Abstract— Analysis of data is important to find the meaningful information contained in it. There are many data storage and manipulation tools. Initially data was stored and analysed using files, tables, databases, data warehouse. However, in the current scenario of Big Data, these traditional methods are not efficient enough to do the analysis. Hadoop, open source software which provides support for distributed processing is implemented. In this paper, a detailed explanation about Hadoop and its components is given. Also, comparison of Hadoop components Pig, Hive and Map Reduce with traditional methods is explained.

Keywords— Big Data, Data Management, RDBMS, Hadoop, Map Reduce

I. INTRODUCTION

Hadoop is a complete eco-system of open source projects that provide us the framework to deal with big data. However, there are many challenges of dealing with big data (on traditional systems) [1], [2]:

1. High capital investment in procuring a server with high processing capacity: Hadoop clusters work on normal commodity hardware and keep multiple copies to ensure reliability of data. A maximum of 4500 machines can be connected together using Hadoop.
2. Enormous time taken: The process is broken down into pieces and executed in parallel, hence saving time. A maximum of 25 Petabyte (1 PB = 1024 TB) data can be processed using Hadoop.
3. Error Handling: Hadoop builds back up data-sets at every level. It also executes query on duplicate datasets to avoid process loss in case of individual failure. These steps make Hadoop processing more precise and accurate.
4. Difficulty in program query building: Queries in Hadoop are as simple as coding in any language. Building the query should enable parallel processing.

Big Data is a shift to scalable, elastic computing infrastructure; an explosion in the complexity and variety of data available; and the value that come from combining disparate data for comprehensive analysis make Hadoop a critical new platform for data-driven enterprises like restaurants. Our Database consists of two main components [3]:

1. HDFS (Hadoop Distributed File System).
2. MapReduce

1.1. HDFS [4]

The file store is called the Hadoop Distributed File System, or HDFS. HDFS provides scalable fault-tolerant storage at low cost. The HDFS software detects and compensates for hardware issues, including disk problems and server failure. HDFS stores files across a collection of servers in a cluster. Files are decomposed into blocks, and each block is written to more than one (the number is configurable, but three is common) of the servers. This replication provides both fault-tolerance (loss of a single disk or server does not destroy a file) and performance (any given block can be read from one of several servers, improving system throughput).

HDFS ensures data availability by continually monitoring the servers in a cluster and the blocks that they manage. Individual blocks include checksums. When a block is read, the checksum is verified, and if the block has been damaged it will be restored from one of its replicas. If a server or disk fails, all of the data that it has stored is replicated to some other node or nodes in the cluster, from the collection of replicas. As a result, HDFS runs very well on commodity hardware. It tolerates and compensate for failures in the cluster [5]. As clusters get large, even very expensive fault-tolerant servers are likely to fail. Because HDFS expects failure, organizations can spend less on servers and let software compensate for hardware issues.
1.2. MapReduce [6]

Map Reduce is the main programming paradigm behind Hadoop. It includes a software component called the job scheduler. The job scheduler is responsible for choosing the servers that will run each user job and for scheduling execution of multiple user jobs on a shared cluster. The job scheduler consults the NameNode for the location of all of the blocks that make up the file or files required by a job. Each of those servers is instructed to run the user’s analysis code against its local block or blocks.

The MapReduce processing infrastructure includes an abstraction called an input split that permits each block to be broken into individual records. There is special processing built in to reassemble records broken by block boundaries. The user code that implements a map job can be virtually anything. MapReduce allows developers to write and deploy code that runs directly on each Data Node server in the cluster. That code understands the format of the data stored in each block in the file, and can implement simple algorithms (count the number of occurrences of a single word, for example) or much more complex ones (e.g. natural language processing, pattern detection and machine learning, feature extraction, or face recognition). At the end of the map phase of a job, results are collected and filtered by a reducer.

MapReduce guarantees that data will be delivered to the reducer in sorted order, so output from all mappers is collected and passed through a shuffle and sort process. The sorted output is then passed to the reducer for processing. Results are typically written back to HDFS. Because of the replication built into HDFS, MapReduce is able to provide some other useful features.

For example, if one of the servers involved in a MapReduce job is running slowly most of its peers have finished, but it is still working the job scheduler can launch another instance of that particular task on one of the other servers in the cluster that stores the file block in question. This means that overloaded or failing nodes in a cluster need not stop, or even dramatically slow down, a MapReduce job.

This document is a template. An electronic copy can be downloaded from the Journal website. For questions on paper guidelines, please contact the journal publications committee as indicated on the journal website. Information about final paper submission is available from the conference website.

II. APACHE OPEN SOURCE HADOOP ECOSYSTEM ELEMENTS

The Apache Hadoop project actively supports multiple projects intended to extend Hadoop’s capabilities and make it easier to use. There are several top-level projects to create development tools as well as for managing Hadoop data flow and processing. Many commercial third-party solutions build on the technologies developed within the Apache Hadoop ecosystem. Spark, Pig, and Hive are three of the best-known Apache Hadoop projects. Each is used to create applications to process Hadoop data.

2.1 Hive [7], [8]

Hive is data warehousing software that addresses how data is structured and queried in distributed Hadoop clusters. Hive is also a popular development environment that is used to write queries for data in the Hadoop environment. It provides tools for ETL operations and brings some SQL-like capabilities to the environment. Hive is a declarative language that is used to develop applications for the Hadoop environment, however it does not support real-time queries. Hive has several components, including:

- HCatalog – Helps data processing tools read and write data on the grid. It supports MapReduce and Pig.
- WebHCat – It helps to use an HTTP/REST interface to run MapReduce, Yarn, Pig, and Hive jobs.
- HiveQL – Hive’s query language intended as a way for SQL developers to easily work in Hadoop. It is similar to SQL and helps both structure and query data in distributed Hadoop clusters.

Hive queries can run from the Hive shell, JDBC, or ODBC. MapReduce (or an alternative) breaks down HiveQL statements for execution across the cluster.

Hive also allows MapReduce-compatible mapping and reduction software to perform more sophisticated functions. However, Hive does not allow row-level updates or support for real-time queries, and it is not intended for OLTP workloads. Many consider Hive to be much more effective for processing structured data than unstructured data, for which Pig is considered advantageous.

2.2 Pig [9], [10]

Pig is a procedural language for developing parallel processing applications for large data sets in the Hadoop environment. Pig is an alternative to Java programming for MapReduce, and automatically generates MapReduce functions. Pig includes Pig Latin, which is a scripting language. Pig translates Pig Latin scripts into MapReduce, which
can then run on YARN and process data in the HDFS cluster. Pig is popular because it automates some of the complexity in MapReduce development.

Pig is commonly used for complex use cases that require multiple data operations. It is more of a processing language than a query language. Pig helps develop applications that aggregate and sort data and supports multiple inputs and exports. It is highly customizable, because users can write their own functions using their preferred scripting language. Ruby, Python and even Java are all supported. Thus, Pig has been a popular option for developers that are familiar with those languages but not with MapReduce. However, SQL developers may find Hive easier to learn.

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

To write data analysis programs, Pig provides a high-level language known as Pig Latin. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data. To analyse data using Apache Pig, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as Pig Engine that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

### 2.2.1 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.
- **Extensibility**: Using the existing operators, users can develop their own functions to read, process, and write data.
- **UDF’s**: Pig provides the facility to create User-defined Functions in other programming languages such as Java and invoke or embed them in Pig Scripts.
- **Handles all kinds of data**: Apache Pig analyses all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

### 2.2.2 Applications of Apache Pig

Apache Pig is generally used by data scientists for performing tasks involving ad-hoc processing and quick prototyping. Apache Pig is used –

- To process huge data sources such as web logs.
- To perform data processing for search platforms.
- To process time sensitive data loads.

### 2.2.3 Pig Queries:

The pig queries run on Pig shell known as Grunt shell. To access Pig, we have three modes to open grunt shell:

- **Local mode**: Only local is accessible
  
  Pig –x local

- **MapReduce**: only HDFS is accessible
  
  Pig –x mapreduce

- **Embedded**: A script of pig queries are executed
  
  Pig –x local ‘/location’

**Queries:**

- **Data Loading**
  
  Mysample = load '/path' using PigStorage('\t') as (id: int, name:chararray….)

- **Data Loading without schema**
  
  Mysample = load '/path' using PigStorage('\t');

- **Describe the schema of the Mysample**
  
  Mysample = load '/path' using PigStorage('\t');
To select particular fields
Y = foreach Mysample generate name, salary;

Tell the history of queries
History;

To display the data on grunt shell
Dump Mysample;

Used to define the schema after loading the data
Z = foreach mysample generate $0 as name: chararray, $2 as salary: int;

Order by
x = order Mysample by name desc;

Group by
x = group Mysample by name quarter;

Cogroup:
cg = cogroup Mysample by id, sales by id;

Join:
j = join rel1 by id, rel2 by id;
j = join rel1 by id left outer, rel2 by id;
j = join rel1 by id right outer, rel2 by id;
j = join rel1 by id inner, rel2 by id;

Union:
a = load '/path';
b = load '/path';
u = union a, b;

Predefined function:
a = load '/path'....
b = group a by id;
c = foreach b generate group, SUM or COUNT or MAX or MIN (a.sales);

2.3 HBase [11], [12]
HBase is a scalable, distributed, NoSQL database that sits on top of HFDS. It was designed to store structured data in tables that could have billions of rows and millions of columns. It has been deployed to power historical searches through large data sets, especially when the desired data is contained within a large amount of unimportant or irrelevant data (also known as sparse data sets). It is also an underlying technology behind several large messaging applications, including Facebook’s.

HBase is not a relational database and wasn’t designed to support transactional and other real-time applications. It is accessible through a Java API and has ODBC and JDBC drivers. HBase does not support SQL queries, however there are several SQL support tools available from the Apache project and from software vendors. For example, Hive can be used to run SQL-like queries in HBase.

2.4 Oozie [13]
Oozie is the workflow scheduler that was developed as part of the Apache Hadoop project. It manages how workflows start and execute, and also controls the execution path. Oozie is a server-based Java web application that uses workflow definitions written in hPDL, which is an XML Process Definition Language similar to JBOSS JBPMT jPDL. Oozie only supports specific workflow types, so other workload schedulers are commonly used instead of or in addition to Oozie in Hadoop environments.
2.5 Sqoop (SQL+Hadoop) [14]

Think of Sqoop as a front-end loader for big data. Sqoop is a command-line interface that facilitates moving bulk data from Hadoop into relational databases and other structured data stores. Using Sqoop replaces the need to develop scripts to export and import data. One common use case is to move data from an enterprise data warehouse to a Hadoop cluster for ETL processing. Performing ETL on the commodity Hadoop cluster is resource efficient, while Sqoop provides a practical transfer method.

An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

III. COMPARISON BETWEEN PIG, HIVE, MAP REDUCE & SQL

3.1 Apache Pig vs. MapReduce

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

<table>
<thead>
<tr>
<th>Apache Pig</th>
<th>MapReduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Pig is a data flow language.</td>
<td>MapReduce is a data processing paradigm.</td>
</tr>
<tr>
<td>It is a high level language.</td>
<td>MapReduce is low level and rigid.</td>
</tr>
<tr>
<td>Performing a Join operation in Apache Pig is pretty simple.</td>
<td>It is quite difficult in MapReduce to perform a Join operation between datasets.</td>
</tr>
<tr>
<td>Any novice programmer with a basic knowledge of SQL can work conveniently with Apache Pig.</td>
<td>Exposure to Java is must to work with MapReduce.</td>
</tr>
<tr>
<td>Apache Pig uses multi-query approach, thereby reducing the length of the codes to a great extent.</td>
<td>MapReduce will require almost 20 times more the number of lines to perform the same task.</td>
</tr>
<tr>
<td>There is no need for compilation. On execution, every Apache Pig operator is converted internally into a MapReduce job.</td>
<td>MapReduce jobs have a long compilation process.</td>
</tr>
</tbody>
</table>

3.2 Apache Pig vs. SQL

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

<table>
<thead>
<tr>
<th>Pig</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pig Latin is a procedural language.</td>
<td>SQL is a declarative language.</td>
</tr>
<tr>
<td>In Apache Pig, schema is optional. We can store data without designing a schema</td>
<td>Schema is mandatory in SQL.</td>
</tr>
<tr>
<td>The data model in Apache Pig is nested relational.</td>
<td>The data model used in SQL is flat relational.</td>
</tr>
<tr>
<td>Apache Pig provides limited opportunity for Query optimization.</td>
<td>There is more opportunity for query optimization in SQL.</td>
</tr>
</tbody>
</table>

In addition to above differences, Apache Pig Latin –
- Allows splits in the pipeline.
- Allows developers to store data anywhere in the pipeline.
- Declares execution plans
- Provides operators to perform ETL (Extract, Transform, and Load) functions.

3.3 Apache Pig vs. Hive

Both Apache Pig and Hive are used to create MapReduce jobs. And in some cases, Hive operates on HDFS in a similar way Apache Pig does. In the following table, we have listed a few significant points that set Apache Pig apart from Hive.

<table>
<thead>
<tr>
<th>Apache Pig</th>
<th>Hive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Pig uses a language called Pig Latin. It was originally created at Yahoo.</td>
<td>Hive uses a language called HiveQL. It was originally created at Facebook.</td>
</tr>
<tr>
<td>Pig Latin is a data flow language.</td>
<td>HiveQL is a query processing language.</td>
</tr>
<tr>
<td>Pig Latin is a procedural language and it fits in pipeline paradigm.</td>
<td>HiveQL is a declarative language.</td>
</tr>
</tbody>
</table>
IV. CONCLUSIONS

The paper discusses about the basic concepts of Big Data and its importance in current systems. It also explains the basic components and methodology of Hadoop. Hadoop works on the concept of distributed parallel processing. The later part of the paper includes the comparison between traditional data analysis methods with the current technologies. Also, a comparison between Hadoop components Pig and Hive is also explained.

REFERENCES

[1] Leons Petrazickis and Marius Butuc, “Crunching Big Data with Hadoop and BigInsights in the Cloud”, pg 241-242, Information Management Technologies


