**Big Data and Hadoop Data Analysis on**

**Movie Dataset and YouTube Dataset**

**A Project Report for Industrial Training**

###### ***Submitted by***

Vipul Kumar

Vijay Kumar

Rohit Gaur

Gautam Kumar

Md Washid Husain

Md Sharique Khan

Aisha Ali

Ankita Prasad

***In the partial fulfillment for the award of the degree of***

##### **B. Tech**

in

IT

College Logo

**NETAJI SUBHASH ENGINEERING COLLEGE**

**February 2017**



**CERTIFICATE FROM SUPERVISOR**

This is to certify that Vipul Kumar, Vijay Kumar, Rohit Gaur, Gautam Kumar, Md Washid Husain, Md Sharique Khan, Aisha Ali and Ankita Prasad have successfully completed the project entitled **"Big Data And Hadoop Data Analysis on Movie Dataset and YouTube Dataset**" under my supervision during the period from June to July which is in partial fulfilment of requirements for the award of the Bachelor in Technology in Information Technology and submitted to Department of Information Technology, NSEC, Kolkata.

*Signature of the Supervisor*

**Date:**

**Name & Designation**

**Project Supervisor**

**ACKNOWLEDGEMENT**

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentors, Mr. Rajmohan De Sarkar and Ms. Stuti Kumari for giving such valuable suggestions, guidance and encouragement during the development of this project work.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **TOPICS** | **PAGE NO** |
| 1 | ABSTRACT | 5 |
| 2 | INTRODUCTION | 6 |
| 3 | THEORY | 9 |
| 4 | PROBLEM DOMAIN | 13 |
| 5 | METHODOLOGY | 13 |
| 6 | DATA ANALYSIS AND FLOW DIAGRAM | 15 |
| 7 | IMPLEMENTATION | 17 |
| 8 | CONCLUSION | 23 |
| 9 | REFERENCES | 24 |

**List Of Figures:**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Name** | **Page No.** |
| 1 | A Multi-Node Hadoop Cluster | 9 |
| 2 | Basic Cluster Configuration | 10 |
| 3 | The Overall MapReduce Word Count Process | 12 |
| 4 | A SDLC life cycle model (Iterative Waterfall Model) | 16 |
| 5 | HIVE query execution | 17 |
| 6 | Hive Flow Chart | 18 |

1. **ABSTRACT**

We live in the data age. It’s not easy to measure the total volume of data stored electronically, but according to an IDC estimate, from 2005 to 2020, the digital universe will grow by a factor of 300, from 130 Exabytes to 40,000 Exabytes, or 40 trillion gigabytes (more than 5,200 gigabytes for every man, woman, and child in 2020). From now until 2020, the digital universe will about double every two years.

* The New York Stock Exchange generates about one terabyte of new trade data per day.
* Facebook hosts approximately 10 billion photos, taking up one petabyte of storage.
* The Internet Archive stores around 2 petabytes of data, and is growing at a rate of

1. Terabytes per month.

* YouTube users upload 48 hours of new video every minute of the day.
* The Large Hadron Collider near Geneva, Switzerland, will produce about 15petabytes of data per year.

It has been said that “More data usually beats better algorithms,” which is to say thatfor some problems (such as recommending movies or music based on past preferences),however fiendish your algorithms are, they can often be beaten simply by having more data (and a less sophisticated algorithm).

The good news is that Big Data is here. The bad news is that we are struggling to storeand analyze it.

The problem is simple: while the storage capacities of hard drives have increased massively over the years, access speeds—the rate at which data can be read from drives— have not kept up. One typical drive from 1990 could store 1,370 MB of data and had a transfer speed of 4.4 MB/s. Over 20 years later, one terabyte drives are the norm, but the transferspeed is around 100 MB/s. This is a long time to read all data on a single drive—and writing is even slower. Theobvious way to reduce the time is to read from multiple disks at once. Imagine if wehad 100 drives, each holding one hundredth of the data. Working in parallel, we couldread the data in two minutes.

There’s more to being able to read and write data in parallel to or from multiple disks, though.

One of the problems is that most analysis tasks need to be able to combine the data in some way; data read from one disk may need to be combined with the data from anyof the other 99 disks. Various distributed systems allow data to be combined frommultiple sources, but doing this correctly is notoriously challenging. MapReduce providesa programming model that abstracts the problem from disk reads and writes,transforming it into a computation over sets of keys and values.

Thus, our project is to analyse the big data over the datasets of Movie Data Analytics and YouTube using Apache PIG and Hive.

1. **INTRODUCTION**
   1. **INTRODUCTION TO BIGDATA**

**Big data** is a term for data sets that are so large or complex that traditional data processing applications are inadequate. **Challenges** include

* Analysis,
* Capture,
* Data curation,
* Search,
* Sharing,
* Storage,
* Transfer,
* Visualization,
* Querying,
* Updating and information privacy.

**Big data can be described by the following characteristics**:

**Volume**

The quantity of data generated and stored. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not.

**Variety**

The type and nature of the data. This helps people who analyze it to effectively use the resulting insight.

**Velocity**

In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.

**Variability**

Inconsistency of the data set can hamper processes to handle and manage it.

**Veracity**

The quality of captured data can vary greatly, affecting accurate analysis.

**2.2 INTRODUCTION TO HADOOP**

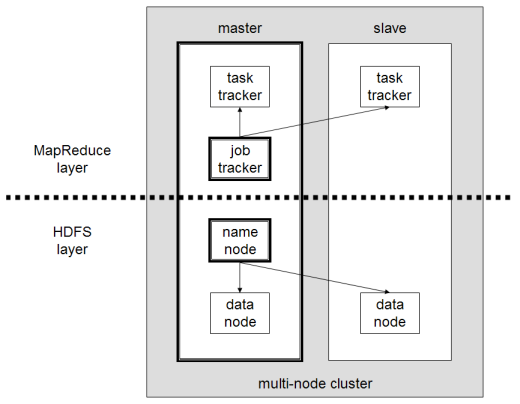
**What is Hadoop?**

We live in the age of big data, where the data volumes we need to work with on a day-to-day basis have outgrown the storage and processing capabilities of a single host. Big data brings with it two fundamental challenges: how to store and work with voluminous data sizes, and more important, how to understand data and turn it into a competitive advantage. Hadoop fills a gap in the market by effectively storing and providing computational capabilities over substantial amounts of data. It’s a distributed system made up of a distributed filesystem and it offers a way to parallelize and execute programs on a cluster of machines.

Distributed Computation The computation tier uses a framework called MapReduce.

Distributed Storage A distributed filesystem called HDFS provides storage.

Hadoop is a platform that provides both distributed storage and computational capabilities. Hadoop was first conceived to fix a scalability issue that existed in Nutch,2 an open source crawler and search engine. At the time Google had published papers that described its novel distributed filesystem, the Google File System (GFS), and Map-Reduce, a computational framework for parallel processing. The successful implementation of these papers’ concepts in Nutch resulted in its split into two separate projects, the second of which became Hadoop, a first-class Apache project. Hadoop proper, is a distributed master-slave architecture3 that consists of the Hadoop Distributed File System (HDFS) for storage and Map-Reduce for computational capabilities. Its storage and computational capabilities scale with the addition of hosts to a Hadoop cluster, and can reach volume sizes in the petabytes on clusters with thousands of hosts. Hadoop is capable of analyzing large amount of data. Hadoop is developed by keeping most of the things in mind like-large dataset, write once read many access models, moving computation is cheaper than moving data etc. All this capability makes Hadoop suitable for most mining problems. Hadoop’s own file system, Hadoop Distributed File system (HDFS) is capable of running on commodity hardware with high fault tolerance ability. Data replication is one of the important features of HDFS, which ensures data availability and automatic re-execution on multiple node failure. In this project, the algorithm will use the power of Hadoop for mining the frequent Itemset. Hadoop is a, Java-based programming framework that supports the processing of large data sets in a distributed computing environment and is part of the Apache project sponsored by the Apache Software Foundation. Hadoop can provide much needed robustness and scalability option to a distributed system as Hadoop provides inexpensive and reliable storage. The Apache Hadoop software library can detect and handle failures at the application layer, and can deliver a highly-available service on top of a cluster of computers, each of which may be prone to failures.



**Figure 1: A Multi-Node Hadoop Cluster**

**Core Hadoop components**

Hadoop consists of two core components:

– The Hadoop Distributed File System (HDFS)

– MapReduce

**Hadoop Distributed File System (HDFS)**

The **H**adoop ***D***istributed ***F***ile ***S***ystem (HDFS) is a sub-project of the Apache Hadoop project. This Apache Software Foundation project is designed to provide a fault-tolerant file system designed to run on commodity hardware.

**How Files Are Stored**

* Files are split into blocks
* Each block is usually 64 MB or 128 MB
* Data is distributed across many machines at load time
* Different blocks from the same file will be stored on different machines
* This provides for efficient MapReduce processing
* Blocks are replicated across multiple machines, known as DataNodes
* Default replication is three-fold
* Meaning that each block exists on three different machines
* A master node called the NameNodekeeps track of which blocks make up a file, and where those blocks are located
* Known as the metadata

**Accessing HDFS**

* Applications can read and write HDFS files directly via the Java API
* Typically, files are created on a local filesystem and must be moved into HDFS
* Likewise, files stored in HDFS may need to be moved to a machine’s local filesystem
* Access to HDFS from the command line is achieved with the hadoop fs command

**The Five Hadoop Daemons**

Hadoop is comprised of five separate daemons:

1. NameNode
   * Holds the metadata for HDFS
2. Secondary NameNode
   * Performs housekeeping functions for the NameNode
   * Is not a backup or hot standby for the NameNode.
3. DataNode
   * Stores actual HDFS data blocks
4. JobTracker
   * Manages MapReduce jobs, distributes individual tasks to machines running the TaskTracker
5. TaskTracker
   * Instantiates and monitors individual Map and Reduce tasks

We can consider nodes to be in two different categories:

1. Master Nodes
   * Run the NameNode, Secondary NameNode, JobTracker daemons
   * Only one of each of these daemons runs on the cluster
2. Slave Nodes
   * Run the DataNode and TaskTracker daemons
   * A slave node will run both of these daemons



**Figure 2: Basic Cluster Configuration**

On very small clusters, the NameNode, JobTracker and Secondary NameNode can all reside on a single machine. It is typical to put them on separate machines as the cluster grows beyond 20-30 nodes. Each dotted box on the diagram represents a separate Java Virtual Machine (JVM).

1. **THEORY**

**MapReduce**

**MapReduce** is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster.

**Inputs and Outputs**

The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types. The key and value classes have to be serializable by the framework and hence need to implement the Writable interface. Additionally, the key classes have to implement the WritableComparable interface to facilitate sorting by the framework.

Input and Output types of a MapReduce job:

(input) <k1, v1> ->map -><k2, v2> ->combine -><k2, v2> ->reduce -><k3, v3> (output)

**MapReduce consists of two phases:**

– Map

– Reduce

## **The Algorithm**

* Generally MapReduce paradigm is based on sending the computer to where the data resides!
* MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
  + **Map stage**: The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
  + **Reduce stage**: This stage is the combination of the*Shuffle*stage and the *Reduce* stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
* During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
* The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
* Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
* After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



**Figure 3: The Overall MapReduce Word Count Process**

**Features of MapReduce**

* Automatic parallelization and distribution
* Fault-tolerance
* Status and monitoring tools
* A clean abstraction for programmers
* MapReduce abstracts all the ‘housekeeping’ away from the developer, i.e., the developer can concentrate simply on writing the Map and Reduce functions.

**Advantages of Hadoop**

* Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
* Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
* Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
* Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

**Limitations of Hadoop**

## **Security Concerns**

Just managing a complex application such as Hadoop can be challenging. A classic example can be seen in the Hadoop security model, which is disabled by default due to sheer complexity. If whoever’s managing the platform lacks the knowhow to enable it, your data could be at huge risk. Hadoop is also missing encryption at the storage and network levels, which is a major selling point for government agencies and others that prefer to keep their data under wraps.

## **Vulnerable By Nature**

Speaking of security, the very makeup of Hadoop makes running it a risky proposition. The framework is written almost entirely in Java, one of the most widely used yet controversial programming languages in existence. Java has been heavily exploited by cybercriminals and as a result, implicated in numerous security breaches. For this reason, several experts have suggested dumping it in favour of safer, more efficient alternatives.

## **Not Fit for Small Data**

While big data isn’t exclusively made for big businesses, not all big data platforms are suited for small data needs. Unfortunately, Hadoop happens to be one of them. Due to its high capacity design, the Hadoop Distributed File System or HDFS, lacks the ability to efficiently support the random reading of small files. As a result, it is not recommended for organizations with small quantities of data.

## **Potential Stability Issues**

Hadoop is an open source platform. That essentially means it is created by the contributions of the many developers who continue to work on the project. While improvements are constantly being made,

Like all open source software, Hadoop has had its fair share of stability issues. To avoid these issues, organizations are strongly recommended to make sure they are running the latest stable version, or run it under a third-party vendor equipped to handle such problems.

## **General Limitations**

One of the most interesting highlights of the Google article referenced earlier mentions that when it comes to making the most of big data, Hadoop may not be the only answer. The article introduces Apache Flume, MillWheel, and Google’s own Cloud Dataflow as possible solutions. What each of these platforms have in common is the ability to improve the efficiency and reliability of data collection, aggregation, and integration. The main point the article stresses is that companies could be missing out on big benefits by using Hadoop alone.

**Apache Pig**

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

**Features of Pig**

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

**R Programming**

R is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University Of Auckland, New Zealand, and named after the first letters of names of the two authors.

**Features of R**

* R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
* R has an effective data handling and storage facility,
* R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
* R provides a large, coherent and integrated collection of tools for data analysis.
* R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

**Hive**

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

## **Features of Hive**

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

1. **PROBLEM DOMAIN**

The problem is to analyse the huge static structured datasets of movie websites and youtube. This is done by generating various queries to the hadoop framework in which they are stored. We will use Pig framework to analyse the movie dataset and Hive framework to analyse the youtube dataset.

We have been provided with the datasets of:

* Movie Data Analytics, and
* YouTube

For **Movie Data Analytics**, we have been provided with the following relations with the attributes described correspondingly:

Column1: Movie ID

Column2: Movie name

Column3: Year of release

Column4: Rating of the movie

Column5: Movie duration in seconds

For **YouTube Data Analytics**, we have been provided with the following relations with the attributes described correspondingly:

Column1: Video id of 11 characters.

Column2: Uploader of the video of string data type.

Column3: Interval between day of establishment of YouTube and the date of uploading of the video of integer data type.

Column4: Category of the video of String data type.

Column5: Length of the video of integer data type.

Column6: Number of views for the video of integer data type.

Column7: Rating on the video of float data type.

Column8: Number of ratings given on the video.

Column9: Number of comments on the videos in integer data type.

Column10: Related video ids with the uploaded video.

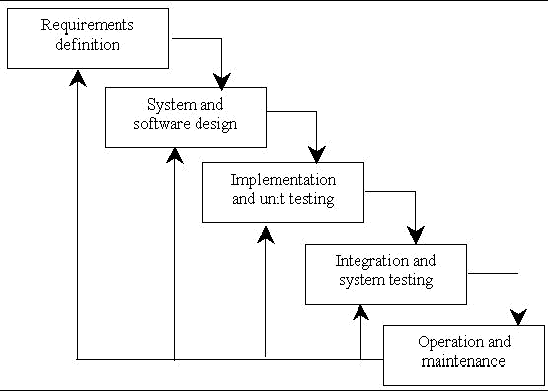
1. **METHODOLOGY**

The intent of a SDLC process it to help produce a product that is cost efficient, effective, and of high quality. Once an application is created, the SDLC maps the proper deployment and decommissioning of the software once it becomes a legacy. The SDLC methodology usually contains the following stages: Analysis (requirements and design), construction, testing, release, and maintenance (response).

There are various types of Software Development Life Cycle models. The types are:

1. Water Fall Model
2. Spiral Model
3. V Model
4. Iterative Model

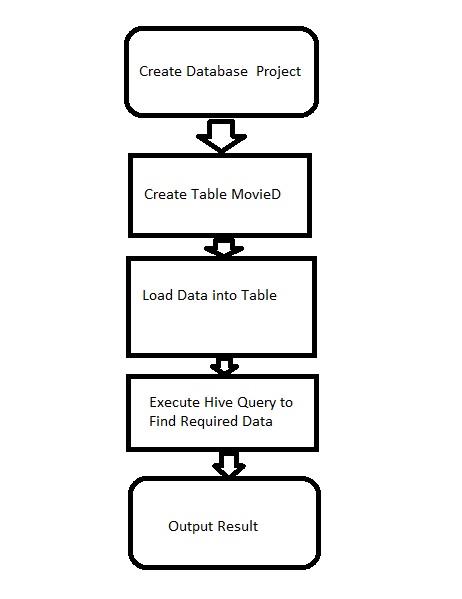
We use iterative waterfall model in our project.

**Fig.4: A SDLC life cycle model (Iterative Waterfall Model)**

**DATA ANALYSIS FLOW DIAGRAM**

****

**Fig 5: HIVE query execution**

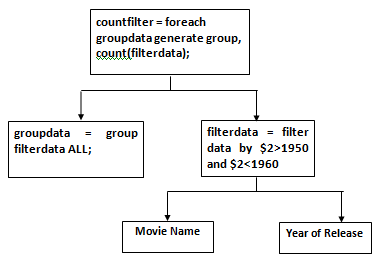
****

**Fig 6: Hive Flow Chart**

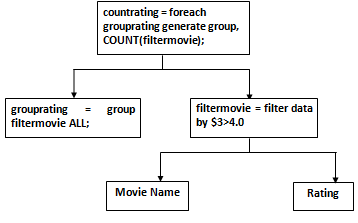
1. **DATA ANALYSIS AND FLOW DIAGRAM**

The data analysis flow diagrams for the movie dataset are as follows:

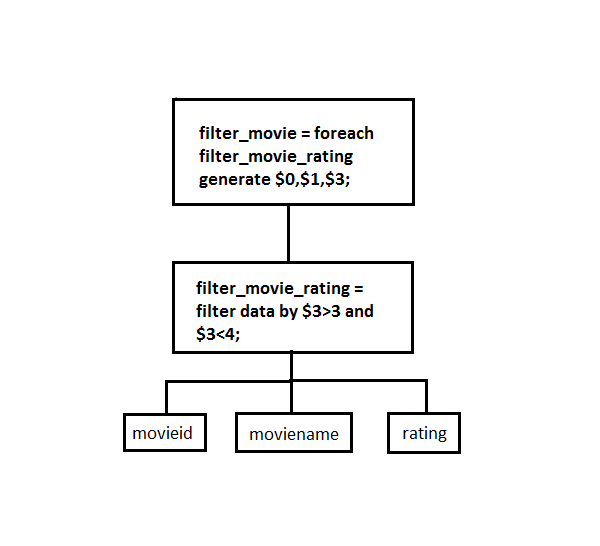
* 1. **Find the no of movies released between 1950 and 1960.**

****

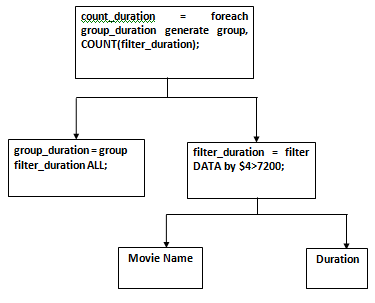
* 1. **Find the number of movies having rating more than 4.**



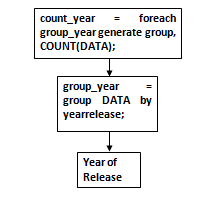
**C. Find the movies whose ratings are between 3 and 4.**

****

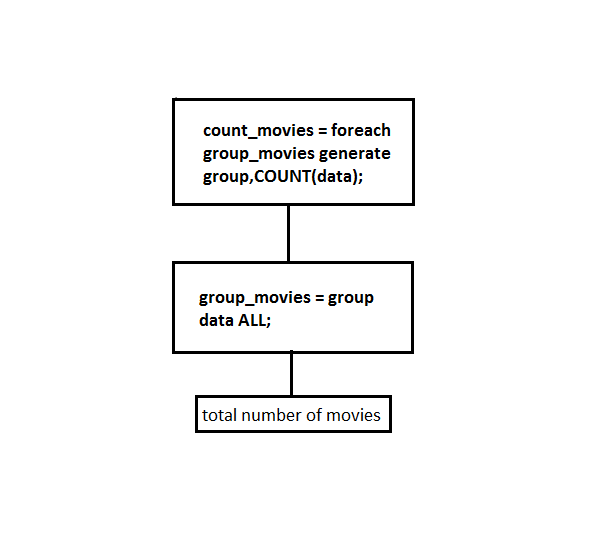
**D. Find the number of movies with duration more than 2 hours(7200 seconds).**

****

**E. Find the list of years and number of movies released each year.**

****

**F. Find the total number of movies in the dataset.**

****

1. **IMPLEMENTATION** 
   1. **MOVIE DATA ANALYTICS USING PIG**
2. **Creation of directory in Local File System(LFS) and storing the dataset:**

$mkdir proj\_lfs

$gedit proj\_lfs/moviedata.txt //complete movie dataset

1. **Creation of directory in Hadoop File System(HDFS) and copying the dataset from LFS TO HDFS:**

$hadoop fs -mkdir proj\_hdfs

$hadoop fs -copyFromLocal ‘proj\_lfs/moviedata.txt’ ‘/proj\_hdfs’

1. **Switching to HDFS grunt mode of pig shell:**

$pig

1. **Loading the dataset from HDFS to pig shell:**

grunt>A = load '/MOVIES\_DATASET' using PigStorage(',') as (mid:int, mname:chararray, year:int, rating:float, dur:int);

1. **Queries:**

**A. Find the number of movies released between 1950 and 1960.**

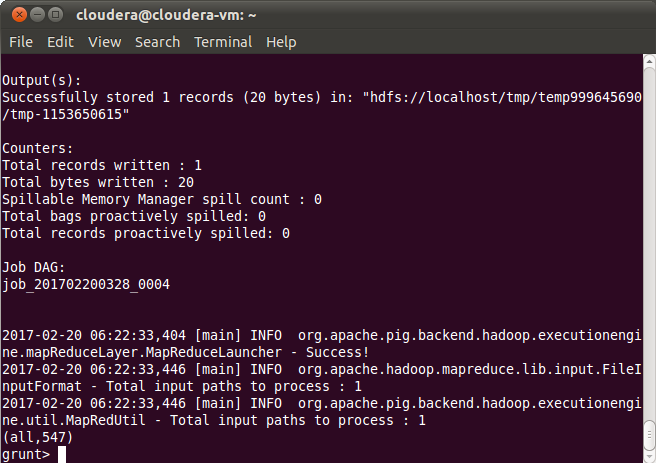
grunt>filterdata films\_year = filter A by $2 >= 1950 AND $2 <= 1960;

grunt>year\_group = group films\_year ALL;

grunt>result1 = foreach year\_group generate group,COUNT(films\_year);

grunt>dump result1;

**Output:**

****

**B. Find the number of movies having rating more than 4.**

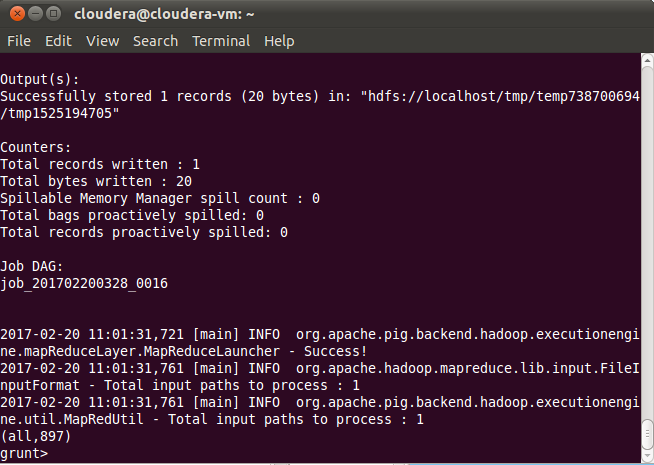
grunt> films\_rating = filter A by $3 > 4.0;

grunt> rating\_group = group films\_rating ALL;

grunt> result2 = foreach rating\_group generate group,COUNT(films\_rating);

grunt>dump result2;

**Output:**

****

**C. Find the movies whose ratings are between 3 and 4.**

grunt> films\_rating = filter A by $3 >= 3 AND $3 <= 4;

grunt> result3 = foreach films\_rating generate $0,$1;

grunt>dump result3;

**Output:**

****

**D. Find the number of movies with duration more than 2 hours (7200 second).**

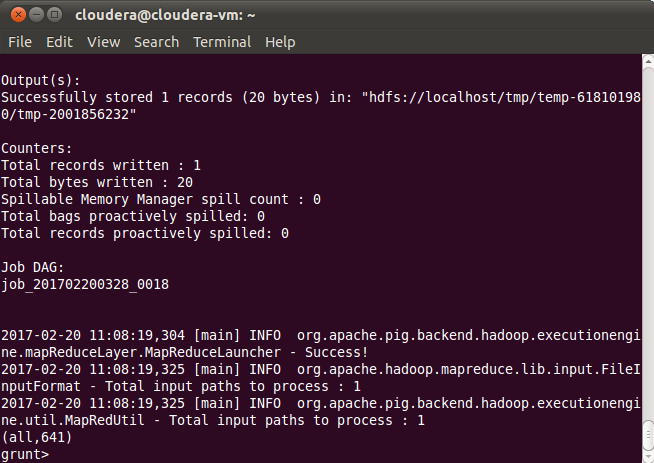
grunt> films\_duration = filter A by $4 > 7200;

grunt> duration\_group = group films\_duration ALL;

grunt> result4 = foreach duration\_group generate group,COUNT(films\_duration);

grunt>dump result4;

**Output:**

****

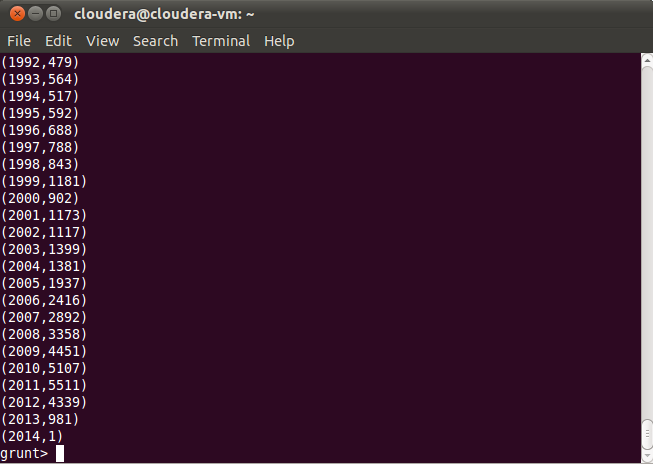
**E. Find the list of years and number of movies released each year.**

grunt> grouped\_years = group A by year;

grunt> result5 = foreach grouped\_years generate group, COUNT(A);

grunt>dump result5;

**Output:**

****

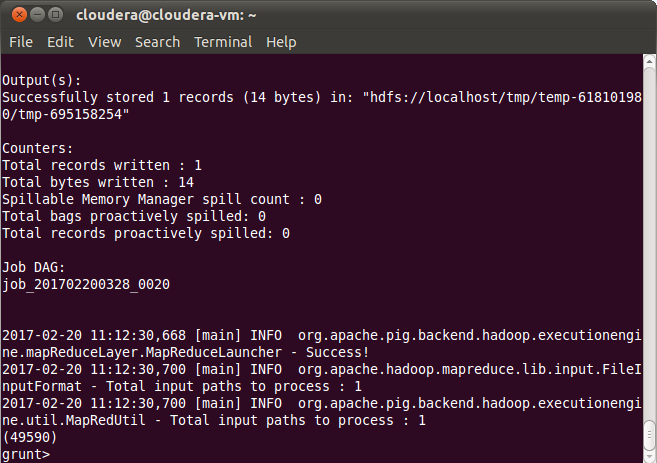
**F. Find the total number of movies in the dataset.**

grunt> group\_all = group A ALL;

grunt> result6 = foreach group\_all generate COUNT(A);

grunt>dump result6;

**Output:**

****

**7.2 YOUTUBE USING HIVE (Hive Query Language)**

$sudo hive

1. **Creating database:**

hive>create database youtube;

1. **Crating table to store the dataset:**

hive>use youtube;

hive> create table if not exists youtube(vid string,uploader string,upload\_interval int,category string,length int,views int,rating float,rating\_no int,comments int,videoids string)

row format delimited

fields terminated by '\t';

1. **Loading the dataset to the table from LFS:**

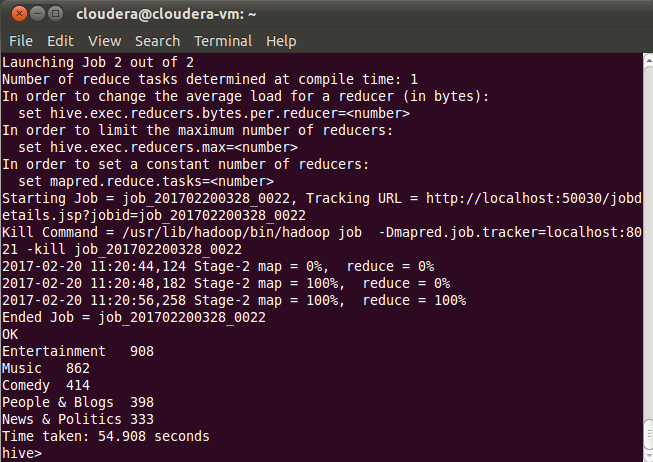
hive> load data local inpath 'youtubedata.txt' overwrite into table youtube;

1. **Queries:**

**A. Find out the top 5 categories with maximum number of videos uploaded.**

hive> select category,count(category) as ccount from youtube group by category order by ccount desc limit 5;

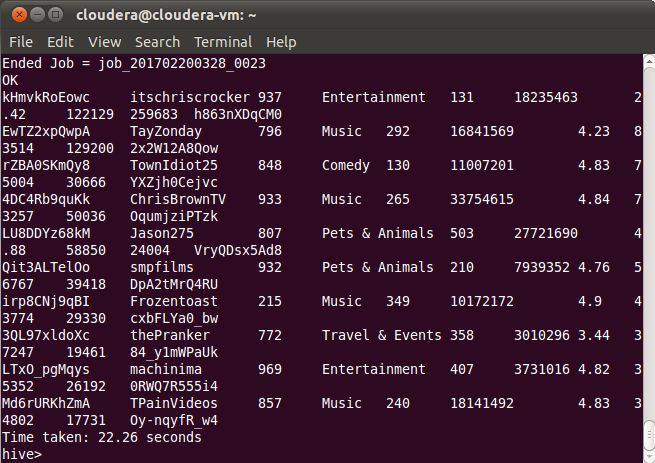
**Output:**

****

**B. Find out the top 10 rated videos.**

hive> select \* from youtube order by rating\_no desc limit 10;

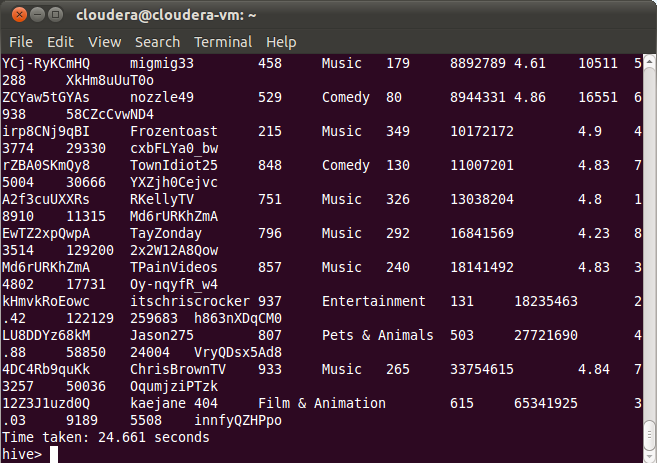
**Output:**



**C. Find out the most viewed videos.**

hive> select \* from youtube order by views;

**Output:**



1. **CONCLUSION**

Our project was an introduction and to Big Data and Hadoop along with a hands-on implementation of our knowledge to analyze the datasets of Movie Data and YouTube using Pig data flow language and an important tool for query processing, i.e., HIVE as well as using MapReduce.

We, with the help of our mentor, were successfully able to accomplish the project.

1. **REFERENCES**
2. Hadoop: The Definitive Guide, Second Edition by Tom White
3. Programming Hive by Edward Capriolo, Dean Wampler, and Jason Rutherglen
4. developer.yahoo.com/hadoop/tutorial
5. <http://hadoop.apache.org/>
6. <http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html>
7. <http://hadooptutorial.wikispaces.com/MapReduce>