# SYLLABUS

## Big Data Analytics - (3170722 / 3161607)

| Credits | Examination Marks | | | | Total Marks |
|---|---|---|---|---|---|
| C | Theory Marks | | Practical Marks | | |
| | ESE (E) | PA (M) | ESE (V) | PA (I) | |
| 4 | 70 | 30 | 30 | 20 | 150 |

### 1. Introduction to Big Data

Introduction to Big Data, Big Data characteristics, Challenges of Conventional System, Types of Big Data, Intelligent data analysis, Traditional vs. Big Data business approach, Case Study of Big Data Solutions. **(Chapter - 1)**

1 2 3 6

### 2. Hadoop

History of Hadoop, Hadoop Distributed File System : Physical organization of Compute Nodes, Components of Hadoop Analyzing the Data with Hadoop, Scaling Out, Hadoop Streaming, Design of HDFS,Java interfaces to HDFS Basics, Developing a Map Reduce Application, How Map Reduce Works, Anatomy of a Map Reduce Job run, Failures, Job Scheduling, Shuffle and Sort, Task execution, Map Reduce Types and Formats, Map Reduce Features, Hadoop environment, Setting up a Hadoop Cluster, Cluster specification, Cluster Setup and Installation, Hadoop Configuration, security in Hadoop, Administering Hadoop, Monitoring-Maintenance, Hadoop benchmarks, Hadoop in the cloud. **(Chapter - 2)**

### 3. NoSQL

What is NoSQL ? NoSQL business drivers; NoSQL case studies; NoSQL data architecture patterns : Key-value stores, Graph stores, Column family (Bigtable) stores, Document stores, Variations of NoSQL architectural patterns; Using NoSQL to manage big data : What is a big data NoSQL solution ? Understanding the types of big data problems; Analyzing big data with a shared-nothing architecture; Choosing distribution models : master-slave versus peer-to-peer; Four ways that NoSQL systems handle big data problems. **(Chapter - 3)**

### 4. Mining Data Stream

Introduction to Streams Concepts, Stream Data Model and Architecture, Stream Computing, Sampling Data in a Stream, Filtering Streams, Counting Distinct Elements in a Stream, Estimating moments, Counting oneness in a Window, Decaying Window, Real time Analytics Platform (RTAP) applications, Case Studies, Real Time Sentiment Analysis, Stock Market Predictions, Using Graph Analytics for Big Data : Graph Analytics. **(Chapter - 4)**

### 5. Frameworks

Applications on Big Data Using Pig and Hive, Data processing operators in Pig, Hive services, HiveQL, Querying Data in Hive, fundamentals of HBase and ZooKeeper, IBM InfoSphere BigInsights and Streams. **(Chapter - 5)**

### 6. Spark

Introduction to Data Analysis with Spark, In-Memory Computing with Spark, Spark Basics, Interactive Spark with PySpark, Writing Spark Applications. **(Chapter - 6)**

# 1 Introduction to Big Data

## *Syllabus*

*Introduction to Big Data, Big Data Characteristics, Challenges of Conventional System, Types of Big Data, Intelligent Data Analysis, Traditional vs. Big Data Business Approach, Case Study of Big Data Solutions.*

## *Contents*

## 1.1 Introduction

GTU : Winter-16, 17, Summer-17, 18, 19

- In order to understand 'Big Data', we first need to know what 'data' is. Oxford dictionary defines 'data' as -

  "The quantities, characters, or symbols on which operations are performed by a computer, which may be stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media."
- Data with huge volume is known as 'Big Data'. This term is used to define collection of data which have huge volume and which is growing at the faster rate day by day. This kind of data is high volume and complex which cannot be stored or efficiently processed in traditional data management tool.
- Following are some the examples of 'Big Data'.
  - The New York Stock Exchange generates about one terabyte of new trade data per day.
  - Social Media Impact : Statistic shows that 500 + terabytes of new data gets ingested into the databases of social media site Facebook, every day. The data is mainly generated in terms of video, images, messages, comment, likes etc.
  - Single Jet engine can generate 10+terabytes of data in 30 minutes of a flight time. With many thousand flights per day, generation of data reaches up to many *Petabytes*.
- Big data is a term for data sets that are complex or high volume which traditional data processing applications are inadequate. Challenges include capture, analysis, data curation, searching, sharing, storage, transfer, visualization, querying, updating and information privacy. The term often refers simply to the use of user behavior analysis, predictive analytics, or other advanced approaches of data analytics which extract value from data, and seldom to a particular size of data set. Productivity and precision in Big Data may lead to confident and successful decision making, and better choices brings about more prominent operational effectiveness, decreased cost and risk.
- Analysis of data sets can acquire new equivalency to, identify business trends, fight against crime, prevent diseases, etc. Executives, scientists, medicine practitioners, advertising agencies and government commonly meet problems with large amount of data such as web search, business information processing, urban information processing and finance.

- Data sets are increasing very quickly in part because they are increasingly gathered by number of information-sensing mobile devices, web logs, software logs, remote sensing devices, RFID readers, microphones, cameras and wireless sensor networks. The world's capacity to store information is being doubled every 40 months since 1980; as of 2012, every day 2.5 Exabyte of data is generated.
- What is considered "Big Data" differs based on the capacity of the users and their tools, and extending abilities make Big Data a moving objective.
- Some companies will need to rethink about their data management possibilities, who are facing hundred of gigabytes of data for first time.
- For others, it may take tens or hundreds of terabytes before data size becomes very large in volume which can be taken into consideration."

### 1.1.1 Definition

GTU : Winter-19

'Big Data' is a term used to describe collection of data that is huge in size and yet growing exponentially with time. Such a data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently.

## 1.2 Big Data Characteristics

GTU : Winter-16, 17, 18, 19, Summer-17, 18, 19

There are specific characteristics that define big data. These characteristics are also known as the four V's of Big Data which are volume, variety, velocity, and veracity.

**Volume**

The main characteristic that makes data "big" is the sheer volume. It makes no sense to focus on minimum storage units because the total amount of information is growing exponentially every year.

The name 'Big Data' itself is related to a size which is enormous. Size of data plays pivotal part in deciding a value out of data. Likewise, regardless of whether a specific data can really be considered as a Big Data or not, is based on the size of data. Hence, '**Volume**' is one characteristic which needs to be considered while dealing with 'Big Data'.

For that same year, EMC, a hardware company that makes data storage devices, thought it was closer to 900 Exabyte and would grow by 50 percent every year. No one really knows how much new data is being generated, but the amount of information being collected is large.

If your store of old data and new incoming data has gotten so large that you are having difficulty handling it, that's big data. Remember that it's going to keep getting bigger. Your consultant needs to recommend a scalable solution that can grow with your data.

### Variety

Variety is one the most interesting developments in technology as more and more information is digitized. Traditional data types (structured data) include things on a bank statement like date, amount, and time. These are things that fit neatly in a relational database.

Variety refers to heterogeneous sources and the nature of data, both structured and unstructured. During prior days, database and spreadsheets were the solitary source of data considered by the majority of the applications. Presently, data as messages, photographs, recordings, monitoring gadgets, PDFs, sound, and so on is also being considered in the analysis applications. These different forms in data have some problems related to storage, mining and analyzing data.

Structured data is expanded by unstructured data, which is the place where things like Twitter channels, sound documents, MRI pictures, site pages, web logs are put – anything that can be caught and put away however doesn't have a meta model (a bunch of rules to outline an idea or thought - it characterizes a class of data and how to communicate it) that neatly defines it.

Unstructured data is a principal idea in big data. The most ideal approach to comprehend unstructured data is by contrasting it with structured data. Consider structured data that is well described in a bunch of rules. For instance, cash will consistently be numbers and have at least two decimal points; names are communicated as text; and dates follow a particular pattern.

With unstructured data, there are no rules. An image, a voice recording, a tweet – they all can be distinctive yet express thoughts and ideas dependent on human understanding. One of the objectives of Big Data is to make use of technology to take this unstructured data and make sense of it.

The definition of big data depends on whether the data can be ingested, processed, and examined in a time that meets a particular business's requirements. For one company or system, big data may be 50 TB; for another, it may be 10 PB.

### Veracity

Veracity refers to the trustworthiness of the data.

Veracity is probably the toughest nut to crack. If you can't believe on the data itself, the data sources, or the process used to distinguish which data points are important, you have a veracity issue. Perhaps the most serious issue with big data is the tendency for mistakes to snowball. Manual entry error, repetition and corruption all influence the value of data. The counseling firm requirements to help in cleaning existing data and set up processes to lessen the aggregation of dirty data going ahead.

### Velocity

The term 'velocity' refers to the speed of age of data. How quick the data is generated and processed to fulfill the needs, decides genuine potential in the data.

Big Data Velocity manages the speed at which data streams in from sources like business measures, application logs, networks and online media websites, sensors, cell phones, and so on. The progression of information is enormous and constant.

Velocity is the frequency of incoming data that needs to be processed. Consider the number of SMS messages, Facebook notices, or charge card swipes are being sent on a specific telecom transporter the entire day, and you'll have a decent appreciation of velocity. A real time application like Amazon Web Services Kinesis is an illustration of an application that handles the velocity of data.

If you have one or more business processes that require real-time data analysis, you have a velocity challenge. Solving this issue may mean extending your private cloud using a hybrid model that permits blasting for extra compute power as required for data analysis. Expert may offer recommendations for equipment, programming, and business measure changes to deal with the present high speed data.

## 1.3 Challenges of Conventional System

GTU : Winter-16, 17, Summer-17, 18

There are majorly three challenges of conventional system, which are as follows :

**i. Volume of Data :**

- The volume of data is increasing day by day, especially the data generated from machine, telecommunication service, airline service, data from sensors etc.
- The rapid growth in data every year is coming with new source of data which are emerging.
- The growth in volume of data is so rapid that it is expected by IBM that by 2020 around 35 zettabyte of data will get stored in the world.

**ii. Processing and Analyzing :**

- Processing of such large volume of data is major challenge and is very difficult. Organization make use of such large volume of data by analyzing in order to achieve their business goals.
- Taking out insights from such large amount of data is time consuming and it also takes lot of effort to do.
- Processing and analyzing of data is also costly since the data is in different format and is complex.

**iii. Management of Data :**

- As the data gathered have different formats like structured, semi-structured and unstructured, it is very challenging to manage such different variety of data.

**GTU : Winter-17, 18, 19, Summer-19**

## 1.4 Types of Big Data

There are three types Big data could be found

1. Structured  2. Un-structured  3. Semi-structured

### Structured

Structured data is any data which can be stored, accessed and processed in the form of particular format. Over the period of time, talent in computer science have achieved greater success in building approaches for working with kind of data having fixed format and also determining value out of it. However, now days, we are forecasting issues when size of structured data grows to a huge extent, typical sizes are being in the range of multiple zettabyte.

Looking at these numbers one can easily understand why the name 'Big Data' is given and imagine the challenges involved in its storage and processing.

Data stored in a relational database management system is one example of a 'structured' data.

Examples of Structured Data

An 'Employee' table in a database is an example of Structured Data

| Employee_ID | Employee_Name | Gender | Department | Salary |
|---|---|---|---|---|
| 2125 | Rajesh Kulkarni | Male | Finance | 60000 |
| 6542 | Pratibha Joshi | Female | Admin | 63000 |
| 6472 | Shushil Roy | Male | Admin | 55000 |
| 7684 | Shripad Deshpande | Male | Finance | 51000 |

### Un-structured

Any data with unknown format is classified as unstructured data. In addition to the size being large, un-structured data poses multiple problems in its processing for determining value out of it. Typical example of unstructured data is, a diversified data source having a combination of simple text files, images, videos etc. Now a day organizations have large amount of data available with them but unfortunately they don't *know how to determine* value out of it since this data is in unstructured format.

*Output returned by "Google Search"* is an example of unstructured data.

### Semi-structured

Semi-structured data can contain both the forms of data. Semi- structured data can be called as structured is form but it is not defined with certain format, e.g. in relation DBMS definition of table. Example of semi-structured data is a data represented in XML file.

Example of Semi-structured Data

Personal data stored in a XML file –

```
<rec><name>Avinash Jha</name><sex>Male</sex><age>29</age></rec>
<rec><name>Charmi Furia</name><sex>Female</sex><age>26</age></rec>
<rec><name>Pinal Hansora</name><sex>Male</sex><age>31</age></rec>
<rec><name>Tushar Patil</name><sex>Male</sex><age>30</age></rec>
<rec><name>Paresh Chauhan</name><sex>Male</sex><age>29</age></rec>
```

## 1.5 Intelligent Data Analysis

- Intelligent Data Analysis provides a forum for the assessment of issues identified with the research and applications of Artificial Intelligence methods in data analysis across different type of controls.
- In big data, large amount of unstructured data is a challenge. Diverse data sets make it more difficult to structure it.
- These strategies includes : all zones of data visualization, data pre-preparing (combination, altering, change, separating, examining), data engineering, database mining procedures, devices and applications, use of domain knowledge in data analysis, big data applications, developmental algorithms, AI, neural nets, fuzzy logic, statistical pattern identification, knowledge filtering, and post-processing.
- It reveals implicit, previously unknown and potentially valuable information or knowledge from large amounts of data.
- Intelligent data analysis is also a kind of decision support process. It is a research area and is still under the research.
- Based on artificial intelligence, machine learning, pattern recognition, statistics, database and visualization technology mainly, IDA automatically extracts useful information, necessary knowledge and interesting models from a lot of online data in order to help decision makers make the right choices.
- IDA has basically following major steps : -

**i) Data preparation :** Data preparation includes, extracting or collecting relevant data from source and then creating an data set.

**ii. Rule finding or data mining :** Rule finding is working out rules contained in the data set by means of certain methods or algorithms.

**iii. Result validation and explanation :** Result validation means examining these rules. Also whether result explanation is giving intuitive, reasonable and understandable descriptions using logical reasoning.

- The main goal of intelligent data analysis is to extract useful knowledge, the process demands a combination of extraction, analysis, conversion, classification, organization, reasoning, and so on.
- Again for the data preparation and rule based combinations, we can use machine learning and deep learning concepts.
- To improve the results few concepts can be used in some proper manner like theory and model, algorithm and simulation, feature extraction, parallel and distributed data analysis and many more.
- It will be helpful in many areas such as :

  Banking and Securities.

  Communications, Media and Entertainment.

  Healthcare Providers.

  Education. ...

  Manufacturing and Natural Resources. ...

  Government. ...

  Insurance. ...

  Retail and Wholesale trade.

## 1.6 Traditional vs Big Data Business Approach

GTU : Winter-19

- The technology is changing every passing day, individuals are getting familiar with different techniques. This helps to solve various problems which were ignored due to lack of sources and assets. Much equivalent to that, data storing is something that is excessively shabby and cares filled work for any affiliation.
- Individuals are trading their mode; number of individual discover large information less complex than regular information so it might be anything but hard to deal with a wide range of issues and troubles that occur during this process. It might be only possible by installing the huge instruments like Big Data which can have the choice to store such data rapidly and dissect it in a large sum without requiring some genuine energy.

- Considering everything, the terms are not agreeable for lots of individuals. Here is the point that can help you with that, and let you know how function in both cases.
- Below are few majors to compare tradition and big data approach.

### Confidentiality and Data Accuracy

Conventional information, it's difficult to keep up the exactness and private as the nature of the information is high thus as to store large volume of information is expensive. It impacts the data breaking which similarly declines the eventual outcome of precision and mystery. However, Big data makes this work significantly easy and trouble free when appeared differently in relation to standard data. Moreover, it gives high exactness and makes the results increasingly exact.

### Data Relationship

Large volume of data contains an immense amount of information which makes the database relationship hard to understand. It impacts the information things which moreover makes the interpretation level problematic. In any case, with Traditional information, it's definitely not hard to encounter all data and information without standing up to a large amount of trouble. It similarly helps in figuring out the association among information and information things with no issue.

### Data storage size

The size of limit in data is huge. In Traditional Data, it's hard to store a large amount of data. The primary certain entirety can be taken care of; regardless, Big Data can store large volume of data with no issue. The traditional database can save information in the number of GB to TB. Considering everything, the large information can spare many terabytes, petabytes, and altogether more. It similarly helps in putting aside the proportion of money that spends on the standard database for capacity. By taking care of large amount of data decreases extra sources and money.

### Different types of data

The standard database is essentially for custom structure, for instance, taking care of information in different or mixed organizations in a record. Moreover, it just gives a brief about the issues. Regardless, for any affiliation it's critical to see each and every issue and get the best comprehension of information to give indications of progress information about the structure, in any case, it's unreasonable with Traditional information. Large volume of Data however gives better variation and metadata structure gives better access to information which helps in improving the work.

## 1.7 Case Study of Big Data Solution

- Undoubtedly Big Data has become a major game change in the most part of the cutting edge industries over the last few of years. As Big Data keeps on going day by day, the number of various organizations that are adopting Big Data keeps on expanding.
- Following are the few interesting big data case studies.

### 1. Walmart

- Walmart is the biggest retailer in the world and the world's biggest organization by revenue, with more than 2 million workers and 20000 stores in 28 nations. It started making use of big data analytics much before the word Big Data came into the picture.
- Walmart uses Data Mining to find designs that can be used to give product suggestions to the client, depending on which products were brought together.
- WalMart by applying successful Data Mining has expanded its conversion rate of customers. It has been speeding along big data analysis to give top tier e-commerce technologies with an intention to convey prevalent client experience.
- The main target of holding big data at Walmart is to enhance the shopping experience of customers when they are in a Walmart store.
- Big data solutions at Walmart are created with the expectation of upgrading worldwide sites and building innovative applications to customize the shopping experience for customers while expanding co-ordinations proficiency.
- Hadoop and NoSQL technologies are used to furnish internal customers with access to real-time data gathered from various sources and centralized for effective use.

### 2. Uber

- Uber is the best option for individuals around the globe when they consider moving people and making conveyances. It utilizes the individual information of the user to intently monitor which features of the service are generally used, to analyze usage pattern and to figure out where the services should be more engaged. Uber focuses around the organic market of the services because of which the costs of the services gave changes. In this manner perhaps the greatest use of data is surge pricing. For example, in the event that you are running late for an appointment and you book a taxi in a jam-packed spot then you should be prepared to pay double the amount.

- For instance, On New Year's Eve, the cost of driving for one mile can go from 150 to 1200. For the time being, surge pricing influences the rate of demand, while long term use could be the key for retaining or losing customers. Machine learning algorithms are considered to figure out where the demand is solid.

### 3. Netflix

- It is the most cherished American entertainment company work in online on-request web based video streaming for its customers. Netflix has been determined to be able to predict what precisely its customers will appreciate viewing with Big Data.
- All things considered, Big Data analytics is the fuel that fires the 'recommendation engine' intended to fulfill this need.
- More recently, Netflix began positioning itself as a content creator, not simply a distribution medium. Obviously, this technique has been solidly determined by data. Netflix's recommendation engines and new content choices are taken care of by data points, for example, what titles customers watch, how regularly playback halted, ratings are given, and so on.
- The organization's data structure incorporates Hadoop, Hive and Pig with a lot other traditional business intelligence.
- Netflix shows us that knowing precisely what customers need is easy to understand if the companies simply don't go with the assumptions and make choices dependent on Big Data.

## University Questions with Answers

**Winter - 2016 (IT)**

**Q.1** What is Big Data ? Explain characteristics of Big Data. **(Refer sections 1.1 and 1.2)** [7]

**Q.2** What are the benefits of Big Data ? Discuss challenges under Big Data. How Big Data Analytics can be useful in development of smart cities. **(Refer sections 1.1 and 1.3)** [7]

**Summer - 2017 (IT)**

**Q.3** What is big data ? Discuss it in terms of four dimensions, volume, velocity, variety and veracity. **(Refer sections 1.1 and 1.3)** [7]

**Summer - 2017 (CSE)**

**Q.4** What is Big Data Analytics ? Explain 4 V's of Big Data. Briefly discuss applications of Big Data. **(Refer sections 1.1 and 1.2)** [7]

Notes

# 2 Hadoop

## Syllabus

*History of Hadoop, Hadoop Distributed File System : Physical organization of Compte Nodes, Components of Hadoop Analyzing the Data with Hadoop, Scaling Out, Hadoop Streaming, Design of HDFS, Java interfaces to HDFS Basics, Developing a Map Reduce Application, How Map Reduce Works, Anatomy of a Map Reduce Job run, Failures, Job Scheduling, Shuffle and Sort, Task execution, Map Reduce Types and Formats, Map Reduce Features, Hadoop environment. Setting up a Hadoop Cluster, Cluster specification, Cluster Setup and Installation, Hadoop Configuration, ecurity in Hadoop, Administering Hadoop, Monitoring - Maintenance, Hadoop benchmarks, Hadoop in the cloud.*

## Contents

## 2.1 History of Hadoop

- In the history of hadoop, Doug Cutting and Mike Cafarella are two important people. They wanted to invent an approach to return Web search output quicker by dividing the data over more than a few machines and make calculations, so a few positions could be performed simultaneously.
- At that time, they were working on "Nutch", an open source search engine project. But during same period, the Google search engine project was also in progress.
- At that time, they were working on an open source search engine project called Nutch. But, at the same time, the Google search engine project also was in progress. So, Nutch was divided into two parts-one of the parts managed data processing, which the duo named Hadoop after the toy elephant that belonged to Cutting's child.
- Hadoop was released by Yahoo as an open source venture in 2008. Today, the Apache Software Foundation keeps up the Hadoop ecosystem.

## 2.2 Hadoop Distributed File System

**GTU : Winter -16**

### 2.2.1 Physical Organization of Compute Nodes

**GTU : Summer-17, 18, 19, Winter-17, 18, 19**

- The HDFS is based on the Google File System's design. Implementation of HDFS solves many problems which were present in distributed file systems like NFS (Network File System).
- HDFS implementation solves mainly following problem :
  - To be able to store huge volume of data.
  - To store data reliably.
  - To integrate with Hadoop's MapReduce.
- HDFS is implemented as a block-structured filesystem. HDFS architecture is shown in Fig. 2.2.1; every file is first divided into fixed size block and stored over a Hadoop cluster. These fixed size blocks are stored on various DataNodes, which are selected randomly one block after another.
- Since the file is stored on various DataNodes across the cluster, rights to use a file require rights to access multiple DataNodes. That brings to the conclusion that using HDFS you can store file even larger than the storage capacity of individual machine.
- Each data block is stored in different files by DataNode on its local file system.

- One of the requisite for file system like HDFS is the ability store, access and manages information about the blocks and files (i.e. metadata) reliably and also to give faster access to metadata store.

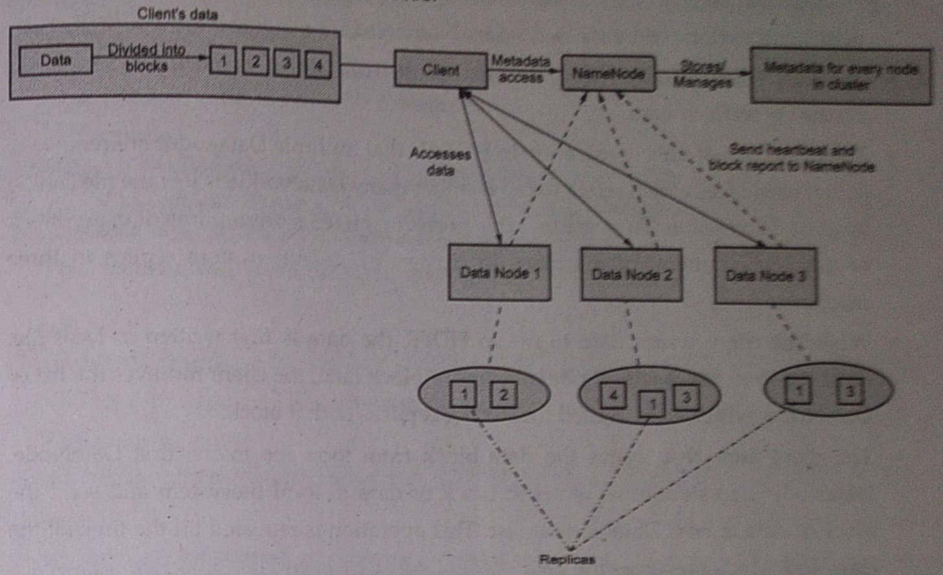


Fig. 2.2.1 HDFS Architecture

- So that the data remains in sync, HDFS provides dedicated machine called NameNode that is used to stores metadata for the filesystem over the cluster.
- Management of filesystem and controlling access to files by client is done by NameNode.
- Since metadata of every file is very less in size, the NameStore stores this in main memory, which permits faster access. Because of this, a NameNode having 4 GB of RAM is capable enough to store large number of files and directories.
- In case NameNode fails we can loss entire HDFS. In order to solve this problem Hadoop implemented a *Secondary NameNode.*
- Secondary NameNode cannot perform the operations of primary NameNode. It only performs checkpointing mechanism for primary NameNde i.e it stores the working state of HDFS NameNode.
- It also maintains two data structures which preserves the current state of HDFS : *an image file and an edit log.*

- The image file stores metadata state of HDFS and edit log stores the log of changes done on metadata since image file was made. When the NameNode starts or restarts, the current state is taken from the image file and edit log. This check pointing is performed by the Secondary NameNode.
- The HDFS block is of 64 MB in size by default. This large data block provides faster streaming reads of data.
- The drawback of file organization in HDFs is that multiple Datanodes are require to access single file, which refers to the fact that if any DataNodde is lost the file can no longer be accessible. To overcome this problem, HDFS store replicas of every block of data over number of machines in cluster. By default it store replica in three machines.
- While the client writes data to file in HDFS, the data is first written in local file. Once the local file reaches to default size of block data, the client retrieves the list of DataNodes which are assigned for storing replicas of that block.
- The client after that writes the data block from local file to the first DataNode. DataNode then stores this received block of data in local filesystem and send the block of data to next DataNode in list. This operation is repeated till the time all the DataNodes in list receives the data.
- In case while writing the block if any DataNode fails, then it is removed from the list. Once the write operation on current DataNode is done, the NameNode re-replicate the data to make up for the missing replica which is caused due to failure of DataNode.
- Optimization in placement of replica is one of the powerful features of HDFS, which is important to HDFS reliability and performance. NameNode takes care about the replication of blocks; NameNode also receives a heartbeat and block report periodically from every DataNode. A heartbeat is mainly used to make sure that DataNodes are operating properly, and block report permits to verify the list of blocks available on DataNode corresponding to the information in NameNode.
- Heartbeats are also used to find the failure of DataNode. The NameNode marks DataNode dead if it does not receive a heartbeat, and as a result NameNode does not send any new I/O request to that DataNode.
- The File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS).

- Following are frequently used HDFS commands :

### 1) appendToFile

```
Syntax : hadoop fs -appendToFile <localsrc> ... <dst>
```

- This command appends single or multiple sources from local file system to the destination file system. This command also reads input from standard input and appends it to destination file system.
- Example :

```
hadoop fs -appendToFile localfile /user/hadoop/hadoopfile
hadoop fs -appendToFile localfile1 localfile2 /user/hadoop/hadoopfile
hadoop fs -appendToFile - hdfs://avinash/sample/destfile
```

This command will read the input from stdin.

### 2. cat

```
Syntax : hadoop fs -cat URI
```

- This command copies source path to standard output i.e. stdout
- Example :

```
hadoop fs - cat /avinash/sample/test.txt
```

### 3. put

```
Syntax : hadoop fs -put [-f] [ - | <localsrc1> .. ]. <dest>
```

- This command copies single or multiple source from local file system to destination file system. The -f option will overwrite the destination if it already exists.
- The above command also reads input through stdin if the source is set to "-"

### 4. get

```
Syntax : hadoop fs -get <src> <localdest>
```

- The above command copies file from source to the local file system.
- Example :

```
hadoop fs -get /user/cloudera/jgd_avinash/Sample2.txt /home/cloudera/backup/
```

### 5. copyFromLocal

```
Syntax : hadoop fs -copyFromLocal <local src> URI
```

- The above command copies file available in local file system to the hdfs path give as destination.
- This command is similar to put command but in this source has to be local file system.
- Example :

```
hadoop fs -copyFromLocal Test1.txt /user/cloudera/jgd_avinash
```

### 6. copyToLocal

```
Syntax : hadoop fs -copyFromLocal URI <localdest>
```

- The above command copies the files from the path given in URI to the local file system path given in destination
- It is similar to the get command but in this command the destination has to be local file system.
- Example :

```
hadoop fs -copyFromLocal user/cloudera/jgd_avinash/test.txt  /home/cloudera/backup/
```

### 7. df

```
Syntax : hadoop fs -df [-h] URI [URI ...]
```

- The above command displays free space in the specified URI.
- The -h option will display the file sizes in human readable format.
- Example :

```
hadoop fs -df /user/cloudera/
```

### 8. cp

```
Syntax : hadoop fs -cp [-f] <hdfssrc><hdfsdest>
```

- The above command copies file from one HDFS location to another HDFS location.
- Example :

```
hadoop fs -cp /user/cloudera/jgd_avinash/test1 /user/cloudera/jgd_avinash/test2
```

### 9. mv

```
Syntax : hadoop fs -mv <source><dest>
```

- The above command will move the file from one HDFS location to another HDFS location.
- Example :

```
hadoop fs -mv /user/cloudera/jgd_avinash/test1.txt /user/cloudera/BDA_Backup/
```

### 10. moveFromLocal

```
Syntax : hadoop fs -moveFromLocal <localsource> <dest>
```

- The above command works similar as put command but the only difference is that the local source is deleted after copying.
- Example :

```
hadoop fs -moveFromLocal /home/avinash/sample.txt /user/jgd_avinash/demo/
```

### 11. mkdir

```
Syntax : hadoop fs -mkdir <location>
```

- The above command will create the new directory in the location specified.
- Example :

```
hadoop fs -mkdir /user/jgd_avinash/test
```

### 12. rm

```
Syntax : hadoop fs -rm URI
```

- The above command will delete the file specified in the URI.
- Example :

```
hadoop fs -rm /user/jgd_avinash/demo/sample.txt
```

### 13. rmdir

```
Syntax : hadoop fs -rmdir URI
```

- The above command will delete the directory.
- Example :

```
hadoop fs -rmdir /user/jgd_avinash/TestDir
```

### 14. chmod

```
Syntax : hadoop fs -chmod URI
```

- The above command is used to change the permission of the file or directory specified in URI.
- Example :

```
hadoop fs -chmod444  /user/jgd_avinash/demo
```

  - This command will change the permission for the directory demo to read (-r--r--r--) by user, group and others

### 15. chgrp

```
Syntax : hadoop fs -chgrp <Group> URI
```

- The above command is used to change the group name.
- Example :

```
hadoop fs -chgrp lit /student
```

### 16. chown

```
Syntax : hadoop fs -chown <OWNER><GROUP> URI
```

- This command will change the owner of the file or directory.

- Example :

```
hadoop fs -chown lit /student
```

**17. ls**

```
Syntax : hadoop fs -ls <path>
```

- The above command will list all the available files and subdirectories in the path specified
- Example :

```
hadoop fs -ls /user/jgd_avinash/
```

## 2.2.2 Component of Hadoop

**GTU : Summer-17, 18, 19, Winter-17, 18**

**Following are the core components of Hadoop ecosystem :**

- **HDFS :** A basic component of the Hadoop ecosystem is HDFS (Hadoop Distributed File System). HDFS is the technique using which a high volume of data can be distributed over a cluster of computers, and data is written once, but read many times for analysis. It provides the base for other tools, like HBase.
- **MapReduce :** Main execution framework of Hadoop is MapReduce, which is programming model for parallel, distributed data processing, dividing jobs into map and reduce phase. Programmers write MapReduce tasks for Hadoop, to access data fast by using data stored in HDFS. Because of the working nature of MapReduce, Hadoop provides parallel data processing, which resulting faster implementation.
- **HBase :** It is a column-oriented NoSQL database developed on top of HDFS, for faster read/write access to data HBase is used. HBase make use of Zookeeper for managing and ensuring that all of its components are working properly.
- **Zookeeper :** Zookeeper provides distributed coordination service in Hadoop. It is designed to run in a cluster environment and is highly available service which is used to manage Hadoop operations and various components that depends on it.
- **Oozie :** It is workflow system in Hadoop. Oozie is merged into the Hadoop stack, which is used for coordination of multiple MapReduce jobs execution. It is effective for management of significant amount of complexity, basing execution on external events which involves timing and presence of required data.

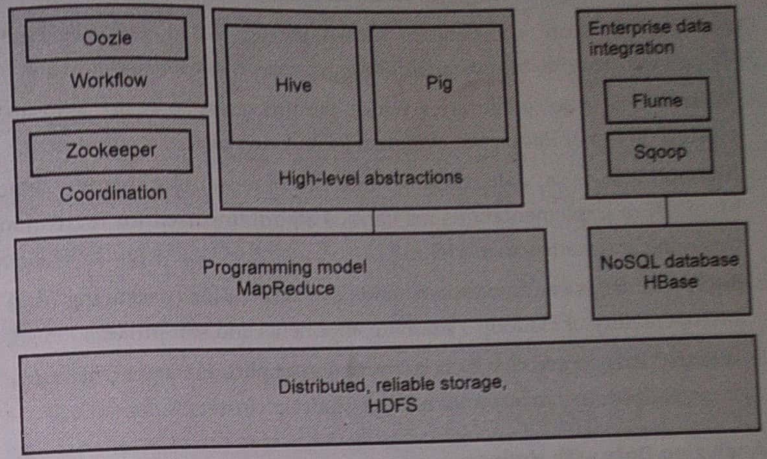


Fig. 2.2.2 High-level input and output actors in MapReduce

- **Pig :** It is an abstraction in the complexity of MapReduce programming. The Pig platform contains an environment for execution and a scripting language i.e. Pig Latin which is used for analysis of Hadoop data sets. Compiler of Pig converts the Pig Latin script into sequence of MapReduce programs.
- **Hive :** It is high-level, SQL-like language which is used to execute queries on data stored in Hadoop. Hive allows programmers who are not familiar with MapReduce to write data queries which are converted into MapReduce jobs in Hadoop. Like Pig, Hive was developed as an abstraction layer, but geared more toward database analysts who are more familiar using SQL than Java programming.
- The Hadoop ecosystem also includes various framework to integrate with rest of the enterprise :
  - **Sqoop** is a tool used for moving data between Hadoop and relational databases and data warehouses. Sqoop gives advantage to database to describe the schema for imported or exported data and MapReduce for performing operation in parallel and fault tolerance.
  - **Flume** is a reliable, distributed, and highly available service for collecting, merging and moving high volume of data sets from individual machines to HDFS efficiently. It is built on a simple and flexible architecture, and gives a streaming of data flows. It permits moving of data from different machines into HDFS.

- Other than the core components shown in Fig. 2.2.2, Hadoop's ecosystem is developing to provide recent capabilities and components, like following:
  - o **Whirr :** It is a set of libraries which permits users to easily spin-up Hadoop clusters on top of Rackspace, Amazon EC2, or any virtual environment.
  - o **Mahout :** It is a data mining and machine-learning library which gives MapReduce implementations for various algorithms used for regression testing, clustering and statistical modelling.
  - o **BigTop :** It is a framework and procedure for packaging and testing interoperability of Hadoop's similar components and sub-projects.
  - o **Ambari :** It is a project which is aimed to simplify management of Hadoop by giving support to manage and monitor Hadoop clusters.

## 2.2.3 Analyzing Data with Hadoop

- Big Data is clumsy due to its huge size, and needs devices to efficiently process and take out important outcomes from it. Hadoop is an open source software system and platform for storing, analyzing and processing data.
- The vast collection of data containing structured data of traditional databases and unstructured data such as text document, audio, video, is named as Big Data. Big data is also collection of different tools, techniques, platforms and frameworks. Big data also contains stock exchange data, search data, transport data, social media data and so on.
- Big Data isn't about the volume of the data, however more about what people use it for. Numerous associations like business corporations and educational institutions are utilizing this data to analyze and predict the results of certain activities.
- After gathering the data, it can be used for various functions like :
  - o Cost reduction
  - o The development of new products
  - o Making faster and smarter decisions
  - o Detecting faults
- Today, Big Data is used by almost all sectors including airline, government, banking, manufacturing and hospitality.
- There are four main libraries in Hadoop.

**1. Hadoop Common :**

In Hadoop, all the modules which uses utilities are provided by this library.

**2. Hadoop MapReduce :**

This works as a parallel framework for scheduling and processing the data.

**3. Hadoop YARN :**

YARN stands for Yet Another Resource Navigator. It is an upgraded version of MapReduce which is being used for processes running over Hadoop

**4. Hadoop Distributed File System**

HDFS : This stores and manages records over various machines or clusters. It also permits the data to be stored in an accessible format.

HDFS sends data to different nodes in cluster once and uses it number of times. When a query is raised, NameNode manages all the DataNode slave nodes that serve the given query. Hadoop MapReduce performs all the jobs assigned sequentially. Instead of MapReduce, Pig Hadoop and Hive Hadoop are used for better performances.

- Other packages that can support Hadoop are listed below.
  - o Apache Oozie : A scheduling system that manages processes taking place in Hadoop
  - o Apache Pig : A platform to run programs made on Hadoop
  - o Cloudera Impala : A processing database for Hadoop. Originally it was created by the software organisation Cloudera, but was later released as open source software
  - o Apache HBase : A non-relational database for Hadoop
  - o Apache Phoenix : A relational database based on Apache HBase
  - o Apache Hive : A data warehouse used for summarisation, querying and the analysis of data
  - o Apache Sqoop : Is used to store data between Hadoop and structured data sources
  - o Apache Flume : A tool used to move data to HDFS
  - o Cassandra : A scalable multi-database system

## 2.2.4 Scaling Out

- This is also referred to as Horizontal Scalling. Scaling out is basically the expansion of more machines or setting up the cluster. In horizontal scaling as opposed to expanding hardware capacity of individual machine you add more nodes to existing cluster and in particular, you can add more machines without disturbing the system.

- Subsequently there won't be any downtime or green zone, nothing of such sort while scaling out. So finally to meet your prerequisites you will have more machines working in parallel.

## 2.2.5 Hadoop Streaming

- Hadoop Streaming is a conventional API which permits writing Mappers and Reduces in any language
- Hadoop streaming is a utility that accompanies the Hadoop distribution. The utility permits you to write and run Map/Reduce jobs with any executable or content as the mapper as well as the reducer.
- For example :

```
$HADOOP_HOME/bin/hadoop jar
$HADOOP_HOME/hadoop-streaming.jar \
    -input myInputDirs \
    -output myOutputDir \
    -mapper /bin/cat \
    -reducer /bin/wc
```

- Any job in Hadoop must have two phases:
  - o Mapper and
  - o Reducer.

### 2.2.5.1 Working of Streaming

- When Mappers are described with scripts, each mapper task will launch the script as a different process. As it runs, it transforms input into lines and stores the lines to the standard information (STDIN).
- Parallelly, the mapper gathers the line-oriented outcomes from the standard output (STDOUT) and converts each line into a key-value pair. This is gathered as the output of the mapper.
- In this process, by default, the prefix of a line up to the first tab character is the key and the value will be the remaining line excluding tab character. The entire line is considered as key in case there is no tab character available in line
- For reducers, every single reducer task will launch the script as a different process, after then the reducer is initialized. After the reducer task runs, it changes over its input key-values pairs into lines and advances the lines to the standard input (STDIN) of the process.

- Parallelly, the reducer gathers the line-oriented outcomes from the standard output (STDOUT), and transforms each line into a key-value pair. This is gathered as the output of the mapper.
- In this process, by default, the prefix of a line up to the first tab character is the key and the value will be the remaining line excluding tab character.
- A simple streaming process :

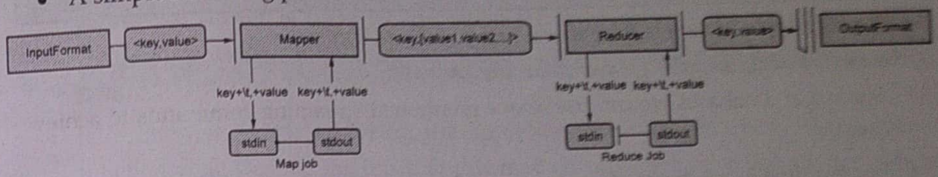


Fig 2.2.3 Streaming process

- Some Hadoop streaming commands are as follows.

| Parameters | Course of action | Description |
|---|---|---|
| -input directory/file-name | Mandatory | Input location for mapper. |
| -output directory-name | Mandatory | Output location for reducer. |
| -mapper executable or script or JavaClassName | Mandatory | Mapper executable. |
| -reducer executable or script or JavaClassName | Mandatory | Reducer executable. |
| -file file-name | Optional | Mapper, reducer, or combiner executable were made available locally on the compute nodes |
| -inputformat JavaClassName | Optional | Class you supply should restore key-value pairs of Text class. If not indicated, TextInputFormat is used as the default. |
| -outputformat JavaClassName | Optional | Class you supply should take key-value pairs of Text class. If not described, TextOutputformat is used as the default. |
| -partitioner JavaClassName | Optional | Class that decide a key is sent to which reduce. |
| -combiner streamingCommand or JavaClassName | Optional | Combiner executable for map output. |
| -cmdenv name=value | Optional | Passes the environment variable to streaming commands. |

| | | |
|---|---|---|
| -inputreader | Optional | Defines a record reader class for backwards-compatibility |
| -verbose | Optional | Verbose output. |
| -lazyOutput | Optional | Makes output lazily. For instance, if the output format depends on FileOutputFormat, the output file is made distinctly on the first call to output.collect. |

- Practically, following is the general line command syntax :

```
mapred streaming [genericOptions] [streamingOptions]
```

- For larger databases we can use above mentioned streaming commands to achieve output.
- For multiple input directories use :

```
hadoop jar hadoop-streaming.jar -input '/user/foo/dir1' -input '/user/foo/dir2'
```

- To process XML documents,

```
hadoopjar hadoop-streaming. jar -inputreader
"StreamXmlRecord,begin=BEGIN_STRING,end=END_STRING" ..... (rest of the command)
```

- A Sample map function can be as :

```
public void map   (WritableComparable key, Writable value,
OutputCollector output, Reporter reporter) throws IOException
{
output.collect((Text)value, null);
}
```

## 2.2.6 Design of HDFS

- HDFS is designed for storage of very large files having streaming data access pattern, running on cluster of commodity hardware. Refer Fig. 2.2.1 for HDFS architecture.
- Let us analyze the above statement in detail.

*Very large files*

- "Very large" in this setting implies documents that are many megabytes, gigabytes, or terabytes in size. There are Hadoop cluster running today that store petabytes of data.

*Streaming data access*

- HDFS is worked around the possibility that the most productive data processing design is a write once, read many times design. A dataset is typically generated or duplicated from source, and afterward different analyses are performed on that

dataset over the long period of time. Every analysis will include a large part, if not all, of the dataset, so a time to read the entire dataset is of a higher priority than the latency in reading the first record.

*Commodity hardware*

- Hadoop doesn't need costly, exceptionally dependable equipment. It is intended to run on cluster of commodity hardware, for which the possibility of node failure across the cluster is high, in any event for large clusters. HDFS is intended to continue working without an observable interference to the user despite such failure.
- It is likewise worth analyzing the applications for which using HDFS doesn't function well. Despite the fact that this may change later on, these are area where HDFS is not a good fit today :

*Low-latency data access*

- Applications that require low-latency access to data, during the several milliseconds range, won't function well with HDFS. Keep in mind, HDFS is enhanced for providing a high throughput of data, and this may be at the expense of latency. HBase is right now a better option for low-latency access.

*Lots of small files*

- Since the namenode holds filesystem metadata in memory, the limit to the number of records in a filesystem is administered by the amount of memory on the namenode. As a rule of thumb, each record, directory, and block takes around 150 bytes. Thus, for instance, in the event that you had 1,000,000 documents, each taking one block, you would require at least 300 MB of memory. In spite of the fact that storing large number of documents is feasible, billions is past the capacity of current hardware.

*Multiple writers, arbitrary file modifications*

- Records in HDFS might be written to by a single writer. Writes are constantly made toward the finish of the record, in append only style. There is no help for different writers or for changes at arbitrary offsets in the document.

## 2.2.7 Java Interface to HDFS Basics

- Hadoop has a theoretical idea of filesystems, of which HDFS is only one implementation. The Java abstract class org.apache.hadoop.fs.FileSystem represents the client interface to a filesystem in Hadoop, and there are various concrete implementations. Hadoop is written in Java, so most Hadoop filesystem interactions are mediated through the Java API.

- The HTTP REST API uncovered by the WebHDFS protocol makes it simpler for other programming languages to interact with HDFS. Native java client is faster than this HTTP interface, so if possible should avoid to use HTTP interface for huge data transfer.
- Java has the API for interaction with one of Hadoop's filesystems.

### 2.2.7.1 Reading Data using the FileSystem API

- A document in a Hadoop filesystem is represented by a Hadoop Path object. FileSystem is a general filesystem API, so the initial step is to recover an instance for the filesystem we need to use-HDFS, for this situation. There are a few static factory methods for getting a FileSystem instance which are as follows.

```
public static FileSystem get(Configuration conf) throws IOException
public static FileSystem get(URI uri, Configuration conf) throws IOException
public static FileSystem get(URI uri, Configuration conf, String user) throws IOException
public static LocalFileSystem getLocal(Configuration conf) throws IOException
```

- A Configuration object encapsulates a client or server's configuration, which is set using configuration documents read from the classpath, for example, core-site.xml.
- The first method returns the default filesystem as determined in core-site.xml. The second uses the given URI's scheme and authority to decide the filesystem to use, falling back to the default filesystem if no scheme is defined in the given URI. The third retrieves the filesystem as the given user, which is significant with regards to security. The fourth one retrieves a local filesystem instance.
- With a FileSystem instance close by, we call an open() method to get the input stream for a file. The first method uses a default buffer size of 4 kB. The second one gives a choice to user to define the buffer size.

```
public FSDataInputStream open(Path f) throws IOException
public abstract FSDataInputStream open(Path f, int bufferSize) throws IOException
```

**FSDataInputStream**

- The open() method on FileSystem returns an FSDataInputStream rather than a standard java.io class. This class is a specialization of java.io.DataInputStream and also supports random access, so you can read from any part of the stream :

```
package org.apache.hadoop.fs;

public class FSDataInputStream extends DataInputStream
implements Seekable, PositionedReadable {

}
```

- The seekable interface allows to look for a position in the file and provides a query method for the current offset from the start of the file (getPos()) . IOException occurs when seek() method is called with a position greater than length of file. The seek() method can be moved to an absolute, arbitrary position in file, unlike skip() method.

```
public interface Seekable {
void seek(long pos) throws IOException;
long getPos() throws IOException;
}
```

- For reading parts of a file at a given offset FSDataInputStream also implements the PositionedReadable interface.

```
public interface PositionedReadable {
public int read(long position, byte[] buffer, int offset, int length) throws IOException;
public void readFully(long position, byte[] buffer, int offset, int length) throws IOException;
public void readFully(long position, byte[] buffer) throws IOException;
}
```

- The read() method reads up to length bytes from the given position in the file into the buffer at the given offset in the buffer. The return value is the number of bytes actually read; callers should check this value, as it may be less than length.The readFully() methods will read length bytes into the buffer, unless the end of the file is reached, in which case an EOFException is thrown.

### 2.2.7.2 Writing Data

- There are various functions available of FileSystem class for creating a file. The easiest way is the function which takes Path object for a file to be created and returns an output stream to write to.

```
public FSDataOutputStream create(Path f) throws IOException
```

- There are overloaded versions of this function available which permits you to define whether to forcibly overwrite existing files, the replication factor of the file, the buffer size to use when writing the file, size of block for the file, and file permissions.
- Appending to an existing file is an alternative or creating new file by making use of append() function.

```
public FSDataOutputStream append(Path f) throws IOException
```

- The append method permits a single writer to modify an existing file by opening it and writing data from the final offset in the file. By using this API, applications which produce unbounded files, such as logfiles, can write to an existing file after having closed it. The append method is optional and not implemented by all Hadoop filesystems.

### FSDataOutputStream

- The create() method on FileSystem returns an FSDataOutputStream, which, like FSDataInputStream, has a method for querying the current position in the file :

```
package org.apache.hadoop.fs;
public class FSDataOutputStream extends DataOutputStream implements Syncable {
public long getPos() throws IOException {


}
}
```

- However, unlike FSDataInputStream, FSDataOutputStream does not allow seeking. This is because HDFS allows only sequential writes to an open file or appends to an already written file. There is no value while writing to be able to seek, as other than the end of file we cannot write anywhere else.
- FileSystem provides a function to create a directory also.

```
public boolean mkdirs(Path f) throws IOException
```

- This function creates all of the required parent directories if not exists and returns true if its successful.

### 2.2.7.3 Querying the Filesystem

- The FileStatus class encloses filesystem metadata for files and directories, including file length, block size, replication, updation time, ownership, and permission data. And also it gives the capability to navigate its directory structure and retrieve information about the files and directories.
- The method getFileStatus() on FileSystem provides a way of getting a FileStatus object for a single file or directory.
- A FileNotFoundException is thrown in case directory of file does not exist. However, in case you only want to know existence of a directory or file then using exists() function on FileSystem is more appropriate.

```
public boolean exists(Path f) throws IOException
```

### 2.2.7.4 File Patterns

- To process a bunch of files in single operation is common need. For instance, a MapReduce task for processing log may analyze a month's worth of files present in number of directories. In place of hacing to list every file and directory to specify the input, it is more appropriate to use wildcard characters to match multiple files with a single expression, an operation which is known as globbing. There are below two FileSystem methods available for processing globs in Hadoop :

```
public FileStatus[] globStatus(Path pathPattern) throws IOException
public FileStatus[] globStatus(Path pathPattern, PathFilter filter)
throws IOException
```

- The globStatus() methods return an array of FileStatus objects whose paths match the supplied pattern, sorted by path. An optional PathFilter can be defined to restrict the matches further.

### PathFilter

- Glob patterns are not always powerful enough to describe a set of files you need to access. For instance, it is not mainly possible to exclude a particular file using a glob pattern. To permit programmatic authority over matching, the listStatus() and globStatus() methods of FileSystem takes an optional PathFilter

```
package org.apache.hadoop.fs;
public interface PathFilter {
boolean accept(Path path);
}
```

- PathFilter is similar to java.io.FileFilter for Path objects instead of File objects. The filter passes only those files which don't match the regular expression. After the glob selects an initial set of files to include, the filter is used to refine the results

## 2.3 Developing a MapReduce Application

- MapReduce is a programming model for building applications which can process Big Data in parallel on multiple nodes. It provides analytical abilities for analysis of large amount of complex data.
- Traditional model is not suitable to process large amount of data and cannot be incorporated by standard database servers. Google solves this problem using MapReduce algorithm.
- MapReduce is a Distributed Data Processing Algorithm, introduced by Google. It is influenced by Functional Programming Model. In cluster environment, MapReduce

algorithm is used to process large volume of data efficiently, reliably and parallel. It uses Divide and Conquer approach to process large volume of data. It divides input task into manageable sub-tasks to execute parallel.

### 2.3.1 How MapReduce Works ?

**GTU : Winter-16, 17, 18, Summer-17, 18, 19**

- MapReduce Algorithm uses the following three main steps :
  - o Map function
  - o Shuffle function
  - o Reduce function

#### Map Function

- The first step in MapReduce algorithm is Map Function. It takes a data set as a input and convert it into another dataset, where individual element is broken down into key-value pairs. Then performs computation on it parallel.
- Map function performs following steps :
  - o Splitting
  - o Mapping
- Splitting step takes input Data Set from Source and breaks up into smaller Sub-Data Sets.
- Mapping step takes these Sub-Data Sets and performs required computation on each Sub-Data Set.
- The output of this Map Function is a set of key and value pairs as <Key, Value>.

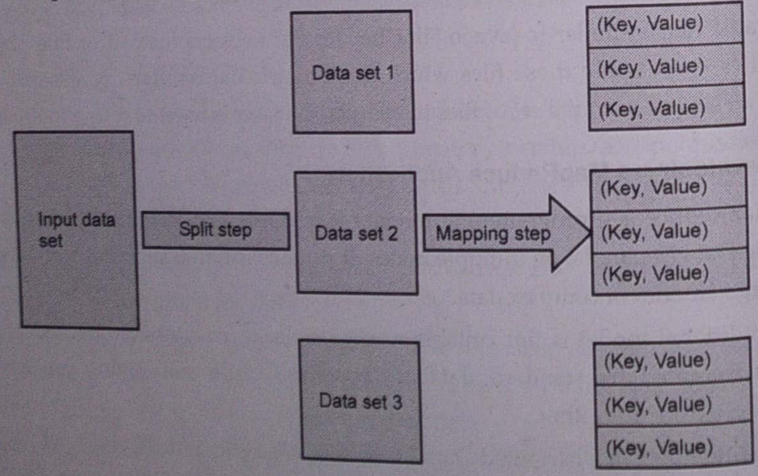


Fig. 2.3.1 MapReduce mapping function

**Output of first step of MapReduce :**

Map Function Output = List of <Key, Value> Pairs

#### Shuffle Function

- Shuffle function is second step of MapReduce algorithm. It is also called as Combine Function.
- It performs following sub-steps :
  - o Merging
  - o Sorting
- It takes output of Map Function as input and perform above sub-steps on every key-value pair.
- Merging step merges all key-value pair that have same keys. It gives <Key, List<Value>> pair as output.
- Output of Merging step is given as input to Sorting step which sorts all key-value pair by using keys. It also returns <Key, List<Value>> with sorted key-value pair as output.
- Shuffle Function gives sorted <Key, List<Value>> pairs to next step.

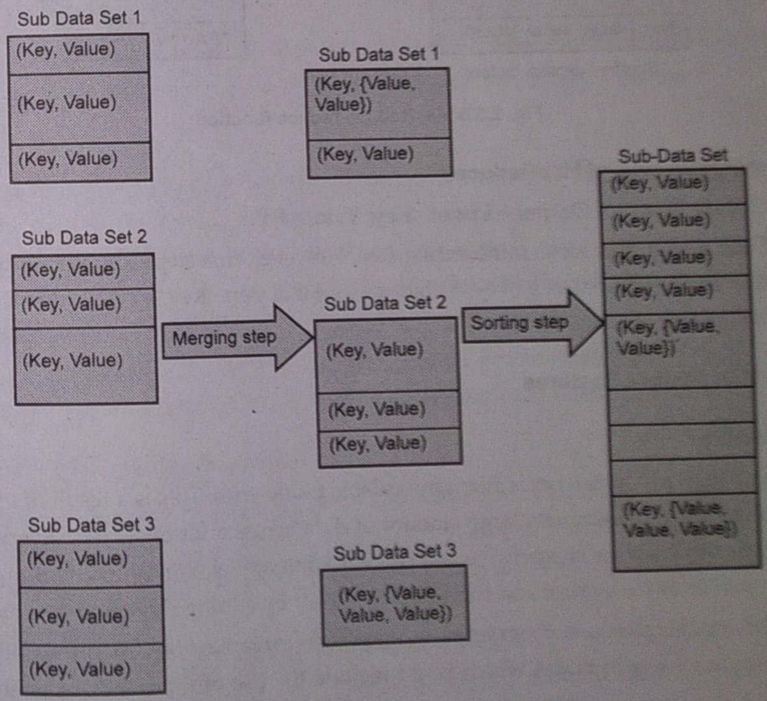


Fig. 2.3.2 MapReduce - shuffle function

**Output of second step of MapReduce :**

Shuffle Function Output = List of <Key, List<Value>> Pairs

**Reduce Function**

- It is last step of MapReduce algorithm. It performs one step: Reduce Step. It tak list of sorted <Key, List<Value>> pairs as input from Shuffle Function and perform reduce operation.

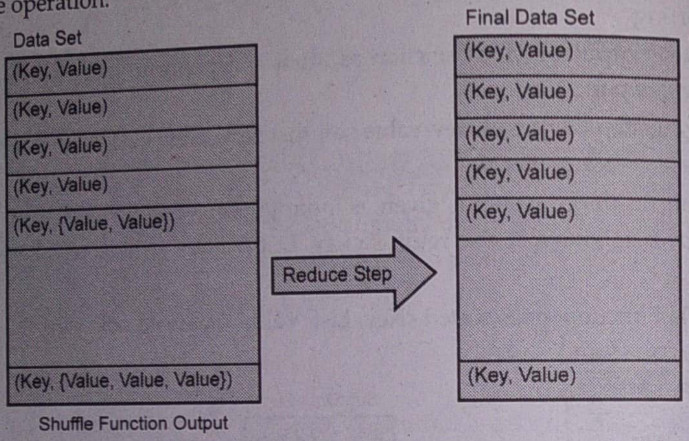


**Fig. 2.3.3 MapReduce reduce function**

**Output of final step of MapReduce :**

- Reduce Function Output = List of <Key, Value> Pairs
- Final step output looks same as first step. However, first step <Key, Value> pairs are different from final step <key, Value> pairs. Final step <Key, Value> pairs are sorted and computed

## 2.3.2 Map Reduce Features

**1. Scalability**

- Apache Hadoop is an exceptionally scalable framework. This is a result of its ability to store and disseminate large amount of data across a lot of servers. Every one of these servers were inexpensive and can work in parallel. We can without much of a stretch scale the storage and computation power by adding servers to the cluster.
- Hadoop MapReduce programming empowers organizations to run applications from huge sets of nodes which could include the use of thousands of terabytes of data.

**2. Flexibility**

- MapReduce programming empowers organizations to access new sources of data. It empowers organizations to work on various kinds of data. It permits companies to access structured as well as unstructured data, and obtain important value by gaining insights from the numerous sources of data. Furthermore, the MapReduce framework likewise offers help for the numerous languages and data from sources going from email, social media, to clickstream.
- The MapReduce processes data in key-value pair along these lines supports data type including meta-data, pictures, and large files. Consequently, MapReduce is elastic to manage data instead of traditional DBMS.

**3. Security and Authentication**

- The MapReduce programming model uses HBase and HDFS security platform that permits access just to the authenticated users to work on the data. Accordingly, it enhances security of system by restricting unauthorized access to the system data.

**4. Cost-effective solution**

- The scalable architecture of Hadoop along with MapReduce framework permit storage and processing of huge data sets in a very cost-effective manner.

**5. Fast**

- Hadoop uses distributed storage technique known as a Hadoop Distributed File System that essentially implements a mapping system for finding data in a cluster. The tool which are used for data processing, for example, MapReduce programming, are for the most part located on exactly the same servers that permits faster processing of data.
- Hadoop MapReduce requires only minutes to process terabytes of data, even in case we are working with huge amount of unstructured data.

**6. Simple model of programming**

- Among the different features of Hadoop MapReduce, perhaps the main highlights is that it depends on a simple programming model. Fundamentally, this permits developers to build the MapReduce programs which can deal with task effectively and efficiently.
- The MapReduce programs can be written in Java, which isn't extremely difficult to get and is likewise used widely. Thus, anybody can without much of a stretch learn and write MapReduce programs and meet their data processing needs.

**7. Parallel Programming**

- One of the major aspect of the working of MapReduce programming is its parall processing. It divides the task in a way that permits their execution in parallel.
- The parallel processing permits multiple processors to execute these divided task So the entire program gets executed in less time.

**8. Availability and resilient nature**

- At whatever point the data is sent to an individual node, a similar set of data is se to some different nodes in a cluster. Thus, in the event that a specific node fails, that point there are consistently different duplicates present on other nodes that ca in any case be accessed to at any point required. This guarantees high availability data.
- One of the main feature offered by Apache Hadoop is its fault tolerance. Th Hadoop MapReduce framework can quickly identifying fault that happen. It at tha point applies a quick and automatic recovery solution. This feature makes it distinct advantage in the world of big data processing.

## 2.4 Hadoop Environment Setup

### 2.4.1 Cluster Specification

- Hadoop is intended to run on commodity hardware. That implies that you are no attached to costly, exclusive offerings from a single merchant; rather, you can selec standardized, commonly available hardware from any of a large range of sellers t build your cluster.
- "Commodity" doesn't signify "low-end." Low-end machines generally have cheap components, which have higher rate of failure than more costly machines.
- At the point when you are operating tens, hundreds, or thousands of machines cheap component end up being a false economy, as the higher rate of failure cause higher maintenance cost. Then again, huge database class machines are no suggested either, since they don't score well on the value/performance curve. Also despite the fact that you would require less of them to assemble a cluster o equivalent performance to one built of mid-range commodity hardware, when one fails it would have bigger effect on the cluster, since a bigger proportion of th cluster hardware would be unavailable.
- Hardware specifications quickly become out of date, yet for representation, an average decision of machine for running a Hadoop datanode and tasktracker in

mid-2010 would have the following specifications :

Processor : 2 Quad-core 2 - 2.5 GHz CPUs

Memory : 16 - 24 GB ECC RAM

Storage : 4 × 1TB SATA disks

Network : Gigabyte Ethernet

- While the hardware specification for your cluster will certainly be unique, Hadoop is intended to use multiple cores and disks, so it will have the option to take full advantage of more powerful hardware.

### 2.4.2 Cluster Setup and Installation

- A Multi Node Cluster in Hadoop contains at least two DataNodes in a distributed Hadoop environment. This is practically used in organizations to store and analyze their Petabytes and Exabytes of data.
- In this we will conside two machines - master and slave. Datanode will be running on both of these machines.
- Let us start with the setup of Multi Node Cluster in Hadoop.
- **Prerequisites**
  - o Cent OS 6.5
  - o Hadoop-2.7.3
  - o JAVA 8
  - o SSH
- **Follow below mentioned steps to Setup Multi Node Cluster in Hadoop**

We have two systems - master and slave with IP :

Master IP: **192.168.12.121**

Slave IP: **192.168.12.124**

**Step 1 :** Check the IP address of all system.

```
> ifconfig
```

**Step 2 :** Now disable firewall restriction

```
>service iptables stop
>sudo chkconfig iptables off
```

**Step 3 :** Open hosts file on both machines to add master and data node.

```
>sudo nano /etc/hosts
```

Same properties will be shown in the host file of master and slave nodes.

Add following lines in host file

192.168.12.121 master

192.168.12.124 slave

**Step 4 :** Restart the sshd service.

```
> service sshd restart
```

**Step 5 :** Create the SSH Key in the master node by using following command

```
>ssh-keygen -t rsa -P ""
```

**Step 6 :** Copy the generated ssh key to master node's authorized keys.

```
> cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

**Step 7 :** Copy the master node's ssh key to slave's authorized keys.

```
> ssh-copy-id -i $HOME/.ssh/id_rsa.pub avi@slave
```

**Step 8 :** Download Java 8 package and save it in home directory.

**Step 9 :** Extract the Java package on all nodes.

```
> tar -xvf jdk-8u101-linux-i586.tar.gz
```

**Step 10 :** Download the Hadoop 2.7.3 Package on all nodes.

```
> wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
```

**Step 11 :** Extract the Hadoop tar File on all nodes.

```
> tar -xvf hadoop-2.7.3.tar.gz
```

**Step 12 :** Add the Java and Hadoop paths in the bash file (.bashrc) on all nodes.

Add Java and Hadoop path as follow.

Open. **bashrc** file.

```
> sudo gedit .bashrc
```

Add following lines.

```
export HADOOP_HOME = $HOME/hadoop-2.7.3
export HADOOP_CONF_DIR = $HOME/hadoop-2.7.3/etc/hadoop
export HADOOP_MAPRED_HOME= $HOME/hadoop-2.7.3
export HADOOP_COMMON_HOME= $HOME/hadoop-2.7.3
export HADOOP_HDFS_HOME= $HOME/hadoop-2.7.3
```

```
export YARN_HOME= $HOME/hadoop-2.7.3
export PATH= $PATH: $HOME/hadoop-2.7.3/bin
export JAVA_HOME = $HOME/jdk1.8.0_101
export PATH = $HOME/jdk1.8.0_101/bin:$PATH
```

After adding above line save and close bash file. Run following command to apply the changes made to current terminal

```
> source .bashrc
```

To ensure whether Java and Hadoop are properly installed, run following command

```
> java -version
> hadoop version
```

Now edit the configuration files in **hadoop-2.7.3/etc/hadoop** directory.

**Step 13 :** In master and slave systems, create master file and edit as shown below :

```
> sudo gedit masters
```

Add following content and save the file.

master

**Step 14 :** Edit slaves file in master machine as shown below :

```
> sudo gedit /home/avi/hadoop-2.7.3/etc/hadoop/slaves
```

Add following content and save the file.

master

slave

**Step 15 :** Edit slaves file in slave machine as follows :

```
> sudo gedit /home/avi/hadoop-2.7.3/etc/hadoop/slaves
```

Add following content and save the file.

slave

**Step 16 :** Edit core-site.xml on both master and slave machines as follows :

```
> sudo gedit /home/avi/hadoop-2.7.3/etc/hadoop/core-site.xml
```

Add following lines.

```
<?xml version="1.0? encoding="UTF-8??>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://master:9000</value>
</property>
</configuration>
```

**Step 17 :** Edit hdfs-site.xml on master as follows :

```
> sudo gedit /home/avi/hadoop-2.7.3/etc/hadoop/hdfs-site.xml
```

Add following lines

```
<?xml version="1.0? encoding="UTF-8??>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>/home/hadoop/hadoop-2.7.3/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/home/hadoop/hadoop-2.7.3/datanode</value>
</property>
</configuration>
```

**Step 18 :** Edit hdfs-site.xml on slave machine as follows :

```
> sudo gedit /home/avi/hadoop-2.7.3/etc/hadoop/hdfs-site.xml
```

Add following lines

```
<?xml version="1.0? encoding="UTF-8??>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/home/hadoop/hadoop-2.7.3/datanode</value>
</property>
</configuration>
```

**Step 19 :** From the template in configuration folder copy mapred-site and on both master and slave edit maored-site.xml as follows :

```
> cp mapred-site.xml.template mapred-site.xml
> sudo gedit /home/avi/hadoop-2.7.3/etc/hadoop/mapred-site.xml
```

Add following lines

```
<?xml version="1.0? encoding="UTF-8??>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

**Step 20 :** Edit yarn-site.xml on both master and slave machines as follows :

```
> sudo gedit /home/avi/hadoop-2.7.3/etc/hadoop/yarn-site.xml
<?xml version="1.0? encoding="UTF-8??>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

**Step 21 :** Format the namenode ,only on master system

```
> hadoop namenode -format
```

**Step 22 :** Start all daemons, only on master machine.

```
> ./sbin/start-all.sh
```

**Step 23 :** Check if all the daemons are running on both master and slave machines.

```
> jps
```

- Now open browser and go to master:50070/dfshealth.html on master system, which will show NameNode interface. If number of live nodes are 2, then we have successfully setup multi node cluster.

## 2.5 Hadoop Configuration

- There are a small bunch of files for controlling the setup of a Hadoop installation the most important ones are listed below

i. *hadoop-env.sh*

Environment variables which are used in the scripts to run Hadoop

ii. *mapred-env.sh*

Environment variables which are used in the scripts to run MapReduce (overrides variables set in hadoop-env.sh)

iii. *yarn-env.sh*

Environment variables which are used in the scripts to run YARN

(overrides variables set in hadoop-env.sh)

iv. *core-site.xml*

Configuration settings for Hadoop Core, like I/O settings which are common to HDFS, MapReduce, and YARN

v. *hdfs-site.xml*

Configuration settings for HDFS daemons: the namenode, the secondary namenode, and the datanodes

vi. *mapred-site.xml*

Configuration settings for MapReduce daemons: the job history server

vii. *yarn-site.xml*

Configuration settings for YARN daemons : the web app proxy server, the resource manager, and the node managers

viii. *Slaves*

A list of machines that each run a datanode and a node manager

ix. *hadoop-metrics2 .properties*

Properties for controlling how metrics are published in Hadoop

x. *log4j.properties*

Properties for system logfiles, the namenode audit log, and the task log for the task JVM process

xi. *hadoop-policy.xml*

Configuration settings for access control lists when running Hadoop in secure mode

- All the files mentioned above are located in the etc/hadoop directory of hadoop distribution.

## 2.6 Security in Hadoop

- Early versions of Hadoop considered that HDFS and MapReduce clusters would be used by a group of cooperating users inside a protected environment. The measures for restricting access were intended to prevent accidental data loss, as opposed to prevent unauthorized access to data. For instance, the file authorizations system in HDFS keeps one client from accidentally clearing out the entire filesystem in view of a bug in a program, or by mistakenly composing hadoop fs -rmr /, however it doesn't keep a malicious client from accepting that root's character to get to or erase any data in the cluster.
- In security language, what was missing was a protected confirmation mechanism to guarantee Hadoop that the user trying to perform a procedure on the cluster is who he professes to be and hence can be trusted. HDFS file permissions give just an mechanism for approval, which controls what a specific user can do to a specific document. For instance, a file might be readable simply by a specific group of users, so anybody not in that group isn't authorized to read it. Nonetheless, approval isn't sufficient by itself, on the grounds that the system is as yet open to exploit by means of spoofing by a malicious user who can gain network access to the cluster.
- It's common to restrict access to data which contains personally recognizable data like an end user's complete name or IP address, to a small group of users inside the organization who are authorized to access such information. Less sensitive data might be made accessible to a bigger group of users. It is convenient to have a mix of datasets with various security levels on the same cluster. Nonetheless, to meet administrative necessities for data protection, secure authentication should be set up for shared clusters.
- This is the circumstance that Yahoo! faced in 2009, which drove a group of engineers there to implement secure authentication for Hadoop. In their design, Hadoop itself doesn't oversee user credentials; all things considered, it depends on Kerberos, a mature open-source network authentication protocol, to authenticate the user. Be that as it may, Kerberos doesn't oversee permissions. Kerberos says that a user is who she says she is; Hadoop must decide if that user has authorization to perform a given action.

### 2.6.1 Kerberos and Hadoop

- At high level, there are three stages that a client should take to access a service when using Kerberos, every one of which includes a message exchange with server :

1. *Authentication* : The client verifies itself to the Authentication Server and gets a timestamped Ticket-Granting Ticket (TGT).
2. *Authorization* : The client uses the TGT to request a service ticket from the Ticket-Granting Server.
3. *Service request* : The client uses the service ticket to confirm itself to the server that is offering the service the client is using. On account of Hadoop, this may be the namenode or the resource manager.

- Together, the Authentication Server and the Ticket Granting Server form the Key Distribution Center (KDC).
- The authorization and service request steps are not client level activities; the clients performs these steps on user's behalf. The authentication step, however, is typically done explicitly by the client using the kinit command, which will prompt for a password. However, this doesn't mean you need to enter your password each time you run a job or access HDFS, since TGTs keep going for 10 hours of course and can be renewed for as long as seven days. It's entirely expected to automate authentication at operating system login time, in this manner giving single sign-on to Hadoop.
- In situations where you would prefer not to be prompted for password , you can make a Kerberos keytab documentby making use of the ktutil command. A keytab is a document that stores passwords and might be provided to kinit with the - t option.

### 2.6.2 Other Security Improvement

- Security has been strengthen all through the Hadoop stack to protect against unauthorized access to resources. The more eminent features are listed as follow :
  - o Tasks can be run by using operating system account for the user who have submitted the task, as opposed to the user running the node manager. This implies that the operating system is used to isolate running jobs, so they can't transfer signals to one another, thus local data, for example, task data, is kept hidden through local filesystem consents. To enable this feature administrators need to make sure that each user is given an account on each node in the cluster.

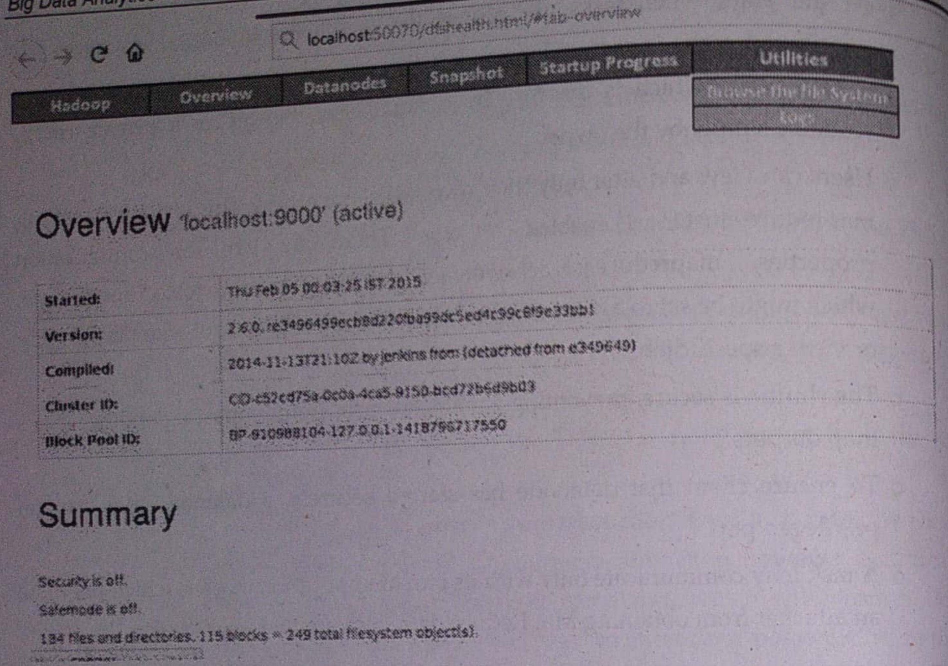
  - o At the point when tasks are run as the user who submitted the task, the distributed cache is secure. Documents that are world-readable are placed in a shared cache which is insecure by default; else, they go in a private cache, readable simply by the owner.
  - o Users can view and alter only their own tasks, not others. This is done by setting mapreduce.cluster.acls.enabled to true. There are two job configuration properties, mapreduce.job.acl--view-job and mapreduce.job.acl-modify-job, which might be set to a comma separated list of users to control who may modify or view a specific job.
  - o The shuffle is secure, preventing a malicious user from requesting another user's map outputs.
  - o To ensure client that datanode has started securely, a datanode may run on privileged port.
  - o A task may communicate only with its parent application master, which prevents an attacker from obtaining MapReduce data from another user's task.

## 2.7 Administering Hadoop

- Hadoop administration includes MapReduce and HDFS administration.
  - o MapReduce administration : It includes monitoring the list of applications, configuration of nodes, application status, etc.
  - o HDFS administration : It includes monitoring the HDFS file structure, locations, and the updated files.

### 2.7.1 HDFS Administration

- HDFS (Hadoop Distributed File System) contains the user directories, input files, and output files. For storing and retrieving make use of MapReduce commands, put and get.
- Once the Hadoop daemons are started by passing the command "start-all.sh" on "/$HADOOP_HOME/sbin", enter the URL "http://localhost:50070" in the browser. Following screen will be seen showing how to browse HDFS.



Overview 'localhost:9000' (active)

| | |
|---|---|
| Started: | Thu Feb 05 00:03:25 IST 2015 |
| Version: | 2.6.0, re3496499ecb8d220fba99dc5ed4c99c8f9e33bb1 |
| Compiled: | 2014-11-13T21:10Z by jenkins from (detached from e349649) |
| Cluster ID: | CID-c52cd75a-0c0a-4ca5-9150-bcd72b6d9b03 |
| Block Pool ID: | BP-910988104-127.0.0.1-1418796717550 |

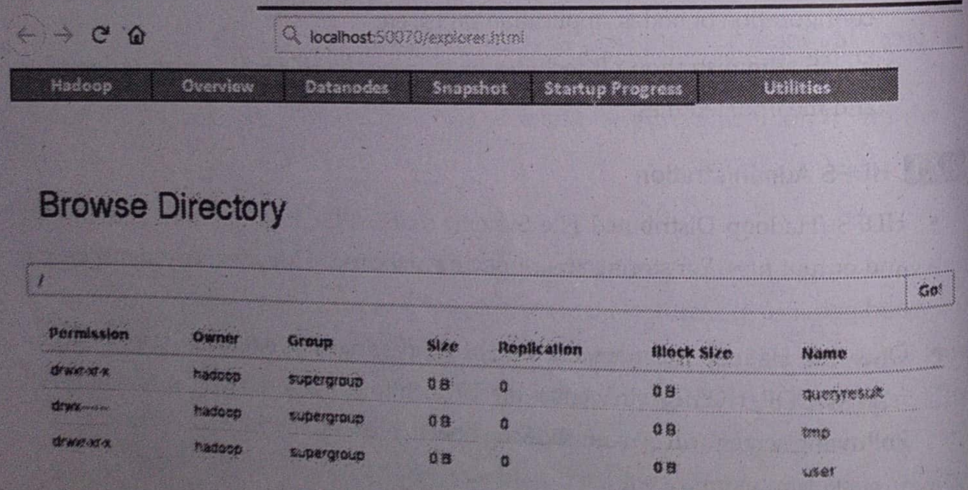
Summary

Security is off.

Safemode is off.

134 files and directories, 115 blocks = 249 total filesystem object(s).

- File structure of HDFS is shown below, displaying list of file in "/user/hadoop" directory.

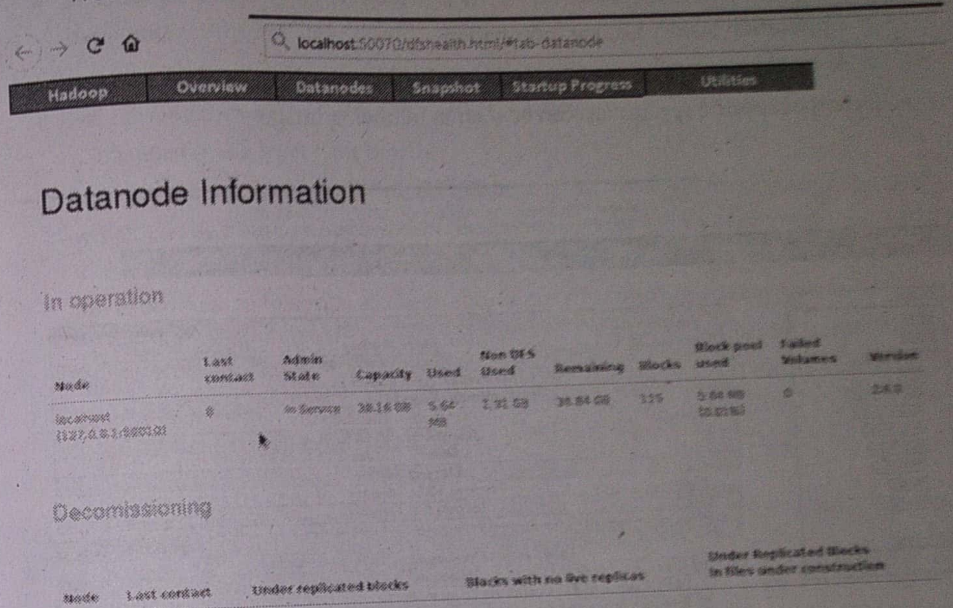


Browse Directory

/

| Permission | Owner | Group | Size | Replication | Block Size | Name |
|---|---|---|---|---|---|---|
| drwxr-xr-x | hadoop | supergroup | 0 B | 0 | 0 B | queryresult |
| drwx------ | hadoop | supergroup | 0 B | 0 | 0 B | tmp |
| drwxr-xr-x | hadoop | supergroup | 0 B | 0 | 0 B | user |

- Data node with its configuration and capacities can be seen in Datanodes section on WebUI as follows.



Datanode Information

In operation

| Node | Last contact | Admin State | Capacity | Used | Non DFS Used | Remaining | Blocks | Block pool used | Failed Volumes | Version |
|---|---|---|---|---|---|---|---|---|---|---|
| localhost (127.0.0.1:50010) | 0 | In Service | 38.16 GB | 5.64 MB | 1.31 GB | 36.84 GB | 115 | 5.64 MB (0.01%) | 0 | 2.6.0 |

Decomissioning

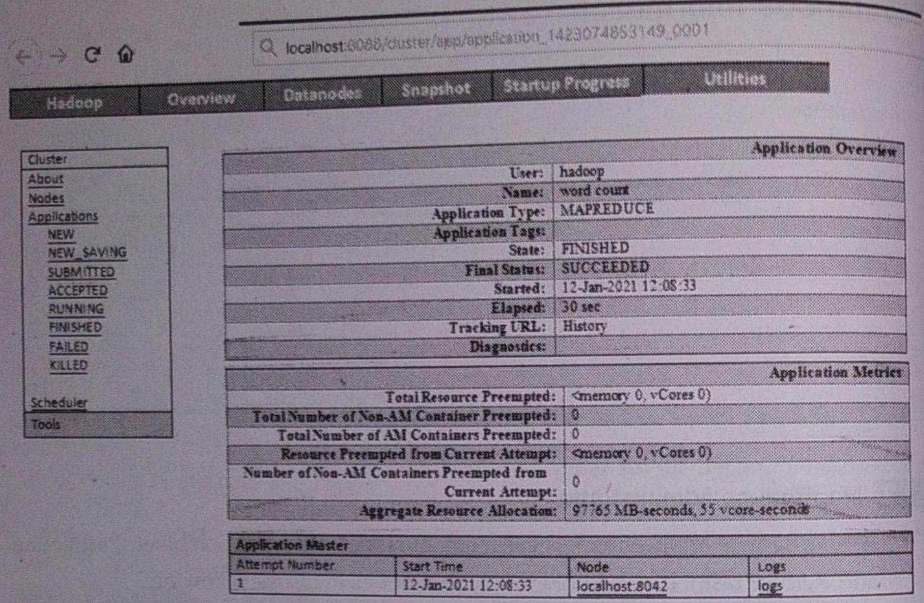
| Node | Last contact | Under replicated blocks | Blocks with no live replicas | Under Replicated Blocks In files under construction |
|---|---|---|---|---|

## 2.7.2 MapReduce Administration

- A MapReduce application is a collection of tasks (Map task, Combiner, Partitioner, and Reduce task). It is required to monitor and maintain the following ?
  - o Datanode configuration where the application is suitable.
  - o The number of datanodes and resources used per application.
- To monitor above mentioned things, it is essential to have user interface.
- Once the Hadoop daemons are started by passing the command "start-all.sh" on "/$HADOOP_HOME/sbin", enter the URL "http://localhost:8080" in the browser.
- After clicking on Application ID, it will show the screen on browser containing following details.
  - o On which user the current application is running
  - o The application name
  - o Application type
  - o Current and Final status

- o Application started time, in case it complete at the time of monitoring then completed time will be shown
- o Application history i.e., log information
- o And at last, information of the node involved in executing the application

- Details of specific application can be seen in following image.



Application Overview

| | |
|---|---|
| User: | hadoop |
| Name: | word count |
| Application Type: | MAPREDUCE |
| Application Tags: | |
| State: | FINISHED |
| Final Status: | SUCCEEDED |
| Started: | 12-Jan-2021 12:08:33 |
| Elapsed: | 30 sec |
| Tracking URL: | History |
| Diagnostics: | |

Application Metrics

| | |
|---|---|
| Total Resource Preempted: | <memory 0, vCores 0> |
| Total Number of Non-AM Container Preempted: | 0 |
| Total Number of AM Containers Preempted: | 0 |
| Resource Preempted from Current Attempt: | <memory 0, vCores 0> |
| Number of Non-AM Containers Preempted from Current Attempt: | 0 |
| Aggregate Resource Allocation: | 97765 MB-seconds, 55 vcore-seconds |

Application Master

| Attempt Number | Start Time | Node | Logs |
|---|---|---|---|
| 1 | 12-Jan-2021 12:08:33 | localhost:8042 | logs |

## 2.8 Monitoring

- Monitoring is an important part of system administration. The reason behind monitoring is to detect when the cluster is not giving the required level of service. The master daemons are the most important to monitor: the namenodes (primary and secondary) and the resource manager. Datanodes and node managers Failure is to be expected, especially on bigger clusters, so you should give additional capacity so that the cluster can tolerate having a small percentage of dead nodes at any time.
- In addition to the facilities explained next, some administrators run test jobs on a periodic basis as a trial of the cluster's health.

### 2.8.1 Logging

- Logfiles produces by all Hadoop daemons can be very helpful in knowing what is happening in the system.

*Setting log levels*

- For specific component in the system when debugging is a problem, appropriate is to change the log level temporarily
- Hadoop daemons have a web UI for changing the log level for any log4j log name, which can be found at /logLevel in the daemon's web UI. By convention, log names in Hadoop correspond to the names of the classes doing the logging, although there are exceptions to this rule, so check source code for finding out log names.
- To enable logging for all packages which start with provided prefix is possible. For instance, to enable debug logging for all classes related to the resource manager, we would visit the its web UI at *http://resource-manager-host:8088/logLevel* and set the log name org.apache.hadoop.yarn.server.resourcemanager to level DEBUG.
- The same can be enabled from the command line as follows :

```
% hadoop daemonlog -setlevel resource-manager-host:8088 \
org.apache.hadoop.yarn.server.resourcemanager DEBUG
```

- Log level changed using above mentioned ways will get reset in case daemon restarts. This log level change can be made constant by changing log4j.properties file in configuration directory. Following line need to be added for this :

```
log4j.logger.org.apache.hadoop.yarn.server.resourcemanager=DEBUG
```

*Getting stacktraces*

- Hadoop daemons have a web UI that produces a thread dump for all running threads in the daemon's JVM. For instance, you can get a thread dump for a resource manager from http://resource-manager-host:8088/stacks.

### 2.8.2 Metrics and JMX

- The information gathered by Hadoop daemons about measurement and events are known as metrics. For instance, few metrics captured by datanodes are: the number of bytes written, the number of blocks replicated, and the number of read requests from local and remote clients
- Metrics belong to a context like mapred, dfs, rpc & yarn. Hadoop daemons usually captures metrics under several contexts. For instance, for the "dfs" and "rpc" contexts, datanodes collect metrics
- Since all Hadoop metrics are published to JMX (Java Management Extensions), so

standard JMX tools such as JConsole can be used to view metrics. It can also be viewed by connecting to /jmx web UI. This is easy for debugging. For instance, to view namenode metric, enter http://namenode-host:50070/jmx in browser.

## 2.9 Maintenance

### 2.9.1 Routine Administration Procedures

#### 2.9.1.1 Metadata Backups

- In the event that the namenode's persistent metadata is lost or damaged, the whole filesystem is rendered unusable, so it is important that backups are made of these documents. You should keep numerous duplicates of various ages (hourly, daily, weekly and monthly) to secure against data corruption, either in the duplicates themselves or in the live documents running on the namenode.
- A simple method to make backups is to write a script to periodically archive the secondary namenode's previous.checkpoint subdirectory to an offsite location. The script should moreover test the integrity of the copy. This should be possible by starting a local namenode daemon and verifying that it has effectively read the fsimage and edits records into memory.

#### 2.9.1.2 Data Backups

- Despite the fact that HDFS is intended to store data reliably, data loss can happen, just like in any storage system, and subsequently a backup strategy is essential. With the huge data volumes that Hadoop can store, choosing what data to back up and where to store it is a challenge.
- The key here is to prioritize your data. The highest priority is the data that can't be regenerated and that is important for the business; in any case, data that is straight forward to regenerate, or essentially dispensable because it is of restricted business value, is of least priority, and you may decide not to make backups of this category of data.
- HDFS replications are not substitute for make backups. Defect in HDFS can be a reason replica to be lost, and so can failure in hardware. Despite the fact that Hadoop is explicitly designed so failure of hardware is probably not going to bring about data loss, the chance can never be totally ruled out, especially when combined with software bugs or human error.
- With regards to backups, consider HDFS similarly you would RAID. Despite the

fact that the data will survive the loss of an individual RAID disk, it may not if the RAID regulator fails, or the whole array is damaged.

- It's common to have a policy for user directories in HDFS. For example, they may have space quotas and be backed up nightly. Whatever the policy, make sure your users know what it is, so they know what to expect.
- The distcp device is ideal for making backups to other HDFS clusters or other Hadoop filesystems, since it can duplicate records in parallel. On the other hand, you can utilize an entirely different storage system for backups, using one of the approaches to export data from HDFS explained in "Hadoop Filesystems".

#### 2.9.1.3 Filesystem Check (fsck)

- It is recommended to run HDFS's fsck tool consistently on entire filesystem to proactively search for absent or corrupt blocks.

#### 2.9.1.4 Filesystem Balancer

- To keep the filesystem datanodes evenly balanced, execute the balancer tool on regular basis.

### 2.9.2 Commissioning and Decommissioning Nodes

- As an administrator of a Hadoop cluster, you should add or remove nodes every now and then. For instance, to develop the storage available to a cluster, you commission new nodes. Alternately, some of the time you may wish to shrink a cluster, and to do so, you decommission nodes. It can in some cases be important to decommission a node in the event that it is misbehaving, maybe on the grounds that it is failing more regularly than it should or its performanceis noticeably slow.
- Nodes normally run both a datanode and a tasktracker, and both are normally commissioned or decommissioned in order.

#### 2.9.2.1 Commissioning New Nodes

- In spite of the fact that commissioning a new node can be as simple as configuring the hdfssite. xml file to point to the namenode and the mapred-site.xml file to point to the jobtracker, and starting the datanode and jobtracker daemons, it is for the most part best to have a list of approved nodes.
- It is a potential security danger to permit any machine to connect with the namenode and go about as a datanode, since the machine may access data that it isn't authorized to view. Besides, since such a machine is certainly not a genuine

datanode, it isn't under your control, and may stop at any point of time, causin potential data loss. This situation is a risk even inside a firewall, throug misconfiguration, so datanodes and tasktrackers should be explicitly overseen o all production clusters.

- Datanodes that are allowed to connect with the namenode are defined in a fi whose name is specified by the dfs.hosts property. The file resides on th namenode's local filesystem, and it contains a line for each datanode, specified b network address. In case you want to specify different network addresses for datanode, put them on one line, separated by whitespace.
- Likewise, tasktrackers that may connect with the jobtracker are specified in a fi whose name is specified by the mapred.hosts property. In most cases, there is on shared record, referred to as the include file, that both dfs.hosts and mapred.host refer to, since nodes in the cluster run both datanode and tasktracker daemons.
- The file specified by the dfs.hosts and mapred.hosts properties is different from th slaves files. The previous is utilized by the namenode and jobtracker to figure ou which worker node may connect. The slavesfile is used by the Hadoop contro scripts to perform cluster wide activities, for example, cluster restarts. It is neve used by the Hadoop daemons.
- To add new nodes to the cluster :

1. Add the network addresses of the new nodes to the include file.
2. Update the namenode with the new set of allowed datanodes using thi command :

```
% hadoop dfsadmin -refreshNodes
```

3. Update the slaves file with the new nodes, so that they are included in futur activities performed by the Hadoop control scripts.
4. Start the new datanodes.
5. Restart the MapReduce cluster.
6. Check that the new datanodes and tasktrackers show up in the web UI.

- HDFS won't move blocks from old datanodes to new datanodes to balance th cluster. To do this, you should execute the balancer defined in "balancer".

### 2.9.2.2 Decommissioning Old Nodes

- Despite the fact that HDFS is designed to tolerate failure of datanode. With replication level of three, for instance, the odds are extremely high that you will los

data by simultaneously shutting down three datanodes in case they are on separate racks. The best approach to decommission datanodes is to inform the namenode of the nodes that you wish to remove from circulation, so that it can replicate the blocks to other datanodes before the datanodes are shut down.

- In case you shut down a tasktracker which was executing taska, then the jobtracker will observe this failure and reschedule the tasks running on that tasktracker to other tasktracker.
- The decommissioning process is constrained by an exclude file, which for HDFS is set by the dfs.hosts.exclude property and for MapReduce by the mapred.hosts.exclude property. It is often the situation that these properties refer to a similar file. The exclude file lists the nodes which are not allowed to connect to the cluster.
- The standards for whether a tasktracker may connect with the jobtracker are simple: a tasktracker may connect just in case that it shows up in the include file and doesn't show up in the exclude file. An unspecified or void include file is considered as all nodes are in the include file.
- For HDFS, the rules are slightly different. Following table summarizes the different combinations for datanodes.

| Nodes in include file | Nodes in exclude file | Explanation |
|---|---|---|
| No | No | Node might not connect |
| No | Yes | Node might not connect |
| Yes | No | Node might connect |
| Yes | Yes | Node might connect and will be decommissioned |

- To decommission nodes from the cluster :
  - o Add the network addresses of the nodes to be decommissioned to the exclude file. At this point do not update the include file.
  - o To stop tasktracker on the nodes which are being decommissioned by restarting MapReduce cluster
  - o Update the namenode with the new set of allowed datanodes, by executing below command :

```
% hadoop dfsadmin -refreshNodes
```

- For datanodes being decommissioned, check in web UI whether the admin state has changed to "Decommission in progress".
- All the blocks have been replicated when all the datanodes reports their state as "Decommissioned". Shut down the nodes which are decommissioned.
- Remove the nodes from the include file, and run :

```
% hadoop dfsadmin -refreshNodes
```

- Remove the nodes from the slaves file.

## 2.9.3 Upgrades

- Upgrading an HDFS and MapReduce cluster needs careful planning. The main thought is the HDFS upgrade. In case the layout version of the filesystem has changed, at that point the upgrade will consequently migrate the filesystem data and metadata to a format which is compatible with the new version. As with any procedure which includes data migration, there is a risk of data loss, so you should be certain that both your data and metadata is backed up.
- Some portion of the planning process should incorporate a demo run on a small test cluster with a sample data which you can afford to lose. A demo run will permit you to acclimate yourself with the process, customize it to your specific cluster configuration and toolset, and iron out any tangles prior to running the update technique on a production cluster. A test cluster likewise has the advantage of being accessible to test client upgrades on.

### 2.9.3.1 Version Compatibility

- All prior 1.0 Hadoop components have very inflexible requirement of version compatibility. Only components from the same release are assured to be compatible with one another, which imply the entire system from daemons to clients must be upgraded at the same time, in lockstep. This requires a period of cluster downtime.
- Hadoop version 1.0 guarantees to loosen these necessities so that, for instance, older clients can communicate with recent servers. In later releases, rolling upgrades may be supported, which would permit cluster daemons to be upgraded in stages, so the server would even now be accessible to clients during the upgrade.
- Upgrading a cluster when the layout of filesystem has not changed is quite easy: install the latest versions of HDFS and MapReduce on the cluster and on clients at the same time, shut down the old daemons, update configuration files, then start up the new daemons and switch clients to use the new libraries. This process is reversible, so rolling back an upgrade is also straight forward.

- Couple of final cleanup steps should be performed after each successful upgrade, which are as follow :
  - Remove the old installation and configuration files from the cluster.
  - Fix any deprecation warnings in configuration and your code if any.

### 2.9.3.2 HDFS Data and Metadata Upgrades

- In the event that you use the strategy just described to upgrade to latest version of HDFS and it expects another layout version, at that point the namenode will won't run. A message like the following will appear in its log: "File system image contains an old layout version -16.An upgrade to version -18 is required. Please restart NameNode with -upgrade option."
- The most dependable way of finding out if you need to upgrade the filesystem is by performing a demo on a test cluster.
- HDFS upgrade makes a copy of the previous version's metadata and data. Storage requirements of cluster does not get doubles after upgrade, as the datanodes use hard links to keep two references to a similar block of data. This design makes it simple to roll back to the previous version of the filesystem, in case you need to. You should understand that any changes made to the data on the upgraded system will be lost after the rollback finishes.
- Only the previous version of filesystem can be kept and you cannot roll back several versions. Hence, to perform another upgrade to HDFS data and metadata, you will need to finalize the upgrade by removing previous version. Once the upgrade is finalized, you cannot roll back to previous version.
- In few scenarios, you will have to upgrade to every intermediate releases and this will be mentioned in release not if it is required. In other cases you may opt to skip releases at the time of upgrade.
- You should run a full fsck before and after upgrade, so that you can compare output of both fsck.
- With these fundamentals out of the way, here is the high-level procedure for upgrading a cluster when the filesystem layout needs to be migrated :
  1. Make sure finalize previous upgrade before performing another update.
  2. Kill any orphaned task processes on the tasktrackers and shut down MapReduce.
  3. Shut down HDFS and backup the namenode directories.

4. Install new forms of Hadoop HDFS and MapReduce on the cluster and on clients.
5. Start HDFS with the -upgrade option.
6. Wait until the upgrade is finished.
7. Perform some sanity checks to verify everything is ok on HDFS.
8. Start MapReduce.
9. Roll back or finalize the upgrade (optional).

- While running the upgrade procedure, it is better to remove the Hadoop scripts from your PATH environment variable. It can be convenient to specify two environment variables for the new installation directories; in the following instructions, we have defined OLD_HADOOP_INSTALL and NEW_HADOOP_INSTALL.

### 2.9.3.3 Start the Upgrade

- To perform the upgrade, execute the following command :

```
% $NEW_HADOOP_INSTALL/bin/start-dfs.sh -upgrade
```

- This will upgrade metadata of namenode, placing the older version in new directory created name old:

```
${dfs.name.dir}/new/VERSION
/edits
/fsimage
/fstime
/old/VERSION
/edits
/fsimage
/fstime
```

- Similarly, datanodes upgrade their storage directories, preserving the old copy in a directory called old.

### 2.9.3.4 Wait Until the Upgrade is Complete

- The upgrade is not instant process, but the progress can be checked using dfsadmin :

```
% $NEW_HADOOP_INSTALL/bin/hadoop dfsadmin -upgradeProgress status
```

Upgrade for version -18 has been completed.

Upgrade is not finalized.

- This shows that the upgrade is complete. At this point, you should run some sanity checks on the filesystem . You can put HDFS in safe mode during the sanity checks, so that others cannot make any changes to it.

### 2.9.3.5 Roll Back the Upgrade (Optional)

- In case the upgraded version is not working correctly, you can opt to roll back to previous version. Roll back in only possible if upgrade is not finalized
- A rollback reverts the filesystem state to before the upgrade was performed, so any changes made after the upgrade will be lost.
- First, shut down the new daemons:

```
% $NEW_HADOOP_INSTALL/bin/stop-dfs.sh
```

- Then start up the old version of HDFS with the -rollback option :

```
% $OLD_HADOOP_INSTALL/bin/start-dfs.sh -rollback
```

- This command gets the namenode and datanodes to replace their current storage directories with their previous copies. The filesystem will be returned to its previous state.

### 2.9.3.6 Finalize the Upgrade (Optional)

- If the upgraded version is working correctly as per your requirement, you can finalize the upgrade to remove the older storage directories. Once the upgrade is finalized, you cannot roll back to previous version of filesystem.
- This step is required before performing another upgrade :

```
% $NEW_HADOOP_INSTALL/bin/hadoop dfsadmin -finalizeUpgrade
% $NEW_HADOOP_INSTALL/bin/hadoop dfsadmin -upgradeProgress status
```

There are no upgrades in progress.

HDFS is now fully upgraded to the new version.

## 2.10 Hadoop Benchmark

- Hadoop comes with various benchmarks that you can run effectively with least setup cost. Benchmarks are bundled in the demo JAR document, and you can get a list of them, with explanation, by using the JAR file without any arguments :

```
% hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-*-demo.jar
```

- Most of the benchmarks show usage instructions when invoked with no arguments. For instance :

```
% hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-*-demo.jar
\ TestDFSIO
TestDFSIO.1.7
```

Missing arguments.

```
Usage: TestDFSIO [genericOptions] -read [-random | -backward |
-skip [-skipSize Size]] | -write | -append | -clean [-compression codecClassName]
[-nrFiles N] [-size Size[B|KB|MB|GB|TB]] [-resFile resultFileName]
[-bufferSize Bytes] [-rootDir]
```

### Benchmarking MapReduce with TeraSort

- Hadoop accompanies a MapReduce program called TeraSort which does a complete sort of its input. It is exceptionally valuable for benchmarking HDFS and MapReduce together, as the full input dataset is moved through the shuffle. The three stages are: generate random data, perform the sort, then validate the outcomes.
- First, random data is generated using teragen. It executes map-only task which generate a defined number of rows of binary data. Every row is 100 byte long, so in order to generate 1 terabyte of data by using 1,000 maps, execute the following command (10t is abbreviation for 10 trillion) :

```
% hadoop jar \
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar \ teragen
Dmapreduce.job.maps=1000 10t random-data
```

- Next, execute terasort :

```
% hadoop jar \
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar \ terasort
random-data sorted-data
```

- The overall execution time of the sort is the metric we are keen on, however it's informational to watch the job's advancement through the web UI, where you can discover what amount of time each phase of the job requires. Changing the boundaries is a helpful exercise. As a last once-over to verify everything is ok, we validate that the data in sorted data is, correctly sorted:

```
% hadoop jar \
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
\ teravalidate sorted-data report
```

- This command is executes a short MapReduce task which performs sequence of checks on the final sorted data to validate whether the sort is accurate. In case there are any errors, that can be seen in report/part-r-00000 output file

## 2.11 Hadoop in the Cloud

- Although many organizations decide to run Hadoop in-house, it is also good choice to run Hadoop in the cloud on leased hardware or as a service. For example, Cloudera provides tools for running Hadoop in a public or private cloud, and Amazon have Elastic MapReduce which is Hadoop cloud service.
- In this section, we will check how to run Hadoop on Amazon EC2.

### 2.11.1 Hadoop on Amazon EC2

- Amazon Elastic Compute Cloud (EC2) is a computing service which permits clients to lease instances/ computers on which they can run their own applications. A client can launch and terminate instances as per their need, paying by hour for instances which are active. The Apache Whirr project gives a set of scripts which make it simple to run Hadoop on EC2 and other cloud suppliers. The scripts permit you to perform such tasks as launching or terminating a cluster, or adding instances to an existing cluster.
- For specific workflows, running Hadoop on EC2 is very suitable. For instance, In case data is stored on Amazon S3 bucket, then you can run cluster on EC2 and execute MapReduce tasks which will read data from S3 and write final result back to S3, before the cluster shuts down.
- In case working with longer-lived cluster, S3 data can be copied to HDFS which is runing on EC2 for making processing more efficient, as HDFS can take benefit of data locality.

#### 2.11.1.1 Setup

- First, you will need to setup an account, install CE2 command line tools and launch new instance. After doing this, install Whirr, then configure scripts to set AWS credentials, security key details and size & type of EC2 instance to use.

#### 2.11.1.2 Launching a Cluster

- Now we are ready to launch cluster. To launch a cluster name demo-cluster having one master node and five worker nodes enter the following command :

```
% hadoop-ec2 launch-cluster demo-cluster 5
```

- In case security group does not exist, it will create EC2 security group for the cluster we have launched. And also provide master and worker node with unrestricted access to one another. It will also enable SSH access so that these instances can be accessed from anywhere. Once the setup of security group is done, the master instance will get launched, once master instance is started, five worker instances will be launched.
- The purpose to launch worker node separately is so that the hostname of master can be passed to worker instances, and permit the tasktrackers and datanodes to connect to master once they start up.

- To use the cluster, network traffic from the client require to be proxied through the master node of the cluster using an SSH tunnel, which can be done by using following command :

```
% eval 'hadoop-ec2 proxy demo-cluster'  Proxy pid 21763
```

### 2.11.1.3 Running a MapReduce Job

- MapReduce task can be executed from an external system or within the cluster. Here we show how to execute a task from the machine we launched the cluster on. Note that this need to have the same version of Hadoop has been installed locally as is running on the cluster.
- When we launched the cluster, a hadoop-site.xml file was created in the directory ~/.hadoop-cloud/demo-cluster. This can be used in order to connect to the cluster by defining environment variable HADOOP_CONF_DIR as follow :

```
% export HADOOP_CONF_DIR=~/.hadoop-cloud/demo-cluster
```

- The cluster's filesystem is empty, so before we execute a task, we require to populate it with data. Using Hadoop's distcp tool to transfer data from S3 to HDFS is an effective way.

```
% hadoop distcp s3n://hadoopbook/ncdc/all input/ncdc/all
```

- After the data has been transferred, we can execute a task in the normal way :

```
% hadoop jar job.jar MaxTemperatureWithCombiner input/ncdc/all output
```

- On the other hand, we could have defined S3 as input that would have the same impact. When executing number of tasks over the same input data, it's best to copy the data to HDFS first to save bandwidth :

```
% hadoop jar job.jar MaxTemperatureWithCombiner s3n://hadoopbook/ncdc/all output
```

- By using jobtrackers web UI by entering "http://master_host:50030/ in browser, we can track progress of the task. We will require to setup a proxy auto config in order to access web UI running on worker node.

### 2.11.1.4 Terminating a Cluster

- To shut down the cluster, enter the terminate-cluster command as follow :

```
% hadoop-ec2 terminate-cluster demo-cluster
```

- It will ask whether to terminate all the instances of cluster. At the end, stop the proxy process :

```
% kill $HADOOP_CLOUD_PROXY_PID
```

## University Questions with Answers

**Winter - 2016 (IT)**

**Q.1** Explain Hadoop architecture and its components with proper diagram. **(Refer section 2.2)** [7]

**Q.2** Write a Map Reduce code for counting occurrences of specific words in the input text file(s). Also write the command to compile and run the code. [7]

**Ans. :** **MapReduce code for Word Count**

**Steps to Compile & Run MapReduce Application in HortonWorks Sandbox**

**1) Create following files in Eclipse IDE**

**WordCount.java**

```
importorg.apache.hadoop.conf.Configuration;
importorg.apache.hadoop.fs.Path;
importorg.apache.hadoop.io.IntWritable;
importorg.apache.hadoop.io.Text;
importorg.apache.hadoop.mapreduce.Job;
importorg.apache.hadoop.mapreduce.lib.input.FileInputFormat;
importorg.apache.hadoop.mapreduce.lib.input.TextInputFormat;
importorg.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
importorg.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class WordCount {
public static void main(String[] args) throws Exception {
if (args.length != 2) {
System.out.println("usage: [input] [output]");
System.exit(-1);
   }

Job job = Job.getInstance(new Configuration());
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

job.setMapperClass(WordMapper.class);
job.setReducerClass(SumReducer.class);

job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);

FileInputFormat.setInputPaths(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.setJarByClass(WordCount.class);
```

```
    job.submit();
    }
    }
```

**WordMapper.java**

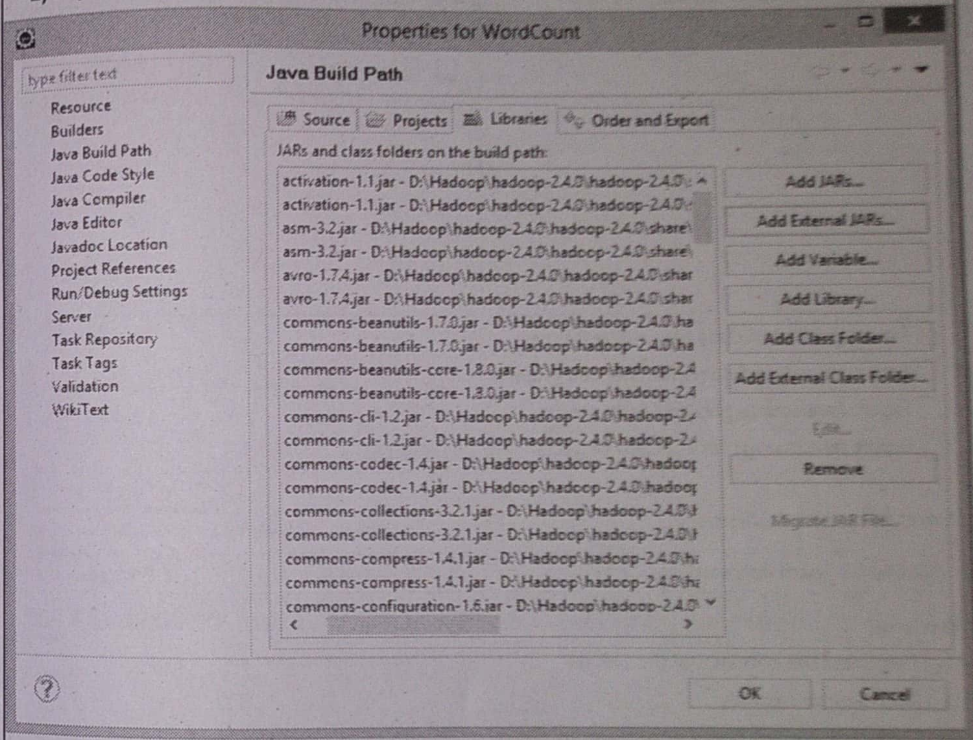
```
    importjava.io.IOException;
    importjava.util.StringTokenizer;

    importorg.apache.hadoop.io.IntWritable;
    importorg.apache.hadoop.io.Text;
    importorg.apache.hadoop.mapreduce.Mapper;
    public class WordMapper extends Mapper<Object, Text, Text, IntWritable> {
    private Text word = new Text();
    private final static IntWritable one = new IntWritable(1);

    public void map(Object key, Text value, Context context_W) throws IOException,
    InterruptedException {
    String line = value.toString();
    line = line.replaceAll("[^a-zA-Z0-9\\s]", "").toLowerCase();
    StringTokenizerwordList = new StringTokenizer(line);
    while (wordList.hasMoreTokens()) {
    word.set(wordList.nextToken());
    context_W.write(word, one);
     }
    }
   }
```

**SumReducer.java**

```
    importjava.io.IOException;
    importjava.util.Iterator;

    importorg.apache.hadoop.io.IntWritable;
    importorg.apache.hadoop.io.Text;
    importorg.apache.hadoop.mapreduce.Reducer;

    public class SumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    privateIntWritabletotalWordCount = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context_W)
    throwsIOException, InterruptedException {
    intwordCount = 0;
      Iterator<IntWritable> it=values.iterator();
      while (it.hasNext()) {
```

```
    wordCount += it.next().get();
    }
    totalWordCount.set(wordCount);
    context_W.write(key, totalWordCount);
    }
    }
```

**2) Import the required jar files from the hadoop of the same version as of hortonworks.**



Properties for WordCount

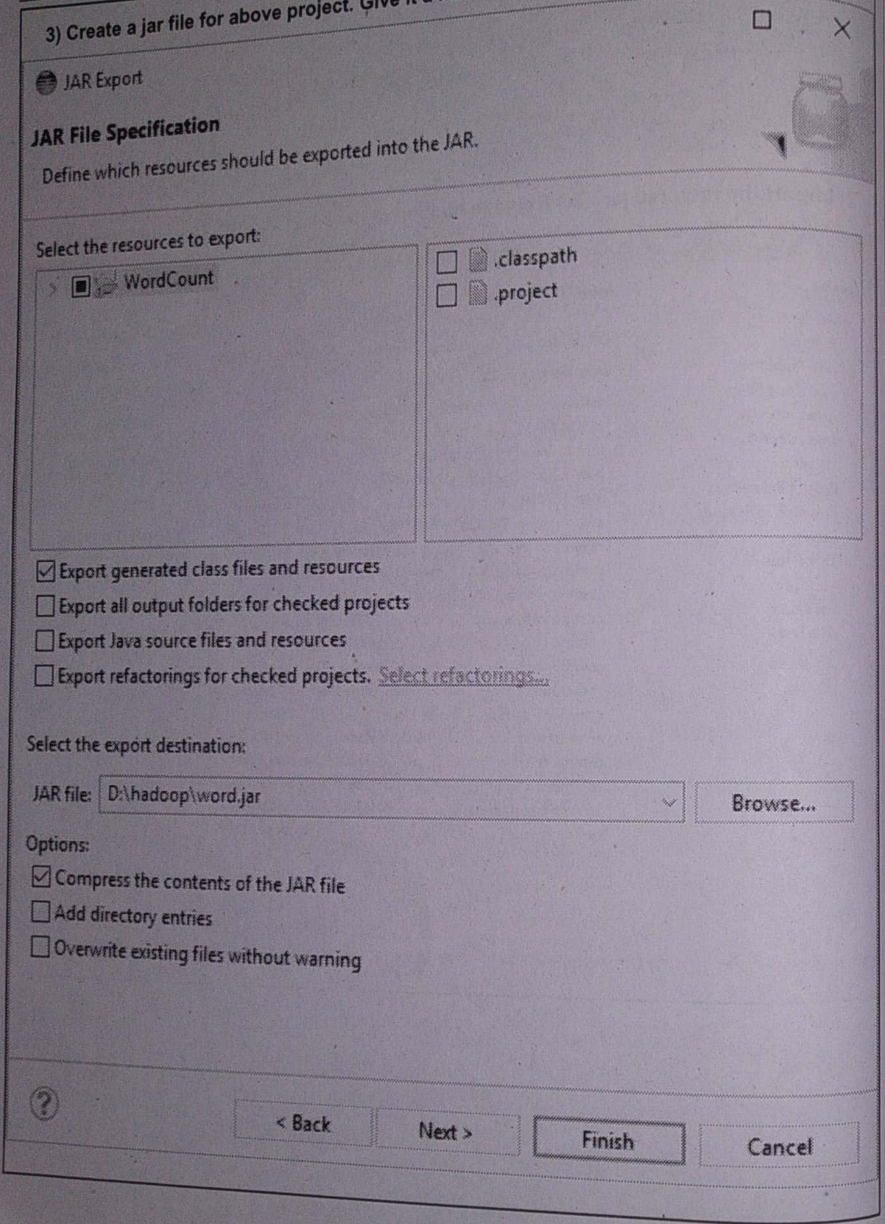
Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

```
activation-1.1.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\
activation-1.1.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\
asm-3.2.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\share\
asm-3.2.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\share\
avro-1.7.4.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\shar
avro-1.7.4.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\shar
commons-beanutils-1.7.0.jar - D:\Hadoop\hadoop-2.4.0\ha
commons-beanutils-1.7.0.jar - D:\Hadoop\hadoop-2.4.0\ha
commons-beanutils-core-1.8.0.jar - D:\Hadoop\hadoop-2.4
commons-beanutils-core-1.8.0.jar - D:\Hadoop\hadoop-2.4
commons-cli-1.2.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.
commons-cli-1.2.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.
commons-codec-1.4.jar - D:\Hadoop\hadoop-2.4.0\hadoop
commons-codec-1.4.jar - D:\Hadoop\hadoop-2.4.0\hadoop
commons-collections-3.2.1.jar - D:\Hadoop\hadoop-2.4.0\
commons-collections-3.2.1.jar - D:\Hadoop\hadoop-2.4.0\
commons-compress-1.4.1.jar - D:\Hadoop\hadoop-2.4.0\h
commons-compress-1.4.1.jar - D:\Hadoop\hadoop-2.4.0\h
commons-configuration-1.6.jar - D:\Hadoop\hadoop-2.4.0\
```

3) Create a jar file for above project. Give it a name "word.jar"



JAR Export

**JAR File Specification**

Define which resources should be exported into the JAR.

Select the resources to export:

- [■] WordCount
- [ ] .classpath
- [ ] .project

- [✓] Export generated class files and resources
- [ ] Export all output folders for checked projects
- [ ] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

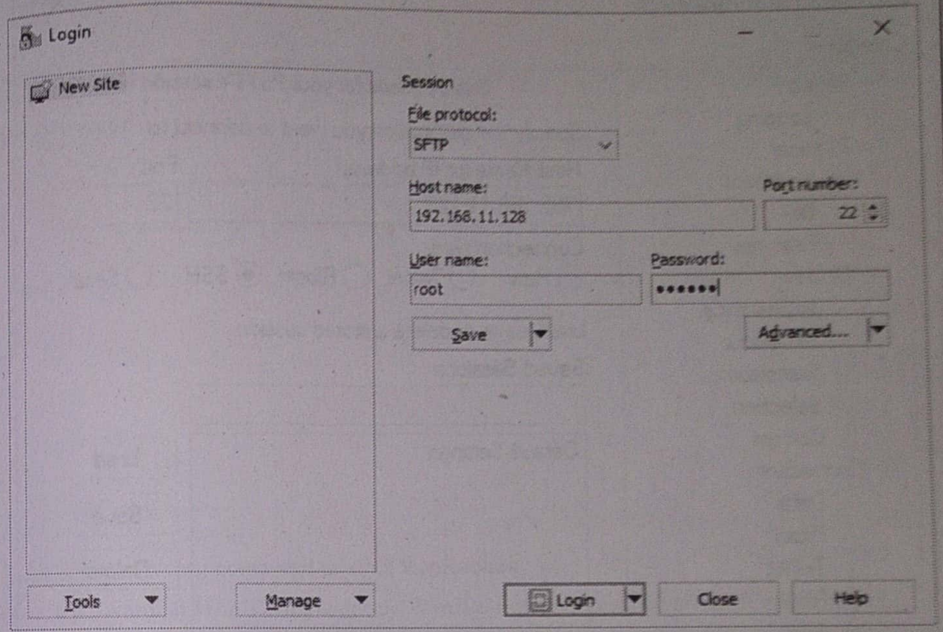
Select the export destination:

JAR file: D:\hadoop\word.jar

Options:

- [✓] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

4) OpenWinSCP, connect it with the IP address of hortonworks. Transfer "word.jar" to "usr/lib/hue" for further use



Login

New Site

Session

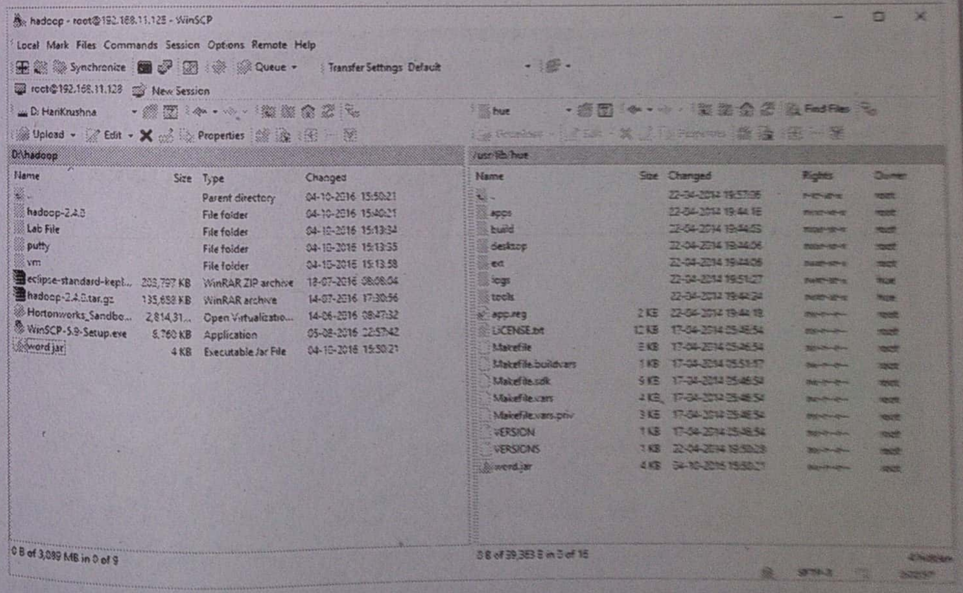
File protocol: SFTP

Host name: 192.168.11.128

Port number: 22

User name: root

Password: ••••••

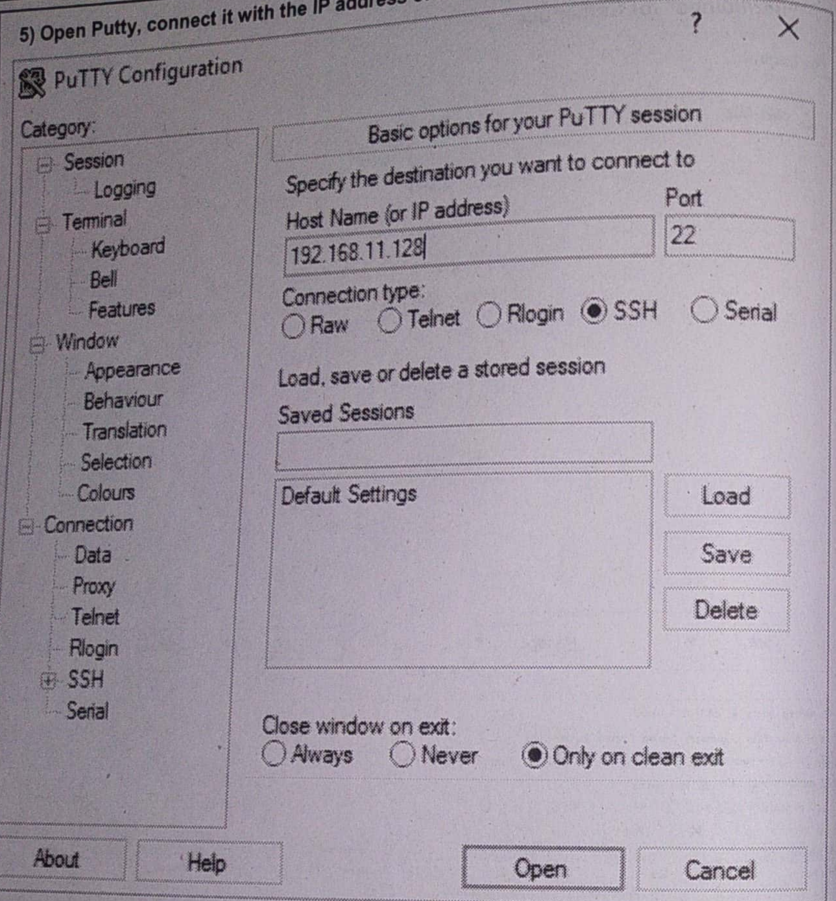


hadoop - root@192.168.11.128 - WinSCP

```
D:\hadoop
Name                          Size        Type                  Changed
..                                        Parent directory      04-10-2016 15:50:21
hadoop-2.4.0                              File folder           04-10-2016 15:40:21
Lab File                                  File folder           04-10-2016 15:13:34
putty                                     File folder           04-10-2016 15:13:35
vm                                        File folder           04-10-2016 15:13:58
eclipse-standard-kepl...      203,797 KB  WinRAR ZIP archive    18-07-2016 08:08:04
hadoop-2.4.0.tar.gz           135,658 KB  WinRAR archive        14-07-2016 17:30:56
Hortonworks_Sandbo...         2,814,31... Open Virtualizatio... 14-06-2016 08:47:32
WinSCP-5.9-Setup.exe          8,760 KB    Application           05-08-2016 22:57:42
word.jar                      4 KB        Executable Jar File   04-10-2016 15:50:21

0 B of 3,089 MB in 0 of 9
```

```
/usr/lib/hue
Name                 Size   Changed               Rights        Owner
..                          22-04-2014 19:57:06   [illegible]   root
apps                        22-04-2014 19:44:18   [illegible]   root
build                       22-04-2014 19:44:53   [illegible]   root
desktop                     22-04-2014 19:44:06   [illegible]   root
ext                         22-04-2014 19:44:06   [illegible]   root
logs                        22-04-2014 19:51:27   [illegible]   hue
tools                       22-04-2014 19:44:24   [illegible]   hue
app.reg              2 KB   22-04-2014 19:44:18   [illegible]   root
LICENSE.txt          12 KB  17-04-2014 05:46:54   [illegible]   root
Makefile             8 KB   17-04-2014 05:46:54   [illegible]   root
Makefile.buildvars   1 KB   17-04-2014 05:51:17   [illegible]   root
Makefile.sdk         5 KB   17-04-2014 05:46:54   [illegible]   root
Makefile.vars        4 KB   17-04-2014 05:46:54   [illegible]   root
Makefile.vars.priv   3 KB   17-04-2014 05:46:54   [illegible]   root
VERSION              1 KB   17-04-2014 05:46:54   [illegible]   root
VERSIONS             1 KB   22-04-2014 19:50:23   [illegible]   root
word.jar             4 KB   04-10-2016 15:50:21   [illegible]   root

0 B of 39,383 B in 0 of 16
```

5) Open Putty, connect it with the IP address of hortonworks.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.11.128

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

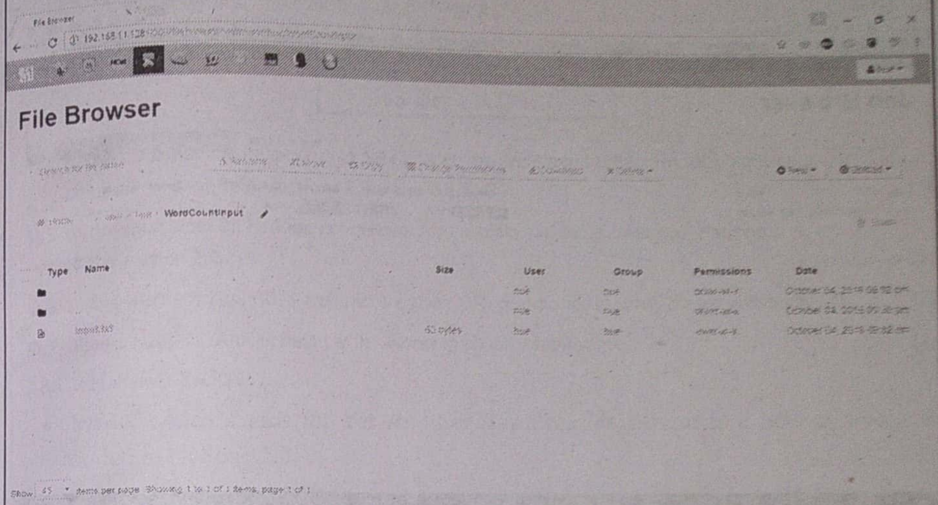
Load, save or delete a stored session

Saved Sessions

Default Settings

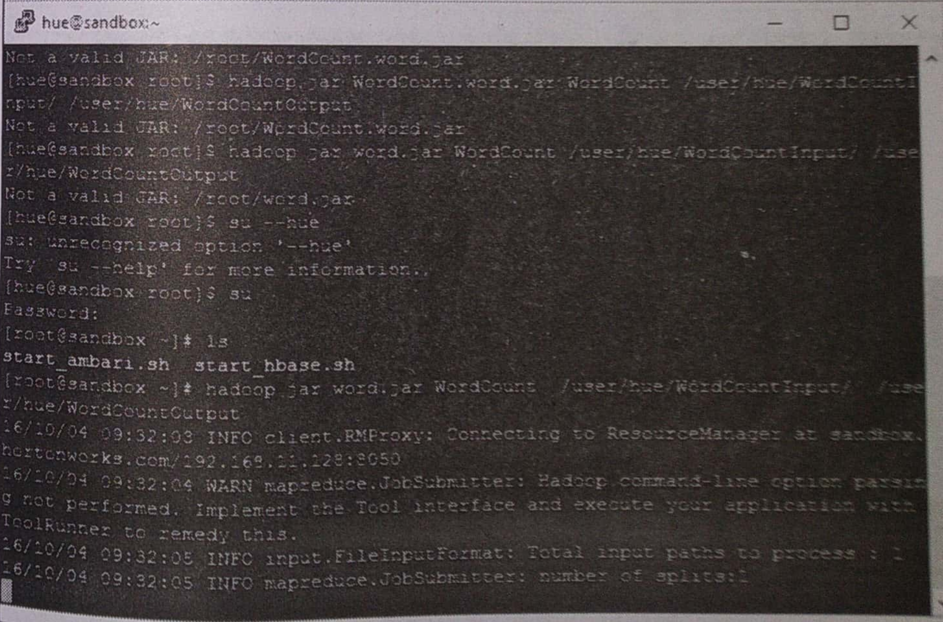
Close window on exit: Always Never [✓ Only on clean exit]

6) Upload input file to HDFS using web UI. Create a directory of name "WordCountInput" and add the input file in this directory.



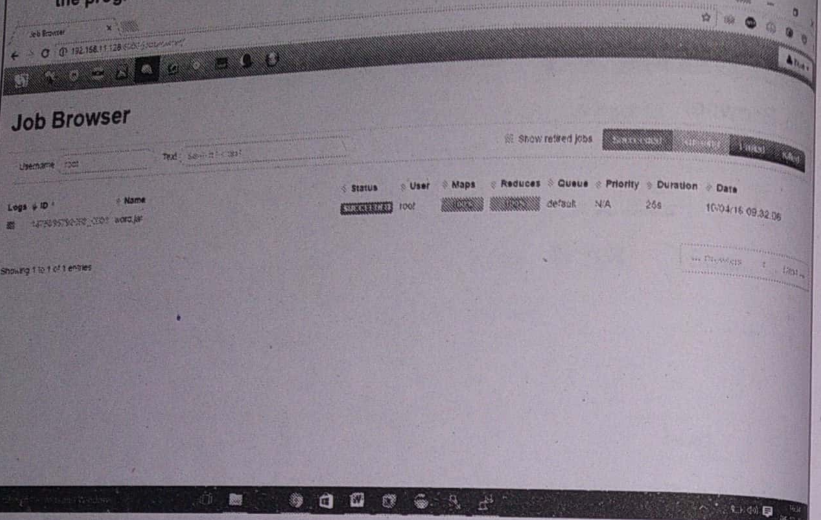
7) Run the following command to start MapReduce

Hadoop jar word.jar WordCount/user/hue/WordCountInput/ /user/hue/WordCountOutput



```
Not a valid JAR: /root/WordCount.word.jar
[hue@sandbox root]$ hadoop jar WordCount.word.jar WordCount /user/hue/WordCountI
nput/ /user/hue/WordCountOutput
Not a valid JAR: /root/WordCount.word.jar
[hue@sandbox root]$ hadoop jar word.jar WordCount /user/hue/WordCountInput/ /use
r/hue/WordCountOutput
Not a valid JAR: /root/word.jar
[hue@sandbox root]$ su --hue
su: unrecognized option '--hue'
Try 'su --help' for more information.
[hue@sandbox root]$ su
Password:
[root@sandbox ~]# ls
start_ambari.sh  start_hbase.sh
[root@sandbox ~]# hadoop jar word.jar WordCount  /user/hue/WordCountInput/  /use
r/hue/WordCountOutput
16/10/04 09:32:03 INFO client.RMProxy: Connecting to ResourceManager at sandbox.
hortonworks.com/192.168.11.128:8050
16/10/04 09:32:04 WARN mapreduce.JobSubmitter: Hadoop command-line option parsin
g not performed. Implement the Tool interface and execute your application with
ToolRunner to remedy this.
16/10/04 09:32:05 INFO input.FileInputFormat: Total input paths to process : 1
16/10/04 09:32:05 INFO mapreduce.JobSubmitter: number of splits:1
```

**8) You can check the status of job in web UI. Click on Job Browser you will be able to see the progress of the job.**



Job Browser

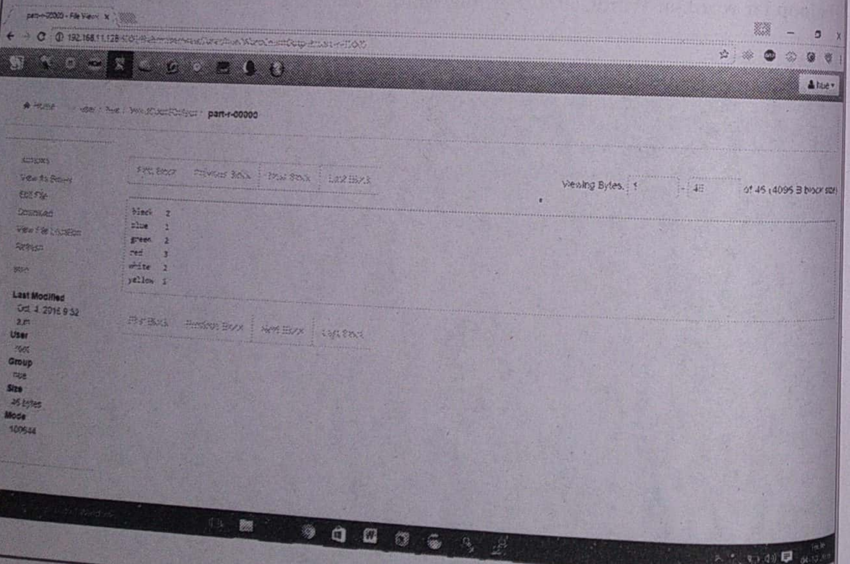
Username root

Show retired jobs

| Logs | ID | Name | Status | User | Maps | Reduces | Queue | Priority | Duration | Date |
|---|---|---|---|---|---|---|---|---|---|---|
| | [illegible] | word.jar | SUCCEEDED | root | 100% | 100% | default | N/A | 25s | 10/04/16 09:32:06 |

Showing 1 to 1 of 1 entries

**9) After completion of job it will create an output file in "WordCountOutput" directory.**



part-r-00000

```
black   2
blue    1
green   2
red     3
white   2
yellow  1
```

Last Modified
Oct. 4, 2016 9:32 a.m.
User
root
Group
root
Size
45 bytes
Mode
100644

**Q.3** Explain working of following phases of MapReduce with one common example.

i. Map Phase ii. Combiner Phase

iii. Shuffle and Sort Phase iv. Reducer Phase

**(Refer section 2.3.1)** [7]

**Summer - 2017 (IT)**

**Q.4** Explain MapReduce framework in detail. Draw the architecture diagram for physical organization of compute nodes. **(Refer sections 2.3.1 and 2.2.1)** [7]

**Q.5** Write short note on Hadoop Ecosystem. Also explain various elements of Hadoop. **(Refer section 2.2.2)** [7]

**Q.6** Define HDFS. Discuss HDFS architecture and HDFS commands in brief. **(Refer section 2.2.1)** [7]

**Q.7** Discuss Hadoop YARN in detail with failures in classic MapReduce. [7]

**Ans. : Hadoop YARN**

- YARN which stands for Yet Another Resource Negotiator is a new component added in Hadoop 2.0
- Following figure show how hadoop architecture has changed from Hadoop 1.0 to Hadoop 2.0

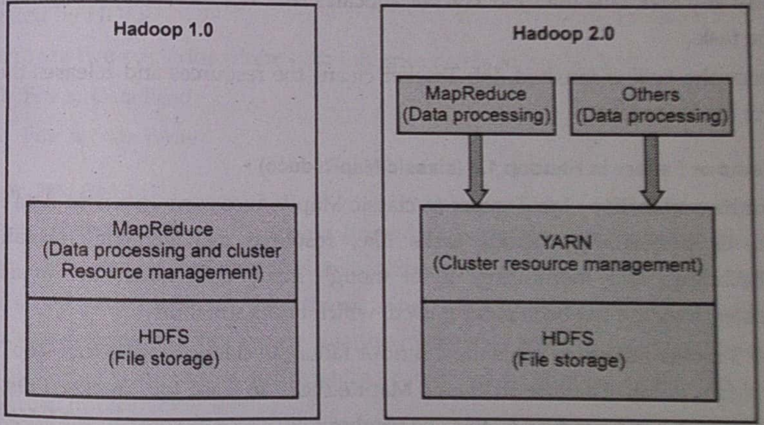


**Fig. 2.1**

- As shown in the figure above, a new layer is added between HDFS and MapReduce. This layer is of YARN which is responsible for Cluster Resource Management.
- Cluster resource management means managing various resources of Hadoop like Memory, CPU etc.

- In Hadoop 1.0 MapReduce was performing both cluster resource management and data processing, but in Hadoop 2.0 YARN took the task of cluster resource management and MapReduce is kept with the task of data processing.

**Cluster Resource Management in Hadoop 1.0**

- In Hadoop 1.0, tight coupling was there between MapReduce programming model and Cluster resource management.
- Job Tracker is the part of MapReduce framework which does the task of resource management.
- In MapReduce framework, MapReduce application or job is divided into number of mapper and reducer tasks. Every task runs on Data Node of cluster, and these data nodes have predefined map and reduce slot in order to execute task concurrently.
- In Hadoop 1.0, Job Tracker is responsible for cluster resource management as well as for driving execution of MapReduce application.
- The work of Job Tracker is to reserve and schedule slots for all tasks. The configuration, execution and monitoring of every task is done by Job Tracker. In case any task fails then Job Tracker allocates new slot for that task and reattempts the task.
- Once the task is finished, Job Tracker cleans the resources and releases the task's slot for other jobs.

**Problems or Failure in Hadoop 1.0 (classic MapReduce)**

- **Limits scalability :** Job Tracker in classic MapReduce runs on single data node in cluster performing various tasks like, resource management, Job and task scheduling, and monitoring. Even though many data nodes are available in cluster; they are not been getting used, which limits scalability.
- **Job Tracker failure :** This is most serious failure in classic MapReduce. Job Tracker is single point of failure in classic MapReduce. In Case Job Tracker fails, all the jobs which were running need to be resubmitted.
- **Resource Utilization Issue :** In classic MapReduce there is predefined number of map and reduce slots for every Task Tracker. The problem of resource utilization can occur because map slots might be full while reduce slots are empty. In such case the data node can sit idle which are reserved for reducer slot, even thought there might me immediate requirement of data nodes for mapper slot.
- YARN solves all the problems or failures of classic MapReduce.

- YARN contains central resource manager component that manages and allocates resources to the application. Multiple applications can be executed on Hadoop using YARN and these applications can share common resource manager.

**Advantages of YARN**

- YARN performs efficient utilization of cluster resources. It provides central resource manager.
- Using YARN you can execute the application which does not follow MapReduce model.

**Q.8** How does HDFS ensure data integrity in a Hadoop cluster. [7]

**Ans. : HDFS ensures data integrity as follows :**

- There is large possibility that the data might get corrupted during storage or processing of data.
- Since Hadoop deals with large volume of data, the probability of corruption of data increases, but Hadoop have checksum mechanism which deals with corruption of data.
- At the time of creating or writing the file in HDFS another checksum file will be created by HDFS
- There are two scenarios where data integrity needs to be checked :
  1. File or Data Read
  2. File or Data Write

**1. File or Data Read :**

- When client tries to read the data from data node, the checksum of file will be compared with the one that is already stored on data node. If both the checksum are equal and identical, then there is not corruption otherwise it will throw checksum exception, but before this the client tries to read the data using name node. Name node marks such data as corrupted so that no other client can be redirected to it.

**2. File/Data Write :**

- At the time of writing the data, data nodes are responsible for verifying the checksum of file or block of data. When checksum is created for the new file which is going to be written, the third data node in pipeline compares the checksum of data to the checksum created at the start. If both this checksum are same, then no corruption is there in data block otherwise it will throw CHecksumException.

- In this way Data Integrity in Hadoop is ensured by HDFS.

**Summer - 2017 (CSE)**

**Q.9** What is MapReduce ? Explain working of various phases of MapReduce with appropriate example and diagram. **(Refer section 2.3.1)** [7]

**Q.10** What is Hadoop Ecosystem? Discuss various components of Hadoop Ecosystem. **(Refer section 2.2.2)** [7]

**Q.11** With suitable block diagram explain architecture of HDFS. Discuss role of data node and name node in HDFS. Give commands with appropriate arguments to perform data transfer between local file system and HDFS. **(Refer section 2.2.1)** [7]

**Winter - 2017 (IT)**

**Q.12** Explain Map-Reduce framework in brief. **(Refer section 2.3.1)** [4]

**Q.13** What are the advantages of Hadoop ? Explain Hadoop Architecture and its Components with proper diagram. **(Refer section 2.2.2)** [7]

**Q.14** Draw and explain HDFS architecture. Explain the functions of NameNode and DataNode. What is secondary NameNode ? Is it a substitute to the NameNode ? **(Refer section 2.2.1)** [7]

**Q.15** Explain following HDFS commands of HDFS with syntax and at least one example of each.
i) get ii) cp iii) chown **(Refer section 2.2.1)** [7]

**Winter - 2017 (CSE)**

**Q.16** How MapReduce works and explain terminology used in MapReduce ? **(Refer section 2.3.1)** [4]

**Summer - 2018 (IT)**

**Q.17** What is Hadoop Ecosystem ? Discuss various components of Hadoop Ecosystem. **(Refer section 2.2.2)** [7]

**Q.18** Explain following commands with syntax and at least one example of each. 1) copyFromLocal 2) showing the content of output file. **(Refer section 2.2.2)** [3]

**Q.19** Explain "Map Phase" and "Combiner Phase" in MapReduce. **(Refer section 2.3.1)** [4]

**Q.20** Define HDFS. Discuss the HDFS Architecture and HDFS Commands in brief. **(Refer section 2.2.1)** [7]

**Q.21** Write a MapReduce code for counting occurrences of specific words in the input text file(s). Also write the command to compile and run the code. [7]

**Ans. : MapReduce code for Word Count**

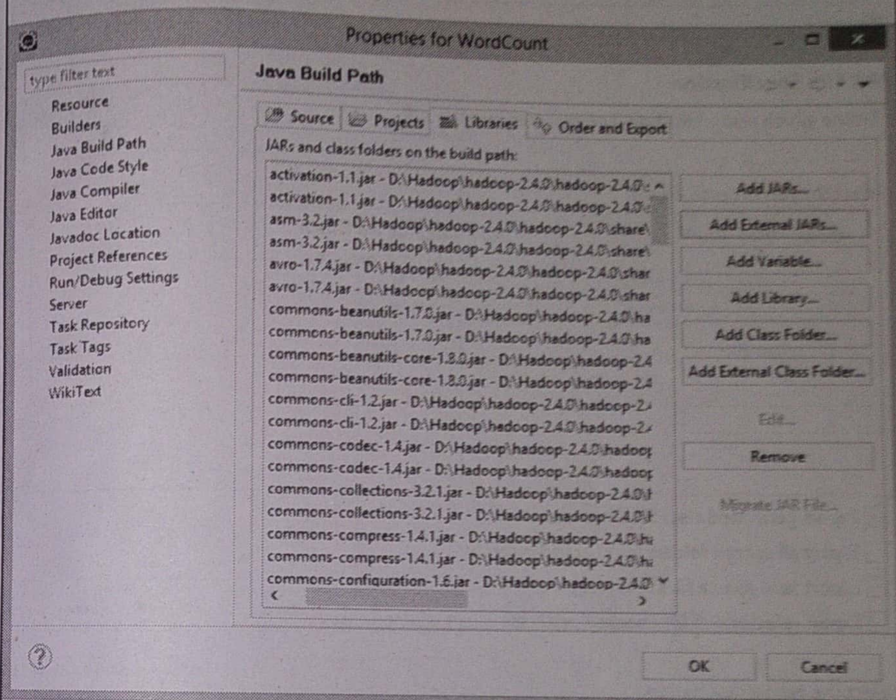
Steps to Compile & Run MapReduce Application in HortonWorks Sandbox

**1) Create following files in Eclipse IDE**

```
WordCount.java
importorg.apache.hadoop.conf.Configuration;
importorg.apache.hadoop.fs.Path;
importorg.apache.hadoop.io.IntWritable;
importorg.apache.hadoop.io.Text;
importorg.apache.hadoop.mapreduce.Job;
importorg.apache.hadoop.mapreduce.lib.input.FileInputFormat;
importorg.apache.hadoop.mapreduce.lib.input.TextInputFormat;
importorg.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
importorg.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class WordCount {
public static void main(String[] args) throws Exception {
if (args.length != 2) {
System.out.println("usage: [input] [output]");
System.exit(-1);
   }

Job job = Job.getInstance(new Configuration());
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

job.setMapperClass(WordMapper.class);
job.setReducerClass(SumReducer.class);

job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);

FileInputFormat.setInputPaths(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.setJarByClass(WordCount.class);
job.submit();
}
}

WordMapper.java
importjava.io.IOException;
importjava.util.StringTokenizer;
```

```
importorg.apache.hadoop.io.IntWritable;
importorg.apache.hadoop.io.Text;
importorg.apache.hadoop.mapreduce.Mapper;
public class WordMapper extends Mapper<Object, Text, Text, IntWritable> {
private Text word = new Text();
private final static IntWritable one = new IntWritable(1);

public void map(Object key, Text value, Context context_W) throws IOException,
InterruptedException {
String line = value.toString();
line = line.replaceAll("[^a-zA-Z0-9\\s]", "").toLowerCase();
StringTokenizerwordList = new StringTokenizer(line);
while (wordList.hasMoreTokens()) {
word.set(wordList.nextToken());
context_W.write(word, one);
  }
 }
}

SumReducer.java
importjava.io.IOException;
importjava.util.Iterator;

importorg.apache.hadoop.io.IntWritable;
importorg.apache.hadoop.io.Text;
importorg.apache.hadoop.mapreduce.Reducer;

public class SumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

privateIntWritabletotalWordCount = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values, Context context_W)
throwsIOException, InterruptedException {
intwordCount = 0;
   Iterator<IntWritable> it=values.iterator();
while (it.hasNext()) {
wordCount += it.next().get();
}
totalWordCount.set(wordCount);
context_W.write(key, totalWordCount);
 }
}
```

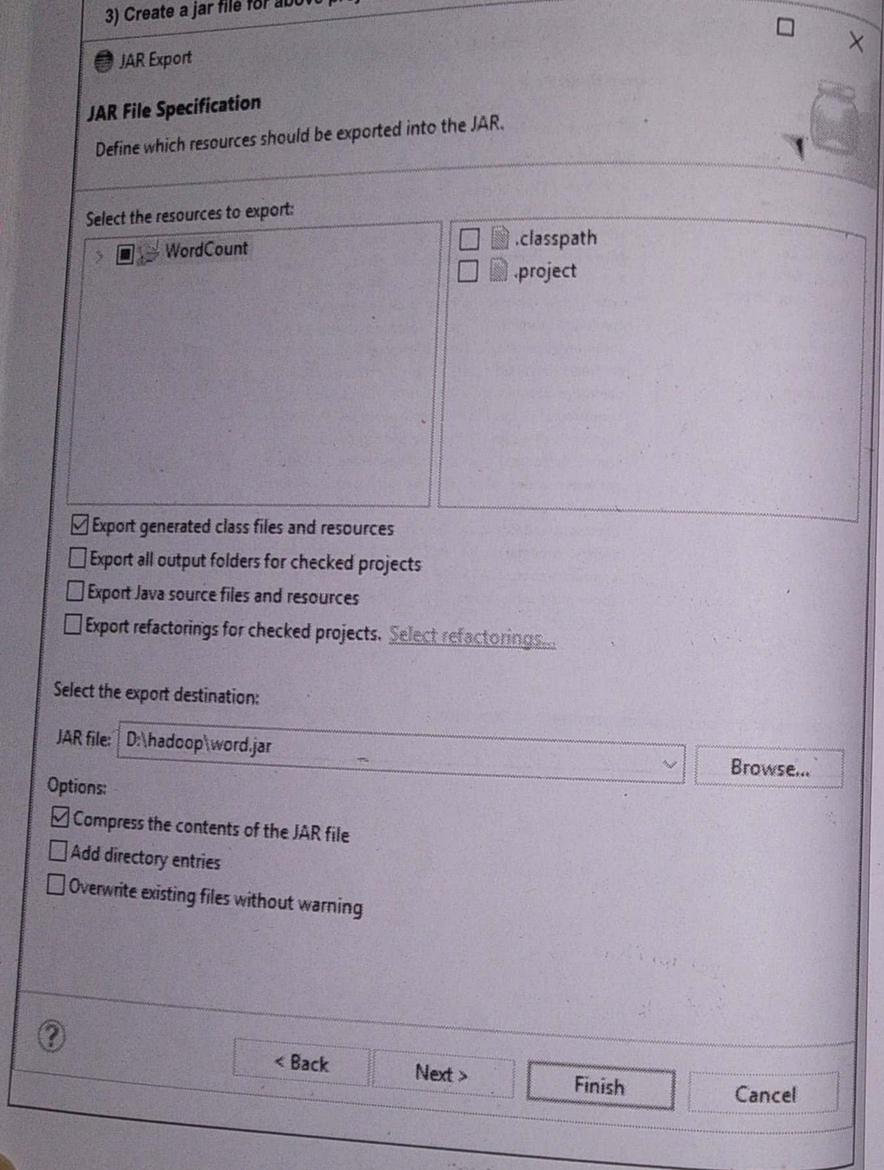
**2) Import the required jar files from the hadoop of the same version as of hortonworks.**



JARs and class folders on the build path:

```
activation-1.1.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\
activation-1.1.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\
asm-3.2.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\share\
asm-3.2.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\share\
avro-1.7.4.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\shar
avro-1.7.4.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.4.0\shar
commons-beanutils-1.7.0.jar - D:\Hadoop\hadoop-2.4.0\ha
commons-beanutils-1.7.0.jar - D:\Hadoop\hadoop-2.4.0\ha
commons-beanutils-core-1.8.0.jar - D:\Hadoop\hadoop-2.4
commons-beanutils-core-1.8.0.jar - D:\Hadoop\hadoop-2.4
commons-cli-1.2.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.
commons-cli-1.2.jar - D:\Hadoop\hadoop-2.4.0\hadoop-2.
commons-codec-1.4.jar - D:\Hadoop\hadoop-2.4.0\hadoop
commons-codec-1.4.jar - D:\Hadoop\hadoop-2.4.0\hadoop
commons-collections-3.2.1.jar - D:\Hadoop\hadoop-2.4.0\
commons-collections-3.2.1.jar - D:\Hadoop\hadoop-2.4.0\
commons-compress-1.4.1.jar - D:\Hadoop\hadoop-2.4.0\ha
commons-compress-1.4.1.jar - D:\Hadoop\hadoop-2.4.0\ha
commons-configuration-1.6.jar - D:\Hadoop\hadoop-2.4.0\
```

**3) Create a jar file for above project. Give it a name "word.jar"**



JAR Export

**JAR File Specification**

Define which resources should be exported into the JAR.

Select the resources to export:

- ■ WordCount
- ☐ .classpath
- ☐ .project

☑ Export generated class files and resources

☐ Export all output folders for checked projects

☐ Export Java source files and resources

☐ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: D:\hadoop\word.jar

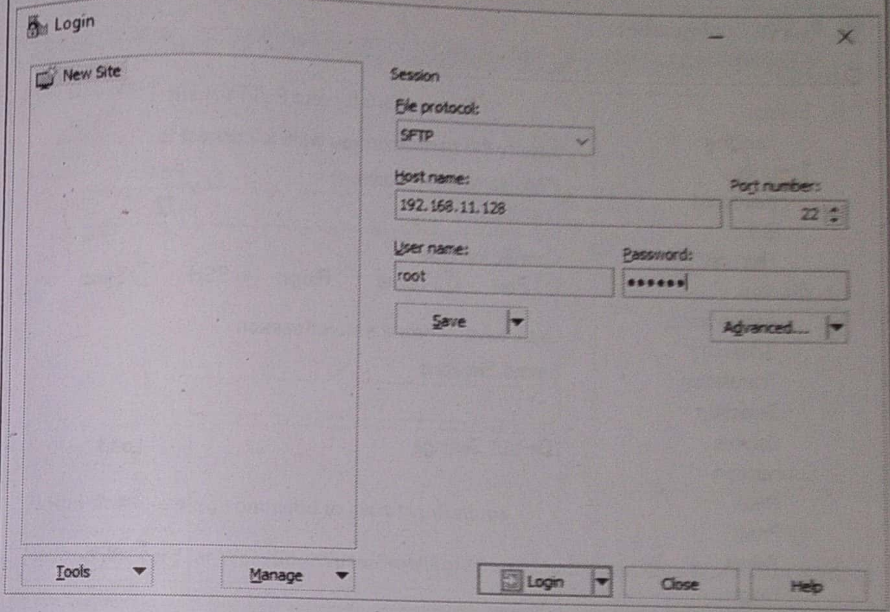
Options:

☑ Compress the contents of the JAR file

☐ Add directory entries

☐ Overwrite existing files without warning

**4) OpenWinSCP, connect it with the IP address of hortonworks. Transfer "word.jar" to "usr/lib/hue" for further use.**



Login

New Site

Session

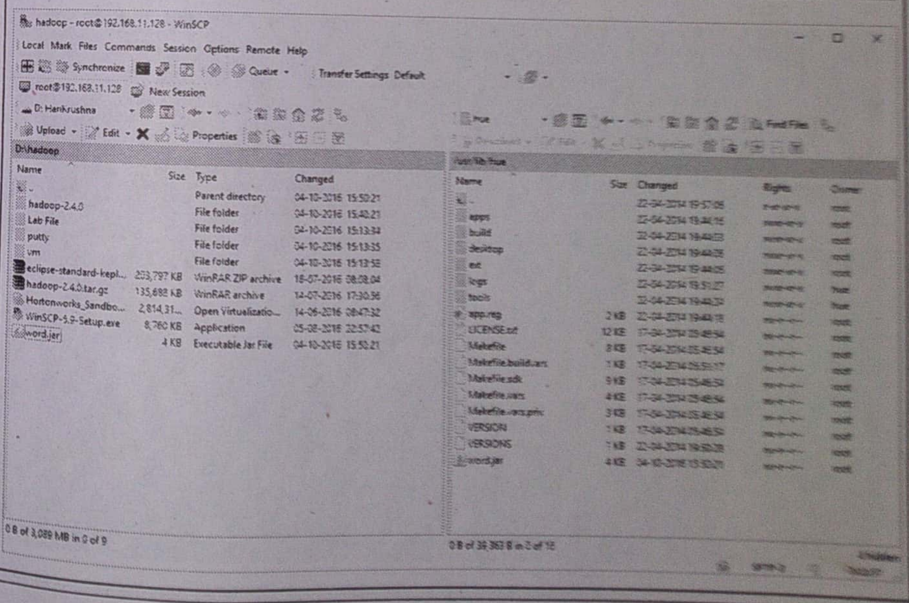
File protocol: SFTP

Host name: 192.168.11.128

Port number: 22

User name: root

Password: ••••••



hadoop - root@192.168.11.128 - WinSCP

D:\hadoop

| Name | Size | Type | Changed |
|---|---|---|---|
| .. | | Parent directory | 04-10-2016 15:50:21 |
| hadoop-2.4.0 | | File folder | 04-10-2016 15:40:21 |
| Lab File | | File folder | 04-10-2016 15:13:34 |
| putty | | File folder | 04-10-2016 15:13:35 |
| vm | | File folder | 04-10-2016 15:13:58 |
| eclipse-standard-kepl... | 203,797 KB | WinRAR ZIP archive | 18-07-2016 08:08:04 |
| hadoop-2.4.0.tar.gz | 135,692 KB | WinRAR archive | 14-07-2016 17:30:56 |
| Hortonworks_Sandbo... | 2,814,31... | Open Virtualizatio... | 14-06-2016 08:47:32 |
| WinSCP-5.9-Setup.exe | 8,760 KB | Application | 05-08-2016 22:57:42 |
| word.jar | 4 KB | Executable Jar File | 04-10-2016 15:50:21 |

0 B of 3,089 MB in 0 of 9

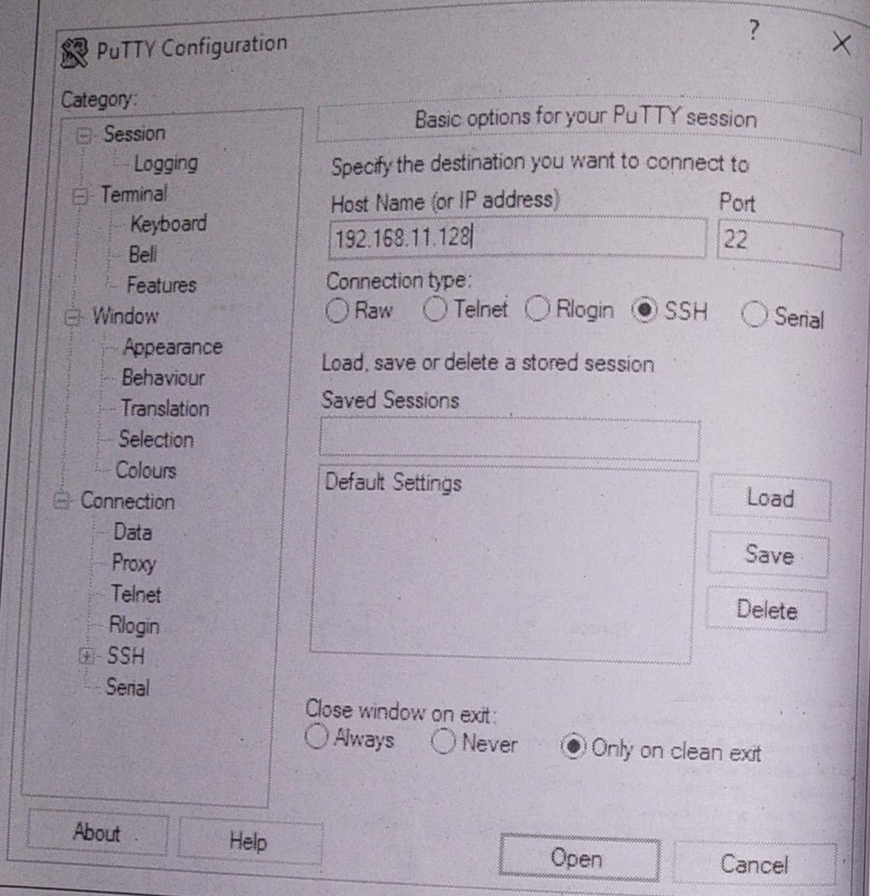
/usr/lib/hue

| Name | Size | Changed | Rights | Owner |
|---|---|---|---|---|
| .. | | 22-04-2014 19:57:06 | [illegible] | root |
| apps | | 22-04-2014 19:44:15 | [illegible] | root |
| build | | 22-04-2014 19:44:03 | [illegible] | root |
| desktop | | 22-04-2014 19:44:08 | [illegible] | root |
| ext | | 22-04-2014 19:44:05 | [illegible] | root |
| logs | | 22-04-2014 19:51:27 | [illegible] | hue |
| tools | | 22-04-2014 19:44:34 | [illegible] | hue |
| app.reg | 2 KB | 22-04-2014 19:44:15 | [illegible] | root |
| LICENSE.txt | 12 KB | 17-04-2014 05:46:54 | [illegible] | root |
| Makefile | 8 KB | 17-04-2014 05:46:54 | [illegible] | root |
| Makefile.buildvars | 1 KB | 17-04-2014 05:51:17 | [illegible] | root |
| Makefile.sdk | 9 KB | 17-04-2014 05:46:54 | [illegible] | root |
| Makefile.vars | 4 KB | 17-04-2014 05:46:54 | [illegible] | root |
| Makefile.vars.priv | 3 KB | 17-04-2014 05:46:54 | [illegible] | root |
| VERSION | 1 KB | 17-04-2014 05:46:54 | [illegible] | root |
| VERSIONS | 1 KB | 22-04-2014 19:50:28 | [illegible] | root |
| word.jar | 4 KB | 04-10-2016 15:50:21 | [illegible] | root |

0 B of 39,363 B in 0 of 16

Scanned by CamScanner

**5) Open Putty, connect it with the IP address of hortonworks.**



Specify the destination you want to connect to

Host Name (or IP address): 192.168.11.128

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

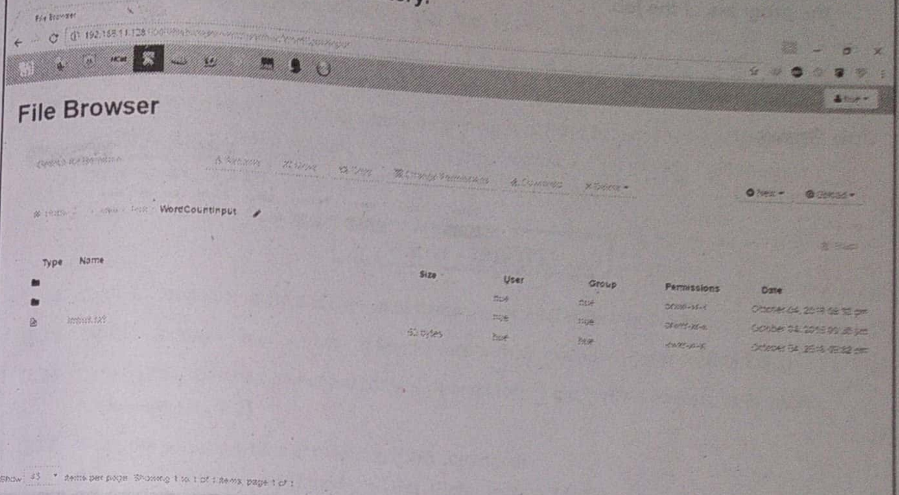
Load, save or delete a stored session

Saved Sessions

Default Settings

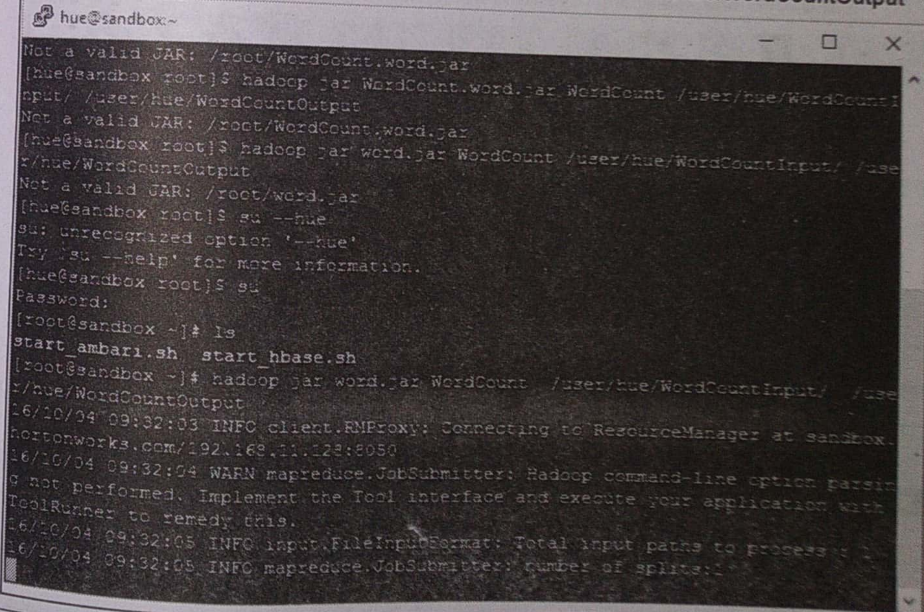
Close window on exit: Always Never [✓ Only on clean exit]

**6) Upload input file to HDFS using web UI. Create a directory of name "WordCountInput" and add the input file in this directory.**



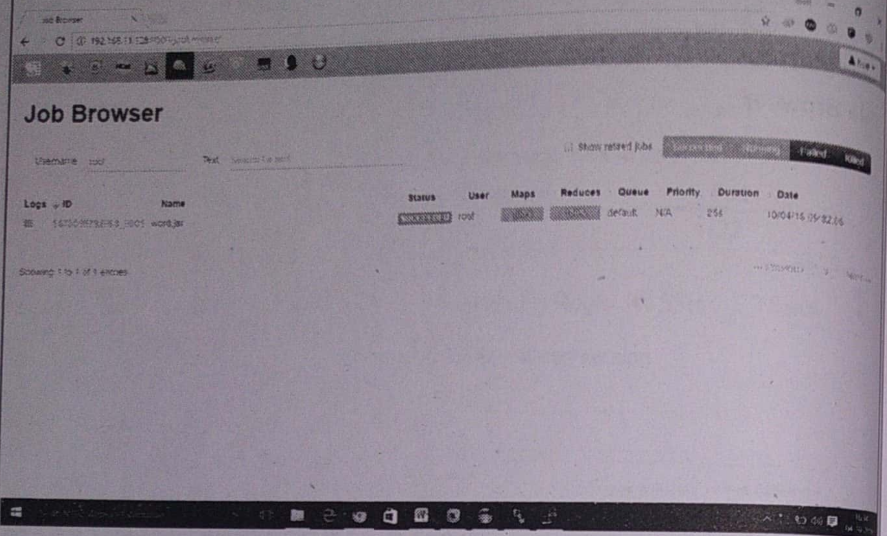
**7) Run the following command to start MapReduce**

**Hadoop jar word.jar WordCount/user/hue/WordCountInput/ /user/hue/WordCountOutput**



```
Not a valid JAR: /root/WordCount.word.jar
[hue@sandbox root]$ hadoop jar WordCount.word.jar WordCount /user/hue/WordCountI
nput/ /user/hue/WordCountOutput
Not a valid JAR: /root/WordCount.word.jar
[hue@sandbox root]$ hadoop jar word.jar WordCount /user/hue/WordCountInput/ /use
r/hue/WordCountOutput
Not a valid JAR: /root/word.jar
[hue@sandbox root]$ su --hue
su: unrecognized option '--hue'
Try 'su --help' for more information.
[hue@sandbox root]$ su
Password:
[root@sandbox ~]# ls
start_ambari.sh  start_hbase.sh
[root@sandbox ~]# hadoop jar word.jar WordCount  /user/hue/WordCountInput/  /use
r/hue/WordCountOutput
16/10/04 09:32:03 INFO client.RMProxy: Connecting to ResourceManager at sandbox.
hortonworks.com/192.168.11.128:8050
16/10/04 09:32:04 WARN mapreduce.JobSubmitter: Hadoop command-line option parsin
g not performed. Implement the Tool interface and execute your application with
ToolRunner to remedy this.
16/10/04 09:32:05 INFO input.FileInputFormat: Total input paths to process : 1
16/10/04 09:32:05 INFO mapreduce.JobSubmitter: number of splits:1
```

8) You can check the status of job in web UI. Click on Job Browser you will be able to see the progress of the job.



Job Browser

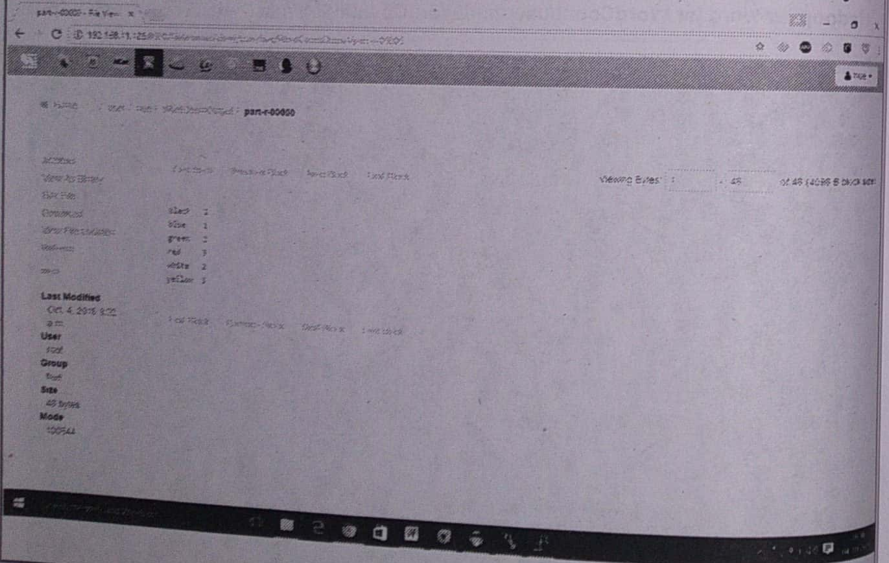
Username root Text

Show retired jobs Succeeded Running Failed Killed

| Logs | ID | Name | Status | User | Maps | Reduces | Queue | Priority | Duration | Date |
|---|---|---|---|---|---|---|---|---|---|---|
|  | [illegible]_0001 | word.jar | SUCCEEDED | root | [illegible] | [illegible] | default | N/A | 25s | 10/04/16 09:32:06 |

Showing 1 to 1 of 1 entries

9) After completion of job it will create an output file in "WordCountOutput" directory.



part-r-00000

Viewing Bytes: 1 - 48 of 48 (4096 B block size)

```
black  1
blue   1
green  1
red    3
white  2
yellow 1
```

Last Modified
Oct. 4, 2016 9:32 a.m.
User
root
Group
root
Size
48 bytes
Mode
100644

**Summer - 2018 (CSE)**

Q.22 Explain working of following phases of Map Reduce with one common example.
(i) Map Phase (ii) Combiner Phase (iii) Shuffle and Sort Phase (iv) Reducer Phase
**(Refer section 2.3.1)** [7]

Q.23 Draw HDFS Architecture. Explain any two commands of HDFS from following commands with syntax and at least one example of each. (i) copyFromLocal (ii) setrep (iii) checksum
**(Refer section 2.2.1)** [7]

**Winter - 2018 (IT)**

Q.24 Draw Hadoop ecosystem and explain its components. **(Refer section 2.2.2)** [7]

Q.25 Explain working of reduce phase of MapReduce with an example. **(Refer section 2.3.1)** [4]

Q.26 Define HDFS. Describe namenode, datanode and block. Explain HDFS operations in detail.
**(Refer section 2.2.1)** [7]

Q.27 Write the use and syntax of following HDFS commands :
I) Put ii) Expunge iii) Chmod iv) Get **(Refer section 2.2.1)** [4]

**Winter - 2018 (CSE)**

Q.28 What is MapReduce ? Explain working of various phases of MapReduce with word count example. **(Refer section 2.3.1)** [7]

Q.29 Write a short note on Hadoop Ecosystem. **(Refer section 2.2.2)** [7]

Q.30 Give commands with appropriate arguments to perform data transfer between local file system and HDFS. **(Refer section 2.2.1)** [7]

Q.31 With suitable block diagram explain architecture of HDFS. **(Refer section 2.2.1)** [7]

Q.32 Discuss role of Data node and Name node in HDFS. **(Refer section 2.2.1)** [4]

**Summer - 2019 (IT)**

Q.33 Explain Hadoop components with diagram. **(Refer section 2.2.2)** [7]

Q.34 Describe Map and Reduce phases of MapReduce. **(Refer section 2.3.1)** [4]

Q.35 Define HDFS. Explain HDFS Architecture with diagram. **(Refer section 2.2.1)** [7]

Q.36 Write the use and syntax of following HDFS commands : **(Refer section 2.2.1)**
i) mkdir ii) chown iii)put iv) copyToLocal [4]

# 3 NoSQL

***Syllabus***

*What is NoSQL ? NoSQL business drivers; NoSQL case studies; NoSQL data architecture patterns: Key-value stores, Graph stores, Column family (Bigtable) stores, Document stores, Variations of NoSQL architectural patterns; Using NoSQL to manage big data : What is a big data NoSQL solution ? Understanding the types of big data problems; Analyzing big data with a shared-nothing architecture; Choosing distribution models : master-slave versus peer-to-peer; Four ways that NoSQL systems handle big data problems.*

***Contents***

## 3.1 What is NoSQL ?

GTU : Winter-16, 17, 18, 19, Summer-17, 18, 19

- A NoSQL database demonstrates a system for the recovery and capacity of information which is composed in means other than relations utilized as a part of RDBMS.
- It comprises of a wide variety of advances in database that were created and gave a responsive ascent in the volume of information put away about clients, articles and items, the recurrence in which this information is gotten to, and execution and preparing needs. For the most part, these NoSQL databases are been organized in a key-value collection, graph database, document oriented structure.
- Over last few years, databases are used in form of SQL where data is stored in relational tables. Whereas, in recent times there was exponential rise in the use of web applications and internet, the size or database increased very rapidly.
- A complete new era of data is arise when applications like Amazon, whatsapp, facebook etc, came into use where management of data is done by following techniques of basic design, scaling and speed in access. These sorts of databases are utilized as a part of big data, real time applications and analytics.
- For instance, suppose that you have a blogging application that stores client's blogs. Presently assume that you need to add some new features in your application, such as, liking blog or commenting on blog or liking comments. Using RDBMS system, this will require an entire upgrade to your current database design.
- In such cases use of NoSQL makes it easy to modify the data structure to match with the requirement. Using NoSQL you can directly insert data in existing design without adding any new column definition.

## 3.2 NoSQL Business Drivers

- The demands of volume, velocity, variability, and agility play a key role in the emergence of NoSQL solutions. As each of these drivers applies pressure to the single processor relational model, its foundation becomes less stable and, in time, no longer meets the organization's needs.
- Volume and velocity refer to the ability to handle large datasets that arrive quickly. Variability refers to how diverse data types don't fit into structured tables and agility refer to how quickly an organization responds to business change.

### 3.2.1 Volume

- In organization, to query Big data using clusters of commodity processors has become key factor to look for alternatives to their current RDBMS. Concerns with performance at earlier time were resolved by purchasing faster processors. Later, increase in processing speed was no longer an option.
- Because of increase in chip density, heat is not able to dissipate faster which results in overheating of chip. This objective has forced system designers to cornerstone in using multiple processor instead of increasing speed on single chip and this is also known as Power Wall.
- The need to scale horizontally has made organizations to move to parallel processing where queries are divided and sent to different processor.

### 3.2.2 Velocity

- While Big Data issues are a thought for some, associations moving endlessly from RDBMS frameworks, the capacity of a solitary processor framework to quickly peruse and compose information is likewise key. Many single processor RDBMS frameworks can't stay aware of the requests of continuous real-time inserts and online queries to the database made by public facing sites.
- RDBMS frameworks as often as possible record numerous segments of each new column, a cycle that diminishes framework execution. At the point when single processors RDBMSs are utilized as a back finish to a web retail facade, the irregular rushes in web traffic hinder reaction for everybody and tuning these frameworks can be expensive when both high peruse and compose throughput is wanted.

### 3.2.3 Variability

- Organizations that need to catch and write about special case information battle when endeavoring to utilize inflexible database schema structures forced by RDBMS frameworks.
- For instance, if a business unit needs to catch a couple of custom fields for a specific client, all client columns inside the data set require to store this data despite the fact that it doesn't matter.
- Adding new columns to a RDBMS requires the framework to be shut down and ALTER TABLE command to be run. At the point when database is huge, this cycle can affect framework accessibility, losing time and money all the while.

### 3.2.4 Agility

- The most complex part of building applications utilizing RDBMSs is the way toward placing information into and getting information out of the database. In the event that your information has settled and repeated subgroups of data structures you need to incorporate an object relational mapping layer. The duty of this layer is to produce the right blend of INSERT, UPDATE, DELETE and SELECT SQL statements to move object data to and from the RDBMS persistence layer. This cycle isn't straightforward and is related with the biggest barrier to fast change when growing new or altering existing applications.
- Object-relational mapping requires experienced software developers who know about object-relational frameworks, for example, Java Hibernate. Indeed, even with experienced staff, little change request can cause slowdowns in development and testing plans.
- We perceive how velocity, volume, variability, and agility are the significant level drivers most much of the time related with the NoSQL development.
- At the point when we venture into any new innovation it is basic to understand that every region has its own examples of critical thinking. These examples differ drastically from innovation to innovation. Making the progress from SQL to NoSQL is the same. NoSQL is another worldview and requires another arrangement of example acknowledgment aptitudes, better approaches for speculation and better approaches for taking care of issues. It requires another intellectual style.
- Selecting to utilize NoSQL innovations can help associations pick up a serious edge in their market, making them more agile and better prepared to adjust to changing business conditions. NoSQL approaches that influence large quantities of commodity processors set aside organizations time and money and increment administration dependability.

## 3.3 NoSQL Case Studies

- As economy is changing, to remain in competition organizations required to look at other ways to retain existing customer and on-board new customers. To achieve this support to the efforts need to be provided quickly and in cost effective manner by the technology and creators of such technologies.
- Following are few case studies which represent how companies have been able to solve their business problem faster and in cost effective way yet being effective.

**1. Case study : Google's MapReduce - use commodity hardware to create search indexes**

- In NoSQL movement, the most important case study is the Google MapReduce System. Google have shared how they have done transformation of large volume of data into search indexes using commodity hardware.
- Even though sharing of this process was important, the techniques behind map and reduce were not new. The two stages in transformation of data are named as Map and reduce.
- Map operation is initial stage of data transformation which performs transformation, extraction and filtering of data. The output of map function is then sent to reduce function. The reduce function is responsible for sorting, combining and summarizing data to produce final output.
- The programmers at MIT have implemented these functions in dominant LISP system, this outstanding work in field of computer science is the main concept behind map and reduce functions. LISP was unlike any other programming language since it has functions which can transform isolated lists of data.
- Google expanded map and reduce functions so that it can efficiently execute on large volume of web pages on number of cost effective commodity hardware. Google's use of MapReduce has inspired other organizations to look at strength of functional programming and potential to scale over number of low cost commodity hardware.
- The use of MapReduce has also motivated engineers from Yahoo! and other companies to create different open source version of MapReduce.
- It shows awareness regarding limitations of traditional procedural programming is increasing day by day and motivating them to go with functional programming.

**2. Case study : Google's Bigtable - a table with a billion rows and a million columns**

- When Google announced white paper on Bigtable system they have guided many developers. The requirement to store results from different search engines which take out HTML pages, audio, images, video and other media from the internet, was the main inspiration behind Bigtable. The resulting dataset was very large in volume that it could not fit into single relational database, so Google developed their own storage system.
- Google's primary objective was to develop a system which can be scaled easily as the increase in data without buying costly commodity hardware. This solution was

neither complete file system nor a relational database, Google have named it as "distributed storage system" which works with structured data.

- Bigtable project was very much successful and it provided single view of entire data to Google developers by creating large table which stores all the required data. Also they have created the system which facilitate the hardware to be located at any data center in the world, and developed an environment where developer don't have to care about actual location of data they manipulated.

**3. Case study : Amazon's Dynamo - accept an order 24 hours a day, 7 days a week**

- Google's work focused on different techniques to make distributed processing and reporting easy, but it was not planned to support requirement of highly scalable web storefront which runs 24/7. This development has been done by Amazon. Amazon has published their NoSQL paper : Amazon's Dynamo - highly available key value store. The inspiration for building Dynamo has arised from amazon's requirement to make highly dependable web storefront which could support transaction from all over the world every day without any interruption.
- Amazon's business model was different than traditional business model of retailers. Their customers are from all part of the world and also they do shopping at all hours of the day. Any kind of downtime could result in loss of millions of dollar. Amazon's system required to be scalable and dependable without any loss in service.
- Initially Amazon has used relational database to support their day today transactions. But going forward they have realized that in future relational database model could not fulfill their requirements
- Amazon's Dynamo paper was cited as important turning point in NoSQL movement by many organizations in NoSQL community.
- Amazon has built turnkey system by using key-value store which is dependable, scalable and will be able to support their 24/7 business model, making Amazon one of most successful online retailer in the world.

## 3.4 NoSQL Data Architecture Pattern

**GTU : Summer-17, 18, 19, Winter-17, 18, 19**

- There are four data architecture pattern of NoSQL databases. Each of these has their own benefits and limitations.
- Considering NoSQL databases, following are widely identified data architecture patterns :

  - o Key-value stores
  - o Column-oriented
  - o Graph
  - o Document oriented

### 3.4.1 Key-value Stores

- Key value stores handles large volume of data and it is most essential data architecture pattern of NoSQL databases.
- Schema less data can be easily stored into such databases. In this data is stored as hash table.
- Each key is unique and value can be string, JSON, BLOB (Binary Large Object), etc.
- A key must be string, hashes, lists, sets and values are stored against these keys.
- For example, key-value pair may be made up of a key like "Name" which is connected with value such as "Avinash".
- These can be utilized as dictionaries, collections, associative arrays, etc.
- It follows the 'availability' and 'partition' aspect of CAP theorem.
- For shopping cart content key-value store functions well.

**Example of Key-value store DataBase :** Redis, Dynamo, Riak and so on.

**Pictorial Presentation :**

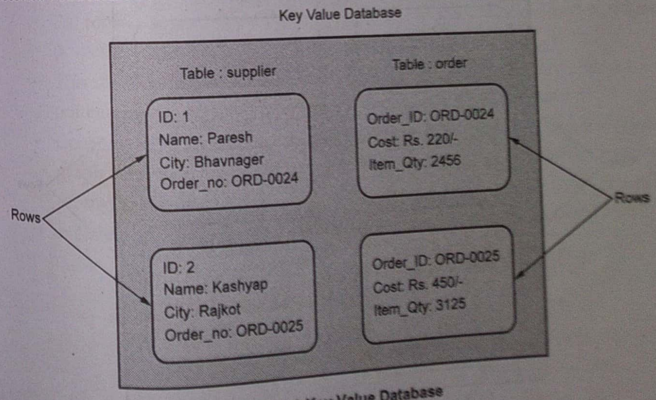


Fig. 3.4.1 Key-Value Database

## 3.4.2 Column-Oriented Databases

- Column-oriented databases works on columns where every column is treated individually.
- Storage of single values is in contiguous manner.
- In Column stores, query processors work on columns too.
- Data within each column data file have the similar type which makes it perfect for compression.
- Performance of queries is improved as specific column data is accessed.
- High performance on aggregation queries (e.g. COUNT, SUM, AVG, MIN, MAX).
- Works on data warehouses and business intelligence, Customer Relationship Management (CRM), Library card catalogs etc.
- The column-oriented storage permits data to be stored effectively. Nulls are stored by simply not storing a column when a value doesn't exist for that column.
- Each unit of data can be considering of as a set of key/value pairs, often referred to as the primary key. BigTable and its clones tend to call this primary key the row-key.

**Example :** BigTable, Cassandra, SimpleDB etc.

**Pictorial Presentation :**

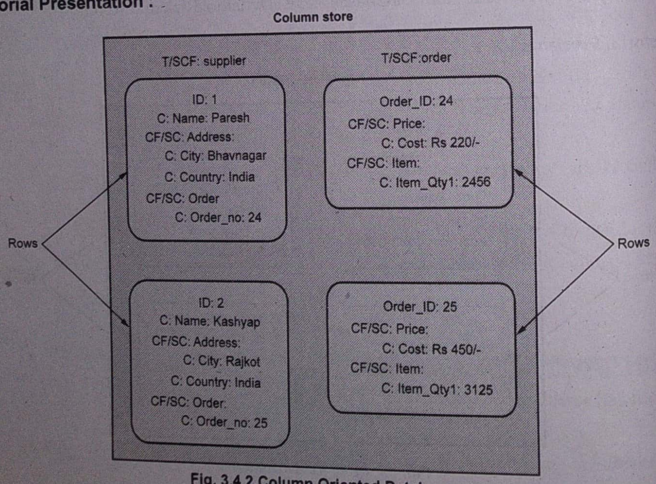


**Fig. 3.4.2 Column Oriented Database**

## 3.4.3 Graph Databases

- A graph data structure comprises of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices. A labeled graph of 6 vertices and 7 edges is shown below.

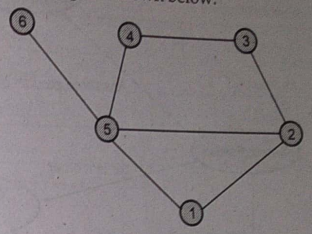


**Fig. 3.4.3 Graph**

- Data is stored as graph.
- Any type of data can be represented in highly accessible way.
- It is collection of edges and nodes.
- An entity is represented by each node and relation between two nodes is represented by edge.
- A specific identifier defines nodes and edges. Adjacent nodes are identified by each node.
- To represent and store data graph database make use of nodes, edges and properties of graph structure.
- This database provides index-free adjacency, which means every node have direct pointer to its nearby node and look up of indexes are not necessary.
- General graph databases that can store any graph are distinct from specialized graph databases, for example, triple-stores and network databases. Indexes are used for traversing the graph.
- Following is the comparison between graph model and relation model :

| Relational model | Graph model |
|---|---|
| Tables | Vertices and Edges set |
| Rows | Vertices |
| Columns | Key/value pairs |
| Joins | Edges |

Scanned by CamScanner

**Example :** OrientDB, Neo4J, Titan.etc.

**Pictorial Presentation :**

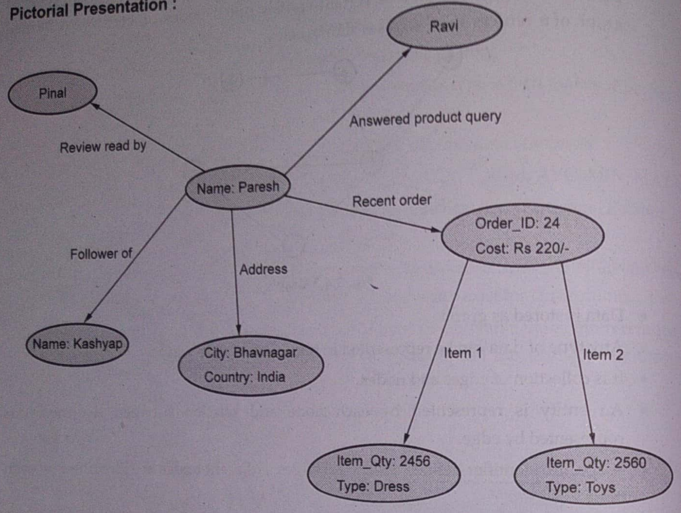


Fig. 3.4.4 Graph database

### 3.4.4 Document Oriented Databases

- Data is stored as a document and the database is nothing but the collection of documents.
- Data is stored as key-value pair, where key allows access to its value.
- These documents are easy to modify.
- To combine different type of data, the documents are stored into collections.
- Document contains wide variety of key-value pair and nested documents.
- It considers a document as whole and does not split document in its key-value pairs.
- At a collection level, this allows for putting together a diverse set of documents into a single collection.
- Document databases permits indexing of documents on the basis of not only its primary identifier but also its properties.

- Various document oriented databases are available, but most widely used document oriented databases are MongoDB and CouchDB.
- Following is the comparison between relational model and document model :

| Relational model | Document model |
|---|---|
| Tables | Collections |
| Rows | Documents |
| Columns | Key/value pairs |
| Joins | Unavailable |

**Example :** MongoDB, CouchDB etc.

**Pictorial Presentation :**

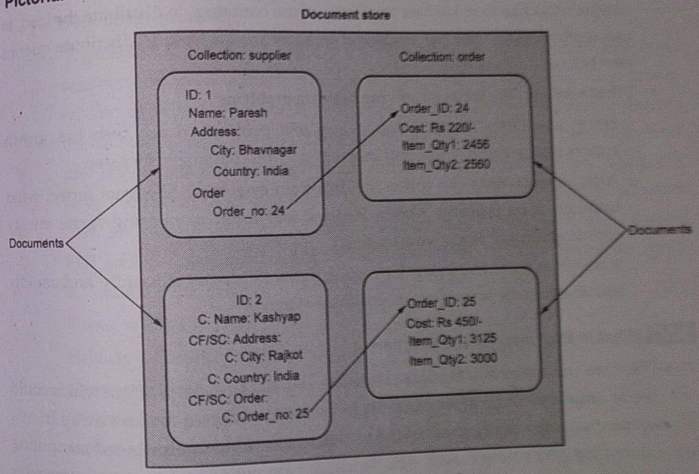


Fig. 3.4.5 Document oriented database

## 3.5 Using NoSQL to Manage Big Data

- The main reason behind organizations moving towards a NoSQL solution and leaving the RDMS system behind is the requirement to analyze a large volume of data.

- Datasets containing about a million internal sales transactions that were stored on a single processor in a relational database were managed by companies about twenty years ago.
- The datasets expanded to billions and trillions of items when organizations started generating more data from internal as well as external sources. To continue to use a single system to process the amount of data was getting difficult for the organizations.
- Learning distribution of the task among many processors was they had to do. A big data problem is the name of this situation.
- To solve these big data problems and manage it, usage of a NoSQL solution gives unique ways to handle.
- We can manage our data and keep our system running faster, we can use these unique ways like to move data to queries, to use hash rings to distribute the load, to use replication to scale our reads and to allow the database to distribute queries evenly to our data nodes.
- **What's driving the focus on solving big data problems ?**
  - o Since late 1990s the information available publicly on the web has grown exponentially and the expectations are that it will continue to increase.
  - o Also, organizations can collect the data from everything like from farms, wind turbines, manufacturing plants, vehicles, and meter monitoring home energy consumption due to low-cost sensors.
  - o It becomes more important for organizations to process efficiently and rapidly and analyzes the large datasets due to these trends.

### 3.5.1 What is Big Data NoSQL Solution

- First we will see what a big data problem is. It is any business problem which could be so large and single processor cannot manage it. We forced to move away from a single processor environment to the more complex world of distributed computing due to big data problem. Although, distributed computing environments come with their own set of challenges while solving big data problems.
- It needs to be highlighted that big data is not the same as NoSQL, since the NoSQL is more than dealing with large datasets. NoSQL have a positive impact on agility and data quality as it includes concepts and use cases that can be managed by a single processor. But for NoSQL, it is considered that big data problems, a primary use case.

- It is necessary to consider whether we need all or our data or a subset of our data to solve our problem before we assume that we have a big data problem. The use of a subset of our data and look for a pattern in the subset is possible when we use a statistical sample.
- The idea is to come up with a process to ensure the sample you choose is a fair representation of the full dataset. Also how quickly we need our data processed should also be considered.
- A batch-type solution running on a single processor handles data analysis problems. To understand the true time critical nature of your situation is the key.
- The distributed databases are more complex than a single processor system and there are alternatives to use a full dataset.
- Here are some typical big data use cases :
  - o **Bulk image processing**

    Organizations like NASA regularly receive terabytes of incoming data from satellites or even rovers on Mars. NASA perform functions like image enhancement and photo stitching by using a large number of servers to process the images. Medical imaging systems like CAT scans and MRIs need to convert raw image data into formats that are useful to doctors and patients. Renting a large number of processors on the cloud when they're needed is less expensive than the custom imaging hardware. For example, by using tools like Amazon EC2 and Hadoop, with a few hundred dollars, the New York Times converted 3.3 million scans of old newspaper articles into web formats.
  - o **Public web page data**

    Publicly accessible pages are full of information that organizations can use to be more competitive. They contain news stories, RSS feeds, new product information, product reviews, and blog postings. Authentication of all the information is not there. Competitors have created fake product reviews in millions of pages, also third parties paid to disparage other sites. The topic for careful analysis is finding out which product reviews are valid.
  - o **Remote sensor data**

    Small, low-power sensors can now track almost any aspect of our world. Tracking location, speed, acceleration, and fuel consumption, and informing our insurance company about our driving habits is being done by devises installed on vehicles. Also, warning about traffic jams in real time and suggest alternate routes is being done by the road sensors. To suggest a watering plan for our home, we can even track the moisture in our garden, lawn, and indoor plants.

- **Event log data**

  Computer systems create logs of read-only events from web page hits, email messages sent, or login attempts. Organizations understand who's using what resources and when systems may not be performing according to specification with the help of each of these events. To send alerts to users when key indicators fall out of acceptable ranges, event log data can be fed into operational intelligence tools.

- **Mobile phone data**

  Applications can track the events every time users move to new locations. We can see when our friends are around us or when customers walk through our retail store. Although in accessing this data there are privacy issues involved. It's forming a new type of event stream that can be used in innovative ways to give companies a competitive advantage.

- **Social media data**

  A continuous real-time data feed that can be used to see relationships and trends is being provided by social networks such as Twitter, Facebook, and LinkedIn. Each site creates data feeds that we can use to look at trends in customer mood or get feedback on our own as well as competitor products.

- **Game data**

  Back-end datasets of games that run on PCs, video game consoles, and mobile devices, need to scale quickly. These games store and share high scores for all users as well as game data for each player. If viral marketing campaigns catch on with their users, game site back ends must be able to scale by orders of magnitude.

- We can see that, since the output from one transform isn't used as an input to another, some problems can be described as independent parallel transforms. This includes image and signal processing problems. The efficient and reliable data transformation at scale is their focus. These use cases don't need the query or transactions support provided by many NoSQL systems. They may not need the advanced features of a document store or an RDBMS as they read and write to key-value stores or distributed file systems like Amazon's Simple Storage Service (S3) or Hadoop Distributed File System (HDFS).

- Other use cases are more demanding and need more features. Big data problems like event log data and game data do need to store their data directly into structures that can be queried and analyzed, so they will need different NoSQL solutions.
- To be a good candidate for a general class of big data problems, there are some ways NoSQL solutions should do, they are as follows :

  i. Scaling linearly with growing data size by becoming an efficient with input and output.

  ii. Organizations not able to afford to hire many people to run the servers, so becoming operationally efficient.

  iii. Not every business can afford a full-time Java programmer to write on-demand queries, so it is require that reports and analyses be performed by nonprogrammers using simple tools.

  iv. Meeting the challenges of distributed computing, with consideration of latency between systems and eventual node failures.

  v. Meeting both the needs of overnight batch processing economy-of-scale and time critical event processing.

- To solve some big data problems, with enough time and effort, RDBMS can be customized. To distribute SQL queries to many processors and merge the results of the queries, applications can be rewritten. To remove joins between tables that are physically located on different nodes, databases can be redesigned. To use replication and other data synchronization processes, SQL systems can be configured. Yet these steps all take considerable time and money. It might make sense to move to a framework that has already solved many of these problems in the long run.
- Original SQL systems were revolutionary with their standardized declarative language. A developer can "declare" what data they want and yet not be concerned with how they get it or where they get the data from is meant declarative. Optimizing a query, fetching the data, and what server the data is on, are the questions which SQL developers want and need to be isolated from. We lose many of the benefits of declarative systems like SQL, unless our database isolates us from these questions.
- NoSQL systems try isolating the developers from the complexities of distributed computing. They provide interfaces that allow users to tell a cluster how many nodes a record must be read to or written from before a valid response is returned. As we move to distributed computing platforms, the goal is to keep the benefits of both declarative systems and horizontal scalability.

- We need to be able to measure these characteristics if NoSQL systems do have better horizontal scaling characteristics. So, we will see now, how horizontal scalability and NoSQL might be measured.

### 3.5.2 Understanding the Types of Big Data Problems

- A big data problem is in different types, which requires a different combination of NoSQL systems. We will find that there are different solutions once we categorize our data and determine its type. The process of differentiating data types should be similar however building our own big data classification system might be different from this example.

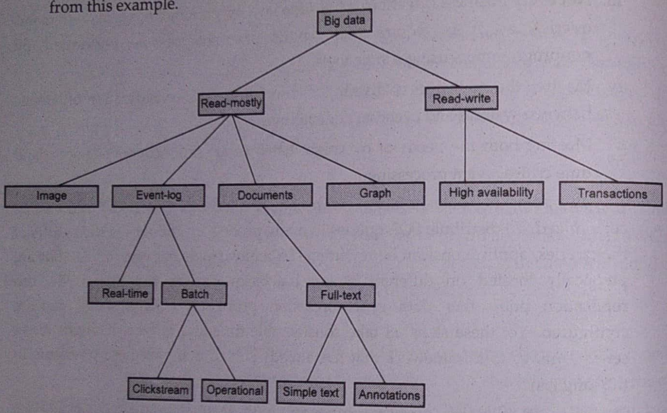


**Fig. 3.5.1 Sample big data types taxonomy**

- We will see, some ways we classify big data problems and how NoSQL systems are changing the way organizations use data.
  - **Read-mostly**

    Read-mostly data is the most common classification. It includes data which is created once and rarely altered. This kind of data can be found in data warehouse applications. It is also recognized as a set of non-RDBMS items like videos or images, published documents, event-logging data, or graph data. Things like retail sales events, hits on a website, system logging data, or real-time sensor data are included in event data.

  - **Log events**

    We can record operational events which occur in our enterprise, in a log file and include a timestamp, so we know when the event occurred. Log events can be a web page click or an out-of-memory warning on a disk drive. Many organizations, in the past, opted not to gather or analyze it as the cost and amount of event data produced were so large. But today, storage and analysis of data has become more cost-effective hence NoSQL systems are changing companies' thoughts on the value of log data.

    The ability to cost-effectively collect and store log events from all systems in your organization has lead to BI operational intelligence systems. Operational intelligence goes beyond analyzing trends in your web traffic or retail transactions. We can detect problems before they impact our customers, as it can integrate information from network monitoring systems. Cost-effective NoSQL systems can be part of good operations management solutions.

  - **Full-text documents**

    A document which contains natural-language text like the English language belongs to this category of data. We can query the entire contents of our office document in the similar way which we would query rows in your SQL system is an important aspect of document stores.

    This means that we can create new reports that combine traditional data in RDBMSs as well as the data within our office documents. It is like; we can create a single query that could extract all the authors of titles of PowerPoint slides which contains the keywords NoSQL or big data. As a result, the list of authors can be filtered with a list of titles in the HR database, which can be shown which people had the title of Data Architect or Solution Architect.

    This will be a good example about how organizations, for training and mentorship, are trying to tap into the hidden skills that already exist within an organization. Integrating documents into what can be queried is opening new doors in knowledge management and efficient staff utilization.

    We might encounter many different flavors of big data. We will see how using a shared-nothing architecture can help with most of our big data problems, if they are read-mostly or read/write data, as we move forward.

## 3.5.3 Analyzing Big Data with Shared Nothing Architecture

- The resources can be shared between computer systems by three ways. By shared RAM, shared disk, and shared-nothing. Fig. 3.5.2 shows a comparison of these three distributed computing architectures.
- One alternative i.e. a shared-nothing architecture is most cost effective in terms of cost per processor when we use commodity hardware.

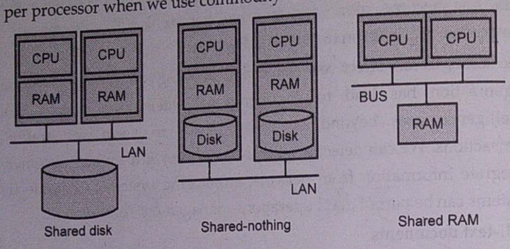


**Fig. 3.5.2 Three ways to share resources**

- These architectures work to solve big data problems with different types of data. Out of the architectural data patterns, i.e. row store, key-value store, graph store, document store, and Bigtable store, there are only two lend themselves to cache-friendliness. Those are key-value store and document store. Bigtable stores row-column identifiers are similar to key-value stores hence they scale well on shared nothing architectures. However, row stores and graph stores are not cache-friendly. Because they don't allow a large BLOB to be referenced by a short key that can be stored in the cache.
- The entire graph should be in main memory for graph traversals to become fast. This is why when we have enough RAM to hold the graph, graph stores work most efficiently. Graph stores will try to swap the data to disk, that results in decrease graph query performance by a factor of 1,000, if we cannot keep our graph in RAM. Moving to a shared-memory architecture, where multiple threads all access a large RAM structure without the graph data moving outside of the shared RAM, this could be the only way to combat the problem.
- We should look for an alternative to a shared-nothing architecture, if we have over a terabyte of highly connected graph data and we need real-time analysis of this graph, this could be the rule of thumb. To hold our graph in RAM, a single CPU with 64 GB of RAM won't be sufficient. Even if we work hard to only load the

necessary data elements into RAM, our links may traverse other nodes that need to be swapped in from disk. This will slow down our graph queries. We'll look into alternatives to this in a case study later in this chapter.

- The first step is, knowing the hardware options available to big data but distributing software in a cluster is also important. We will see how software can be distributed in a cluster.

## 3.5.4 Choosing Distribution Models : Master-Slave Versus Peer-to-peer

- There are two main models from a distribution perspective, one is master-slave and another is peer-to-peer. When a request is made, distribution models determine the responsibility for processing data and when we are looking at a potential big data solution, understanding the pros and cons of each distribution model is important.
- Master-slave models are less resilient to failure than the Peer-to-peer models. Some master-slave distribution models have single points of failure that might impact our system availability, that time we might need to take special care while configuring these systems.
- Distribution models get to the heart of the question who's in charge here ? There are two ways to answer this question: one node or all nodes. In the master-slave model, one node is in master. When there's no single node with a special role in taking charge, you have a peer-to-peer distribution model.
- Fig. 3.5.3 shows how these models each work.

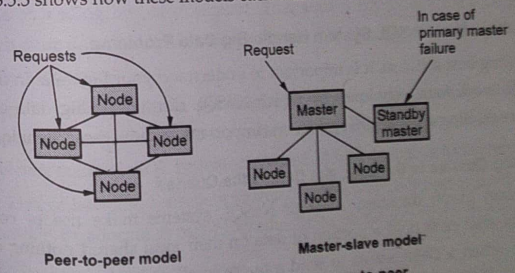


**Fig. 3.5.3 Master-slave vs peer-to-peer**

- We will see the trade-offs. The role of managing the cluster is done on a single master node with a master-slave distribution model. To lower the probability that it crashes, the node can run on specialized hardware such as RAID drives. The cluster can also be configured with a standby master that's continually updated from the master node. But the challenge is that, without jeopardizing the health of the cluster it's difficult to test the standby master. Failure of the standby master to take over from the master node is a real concern for high-availability operations.

- The responsibility of the master to each node in the cluster is being distributed by the Peer-to-peer systems. Since we can remove any node in the cluster and the other nodes will continue to function, in this situation, testing is much easier. The disadvantage of peer-to-peer networks is that there's an increased complexity and communication overhead that must occur for all nodes to be kept up to date with the cluster status.
- The initial versions of Hadoop were designed to use master-slave architecture with the NameNode of a cluster being responsible for managing the status of the cluster. NameNodes usually don't deal with any MapReduce data themselves. Their job is to manage and distribute queries to the correct nodes on the cluster. To remove single points of failure from a Hadoop cluster, Hadoop 2.x versions are designed.
- Using the right distribution model will depend on our business requirements : A peer-to-peer network might be the best solution, if high availability is a concern. And the simpler master-slave model might be best, if we can manage our big data using batch jobs that run in off hours. As we move to the next section, we will see how MapReduce systems can be used in multiprocessor configurations to process our big data.

### 3.5.5 Four Ways that NoSQL System Handle Big Data Problems

- For finding best solution, it is important to understand your big data. In this section we will check four techniques by which NoSQL can handle big data challenges. While evaluating any NoSQL system it is important to know these techniques.

#### 3.5.5.1 Moving Queries to the Data, Not Data to the Queries

- Other than graph database, most of NoSQL systems make use of commodity hardware that each hold a portion of data on their local shared nothing drives. In scenario when a client wants to send a query to all nodes in cluster which holds data, it is always efficient to send query to every node rather than sending entire dataset to central processor.
- The traditional database system as of now could not distribute queries and aggregate query result. This simple rule helps to know that NoSQL database have

performance advantages over other systems which were not designed to distribute queries to all the nodes in cluster.

- Consider a RDBMS system having tables which are distributed over two different nodes. In order for the SQL query to execute properly, information about rows on one table need to be moved to other node. In case the size of table is large the movement of data will take longer time and that will result in slow query processing. By keeping all data within every data node as logical documents will make only query and final output required to be moved over a network. This will make big data queries faster.

#### 3.5.5.2 Using Hash Rings to Evenly Distribute Data on a Cluster

- Finding out a consistent technique of allocating document to a processing node is one of the most challenging problems faced with distributed database. The good technique to distribute a network load evenly is using hash ring technique which evenly distribute load of big data over multiple servers by generating 40 characters key randomly.
- Since hash ring technique consistently determine how a part of data can be allocated to particular processor, that is why in big data solution hash ring technique is common.
- To determine which node the document should be allocated, hash ring takes hash value of leading bits of document. This allows any node of the cluster to know which node data is on and how to adjust to new document assignment technique as the data increases.
- Keyspace management is partitioning keys into specific ranges and allocating various key ranges to particular nodes. In order to manage distributed processing problems, many NoSQL systems including MapReduce uses technique of keyspace.
- The idea of hashing, consistent hashing and key-value stores is used by hash ring technique to allocate a part of document to particular node in NoSQL database cluster.
- On the basis of 40 character random key, every document is allocated to a node. This concept can be further extended as to include need that part of document must be stored on multiple nodes.

### 3.5.5.3 Using Replication to Scale Reads

- By using replication it is possible to scale read requests horizontally. In most of the cases this replication technique works well.
- Read of the same record after a write is one of the most common operations. There will not be any problem if in case a client does a write and the read from that same node. In case before the updation happens there is a read from a replica node then problem will occur. This problem is termed as inconsistent read.
- This can be resolved by allowing read to the same write node only after write has been done. At application layer this logic can be added to state management system or session.
- In case of large number of nodes permits write, almost all distributed databases relax rule of database consistency. In order to get faster read/write consistency, this needs to be tackle at application layer.

### 3.5.5.4 Letting the Database Distribute Queries Evenly to Data Nodes

- It is important to separate worry of query evaluation from query execution, to get higher performance from queries spanning different nodes.
- In place of moving data to query, moving query to data is an important part of NoSQL big data strategies. In such scenario, database server is responsible for moving the query and database is responsible for distribution of query and waiting for response from all nodes.
- This technique is similar to the idea of federated search. Federated search takes single query and distributes this query to distinct server and combining the result together. In some scenarios, servers may be located in different geographic locations. In this case, query is being sent to single cluster which is not only performing searching of query on single local cluster but also performs delete and update operations.

## University Questions with Answers

**Winter - 2016 (IT)**

Q.1 Write a short note on NoSQL databases. **(Refer section 3.1)** [3]

**Summer - 2017 (CSE)**

Q.2 What is NoSQL Database ? Explain in brief various types of NoSQL databases in practice. **(Refer sections 3.1 and 3.4)** [7]

**Winter - 2017 (CSE)**

Q.3 Write short note on

i) Document Oriented Database

ii) Graph based Database **(Refer section 3.4)** [4]

**Summer - 2018 (CSE)**

Q.4 Explain NoSQL. **(Refer section 3.1)** [3]

**Winter - 2018 (IT)**

Q.5 What is NoSQL ? List out the features of NoSQL. Explain types of NoSQL databases in brief. **(Refer sections 3.1 and 3.4)** [7]

**Winter - 2018 (CSE)**

Q.6 What is NoSQL database ? Discuss key characteristics and advantages of NoSQL database. **(Refer section 3.1)** [4]

Q.7 Discuss different types of NoSQL databases with proper example. **(Refer section 3.4)** [4]

**Summer - 2019 (IT)**

Q.8 List out the features of NoSQL. Explain types of NoSQL databases in brief. **(Refer sections 3.1 and 3.4)** [7]

**Winter - 2019 (IT)**

Q.9 Write a short note on NoSQL databases. **(Refer section 3.1)** [3]

# 4 Mining Data Stream

**Syllabus**

*Introduction to Streams Concepts, Stream Data Model and Architecture, Stream Computing, Sampling Data in a Stream, Filtering Streams, Counting Distinct Elements in a Stream, Estimating moments, Counting oneness in a Window, Decaying Window, Real time Analytics Platform (RTAP) applications, Case Studies, Real Time Sentiment Analysis, Stock Market Predictions. Using Graph Analytics for Big Data : Graph Analytics.*

**Contents**

## 4.1 Introduction

- Data Stream Mining is an activity of collecting insights from continuous high-speed data records which comes to the system in a stream.

### 4.1.1 Data Stream Mining Fulfill the Following Characteristics

- **Continuous Stream of Data :** High amount of data in an infinite stream. We do not know the entire dataset.
- **Concept Drifting :** The data keep evolving or changing over time.
- **Volatility of Data :** The system does not store the data received (Limited resources). When data is analysed it's discarded or summarised.

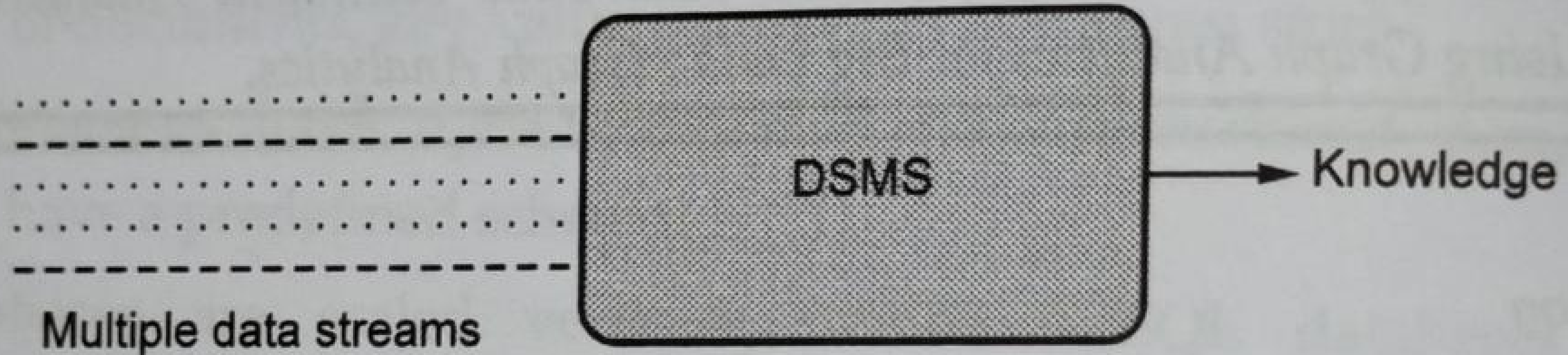


Fig 4.1.1 Data stream mining

## 4.2 Stream Data Architecture

- A streaming data architecture is a framework of software components construct to absorb and process large amount of streaming data from numerous sources. Compare to traditional data solutions which focused on searching and editing data in batches, a streaming data architecture ingest data immediately as it is produce, is persistence in its storage, and may include various additional components per use case - such as tools for real-time processing, data manipulation and analytics.
- Streaming architectures must be able to address for the unique feature of data streams, which tend to make huge amounts of data (terabytes to petabytes) that it is at best semi-structured and requires significant pre-processing and ETL to become useful.

### 4.2.1 Why Streaming Data Architecture ? Benefits of Stream Processing

- Stream processing is to be a 'niche' technology used only by a small segment of organizations. However, with increase growth of SaaS, IoT and ML, big firms and corporates.
- In today's world, it is impossible for an organization of not having an app or

webpage; this increases the traffic, and with increasing size for complex and real-time analytics, the need to adopt modern data infrastructure is quickly becoming mainstream.

- While traditional batch frameworks can be enough for smaller scales, stream processing includes several benefits that other data platforms cannot :
- **Able to deal with never-ending streams of events :** Several kinds of data, naturally structured, in this way. Traditional batch processing tools acquire stopping the stream of events, capturing batches of data and integrate the batches to collect overall conclusions. Stream processing provides immediate insights from large volume of streaming data, even if it is difficult to capture and merge data from various streams.
- **Real-time or near-real-time processing :** Most enterprises adopt stream processing to allow real time data analytics. Even though with high performance database real time analytics is possible, but the data is moved to stream processing models.
- **Detecting patterns in time-series data :** Finding patterns over time, for example looking for trends in website traffic data, acquire data to be continuously processed and analyzed. Since batch processing divides the data into multiple batches, it makes pattern detection more difficult.
- **Easy data scalability :** Increasing data volumes can break a batch processing system, requiring to provision additional resources or changing framework. With single stream CPU, modern stream processing infrastructure is highly scalable and can deal with large amount of data per second. This increase data volumes without infrastructure changes.

## 4.2.2 Streaming Architecture Components

### 1. The Message Broker / Stream Processor

- This takes data from source named as producer, translate it into standard message format, and and streams it on an ongoing basis. The messages passed on by the broker can then be listened and consumed by other components.

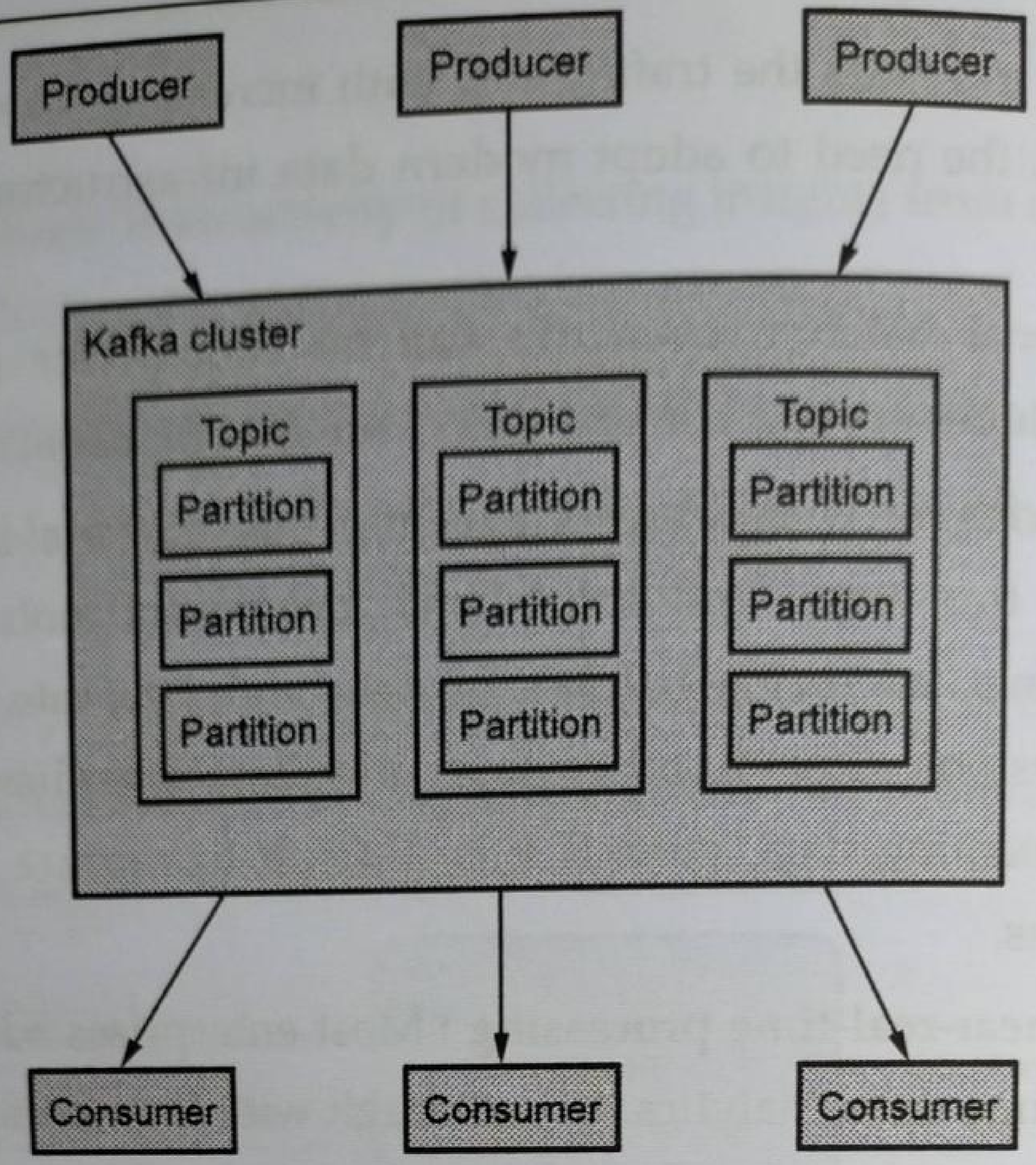


**Fig. 4.2.1 Message broker / stream processor**

**2. Batch and Real-time ETL tools**

- Data streams from one or more message brokers need to be accumulated, reconstruct and restructured before data can be analyzed with SQL-based analytics tools.
- This should be possible by an ETL tool or platform which gets queries from users, takes events from queues of message and applies the query, to gather an outcome - regularly performing additional joins, transformations on aggregation on the data. The outcome might be an API call, an activity, a visualization, an alert, or in certain stages another data stream.

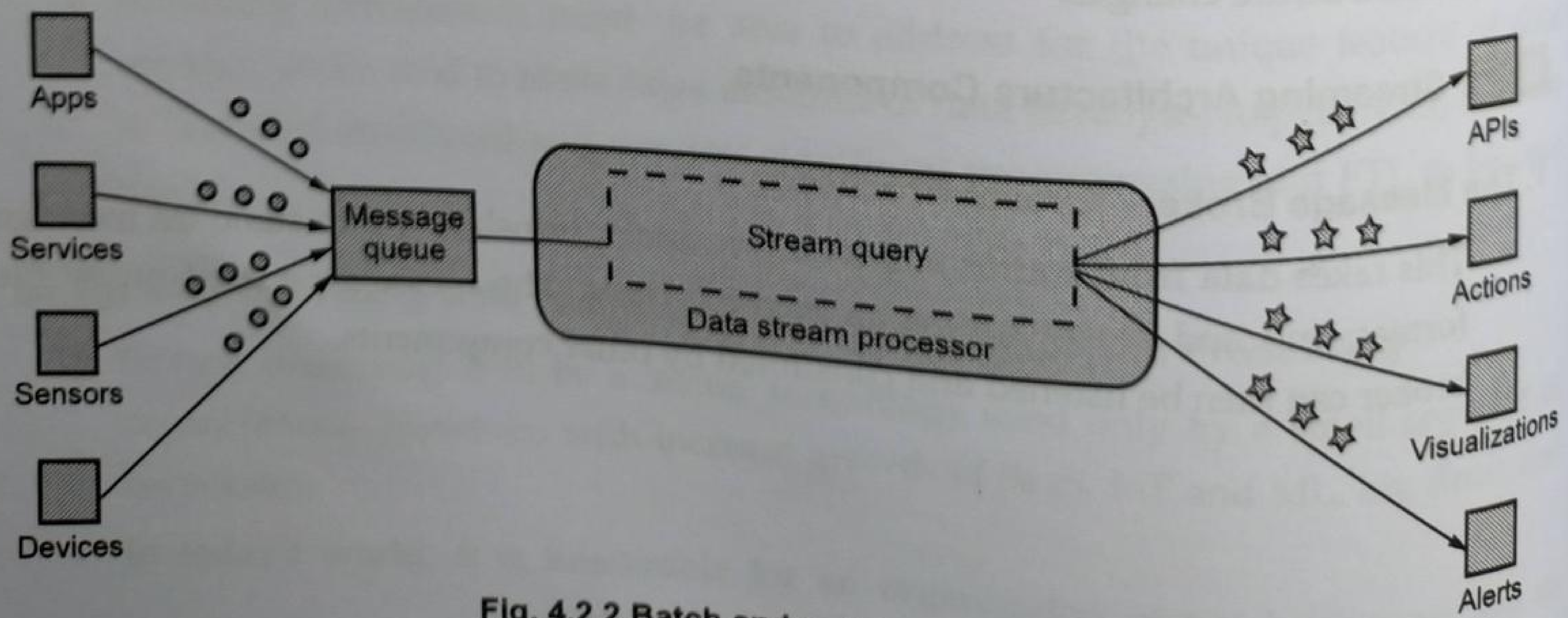


**Fig. 4.2.2 Batch and real-time ETL tools**

**3. Data Analytics / Server less Query Engine**

- After streaming data is prepared for consumption by the stream processor, it must be visualizing providing value. There are many numerous approaches to streaming data analytics.

**4. Streaming Data Storage**

- As every company has a massive data to store, they opt for cheap storage options. So they choose storing data in streaming way.

## 4.3 Stream Computing

- In this, the data is taken in streams; data processing and streaming it back out as a single flow. Stream processing utilize software algorithm that visualize the data continuously while it streams fastly and precision when managing data taking care of and investigation.

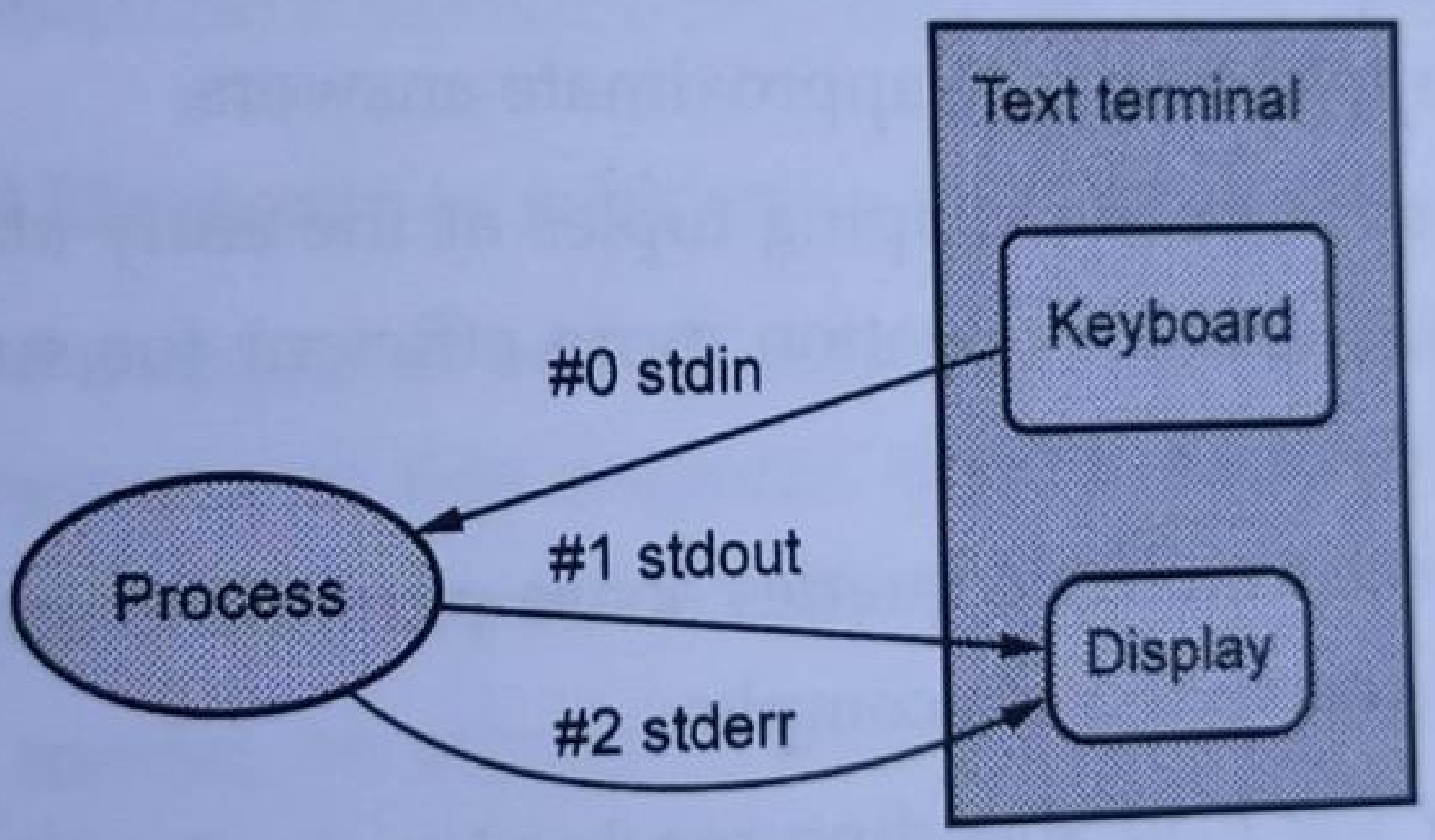


Fig. 4.3.1 Stream computing

### 4.3.1 Sampling Data in a Stream

- The sample in data stream is taken much smaller than the whole stream. This can be designed to retain many relevant features of the stream. The same can be used to calculate many relevant combinations on the stream.
- Unlike sampling from a stored data set, stream sampling should be performed on the web, when the data arrives. Any component that isn't stored inside the sample is lost everlastingly, and can't be recovered.

### 4.3.2 Filtering Stream

- Due to the nature of data streams, stream filtering is one out of the major useful and practical approaches to efficient stream evaluation.

### 4.3.2.1 Implicit Filtering in Data Stream Management Systems

- Data Stream Management Systems cope with the high rates and the bursty nature of streams in a number ways in order to guarantee stability under heavy loads.
- Some of them employ various load-shedding techniques which reduce the load by processing only a fraction of the items from the stream and discarding others without any processing.
- The Aurora DSMS employs random and semantic load shedding techniques to deal with the unpredictable nature of data streams, where semantic load shedding makes use of tuple utility computed based on quality-of-service (QoS) parameters.
- Intuitively, the system drops tuples that are believed to be less important for stream evaluation that others. QoS of the system is captured by a number functions: latency graph, which specifies the utility of a tuple as a function of tuple propagation through the query plan; value-based graph, which specifies which values of the output are more important than others; and loss tolerance graph, which describes how sensitive the application is to approximate answers.
- It is clear that the strategy of dropping tuples at the early stages of the query plan makes the process of query evaluation more efficient for subsequent operators in the plan.
- However in case when multiple queries share parts of their plans, the question of where to shed load becomes more complex.
- Tuples dropped by the load shedding mechanism in the TelegraphCQ DSMS are collected by the Data Triage components into data synopses and are later combined with the standard query that results which in better grasp the properties of the entire input. Thus load sharing mechanisms is best executed.

### 4.3.2.2 Explicit Filtering of Data Streams

- As it was mentioned previously, the implicit filtering techniques may often have a negative impact on a variety of data streams analyses problems, such as, for example, computation of sketches and samples of distinct items for estimation of quantiles, heavy hitters and other properties of a stream.
- Other problems in this category include estimation of IP network flow sizes, detection of most prevalent substrings in the network for worm signature generation and others.
- Below we give some more detailed examples of such explicit filtering procedures on data streams.

- Fine-grained estimation of network traffic (flows) volume is very important in various network analysis tasks. The work presented in it offers a solution by proposing the threshold sampling algorithm that generates a sample of stream items with guarantees on estimated flow sizes.
- At each step of sample generation, the procedure maintains a value of a threshold, which is compared to the size of the item, and the decision is made on whether the item of the stream should be filtered out or retained in the sample. Depending on the version of the algorithm, the threshold value can remain constant throughout the process or be dynamically adjusted to ensure the size of produced sample.
- Due to the nature of this sampling algorithm, with a number of various parameters, such as a number of items in the final sample, item size, threshold value, count of items larger than the threshold value, etc., playing a role in how the procedure is executed and what results it produces, its results would be seriously compromised if it was to be evaluated on a stream which was a product of random load shedding.

## 4.3.3 Counting Distinct Elements in a Stream

- Let a = a1 ,... , an be a sequence of n elements from the domain [m] = {1,... ,m}. The zeroth-frequency moment of this sequence is the number of distinct elements that occur in the sequence and is denoted F0 = F0(a).
- In the data stream model, an algorithm is considered efficient if it makes one (or a small number of) passes over the input sequence, uses very little space, and processes each element of the input very quickly. In our context, a data stream algorithm to approximate F0 is considered efficient if it uses only poly (1/∈, log n, log m) bits of memory, where 1 ± ∈ is the factor within which F0 must be approximated.
- Let, ϵ δ > 0 be given.
- An algorithm A is said to (ϵ, δ)-approximate F0 if for any sequence a = a1,... ,an, with each ai ϵ [m], it outputs a number F˜0 such that Pr[ ! ! ! F0 – F˜0 ! ! ! ≤ ∈ F0] ≥ 1–δ,
- Here the probability is assumed over the internal coin tosses of A.
- Two main parameters of A are of interest: the workspace and the time to process each item. We study these quantities as functions of the domain size m, the number n of elements in the stream, the approximation parameter ϵ, and the confidence parameter δ.

### 4.3.4 Estimating Moments

- Alon-Matias-Szegedy (AMS) Algorithm
  - Ex Stream : a, b, c, b, d, a, c, d, a, b, d, c, a, a, b
  - n=15; a(x5), b(x4), c(x3), d(x3); 2nd Moment = 59
  - **Estimate = n * (2 * X.value – 1)**
  - Pick random positions X1(3rd), X2(8th), X3(13th)
  - X1.element = "c", X1.value = 3 (# of "c" in the set from 3rd place)
  - Estimate for X1= 15*(2*3-1) = 75
  - Estimates for X2 = 15*(2*2-1) = 45, (# "d" beyond 8th place = 2)
  - Estimate for X3 = 45, (# "a" beyond 13th place = 2)
  - Average for X1, X2, X3 = 165/3 = 55 (close to 59)
- In case of infinite streams
  - As we store one variable per randomly chosen position so the challenge is not selecting 'n'.
  - Selecting the position is challenge.
- Strategy for position selection assuming we have space to store "s" variables and we have seen "n" elements.
  - First "s" positions are chosen.
  - When (n+1)th token arrives, random selection takes place.
  - If (n+1) is selected discard from existing n position randomly and insert new element with value 1.

### 4.3.5 Counting Oneness in a Window

- Let's suppose a window of length N on a binary stream.
- For questions regarding queries of the form "how many 1's are there in the last k bits?" for any $k \leq N$, we use the DGIM algorithm.
- The basic version of the algorithm uses $O(\log_2 N)$ bits to represent a window of N bits, and allows us to estimate the number of 1's in the window with an error of no more than 50 %.
- To start, every bit of the stream has a timestamp, the location in which it arrives. The first bit has timestamp 1, the second has timestamp 2, etc.
- For distinct positions in window of length N, we will represent timestamps modulo N, and then we represent it by $\log_2 N$ bits.

- We divide the window into buckets, 5 consisting of :-
  1. The timestamp of its right (most recent) end.
  2. The number of 1's in the bucket. Then number must be a power of 2.
- For bucket, we need $\log_2 N$ bits for representation of the timestamp (modulo N) in its right end. For representation of the number of 1's we only need $\log_2 \log_2 N$ bits.
- Now, j which is at most $\log_2 N$, so it wants $\log_2 \log_2 N$ bits. Thus, O(log N) bits is used to represent a bucket.
- There are six rules that must be followed when representing a stream by buckets.
  - o In bucket, right position is always 1.
  - o For some bucket, there may be a 1.
  - o No position is in more than one bucket.
  - o There can be single or two buckets of any given size, up to some maximum size.
  - o Every size should have power of 2.
  - o While moving left side, the size of the bucket cannot decrease.

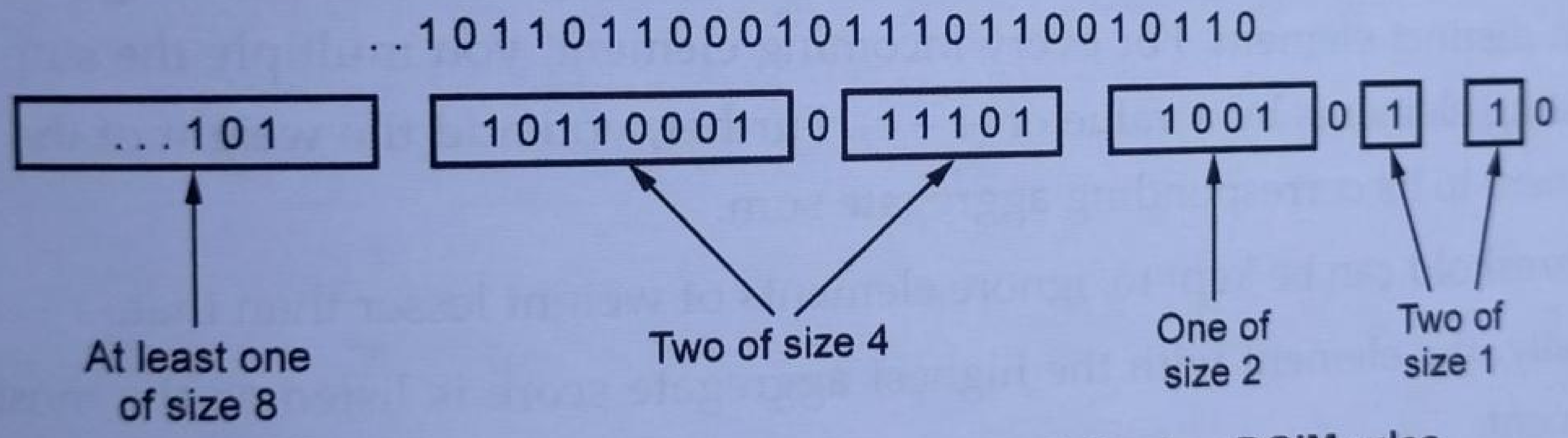


**Fig. 4.3.2 A bit-stream divided into bucket following DGIM rules**

## 4.3.6 Decaying Window

- This algorithm allows you to identify the most popular elements (trending, in other words) in an incoming data stream.
- The decaying window algorithm not only tracks the most recurring elements in an incoming data stream, but also discounts any random spikes or spam requests that might have boosted an element's frequency.
- In a decaying window, you assign a score or weight to every element of the incoming data stream. Further, we calculate the aggregate sum for each distinct element by adding all the weights assigned to that element. The element with the highest total score is listed as trending or the most popular.
- Then,
  1. Assign each element with a weight/score.

2. Calculate aggregate sum for each distinct element by adding all the weights assigned to that element.

- In a decaying window algorithm, you assign more weight to newer elements. For a new element, you first reduce the weight of all the existing elements by a constant factor k and then assign the new element with a specific weight.
- The aggregate sum of the decaying exponential weights can be calculated using the following formula :

$$\sum t - 1i = 0 \ at - i(1 - c)i$$

Here, c is usually a small.

- Whenever a new element, say at + 1 , arrives in the data stream you perform the following steps to achieve an updated sum :
  1. Multiply the current sum/score by the value (1 – c).
  2. Add the weight corresponding to the new element.
- Weight decays exponentially over time.

  In a data stream consisting of various elements, you maintain a separate sum for each distinct element. For every incoming element, you multiply the sum of all the existing elements by a value of (1 – c). Further, you add the weight of the incoming element to its corresponding aggregate sum.
- A threshold can be kept to, ignore elements of weight lesser than that.
- Finally, the element with the highest aggregate score is listed as the most popular element.
- **Example**

For example, consider a sequence of twitter tags below :

fifa, ipl, fifa, ipl, ipl, ipl, fifa

Also, let's say each element in sequence has weight of 1.

Let's c be 0.1

The aggregate sum of each tag in the end of above stream will be calculated as below :

**fifa**

fifa - 1 * (1-0.1) = 0.9

ipl - 0.9 * (1-0.1) + 0 = 0.81 (adding 0 because current tag is different than fifa)

fifa- 0.81 * (1-0.1) + 1 = 1.729 (adding 1 because current tag is fifa only)

ipl - 1.729 * (1-0.1) + 0 = 1.5561

ipl - 1.5561 * (1-0.1) + 0 = 1.4005

ipl - 1.4005 * (1-0.1) + 0 = 1.2605

fifa- 1.2605 * (1-0.1) + 1 = **2.135**

**ipl**

fifa - 0 * (1-0.1) = 0

ipl - 0 * (1-0.1) + 1 = 1

fifa- 1 * (1-0.1) + 0 = 0.9 (adding 0 because current tag is different than ipl)

ipl - 0.9 * (1-0.01) + 1 = 1.81

ipl - 1.81 * (1-0.01) + 1 = 2.7919

ipl - 2.7919 * (1-0.01) + 1 = 3.764

fifa - 3.764 * (1-0.01) + 0 = **3.7264**

In the end of the sequence, we can see the score of **fifa is 2.135** but **ipl** is **3.7264.**

So, **ipl** is more trending then **fifa.**

Even though both of them occurred same number of times in input there score is still different.

- **Advantages of Decaying Window Algorithm :**
  1. Sudden spikes or spam data is taken care.
  2. New element is given more weight by this mechanism, to achieve right trending output.

## 4.4 Real Time Analytics Platform Applications

### 4.4.1 Following Applications are of RTAP

1. Fraud detection systems for online transactions.
2. Log analysis for understanding usage pattern.
3. Click analysis for online recommendations.
4. Social media analytics.
5. Push notifications to the customers for location-based advertisements for retail.
6. Action for emergency services such as fires and accidents in an industry.
7. Any abnormal measurements require immediate reaction in healthcare monitoring.

## 4.4.2 Some Real-time RTAP

**1. Apache Samza**

- Apache Samza is an open-source, near-real time, asynchronous computational framework for stream processing developed by the Apache Software Foundation in Scala and Java.
- Samza allows users to build stateful applications that process data in real-time from multiple sources including Apache Kafka.
- Samza provides fault tolerance, isolation and stateful processing. Samza is used by multiple companies. The biggest installation is in LinkedIn.

**2. Apache Flink**

- Apache Flink is an open-source, unified stream-processing and batch-processing framework developed by the Apache Software Foundation. The core of Apache Flink is a distributed streaming data-flow engine written in Java and Scala.
- Flink provides a high-throughput, low-latency streaming engine as well as support for event-time processing and state management.
- Flink does not provide its own data-storage system, but provides data-source and sink connectors to systems such as Amazon Kinesis, Apache Kafka, Alluxio, HDFS, Apache Cassandra, and ElasticSearch.

**3. SAS Event Stream Processing**

- For processing and streaming of data SAS event stream processing is used. Based on the streaming events i.e continuous queries of data, analytical models and business rules, Event actions are triggered.

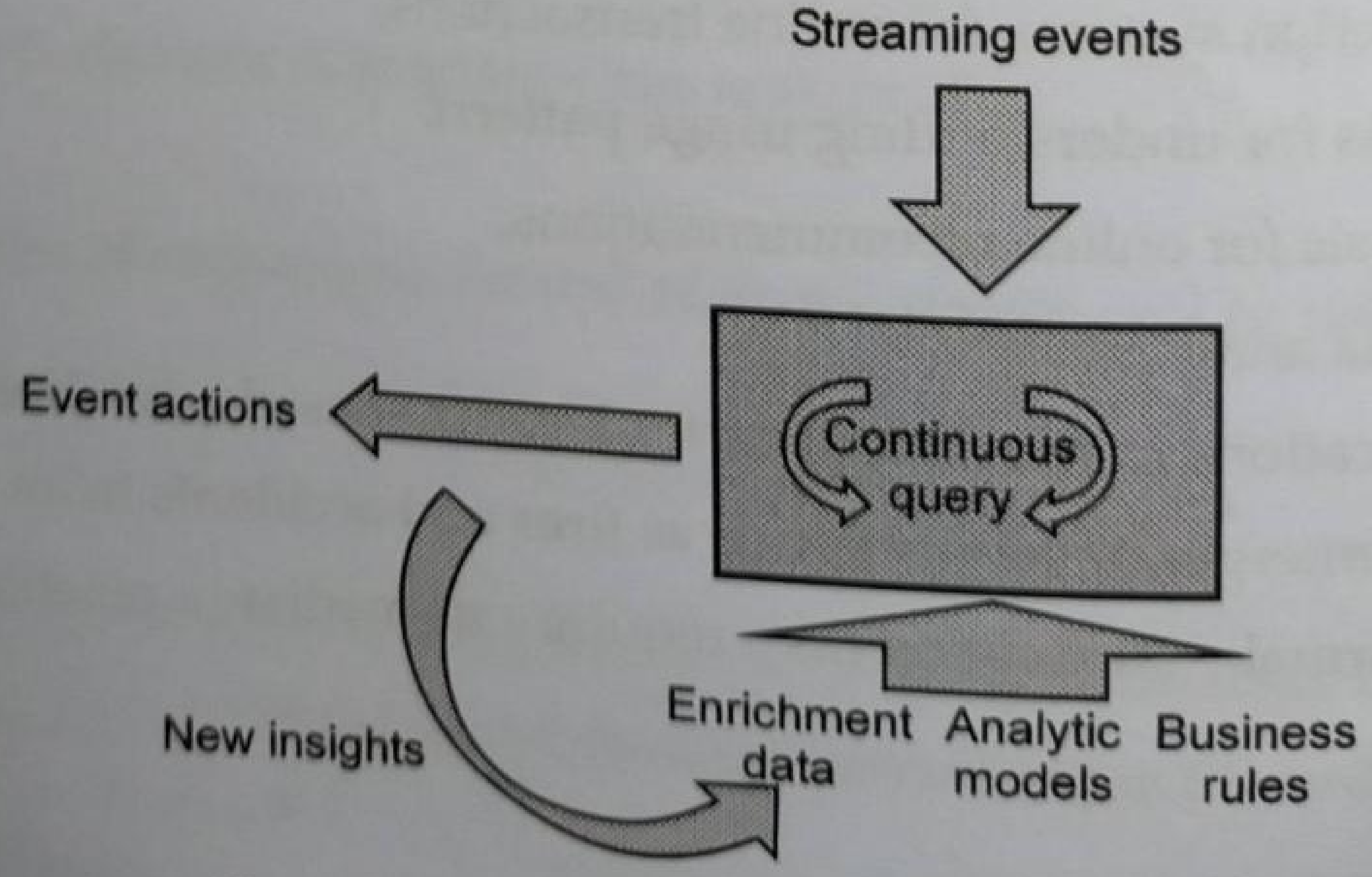


**Fig. 4.4.1 SAS event stream processing**

## 4.5 Case Studies

### 4.5.1 Real Time Sentiment Analysis

- Real-time sentiment analysis is an AI-powered solution to track mentions of your brand and products, wherever they may appear, and automatically analyze them with almost no human input needed.

#### 4.5.1.1 What Is Sentiment Analysis ?

- Sentiment analysis is a text analysis tool that uses machine learning with Natural Language Processing (NLP) to automatically read and classify text as positive, negative, neutral, and everywhere in between. It can read all manner of text (online and elsewhere) for opinion and emotion - to understand the thoughts and feelings of the writer.
- Some of the benefits of performing real-time sentiment analysis :
  - o Marketing campaign success analysis
  - o Prevention of business disasters
  - o Instant product feedback
  - o Stock market predictions.
- Sentiment analysis is also known as opinion mining.
- Attempts to identify the opinion/sentiment that a person may hold towards an object.
- Basic components of an opinion :
  - o **Opinion holder :** The person or organization that holds a specific opinion on a particular object.
  - o **Object :** On which an opinion is expressed.
  - o **Opinion :** A view, attitude, or appraisal on an object from an opinion holder.
  - o Opinion mining tasks :
    - o At the document (or review) level :
      - ▪ Task : Sentiment classification of reviews
      - ▪ Classes : Positive, negative, and neutral
    - o At the sentence level :
      - ▪ Task 1 : Identifying subjective/opinionated sentences
        - Classes : Objective and subjective (opinionated)
      - ▪ Task 2 : Sentiment classification of sentences.
        - Classes : positive, negative and neutral.

- Opinion mining tasks :
  - At the feature level :
    - Task 1 : Identify and extract object features that have been commented on by an opinion holder (e.g., a reviewer).
    - Task 2 : Determine whether the opinions on the features are positive, negative or neutral.
    - Task 3 : Group feature synonyms.
      - Produce a feature-based opinion summary of multiple reviews.
- Two **types of evaluations** :
  - Regular Opinions : Sentiment/opinion expressions on some target entities, e.g., products, events, topics, persons.
    - Direct opinions :
      - E.g., "The touch screen is really good"
    - Indirect opinions :
      - E.g., "After taking the drug, my pain has gone."
  - Comparative Opinions : Comparisons of more than one entity.
- An Example :
  - Id : **Abc123** on **5-1-2008** *"I bought an **iPhone** a few days ago. It is such a nice phone. The **touch screen** is really cool. The **voice quality** is clear too. It is much better than my old **Blackberry**, which was a terrible **phone** and so **difficult to type** with its tiny keys. However, my mother was mad with me as I did not tell her before I bought the phone. She also thought the phone was too **expensive**, ..."*
  - In quintuples :

    (iPhone, GENERAL, +, Abc123, 5-1-2008)

    (iPhone, touch_screen, +, Abc123, 5-1-2008)
- The below Fig. 4.5.1 shows the process of sentiment analysis.

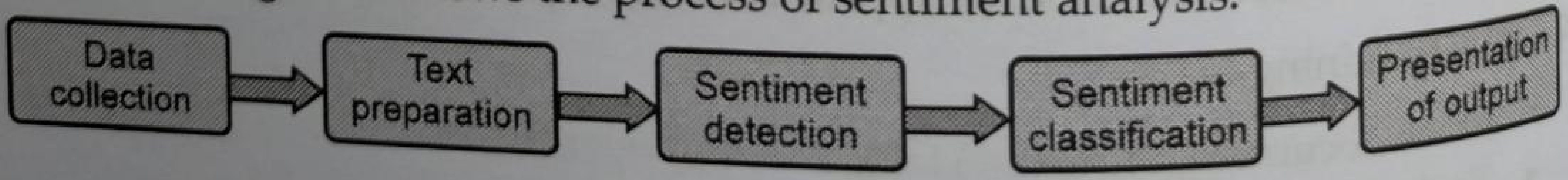


**Fig. 4.5.1 Sentiment analysis process**

- Real-time sentiment gives you a window into what your customers and the public at large are expressing about your brand "right now" for targeted, minute-by-minute analysis, and to follow brand sentiment over time.

- Let's take a look at how easy it can be to perform real-time sentiment analysis.

1. **Set your goals**
   - What sentiments to compare ? Sentiments of customer's brand with that of the market products. It's easy to set end to end goals for real time analysis.

2. **Gather your data**
   - By various ways we can gather data; through various tools we can integrate it. Web mining and web scraping tools, like Dexi, Content Grabber, and Pattern allow you to link APIs or extract content directly from the web into CSV or Excel files, and more.

3. **Clean your data**
   - Repetitive data like emojis, banner ads, etc. and other "noise" in social media and sites are to be removed.
   - The email cleaner automatically removes email signatures, legal notices, and previous replies to give you only the most recent message in the chain :
   - Also, the opinion unit's extractor breaks up sentences or whole pages of text into individual sentiments or thoughts called "opinion units". It can divides many pages and a large number of opinion units automatically to prepare your data for analysis.

4. **Analyze and visualize sentiments in real time**
   - All data is to be analyzed in terms of sentiments. Then the related data is used to process the real time scenario.
   - At the end the results are tested. Various ML models are used to execute real time sentiments.
   - As machine learning models are usually mathematical, there is a need to somehow map the texts from letters into numbers. This is so called vectorization and for our purposes we've chosen TFIDF vectorization which splits the text into words and assigns each word a weight, depending on its importance.

### 4.5.2 Stock Market Predictions

- Stock market prediction has been an area of intense interest due to the potential of obtaining a very high return on the invested money in a very short time.
- Stream data is widely used in all digital mediums and communications.
- Continuous variation in stock market lays the user into the situation of confusion.

#### 4.5.2.1 The Necessity of Data Mining in Stock Market

- Data Mining is a process of abstracting unaware, potential and useful information and knowledge from plentiful, incomplete, noisy, fuzzy and stochastic data. These information and knowledge can't be achieved relying on a simple data search.
- The key of data mining include three parts :

  a) Data, b) Information c) And business decisions.
- The stock market data is stream data, at the same time, the stock market data shares sequential nature, which can be used to analyze stream data time-series pattern mining methods.
- Stock market data depends on various parameters. The variance of the data stream is non-linear and it is difficult to set the regularity. Lots of data is gathered at glance of time.
- Following principles are used for Stock Market Data Mining Algorithm :
  1. Single linear scanning.
  2. Low space-time complexity.
  3. Adapting to the dynamic changes and velocity of data, the results approximation is to a higher degree.
  4. Powerful ability of processing nonlinear problems.
  5. Effectively dealing with noise and null.
  6. Anytime online response to the request of the excavation.
  7. Summary data structure will not only support the goal of the current calculation method, but also support other calculations.

### 4.5.3 Graph Analystics and Big Data

- Graph analytics, which is an analytics alternative that uses an abstraction called a graph model.
- Graph analytics is an alternative to the traditional data warehouse model as a framework for absorbing both structured and unstructured data.

#### 4.5.3.1 Introduction Graph Analytics

- Graph analytics is worth if it makes it easy to analyse the data.
- What constitutes graph analytics ?
  - Types of problems those are suited to graph analytics.
  - Types of questions that are addressed using graph analytics.
  - Types of graphs that are commonly encountered.
  - The degree of prevalence within big data analytics problems.

- This motivates an understanding of its utility and flexibility for discovery-style analysis in relation to specific types of business problems, how common those types of problems are, and why they are nicely abstracted to the graph model.
- Representation of individual entities and numerous kinds of relationships that connect those entities is used as basis for graph analytics.
- More precisely, it employs the graph abstraction for representing connectivity, consisting of a collection of vertices (which are also referred to as nodes or points) that represent the modeled entities, connected by edges (which are also referred to as links, connections, or relationships) that capture the way that two entities are related.
- More simple the model, more flexible it is.
- A simple unlabeled undirected graph, in which the edges between vertices neither reflect the nature of the relationship nor indicate their direction, has limited utility.
- Nodes and edges can be defined as :
  - o Vertices can be labeled to indicate the types of entities that are related.
  - o Edges can be labeled with the nature of the relationship.
  - o Edges can be directed to indicate the "flow" of the relationship.
  - o Weights can be added to the relationships represented by the edges.
  - o Additional properties can be attributed to both edges and vertices.
  - o Multiple edges can reflect multiple relationships between pairs of vertices.
- Almost any type of entity and relationship can be represented in a graph model, which means two key things: the process of adding new entities and relationships is not impeded when new datasets are to be included.

### 4.5.3.2 Choosing Graph Analytics

- Following parameters needs to be taken care of for choosing it :
  1. Connectivity
  2. Undirected discovery
  3. Absence of structure
  4. Flexible semantics
  5. Extensibility
  6. Ad hoc nature of the analysis : There is a need to run ad hoc queries to follow lines of reasoning.
  7. Predictable interactive performance.

**Review Questions with Answers**

Q.1 What is data? Explain data stream? **(Refer section 4.1)** [3]

Q.2 Explain stream data architecture in detail ? **(Refer section 4.2)** [7]

Q.3 Name the components of streaming architecture. **(Refer section 4.2.3)** [4]

Q.4 Explain two types of filtering stream. **(Refer section 4.3)** [4]

Q.5 How Implicit filtering is differing from explicit filtering **(Refer section 4.3.2)** [4]

Q.6 In filtering stream, explain how Alon-Matias-Szegedy (AMS) algorithm works. **(Refer section 4.3.4)** [7]

Q.7 Explain decaying window with example **(Refer sections 4.3.6)** [4]

Q.8 Describe applications of RTAP. **(Refer section 4.4.1)** [7]

Q.9 Write full form of RTAP. Name some real time RTAP. **(Refer section 4.4)** [3]

Q.10 Explain sentiment analysis process in detail. **(Refer section 4.5.1.1)** [7]

Q.11 How big data helps stock market analysis. **(Refer section 4.5.2)** [7]

❑❑❑

# 5 Frameworks

***Syllabus***

*Applications on Big Data Using Pig and Hive, Data processing operators in Pig, Hive services, HiveQL, Querying Data in Hive, fundamentals of HBase and ZooKeeper, IBM InfoSphere BigInsights and Streams.*

***Contents***

## 5.1 Hive

### 5.1.1 Hive Architecture

GTU : Summer-17, 18, 19, Winter-18

- Fig. 5.1.1 shows the architecture of hive. It shows various clients that can connect to Hive Server and different services of Hive. It also has Metastore which acts as central repository of metadata.
- Hive architecture contains two main components, first is Hive Services and second Hive clients.

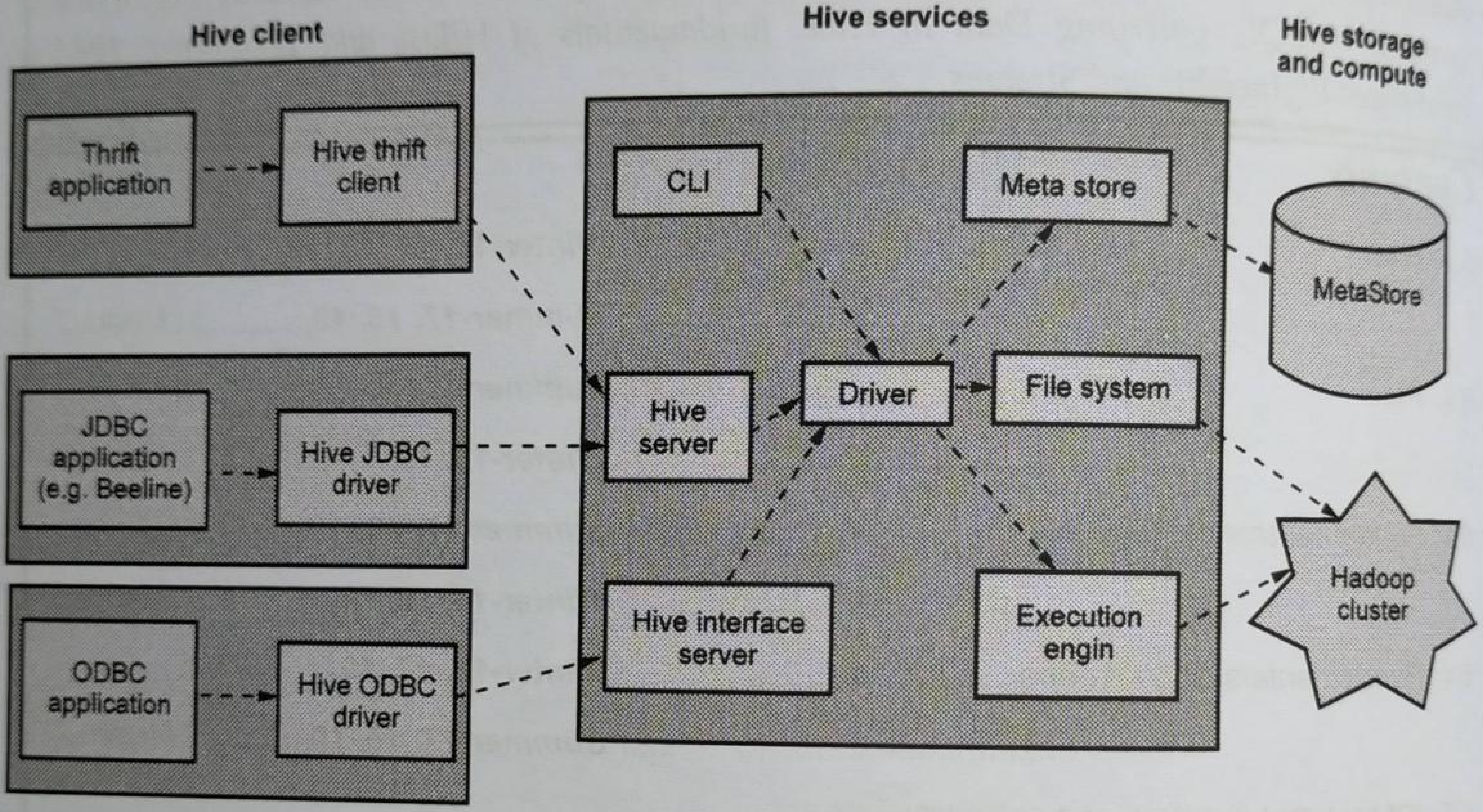


Fig. 5.1.1 Hive architecture

#### 5.1.1.1 Hive Services

- Hive command is used to start different services. To retrieve the list of service names available type '–service help'.
- Few of the useful services are as follows :

**1. cli**

- It is default service command-line interface to Hive shell.

**2. hiveserver2**

- Hive is run as a server and it also enables access from different clients which are written in various languages.
- Hiveserver2 improves the hive server by providing multiuser concurrency and authentication.

- In order to interact with Hive, applications using JDBC, Thrift and ODBC drivers need to run hiverserver2.

**3. beeline**

- It is a command line interface to Hive which works by connecting to hiverserver2 using JDBC or in embedded mode.

**4. hwi**

- It is a web interface to Hive. It can be used as a alternative to the command line interface without having to install any client software.

**5. jar**

- It is a Hive equivalent of hadoop jar, and is a appropriate approach to run applications written in Java which include Hadoop as well as Hive classes on the classpath.

**6. metastore**

- By default, the metastore runs on the same process on which Hive service runs. By using this service, it is possible to run metastore as standalone process.
- You can set the METASTORE_PORT environment variable to define the port number on which server will listen. The default port number is 9083.

### 5.1.1.2 Hive Clients

- Once you run Hive as server by using "hive –service hiveserver2" command, there are various approaches available in order to connect to it from different applications :

**1. Thrift Client**

- The Hive server is exposed as a Thrift service, so it is feasible to connect with it by using any programming language which supports Thrift.
- Third party projects are available which provide clients for Ruby and Python.

**2. JDBC driver**

- In Hive org.apache.hadoop.hive.jdbc.HiveDriver provides Type 4 JDBC driver i.e. pure java driver.
- By configuring JDBC URI as "jdbc:hive://host:port/dbname, a Java application can connect to Hiver server which is running on provided host and port in different process.

- The alternate way to connect to Hive server using JDBC is in embedded mode as "jdbc:hive2://" URI. In this mode, Hive server and java application runs on same JVM.
- The Beeline command line interface makes use of JDBC driver to connect with Hive server.

**3. ODBC driver**

- In order to connect with Hive, business intelligence software uses ODBC driver.
- An ODBC driver does not bring apache hive distribution with it, but various vendors are available which make it freely available.

#### 5.1.1.3 The Metastore

**GTU : Winter-16, 17, 19**

- Hive have central store of metadata called metastore. The metastore is divided into two parts : a service and backing store for data. There are three types of metastore arrangement possible: embedded metastore, local metastore and remote metastore.
- By default, metastore service and Hive service runs on the same JVM and have an embedded Derby database instance backed by local disk. This arrangement of metastore is called as embedded metastore.
- To start with Hive, simple approach is to use embedded metastore; but database files can access only one embedded Derby database at any one time. This means only one Hive session will be open by the time metastore is being accessed.
- Using of standalone database is a solution for providing multiple sessions. This configuration is known as local metastore, since the metastore service and Hive service runs on same JVM, but database runs on separate process, on the same machine or on remote machine.
- Any JDBC supported database can be used by setting javax.jdo.option.* configuration properties.
- For local metastore MySQL is popular choice.
- jdbc:mysql://host/dbname?createDatabaseIfNotExist=true is the value for javax.jdo.option.ConnectionURL property and com.mysql.jdbc.Driver is the value for javax.jdo.option.ConnectionDriverName.
- You must keep the JDBC driver jar file on Hive's classpath, which can be done by storing the jar file in Hive's lib directory.
- The third metastore is known as remote metastore, in which more than one metastore runs on different processes to the Hive service.

- To use remote metastore Hive service is configured by assigning hive.metastore.uris to metastore server URI. Metastore server URIs are of type thrift://host:port, where the port number is set while starting metastore server by METASTORE_PORT.

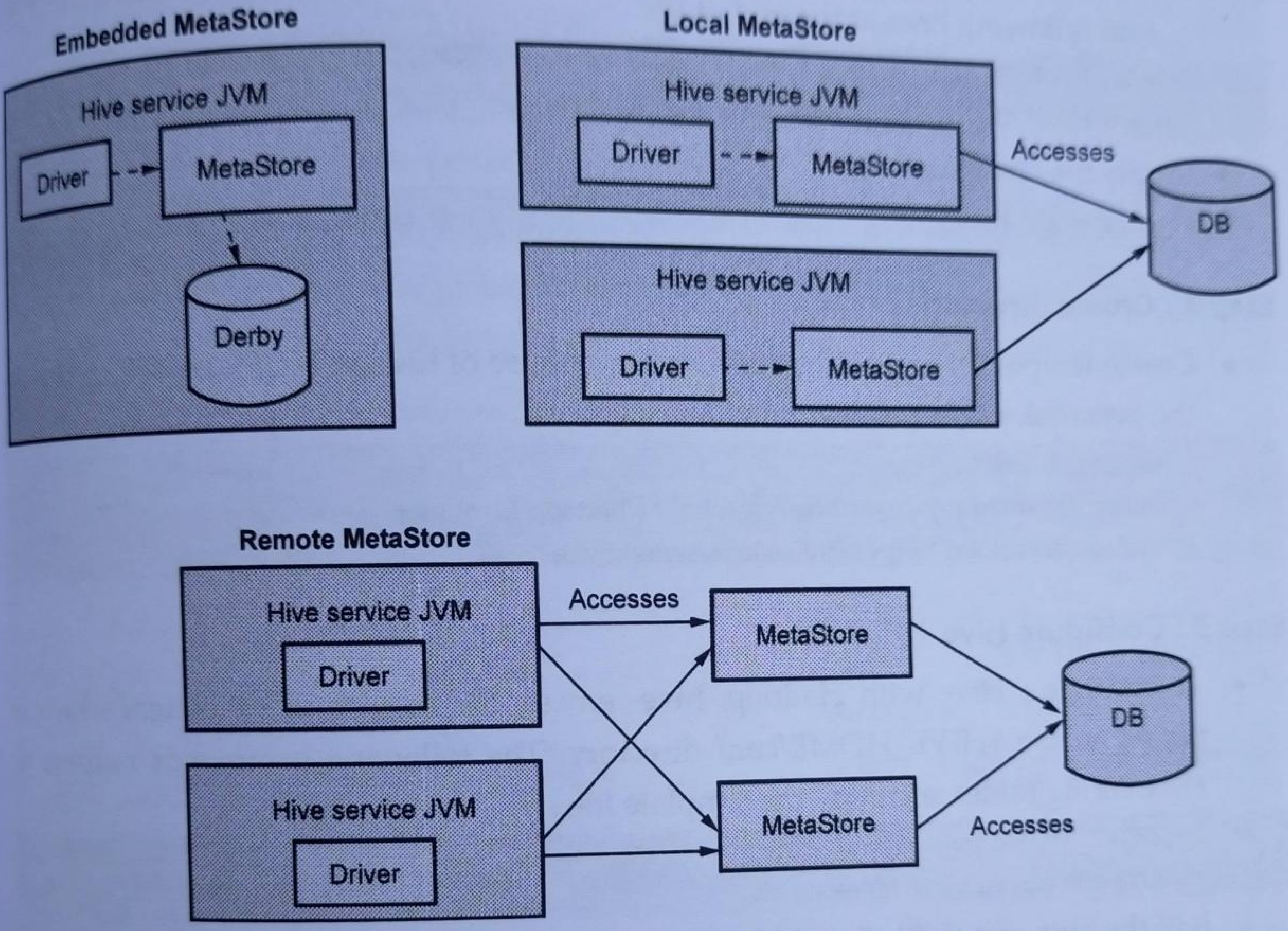


Fig. 5.1.2 Hive metastore

## 5.1.2 Hive Installation

**Step 1 : Download and copy Hive**

- Download hive on below path (nearly 93 MB) :

```
http://www.apache.org/dyn/closer.cgi/hive/
```

- Extract the .tar.gz file in Downloads/ and rename it to hive/ and move the folder to /usr/lib/ path using following command :

```
sudo mv Downloads/hive /usr/lib
```

**Step 2 : Change owner**

- Provide access to hive path by changing the owners and groups to hduser and hadoop respectively.

```
sudo chown -R hduser:hadoop /usr/lib/hive
```

**Step 3 : Configure environment variable**

- Configure environment variables in .bashrc file.

```
su - hduser
vim ~/.bashrc
```

Add following lines at the end of file

```
export HIVE_HOME=/usr/lib/hive/ export PATH=$PATH:$HIVE_HOME/bin
export HADOOP_USER_CLASSPATH_FIRST=true
```

- Apply the changes :

```
source ~/.bashrc
```

**Step 4 : Create directories**

- Create temporary and folder for data warehouse of hive in HDFS as well as change the permissions.

```
hadoop fs -mkdir /tmp
hadoop fs -mkdir -p /user/hive/warehouse hadoop fs -chmod g+w   /tmp
hadoop fs -chmod -R g+w/user/hive/warehouse
```

**Step 5 : Configure Hive**

- To configure Hive with Hadoop, hive- env.sh file require to be edited, which is placed in the $HIVE_HOME/conf directory. The following commands redirect to Hive config folder and copy the template file :

```
cd $HIVE_HOME/conf
cp hive-env.sh.template hive-env.sh
```

- Edit the hive-env.sh file by adding the following line :

```
export HADOOP_HOME=/usr/local/Hadoop
```

**Step 6 : Run Hive**

- Before starting Hive, make sure that all Hadoop services are running. After that type hive.

### 5.1.3 HiveQL

**GTU : Winter-16, 17, 18, 19, Summer-17, 18, 19**

- Hive's SQL language is known as HiveQL, it is combination of SQL-92, Oracle's SQL language and MySQL.
- HiveQl provides some improved features from previous version of SQL standards, like analytics function from SQL 2003.
- Some Hive's extension like multitable inserts, TRANSFORM, MAP and REDUCE clauses, are inspired by MapReduce.

- Table 5.1.1 shows comparison between SQL and HiveQL with respect to some feature.

| Feature | SQL | HiveQL |
|---|---|---|
| Updates | UPDATE, INSERT, DELETE | UPDATE, INSERT, DELETE |
| Transaction | Supported | Limited Support |
| Indexes | Supported | Supported |
| Data Types | Integral, floating-point, fixedpoint, text and binary strings, temporal | Boolean, integral, floatingpoint, fixed-point, text and binary strings, temporal, array, map, struct |
| Functions | Hundreds of built-in functions | Hundreds of built-in functions |
| Multitable inserts | Not supported | Supported |
| Create table.....as Select | Not supported | Supported |
| Select | Supported | Supported with SORT BY clause for partial ordering and LIMIT to restrict number of rows returned. |
| Joins | Supported | Inner joins, outer joins, semi joins, map joins, cross joins |
| Subqueries | Used in any clause | Used in FROM, WHERE, or HAVING clauses |
| Views | Updatable | Read-only |

Table 5.1.1 A high-level comparison of SQL and HiveQL

### 5.1.3.1 Tables

**GTU : Winter-16, Summer-19**

- A Hive table is logically comprised of the information being stored and the related metadata describing the layout of the information in the table. The data normally resides in HDFS, in spite of the fact that it might reside in any Hadoop filesystem, including the local filesystem or S3.
- Relational database is used to store Hive metadata.

## Managed Tables and External tables

Two types of table can be created in Hive: managed table and external table.

- When the table is created in Hive, by default Hive will manage the data that means Hive moves the data into Hive's warehouse directory. Such a table is known as managed table.
- You can also create an external table which notifies Hive that the data is stored at a location which is outside warehouse directory.
- The difference in these two types of table can be observed in LOAD and DROP query.
- First let's take example of managed table.
- When the data is loaded into managed table, it is moved to the Hive's warehouse directory. For example :

```
CREATE TABLE managed_tbl (dummy STRING);
LOAD DATA INPATH '/user/jerry/test.txt' INTO table managed_tbl;
```

- Above query moves the file hdfs://user/jerry/test.txt into Hive's warehouse directory for managed_tbl table, that is hdfs://user/hive/warehouse/managed_tbl
- In hive the directory name in Hive's warehouse is same as managed table name.
- To drop the table following query is used :

```
DROP TABLE managed_tbl;
```

- After the above query get executed, the table along with it's data and metadata is deleted.
- The LOAD operation performs move operation and the DROP performs delete operation after which data is no longer exists. This is the meaning of Hive to manage the data.
- An external table acts differently. In this creation and deletion of data is controlled. The external data location is specified at the time of table creation, as follows :

```
CREATE EXTERNAL TABLE external_tbl (dummy STRING)
LOCATION '/user/jerry/external_tbl';
LOAD DATA INPATH '/user/jerry/test.txt' INTO TABLE external_tbl;
```

- The keyword EXTERNAL notifies Hive that Hive is not managing the data; hence the data is not moved to the warehouse directory. Also it does not verify if the external location specified is exists or not at the time it is defined.
- This feature specifies that the data can be created after the table is created.
- When the external table is dropped, Hive only deletes the metadata and leaves the data untouched.

## Importing Data

- Data can be imported using Load Data, Inserts and Create Table ..... As Select.

1. LOAD DATA
   - It imports the data by moving or copying files to the table's directory.
   - LOAD DATA is used as follows to import data :

```
CREATE TABLE test (dummy STRING);
LOAD DATA INPATH '/user/jerry/demo.txt' INTO table test;
```

2. Inserts
   - Insert statements are used as follows to import data into table :

```
INSERT OVERWRITE TABLE target_tbl
SELECT col1, col2
FROM source_tbl;
```

   - In the above query OVERWRITE keyword specifies that the content of targettable is replaced by the results of the SELECT statement.
   - In order to insert records into already created table Insert into table...... query can be used.
   - In Hive we can do Multitable insert. For multitable insert, the statement will start as follows :

```
FROM source_table
INSERT OVERWRITE TABLE target_table
SELECT col1, col2;
```

- Above query shows that multitable insert statement can be added in above query. Because of this reason it is called Multitable Insert.
- Multitable Insert statement is more efficient than multiple INSERT statements because the source table needs to be scanned only once to produce the multiple disjoint outputs.
- Following is an example of Multitable Insert :

```
FROM records2_tbl
INSERT OVERWRITE TABLE stations_by_year
SELECT year, COUNT(DISTINCT station)
GROUP BY year
INSERT OVERWRITE TABLE records_by_year
SELECT year, COUNT(1)
GROUP BY year
INSERT OVERWRITE TABLE good_records_by_year
SELECT year, COUNT(1)
WHERE temperature != 9999 AND quality IN (0, 1, 4, 5, 9) GROUP BY year;
```

- The above shows that records2_tbl is the only source table, but three different tables are there to store the results of different select statement.

3. CREATE TABLE.... AS SELECT

- Any time it is suitable to store the output of Hive query into new table, so as to perform further processing on the output.
- The column definition of new table is derived by the SELECT clause.
- In the following query, the target_table have two columns named col1 and col2 whose data type is same as source_table :

```
CREATE TABLE target_table
AS
SELECT col1, col2
FROM source_table;
```

- A CTAS operation is atomic, hence in any case if SELECT query fails, the table is not created.

**Altering table**

- In Hive table definition can be changed after the table is created, since Hive supports schema on read approach.
- In Hive table can be renamed by using ALTER TABLE statement:

ALTER TABLE source_name RENAME TO target_name;

- Along with updating table metadata, ALTER TABLE statement also moves the table directory so that it shows the new name. For above query the directory /user/hive/warehouse/source_name renamed to /user/hive/warehouse/target_name.
- Hive permits to add new columns, change the definition of column or can replace the existing columns with new set of columns.
- A new column can be added to an existing table as follows :

```
ALTER TABLE target_name ADD COLUMNS (col3 STRING);
```

- After the existing columns col3 is aadded. Since the data files are nopt updated, so the queries for col3 will return null value, since Hive does not allow to update existing record.

**Dropping Table**

- To delete the data and metadata of table the DROP TABLE statement is used. If the table is external, then only metadata is deleted and the data is left as it is.
- The DROP TABLE query can be used as follows :

```
DROP TABLE test_table;
```

- In can you want to keep the table definition and only delete the data TRUNCATE statement is used. Query will be as follow :

```
TRUNCATE TABLE test_table;
```

## 5.1.4 Querying Data

GTU : Winter-17, Summer-19

### 5.1.4.1 Sorting and Aggregating

- Data in Hive can be sorted by using ORDER BY clause. It performs sorting of input in parallel. In case globally sorted result is not necessary, in such cases SORT BY clause can be used. Sorted file for each reducer is generated using SORT BYclause.
- When you want to control which row should go to which reducer, in such case DITRIBUTE BY clause of Hive is used.
- Following example sorts the weather dataset by temperature and year, in such a way that all rows for specified year go to same reducer :

```
hive> FROM records2_tbl
> SELECT year, temp
> DISTRIBUTE BY year
> SORT BY year ASC, temp DESC;
1949 111
1949 78
1950 22
1950 0
1950 -11
```

### 5.1.4.2 MapReduce Scripts

- To call an external program or script from hive, TRANSFORM, MAP and REDUCE clause is used by using Hadoop streaming approach.
- Following python script filters the poor quality readings for temperature:

```
import re
import sys
for line in sys.stdin:
(y, t, q) = line.strip().split()
if (t != "9999" and re.match("[01459]", q)):
print "%s\t%s" % (y, t)
```

- Above script can be used in Hive as follows :

```
hive> ADD FILE /user/jerry/hive-sample/ src/main/python/good_quality.py;
hive> FROM records2
> SELECT TRANSFORM(year, temp, quality)
> USING 'good_quality.py'
> AS year, temp;
1950 0
1950 22
1950 -11
1949 111
```

- The above query streams the year, temp and quality fields as tab separated lines to good_quality.py script, and tab separated output is parsed into year and temp field to form the result of the query.
- The above example doesn't have any reducer. To specify map and reduce function, nested form of query can be used.
- Following example shows the python script for map function to filter poor quality reading for temperature and python script for reduce function to find max temp. The MAP and REDUCE keyword is used in Hive query.
- Map function for filtering poor quality reading for temperature in Python :

```
import re
import sys
for line in sys.stdin:
(y, t, q) = line.strip().split()
if (t != "9999" and re.match("[01459]", q)):
print "%s\t%s" % (y, t)
```

- Reduce function for maximum temperature in Python :

```
import sys
(last_key, max_val) = (None, -sys.maxint)
for line in sys.stdin:
(key, val) = line.strip().split("\t")
if last_key and last_key != key:
print "%s\t%s" % (last_key, max_val)
(last_key, max_val) = (key, int(val))
else:
(last_key, max_val) = (key, max(max_val, int(val)))
```

- Above scripts can be used in Hive query as follows :

```
FROM (
FROM records2_tbl
MAP year, temp, quality
USING 'good_quality.py'
AS year, temp) map_output
REDUCE year, temp
USING 'max_temperature_reduce.py'
AS year, temp;
```

### 5.1.4.3 Joins

**GTU : Winter-17, 19**

- Hive supports four type of join which are, Inner Join, Outer Join, Semi Join and Map Join.

#### 1. Inner Join

1. Inner join is simple type of join, where the match in tables given as input results in rows in the output.
2. Consider two tables, first things that contain IDs and their names and second sales that contain IDs of items user brought and name of people.
3. Following query shows the content of both the table :

```
hive> SELECT * FROM sales;
Jay 3
Paresh 5
Avinash 0
Shri 4
Harshal 3
hive> SELECT * FROM things;
3 Tie
5 Coat
4 Hat
1 Scarf
```

4. Inner join is performed on the above tables as follows :

```
hive> SELECT sales.*, things.*
> FROM sales JOIN things ON (sales.id = things.id);
Jay 3 3 Tie
Paresh 5 5 Coat
Shri 4 4 Hat
Harshal 3 3 Tie
```

5. By using the predicate in the ON clause, the table in the FROM clause is joined with the table in JOIN clause.
6. In Hive you can only use equality in join predicate, which in the above example is matching on the id column on both the tables.

#### 2. Outer Join

1. Outer join permits to discover non-matches in the table being joined.
2. In case you use LEFT OUTER JOIN, the query will display a row for each row in left table, even though there does not exists the matching row in the table to which it is being joined.

3. In following example left table is sales and right table is things. LEFT OUTER JOIN can be performed as follows :

```
hive> SELECT sales.*, things.*
> FROM sales LEFT OUTER JOIN things ON (sales.id = things.id);
Jay 3 3 Tie
Paresh 5 5 Coat
Avinash 0 NULL NULL
Shri 4 4 Hat
Harshal 3 3 Tie
```

4. When we have performed inner join the row for Avinash was not returned, but in this example row for Avinash is returned, and column from the corresponding things table are NULL as there is no match.
5. RIGHT OUTER JOIN is also supported by Hive. In this join as compared to left outer join roles of table are reversed.
6. In this join, all the items from things table is included, even if they don't have matching row in left table.
7. In our example scarf is also included in output even though it is not purchased by anyone.
8. The query will be like :

```
hive> SELECT sales.*, things.*
> FROM sales RIGHT OUTER JOIN things ON (sales.id = things.id);
Jay 3 3 Tie
Harshal 3 3 Tie
Paresh 5 5 Coat
Shri 4 4 Hat
NULL NULL 1 Scarf
```

9. Hive supports a full outer join, in which the result have rows from both the table even though they don't datisfy the condition.
10. The query will look like

```
hive> SELECT sales.*, things.*
> FROM sales FULL OUTER JOIN things ON (sales.id = things.id);
Avinash 0 NULL NULL
NULL NULL 1 Scarf
Harshal 3 3 Tie
Jay 3 3 Tie
Shri 4 4 Hat
Paresh 5 5 Coat
```

**3. Semi Join**

1. To retrieve all items from things table which are also in sales table, SEMI JOIN can be used
2. The query will be like :

```
hive> SELECT *
> FROM things LEFT SEMI JOIN sales ON (sales.id = things.id);
2 Tie
4 Coat
3 Hat
```

3. Hive limits on using only LEFT SEMI JOIN query.

**4. Map Join**

1. Consider the following inner join query :

```
SELECT sales.*, things.*
FROM sales JOIN things ON (sales.id = things.id);
```

2. In case one table small in size to fit in memory, as things in our example, Hive can load that table into its memory to perform join using mapper. This is known as map join.

### 5.1.4.4 Sub Queries

- If a SELECT statement is embedded inside another SQL statement, then it is known as subquery.
- Hive have limited support for subquery, it allows to use subquery in either FROM clause of SELECT statement or in WHERE clause in some cases.
- Following query retrieves the mean maximum temperature for each year and weather station :

```
SELECT station, year, AVG(max_temp)
FROM (
SELECT station, year, MAX(temp) AS max_temp
FROM records2
WHERE temp != 9999 AND quality IN (0, 1, 4, 5, 9)
GROUP BY station, year
) mt
GROUP BY station, year;
```

- In above query FROM subquery is used to find max temp for every station and date combination, and the outer query uses AVG function to find mean maximum temperature for every year and station.
- The outer query uses output of subquery, due to this the subquery must be assign an alias(mt in above query). The columns of subquery must have unique name so that outer query can refer it.

GTU : Summer-17, Winter-18, 19

## 5.2 Pig

- In Pig, the data structure is multivalued, nested and richer, and the transformations which can be applied to data using pig are much more powerful.
- Pig is composed of two pieces :
    - The language used to express data flows, called Pig Latin.
    - The execution environment to run Pig Latin programs. There are currently two environments : local execution in a single JVM and distributed execution on a Hadoop cluster.
- A Pig Latin program is composed of a sequence of transformations, or operations, which are applied on input data to generate output. Pig execution environment translates the operations into executable form and then executes.
- Pig translates the transformations into a sequence of MapReduce jobs, but a programmer is not aware about it, which allows programmer to concentrate on data instead of nature of execution.
- For exploring large datasets Pig scripting is used. One limitation of MapReduce is that the development cycle is very long. Writing the reducer and mapper, compiling and packaging the code, submitting the job and retrieving the output is time consuming task.
- In response to lines of Pig Latin issued from console, Pig processes large volume of data.
- Pig was developed by Yahoo! for making mining huge dataset easy.
- Sample run is performed on a subset of input data, so as to check for the errors in processing before it run on complete dataset.

### 5.2.1 Execution Types

- Pig has two execution types or modes: local mode and MapReduce mode.

**Local mode**

- In this, single JVM is used to run Pig and accesses the local filesystem. When the dataset is small in size this mode is used.
- The –x option is used for setting the execution type. To use local mode, the option is set to local as follow :

```
% pig -x local
grunt>
```

- The above command starts the interactive shell of pig named Grunt.

**MapReduce mode**

- In this mode, pig translates queries to MapReduce jobs and executes them on Hadoop cluster.
- When the dataset is of large size, MapReduce mode is used.
- In order to use this mode, it is required that the version of Hadoop and version pig should match. Since Pig releases will only run with the particular Hadoop version.
- HADOOP_HOME environment variable is used by pig to find which Hadoop client to execute. In case the environment variable is not set, Pig uses a bundled copy of Hadoop libraries.
- To start using Pig in MapReduce mode, the –x option is set to MapReduce.
- In following command –brief option is used so as to block the timestamp from being logge :

```
% pig -brief
```

- Connecting to hadoop file system at: hdfs://localhost/grunt>

## 5.2.2 Running Pig Programs

- There are three ways of executing Pig programs, all of which work in both local and MapReduce mode :

1. Script
   - Pig can execute script file which contain pig commands. For instance, "pig script.pig" executes the command written in the file script.pig.
   - The - e option can be used to execute small scripts which are specified as string on command line.
2. Grunt
   - Pig commands can be executed on the interactive shell known as Grunt.
   - It is used when file is not specified for pig to execute and the - e option is also not used. Using run and exec pig script can be executed from within Grunt.
3. Embedded
   - PigServer class can be used to run pig scripts from Java. PigRunner is used for programmatic access to Grunt.

## 5.2.3 Pig Latin

- This section provides description about the syntax and semantics of the Pig Latin programming language.

**1. Structure**

- Pig Latin program is composed of collection of statements. A statement can be a command or operation.
- For example, GROUP operation is also a type of statement :

```
grouped_records = GROUP records BY year;
```

- A statement can be split over multiple lines if it has to terminate by a semicolon.

```
records = LOAD 'input/ncdc/micro-tab/sample.txt'
AS (year:chararray, temperature:int, quality:int);
```

- Pig Latin have two types of comment. Single line comment is denoted by double hyphens and multiline comment is denoted by "/* */". Following example shows the type of comment :

```
/*
Information of program
*/
-- My program
A = LOAD 'input/example/data/A';
B = LOAD 'input/ example/data /B';
C = JOIN A BY $0, /* ignored */ B BY $1;
DUMP C;
```

- Pig Latin contains some keyword which have special meaning in language and it cannot be used as identifier. The keywords are like, LOAD and ILLUSTRATE operator, cat and ls command, matches and FLATTEN expressions and DIFF and MAX functions.

**2. Statements**

- Every statement in the program is parsed as Pig latin program is executed.
- The interpreter halts and display error message in case there are syntax or semantic errors.
- The interpreter makes a logical plan for each relational operation that forms core of Pig Latin program.
- The logical plan for the statement in inserted into the logical plan for the program so far, and after that interpreter moves to the next statement in program.
- When the logical plan is being made, no data processing is done. For example, consider following Pig Latin program :

```
-- max_temp.pig: Finds the maximum temperature by year
records = LOAD 'input/test/sample.txt'
```

```
AS (year:chararray, temp:int, quality:int);
filtered_records = FILTER records BY temp != 9999 AND
quality IN (0, 1, 4, 5, 9);
grouped_records = GROUP filtered_records BY year;
max_temp = FOREACH grouped_records GENERATE group,
MAX(filtered_records.temp);
DUMP max_temp;
```

- When the interpreter encounters a statement containing LOAD operator, it confirms that the statement is semantically and syntactically correct and then adds it to the logical plan, but the data from the file is not loaded.
- Pig interpreter validates the GROUP and FOREACH... GENERATE statements, and adds them to the logical plan, but do not execute them.
- The DUMP statement acts as a trigger for starting execution. At this point, logical plan is compiled into physical plan and executed.
- The physical plan which is prepared by Pig is the collection of MapReduce jobs.
- Table 5.2.1 shows the relational operators which can be included in logical plan.

| Category | Operator | Description |
|---|---|---|
| Loading and storing | LOAD | Data is loaded from filesystem to a relation. |
| | STORE | Stores a relation to filesystem or in any other storage. |
| | DUMP (\d) | Prints a relation to the console. |
| Filtering | FILTER | Filter out unnecessary rows from a relation. |
| | DISTINCT | Removes duplicate rows from a relation. |
| | FOREACH...GENERATE | Adds or removes fields to or from a relation. |
| | MAPREDUCE | Runs a MapReduce job using a relation as input. |
| | STREAM | Relation is transformed using an external program. |
| | SAMPLE | Selects a random sample of a |

| | | |
|---|---|---|
| | | relation. |
| | ASSERT | Ensures a condition is true for all rows in a relation; otherwise, fails. |
| Grouping and joining | JOIN | Joins two or more relations. |
| | GROUP | Groups the data in a single relation. |
| | CUBE | Creates aggregations for all combinations of specified columns in a relation. |
| Sorting | ORDER | Sorts a relation by one or more fields. |
| | RANK | Assign a rank to each tuple in a relation, optionally sorting by fields first. |
| | LIMIT | Limits the size of a relation to a maximum number of tuples. |

**Table 5.2.1 Pig Latin relational operators**

- Pig Latin provides the statement which is used to enable macros and user defined function into Pig script. The statements are as follows:
  - REGISTER : During runtime it registers a JAR file with the Pig.
  - DEFINE : Creates an alias for a user defined function, macro, streaming script, or command specification.
  - IMPORT : Imports macros which are defined in a separate file into a script.
- Since the above statement does not contain any relation, therefore these commands are not added into logical plan.

**3. Schemas**

- A relation in Pig might have a related schema, that gives names and types to the fields in relation.

- In following example, year is declared as integer instead of chararray. An integer type is suitable in case the year is needed to manipulate arithmetically, while chararray may be suitable if it is going to be used as identifier.
- Pig is built for analyzing plain input files which do not have type information, so the type of fields can be chosen later.

```
grunt> records = LOAD 'input/test/sample.txt'
>> AS (year:int, temp:int, quality:int);
grunt> DESCRIBE records;
records: {year: int,temp: int,quality: int}
```

**4. Functions**

- Functions in Pig come in four types :
  - o Eval function
    - It is a function which takes one or more expression as input and gives another expression as output. MAX is an example of in-built function, which gives maximum value as output.
    - Some eval functions are aggregate functions that mean they perform operation on a set of data to produce a scalar value; example of an aggregate function is MAX.
  - o Filter function
    - It is a special type of function which gives Boolean result.
    - To remove unnecessary rows, filter functions are used in FILTER operator. They can also be used in relational operators which takes Boolean conditions.
    - IsEmpty is an example of built-in filter function, which tests whether a bag have any items.
  - o Load function
    - A function that specifies how to load data into a relation from external storage.
  - o Store function
    - A function that specifies how to save the contents of a relation to external storage.
    - Frequently, load and store functions are implemented by the same type.
    - For example, PigStorage, which is used for loading data from delimited text files, can store data in the same format.

## 5.3 Fundamentals of HBase

**GTU : Summer-17, 18, Winter-17, 18, 19**

- HBase is column-oriented database. It is an open source project.
- HBase data model is same as of Google's big table which is built to provide faster access to large volume of structured data.
- Fault tolerance provided by HDFS is been influenced by HBase.
- It is one of the components of Hadoop ecosystem which give real time random read-write access to the data stored in HDFS.
- User can read the data stored in HDFS by using HBase.
- Following are few concepts which are used in HBase :

**1. Table :**

- Tables are used to store data in HBase. Table names should be string and made up of characters which can be used in file system path.

**2. Row :**

- Data is stored according to its row inside a table.
- Rows are identified uniquely by their row key. Row keys are used to sort table rows. The sorting is ordered by byte.
- The row keys are of type byte array.

**3. Column Family :**

- Rows and columns in HBase table are grouped into column families. A common prefix is given to all the column family members. E.g. info:name and info:gender are the members of same column family i.e. info.
- Arrangement of data storage is affected by column families. Because of this, the column families should be defined up front and are not easily updatable.
- Column family name must be of string type and must be made of characters which can be easily used in file system path.

**4. Column Qualifier :**

- Column qualifier is used to address data within column family.
- It does not require to be described in advance. It does not even have to be consistent between different rows.
- Column qualifier has byte array type.

**5. Cell :**

- A cell recognized as a intersection between column family, column qualifier and row key.
- The data stored in a cell is referred to as that cell's value.
- Cell's value have type byte array.

**6. Timestamp :**

- Call's value is versioned. These versions are recognized by its version number. By default, HBase assigns timestamp as the value for version number. This timestamp is the time when the data in cell is written.
- In case while writing timestamp is not defined, then the current timestamp is used.
- Recent timestamp can be returned if in case timestamp is not provided for read.
- HBase configures retained number of cell value version, for every column family. The default number of cell versions is three.

**7. Region**

- HBase horizontally partitions tables into regions. Each region contains a subset of rows in the table.
- A region is defined by the table to which it belongs, table's first row and last row.
- Initially every table has a single region, but as the number of rows in region grows beyond the configurable size, the region is divided into two regions of around same size.
- Regions are the unit which is distributed over the cluster of HBase. In case a table is very large for single server can be carried by the cluster of servers, where each node stores a subset of total region of table.
- **Example :** HBase data model for storing personal information.

| Row Key | Column Family : Name | | Column Family : Info | |
|---|---|---|---|---|
| | Name:Fname | Name:Lname | Info:phone_no | Info:address |
| C0001 | Paresh | Chauhan | 9978547176 | Bhavnagar |
| C0002 | Shripad | Deshpande | 9595595097 | Nasik |
| C0003 | Avinash | Jha | 9730917725 | Vapi |

**Fig. 5.3.1 HBase data model for storing personal information**

## 5.3.1 HBase Schema Design

- In this section we will discuss about the schema designing of two tables : customer and product_sales.
- *Customer*
  1. Customer data is stored in this table.
  2. In this table row key will be customer_id. This table has two column families named : CName and Info. These column families acts as key-value dictionary for customer name and contact information. Suppose these dictionary keys have columns CName:Fname, CName:Lname, Info:phone_no & Info:address.
- *Product_sales*
  1 This table stores product sales information. Suppose for this table row key is composite key of customer_id and a reverse order timestamp.
  2 Suppose the column family for this table is product. This column family have columns: product:Pname, product:Price & product:qnt_sold.
- Our selection approach came from knowledge about most efficient way we can read data from HBase. Rows and columns are stored in increasing lexicographical order.
- The secondary indexing and regular expression matching facility is available, but they come at the cost of performance.
- It is crucial that you should understand the efficient approaches to query the data, so as to select the effective setup for storing and accessing data.
- For customer table, the customer_id is selected as key, since the information about the customer is going to be accessed by its id. The product_sales table, however, uses a composite key which adds sales timestamp at the end. This will collect all the products sold to particular customer together, and by making use of reverse order timestamp and storing it as binary, product sales for every customer will sorted with most recent sales first.
- In the HBase shell, tables can be defined as follows :

```
hbase(main):001:0> create 'customer', { NAME => 'CName'},
            {NAME => 'Info'}
0 row(s) in 0.8600 seconds
hbase(main):002:0> create 'product_sales', {NAME => 'product'}
0 row(s) in 0.1900 seconds
```

## 5.3.2 Advance Indexing

- Advance indexing is used in order to provide support secondary indexes. In many case ranges of rows are scanned ordered by secondary indexes.
- Even though HBase does not support secondary indexes, there are use cases which need them. The cells can be searched not only by just primary coordinates but also an alternative coordinates. Along with it you can scan the rows from table which are sorted by secondary index.
- Secondary indexes stores relationship between existing coordinates and secondary coordinates, which is similar to indexes in RBDMS.
- Following are some of the advance indexing techniques :

  1. *Client-managed*

     - This technique merge data table with lookup or mapping table. Lookup table is also gets updated as soon as data is written into data table.
     - Reading data needs either a direct searching in main table, or, if the key is a secondary index, a lookup of main row key and after that in second operation data is retrieved.
     - As the logic is managed in client code, you can map the key as per your choice.
     - In case table does not have cross-row atomicity, the consistency of main and dependent table cannot be guaranteed. This can be overcome by using MapReduce to scan the data in table and remove the outdated entries.
     - The missing transactional support can result in data being stored in main table without mapping in table of secondary index, as the operation is failed once the table is updated, but before the index table is written. This can be reduced by writing the table of secondary indexes first, and after end of operation to the data table.
     - In order to perform mapping primary and secondary indexes have drawback of having to implement all required writing to store and search the data.
     - External keys need to be recognized to access the correct table, for instance :

```
myrowkey-1
@myrowkey-2
```

- The first key represents direct data table scan, whereas the second key, having the prefix, is a mapping which need to be performed by using secondary indexes.
- The name of table can also be encode and added to the prefix.

2. ***Indexed-Transactional HBase***
   - The solution provided by indexed transactional HBase extends HBase by adding implementation of server and client side classes.
   - To guarantee that updates of secondary indexes are consistent transactions are added in HBase. By giving client-side IndexedTableDescriptor, it adds index support.
   - Client and server classes are replaced by the class which manages indexing support.
   - For example, IndexedTable replaces HTable on client side. To enable scaning of rows present in data table using ordering of secondary index, getIndexedScanner() method is used.
   - It stores relationship between primary and secondary index in different table.
   - The disadvantage of this technique is that it might not support the recent version of HBase. It also decreases the overall performance.

3. ***Indexed HBase***
   - Indexed HBase is another solution for adding secondary indexes to HBase. It maintains different table for every index but maintains it in memory.
   - When the first time region is opened, indexes are generated. The time and I/O resource required are based on the configured region size.
   - It indexes only on-disk data. The memstore data is directly used for searching index related information.
   - The advantage of this approach is that indexes are always in sync and no need of explicit transactional control.
   - Since the necessary details are in memory, this approach can search faster for finding matching rows as compared to table-based indexing. But it needs lot of extra heap for maintaining index.
   - It might get into a situation at point in time that it cannot store all the indexes which are required by user.

## 5.4 Fundamentals of Zookeeper

**GTU : Winter-16, 17, 19, Summer-17, 18, 19**

- Zookeeper is core component of distributed system. For designing distributed system, few coordination services need to be designed and developed, which are as follows :

1. **Name service**
   - It is a service which provide name to some data associated with that name. E.g. A telephone directory is a name service which provide name of user to their phone number.
   - Similarly, DNS is also a name service which provide domain name to IP address.
   - In order to keep track of the services which are running and to search it by using their name, zookeeper can be used.
   - A name service can be extended to group membership service by using which you can retrieve the data covered in that group that are related to the name which is being searched.
2. **Locking**
   - Zookeeper provides an easy approach i.e. implementing mutexes, for permitting the serialized access to the resources which are shared in distributed system.
3. **Synchronization**
   - The data access over the distributed system should be synchronizing, for that purpose Zookeeper provides an easy approach.
4. **Configuration management**
   - To store and manage configuration information of distributed system centrally, Zookeeper can be used. This shows that updated configuration information will be provided from zookeeper to the node which joins the system.
   - This permits you that state of distributed system can be changed by changing the centralized configuration by using zookeeper client.
5. **Leader election**
   - In case a system fails in distributed system, automatic failover approach should be implemented. Zookeeper provides supports for automatic failover by using leader election.

- Zookeeper is itself a distributed application along with the coordination service for distributed system.
- Zookeeper uses client-server model, the nodes who uses the service are clients, and nodes which provide these services are server.
- Zookeeper ensemble is formed by collection of zookeeper servers. At a time, one zookeeper server connected to one client.
- One Zookeeper server can manage any number of client connections at a time. Every client sends a ping after certain interval to notify to the server that it is alive and connected.
- The Zookeeper server responds with acknowledging the pig, specifying that the server is also alive.
- The client connects to the other server in ensemble if it does not receive the acknowledgement sent by server within specified time.
- In such case the client's session is transferred to the new server.
- Figure 5.4.1 shows the client-server architecture of Zookeeper.

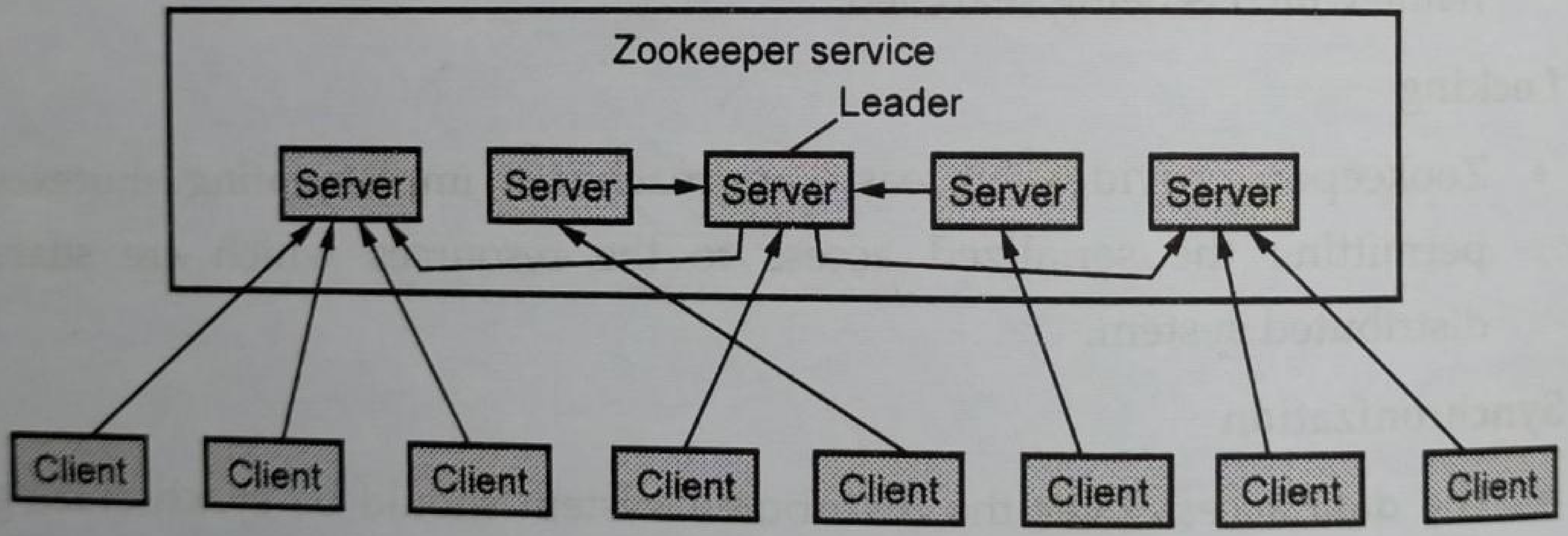


**Fig. 5.4.1 Client server architecture of Zookeeper**

### 5.4.1 Monitoring Cluster

- The Zookeeper services in cluster can be monitored using "the four letter words".
- Zookeeper responds to small set of commands where every command is made up of four letters. These commands are issued via telnet to zookeeper at client port.
- Following are few of the command which are used to monitor cluster by zookeeper :

1. stat : Provide general information about the server and connected clients.
2. srvr : Provide complete list of server.

3. cons : Provides complete list of connections details for the clients connected to this server. The details include session id, last operation performed, number of packets sent or received, etc.
4. conf : Display details about serving configuration
5. crst : Statistics for collection is reset.
6. dump : This command only works on leader node. It lists outstanding nodes and ephemeral nodes.
7. envi : Display detains of serving environment.
8. ruok : This command is issued by connected client to check if the server is running in non-error state.
9. imok : Server responds with this command to client if it is running.
10. srst : Reset server statistics.
11. wchs :Lists details on watches for the server.
12. wchc : Displays detail information about the watches for the server, by session. It display list of sessions with related watches.
13. wchp : Displays detail information about the watches for the server, by path. It display list of paths with related sessions.

- Example of ruok command :

```
$ echo ruok  nc 127.0.0.1 5111
imok
```

### 5.4.2 HBase Uses Zookeeper

GTU : Summer-17, Winter-17

- For real time read or write access of huge dataset distributed NoSQL database HBase is used, which runs on top of HDFS.
- HBase follows master-slave architecture where HBase acts as a server and region server act as slaves.
- Installation of HBase distributed application is based on a running zookeeper cluster.
- To track the status of data which is distributed across master and slaves, HBase uses Zookeeper. It is done by using the centralized configuration and distributed mutex techniques.

- Following are few use cases of HBase :
  - **Telecom :** Billions of call records are stored by telecom industry and in real time scenario accessing of these huge data is very difficult. HBase can be used to process such a large data in real time, efficiently and easily.
  - **Social network :** Social networking sites like, facebook, LinkedIn and Twitter receives a large amount of data in the form of posts created by users. HBase can be used to discover the interesting facts and recent trends.

### 5.4.3 Building Application with Zookeeper

- Zookeeper is a centralized service which is used to managing centralized configuration, naming, providing group services and distributed synchronization. These services are used by distributed applications.
- Whenever these services are implemented there is need of fixing errors and race conditions which are unavoidable. Due to difficulty in implementing these services, applications uses skimp on the services.
- Various implementations of such services can lead to complexity in maintenance of application when it is deployed.
- Andrei Zmievski provides the PHP extension of Zookeeper. The extension can be downloaded from PECL or GitHub.
- In order to develop an application with Zookeeper, first you need to install Zookeeper, which can be downloaded from official site. After that execute following commands :

```
$ tar zxfv zookeeper-3.4.5.tar.gz
$ cd zookeeper-3.4.5/src/c
$ ./configure --prefix=/usr/
$ make
$ sudo make install
```

- The above commands will install library and headers of Zookeeper.
- Now the PHP extension can be compiled.

```
$ cd
$ git clone https://github.com/andreiz/php-zookeeper.git
$ cd php-zookeeper
$ phpize
$ ./configure
$ make
$ sudo make install
```

- Edit the PHP configuration file and add "extension=zookeeper.so".

```
$ vim /etc/php5/cli/conf.d/20-zookeeper.ini
```

- Ensure that the extension in working properly by executing following command :

```
$ php -m grep zookeeper
zookeeper
```

- Now you need to create a directory for storing all data about the services. It can be done as follows :

```
$ mkdir /home/avinash/zoo
$ cd
$ cd zookeeper-3.4.5/
$ cp conf/zoo_sample.cfg conf/zoo.cfg
$ vim conf/zoo.cfg
```

- Find 'dataDir' and point it to the directory '/home/avinash/zoo' as created earlier
- Execute the following command to connect to services and to create '/demo/' znode.

```
$ bin/zkServer.sh start
$ bin/zkCli.sh -server 127.0.0.1:2181
[zk: 127.0.0.1:2181(CONNECTED) 14] create /demo 1
Created /demo
[zk: 127.0.0.1:2181(CONNECTED) 19] ls /
[demo, zookeeper]
```

- Zookeeper saves data in a tree structure. In zookeeper directories can act as directory as well as file simultaneously.
- Each entity which is stored by Zookeeper is known as znode.
- Keep the client connected as it is going to be used.
- Open new terminal and create ZooDemo.php file and add following lines :

```
$zoo = new ZookeeperDemo('127.0.0.1:2181');
$zoo->get( '/test', array($zoo, 'watcher' ) );
while( true ) {
echo '.';
sleep(2);
}
```

- Execute the PHP script by using following command :

```
$ php ZooDemo.php
```

- It will display a dot after an interval of 2 seconds. Now open the Zookeeper client window and update the value of "/demo" using following command :

```
[zk: 127.0.0.1:2181(CONNECTED) 20] set /test foo
```

- It will trigger 'Insider Watcher' message in PHP script.
- Zookeeper provides watchers that can be attached to znodes. The service notifies all interested clients if watched znode changes. Because of this PHP script gets to know about the change.

- Callback is the second parameter of Zookeeper::get method. Watcher gets consumed when event is triggered, so it is required to set it back in the callback.

### 5.4.3.1 Building Distributed Application

- The problem statement for the application is to let independent program to decide which one should be coordinating them and decide which should be executing the job. This process is known as leader election.
- Every process watches the process which is next to it. In case any process crashes or exits the watching process should verify if it is oldest process. If it is a oldest process then it becomes leader.
- Create a new PHP file and call it worker.php.

```
Zookeeper::PERM_ALL,
            'scheme' => 'world',
            'id' => 'anyone' ) );
private $isLeader = false;
private $znode;

public function __construct( $host = '', $watcher_cb = null, $recv_timeout = 10000 ) {
  parent::__construct( $host, $watcher_cb, $recv_timeout );
}
public function register() {
  if( ! $this->exists( self::CONTAINER ) ) {
    $this->create( self::CONTAINER, null, $this->acl );
  }
  $this->znode = $this->create( self::CONTAINER . '/w-',
                  null,
                  $this->acl,
                  Zookeeper::EPHEMERAL | Zookeeper::SEQUENCE );
  $this->znode = str_replace( self::CONTAINER .'/', '', $this->znode );
  printf( "Registered as: %sn", $this->znode );
  $watching = $this->watchPrevious();
  if( $watching == $this->znode ) {
    printf( "No one is here, I am the leader" );
    $this->setLeader( true );
  }
  else {
    printf( "I am watching %sn", $watching );
  }
}
public function watchPrevious() {
  $workers = $this->getChildren( self::CONTAINER );
```

```
sort( $workers );
$size = sizeof( $workers );
for( $i = 0 ; $i znode == $workers[ $i ] ) {
  if( $i > 0 ) {
   $this->get( self::CONTAINER . '/' . $workers[ $i - 1 ], array( $this, 'watchNode' ) );
   return $workers[ $i - 1 ];

  }
  return $workers[ $i ];
 }
}
throw new Exception(  sprintf( "There is a problem ! I cannot find myself: %s/%s",
              self::CONTAINER,
              $this->znode ) );

}
public function watchNode( $i, $type, $name ) {
 $watching = $this->watchPrevious();
 if( $watching == $this->znode ) {
  printf( "I am the new leader" );
  $this->setLeader( true );
 }
 else {
  printf( "Now I am watching %sn", $watching );
 }
}
public function isLeader() {
 return $this->isLeader;
}
public function setLeader($flag) {
 $this->isLeader = $flag;
}
public function run() {
 $this->register();
 while( true ) {
  if( $this->isLeader() ) {
   $this->doLeaderJob();
  }
  else {
   $this->doWorkerJob();
  }
   sleep( 2 );
  }
 }
public function doLeaderJob() {
 echo "I am Leading";
```

```
  }
  public function doWorkerJob() {
    echo "I am Working";
  }
}
$worker = new Worker( '127.0.0.1:2181' );
$worker->run();
```

- Open multiple terminals and execute the above script in every terminal, as follows :

```
# term1
$ php worker.php
Registred as: w-0000000001
No one is here, I am the leader
I am Leading
# term2
$ php worker.php
Registred as: w-0000000002
I am watching w-0000000001
I am Working
# term3
$ php worker.php
Registred as: w-0000000003
I am watching w-0000000002
I am Working
```

- Now close first terminal. After some Zookeeper will discover timeout and new leader is about to be elected.
- Zookeeper flag are used in the above script, as follow :

```
$this->znode = $this->create( self::CONTAINER . '/w-',
               null,
               $this->acl,
               Zookeeper::EPHEMERAL | Zookeeper::SEQUENCE );
```

- Each znode is as EPHEMERAL and SEQUENCE.
- EPHEMERAL means when client is disconnected that znode will be removed. Because of this PHP script gets to know about the timeout.
- SEQUENCE means that to every znode a sequence string is going to be appended. For workers we used it as unique identifier.

## 5.5 IBM InfoSphere BigInsights and Streams

### 5.5.1 IBM InfoSphere BigInsights

- SEQUENCE means that to every znode a sequence string is going to be appended. For workers we used it as unique identifier.

- Hadoop offers file system named Hadoop Distributed File System (HDFS) in which metadata is stored centrally which is known as NameNode and it is also a single point of failure without availability. It can take longer time to get Hadoop cluster in running state after the namenode is recovered, as metadata needs to be loaded in memory structure of NameNode, which is required for repopulating and rebuilding NameNode.
- Also installation, administration and configuration of Hadoop is complex and there are not enough manpower available with Hadoop skills. There are limited developers available with MapReduce skills. To deploy MapReduce to analytics one need to be skilled programmer of Java.
- IBM InfoSphere BigInsights solves all the issues mentioned above by focusing on two major goals :
  - i. Delivering Hadoop platform for enterprise use which is developed by considering high-availability, performance, scalability and ease of use.
  - ii. Provide business users with tool to analyze big data and also help developers to develop advance analytical tool by providing development and execution environment.

### 5.5.1.1 Why Solution like BigInsights is Required ?

- Assume if we will be able to :
  - i. Develop predictive model by combining available information and flow of big data information.
  - ii. Analyzing system logs from different system to reduce operational risk.
  - iii. Performing brand perception and consumer sentiment analysis on the data collected from different source.
  - iv. Leveraging existing systems and customer knowledge in new ways which were not explored because of higher cost.

### 5.5.1.2 Highlights of InfoSphere BigInsights

- i. It makes organization to analyze large volume and high variety of data cost-effectively in order to get better understanding of data.
- ii. BigInsights enables organizations with capability to meet different business need and also being compatible with Hadoop projects.
- iii. It also contains different IBM technologies which enhance value of Hadoop software to make faster time-to-value, which also includes acceleration of system, facilities for analytics, improving platform and integrating business software.

iv. While BigInsights provides a wide range of functions which extend beyond Hadoop capability, IBM has provided with approach by which IBM extensions can be used to Hadoop depending on the requirements.

v. Along with core functions for installation, configuration and management, BigInsights also contains advance analytics and UI for non-developer analyst.

vi. BigInsights can also be used for semi-structured or unstructured data. This solution does not need preprocessing of data or no schema definition is needed.

vii. The platform runs on low cost, commonly available hardware in parallel, also supporting linear scalability; as data grows more hardware can be added.

### 5.5.2 IBM InfoSphere Streams

- Analysis of data in motion is referred as Stream processing. It is required take action on specific type of data even before it gets stored. This kind of data is mostly generated by sensor or other instrument. For some use cases like healthcare, fraud detection and public safety, in real time insights on data should occur and the decision made to take action can be lifesaving.
- Streaming data may arrive linked to some master data identifier like mobile number. Banking transaction are matched with particular credit card, debit card or account number. But, not complete stream data is valuable, that is the reason the data should first analyzed with tool like IBM InfoSphere Streams before linking it to master data.
- Analysis of this large volume of data is done with micro-latency by IBM InfoSphere Streams. As the data flows in, IBM InfoSphere Streams analyzes data and recognize condition which can trigger alert. When such scenario occurs, the data is passed out of stream and linked to master data for further business use. This tool generates summary of the insights extracted out of the stream analysis done.
- InfoSphere Streams provides development and execution environment where application can be developed which can ingest, analyze and filter large amount of continuously flowing data which notify to take suitable action, within particular time frame.
- The data streams which are generated from cameras, sensor, news feed or any other sources are consumable by InfoSphere Streams. The data in streams can be of different types like audio, video, radar input etc. In InforSphere Streams analytic operators are defined to perform appropriate action on the streams of data.

## University Questions with Answers

**Winter - 2016 (IT)**

**Q.1** Explain working of Hive with proper steps and diagram. **[7]**

**Ans. :**

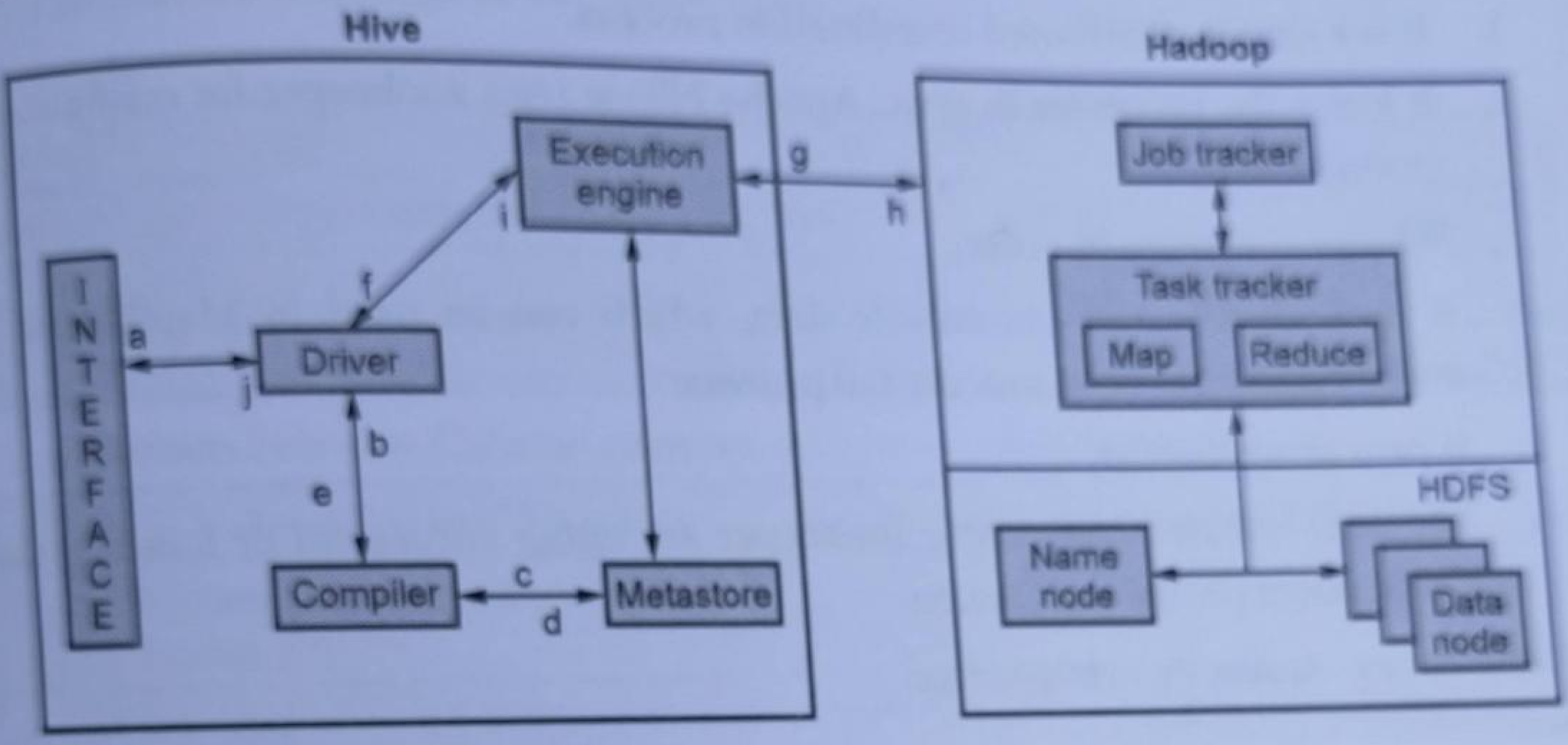


**Fig. 5.1 Working of Hive**

- Working of Hive is as follows :

1. The Hive interface such as Command Line or Web UI sends query to Driver to execute.
2. The driver consults with query compiler that parses the query to check the syntax and query plan or the requirement of query.
3. The compiler sends metadata request to Metastore.
4. Metastore sends metadata as a response to the compiler.
5. The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6. The driver sends the execute plan to the execution engine
7. Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
8. Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
9. The execution engine receives the results from Data nodes.

10. The execution engine sends those resultant values to the driver.
11. The driver sends the results to Hive Interfaces

**Q.2** What is Zookeeper ? List the benefits of it. **(Refer section 5.4)**

**Ans. : Benefits of Zookeeper :** [3]

1. It is a simple distributed coordination process.
2. It keeps the processes in sync. Apache HBase uses zookeeper for configuration management.
3. It keeps messages in order.
4. It uses specific rules to encode data, which can be used in MapReduce for coordinating queue to execute the process.
5. It provides reliability.
6. The data transfer done using zookeeper are either successful or failed, it does not perform partial transaction.

**Q.3** Differentiate : Apache Pig Vs MapReduce.

**Ans. :** [4]

| MapReduce | Apache Pig |
|---|---|
| It is a compiled programming language | It is a scripting language |
| Abstraction is at lower level. | Abstraction is at higher level |
| Lines of code is more | It have less line of code as compared to MapReduce |
| More development efforts are required for MapReduce. | Less effort is needed for Pig. |
| As compared to Pig efficiency of code is higher. | Code efficiency is less as compared to MapReduce. |

**Q.4** What do you mean by HiveQL Data Definition Language? Explain any three HiveQL DDL command with its syntax and example. **(Refer section 5.1.3)** [7]

**Q.5** Explain metastore in Hive **(Refer section 5.1.1.3)** [3]

**Q.6** Explain storage mechanism in HBase. Compare row oriented and column oriented database structures. [4]

**Ans. : Storage mechanism in HBase :**

- HBase is a column-oriented database and row is used to sort table. HBase schema defines column families.
- A table has different column families and every column family can have number of columns.
- Each cell value is given a version number which is by default timestamp.

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

| Rowid | Column Family | | Column Family | |
|---|---|---|---|---|
| | Col1 | Col2 | Col1 | Col2 |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

- Comparison between Column oriented and row oriented database :

| Sr. No. | Row-Oriented Database | Column-Oriented Database |
|---|---|---|
| 1. | It is useful in Online Transaction Process (OLTP) | It is useful in Online Analytical Processing (OLAP). |
| 2. | It is used when the data size is small | It is used when the data size is large |

**Summer - 2017 (IT)**

**Q.7** What is HBase? Discuss in detail the data model and implementation aspects. **(Refer section 5.3)** [7]

**Q.8** Elaborate on HiveQL data manipulation queries in detail **(Refer section 5.1.3)** [7]

**Q.9** Discuss the concept of regions in HBase and Storing Big Data with HBase. **(Refer section 5.3)** [7]

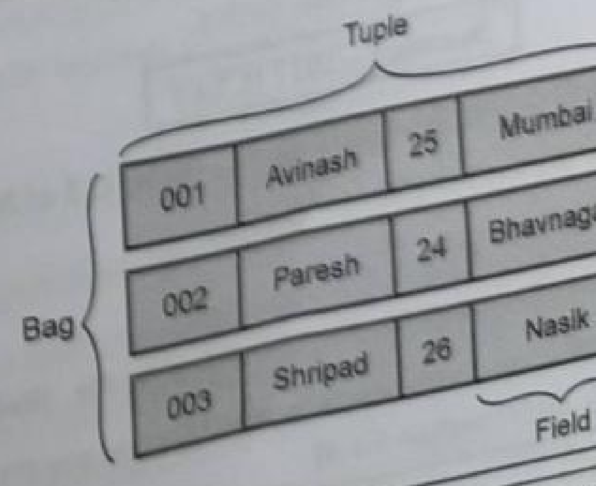
**Q.10** Explain how HBase uses zookeeper to build application with zookeeper. **(Refer section 5.4.2)** [7]

**Q.11** Draw and discuss the architecture of Hive in detail. **(Refer section 5.1.1)** [7]

**Q.12** Explain Pig data model in detail and discuss how it will be help for effective data flow. [7]

**Ans. : Pig Data Model**

Pig's Data Model is fully nested with different data types. This defines the structure of data which Pig is processing. Following figure shows pictorial representation of Pig's Data Model.



**1. Atom**

Atom is any single value in Pig Latin. It is stored as string and can be used as string as well as number. int, long, double, float, chararray and bytearray are atomic values of Pig.

Example : 'avinash' or '123'

**2. Tuple**

A record which is formed by a sequence of fields is known as Tuple. The fields in tuple can be of any type. It is similar to a row in RDBMS table.

Example : (Paresh, 26)

**3. Bag**

A bag is collection of tuples and these tuples are non-unique. Also every tuple can have any number of field and the fields can be of any type. A bag is represented by '{ }'.

Example - {(Paresh, 25), (Rupesh, 24)}

A bag can cpntain another bag it is known as inner bag.

Example - {Paresh, 25, {9730917725, paresh@gmail.com}}

**4. Map**

The collection of key-value pair is known as map. In this value can be of any type but key must be of type chararray and it should be unique.

It is represented by '[]'

Example - [name#Paresh, age#25]

**5. Relation**

A bag of tuples is known as relation. Relations in Pig are unordered. Pig does not guarantee processing of tuple in some specific order.

**Summer - 2017 (CSE)**

**Q.13** Explain working of Hive with proper steps and diagram. **(Refer Q.1 of Nov.16)** [7]

**Q.14** What is zookeeper ? What are the benefits of zookeeper. **(Refer section 5.4 and Q.2 of Nov.16)** [3.5]

**Q.15** Write a short note on Apache Pig. Enlist Applications of Apache Pig. **(Refer section 5.2)** [3.5]

**Q.16** What do you mean by HiveQL Data Definition Language ? Explain any three HiveQL DDL command with its syntax and example. **(Refer section 5.1.3)** [7]

**Winter - 2017 (IT)**

Q.17 Explain storage mechanism in HBase. **(Refer Q.6 of Nov.16)** [3]

Q.18 Explain MetaStore in Hive. **(Refer section 5.1.1.3)** [3]

Q.19 What do you mean by HiveQL Data Definition Language ? **(Refer section 5.1.3)** [4]

Q.20 Explain any three HiveQL DDL command with its syntax and example. **(Refer section 5.1.3)** [3]

Q.21 Compare Row oriented and column oriented database structure. **(Refer Q.6 of Nov.16)** [3]

Q.22 What is Zookeeper? List the benefits of it. **(Refer section 5.4 and Q.2of Nov.16 (IT))** [3]

Q.23 Differentiate : Apache Pig Vs MapReduce. **(Refer Q.3 of Nov.16 (IT))** [4]

Q.24 Explain how HBase uses zookeeper to build application with zookeeper. **(Refer section 5.4.2)** [7]

Q.25 Explain Pig data model in detail and discuss how it will be help for effective data flow. **(Refer Q.8 of May 17 (IT))** [7]

**Winter - 2017 (CSE)**

Q.26 Explain Hive Data types. **(Refer section 5.1.3)** [4]

Q.27 Explain the HiveQL-Select-Order By with suitable Example. **(Refer sections 5.1.3 and 5.1.4)** [7]

Q.28 Explain HBase architecture. **(Refer section 5.3)** [4]

Q.29 Define join and explain types of join. **(Refer section 5.1.4.3)** [7]

**Summer - 2018 (IT)**

Q.30 What is Zookeeper? What are the benefits of Zookeeper? **(Refer section 5.4 and Q.2 of Nov.16)** [3]

Q.31 Draw architecture of APACHE PIG and explain in short. **(Refer Q.7 of May 17 (IT))** [4]

Q.32 What is HBase? Write a query to create a table in HBase. [3]

**Ans. :** Refer section 5.3.

```
HBase query to create table:
    create 'customer', { NAME => 'CName'},
                        {NAME => 'Info'}
    create 'product_sales', {NAME => 'product'}
```

Q.33 Draw and explain Architecture of APACHE HIVE. Explain various data insertion techniques in HIVE with example. **(Refer sections 5.1.1 and 5.1.3.1)** [7]

**Summer - 2018 (CSE)**

Q.34 What is Zookeeper? List the benefits of it. **(Refer Q.2 of Nov.16 (IT))** [3]

Q.35 Explain HBase Architecture. **(Refer section 5.3)** [4]

Q.36 Explain working of Hive with proper steps and diagram. **(Refer Q.1 of Nov.16 (IT))** [7]

Q.37 Explain Storage mechanism in HBase. **(Refer Q.6 of Nov.16 (IT))** [3]

Q.38 Compare Raw oriented and Column Oriented database structures. **(Refer Q.6 of Nov.16 (IT))** [4]

Q.39 What do you mean by HiveQL Data Definition Language ? Explain any three HiveQL DDL command with its syntax and example. **(Refer section 5.1.3)** [7]

Q.40 Differentiate : Apache pig Vs Map Reduce. **(Refer Q.3 of Nov.16 (IT))** [4]

**Winter - 2018 (IT)**

Q.41 What is HBase? Explain storage mechanism of HBase with an example. **(Refer section 5.3 and Q.6 of Nov.16 (IT))** [7]

Q.42 Explain benefits of ZooKeeper. **(Refer Q.2 of Nov.16 (IT))** [3]

Q.43 Explain the architecture and features of HIVE, **(Refer section 5.1.1)** [7]

**Winter - 2018 (CSE)**

Q.44 What is HBase ? Differentiate HBase and RDBMS. **(Refer Q.6 of Nov.16 (IT))** [4]

Q.45 Write a short note on Apache Pig. **(Refer section 5.2)** [4]

Q.46 What is HiveQL ? Explain various statements in HiveQL with example. **(Refer section 5.1.3)** [4]

**Summer - 2019 (IT)**

Q.47 Explain CREATE TABLE statement of HIVE with an example. **(Refer section 5.1.3.1)** [3]

Q.48 Explain Apache Pig architecture and its components with diagram. **(Refer Q.7 of May 2017 (IT))** [7]

Q.49 What is Zookeeper ? Write functions of it. **(Refer section 5.4)** [3]

Q.50 Explain the architecture of HIVE. List out the features of HIVE. **(Refer section 5.1.1)** [7]

**Summer - 2019 (CSE)**

Q.51 Explain working of Hive with proper steps and diagram. **(Refer Q.1 of Nov.16 (IT))** [4]

Q.52 What is Zookeeper? List the Benefits and Limitations of it. **(Refer Q.2 of Nov.16 (IT))** [3]

Q.53 Explain Storage mechanism in HBase. **(Refer Q.6 of Nov.16 (IT))** [4]

Q.54 Explain the HiveQL-Select-Order By with suitable example. **(Refer sections 5.1.3 and 5.1.4)** [7]

**Winter - 2019 (IT)**

Q.55 Implement Join operation using Hive. **(Refer section 5.1.4.3)** [3]

Q.56 What is Zookeeper ? List the benefits of it. **(Refer Q.2 of Nov.16)** [3]

Q.57 Write a short note on Pig. **(Refer section 5.2)** [4]

Q.58 Explain any three HiveQL DDL command with its syntax and example. **(Refer section 5.1.3)** [3]

Q.59 What is HBase ? List out and explain the basic concepts of HBase in detail **(Refer section 5.3)** [3]

Q.60 Explain working of Hive with proper steps and diagram. **(Refer Q.1 of Nov.16 (IT)** [7]

Q.61 Explain Metastore in Hive. **(Refer section 5.1.1.3)** [3]

Q.62 Differentiate : Apache pig Vs Map Reduce. **(Refer Q.3 of Nov.16 (IT))** [3]

**Winter - 2019 (CSE)**

Q.63 Differentiate between HBase Vs. RDBMS. **(Refer Q.6 of Nov.16 (IT)** [4]

Q.64 Write a short note on Zookeeper. **(Refer section 5.4)** [3]

Q.65 Explain HIVE data types. **(Refer section 5.1.3)** [4]

Q.66 Write a short note on Pig. **(Refer section 5.2)** [3]

Q.67 Draw and explain HBase architecture. **(Refer section 5.3)** [4]

Q.68 Explain working of HIVE. **(Refer Q.1 of Nov.16 (IT))** [7]

❑❑❑

ameworks

# 6 Spark

***Syllabus***

*Introduction to Data Analysis with Spark, In-Memory Computing with Spark, Spark Basics, Interactive Spark with PySpark, Writing Spark Applications.*

***Contents***

## 6.1 Spark Basics

GTU : Winter-17, Summer-18, 19

- Apache Spark is a general-purpose and fast cluster computing platform. On the speed side, Spark extends the MapReduce model to support more types of computation efficiently which include query and stream processing. While processing huge volume of data speed is important.
- The key features Spark offers for speed is the ability to execute computation in memory, but Spark is more efficient than MapReduce for complex applications.
- Spark is designed to cover a wide range of workload that earlier required separate distributed systems, which includes iterative algorithm, interactive queries, batch applications and streaming. By providing support to these workloads in the same engine, Spark makes it simple and cost effective to merge various types of processing. Additionally it reduces the management burden needed for maintaining separate tools.
- Spark is designed to be highly accessible, providing rich built-in libraries, simple APIs in Java, Scala, Python and SQL.
- Big data tools can be easily integrated with Spark. Spark can run in Hadoop clusters and access any Hadoop data source, including Cassandra.

### 6.1.1 The Spark Stack

GTU : Winter-16, 17, 18, 19, Summer-17

- The Spark components are as following :
  - Cluster Managers
  - Spark Streaming
  - Spark Core
  - Spark SQL
  - MLlib
  - GraphX
- The above components are shown in the Fig. 6.1.1 the Spark Stack.

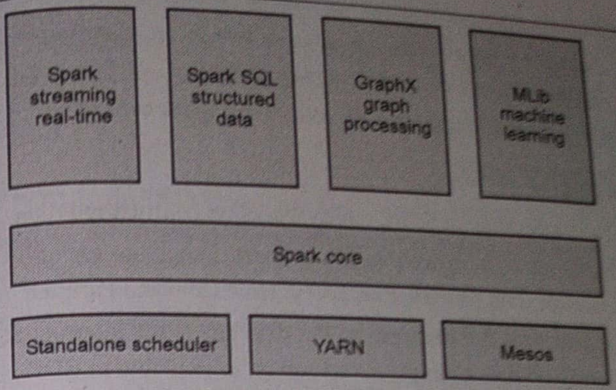


Fig. 6.1.1 The spark stack

#### 6.1.1.1 Cluster Managers

- The design of Spark is efficiently scale up from one to thousands of compute nodes. To achieve this while maximum flexibility, Spark may execute variety of cluster managers, including Apache Mesos, Hadoop YARN and a simple cluster manager comprised in Spark itself is known as the Standalone Scheduler.
- Spark support for these types of cluster managers permits your application to run on them.

#### 6.1.1.2 Spark Streaming

- It is a component of Spark which enables processing of live stream of data. Queue of messages which contains status updates posted by users of a web service or logfiles produced by production web servers, are example of data stream.
- Spark stream provides an API for manipulating stream of data which makes it simple for the programmers to learn the project and move between different applications which manipulate data stored on disk, in memory, or arriving at real time.
- Spark Streaming provides same throughput, fault tolerance, and scalability as Spark Core.

#### 6.1.1.3 Spark Core

GTU : Summer-17

- Spark core have the basic functionality of Spark, which includes components for memory management, task scheduling, fault recovery, interacting with storage system, etc.
- Spark core also contains API which defines Resilient Distributed Datasets (RDDs) and these RDDs are programming abstraction of Spark.

- RDDs correspond to a collection of item which are distributed over number of computing nodes that can manipulated in parallel.
- For creating and manipulating such collections Spark core provides various API.

### 6.1.1.4 Spark SQL

- Spark SQL is a package of Spark which works with structured data.
- It permits use of SQL as well as Hive Query Language for querying data and it supports various data sources, like, JSON, Hive table and Parquet.
- Spark SQL provides the platform to the developers for intermix SQL queries with the programmatic data manipulations which is available to RDDs in Python, Java, and scala, all inside single application, hence combining SQL with complex data analytics.
- Spark SQL was added as part of Spark in version 1.0.

### 6.1.1.5 MLlib

GTU : Winter-17, Summer-19

- The common machine learning functionality provided by spark as library is known as MLlib.
- Machine learning algorithms like, regression, classification, collaborative filtering and clustering, are provided by MLlib library.
- The model evaluation and data import functionality also supported by MLlib.

### 6.1.1.6 GraphX

GTU : Winter-17, Summer-19

- For performing manipulation on graph and graph-parallel computations, the GraphX library is used.
- To create directed graph having random properties associated to every vertex and edge, GraphX extends Spark RDD API.
- GraphX also provides different operators for manipulating graphs and common graph algorithm library.

## 6.1.2 Programming with RDDs

GTU : Summer-17, 18, 19, Winter-17, 18, 19

- RDD is distributed collection of elements. In Spark all job are carried out by creating new RDD, calling operations on RDD or transforming existing RDD to calculate the result.

- Spark automatically parallelize operations performed on RDD and distribute the data contained in RDD over the cluster.

### 6.1.2.1 RDD Basics

GTU : Summer-17, 19, Winter-18

- In Spark RDD is distributed collection of object which can't be modified. Every RDD is split into multiple parts, which can be computed on different nodes of the cluster.
- RDDs can have any type of Python, Java, or Scala objects, as well as user-defined classes.
- Users can create RDDs in two ways: first is by loading an external dataset and second is by distributing a collection of objects in their driver program. SparkContext.textFile() can be used to load a computer file as RDD of strings which is shown below :

```
>>> lines = sc.textFile("Test.md")
```

- RDDs support two types of operations, one is actions and other is transformations.
- Transformation creates new RDD from an existing RDD. Filtering data which matches the conditions is one of the examples of transformation.
- Following Python code shows calling filter() transformation:

```
>>> pythonLines = lines.filter(lambda line: "Python" in line)
```

- Actions compute results based on a RDD and either store it to an external storage system or return it to the driver program. The first() action is used to return the first element in RDD as shown below :

```
>>> pythonLines.first()
u'## Interactive Python Shell'
```

- Even though you can create new RDD anytime, Spark computes these RDD when the first time they are used in an action.
- For example, in above transformation and action, if Spark have to load and store the lines in the file as soon as the lines are written in lines= sc .textFile(...), it will waste a lot of space, given the situation that immediately after that we filter out lines.
- For first() action Spark scans file until it gets the first matching line, it does not read entire file. Hence, Spark's RDD are recomputed every time an action is executed on them.
- In order to reuse RDD in various actions, Spark need to persist RDD using RDD.persist().

- After computing for the first time, Spark saves the RDD contents in memory and reuse it whenever next action is encountered.
- The persist() can be frequently used to load the data into memory and query it repeatedly.
- Following example shows Python code for persisting an RDD in memory :

```
>>> pythonLines.persist
>>> pythonLines.count()
2
>>> pythonLines.first()
u'## Interactive Python Shell'
```

- The summary of Spark program is as follows :
  1. You can create new RDDs from external data.
  2. Transform the input RDD to create new RDDs using transformation like filter().
  3. You can reuse intermediate RDD contents by using persist() on RDD.
  4. Launch actions like count() and first() to start parallel computation, which is then optimized and executed by Spark.

### 6.1.2.2 Creating RDDs

- There are two approaches to create RDDs in Spark: by loading an external dataset and by parallelizing collections in driver program.
- For creating RDDs the simple approach is to take the collection in program and pass it to parallelize() method of SparkContext. By using this approach RDDs can be created quickly in the shell.
- But this approach is not widely used, since it needs complete dataset in memory on one machine.
- parallelize() method in Python can be used as :

```
lines = sc.parallelize(["apple", "i like apple"])
```

- Another approach to create RDDs is by loading datasets from the external storage. SparkContext.textFile() method is used to load external text file as an RDD of strings as show below :

```
lines = sc.textFile("/user/aj/Test.md")
```

### 6.1.2.3 RDD Operations

**GTU : Summer-17, 19, Winter-17, 18**

- Two operations are supported by RDDs which are: transformation and actions.
- A new RDD returned by transformations operations, such as map() and filter().

- Actions are operations which write result to storage or returns it to driver program and remove a computation, such as count() and first().
- In order to decide if the given function is transformation or action, return type can be checked. Data type are returned by actions, whereas RDDs returned by transformations.

#### Transformations

- Transformations are operations on RDDs which return a new RDD.
- When transformed RDDs are used in an action, it is computed lazily.
- Majority of transformations are element-wise; which means, at a time they work on single element; but it is not true for all transformation.
- For example, consider that you have a log file, log.txt, with number of messages and we need to pick error messages. For such cases filter() transformation can be used. filter() transformation in Python is as follows :

```
testRDD = sc.textFile("sample_log.txt")
errRDD = testRDD.filter(lambda x: "error" in x)
```

- The filter() operation does not change the current testRDD. Rather than it returns pointer to a new RDD. testRDD can be reused later in the application.
- Now we will use inputRDD again to search for lines which contain the word warning in it. After that using union() transformation to display lines that contain either warning or error. union() transformation in Python is as shown below :

```
errRDD = testRDD.filter(lambda x: "error" in x)
warningsRDD = testRDD.filter(lambda x: "warning" in x)
badLinesRDD = errRDD.union(warningsRDD)
```

- union() transformation operates on two RDDs rather than one RDD. Transformations can be operated on number of input RDDs.

#### Actions

- Actions are the second type of RDD operation. Actions are the type of an operation which write data to external storage or returns the final value to the driver program. Actions forces the evaluation of the transformations needed for the RDD they were called on, as they want to generate result.
- Continuing with the example from previous section, now we want to display information about the badLinesRDD. To perform this two actions, count(), which returns count as number, and take(), which takes different elements from the RDD. The Python code for counting bad lines is shown as follow :

```
print ("Input had " + badLinesRDD.count() + " bad lines")
print ("Here are 5 examples:")
for line in badLinesRDD.take(5):
print (line)
```

- To retrieve less number of elements in the RDD at driver program, take() action is used.
- RDD also support collect() method for retrieving the entire RDD. This can be helpful, in case the program filters the RDD to very small size and want to perform operation locally. For using collect() method entire dataset must be in memory on single machine. Hence collect() method should not be used for large dataset.
- In many cases, RDDs cannot only be collect() to the driver program as they are large in size. In such cases data is written to the distributed data storage like HDFS or Amazon S3.
- The content of RDD can be saved by using saveAsSequenceFile() or saveAsTextFile() action.

### Passing Functions to Spark

- Maximum transformation of Spark, and few of its actions, relies on passing in functions which are used by Spark to compute data. Every programming language has different approach for passing functions to Spark.

### Python

- In Python, there are three alternative for passing functions into Spark. The lambda expression can be passed for shorter functions. Optionally we can pass in top-level or in the functions defined locally. Passing of function in Python is shown below :

```
word = rdd.filter(lambda s: "error" in s)
def containsError(s):
return "error" in s
word = rdd.filter(containsError)
```

- One issue to handle is when passing functions is unintentionally serializing the object containing the function.
- When you pass a function which is the member of an object, or contains references to fields in an object, Spark sends the entire object to worker nodes, which can be much larger than the required information. Sometimes this can also make your program to fail, if your class contains objects that Python can't figure out how to pickle. Following example shows passing a function with field references, but it is not proffered :

```
class SearchFunctions(object):
def __init__(self, query):
self.query = query
def isMatch(self, s):
return self.query in s
def getMatchesFunctionReference(self, rdd):

return rdd.filter(self.isMatch)
def getMatchesMemberReference(self, rdd):

return rdd.filter(lambda x: self.query in x)
```

- Instead of passing the field references, retrieve the field from object and store it in local variable and pass that variable. Following code shows function passing without references to fields in Python :

```
class WordFunctions(object):
...
def getMatchesNoReference(self, rdd):
query = self.query

return rdd.filter(lambda x: query in x)
```

### Basic RDD Transformation

- The two most common transformations are filter() and map().
- The map() transformation takes in a function and applies it to every element in the RDD with the result of function being the new value for every element in resulting RDD.
- The filter() transformation takes in function and provides a RDD which only contain the elements which pass filter() function.
- The map() and filter() transformation on inputRDD is shown in below Fig. 6.1.2 :

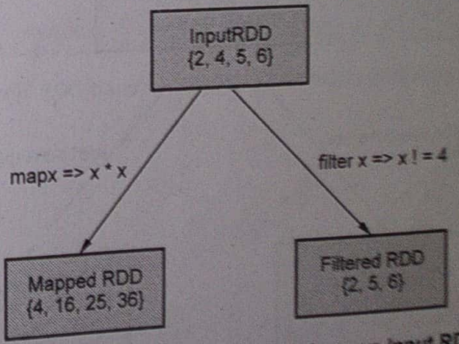


**Fig. 6.1.2 Mapped and filtered RDD from an input RDD**

- We can use map() to do any range of things, from fetching the internet site related to every URL in collection to squaring the numbers.
- The return type of map() does not have to be similar as of its input type. So, in case we have RDD string and map() function have to parse the strings and return double, our input RDD type will be RDD[String] and the resulting RDD type will be RDD[Double].
- Following example used map() which squares all the numbers present in RDD.
- Let's look at a basic example of map() that squares all of the numbers in an RDD . Following code shows Python code to use map() :

```
numbers = sc.parallelize([1, 2, 3, 4])
sqred = numbers.map(lambda x: x * x).collect()
for n in sqred:
print( "%i " ,n)
```

- Many times for every input element we need to generate multiple output elements. The flatMap() is used to perform this operation. Similar with map(), the function provided to flatMap() is called individually for every element in input RDD.
- An iterator is returned along with return value instead of returning a single element. Instead of generating a RDD of iterators, we get RDD which consists of elements from all of the iteratots.
- Following example shows use of flatMap() for splitting input string into words :

```
lines = sc.parallelize(["send mail", "reply"])
words = lines.flatMap(lambda line: line.split(" "))
words.first() # returns "send"
```

- The difference between flatMap() and map() is illustrated in figure below. The flatMap() can be thought of as flattening the iterators returned to it, so that rather than ending up with RDD of lists we get RDD of elements in those lists.

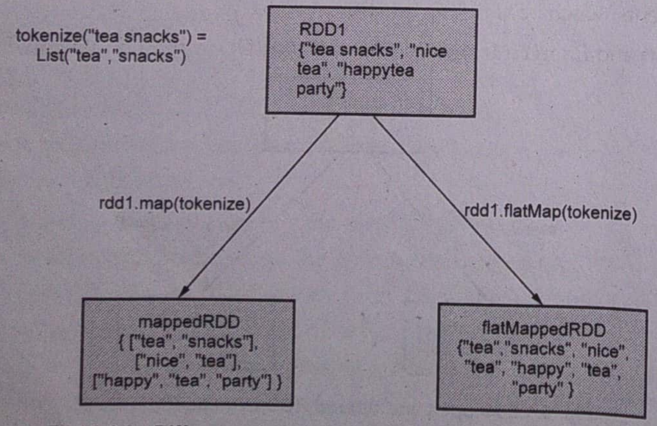

tokenize("tea snacks") = List("tea","snacks")
RDD1 {"tea snacks", "nice tea", "happytea party"}
rdd1.map(tokenize)
rdd1.flatMap(tokenize)
mappedRDD { ["tea", "snacks"], ["nice", "tea"], ["happy", "tea", "party"] }
flatMappedRDD {"tea","snacks", "nice", "tea", "happy", "tea", "party" }

Fig. 6.1.3 : Difference between map() and flatMap() on RDD

- Following Table 6.1.1 shows basic RDD transformations :

| Function Name | Purpose | Example | Result |
|---|---|---|---|
| map() | Function is applied to every element in input RDD and gives an RDD of the result. | rdd.map(x => x + 1) | {2, 3, 4, 4} |
| flatMap() | Function is applied to every element of RDD and gives RDD of contents of iterators returned. Frequently used for extracting words. | rdd.flatMap(x => x.to(3)) | {1, 2, 3, 2, 3, 3, 3} |
| filter() | It gives RDD containing only those elements that pass the condition which is passed to filter(). | rdd.filter(x => x != 1) | {2, 3, 3} |
| distinct() | It removes the duplicate elements. | rdd.distinct() | {1, 2, 3} |
| sample (withReplacement, fraction, [seed]) | RDD is sampled with or without replacement. | rdd.sample(false, 0.5) | Nondeterministic |

Table 6.1.1 : Basic RDD transformations on an RDD containing {1, 2, 3, 3}

- Following Table 6.1.2 shows Two RDD transformation :

| Function Name | Purpose | Example | Result |
|---|---|---|---|
| union() | Generated RDD which contains elements from both RDDs. | rdd.union(other) | {1, 2, 3, 3, 4, 5} |
| intersection() | It returns RDD which contains elements which is found in both the RDDs. | rdd.intersection(other) | {3} |
| subtract() | It removes content of one RDD. | rdd.subtract(other) | {1, 2} |
| cartesian() | It returns Cartesian profuct with other RDD. | rdd.cartesian(other) | {(1, 3), (1, 4), ... (3,5)} |

**Table 6.1.2 : Two-RDD transformations on RDDs containing {1, 2, 3} and {3, 4, 5}**

## Actions

- The reduce() is one of the3 most common RDD action which is used, it takes a function which operates on two element of the type in your RDD and returns new element having the same type.
- An easy example of such function is +, that can be used to sum our RDD. With reduce(), sum of elements of RDD can be easily done, used to count the number of elements and various types of aggregations can be performed.
- Following example shows the use of reduce() in Python :

  sum = rdd.reduce(lambda x, y: x + y)
- The aggregate() can be used to calculate average of RDD.

```
sumCount = nums.aggregate((0, 0),
(lambda acc, value: (acc[0] + value, acc[1] + 1),
(lambda acc1, acc2: (acc1[0] + acc2[0], acc1[1] + acc2[1]))))
return sumCount[0] / float(sumCount[1])
```

- Following Table 6.1.3 shows some actions on RDD.

| Function Name | Purpose | Example | Result |
|---|---|---|---|
| collect() | Returns every elements from the RDD. | rdd.collect() | {1, 2, 3, 3} |
| count() | Counts number of elements in input RDD. | rdd.count() | 4 |
| countByValue() | Returns count of occurrence of every element in the RDD. | rdd.countByValue() | {(1, 1), (2,1), (3, 2)} |
| take(num) | Num elements are returned from the RDD. | rdd.take(2) | {1, 2} |
| top(num) | Top num elements are returned from the RDD. | rdd.top(2) | {3, 3} |
| takeOrdered(num)(ordering) | Num elements are returned depending on given ordering. | rdd.takeOrdered(2)(myOrdering) | {3, 3} |
| takeSample(withReplacement, num,[seed]) | Num elements are returned randomly. | rdd.takeSample(false, 1) | Nondeterministic |
| reduce(func) | Merges elements of RDD together in parallel. | rdd.reduce((x, y) => x + y) | 9 |
| fold(zero)(func) | Same as reduce() but with the given zero value. | rdd.fold(0)((x, y) => x + y) | 9 |
| Aggregate (zeroValue)(seqOp,combOp) | Similar to reduce() but used to return a different type. | rdd.aggregate((0,0))((x, y) => (x_1 + y, x_2 + 1), (x, y) => (x_1 + y_1, x_2 + y_2)) | (9, 4) |
| foreach(func) | Given function is applied to every element of the RDD. | rdd.foreach(func) | Nothing |

**Table 6.1.3 Basic actions on an RDD containing {1, 2, 3, 3}**

## 6.2 Data Analysis with Spark

- As Spark is general-purpose framework for cluster computing, it is used for a various applications. Spark can be used by two type of task, first is Data Science Task and second Data Processing Application.

### 6.2.1 Data Science Tasks

- Data science is a branch which has emerged in past few year focuses on data analysis. To analyze and model data are the main task of data scientist.
- Data scientist might have experience in SQL, machine learning, statistics and programming, generally in Matlab, Python or R. They also possess experience with the techniques required for transforming data into different formats which can be analyzed to discover patterns.
- Data scientists utilize their skills for analyzing data with the aim of discovering patterns. Many times their workflow includes ad hoc analysis, so they make use of interactive shell that displays the result of query and snippet of code is also included less frequently.
- For this purpose the speed and simple APIs of Spark are very helpful.
- Spark provides support for various tasks of data science with number of components. The spark shell makes it simple to perform interactive data analysis using Python or Scala.
- For performing data exploration using SQL, Spark SQL's separate shell is used.
- MLlib libraries support machine learning and data analysis. Spark enables data scientists to solve problem with huge amount of data.
- For example, the initial research of data scientists may cause the creation of product recommender system which is integrated with a web application and used to generate product suggestions for users.

### 6.2.2 Data Processing Applications

- The other use of Spark can be thought of in context of engineer. For engineers, Spark provides a easy approach for parallelizing applications over clusters. Spark also helps to hide the complexity of network communication, distributed system programming and fault tolerance.
- Sufficient control is given to engineers for monitoring, inspecting and tuning applications while permitting them to implement the common task quickly.

- The modular nature of API makes it simple to divide the work into reusable libraries and test it locally.
- Spark's user selects it for data processing applications, as it provide a wide range of functionality and is reliable.

## 6.3 In-Memory Computing with Spark

- When the data is kept in RAM in place of disk drives and processing is done parallelly then it is known as in-memory computation. This helps to perform analysis of large data set and also for pattern detection. Since in-memory computing technique reduces the cost on memory, this processing technique is said to be economic for applications.
- Two main pillars of in-memory processing are RAM and parallel distributed computing.
- Performance of computation improves by keeping data in memory. RDDs are the main abstraction of Spark and which are cashed by use of cache() or persist() functions.
- All RDDs gets stored in-memory when cache() function is used. When these RDDs gets stored in memory, excess data is sent back to disk or it gets recalculated. RDDs can be extracted without going to disk, which makes the process faster.
- The capability of Spark to process data in-memory is beneficial for micro bacth processing and machine learning. It also helps to execute iterative jobs faster.
- RDDs can also get stored in-memory by using persist() function, which can be used across parallel operations. The major difference between cache() and persist() function is that, default storage level while using cache() is MEMORY_ONLY, whereas while using persist() there are various storage level options are available to use.

### 6.3.1 Storage Levels of RDD persist() in Spark

Below are different storage levels of persist() in Spark RDD :

**i. MEMORY_ONLY**

- RDDs get stored as deserialized JAVA object in JVM in this storage level. In case the RDD does not fit in memory, then whenever remaining part of RDD is required it will get recalculated.

### ii. MEMORY_AND_DISK

- In this storage level also RDDs gets stored in JVM as deserialized JAVA object. In case of excess data, remaining part of RDD is sent back to disk in place of recalculating it each time whenever required.

### iii. MEMORY_ONLY_SER

- RDDs gets stored as serialized JAVA object. In this every partition stores one-byte array. It is similar to MEMORY_ONLY, but if used fast serializer it is more space efficient.

### iv. MEMORY_AND_DISK_SER

- RDDs gets stored as serialized JAVA object. In case entire RDD does not fit in memory then remaining part of RDD is stored on disk rather than recomputing it each time when required

### v. DISK_ONLY

- RDD partition gets stored only on disk.

### vi. MEMORY_ONLY_2 and MEMORY_AND_DISK_2

- It is similar to MEMORY_ONLY and MEMORY_AND_DISK. The major difference is, in this storage level every partition is replicated on two nodes in the cluster.

## 6.4 Interactive Spark with PySpark

- For enabling ad-hoc data analysis Spark contains interactive shells. Spark's shell is similar to the shells of R, Python and Scala or OS shells like windows command prompt.
- Spark's shells permits to interact with data which is distributed on disk or in memory over many machines, and Spark handles the automatic distribution of this processing.
- Since Spark loads the data into memory on the worker node, various distributed computation can be executed within few seconds.
- Spark provides Python and Scala shells which have augmented to support connecting to a cluster.
- These shells are helpful for learning the API. In this chapter we recommend to use Python shell. The API is similar in every language.
- First step is to open Python version of Spark shell, which is referred to as PySpark Shell, go into the Spark directory and type :

```
bin/pyspark
```

- The shell prompt will appear in few seconds. When the shell start there will be many log messages, you need to press Enter once to clear the log output and go to shell prompt.

### 6.4.1 Using IPython

- Many Python users prefer IPython which is an enhanced Python shell.
- In oreder to use IPython with Spark you need to set the IPYTHON environment variable to 1, as follow :

```
IPYTHON=1 ./bin/pyspark
```

- To use web-based version of IPython i.e. IPython Notebook :

```
IPYTHON_OPTS="notebook" ./bin/pyspark
```

- On Windows, set the variable and run the shell as follows :

```
set IPYTHON=1
bin\pyspark
```

- In Spark, computations are expressed through operations on distributed collections which are parallelized automatically over the cluster. These collections are known as *resilient distributed datasets* i.e. RDDs.
- For distributed data and computation, RDDs are basic abstractions in Spark.
- Following example shows simple ad hoc analysis on a text file using Python :

```
>>> lines = sc.textFile("Test.md") # Create an RDD called lines
>>> lines.count() # Count the number of items in this RDD
127
```

- To exit either shell, press Ctrl-D.

## 6.5 Writing Spark Application

- Apart from running interactively, Spark can be linked to standalone applications in languages like java, Scala or Python. While using Spark with standalone application you need to initialize your own SparkContext, which was not needed while using Spark in interactive shell.
- The process of linking to Spark varies from language to language. In Python, you need to write applications as Python script, bust you will have to execute those using bin/spark-submit script which is included in Spark.
- The spark-submit includes the Spark dependencies which is helpful to set the environment for Spark's Python API to function.
- Following command shows running of Python script :

```
bin/spark-submit my_pyscript.py
```

### 6.5.1 Initializing a SparkContext

- Once the application is linked to Spark, you will need to import the packages of Spark in your program and create SparkContext.
- Following Python code shows initializing Spark :

```
from pyspark import SparkConf, SparkContext
conf = SparkConf().setMaster("local").setAppName("SampleApp")
sc = SparkContext(conf = conf)
```

- To initialize SparkContext, you need to pass two parameters which are as follow:
- *A cluster URL* : In above example "local", that tells Spark how to connect to Hadoop cluster. Local is a special value which runs Spark on one thread on the local system, without connecting to cluster.
- *An application name* : In above example "SampleApp", it will identify your application on the interface of cluster manager, in case you connect to a cluster.
- Once the SparkContext is initialized, you can use built in methods for creating and manipulating RDDs.
- In order to shut down Spark, you can either exit the application, or you can call stop() method on your SparkContext.

### 6.5.2 Writing Standalone Applications

- In this section we will write a standalone word count application which can run on single machine as well as on Hadoop cluster.
- Following is Python code for Word Count Application :

```
WordCount.py
## Imports
from pyspark import SparkConf, SparkContext
from operator import add
import sys

## Constants
APP_NAME = " WordCount"

##OTHER FUNCTIONS/CLASSES
def main(sc,fname):
sampleRDD = sc.textFile(fname)
words = sampleRDD.flatMap(lambda x: x.split(' ')).map(lambda x: (x, 1))
wordcount = words.reduceByKey(add).collect()
for wc in wordcount:
print wc[0],wc[1]
```

```
if __name__ == "__main__":

# Configure Spark
conf = SparkConf().setAppName(APP_NAME)
conf = conf.setMaster("local[*]")
sc   = SparkContext(conf=conf)
file_name = sys.argv[1]
# Execute Main functionality
main(sc, file_name)

input.txt:
red green blue black
green black
red white orange
red blue red blue
```

- Following command is used to execute the application. The command will submit an application to the cluster for execution :

```
$spark-submit WordCount.py input.txt
```

```
Output:
red 4 blue 3 black 2 green 2 orange 1 white 1
```

**University Questions with Answers**

**Winter - 2016 (IT)**

**Q.1** Explain Spark components in detail. Also list the features of spark. **(Refer section 6.1.1)** [7]

**Summer - 2017 (IT)**

**Q.2** What is RDD ? Explain about transformations and actions in the context of RDDs. State and explain RDD operations in brief. **(Refer sections 6.1.2.1 and 6.1.2.3)** [7]

**Q.3** What is spark ? State the advantages of using Apache Spark over Hadoop MapReduce for Big Data Processing with example. [7]

**Ans. : Refer section 6.1.**

**Advantages of using Apache Spark over Hadoop MapReduce :**

- Spark can process data in batch, interactive, iterative and streaming mode, whereas MapReduce can process data only in batch mode.
- Spark performs in-memory computing which makes Spark faster than MapReduce. On the other side MapReduce reads and writes from disk which slows down the processing speed.