grunt Shell): You can run statements of Apache Pig in grunt shell. You enter the statements in grunt shell and get the detailed output.
- Batch Mode (Script): You can run Apache Pig in Batch mode but writing the script of Apache Pig in a single file with .pig extension.
- Embedded Mode (UDF): Defines user's functions in programming languages like Java and their use in the script.

Shell commands. The Grunt shell of Apache Pig is used to write a Pig Latin script and the user can invoke any shell commands using **sh** and **fs.**

Sh Command: Using this sh command, the user can invoke any shell command from the grunt shell but using this command the user cannot execute the commands from the Grunt shell.

Syntax: Following is the syntax of the sh command.

```
Grunt>sh shell command parameters
```

To list out the files present in Bin of Pig user can use **Is** command.

```
grunt> sh ls
 pig
pig_1444799121955.log
pig.cmd
pig.py
```

fs Command: Using this fs command, the user can invoke any fs shell commands from Grunt shell.

Syntax:

```
grunt> sh File System command parameters
```

Invoking of ls command of HDFS from the Grunt shell using Fs command.

```
grunt> fs –ls
Found 3 items
drwxrwxrwx   - Hadoop supergroup          0 2015-09-08 14:13 Hbase
drwxr-xr-x   - Hadoop supergroup         0 2015-09-09 14:52 seqgen_data
drwxr-xr-x   - Hadoop supergroup         0 2015-09-08 11:30 twitter_data
```

Help Command: This command gives a list of Pig commands or properties of Pig to its users. How to use this command follows.

```
grunt> help

Commands: <pig latin statement>; - See the PigLatin manual for details:
http://hadoop.apache.org/pig

File system commands: fs <fs arguments> - Equivalent to Hadoop dfs  command:
http://hadoop.apache.org/common/docs/current/hdfs_shell.html


Diagnostic Commands: describe <alias>[::<alias] - Show the schema for the alias.
Inner aliases can be described as A:: B.
   explain [-script <pigscript>] [-out <path>] [-brief] [-dot|-xml]
     [-param <param_name>=<pCram_value>]
     [-param_file <file_name>] [<alias>] -
     Show the execution plan to compute the alias or for entire script.
     -script - Explain the entire script.
     -out - Store the output into directory rather than print to stdout.
     -brief - Do not expand nested plans (presenting a smaller graph for overview).
     -dot - Generate the output in .dot format. Default is text format.
```

```
    -xml - Generate the output in .xml format. Default is text format.
    -param <param_name - See parameter substitution for details.
    -param_file <file_name> - See parameter substitution for details.
    alias - Alias to explain.
    dump <alias> - Compute the alias and writes the results to stdout.

Utility Commands: exec [-param <param_name>=param_value] [-param_file <file_name>] <script>
-
    Execute the script with access to grunt environment including aliases.
    -param <param_name - See parameter substitution for details.
    -param_file <file_name> - See parameter substitution for details.
    script - Script to be executed.
  run [-param <param_name>=param_value] [-param_file <file_name>] <script> -
    Execute the script with access to grunt environment.
                -param <param_name - See parameter substitution for details.
    -param_file <file_name> - See parameter substitution for details.
    script - Script to be executed.
  sh  <shell command> - Invoke a shell command.
  kill <job_id> - Kill the hadoop job specified by the hadoop job id.
  set <key> <value> - Provide execution parameters to Pig. Keys and values are case sensitive.
    The following keys are supported:
    default_parallel - Script-level reduce parallelism. Basic input size heuristics used
    by default.
    debug - Set debug on or off. Default is off.
    job.name - Single-quoted name for jobs. Default is PigLatin:<script name>
    job.priority - Priority for jobs. Values: very_low, low, normal, high,
very_high.
    The default is normal stream.skippath - String that contains the path.
    This is used by streaming any hadoop property.
  help - Display this message.
  history [-n] - Display the list statements in cache.
    -n Hide line numbers.
  quit - Quit the grunt shell.
```

History Command: This command displays a list of statements executed so far from the Grunt Shell. Following are three statements that are executed so far.

```
grunt> customers = LOAD 'hdfs://localhost:9000/pig_data/customers.txt' USING PigStorage (',');

grunt> orders = LOAD 'hdfs://localhost:9000/pig_data/orders.txt' USING PigStorage(',');
grunt> student = LOAD 'hdfs://localhost:9000/pig_data/student.txt' USING PigStorage(',');
```

Then the **history** command will show the output.

Set Command: The set command is used to display the assign values to keys used in Pig.

Usage:

Key	Description and Values
Default_parallel	Pass any whole number as a value to this key to set the number of reducers for map job.
debug	You can pass on or off to this key to turn on or turn off debugging feature in Pig.
Job.name	You can pass string value to this key to set the job name.
Job.priority	You can pass the following values to this key to set the job priority • Very low • Low • Normal • High • Very high
Stream.skippath	You can pass string value to this key to set the path where the data is not to be moved and stored.

Quit Command: You can stop from Grunt shell by this command as shown below.

```
grunt> quit
```

Data Types of Pig Latin

S.N.	Data Type	Description
1	int	Represent 32-bit integer, e.g. 10
2	long	64-bit integer, e.g. 4L
3	float	32-bit floating point, e.g. 5.7f
4	double	64-bit floating point, e.g. 10.5
5	Char array	Array of characters in Unicode, e.g. 'tutorial point.'
6	Byte array	Byte array
7	Boolean	Boolean value, e.g. true/false
8	Date time	Date time, e.g. 2016-08-19T00:00:00.000+00:00
9	Big integer	Java big integer
10	Big decimal	Java big decimal, e.g. 290.345567787887

Complex Types

1	Tuple	It is an ordered set of fields, e.g. (jack, 10)
2	Bag	It a collection of tuples, e.g. {(jack, 10), (bill, 20)}
3	Map	It is set of pairs of key values, e.g. ['name'#'jack','age'#'24']

Arithmetic Operators of Pig Latin

The following listed below are the arithmetic operators of Pig Latin. We have two integer values a=5 and b=10.

operator	description	Example
+	Addition : For addition of values	a+b=15
-	Subtraction: For subtraction of values	a-b= -5
*	Multiplication: Multiplies the values	a*b=50
/	Division: Divides the values	b/a=2
%	Modulus: Remainder of the values	b%a=0
?:	Bincond: Evaluates the boolean operators. Variable x=? Value 1 if true:value 2 if false	b=(a= =1)? 5 :10;

		If a=1 the value of b is 5. If a!=1 the value of b is 10.
Case When Then Else, End	Case: it equals to the nested bincode operator.	Case f1 % 2 When 0 then 'even.' When one then 'odd.' End

Comparison Operators of Pig Latin

Operator	Description	Example
= =	Equal: checks the equality of values.	a=b
! =	Not Equal: Checks the values are equal or not.	a!=b
>	Greater than: Checks if one value is greater than other.	a>b
<	Less Than: Checks if one value is less than other value.	a<b
>=	Greater than or equal to: checks if one value is greater than or equal to other value	a>=b
<=	Less than or equal to: Checks if one value is less than or equals to other value.	a<=b
matches	Pattern Match: checks the one value match with other or not	F1 matches'.*f2.*'

Construction Operators

Operator	Description	Example
( )	Tuple constructor operator: used for construction of a tuple.	(jack, 10)
{}	Bag constructor operator: used for construction of a bag.	{(jack, 10),(bill, 20)}
[ ]	Map constructor operator: used to construct tuple	[name#jack,age#25]

HIVE

Hive is a data warehouse software which helps to query and manage large data sets. In other words, HIVE is also known as SQL for Hadoop. Hive provides ETL functionality to

Hadoop. In Hive query execution is done through Map Reduce. Hive provides SQL query type language called Hive Query Language (HQL). Hive is designed for easy data summarization and ad hoc querying. We can analyze large data sets using HIVE (Jain, 2012).

Hive is used for Log Processing, Text Mining, Document Indexing, Predictive Modeling, and Hypothesis Testing.

Initially, Facebook developed Hive and later the Apache Software Foundation took it over and made it an open-source product under the name Apache Hive. Now different companies like Amazon and their Elastic MapReduce use it.

Hive is not a relational d

Features of Hive

- It is used to store schema in a database and then it's processing into HDFS.
- It is designed for OLAP.
- It provides SQL-type language called HiveQL or HQL.
- It is fast, scalable and extensible.

Hive Architecture

- Metastore: Stores system catalog.
- Driver: Manages lifecycle of HQL
- Query Compiler: Compiles HQL into DAG of Map Reduce.
- Execution Engine: The component executes tasks in proper dependencies.
- Hive Server: Provides Thrift Interface and JDBC/ODBC for integrating other applications.
- Client Component: CLI, Web Interface, jdbc/odbc Interface.

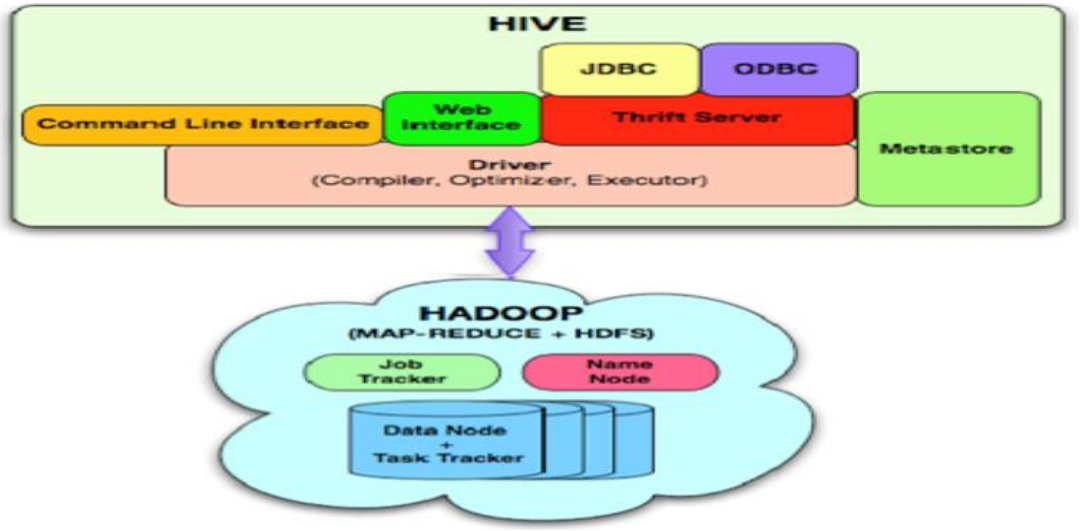


Figure 16. Hive Architecture (Tutorials Point, n.d.c)

Following is a description of each unit of the diagram.

Unit Name	Operation
User interface	Hive is data warehouse infrastructure software that can create interaction between the 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 respect ive 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 a 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 is the data storage techniques to store data into file system.

Working of Hive

Figure 17 shows the working between Hive and Hadoop.

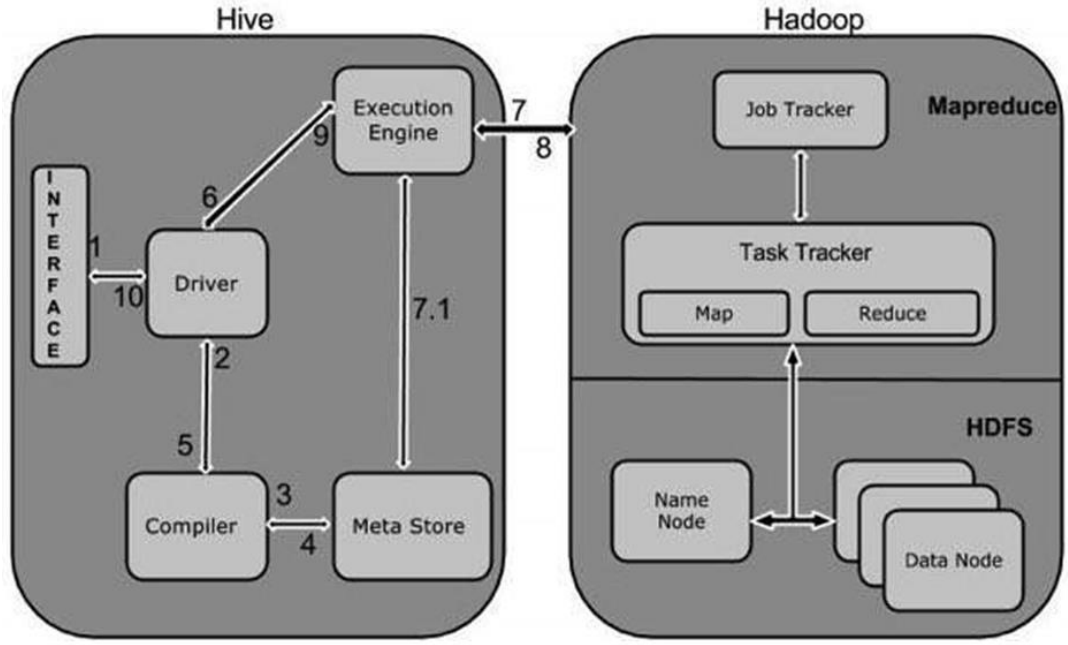


Figure 17. Hive Working (Tutorials Point, n.d.c)

The following defines the interaction between Hive and Hadoop.

Step No.	Operation
1	**Execute Query.** Driver receives query from command line to execute (any database driver like JDBC, ODBC).
2	**Get Plan.** Then driver takes some help from parser to check the query for syntax errors.
3	**Get Metadata.** Then compiler sends metadata request to Metastore that is any database.
4	**Send Metadata.** Then Metastore sends the metadata according to the request from compiler.
5	**Send Plan.** The compiler checks for the requirement and resends the plan to the driver. Up to here the parsing and compiling of a query are complete.
6	**Execute Plan.** Then driver sends the plan to execute to the execution engine.
7	**Execute Job.** MapReduce job will execute the process; the execution engine sends the job to JobTracker, which is in Name node and it assigns this to TaskTracker, which presents in Data node, and here the query is executed.

8	**Fetch Result.** Execution engine receives the results from Data nodes.
9	**Send Results.** Then the result is further sends to the driver.
10	**Send Result.** The driver sends the result to the Hive Interface.

Hive vs. RDBMS

Hive	RDBMS
It enforces schema on read time.	It enforces schema on write time.
Hive does not verify the data when it is loaded but when it is retrieved.	It verifies the data at load time.
It is based on Write once, Read many times	It is based on Read and Writes many times.
Petabytes of data can be stored very easily.	The maximum size of data can be of Terabytes.

Oozie Workflow

Oozie executes and monitors workflow in Hadoop, periodic scheduling of workflow, and Sqoop Trigger execution by data availability. It has both HTTP and command line interface and Web console. Unlike Job Control, which runs on the client machine submitting the jobs, Oozie runs as a server, and a client sends a workflow to the server. In Oozie, a workflow is a DAG (definition?) of action nodes and flow control nodes. An action node performs a workflow task, like moving files in HDFS, running a MapReduce job or running a Pig job. A control-flow node governs the workflow execution between actions by allowing such constructs as conditional logic (so different execution branches may be followed depending on the result of a previous action node) or parallel execution (Tutorials Point, n.d.b).

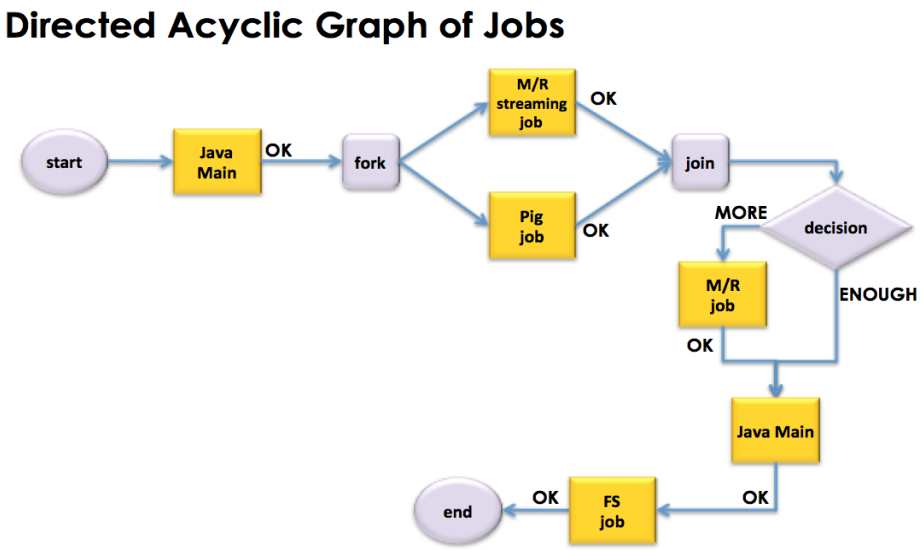


Figure 18. Directed Acyclic Graph of Jobs

When the workflow completes, Oozie can make an HTTP callback to the client to inform it of the workflow status. It is also possible to receive callbacks every time the workflow enters or exits an action node. Oozie allows failed workflows to be re-run from an arbitrary point. This is useful for dealing with transient errors when the early actions in the workflow are time-consuming to execute.

Conclusion

According to Chapter II, the discussion based on background theory apparently concluded that Hadoop is good when compared to the other distributed systems and other processing mechanisms through its remote handling, reducing seek times and combining two datasets with key-value pairs.

Chapter III: Methodology

How it Works

To complete this project, we need to have a single node Hadoop cluster on the local machine. Below are the steps to install the environment on a local machine. To get the environment onto a local machine, we can install Hadoop separately, or we can install one of the vendor's Hadoop distributions available in the market, an example is Hotornworks HDP or Cloudera CDH. If we are installing Hadoop separately, then we need to install required ecosystems independently or else we can go through with these platforms either HDP or CDH where we have everything readily installed.

Figure 19. Hotornworks Data Platform HDP

Like any other system, clusters of Hadoop will also get changed as the time passes and the user should know about the techniques to make the Hadoop keep it in a good state for future use. Hadoop uses different parameters, and their default values move to target running in separate standalone mode without causing any errors on systems their default values easily run on a majority of systems. The default setting for Hadoop's directories is dependent upon the user name. The user should avoid using any property that is dependent upon any user name. Because there can be a clash occurrence due to mismatching of user names provided and actual user name. The user should set something that is entirely independent of username.

Conclusion

From methodologies of installing a Hadoop cluster, the user should provide the platform to Hadoop on which it will run and perform its tasks. The system should be compatible with the requirements of Hadoop as set forth in this document.

Chapter IV: System Requirements

Functional Requirements

- FR1 – A Hadoop cluster setup with all recommended installations
- FR2 – A client machine setup
- FR3 – An installation of Pig, Hive, and Sqoop
- FR4 – At least Cloud era or Hortonworks VM would satisfy as a minimum requirement

Non-Functional Requirements

- NFR1 – Make proper configurations between the client and cluster
- NFR2 – Make sure the data-node space is available for three replicas
- NFR3 – Test for the best results

With the constant increase in data volumes and varieties, especially from social media and the Internet of Things (IoT) Hadoop can cope with these scenarios, so it is critical advantage. The computing power of Hadoop is very fast. The more nodes that are used the more processing power it will provide to the nodes.

If any hardware failure occurs, then data and applications will remain safe and protected by Hadoop. If one node goes down, another node will automatically take the responsibilities of that node and perform operations instead of the node that fails. These nodes automatically redirected to the nodes to make sure communication is not broken, and distributed computing does not fail. There is not only one original copy of data that is saved, but there are multiple copies of data collected in Hadoop.

Unlike other traditional relational databases, you do not have to pre-process the data before its storage. You can store unlimited data as you want and later decide about the processing of data. The data stored is unstructured like videos, images, and text.
Hadoop has an open-source framework and is totally free of cost for the software and uses hardware to store massive sets of data.

Little administration is required while growing your system to handle more data by adding more nodes to your distributed computing system. Cloud infrastructure is a great option and potentially the best area to run and use Hadoop because it allows its users to easily scale to hundreds of nodes and make it financially flexible to avoid heavy investment costs.

Conclusion

From the requirements of Hadoop, user systems should be compatible with Hadoop's requirements for installation. Users should install Pig, Hive, and Sqoop to run and use all the features of Hadoop.

Chapter V: Work in Progress

Generating PIG scripts, HIVE scripts, SQOOP commands including screen shots, and description.

Raw Data

The statement produced by a bank will be of two types an e-statement and paper statements. As per the project, the e-statement is considered. Figure 20 is an example of raw statements generated by a bank.

```
D3A8ADabcd 40052568999000000001 H
D3A8ADabcd 4005256899900000002 E
D3A8ADabcd 4005256899900000003 H
D3A8ADabcd 4005256899900000004 E
D3A8ADabcd 4005256899900000005 H
D3A8ADabcd 4005256899900000006 E
D3A8ADabcd 4005256899900000007 H
D3A8ADabcd 4005256899900000008 E
D3A8ADabcd 4005256899900000009 H
```

Figure 20. Diagram for Bank's Raw Data

As far as the diagram of sample raw data illustrates,

D3A8ADabcd - transaction number (field-1)

400525.....1 - account number (field-2)

E – e-statement copy (field-3)

H – Hard copy (field-3)

As per the project, the e-statements are the only files required for confirmation file.

Note: The data is copied at **data** directory in the **usecase1 of root@192.168.0.125**

Confirmation File

The Confirmation file should contain a record of e-statements only. Extracting the data for 5 to 10 million makes the other system processing slow. However, Hadoop makes the processing fast by selecting **PIG** (a scripting language) for filtering such records within no time.

Pig

- Pig is a scripting language named as pig-Latin.
- Pig works well as a MapReduce and runs parallel on Hadoop.
- Pig has standard operations like filter, group, joins . . . which makes programmer easy to write when compared to Java.

Why Pig chose for Confirmation File

Pig is chosen, because of its easy scripting and maintenance. The current project makes the usage of Pig a good choice, by filtering the records according to the requirement.

Confirmation File Output

Figure 21 shows the confirmation file which contains the account numbers for e-statements only.

Figure 21. Screen Shot for Confirmation File

NOTE: After collecting all reducer outputs, make them merge into one file and load them into the Hive table as stated in the following section: NIT, SUPPLEMENT, and CYCLE files.

NIT, SUPPLEMENT, and CYCLE Files

The Nit files contain cardholder details and account numbers. Supplement files contain account numbers, record type and legal identity and cycle file contains account numbers, product type, language preference, state code, bill type and event time-stamp as fields. It is very easy to generate the files with different fields using **HIVE**.

Hive

1. Hive is a Query language and easy to load data into a Hive table.
2. Hive works well as a MapReduce and runs parallel in Hadoop.
3. Hive is good for joining.

Why Hive

Hive is used to join two or more tables easily with a simple query. It is more useful for joining more tables to achieve the CNE table.

NIT, Supplement, and Cycle File Data

The Nit files contain cardholder details and account numbers as shown in Figure 22.

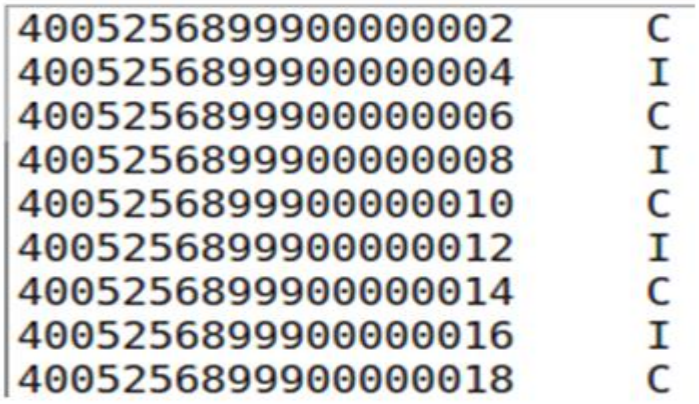


Figure 22. Screen Shot for Nit Input File

Supplement files contain account numbers, record type, and legal identity as shown in Figure 23.

Figure 23. Screen Shot for Supplement Input File

Cycle file contains account numbers, product type, language preference, state code, bill type and event time-stamp as fields as shown in Figure 24.

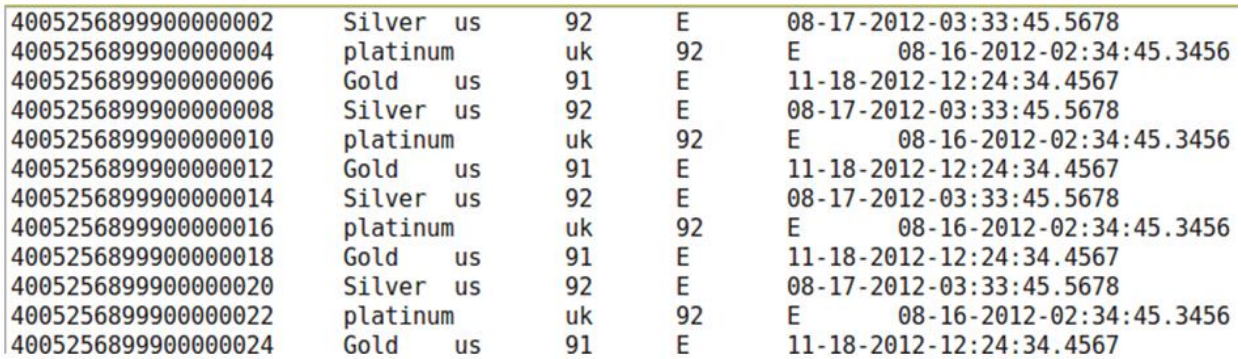


Figure 24. Screen Shot for Input Cycle File

Output for NIT, Supplement and Cycle File

After loading into table, Figure 25 shows the desire fields of **NIT file** are loaded correctly into the Hive table.

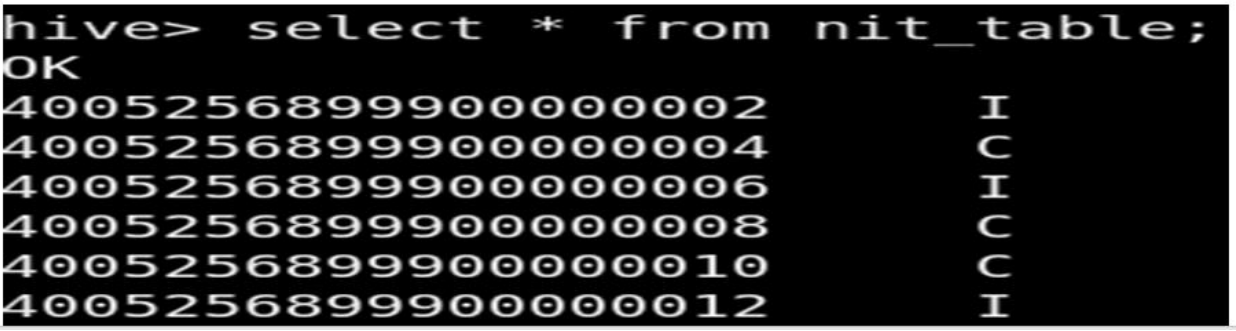


Figure 25. Screen Shot for NIT File

After loading into table, Figure 26 shows the expected fields of **supplement file** are loaded correctly into the Hive table.

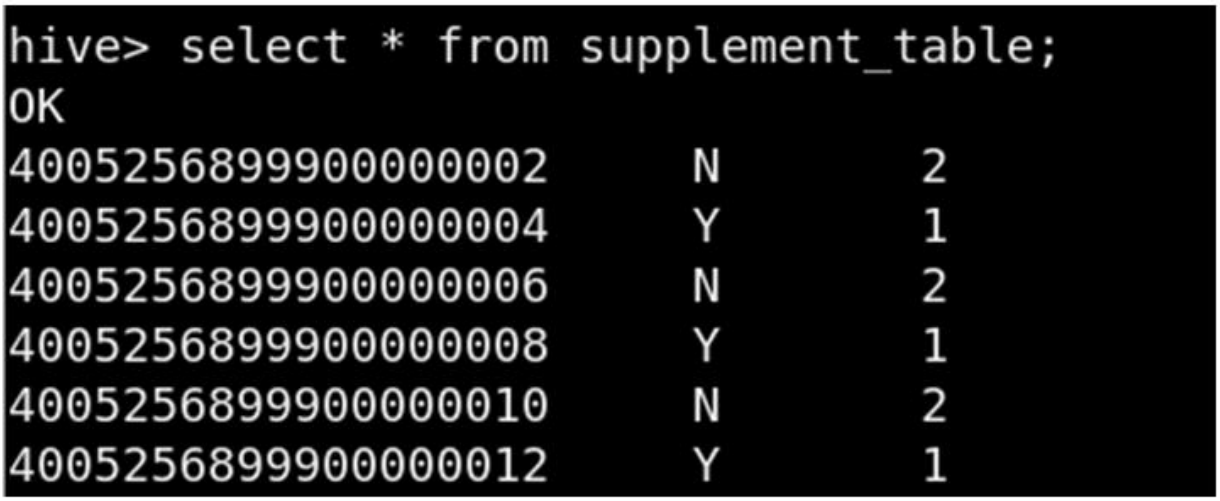


Figure 26. Screen Shot for Supplement File

After loading into table, Figure 27 shows the desired fields of **Cycle file** are loaded correctly into Hive table.

```
hive> select * from cycle_table;
OK
4005256899900000002     SILVER  91      E       Uk      05-26-2012-09:45:34.6789
4005256899900000004     PLATINUM        92      E       Uk      07-19-2012-07:43:34.5678
4005256899900000006     GOLD    91      E       uk      10-18-2012-11:25:34.3456
4005256899900000008     SILVER  91      E       Uk      05-26-2012-09:45:34.6789
4005256899900000010     PLATINUM        92      E       Uk      07-19-2012-07:43:34.5678
4005256899900000012     GOLD    91      E       uk      10-18-2012-11:25:34.3456
```

Figure 27. Screen Shot for Cycle File

WCC File and GAI File

WCC file contains account numbers, first name, and last name details. GAI file contains account numbers, global ID, party ID, e-mail, and last-login as fields. But, these fields are in sql-db and this is the time for using Sqoop import for getting the data to HDFS.

Sqoop

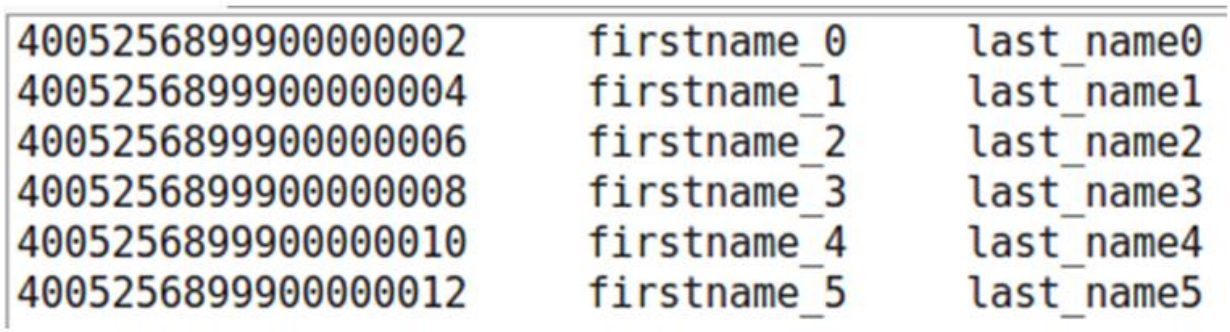
1. Sqoop can import and export data from sql-db to HDFS and vice versa.
2. Sqoop can import data from sql-db to Hive table.
3. Sqoop is a map only process.

Why Sqoop

Sqoop can import data directly to Hive / H-base or HDFS directly without reducing. Because it is known for map only process.

Input data After Importing From sql-db

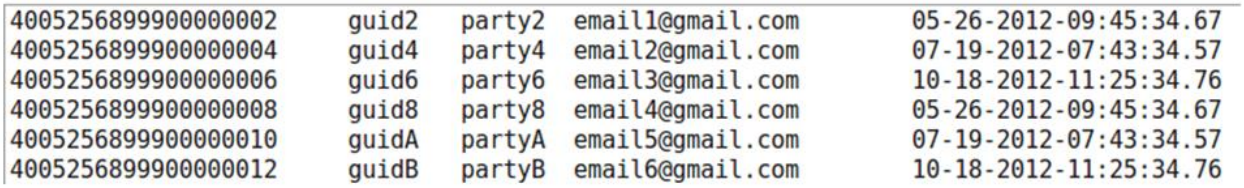
WCC file contains account numbers, first name and last name details as shown in Figure 28.



40052568999000000002	firstname_0	last_name0
40052568999000000004	firstname_1	last_name1
40052568999000000006	firstname_2	last_name2
40052568999000000008	firstname_3	last_name3
40052568999000000010	firstname_4	last_name4
40052568999000000012	firstname_5	last_name5

Figure 28. Screen Shot for WCC Input File

GAI file contains account numbers, global ID, party ID, e-mail, and last-login as fields as shown in Figure 29.



4005256899900000002	guid2	party2	email1@gmail.com	05-26-2012-09:45:34.67
4005256899900000004	guid4	party4	email2@gmail.com	07-19-2012-07:43:34.57
4005256899900000006	guid6	party6	email3@gmail.com	10-18-2012-11:25:34.76
4005256899900000008	guid8	party8	email4@gmail.com	05-26-2012-09:45:34.67
4005256899900000010	guidA	partyA	email5@gmail.com	07-19-2012-07:43:34.57
4005256899900000012	guidB	partyB	email6@gmail.com	10-18-2012-11:25:34.76

Figure 29. Screen Shot for GAI Input File

NOTE: After importing all WCC and GAI files from sql-db, make all the outputs into one WCC and one GAI file and load them into the Hive table as stated in Figure 22.

Output File for WCC and GAI

After loading into table, Figure 30 shows the desired fields of **WCC file** are loaded correctly into hive table.

```
hive> select * from wcc_table;
OK
4005256899900000002     firstname_0     last_name0
4005256899900000004     firstname_1     last_name1
4005256899900000006     firstname_2     last_name2
4005256899900000008     firstname_3     last_name3
4005256899900000010     firstname_4     last_name4
4005256899900000012     firstname_5     last_name5
```

Figure 30. Screen Shot for WCC File

After loading into table, Figure 31 shows the desired fields of **GAI file** are loaded correctly into hive table.

```
hive> select * from gai_table;
OK
4005256899900000002     guid2   party2  email1@gmail.com        05-26-2012-09:45:34.67
4005256899900000004     guid4   party4  email2@gmail.com        07-19-2012-07:43:34.57
4005256899900000006     guid6   party6  email3@gmail.com        10-18-2012-11:25:34.76
4005256899900000008     guid8   party8  email4@gmail.com        05-26-2012-09:45:34.67
4005256899900000010     guidA   partyA  email5@gmail.com        07-19-2012-07:43:34.57
4005256899900000012     guidB   partyB  email6@gmail.com        10-18-2012-11:25:34.76
```

Figure 31. Screen Shot for GAI File

Hive Join

Now, the final output required is the CNE file holding the data of all the files. The required query is Hive join for all Hive tables generated. Before the Hive join, make sure all the files are stored in Hive tables.

Simple Query for Hive Joining Two Or More Tables

```
SELECT a.coloumn, b.coloumn, c.column FROM a JOIN b ON
        (a.key = b.key1) JOIN c ON (c.key = b.key2)
```

Hive Join for All the Files in Hive Tables

Note: The joins are works on the reducer side. So it is tough to manage all the records at the reducer side at once. So the solution for this problem is by doing the joining the two records at a time and finally joining all the splits at the end to achieve the goal.

CNE File

Figure 32 describes the CNE file which is filled with files of Confirmation, GAI, WCC, NIT, supplement and cycle. The file with the default fields is also mentioned in the Chapter VI.

Figure 32. Screen Shot for CNE File Contains All the File Information

Conclusion

Hadoop was elected as the solution for this project and it processed the biggest file of 5 million records in the best way which other systems cannot perform. So the generated CNE file (contains the e-statement) can be sent to the customer through e-mail.

Chapter VI: Test Case Scenarios and Experiences

Test Case Scenario for Ten Records Among Five Million

The Hive Join Query generated for 10 records is the simplest to map and reduce the records. Hence, the CNE file generated in Figure 33 explains the results.

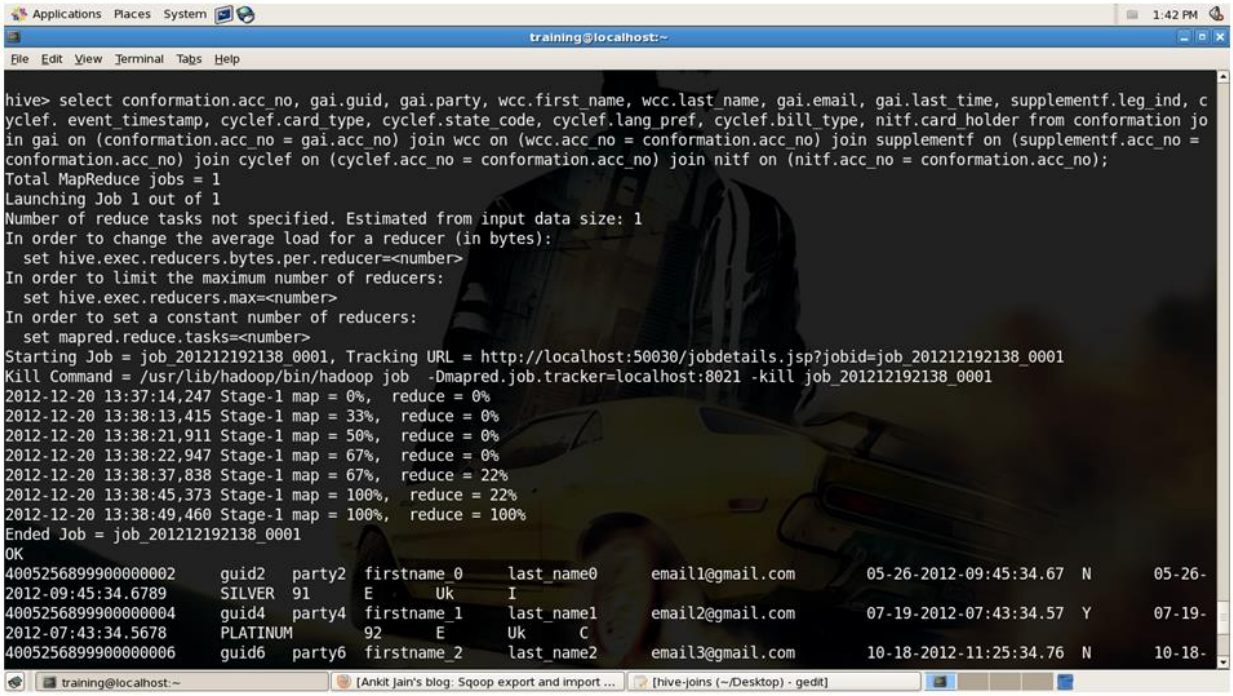


Figure 33. Screen Shot for 10 Records of CNE File

Test Case Scenario for 5,000 Records Among Five Million

The Hive Join Query generated for 5,000 records is sometimes taken process, but yet it produced the right results. Hence, the CNE file generated in Figure 34 explains the results.

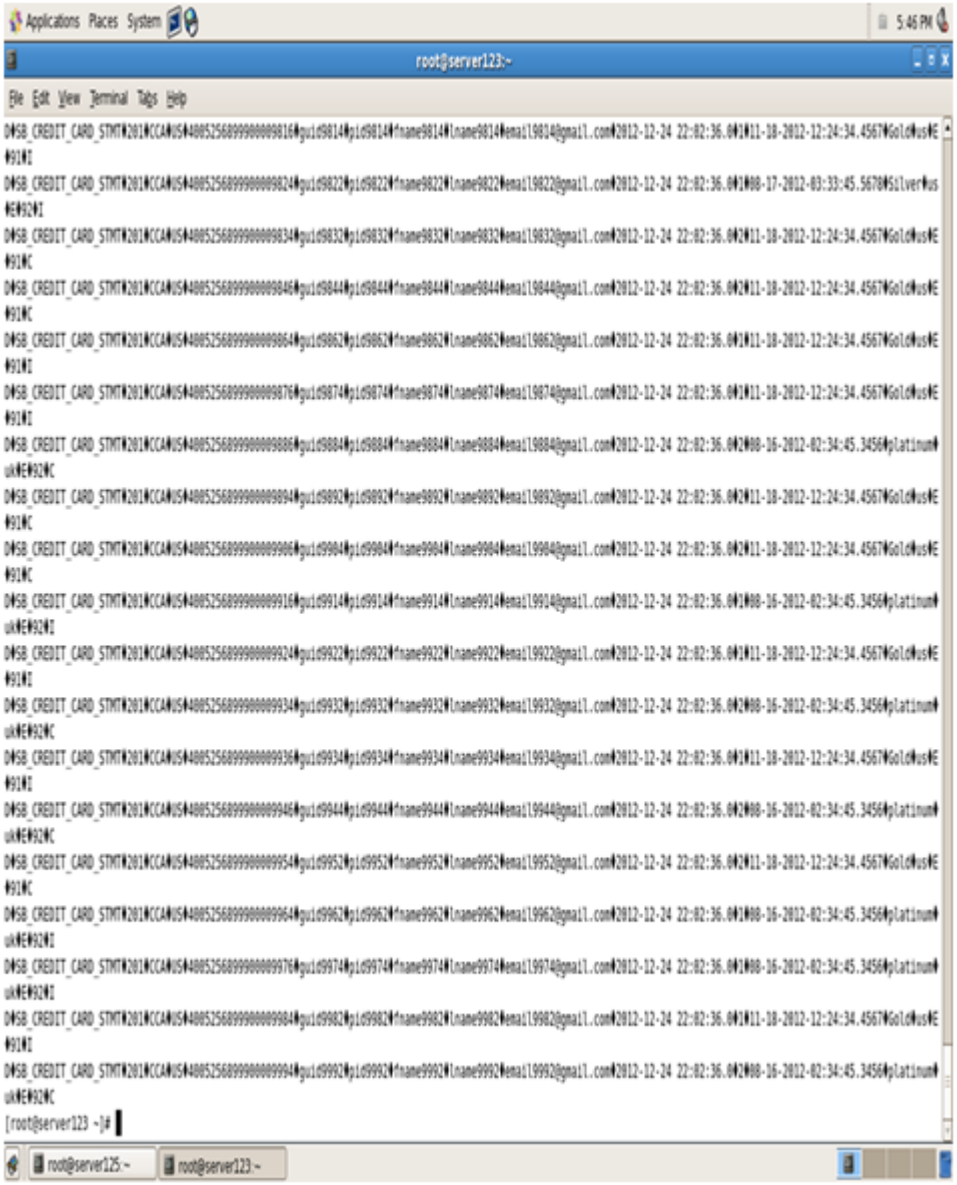


Figure 34. Screen Shot for 5,000 Records of CNE File

Test Case Scenario for Five Million Records of CNE File

The Hive Join Query generated for five million records may take time for showing surprising results. Hence, the CNE file generated in Figure 35 explains the results, which are useful for transferring to all the customers. Now it is copied it into **HDFS.**

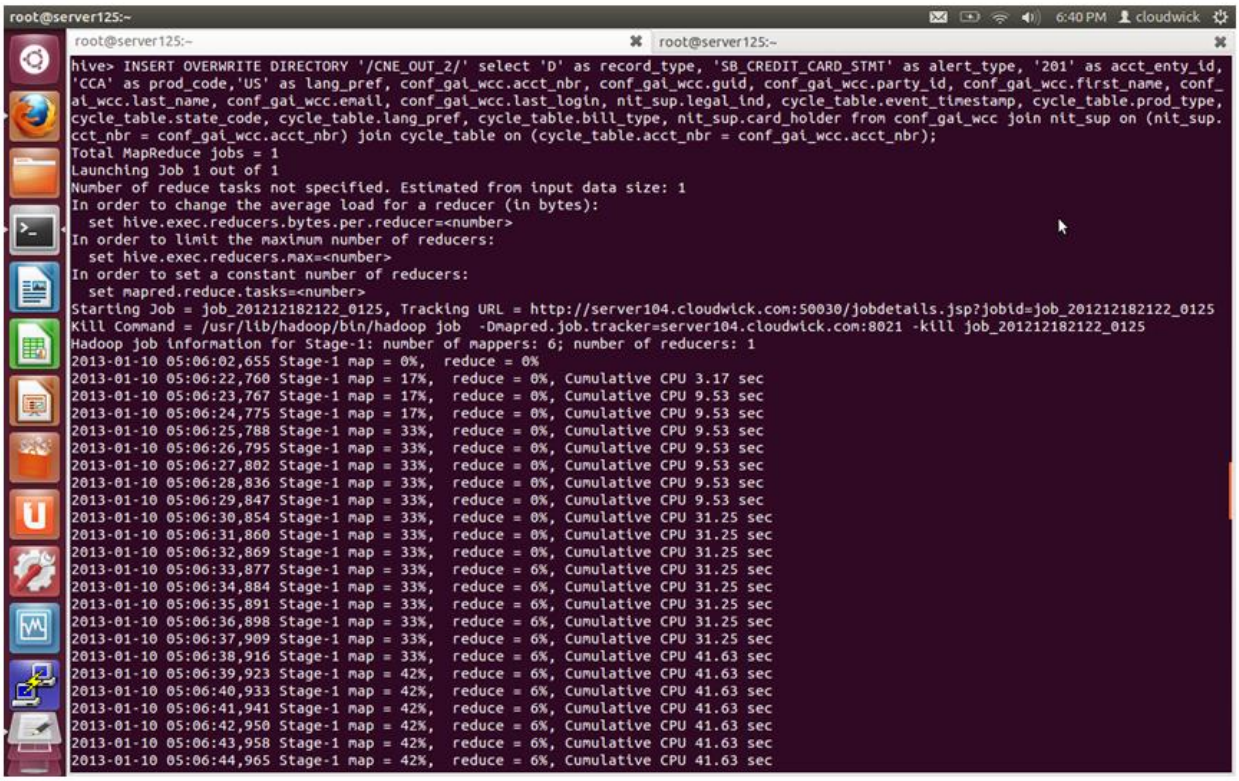


Figure 35. Screen Shot for Five Million Records of CNE File Processed in Hadoop and Stored in HDFS

Experiences

1. Make sure that the disk space is available for storing five million records.
2. When Sqoop import is done, make all the reducer outputs into one (using hadoop fs -get merge) reducer and then load into Hive.
3. While joining all the Hive tables, make sure all the tables has the standard field with common name.
4. Nulls may be raised if the fields in a file are not delimited properly.

5. Once the data is ready, run the sample data first which makes the programmer easy to identify the condition and later it is easy to make the decision of increasing the reducers or stay with it.

Chapter VII: Future Work

In this era, Hadoop is part of many companies that are dealing with the term big data. They have a huge set of data regarding their business, and all of their business processes are entirely dependent upon that data. To manage a large data set like these companies require, the use of Hadoop provides its functionality because of its features to manage large data by splitting them into small divided parts with the help of MapReduce. The main thing is that, at this time, Hadoop is not very well known over the majority of companies. However, in coming years, I think Hadoop will take over all those businesses that are dealing with big data. Many businesses are using it and in the future, many more will take advantage of Hadoop to meet their needs. In the future, there may be some changes, which will happen in the infrastructure, and the architecture of Hadoop due to market demand and advances as time passes. So there are more possibilities that Hadoop will also get some changes in requirements, and it will get more advancement in its features to remain in the big data industry for its survival. Google introduced the concept of Hadoop on the issues raised due to the management of large data, and more possibilities are that again Google will enhance its features in future or some other companies like Amazon or Yahoo will take this step of its advancement. Undoubtedly, in the next iteration of Hadoop, it will perform similarly to the present system. Many experts of Hadoop say that it will change due to changes in industry information and technology. Every day new systems are coming into the market with more numerous and enhanced features as compared to the old existing one in the market. Their requirements are also different from existing ones, so in many future systems whish come into the industry with some changes in their needs regarding the functionality of Hadoop. If Hadoop wants to stay a leader in the industry, then Hadoop must

change according to the needs and feature requirements of new coming systems in the industry. Many small companies are now expanding their business due to the need of the community, and they are also shifting their data towards Hadoop for better security and privacy of their critical data. Now, at this time, Hadoop will serve them well, but maybe not in the future. Perhaps they all want something different from Hadoop as compared to the existing capabilities in Hadoop.

References

Apache Hive. (2016). *Language manual joins: Hive joins*. Retrieved from https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Joins (4)

Apache Software Foundation. (h.d.). *Sqoop user guide (v1.4.6).* Retrieved from http://sqoop.apache.org/docs/1.4.1incubating/SqoopUserGuide.html (5)

Cloudera. (2009). *Introduction to hive*. Retrieved from https://blog.cloudera.com/wp-content/uploads/2010/01/6-IntroToHive.pdf (3)

Hadoop. (n.d.) *Pig Latin reference manual 2*. Retrieved from http://pig.apache.org/docs/r0.7.0/piglatin_ref2.html (2)

Jain, A. (2012, January 10). *Sqoop export and import commands*. Retrieved from http://ankitasblogger.blogspot.in/2012/01/sqoop-export-and-import-commands.html (5)

Tutorials Point. (n.d.a). Apache pig tutorial. Retrieved from https://www.tutorialspoint.com//apache_pig/index.htm (6)

Tutorials Point. (n.d.b). *Hadoop tutorial*. Retrieved from http://www.tutorialspoint.com//hadoop/index.htm (6)

Tutorials Point (n.d.c). *Hive tutorial*. Retrieved from http://www.tutorialspoint.com//hive/index.htm (6)

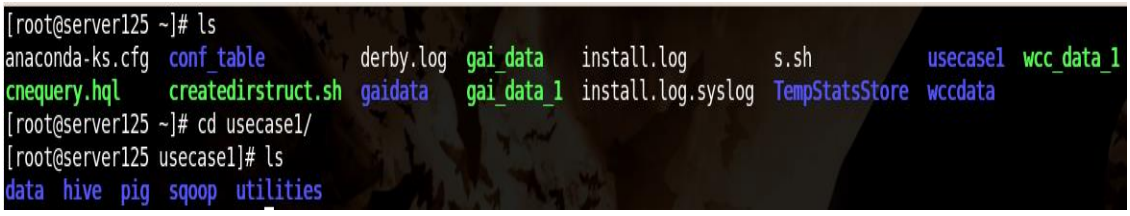
Tutorials Point. (n.d.d). *Sqoop tutorial*. Retrieved from http://www.tutorialspoint.com//sqoop/index.htm

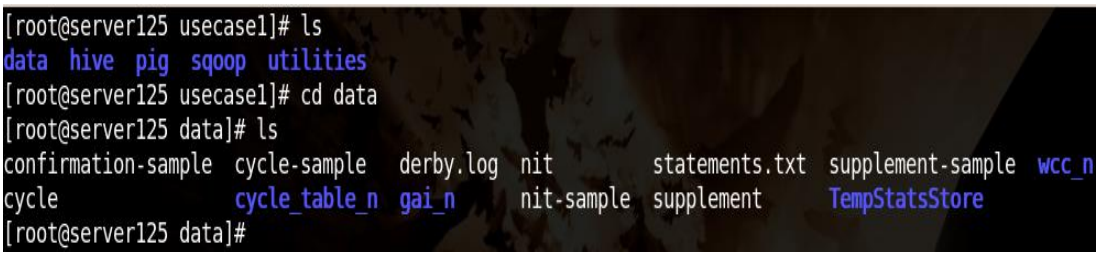
White, T. (2012). *The definitive guide* (3rd ed.). Sebastopol, CA: O'Reilly Media, Yahoo Press. (1)

Appendix

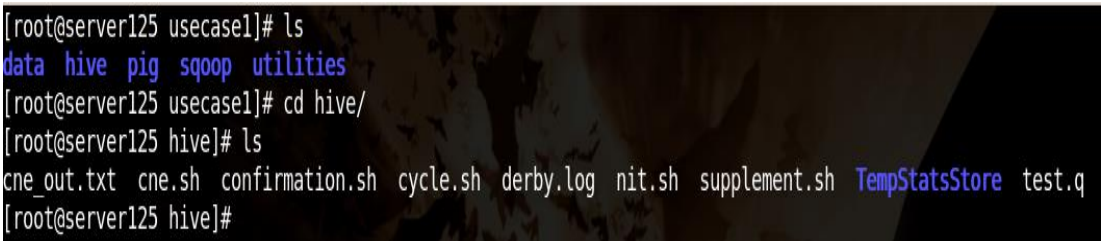
- Logon to server 125 with proper authentication

- Check for the file usecase1 and get access to it





- Data has all the records, pig has confirmation table, statements files, Hive has hive related files



Once, all the requested files have got, apply the split query and process the data.