

Index

Chapter 01. Basic Terminologies and Concepts >> Page Number 2

Chapter 02. Introduction to Big Data >> Page Number 18

Chapter 03. Setting up the Environment >> Page Number 35

Chapter 04. Installing and Configuring Environment Components >> Page Number 44

Chapter 05. Installation of Hadoop Generation 1 >> Page Number 57

Chapter 06. Introduction to Hadoop Architecture >> Page Number 75

Chapter 07. Hadoop Job Process >> Page Number 94

Chapter 08. Hadoop Eco System Tools >> Page Number 129

Chapter 09. Apache Sqoop >> Page Number 135

Chapter 10. Apache Hive >> Page Number 171

Chapter 11. Apache Pig >> Page Number 209

Chapter 12. Apache Flume >> Page Number 231

Chapter 13. Apache Oozie >> Page Number 236

Chapter 14. Hadoop Generation 2 >> Page Number 239

Chapter 15. Introduction to NoSQL Databases >> Page Number 252

Chapter 16. Apache Hbase >> Page Number 259

Chapter 17. Apache Spark >> Page Number 293

Chapter 18. Hadoop Administration >> Page Number 305

Chapter 19. Labs Section >> Page Number 329

Chapter 20. Cheat Sheets >> Page Number 435

Chapter 21. Hadoop Test >> Page Number 504

Chapter 22. Hadoop Interview Questions >> Page Number 522

01. Basic Terminologies and Concepts

Server

A server is a machine that has RAM, Hard Disk, Processor and other peripherals. Based on what on the use case of the server it can be classified as either an Application Server or a Web Server. In both type of Servers, **ideally 30% of the total storage is reserved for the Operating System** and the **remaining 70% can be used as storage by the installed applications.**

For example, if I am installing the game [Need For Speed](#) in my Windows machine then a folder will be created in C://ProgramFiles folder (30% reserved for OS) where the application will store all its programs in terms of .bat and .dll files and all its exe files. Another folder will be created in the location (remaining 70%) I may specify where my profile information is stored. If I tend to delete this folder, then the game will still be there only my profile will be reset and I will have to create it again. But if I delete the folder from Program Files then the entire game is gone.

If only one application is installed in a particular server it is called as a [Dedicated Server](#).

If its an Application Server, then apart from the OS we may require additional software or framework like [Java](#) or [.Net](#) and if its a Web Server then we may require additional software like [WAMP](#) or [LAMP](#).

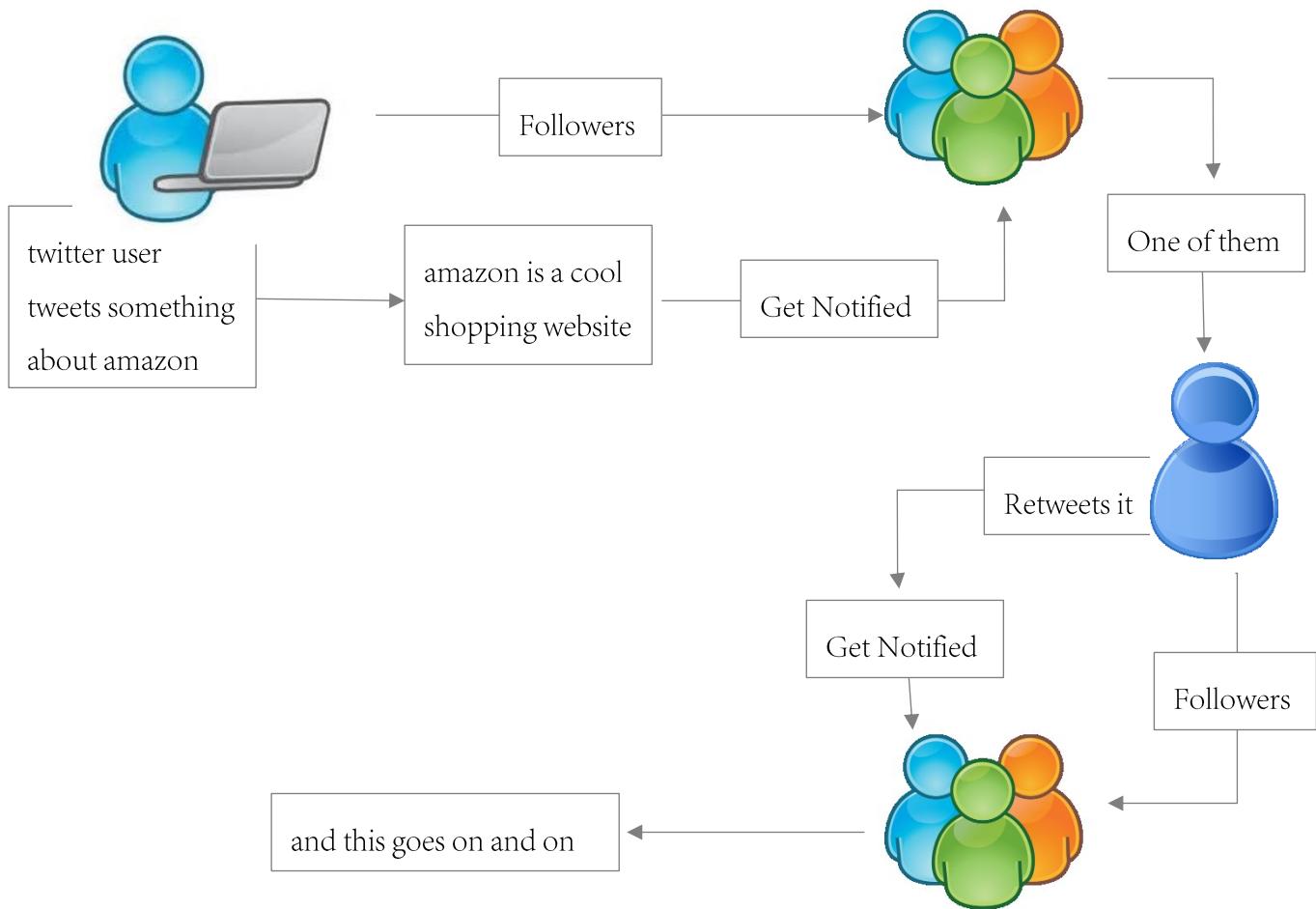
Web Server

A Web Server is a machine that provides space where instructions or programming code for [User Interface](#) and [User Interaction](#) on a Web Site can be stored. The storage area of this server is what is accessed when we enter the url of a website from our Web Browser which acts like a client to this Web Server.

Units of Measuring Data

Unit	Value
bit (b)	0 or 1
byte (B)	8 bits
kilobyte (KB)	1024 bytes
megabyte (MB)	1024 KB
gigabyte (GB)	1024 MB
terabyte (TB)	1024 GB
petabyte (PB)	1024 TB
hexabyte (HB)	1024 PB
zettabyte (ZB)	1024 HB
yottabyte (YB)	1024 ZB
brontobyte (BB)	1024 YB

Twitter Chain



Evolution of File Storage Systems

Earlier industries used to store all their data in physical files stored managed manually by the employees. Very soon there were more files in these offices than empty spaces to walk.

By this time, [Computer](#) was invented. So someone thought why not use a Computer to store this data in a digital format. It was then visualized if I talk about a bottle

then this can have certain characteristics like height, weight, capacity, color, make etc. And if I talk then I can define the same bottle as well.



but another bottle
characteristics for this



This brought a thought that every living and non living thing posses certain characteristics that may help us to group them under one roof. This is how we came across an [Entity](#). So we arranged all properties on an entity in line and added values that different entities under the same group possessed for these attributes. This one roof was then called as a [table](#).

	T Department ID	T Department Name
1	0569	Accounts
2	0341	Sales
3	8221	Marketing
4	2003	HR
5	7641	Development
6		
*		

the entity

the attributes

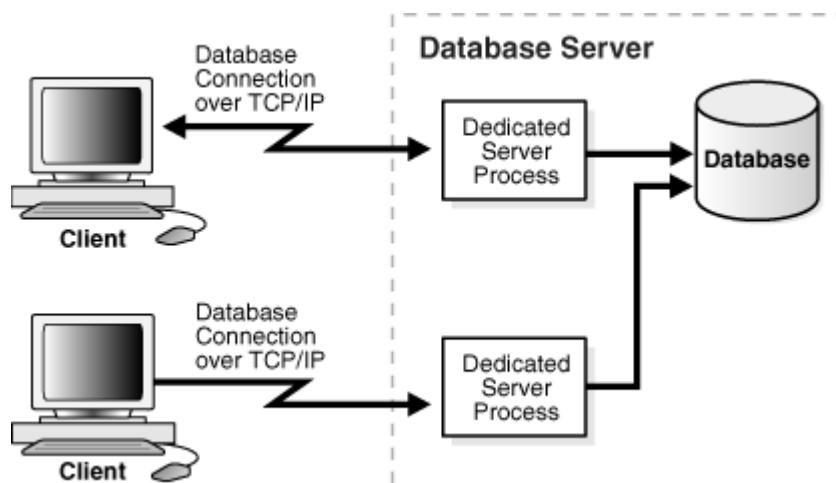
the entries

Then we brought all tables related to same organization under one roof and that became our base for all data and so became popular as a **database**. This new software had two components viz. the programs and instructions that visualized all data digitally and a storage component where this data was actually stored. Now that the software was all ready, it was time to install it somewhere.

So we brought a machine in which we installed the OS and reserved 30% of the total storage of the machine for the OS. Then we installed this new software in this machine and as a result, all programs of this software got stored in the 30% area and the remaining 70% was available for data storage which was now dedicated to our new software.

Then we realized that we have defined a structure for storing our data but there needs to be a *LANGUAGE* which can help us *QUERY* this *STRUCTURED* data. This is how we got the **SQL** (Structured Query Language).

With this things became very easy to manage and work around with. But soon we realized that this machine provided access to only one user at a time and this became a problem. So we separated this machine from direct access and built a tool which will be installed in the users computer and will be able to interact with this machine. In this way the machine where our database was installed started to server the tools installed in the users machine. This is how we got our **Database Server** and **Database Clients**



By now it became very easy for us to manage all our data which was too difficult previously due to the haphazard manner in which it was stored.

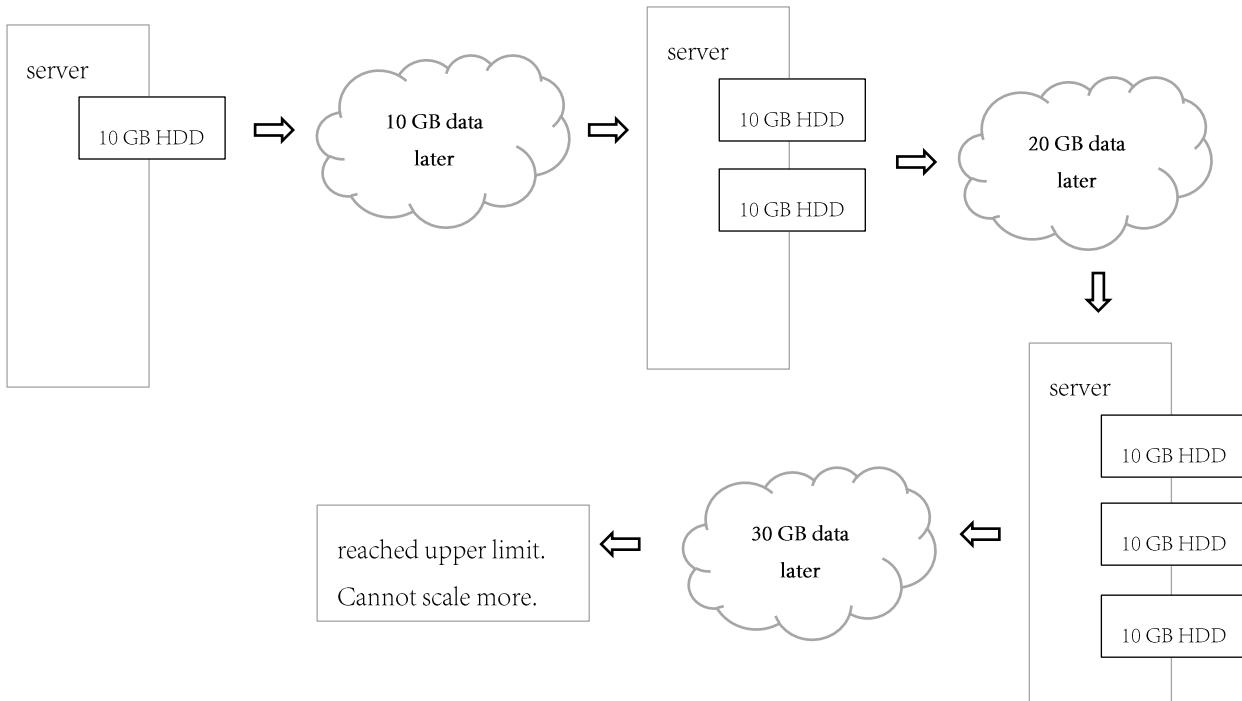
Sometime back my sister had created a week long chaos for buying a green color dupatta to match the new dress she purchased and finally brought that dupatta. After few days Mom visited her and cleaned her wardrobe which was a mess from a long time. Surprising reveals were that she had 3 more dupattas of green color that she was not aware of because of the mess in her wardrobe. But when it was arranged properly then she was able to find stuff which earlier she was not able to.

Same thing happened with the industries as well. With their data arranged in a structured way now, they were able to find insights of their customer which they were not familiar with earlier due to the bad organization of the data. With this they thought of exploitation and built more applications that can leverage this information to earn more profits.

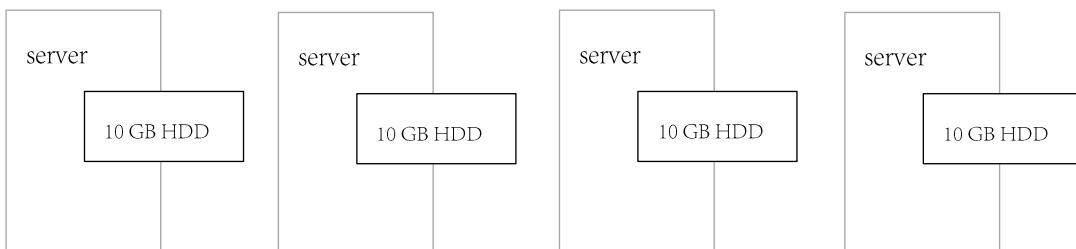
So more data started flowing into the system and sooner we realized that we had ran out of capacity. But we did not panic. We simple added more additional hard disk to our server and we were good again. This was called as [Vertical Scaling](#).

With this additional ease of storing data, more and more applications were built and the rate at which data was generated started increasing.

Very soon we realized that our database server now cannot support more additional hard disks and we are again running out of capacity. This is when we realized that vertical scaling has a [drawback that it has a upper limit](#).



This is when we thought why not use one more server rather than adding more hard disk to the same server. With this our problems were solved and we started calling it as **horizontal scaling**.

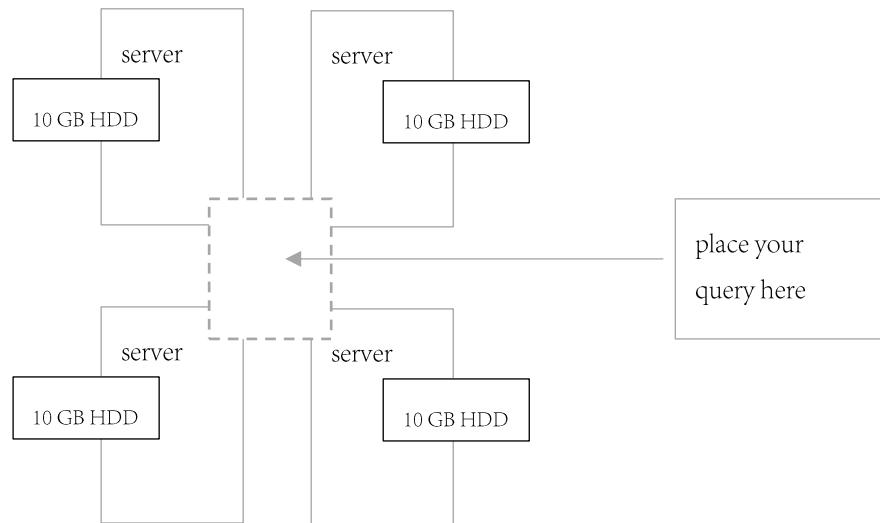


Now we could have as many servers we want and their was nor problem with the capacity. It was like I had 10 mobile phones for storing all image that I clicked because a single mobile phone was not having sufficient capacity. So every thing was good now and there was no limit. I could buy as many phones I wanted. But then my Dad asked me for a image of him that I had clicked couple of months back. Now I had to serially check all phones one by one and then I found it in one of the phones.

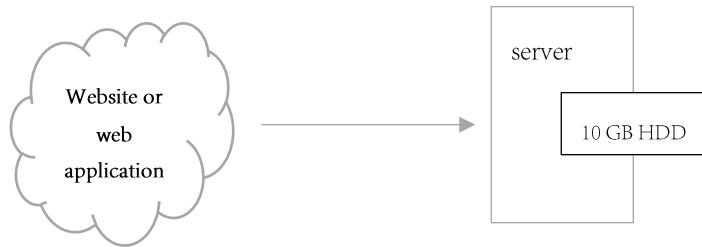
This is where we realized that if number of servers will increase in Horizontal Scaling, it may arise new problems while fetching the data.

So we thought why not build a system in such a way that instead of searching all phone manually I will ask the search to the pool of my phones like a broadcast and who ever has it will return it. In this way I am not even require to worry how many phones are there as it will always be a single point of contact for me on the frontend.

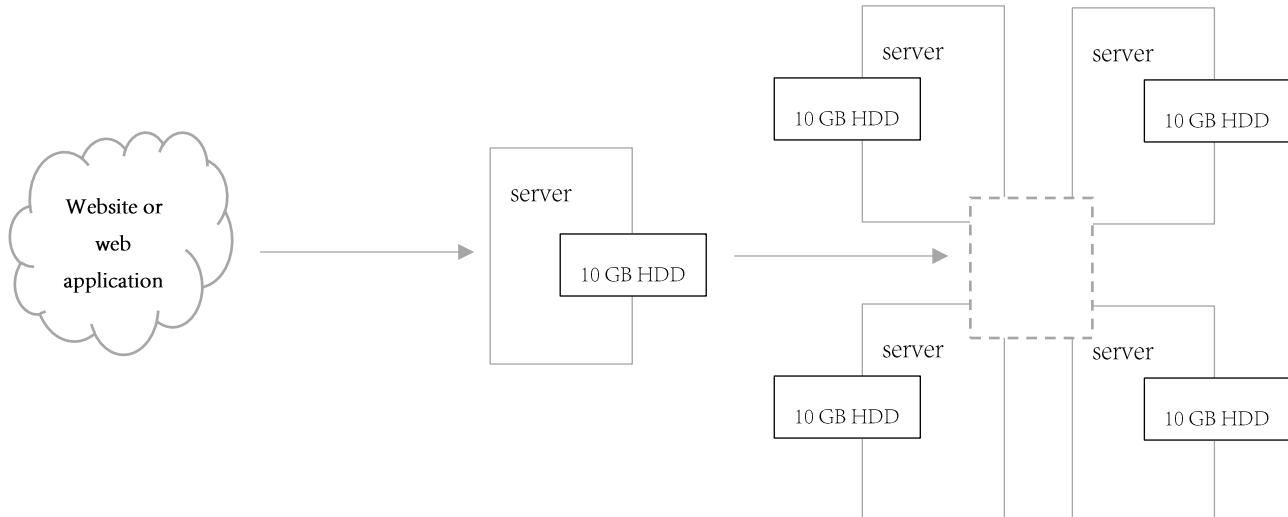
This is called as a **Distributed Environment** where instead of querying to individual servers we query the pool of servers as if we have a single point of contact.



So now thing seemed to be stable and we thought we have a solution. So we approached companies that were using the following architecture and asked them to replace their database models with this new model. For this they had to scrap all investment they did earlier, do some new investment, train their employees on this new model, revamp their program with this new model and then get their normal routine set again. But the obvious answer that we got from the companies was a big No.



With this we thought, instead of replacing the existing model why not introduce this new model as an aid to the existing one in such a way that all excess data can be moved from the exiting model to the new model. This is what was approved by the companies as well.



With this we now had two types of data. One which was stored in the existing model (**database server**) and one which was stored in the new model (**data warehouse**).

The data (**transactional data**) which was stored in the database server was a result of user interactions and was regularly queried and processed. Whereas the data (**historical data**) stored in the data warehouse was the access data collected over a period of time and was used once in a blue moon.

By this time we realized that some more information about our customers is available on Social Media Servers. So we somehow managed to pull this data ([Data Extraction](#)) and store in our data warehouses.

When we got this data, it came as a mixture of both relevant and irrelevant data as far as our business use case was concerned. So we thought of separating the gold from the dirt and keep only relevant information in our warehouse ([Data Mining](#)).

Even after this was done, it was realized that the structure of the data was not exactly how we wanted. So we had to further transform it into a way that our case required ([Data Transformation](#)).

Then we went ahead and applied some aggregation function on this data ([Data Processing](#)) and discovered that now we were able to make predictions about the behavior of our customers and use these predictions to influence the thought process of a customer ([like a recommendation system on an e-commerce website](#)).

This entire process turned out to be a source of additional profit and this process is called as [Data Analysis](#).

So we realized that our warehouse can be used for more beneficial activities rather than simply storage of data.

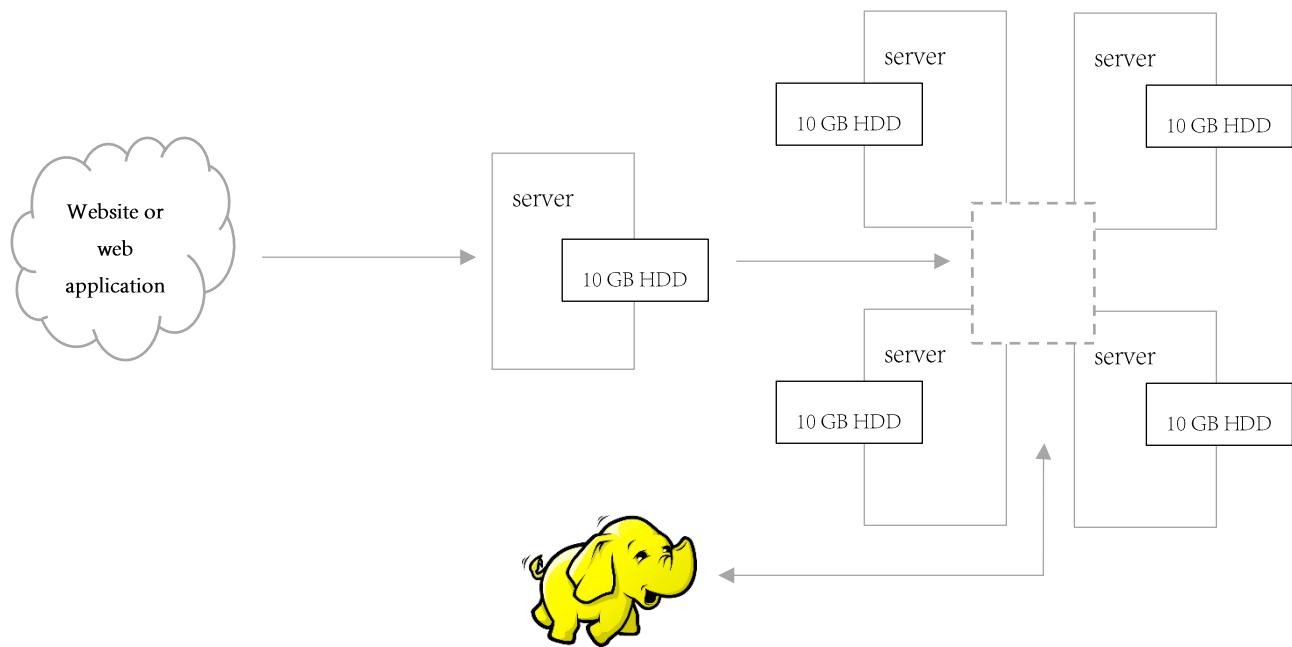
But by now we had made a lot of changes in the warehouse :

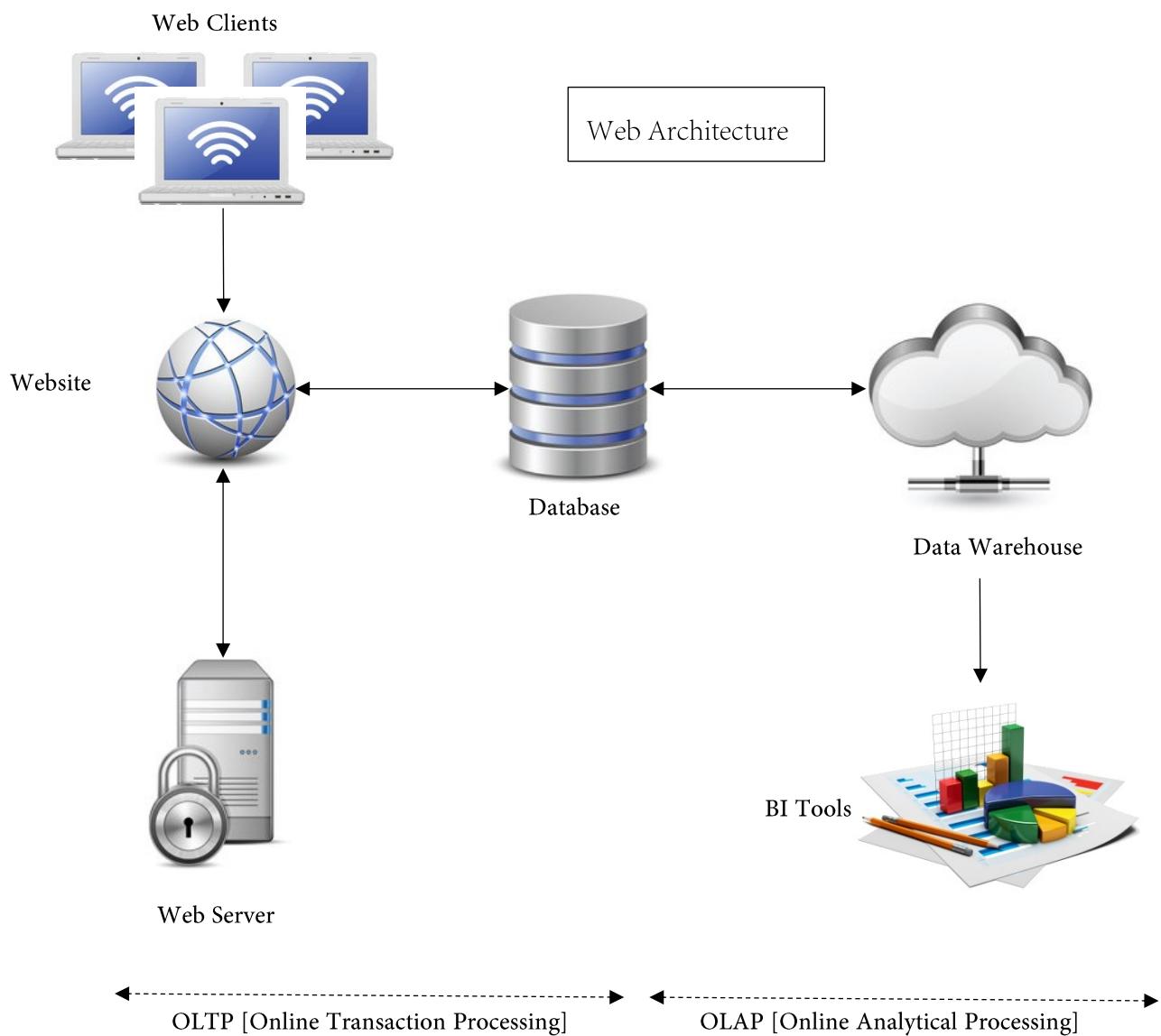
to support different types of structures, we eliminated the concept of structures and introduced [Flat File Systems](#) for data storage. Its like if aliens start coming to offices then the only option the offices have is to remove chairs and make every one sit on ground so that the shape and size of the aliens won't matter.

to support these complex operations we introduced [programming](#) rather than querying.

With all this in place, now things became more easier for industries and they started exploiting this situation to the next level resulting in a exponential increase in the rate of data generation. This rate increased pressure on the warehouse and soon they started degrading in performance.

This is where a new kind of distributed system was launched as an aid to the data warehouses and this is called as **Hadoop**





Understanding Basic Linux Commands and Terminologies

The Root Directory (\)

Contains the following concerned directories:

1. Home Directory:

A **home directory**, also called a login directory, is the directory on Unix-like operating system that serves as the repository for a user's personal file, directories and programs.

2. Lib Directory

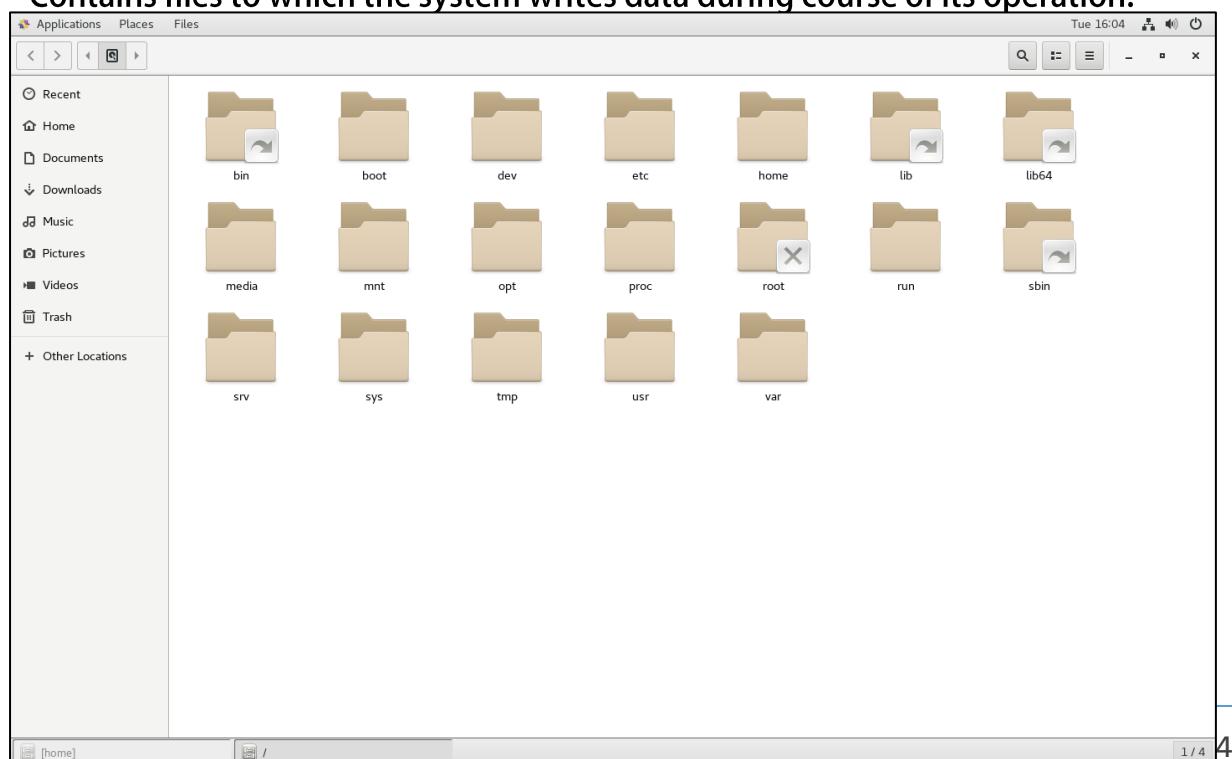
A **lib directory** contains dynamic libraries and support static files for the executables at /usr/bin and /usr/sbin

3. Usr Directory

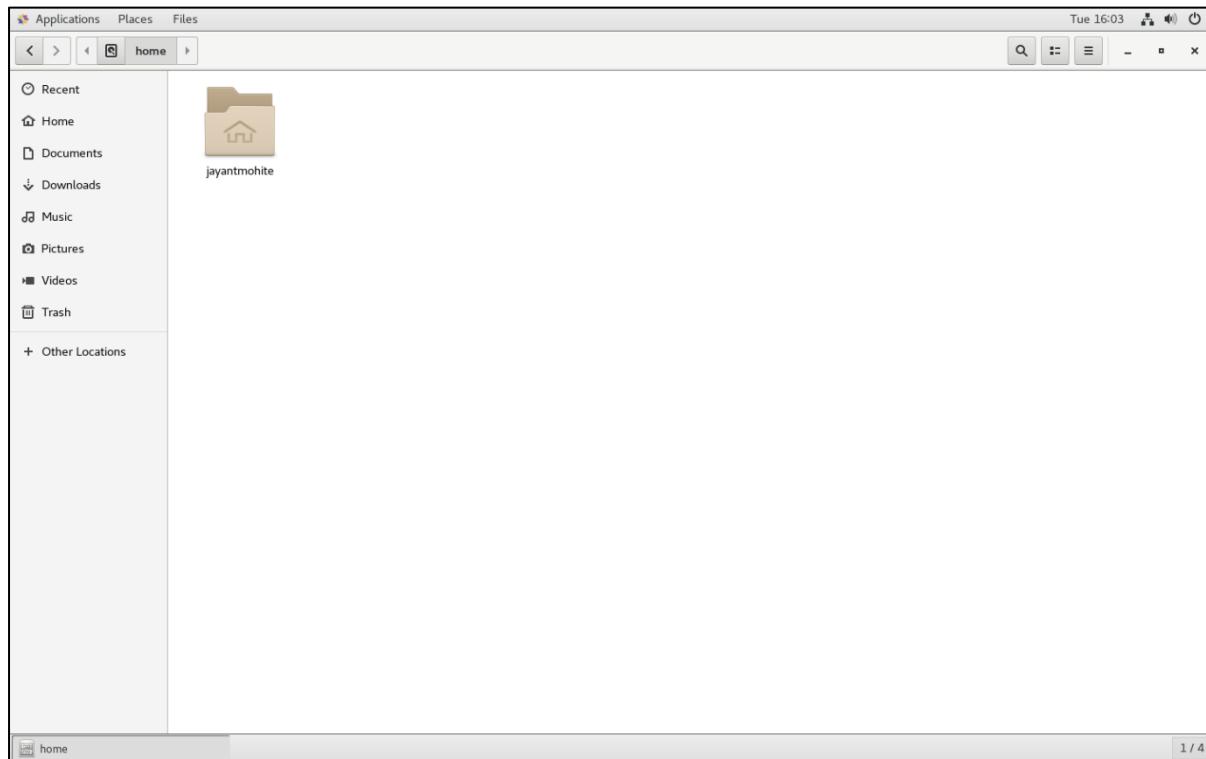
A **usr directory** contains the user binaries, their documentations, libraries, header files, etc.

4. Var Directory

Contains files to which the system writes data during course of its operation.

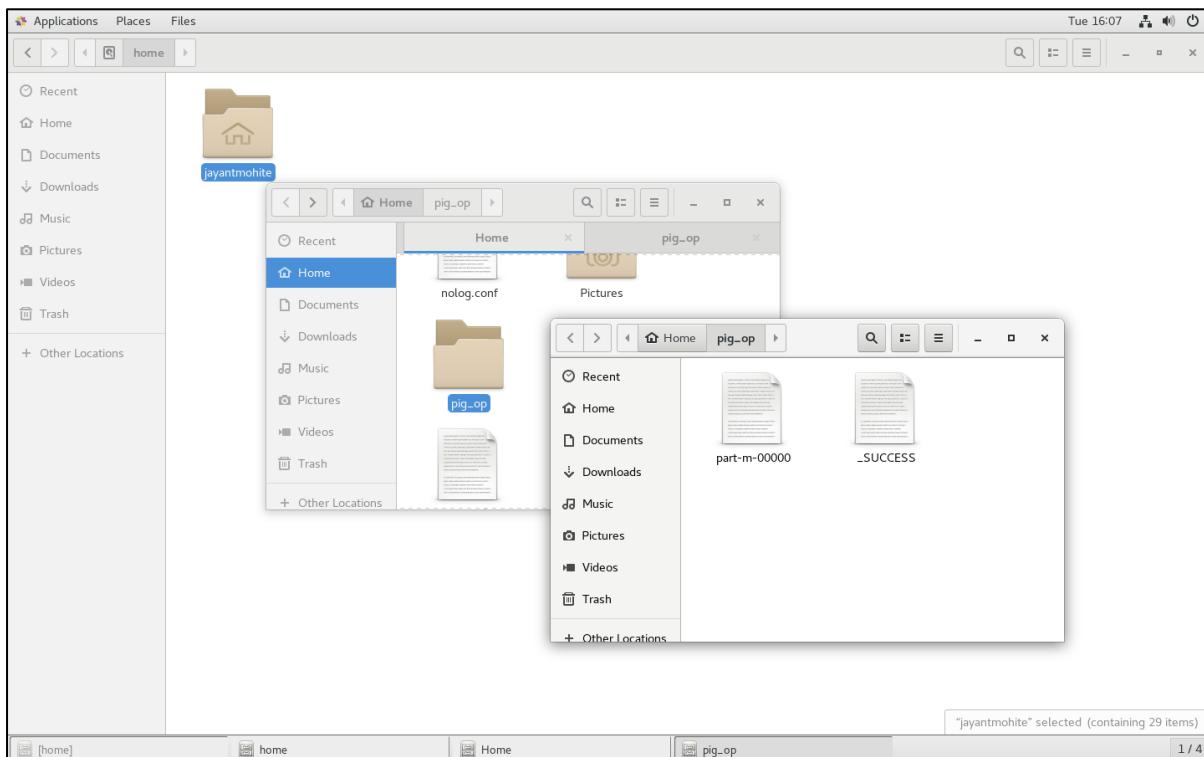


The Linux Home Directory



Let us now have a look at the ls command

The GUI way



The CLI way

```

Applications Places Terminal
jayantmohite@localhost:~$ ls /home
[jayantmohite@localhost ~]$ ls /home/jayantmohite/
cloudera-cdh-5-0.x86_64.rpm  java-json.jar  myemp_us.txt  nolog.conf  Pictures  test1.txt
derby.log  jayant_file.txt  mysql-community-release-el7-5.noarch.rpm  Pictures  test2.txt
Desktop  jayant_table.java  mysql-connector-java-5.1.45  pig_op  test_file.txt
Documents  Music  mysql-connector-java-5.1.45-bin.jar  Public  test.txt
Downloads  myemp_india.txt  mysql-connector-java-5.1.45.tar.gz  QueryResult.java  Videos
employees.java  myemp.txt  mysqlsampledatabase.sql  Templates
[jayantmohite@localhost ~]$ ls /home/jayantmohite/pig_op/
part-m-00000  _SUCCESS
[jayantmohite@localhost ~]$ ls -al /home/jayantmohite/pig_op/
total 16
drwxrwxr-x. 2 jayantmohite jayantmohite 84 Mar 3 01:36 .
drwxrwxr-x. 17 jayantmohite jayantmohite 4096 Mar 6 16:02 ..
-rw-r--r--. 1 jayantmohite jayantmohite 39 Mar 3 01:36 part-m-00000
-rw-r--r--. 1 jayantmohite jayantmohite 12 Mar 3 01:36 .part-m-00000.crc
-rw-r--r--. 1 jayantmohite jayantmohite 0 Mar 3 01:36 _SUCCESS
-rw-r--r--. 1 jayantmohite jayantmohite 8 Mar 3 01:36 ._SUCCESS.crc
[jayantmohite@localhost ~]$ 

```

Now let us have a look at the mkdir command

```

Applications Places Terminal Tue 16:10
jayantmohite@localhost:~ - x

File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ ls
cloudera-cdh-5.0.x86_64.rpm  java-json.jar  myemp_us.txt  nolog.conf  test1.txt
derby.log                      jayant file.txt  mysql-community-release-el7-5.noarch.rpm  Pictures  test2.txt
Desktop                         jayant_table.java  mysql-connector-java-5.1.45  pig_op      test_file.txt
Documents                        Music          mysql-connector-java-5.1.45-bin.jar  Public     test_.txt
Downloads                        myemp_india.txt  mysql-connector-java-5.1.45.tar.gz  QueryResult.java  Videos
employees.java                  myemp.txt    mysqlsampledatabase.sql  Templates
[jayantmohite@localhost ~]$ mkdir EXAMPLE_DIR
[jayantmohite@localhost ~]$ ls
cloudera-cdh-5.0.x86_64.rpm  EXAMPLE_DIR  myemp.txt  mysqlsampledatabase.sql  Templates
derby.log                      java-json.jar  myemp_us.txt  nolog.conf  test1.txt
Desktop                         jayant file.txt  mysql-community-release-el7-5.noarch.rpm  Pictures  test2.txt
Documents                        Music          jayant_table.java  mysql-connector-java-5.1.45  pig_op      test_file.txt
Downloads                        myemp_india.txt  mysql-connector-java-5.1.45-bin.jar  Public     test_.txt
employees.java                  myemp.txt    mysql-connector-java-5.1.45.tar.gz  QueryResult.java  Videos
[jayantmohite@localhost ~]$ mkdir -p EXAMPLE_DIR/SUB_DIRECTORY
[jayantmohite@localhost ~]$ ls EXAMPLE_DIR
SUB_DIRECTORY
[jayantmohite@localhost ~]$ 

```

[home] [home] jayantmohite@localhost:~ 1 / 4

Now that you have a good amount of understanding of all basic terminologies (atleast those you will need to learn Hadoop), lets get ahead and jump in to more details of Big Data Hadoop.

02. Introduction to Big Data

To start off with this lets think of something which has been the main focus of the our service industry from a long time. Its been proved from a lot of results that identifying the **potential customers** has always been a pain for the industry for which **Analyst** have to wash their brains off and get with ideas that can help the company to achieve its goal. In this customer centric world, **digital media** and **digital advertising** plays an important role because this is how you can reach a larger mass of customers. But the point is whether or not you are reaching to customers which real matter.

For this we will make use of some data that can be collected from **Twitter** which is one kind of Social Media where both professional and social people meet, which makes it a better choice for us as compared to **Facebook** which is purely Social and **LinkedIn** which is purely professional.

The Twitter website is considered to have a Web Server where it is deployed and this server contains a file where all user interactions on Twitter are recorded. This is called as the **Twitter Logs**. It contains lot of information about every tweet including some information like the username, gender, age, location, hashtags, tweet, replies etc. It contains one more filed which is important to us at this very point by name the **retweet counter**. The higher number of retweet counter for a user indicates that more influential potential of the user. So if we are able to find out people who have similar interest to our products and then if we can sort them in the descending order of retweet counter we may be able to find our prime customers which we can target for advertisements.

Sounds like a plan. But this has some problems

Problem 1: The Twitter Logs are on the [Twitter Server](#). How do we access them?

Problem 2: The twitter log file is supposed to get appended by more than [10 Peta Bytes](#) of data every hour. This is a huge data to be handled by any traditional storage system.

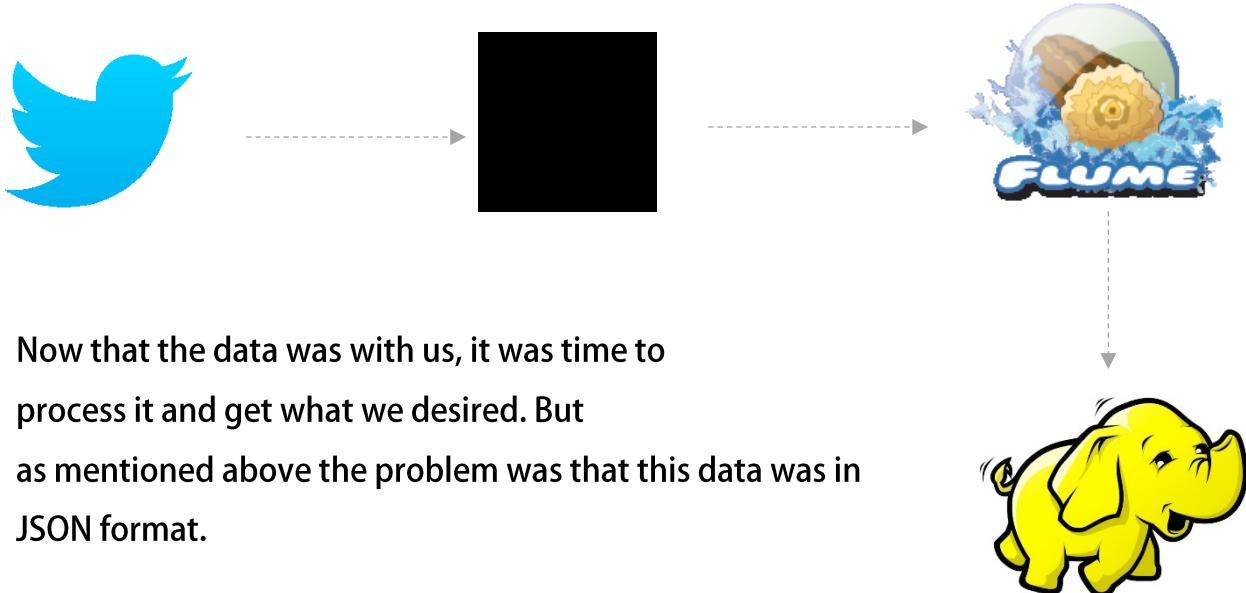
Problem 3: The format of data stored in this file is [JSON](#) which is unstructured in nature and again not the job of any traditional storage system.

To solve these problems, this is what we came up with:

An Application Interface that will connect the Twitter Logs with the outside world. This [API](#) will send events after few activities recorded into the logs.

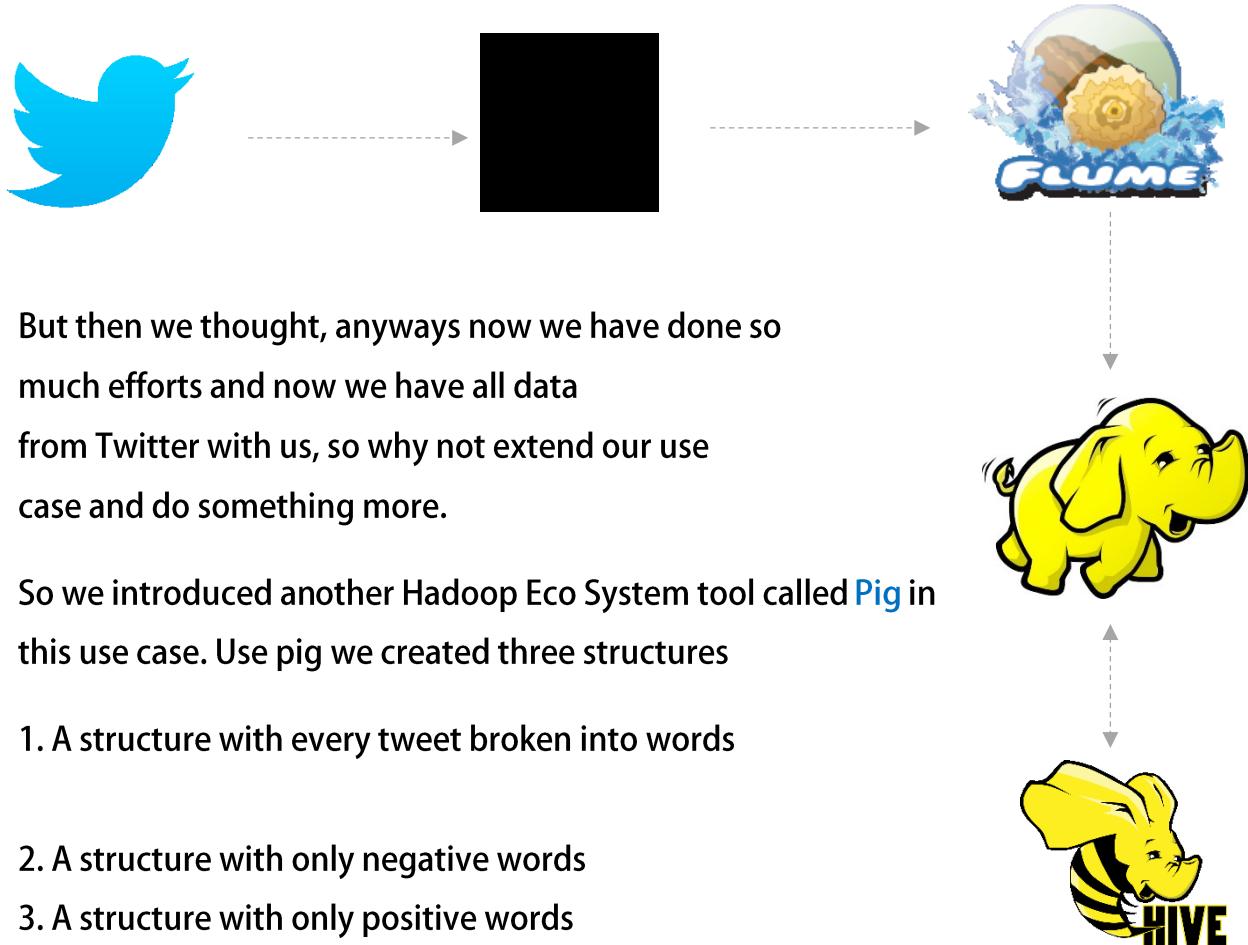
[Hadoop Framework](#) that has the capacity of storing any amount of data available in any format.

A Messaging Tool ([Flume](#)) which will collect the events produced by the API and dump the same in the Hadoop storage.



So we brought up a Hadoop Eco System tool that was capable of converting JSON data in RDBMS like structure and was also capable of executing RDBMS like queries for interacting with the data. This tool is called as [Hive](#).

With Hive in place we were able to solve our last problem as well and get things done as expected and our use case was completed.

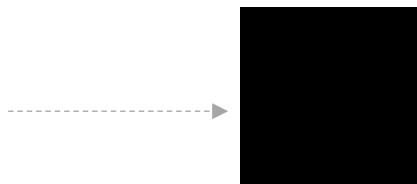


But then we thought, anyways now we have done so much efforts and now we have all data from Twitter with us, so why not extend our use case and do something more.

So we introduced another Hadoop Eco System tool called [Pig](#) in this use case. Use pig we created three structures

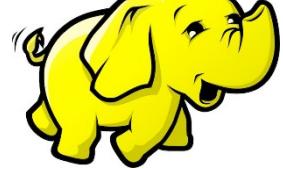
1. A structure with every tweet broken into words
2. A structure with only negative words
3. A structure with only positive words

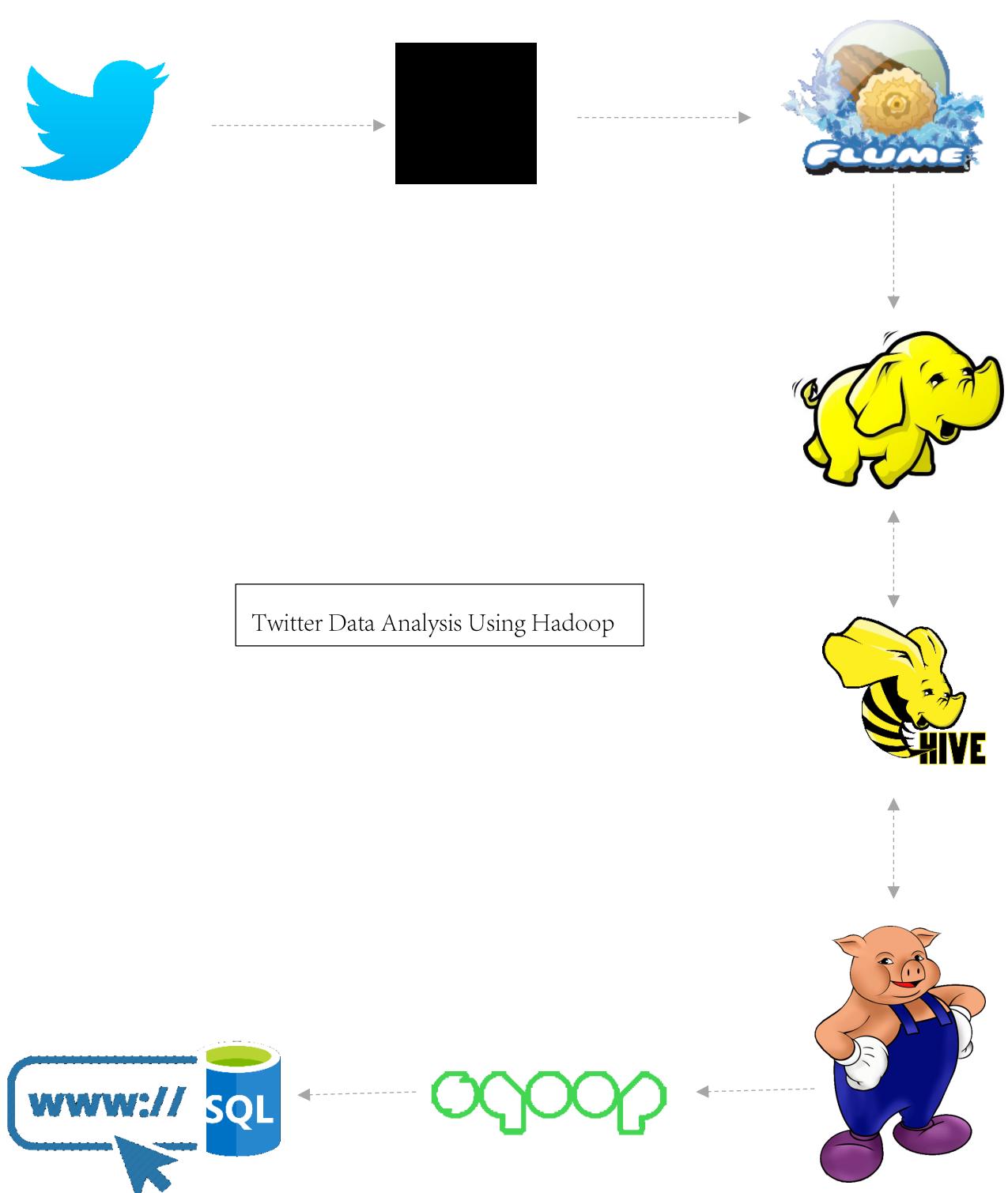
Then we processed every Tweet to find out whether maximum of the word in that Tweet belonged to the structure with negative words or to the structure with positive words. With this we were able to associate [sentiments](#) with every Tweet and we were now able to understand our customers much better.



Then we thought why not display this statistics to our users, so that even they come to know about our popularity and about what others feels about us. But now the problem was that this data was in Hadoop and our Website was connected with our RDBMS.

So we introduced another Hadoop Eco System tool that was capable of exchanging data between RDBMS and Hadoop Storage. This tool is called [Sqoop](#).

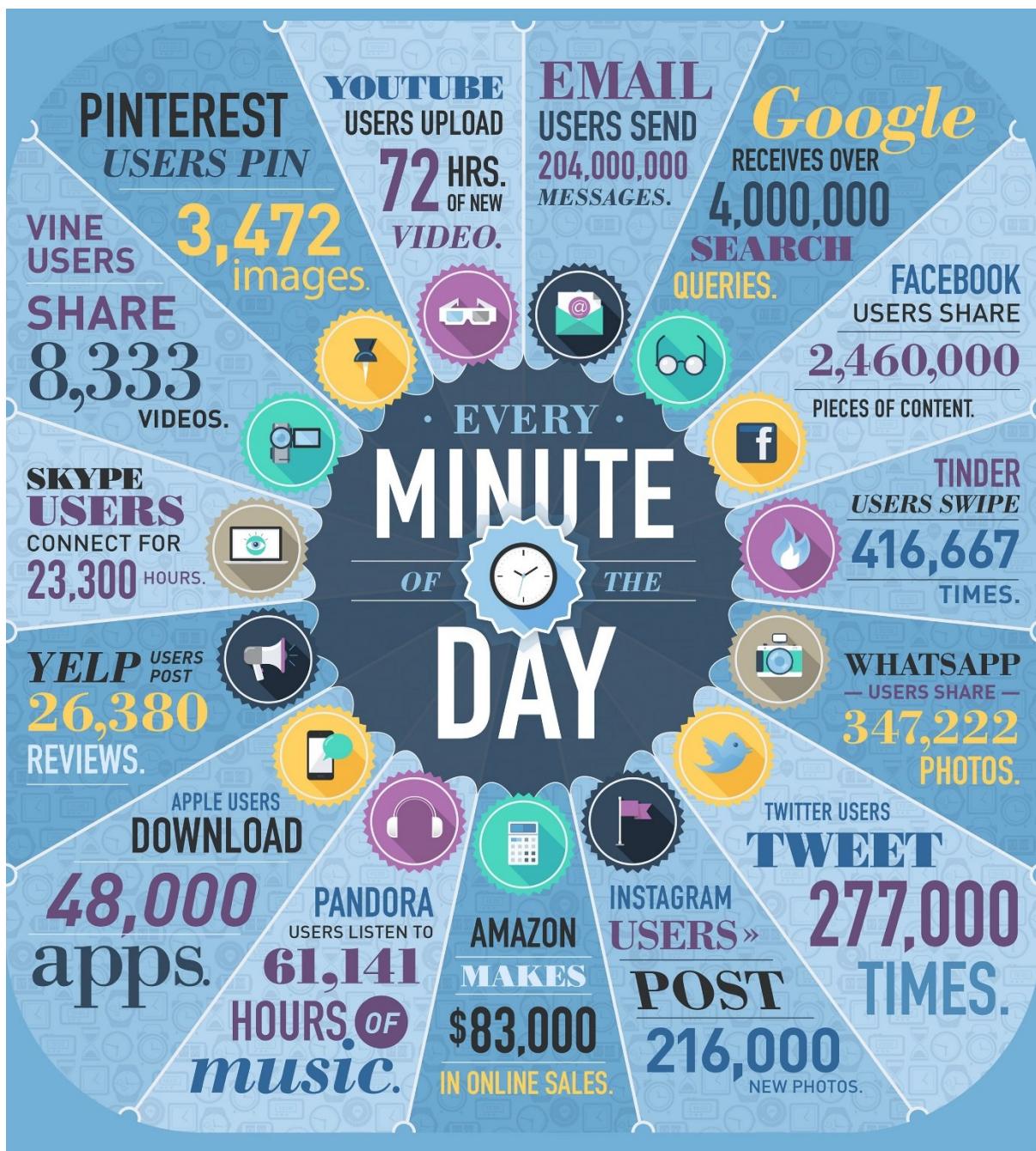




With this we complete our use case.

Now lets talk about something else,

Welcome to the **age of Internet**. Today if an Engineering Students fails in the exam, Facebook is the first one to know about it (dude who will tell this to your parents). Today if a girl' s Dog is not feeling well, Instagram is the first one to know about it (instead go to a doctor babe). Shopping is just a matter of few clicks today. Next season of "**Kaun Banega Crorepati**" is supposed to launch a new life line "**Ping a Friend**" .



In a minute millions of Tweets, Shares, Posts, Uploads, Transactions etc. are happening around us. But what is this? Ultimately this is data. So what this has done is, massively increased the amount of data that gets collected. Sites like Facebook and Amazon are supposed to handle PBs of data every 30 Minutes. This data if compared may be more than what the entire database on the planet had in the year around 2007.

This data is collected and analysed by companies to find out the true sentiments of their Customers.

Companies feel that they have now got a platform to understand the “Real Me” of their Customers. Though bitter but this is the truth. If you get a Feedback form at the end of a Seminar, how would you normally fill it? Remember right?

“Akad Bakad Bambe Bo” . And if there is a section called comments, then it is more like a one word answer - NOTHING!!!! So how can someone really get a true feedback out of this?

But what if the same thing was to be done online? If a friend of yours posts a picture of her, then you will search [Google](#) to get the best words for your comment with only intent that your comment should get more likes than her pic. The “What’ s on your mind” is always “what’ s on Google’ s mind” in your own words. Agree? Why this indiscrimination? Nobody even mentions their names in the feedback forms. So who cares what we feel because nobody will ever find out that I have filled it (But we forget that companies to whom this form belongs, they care for our answers). Same thing on FB!! Man all my friends and family is going to see my comment. So, it has to be the best. And this is why companies now believe that, if they analyse this online data in the proper way then can find a way to influence the behaviour of their customers.

How does this data look like?

The online data that we are collecting over the past few years can be of any format. It can be **Strings, Integers, Images, Videos, Audio, Banners, Graphs**, etc. It can be anything and everything that you can ever imagine. This data can be categorised as **Structured Data** (data can be stored and processed in its true format. For example: Integers and Strings) and **Unstructured Data** (data that has to be converted into some convenient format for the purpose of storage and processing. For example: images converted to pixel values)

What is Data Analysis?

The data we collect, is not always the exact stuff that we are looking for. It is always a **mixture of Gold and Dirt**. For example: A feedback on Snapdeal looks like

★★★★★
good product
Jul 13, 2015

What concerns Snapdeal

good design, faster processor , good camera , every thing is good and fastest service by snapdeal i got mobile with in 2 days, i love to by in snapdeal.

Was this review helpful? YES 5

this:

So fetching out the concerned details from the overall comment is a kind of data analysis.

Data Analysis is a big world in itself. It involves a lot of things like **Data Mining, Data Processing, Predictive Analysis, Descriptive Analysis, Data Visualization** etc.

How is Data Analysis done?

Years of data is accumulated in large storage sectors and then using powerful tools and different algorithms, this data is analysed to figure out Business specific information about it. This does not only help you to understand the customer sentiments, but also helps you in doing a lot of stuff like [understanding the trends, decision making, new launches](#) etc.

But why is years of data needed for all this? We can do this by collecting some data as well, right? For example if I am an e-commerce site selling Poison, Rat Kill, and Sleeping pills ([Dumb Ways To Die](#)), and if around 10 customer had visited me searching for Poison and finally 4 of them ended up buying Poison, 5 of them brought Sleeping pills and 1 of them brought Rat Kill, then the 11th person that comes looking for Poison can be easily recommended the other two items, right? No!! If I do so, that person may charge me of encouraging suicidal tendencies and can sue me. Because he might be there to buy poison for some Lab experiments. But giving recommendation like sleeping pills will indicate that we want him to commit suicide. But what if I have been observing this trend from last 5 years that all those who came looking for poison on my site, have either committed a suicide or a murder the next day. In this case I can surely push up my recommendations. Right? So it's simple [more the data you have, better can be your predictions](#).

Where is this data stored?

So we mean that we are analysing somewhere around 100s of 1000s of PBs of data every day. So store this much amount of data, surely we need strong and powerful servers that can scale endlessly. So we make use of frameworks like [Hadoop](#) for handling this data.

What happened to the traditional systems?

Our mothers are the smartest beings available on the planet. Do know why? Because they have the ability to sit on the last day of every month and plan the requirements for the coming month. Then the mother goes to the super market and gets all the stuff she wants and then she never has to buy anything extra in that month. Possibly the statistics of the previous months consumables or her experience might be giving her this ability to check on the things.

But what if one day suddenly around 15 people come to the house and decide to stay there forever. Now, these guys are not going to pay anything, neither are they going to help nor are they going to adjust on anything. She might have had still managed this number but all these people have different food habits and likes. The girls like fast food, boys like Chinese, males prefer south Indian items whereas the females prefer north Indian items. Apart from this the boys have a very weird habit. If they have a good dream the other night then the next day they will prefer eating like 6 times and if it was a nightmare then they may end up eating only once. So now the scenario stands in front of the mother that she has to cook for 15 people, the preparations differ and moreover she is not sure that how many times she might have to cook the food.

At this point of time, things go beyond her capacity and the most eligible person on the planet decides to quit and kicks off all those guests away.

Same thing happened with the data storage units that we were using traditionally and finally when things went beyond their capacity they proved to be inefficient.

Why did this happen?

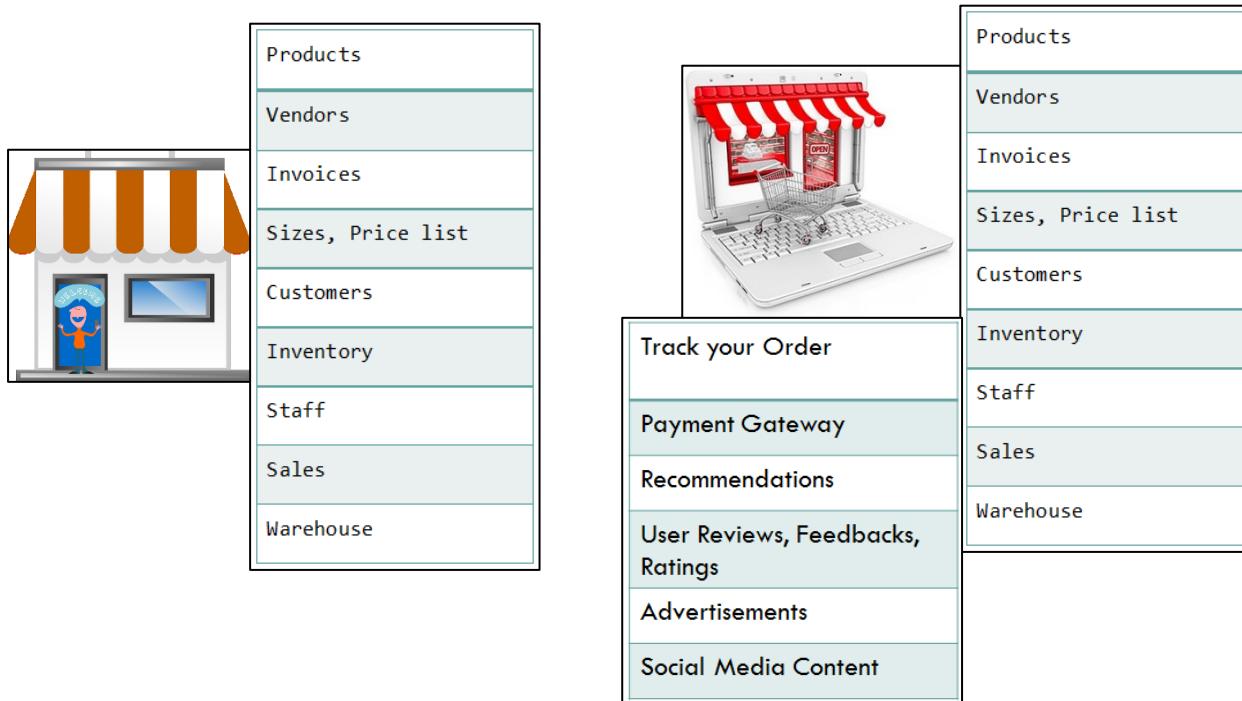
Case 1

How many types of shops are there?

Online Store (popular now)

Offline shops (popular earlier)

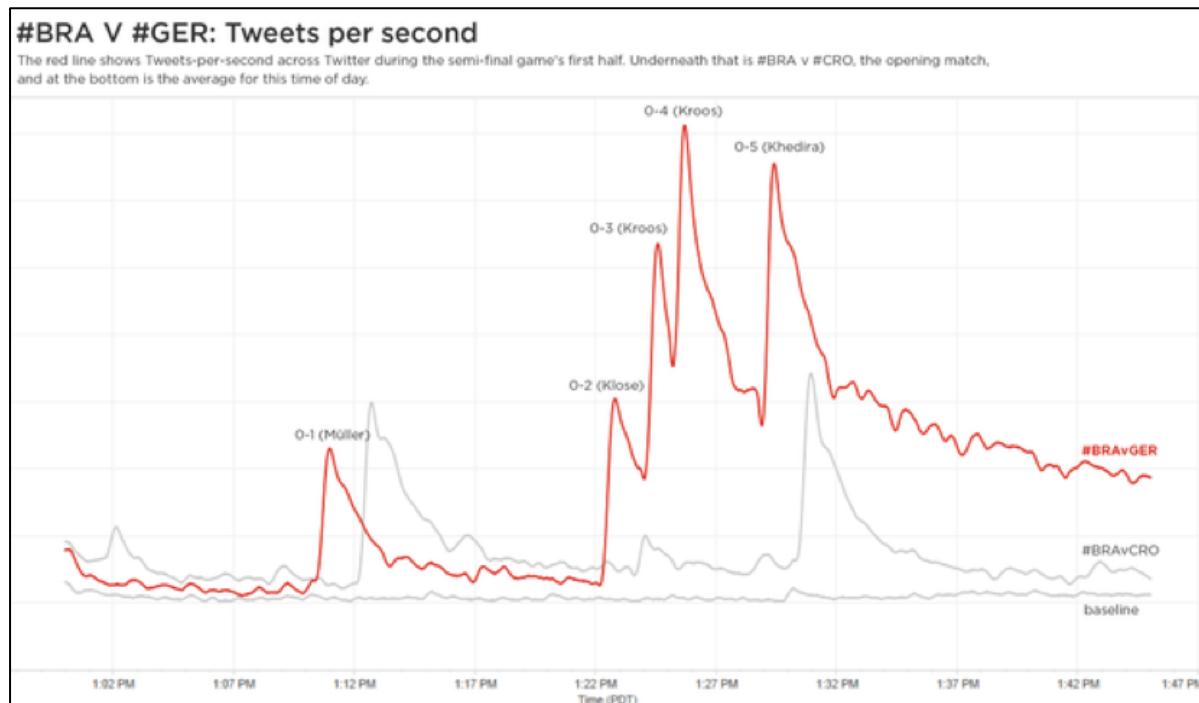
Right? If we try to briefly do a comparison between them on the basis of data that they store, this is what we may get



The difference is pretty much clear. Not only the amount of data stored is multiplied but also if you compare the type of data stored then it has heavily migrated from structured data to unstructured data. Not only this but also the sources from where the data is collected has increased. Earlier it was only your databases from where you collected the information, but it can be somebody else's database as well (payment gateway, track your order, social media pages etc.). Was this situation imaginable before the advent of internet?

Case 2

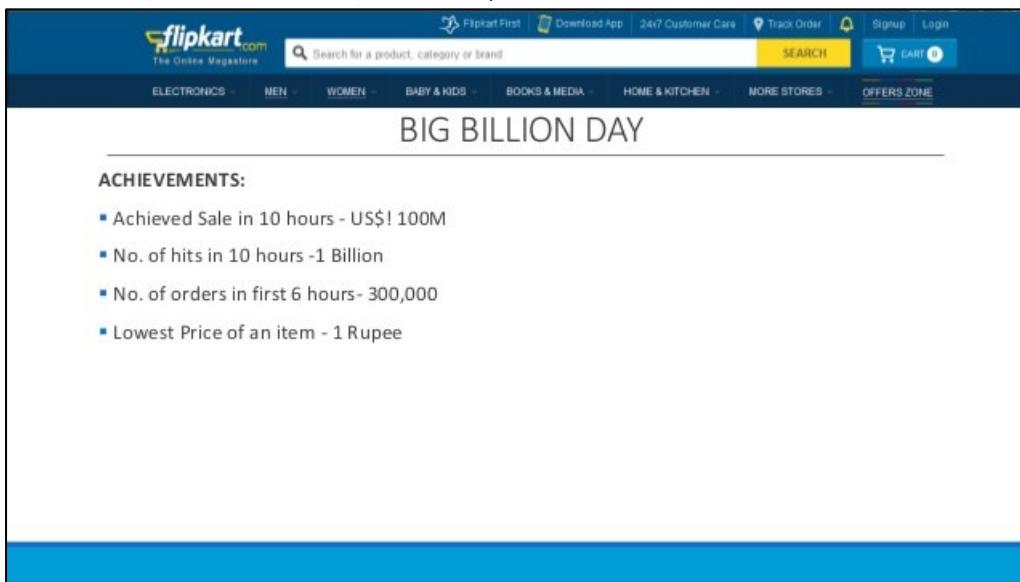
What will happen if one day, 500 customers visit a shop where the average daily number of customers is 50? The shopkeeper will panic and go crazy right? He won't be able to manage this. But online sites can handle this. Here's a small example



Was this situation imaginable before the advent of internet?

Case 3

One place, one shop, one day



Was this situation imaginable before the advent of internet?

All our 3 cases clearly suggest that all that happened was never imaginable before the advent of Internet. So how can we expect the storage entities built in an era when no one had the sight of internet, to be able to handle this situation?

So this is a problem right? What is it called?

Yes this is a problem of data that is huge, data that comes from a number of sources in a number of formats and that too in quick time. This is called as **Big Data**.

This being a problem can be defined in a number of ways, out of which some popular definitions are listed below

Big Data is the amount of data that is beyond the storage and the processing capabilities of a single physical machine.

Big Data is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it.

Every day, we create **2.5 quintillion bytes of data** — so much that 90% of the data in the world today has been created in the last two years alone. This data comes from everywhere: sensors used to gather climate information, posts to social media sites, digital pictures and videos, purchase transaction records, and cell phone GPS signals to name a few. This data is Big Data

Now let us try to pull out some inferences that we actually learned from the above use case and stories

1. Big Data

A term given to large amount of data that is available in huge Volumes, different formats and the rate at which it is generated is very high.

Characteristics of Big Data

Big Data is generally defined as a composition of 3 Vs viz. Volume, Velocity and Variety. There are some other characteristics of this data as well. Let's have a look at all of them.

Volume

The quantity of data that is generated is very important in this context. It is the size of the data which determines the value and potential of the data under consideration and whether it can actually be considered as Big Data or not. The name 'Big Data' itself contains a term which is related to size and hence the characteristic.

Variety

The next aspect of Big Data is its variety. This means that the category to which Big Data belongs to is also a very essential fact that needs to be known by the data analysts. This helps the people, who are closely analysing the data and are associated with it, to effectively use the data to their advantage and thus upholding the importance of the Big Data.

Velocity

The term 'velocity' in the context refers to the speed of generation of data or how fast the data is generated and processed to meet the demands and the challenges which lie ahead in the path of growth and development.

Variability

This is a factor which can be a problem for those who analyse the data. This refers to the inconsistency which can be shown by the data at times, thus hampering the process of being able to handle and manage the data effectively.

Veracity

The quality of the data being captured can vary greatly. Accuracy of analysis depends on the veracity of the source data.

Complexity

Data management can become a very complex process, especially when large volumes of data come from multiple sources. These data need to be linked, connected and correlated in order to be able to grasp the information that is supposed to be conveyed by these data. This situation, is therefore, termed as the 'complexity' of Big Data.

2. Big Data Solutions

This is term used to define all tools and frameworks that help in storing and process this large amount of data which is defined as a problem named Big Data

3. Importance of Big Data

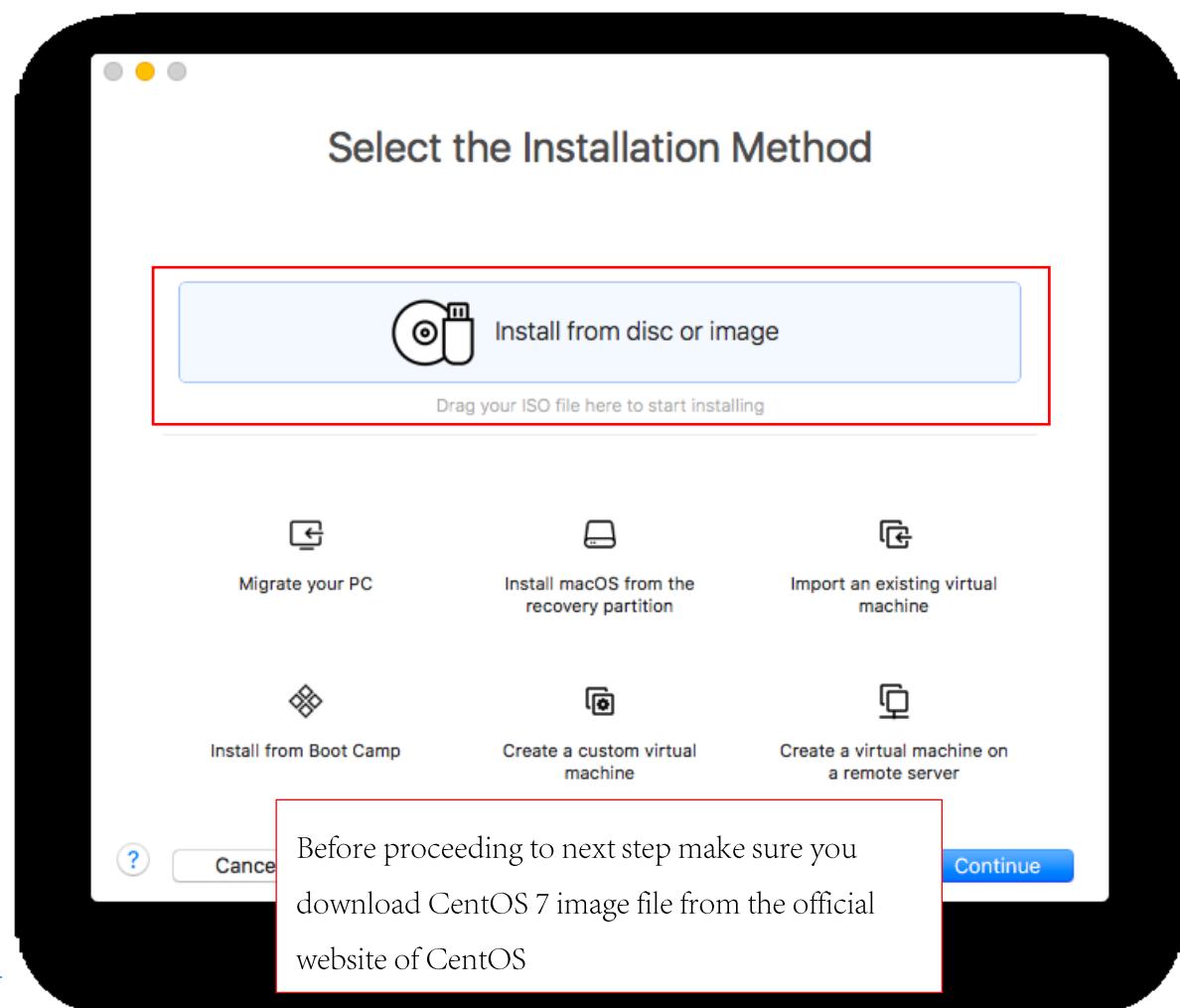
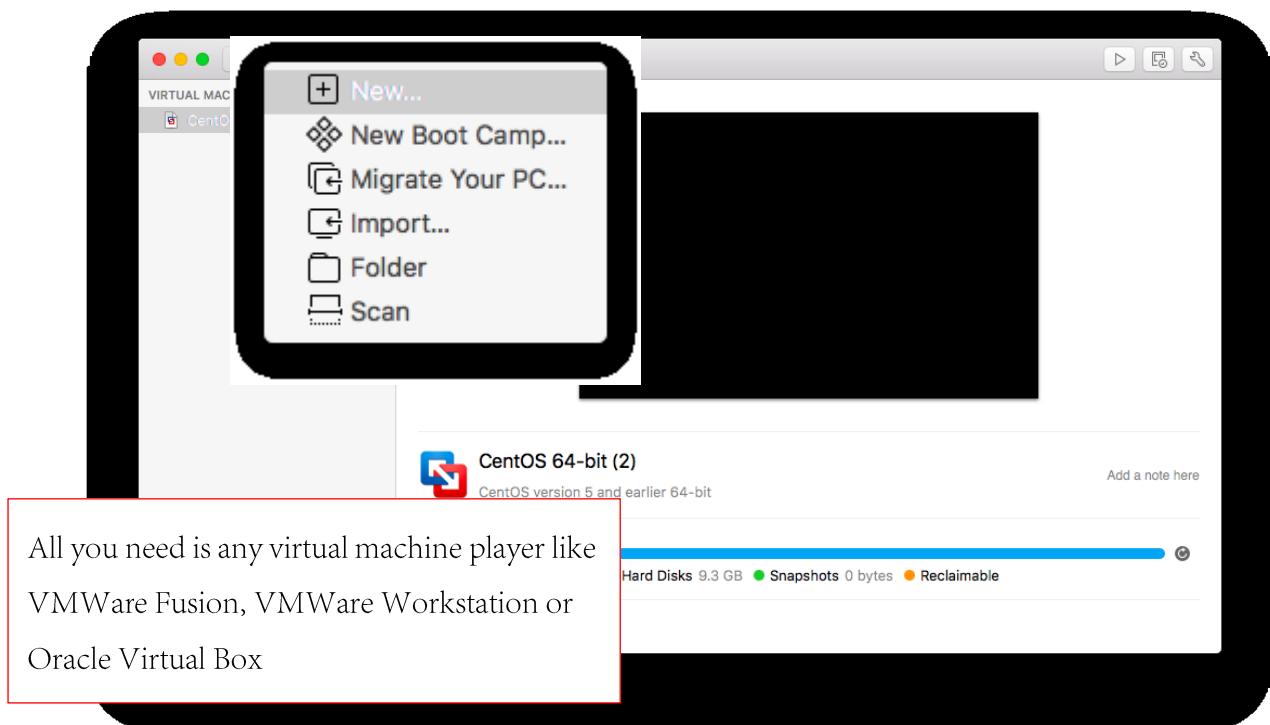
This data gives us customer insights to help us understand our customers in a better way so that we can influence their behavior and get more business.

So is Hadoop a solution?

Hadoop is a framework of software libraries designed in order to handle this problem of Big Data. It is capable of efficiently processing large volumes of Unstructured Data. It can grow limitlessly and hence provides a highly scalable distributed environment as a solution to all our data problems.

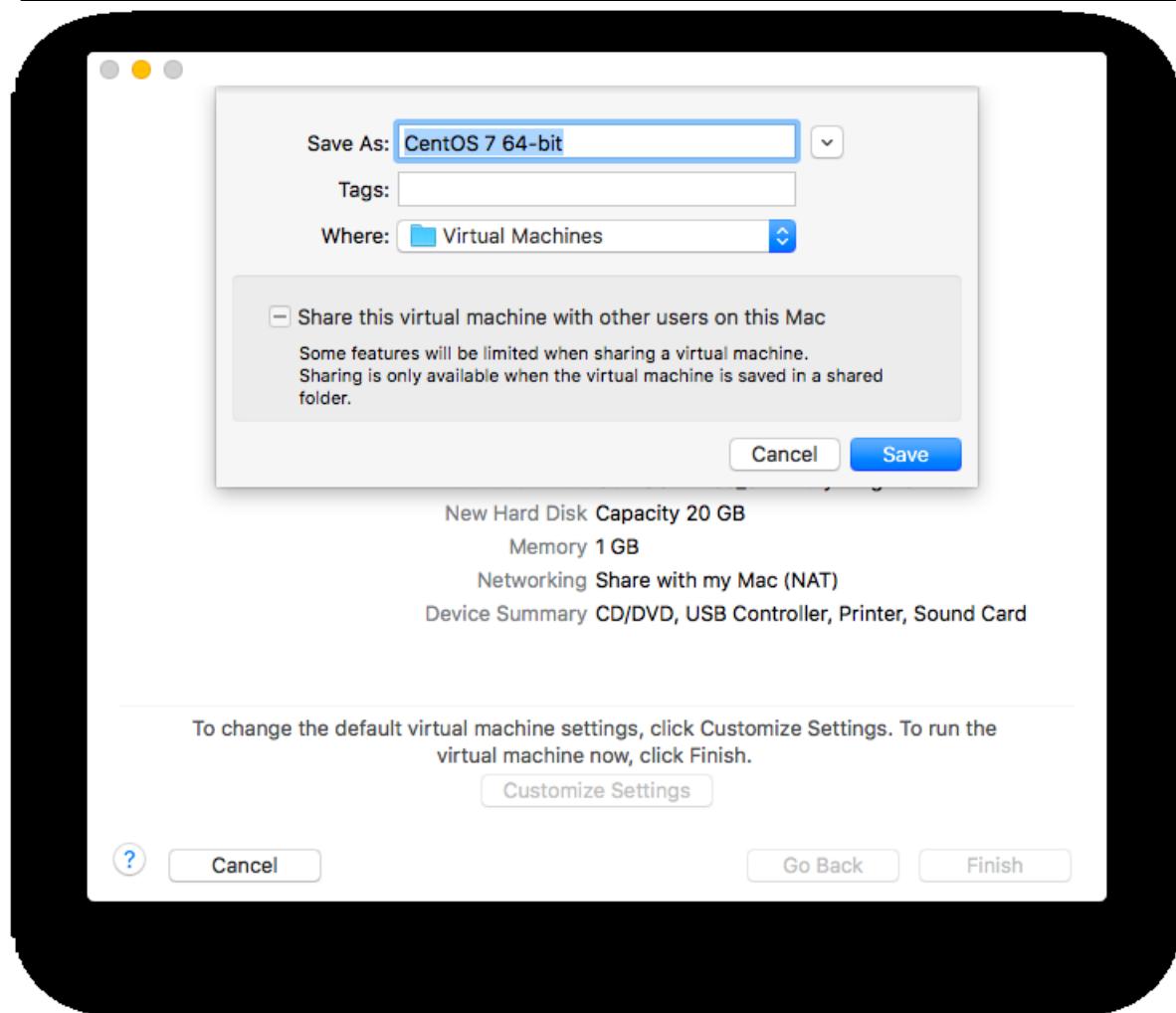
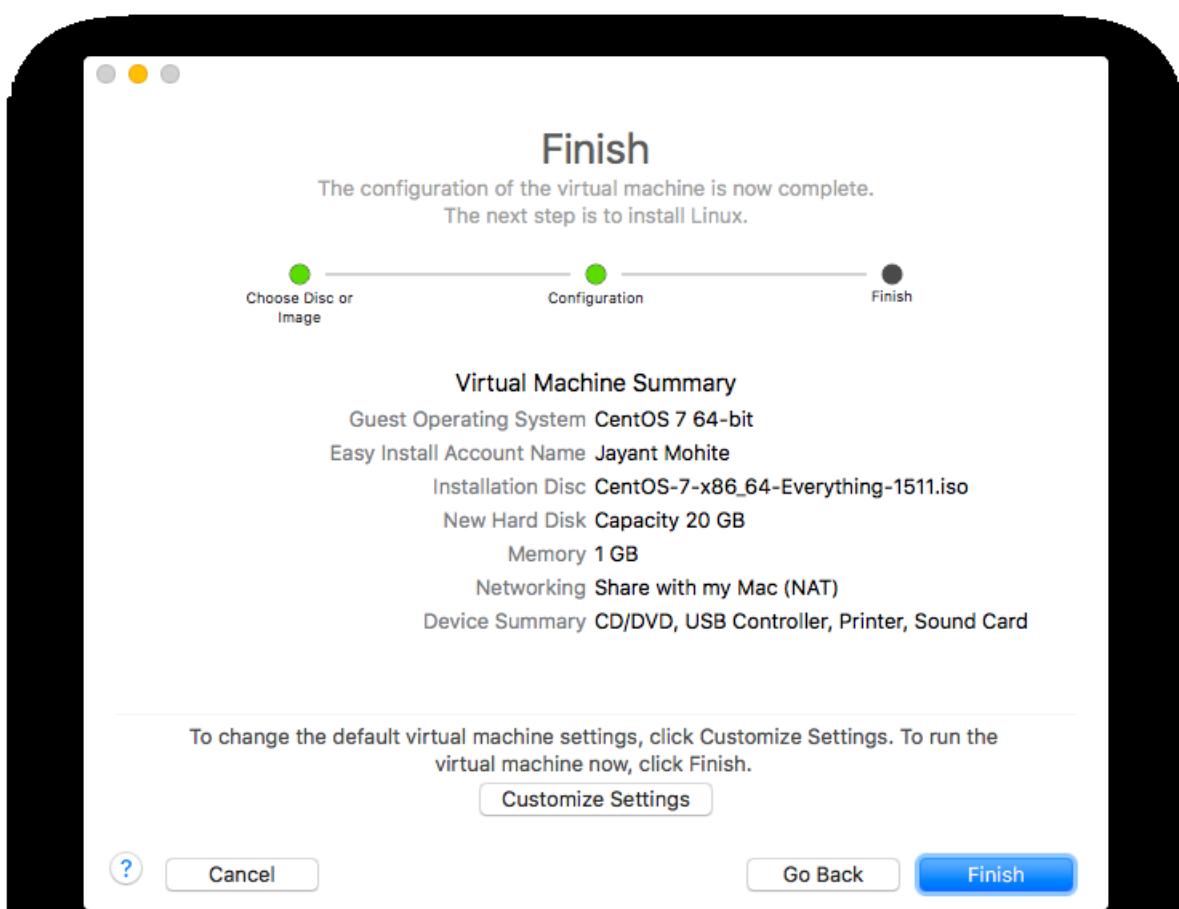
We will talk about Hadoop in more details in the next chapter.

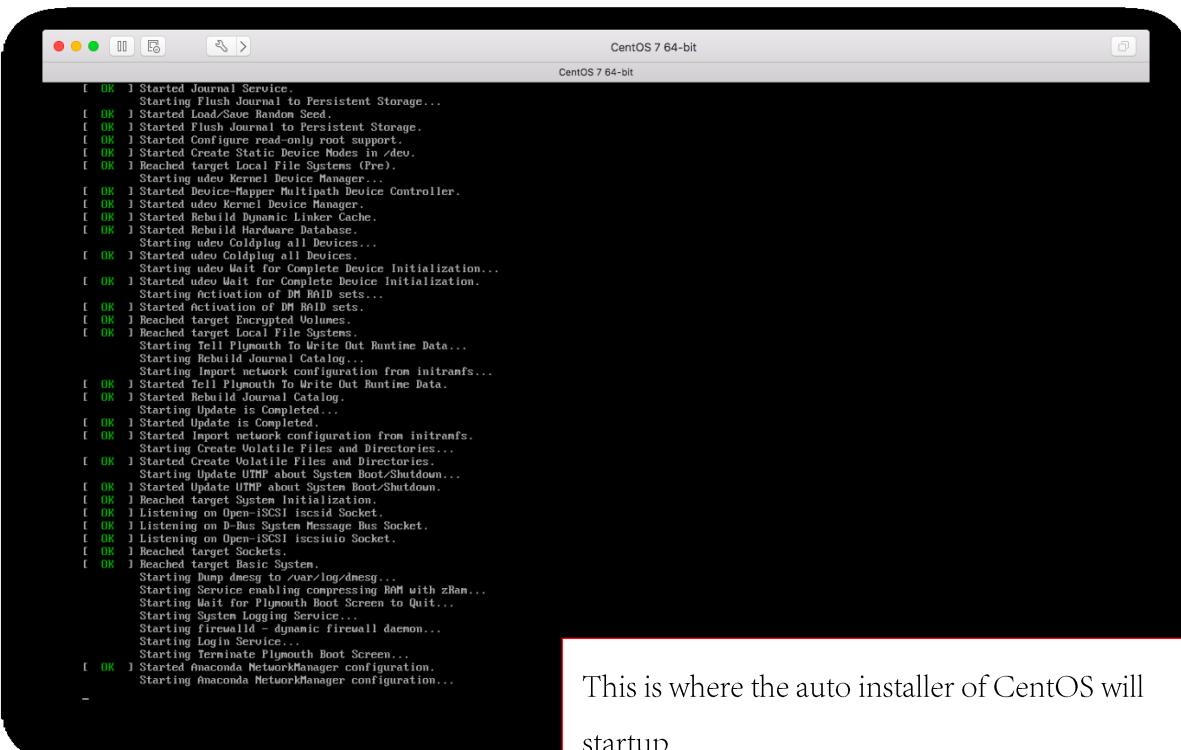
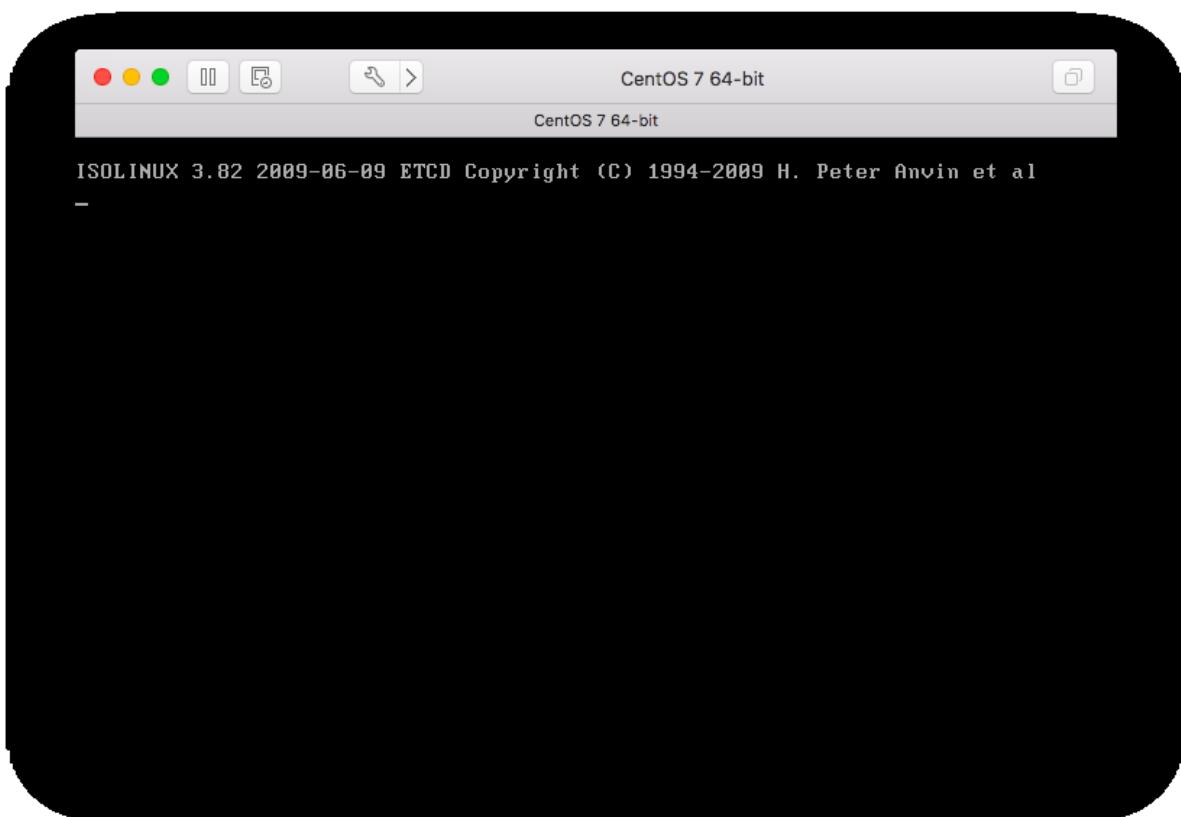
03. Setting up the Environment



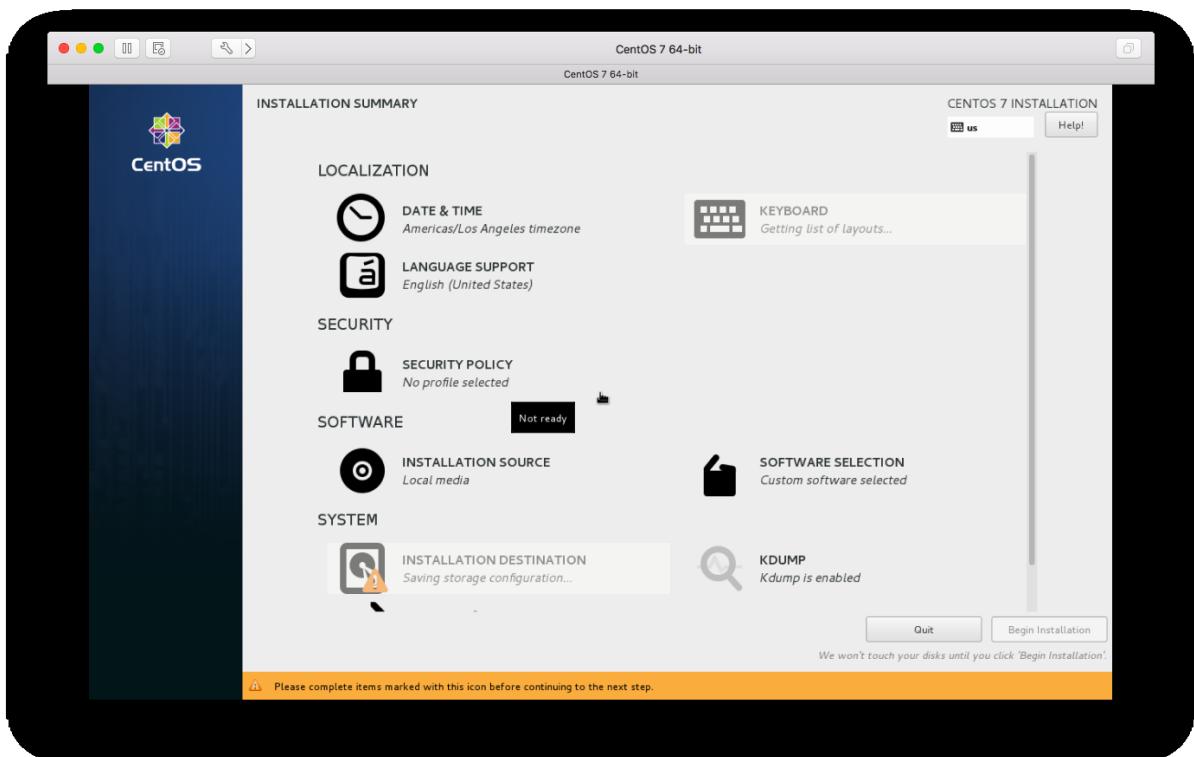
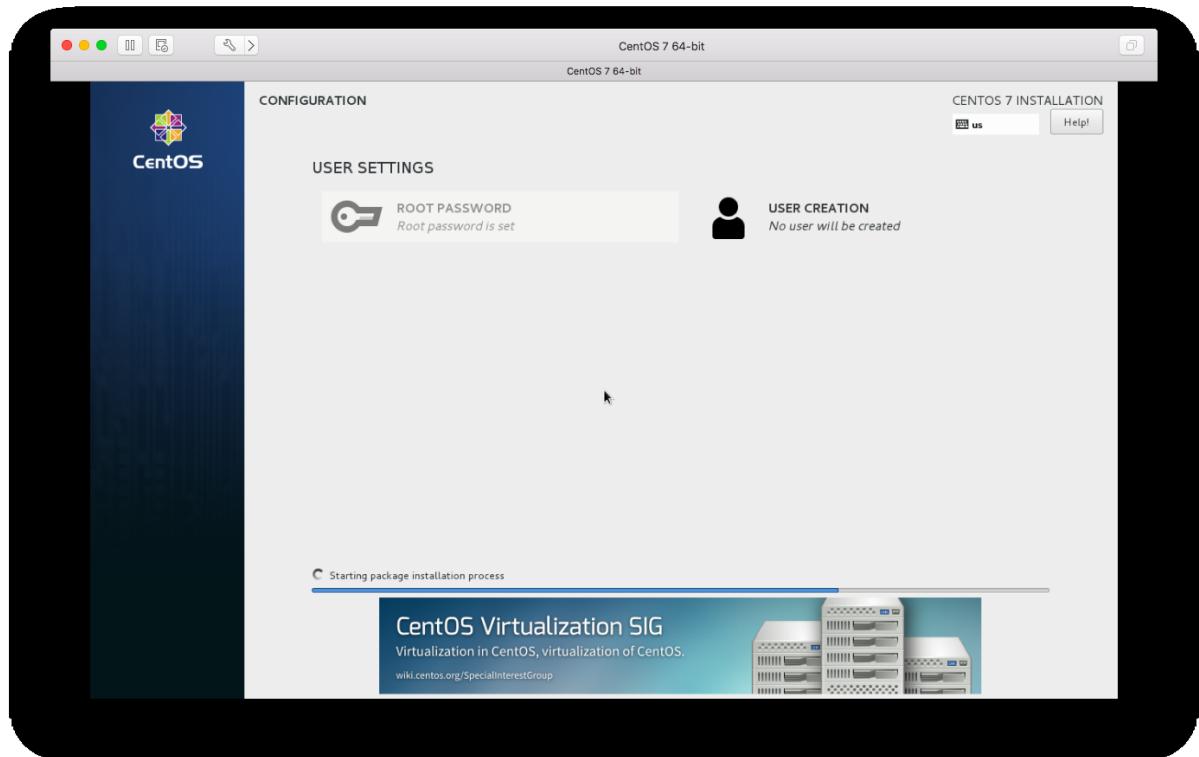
The screenshot shows the first step of the 'Create a New Virtual Machine' wizard. The title is 'Create a New Virtual Machine' and a sub-instruction says 'This will guide you through installing Windows or another operating system in a virtual machine on your Mac.' A progress bar at the top has three steps: 'Choose Disc or Image' (filled), 'Configuration' (empty), and 'Finish' (empty). Below the progress bar, it says 'Choose an operating system installation disc or image:' followed by a dropdown menu containing 'CentOS-7-x86_64-Everything-1511.iso' and 'CentOS 7 64-bit'. A red box highlights the dropdown menu. A callout box with a red border and white background points to the dropdown menu with the text 'Browse to the location where you have save the CentOS image file.' At the bottom are buttons for '?', 'Cancel', 'Go Back', and 'Continue'.

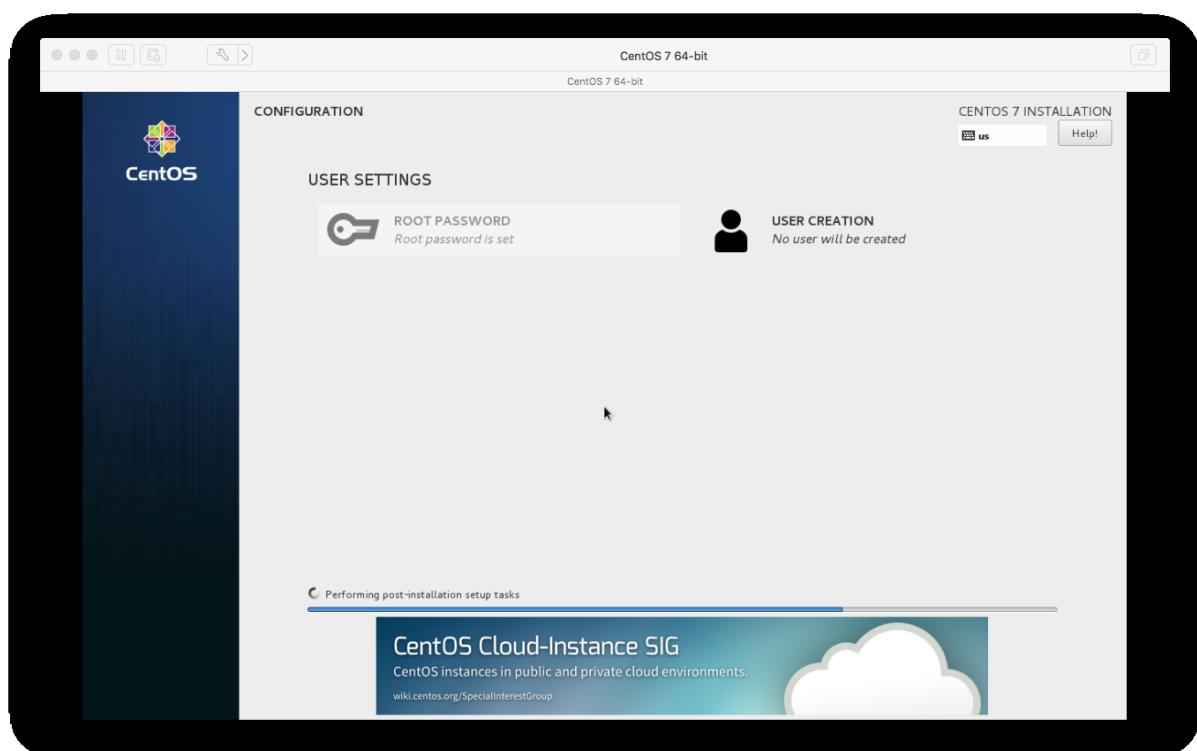
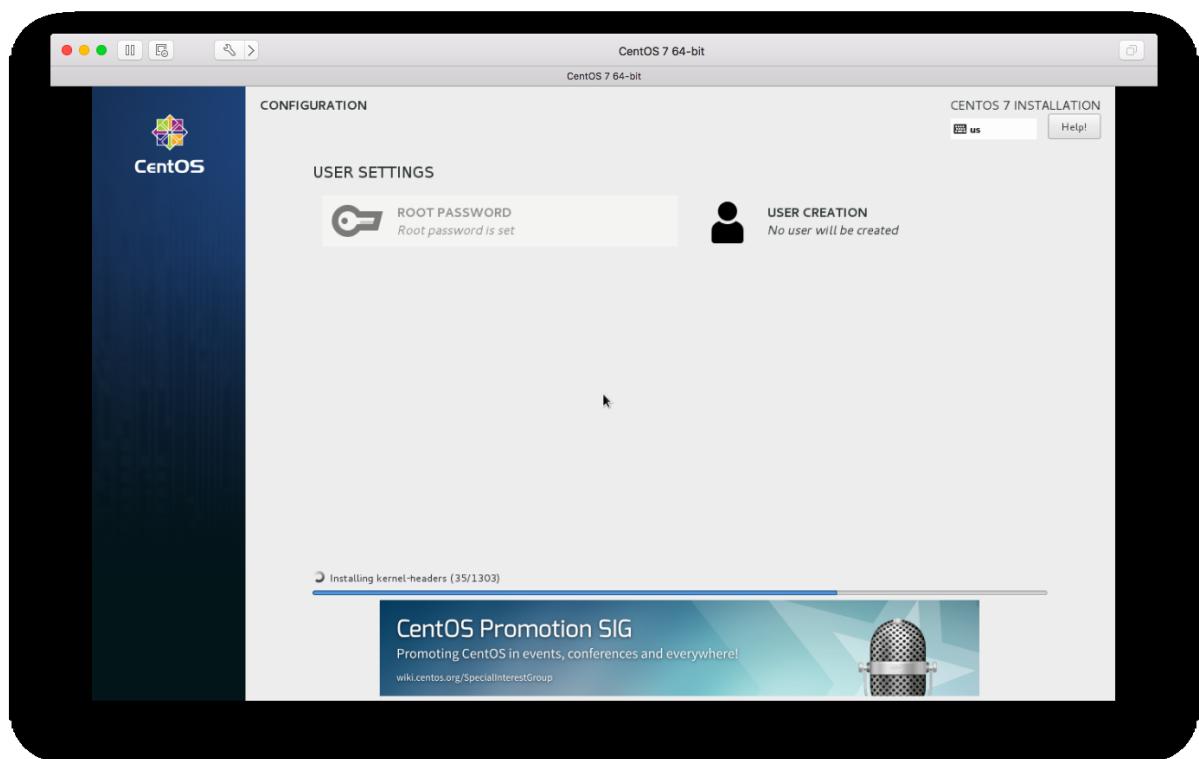
The screenshot shows the 'Configuration' step of the wizard, titled 'Linux Easy Install'. It says 'With Easy Install, VMware Fusion will use the information provided here to automatically install CentOS 7 64-bit from your installation disc and install drivers to optimize your virtual machine.' A progress bar shows 'Choose Disc or Image' (green dot), 'Configuration' (black dot), and 'Finish' (empty). Under 'Use Easy Install', there are fields for 'Display Name' (Jayant Mohite), 'Account Name' (jayantmohite), 'Password' (redacted), and 'Confirm Password' (redacted). A red box highlights the password fields. A callout box with a red border and white background points to the password fields with the text 'This will be the username and password of your machine to be created.' At the bottom are checkboxes for 'Make your home folder accessible to the virtual machine' (unchecked) and 'The virtual machine can' (set to 'Read & Write'), along with buttons for '?', 'Cancel', 'Go Back', and 'Continue'.

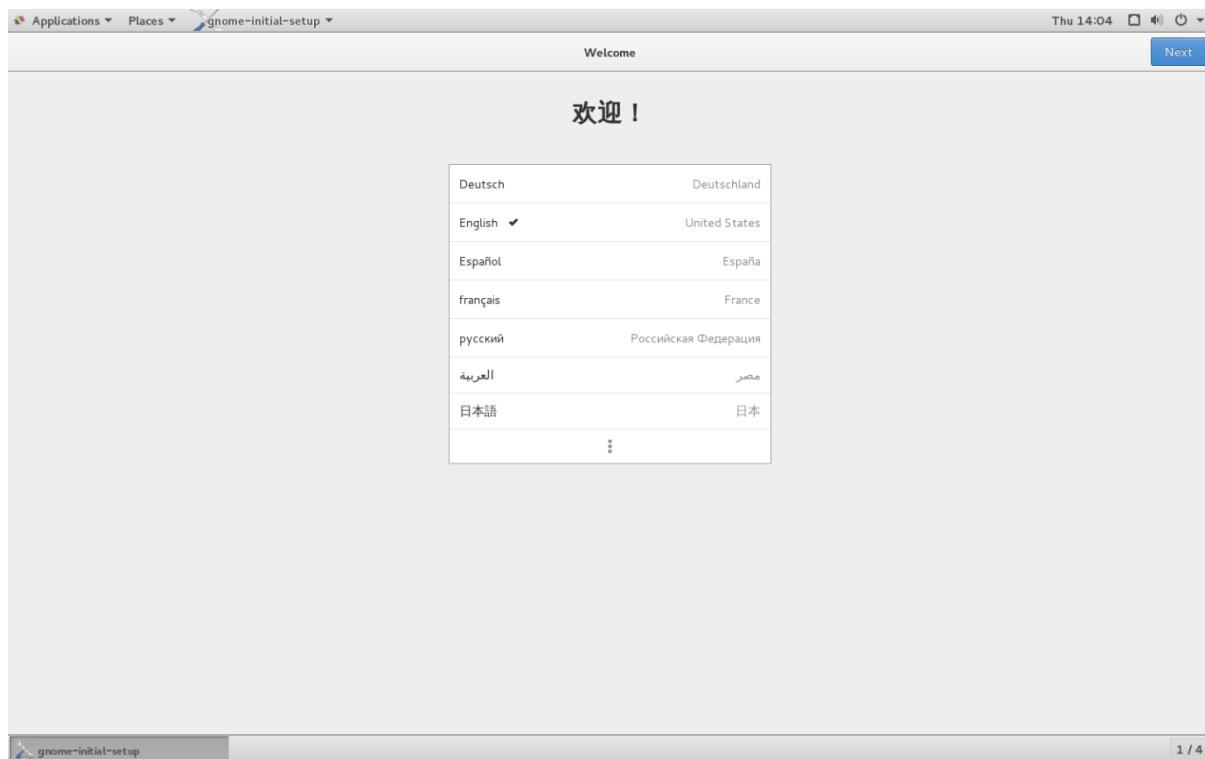
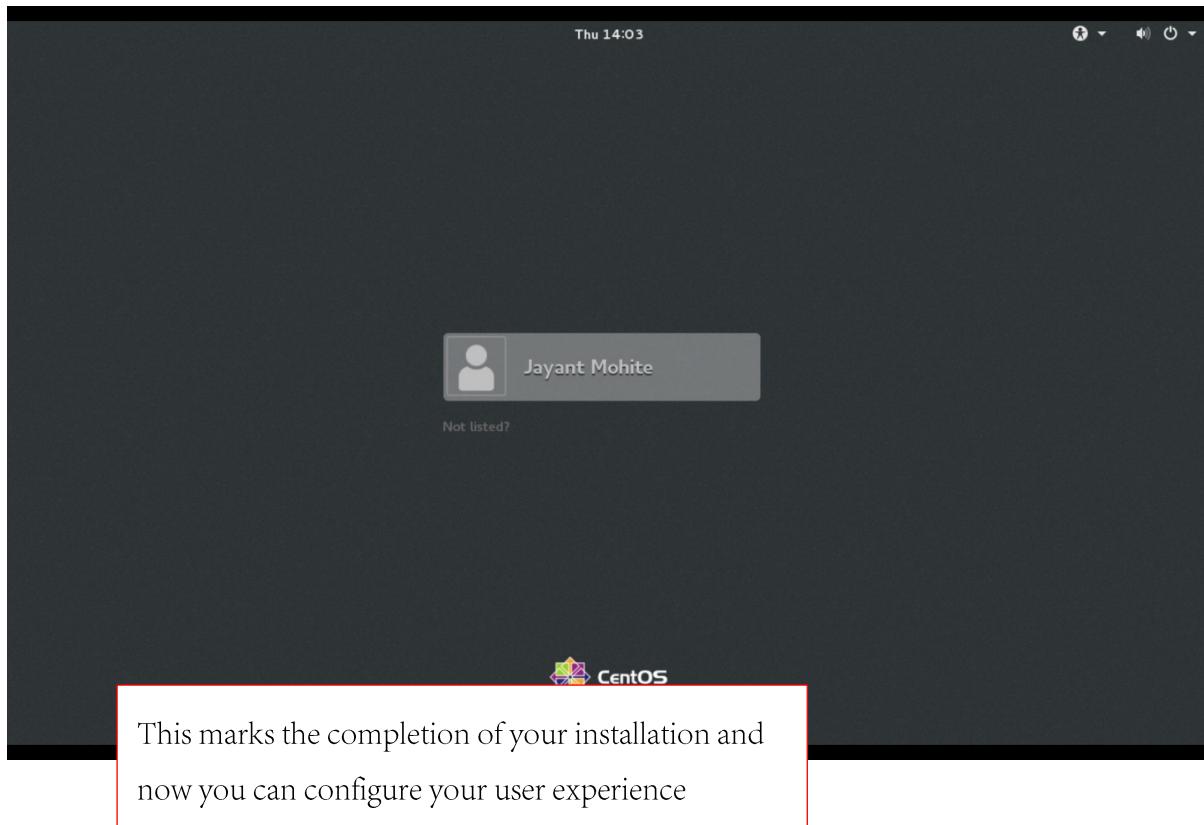


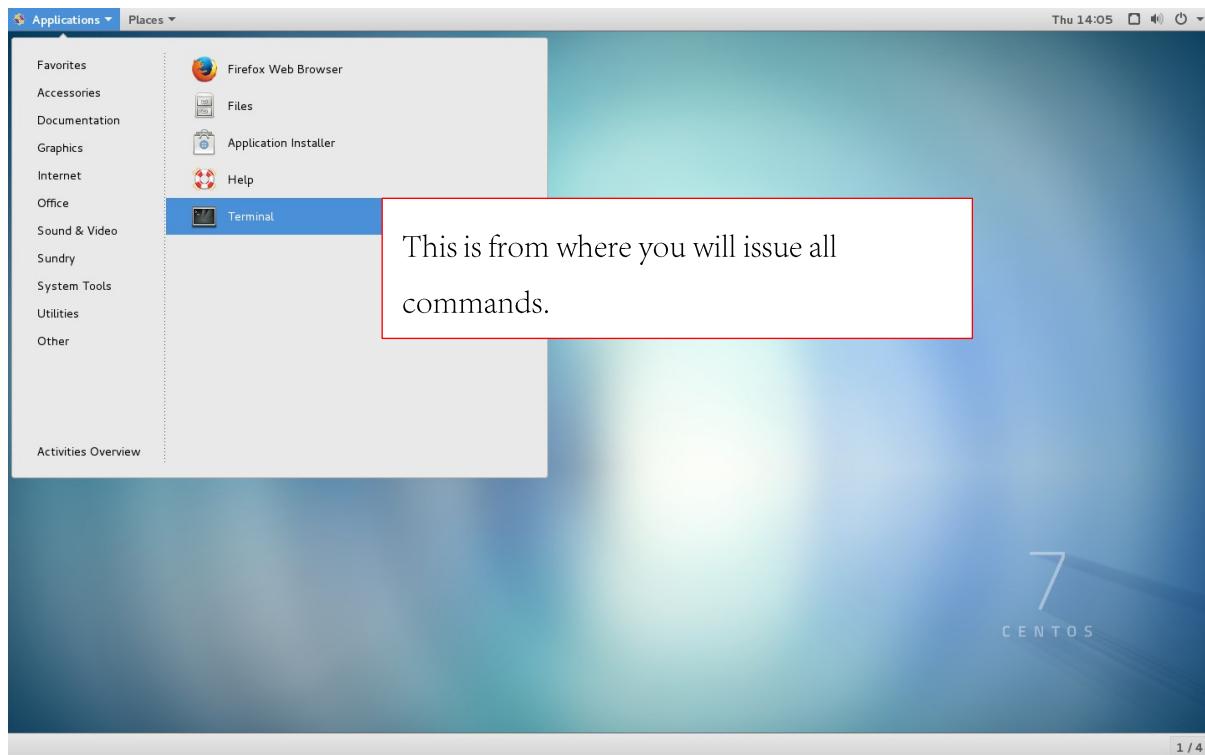
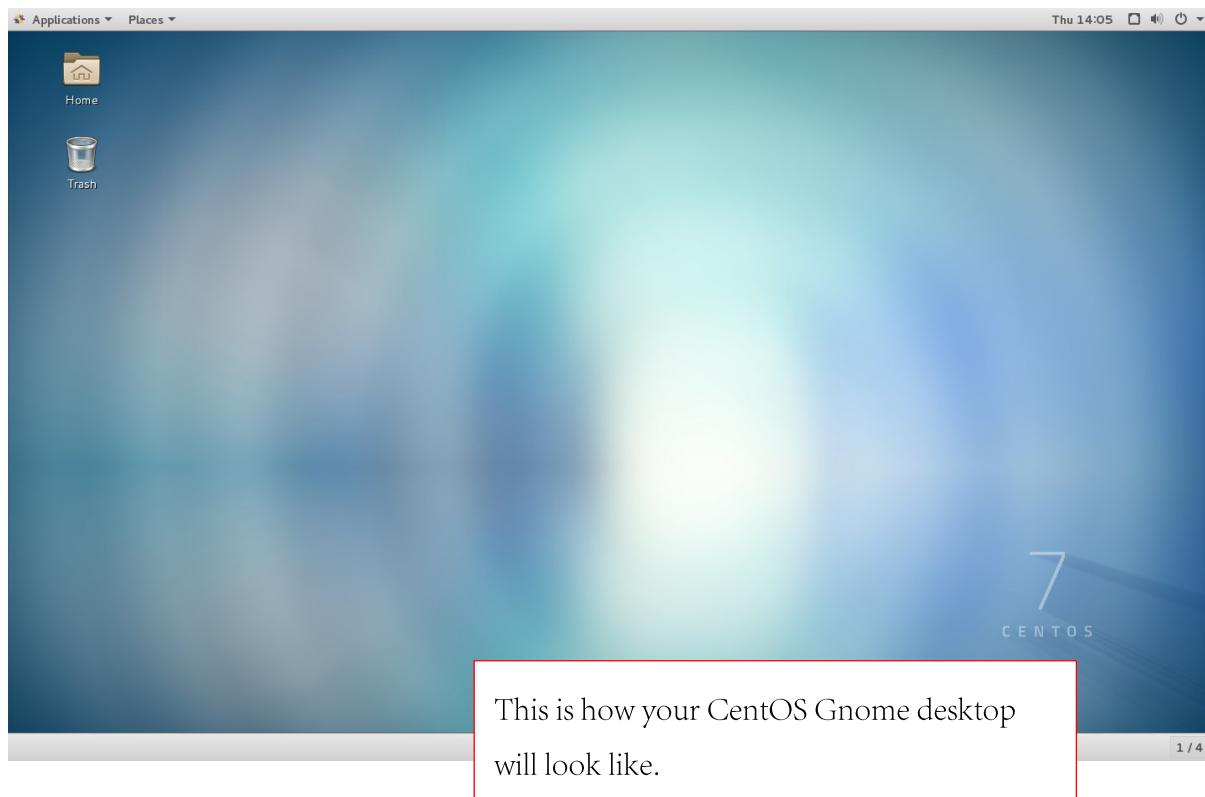


This is where the auto installer of CentOS will startup





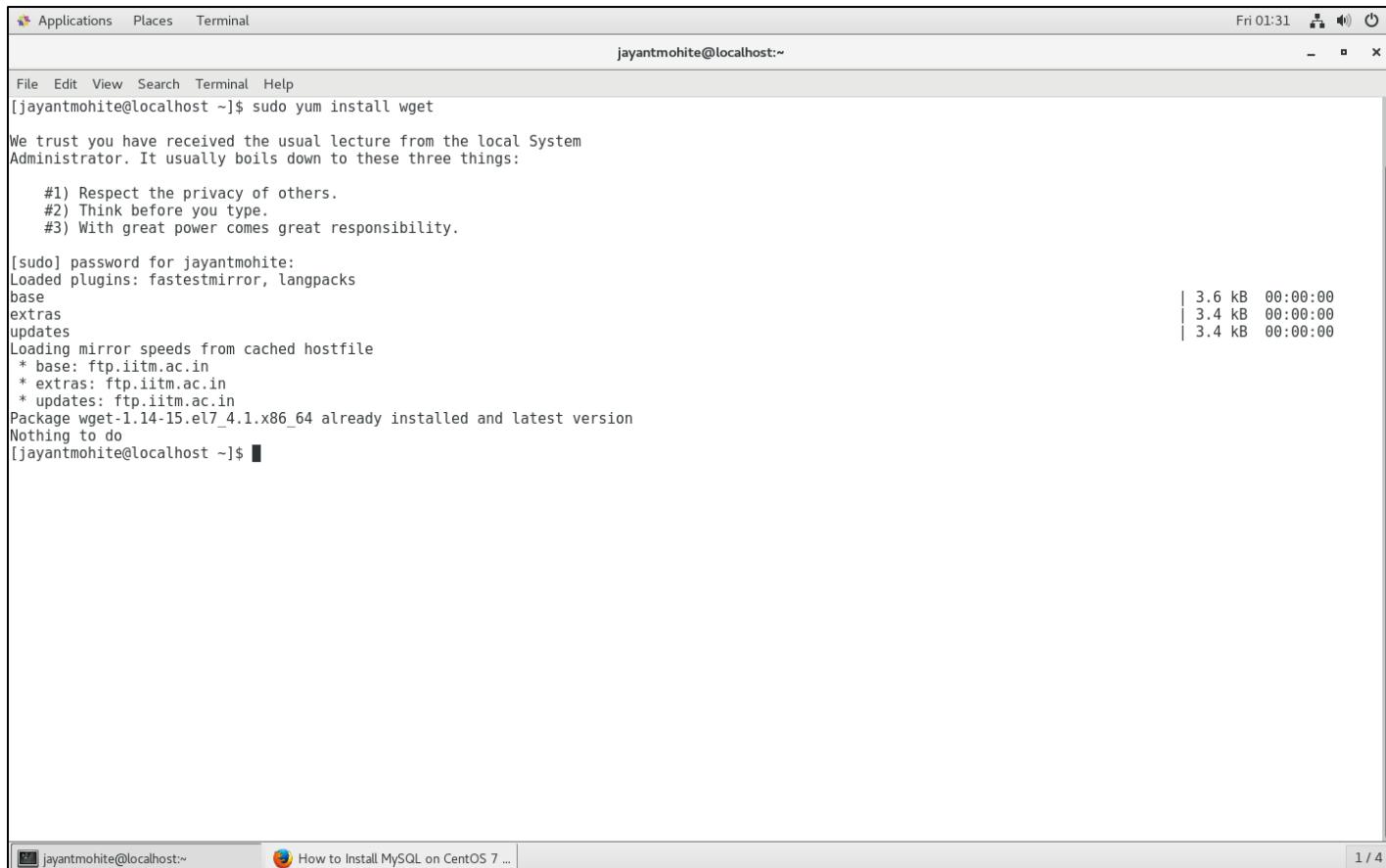




Now that you are done with the installation of CentOS, you can proceed with installation of Java and MySQL before your start installing Hadoop.

04. Installing and Configuring Environment Components

MySQL Installation & Configuration



A screenshot of a Linux terminal window titled "jayantmohite@localhost:~". The window shows the command `sudo yum install wget` being run. The output indicates that wget is already installed and up-to-date. The terminal interface includes a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The status bar at the bottom shows the user's name and the current date and time.

```
[jayantmohite@localhost ~]$ sudo yum install wget
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:
#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for jayantmohite:
Loaded plugins: fastestmirror, langpacks
base
extras
updates
Loading mirror speeds from cached hostfile
 * base: ftp.iitm.ac.in
 * extras: ftp.iitm.ac.in
 * updates: ftp.iitm.ac.in
Package wget-1.14-15.el7_4.1.x86_64 already installed and latest version
Nothing to do
[jayantmohite@localhost ~]$
```

```
[jayantmohite@localhost ~]$ wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
--2018-03-02 01:32:33-- http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
Resolving repo.mysql.com (repo.mysql.com)... 104.120.68.103
Connecting to repo.mysql.com (repo.mysql.com)|104.120.68.103|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6140 (6.0K) [application/x-redhat-package-manager]
Saving to: 'mysql-community-release-el7-5.noarch.rpm'

100%[=====] 6,140      --.-K/s   in 0s

2018-03-02 01:32:34 (296 MB/s) - 'mysql-community-release-el7-5.noarch.rpm' saved [6140/6140]

[jayantmohite@localhost ~]$ sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm
Preparing... ################################ [100%]
Updating / installing...
 1:mysql-community-release-el7-5    ################################ [100%]
[jayantmohite@localhost ~]$
```

jayantmohite@localhost:~ [How to Install MySQL on CentOS 7 ...](#) 1 / 4

```
[jayantmohite@localhost ~]$ wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
--2018-03-02 01:32:33-- http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
Resolving repo.mysql.com (repo.mysql.com)... 104.120.68.103
Connecting to repo.mysql.com (repo.mysql.com)|104.120.68.103|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6140 (6.0K) [application/x-redhat-package-manager]
Saving to: 'mysql-community-release-el7-5.noarch.rpm'

100%[=====] 6,140      --.-K/s   in 0s

2018-03-02 01:32:34 (296 MB/s) - 'mysql-community-release-el7-5.noarch.rpm' saved [6140/6140]

[jayantmohite@localhost ~]$ sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm
Preparing... ################################ [100%]
Updating / installing...
 1:mysql-community-release-el7-5    ################################ [100%]
[jayantmohite@localhost ~]$ yum update
```

jayantmohite@localhost:~ [How to Install MySQL on CentOS 7 ...](#) 1 / 4

```

Applications Places Terminal Fri 01:38
jayantmohite@localhost:~ File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ sudo yum install mysql-server

```

```

jayantmohite@localhost:~ How to Install MySQL on CentOS 7 ...
1 / 4

Applications Places Terminal Fri 01:38
jayantmohite@localhost:~ File Edit View Search Terminal Help
--> Processing Dependency: perl(RPC::PlClient) >= 0.2000 for package: perl-DBI-1.627-4.el7.x86_64
--> Package perl-Data-Dumper.x86_64 0:2.145-3.el7 will be installed
--> Running transaction check
--> Package perl-PlRPC.noarch 0:0.2020-14.el7 will be installed
--> Processing Dependency: perl(Net::Daemon) >= 0.13 for package: perl-PlRPC-0.2020-14.el7.noarch
--> Processing Dependency: perl(Net::Daemon::Test) for package: perl-PlRPC-0.2020-14.el7.noarch
--> Processing Dependency: perl(Net::Daemon::Log) for package: perl-PlRPC-0.2020-14.el7.noarch
--> Processing Dependency: perl(COMPRESS::ZLIB) for package: perl-PlRPC-0.2020-14.el7.noarch
--> Running transaction check
--> Package perl-IO-Compress.noarch 0:2.061-2.el7 will be installed
--> Processing Dependency: perl(COMPRESS::Raw::ZLIB) >= 2.061 for package: perl-IO-Compress-2.061-2.el7.noarch
--> Processing Dependency: perl(COMPRESS::Raw::Bzip2) >= 2.061 for package: perl-IO-Compress-2.061-2.el7.noarch
--> Package perl-Net-Daemon.noarch 0:0.48-5.el7 will be installed
--> Running transaction check
--> Package perl-Compress-Raw-Bzip2.x86_64 0:2.061-3.el7 will be installed
--> Package perl-Compress-Raw-Zlib.x86_64 1:2.061-4.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version            Repository        Size
=====
Installing:
mysql-community-server   x86_64    5.6.39-2.el7      mysql56-community   59 M
Installing for dependencies:
mysql-community-client   x86_64    5.6.39-2.el7      mysql56-community   19 M
perl-Compress-Raw-Bzip2  x86_64    2.061-3.el7       base              32 K
perl-Compress-Raw-Zlib   x86_64    1:2.061-4.el7     base              57 K
perl-DBI               x86_64    1.627-4.el7       base             802 K
perl-Data-Dumper        x86_64    2.145-3.el7       base             47 K
perl-IO-Compress         noarch   2.061-2.el7       base            260 K
perl-Net-Daemon          noarch   0.48-5.el7       base             51 K
perl-PlRPC              noarch   0.2020-14.el7     base            36 K

Transaction Summary
=====
Install 1 Package (+8 Dependent packages)

Total download size: 80 M
Installed size: 343 M
Is this ok [y/d/N]: 
```

Fri 01:42

```
jayantmohite@localhost:~
```

File Edit View Search Terminal Help

(3/9): perl-Compress-Raw-Zlib-2.061-4.el7.x86_64.rpm | 57 kB 00:00:01
(4/9): perl-Net-Daemon-0.48-5.el7.noarch.rpm | 51 kB 00:00:00
(5/9): perl-PLRPC-0.2020-14.el7.noarch.rpm | 36 kB 00:00:00
(6/9): perl-DBI-1.627-4.el7.x86_64.rpm | 802 kB 00:00:01
(7/9): perl-IO-Compress-2.061-2.el7.noarch.rpm | 260 kB 00:00:00
(8/9): mysql-community-client-5.6.39-2.el7.x86_64.rpm | 19 MB 00:00:08
(9/9): mysql-community-server-5.6.39-2.el7.x86_64.rpm | 59 MB 00:00:17

Total 4.5 MB/s | 80 MB 00:00:17

Running transaction check
Running transaction test
Transaction test succeeded
Running transaction

Installing : perl-Data-Dumper-2.145-3.el7.x86_64 1/9
Installing : mysql-community-client-5.6.39-2.el7.x86_64 2/9
Installing : perl-Compress-Raw-Zlib-2.061-4.el7.x86_64 3/9
Installing : perl-Net-Daemon-0.48-5.el7.noarch 4/9
Installing : perl-Compress-Raw-Bzip2-2.061-3.el7.x86_64 5/9
Installing : perl-IO-Compress-2.061-2.el7.noarch 6/9
Installing : perl-PLRPC-0.2020-14.el7.noarch 7/9
Installing : perl-DBI-1.627-4.el7.x86_64 8/9
Installing : mysql-community-server-5.6.39-2.el7.x86_64 9/9
Verifying : mysql-community-server-5.6.39-2.el7.x86_64 1/9
Verifying : perl-Compress-Raw-Bzip2-2.061-3.el7.x86_64 2/9
Verifying : perl-Net-Daemon-0.48-5.el7.noarch 3/9
Verifying : perl-Data-Dumper-2.145-3.el7.x86_64 4/9
Verifying : perl-PLRPC-0.2020-14.el7.noarch 5/9
Verifying : perl-DBI-1.627-4.el7.x86_64 6/9
Verifying : perl-IO-Compress-2.061-2.el7.noarch 7/9
Verifying : mysql-community-client-5.6.39-2.el7.x86_64 8/9

Installed:

mysql-community-server.x86_64 0:5.6.39-2.el7

Dependency Installed:

mysql-community-client.x86_64 0:5.6.39-2.el7 perl-Compress-Raw-Bzip2.x86_64 0:2.061-3.el7 perl-Compress-Raw-Zlib.x86_64 1:2.061-4.el7
perl-DBI.x86_64 0:1.627-4.el7 perl-Data-Dumper.x86_64 0:2.145-3.el7 perl-IO-Compress.noarch 0:2.061-2.el7
perl-Net-Daemon.noarch 0:0.48-5.el7

Complete!

[jayantmohite@localhost ~]\$

jayantmohite@localhost:~ [How to Install MySQL on CentOS 7 ...] 1 / 4

Fri 01:49

```
jayantmohite@localhost:~
```

File Edit View Search Terminal Help

[jayantmohite@localhost ~]\$ sudo service mysqld start
[sudo] password for jayantmohite:
Redirecting to /bin/systemctl start mysqld.service
[jayantmohite@localhost ~]\$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2.
Server version: 5.6.39 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;

Database
information_schema
mysql
performance_schema

3 rows in set (0.01 sec)

mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> update user SET PASSWORD=PASSWORD("password") WHERE USER='root';
Query OK, 4 rows affected (0.03 sec)
Rows matched: 4 Changed: 4 Warnings: 0

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> [How to Install MySQL on CentOS 7 ...] 1 / 4

Applications Places Terminal Fri 13:47

jayantmohite@localhost:~

```
[jayantmohite@localhost ~]$ sudo mysql_secure_installation
[sudo] password for jayantmohite:

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
      SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current
password for the root user. If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL
root user without the proper authorisation.

You already have a root password set, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables...
... Success!

By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n
... skipping.

[ Home jayantmohite@localhost:~ How to Install MySQL on CentOS 7... ] 1 / 4
```

Applications Places Terminal Fri 13:47

jayantmohite@localhost:~

```
File Edit View Search Terminal Help
to log into MySQL without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n
... skipping.

By default, MySQL comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
ERROR 1008 (HY000) at line 1: Can't drop database 'test'; database doesn't exist
... Failed! Not critical, keep moving...
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

All done! If you've completed all of the above steps, your MySQL
installation should now be secure.

Thanks for using MySQL!

Cleaning up...
[jayantmohite@localhost ~]$
```

[Home jayantmohite@localhost:~ How to Install MySQL on CentOS 7...] 1 / 4

```

Applications Places Terminal Fri 14:18
jayantmohite@localhost:~ - x
File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ sudo mkdir -p /var/lib/sqoop
[sudo] password for jayantmohite:
[jayantmohite@localhost ~]$ sudo chown sqoop:sqoop /var/lib/sqoop
[jayantmohite@localhost ~]$ sudo chmod 755 /var/lib/sqoop
[jayantmohite@localhost ~]$ 

```

Home jayantmohite@localhost:~ Browsing HDFS - Mozilla Firefox 1 / 4

MySQL :: Download Connector/J - Mozilla Firefox

Browsing HDFS | localhost Hadoop Ma... | MySQL :: Download C... | +

<https://dev.mysql.com/downloads/connector/j/5.1.html>

The world's most popular open source database

Contact MySQL | Login | Register

MySQL.COM DOWNLOADS DOCUMENTATION DEVELOPER ZONE

f t in g y

Enterprise Community Yum Repository APT Repository SUSE Repository Windows Archives

> MySQL on Windows
MySQL Yum Repository
MySQL APT Repository
MySQL SUSE Repository
MySQL Community Server
MySQL Cluster
MySQL Router
MySQL Utilities
MySQL Shell
MySQL Workbench
MySQL Connectors
Connector/ODBC

Download Connector/J

MySQL Connector/J is the official JDBC driver for MySQL.

Online Documentation:

- [MySQL Connector/J Installation Instructions](#)
- [Documentation](#)
- [MySQL Connector/J X DevAPI Reference](#) (requires Connector/J 8.0)
- [Change History](#)

Please report any bugs or inconsistencies you observe to our [Bugs Database](#).
Thank you for your support!

MySQL open source software is provided under the [GPL License](#).
OEMs, ISVs and VARs can purchase commercial licenses.

Generally Available (GA) Releases Development Releases

Connector/J 5.1.45

Home jayantmohite@localhost:~ MySQL :: Download Connector/J - ... 1 / 4

Applications Places Firefox Web Browser Fri 14:19

MySQL :: Download Connector/J - Mozilla Firefox

Browsing HDFS x localhost Hadoop Ma... x MySQL :: Download C... x +

<https://dev.mysql.com/downloads/connector/j/5.1.html>

MySQL JDBC Repository

- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Utilities
- MySQL Shell
- MySQL Workbench
- MySQL Connectors
- Connector/ODBC
- Connector/Net
- Connector/J
- Connector/Node.js
- Connector/Python
- Connector/C++
- Connector/C
- MySQL Native Driver for PHP
- Other Downloads

MySQL Connector/J Installation Instructions
 Documentation
 MySQL Connector/J DevAPI Reference (requires Connector/J 8.0)
 Change History

Please report any bugs or inconsistencies you observe to our [Bugs Database](#).
 Thank you for your support!

Generally Available (GA) Releases Development Releases

Connector/J 5.1.45

Select Operating System:
 Platform Independent

Platform Independent (Architecture Independent), Compressed TAR Archive	5.1.45	3.3M	Download
(mysql-connector-java-5.1.45.tar.gz)			MD5: ab9ac454a959859a297b53bdbf156f3c Signature

Platform Independent (Architecture Independent), ZIP Archive	5.1.45	3.6M	Download
(mysql-connector-java-5.1.45.zip)			MD5: 2e3a933f8f7642345d5f34bf43374f8e Signature

We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

https://dev.mysql.com/downloads/file/?id=474257

Home jayantmohite@localhost:~ MySQL :: Download Connector/J - ... 1 / 4

Applications Places Firefox Web Browser Fri 14:20

MySQL :: Begin Your Download - Mozilla Firefox

MySQL :: Begin Your ... x +

<https://dev.mysql.com/downloads/file/?id=474257>

Enterprise Community Yum Repository APT Repository SUSE Repository Windows Archives

MySQL on Windows
 MySQL Yum Repository
 MySQL APT Repository
 MySQL SUSE Repository
 MySQL Community Server
 MySQL Cluster
 MySQL Router
 MySQL Utilities
 MySQL Shell
 MySQL Workbench
 MySQL Connectors
 Other Downloads

Begin Your Download

mysql-connector-java-5.1.45.tar.gz

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system
- Comment in the MySQL Documentation

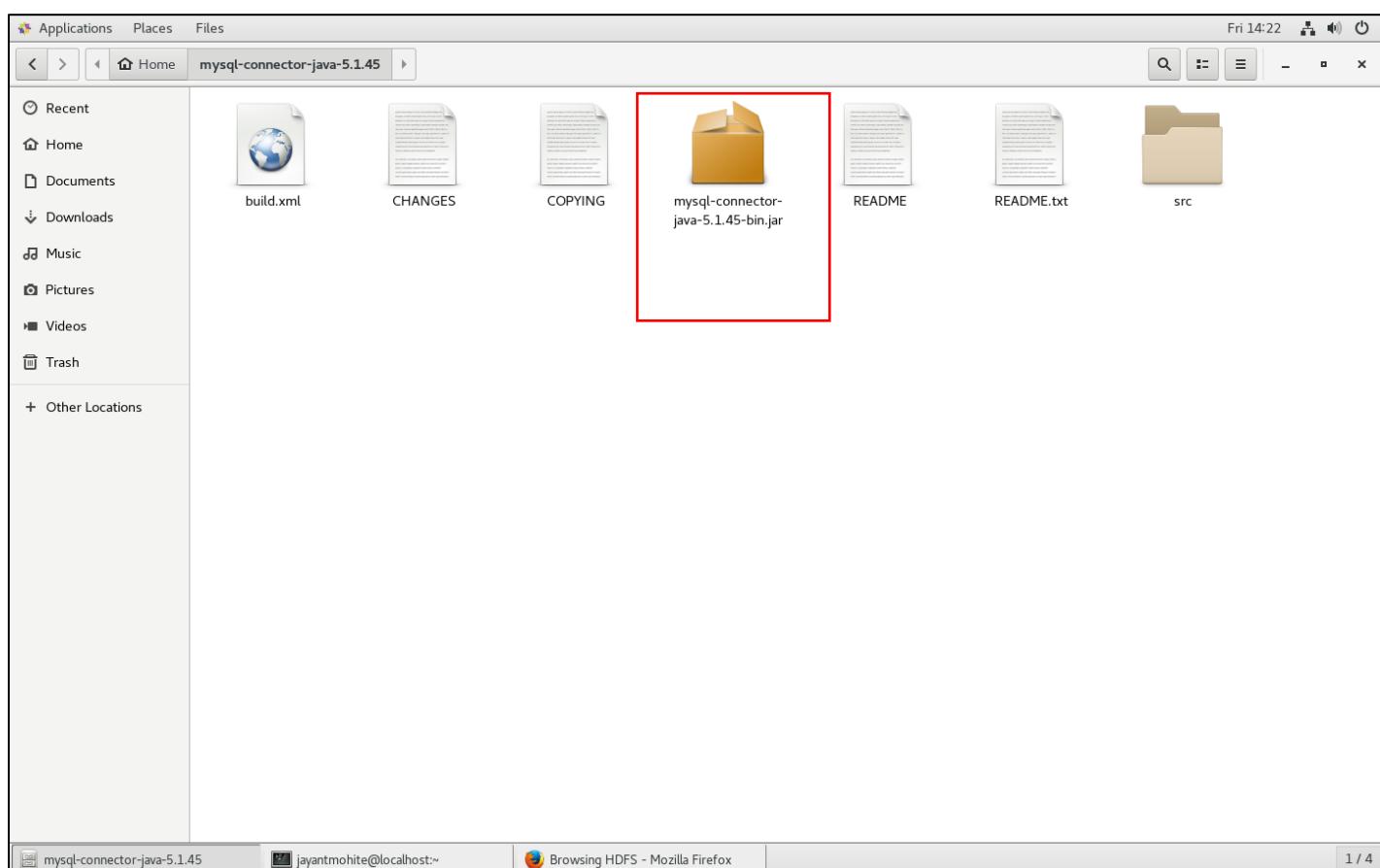
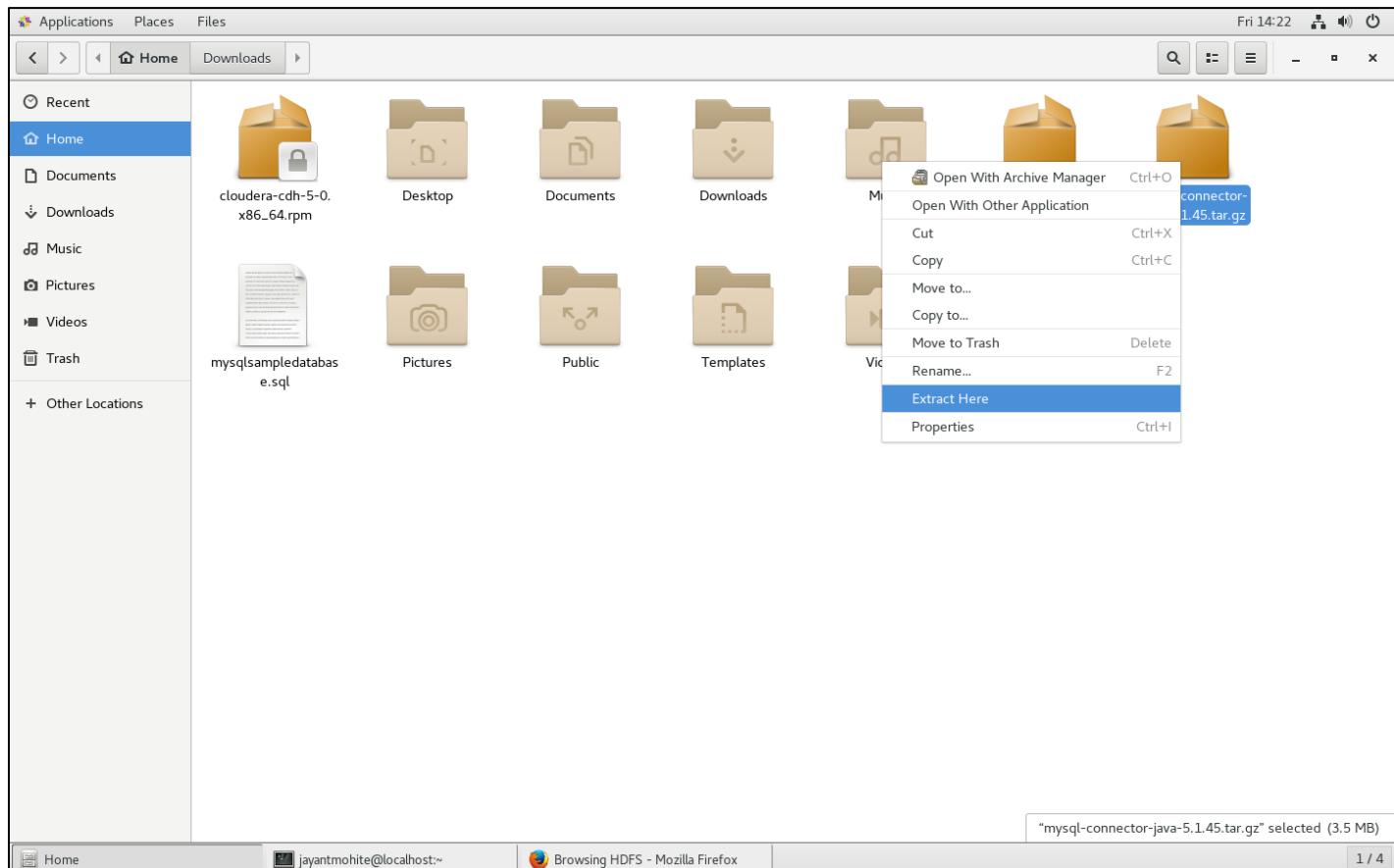
[Login »](#) using my Oracle Web account [Sign Up »](#) for an Oracle Web account

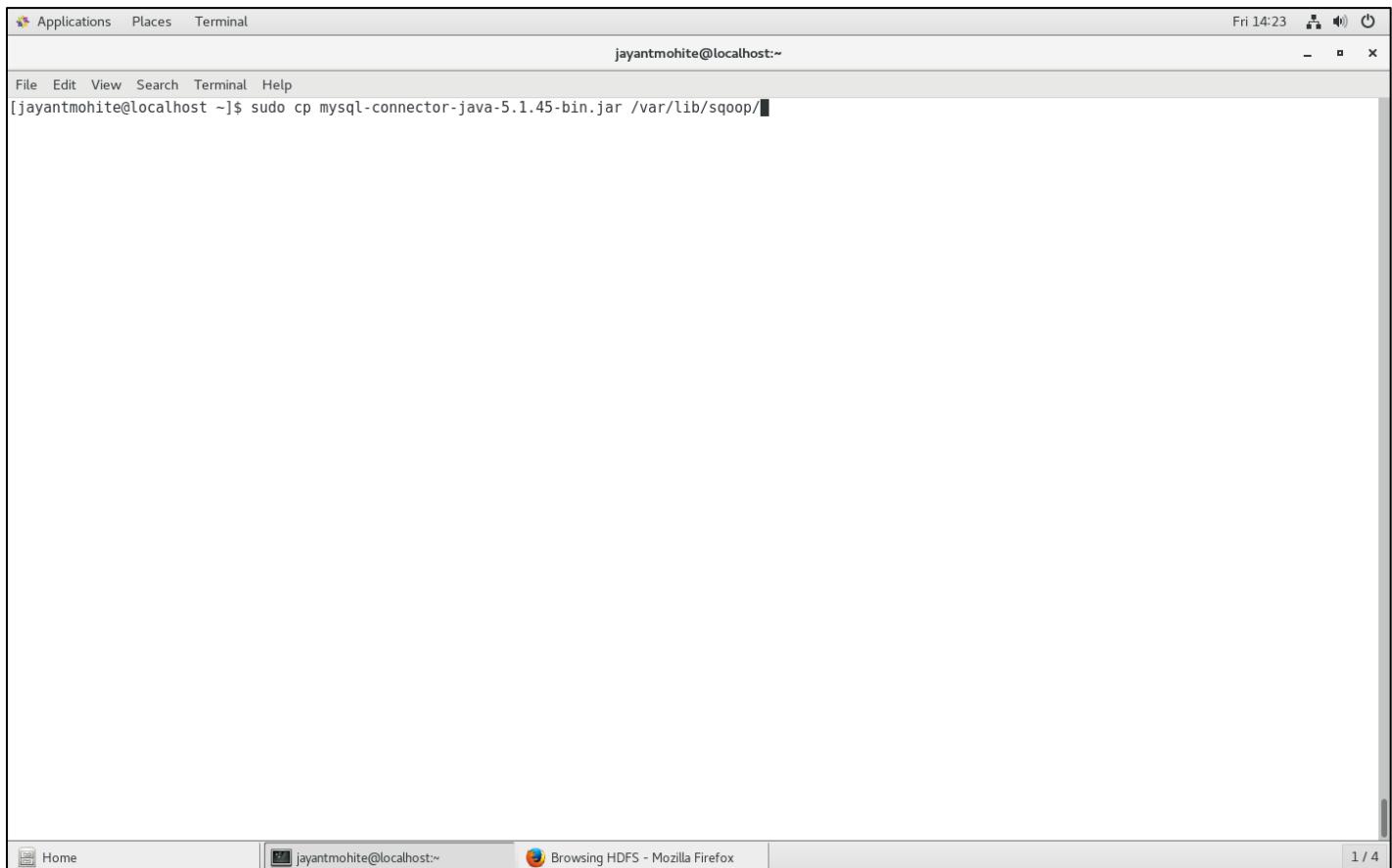
MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

[No thanks, just start my download.](#)

https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.45.tar.gz

Home jayantmohite@localhost:~ MySQL :: Download Connector/J - ... MySQL :: Begin Your Download - M... 1 / 4





A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a title bar with the user's name 'jayantmohite@localhost:~'. The window content shows the command 'sudo cp mysql-connector-java-5.1.45-bin.jar /var/lib/sqoop/'. Below the terminal window, the desktop taskbar is visible with icons for Home, a terminal window labeled 'jayantmohite@localhost:~', and a Firefox browser window titled 'Browsing HDFS - Mozilla Firefox'. The status bar at the bottom right of the screen shows the date and time as 'Fri 14:23'.

With this we complete the installation of MySQL and also the configuration required to use it along with Sqoop.

Java Installation and Configuration

```

Applications Places Terminal Fri 14:36
jayantmohite@localhost:/opt

File Edit View Search Terminal Help
[jroot@localhost ~]# cd /opt/
[jroot@localhost opt]# wget --no-cookies --no-check-certificate --header "Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz"
--2018-03-02 14:34:13-- http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz
Resolving download.oracle.com (download.oracle.com)... 104.120.66.137
Connecting to download.oracle.com (download.oracle.com)|104.120.66.137|:80... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://edelivery.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz [following]
--2018-03-02 14:34:13-- https://edelivery.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz
Resolving edelivery.oracle.com (edelivery.oracle.com)... 104.120.83.172, 2600:140f:3000:1a2::2d3e, 2600:140f:3000:184::2d3e
Connecting to edelivery.oracle.com (edelivery.oracle.com)|104.120.83.172|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz?AuthParam=1520030174_a5af7e839e03d
a4b5d10551dbf2da183 [following]
--2018-03-02 14:34:14-- http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz?AuthParam=1520030174_a5af7e839e03d
a4b5d10551dbf2da183
Connecting to download.oracle.com (download.oracle.com)|104.120.66.137|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 189756259 (181M) [application/x-gzip]
Saving to: 'jdk-8u161-linux-x64.tar.gz'

100%[=====] 189,756,259 3.22MB/s in 54s

2018-03-02 14:35:09 (3.37 MB/s) - 'jdk-8u161-linux-x64.tar.gz' saved [189756259/189756259]

[jroot@localhost opt]# tar xzf jdk-8u161-linux-x64.tar.gz

```

```

Applications Places Terminal Fri 14:38
jayantmohite@localhost:/opt/jdk1.8.0_161

File Edit View Search Terminal Help
[jroot@localhost ~]# cd /opt/
[jroot@localhost opt]# wget --no-cookies --no-check-certificate --header "Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz"
--2018-03-02 14:34:13-- http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz
Resolving download.oracle.com (download.oracle.com)... 104.120.66.137
Connecting to download.oracle.com (download.oracle.com)|104.120.66.137|:80... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://edelivery.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz [following]
--2018-03-02 14:34:13-- https://edelivery.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz
Resolving edelivery.oracle.com (edelivery.oracle.com)... 104.120.83.172, 2600:140f:3000:1a2::2d3e, 2600:140f:3000:184::2d3e
Connecting to edelivery.oracle.com (edelivery.oracle.com)|104.120.83.172|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz?AuthParam=1520030174_a5af7e839e03d
a4b5d10551dbf2da183 [following]
--2018-03-02 14:34:14-- http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz?AuthParam=1520030174_a5af7e839e03d
a4b5d10551dbf2da183
Connecting to download.oracle.com (download.oracle.com)|104.120.66.137|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 189756259 (181M) [application/x-gzip]
Saving to: 'jdk-8u161-linux-x64.tar.gz'

100%[=====] 189,756,259 3.22MB/s in 54s

2018-03-02 14:35:09 (3.37 MB/s) - 'jdk-8u161-linux-x64.tar.gz' saved [189756259/189756259]

[jroot@localhost opt]# tar xzf jdk-8u161-linux-x64.tar.gz
[jroot@localhost opt]# cd jdk1.8.0_161/
[jroot@localhost jdk1.8.0_161]# alternatives --install /usr/bin/java java /opt/jdk1.8.0_161/bin/java 2
[jroot@localhost jdk1.8.0_161]# alternatives --config java

There are 3 programs which provide 'java'.

 Selection Command
-----+
 1      java-1.7.0-openjdk.x86_64 (/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.171-2.6.13.0.el7_4.x86_64/jre/bin/java)
*+ 2      java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/jre/bin/java)
 3      /opt/jdk1.8.0_161/bin/java

Enter to keep the current selection[+], or type selection number: 3


```

File Edit View Search Terminal Help

[root@localhost ~]# alternatives --install /usr/bin/jar jar /opt/jdk1.8.0_161/bin/jar 2
[root@localhost ~]# alternatives --install /usr/bin/javac javac /opt/jdk1.8.0_161/bin/javac 2
[root@localhost ~]# alternatives --set jar /opt/jdk1.8.0_161/bin/jar
[root@localhost ~]# alternatives --set javac /opt/jdk1.8.0_161/bin/javac
[root@localhost ~]# java -version
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)
[root@localhost ~]# javac -version
javac 1.8.0_161
[root@localhost ~]#

A screenshot of a Linux terminal window titled "jayantmohite@localhost:~". The window shows the contents of the .bashrc file. The file includes global alias definitions for rm, cp, and mv, a source command for /etc/bashrc, and environment variable exports for JAVA_HOME, JRE_HOME, and PATH. The bottom status bar indicates the user is in INSERT mode.

```
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

export JAVA_HOME=/opt/jdk1.8.0_161
export JRE_HOME=/opt/jdk1.8.0_161/jre
export PATH=$PATH:/opt/jdk1.8.0_161/bin:/opt/jdk1.8.0_161/jre/bin

-- INSERT --
```

```

Applications Places Terminal
jayantmohite@localhost:~ Fri 14:45
File Edit View Search Terminal Help
[root@localhost ~]# vi ~/.bashrc
[root@localhost ~]# source ~/.bashrc
[root@localhost ~]#

```

Home jayantmohite@localhost:~ ERROR orm.CompilationManager: PL... 1 / 4

Applications Places Firefox Web Browser Fri 15:10

Download java-json.jar : java json « j « Jar File Download - Mozilla Firefox

C Solved: Sqoop import... x Download java-json.ja... +

www.java2s.com/Code/Jar/j/Downloadjavajsonjar.htm

Search

Home Jar File Download a b c d e f g h i [j] k l m n o p q r s t u v w x y z

Download java-json.jar : java json « j « Jar File Download

Linode Cloud Hosting Root Access, 1GB RAM for Only \$5/month! 7 Day Money Back Guarantee.

Jar File Download / j / java json /

Download java-json.jar

[java-json/java-json.jar.zip\(81 k\)](#)

The download jar file contains the following class files or Java source files.

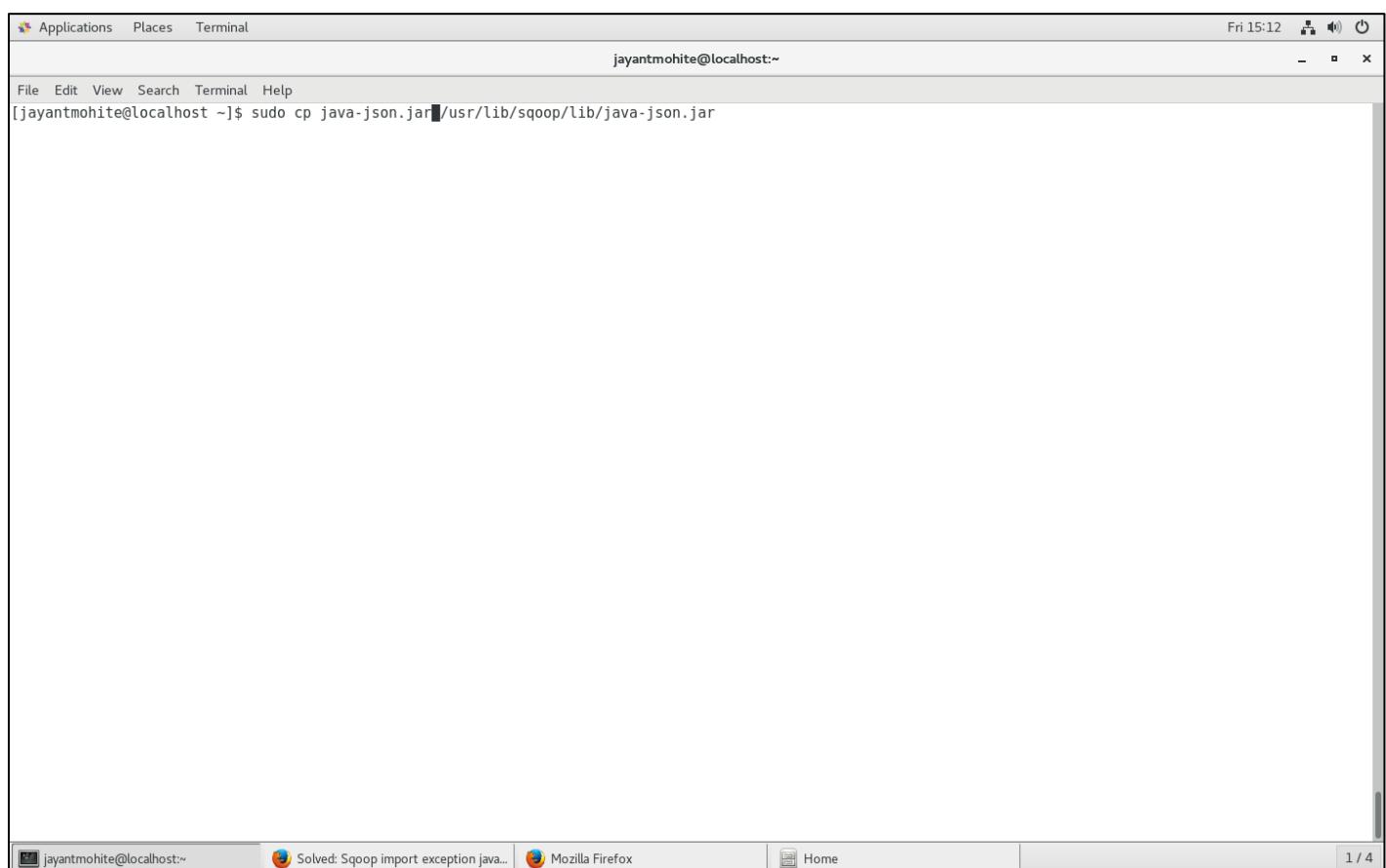
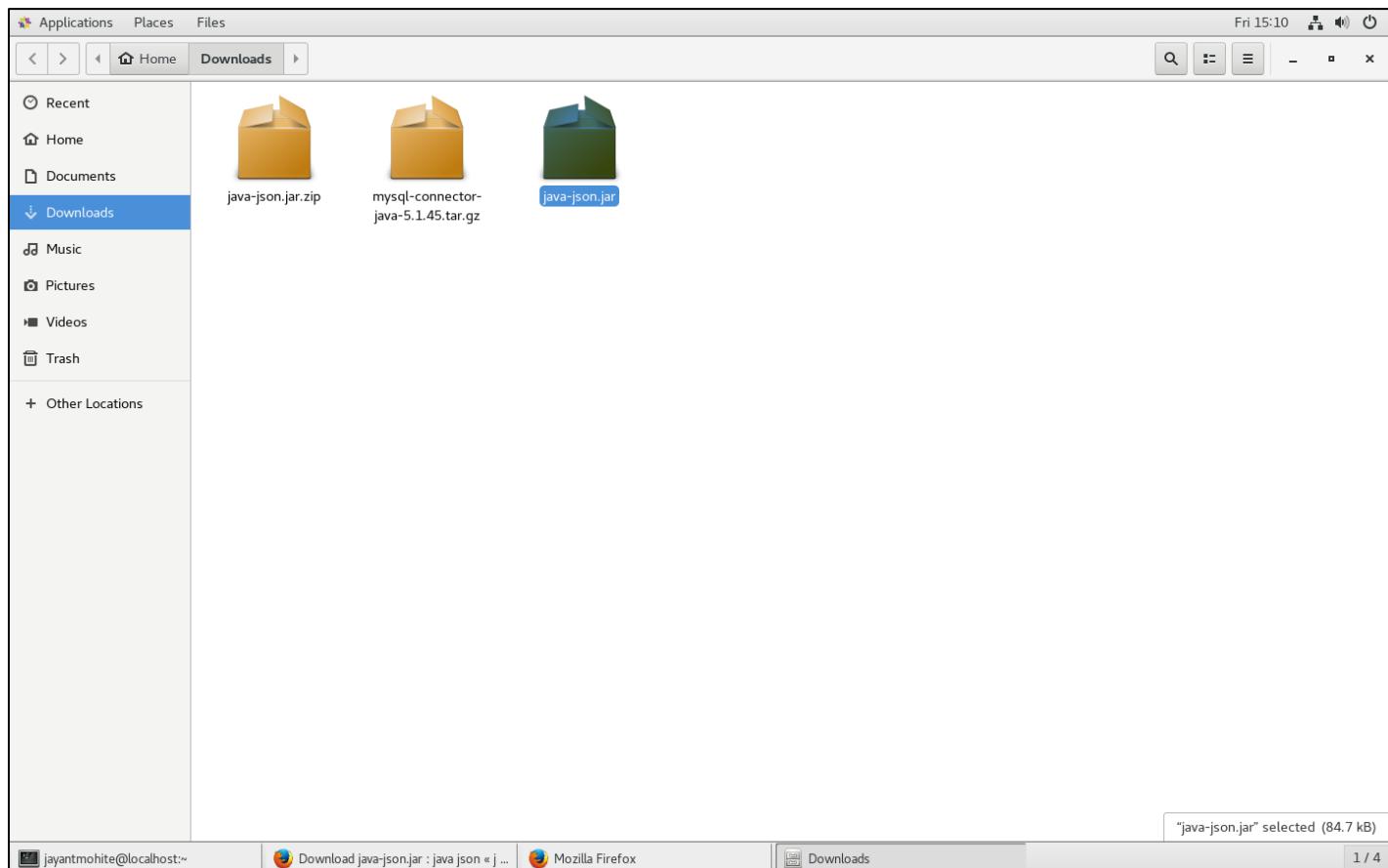
```

META-INF/MANIFEST.MF
org.json.CDL.class
org.json.CDL.java
org.json.Cookie.class
org.json.Cookie.java
org.json.CookieList.class
org.json.CookieList.java
org.json.HTTP.class
org.json.HTTP.java

```

www.java2s.com/Code/JarDownload/java-json/java-json.jar.zip

jayantmohite@localhost:~ Download java-json.jar : java json « j ... 1 / 4



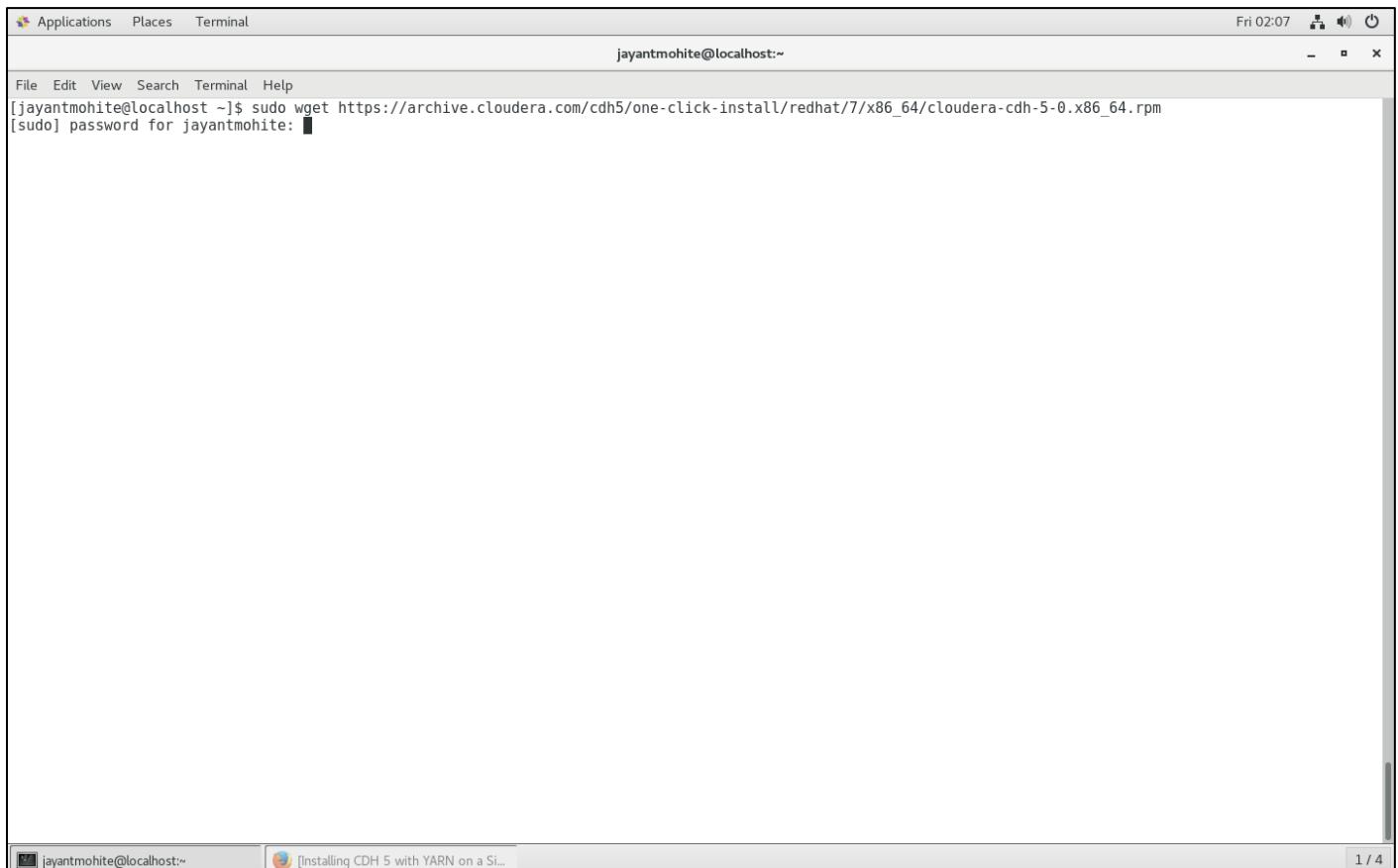
With this we complete the Installation and Configuration of Java for using along side the Hadoop Framework.

Now that you are done with setting up the environment with all components required, step into the next chapter and get yourself acquainted with the details of Hadoop framework.

05. Installation of Hadoop Generation 1

Step1:

Download the Cloudera Package of Hadoop Version 5

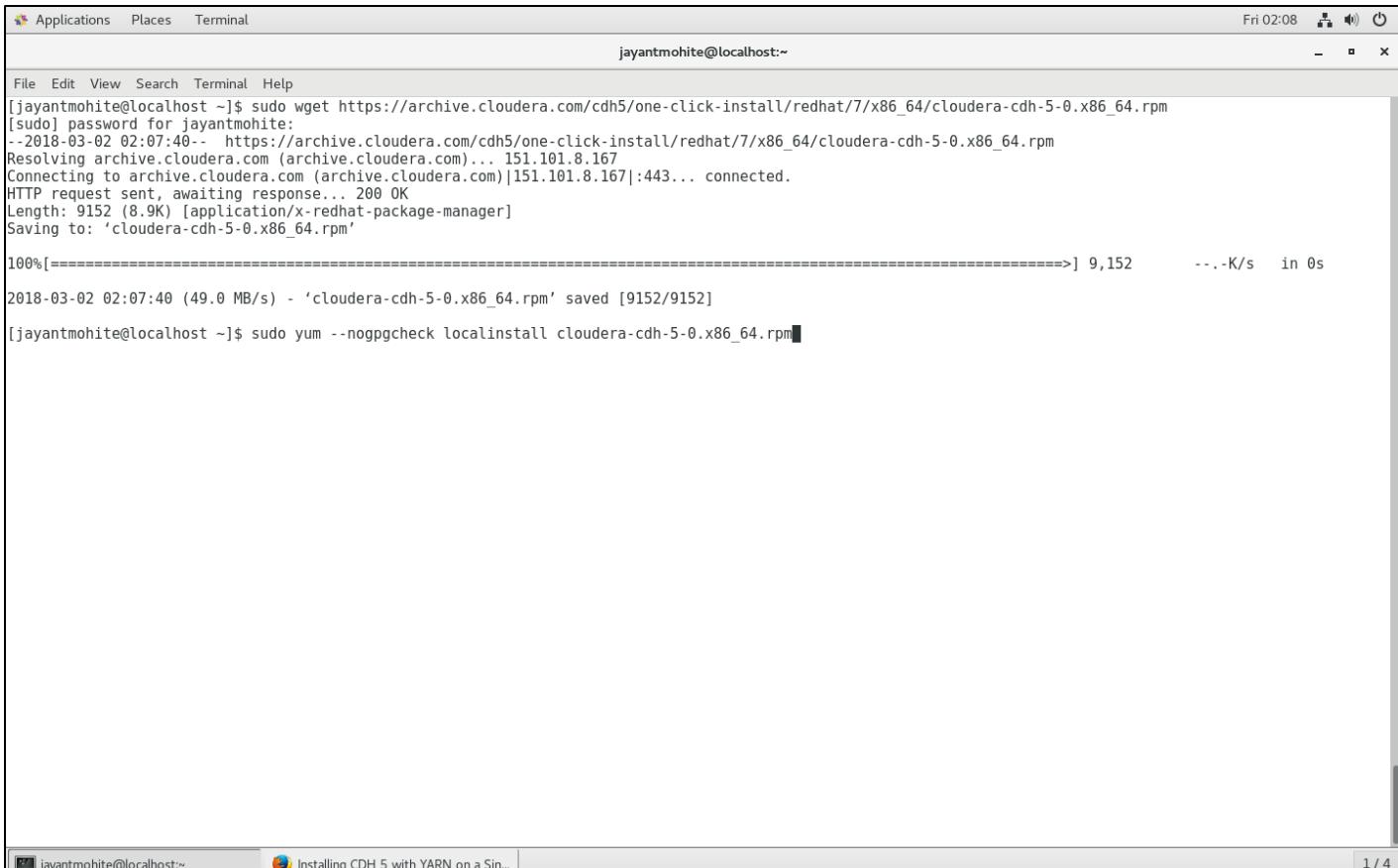


A screenshot of a Linux terminal window titled "Terminal". The window shows the command `sudo wget https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm` being run by the user "jayantmohite". The terminal is running on a Red Hat 7 system, as indicated by the prompt and the command itself. The password for the user is being entered.

```
[jayantmohite@localhost ~]$ sudo wget https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm  
[sudo] password for jayantmohite: [REDACTED]
```

Step 2:

Install the Downloaded Package



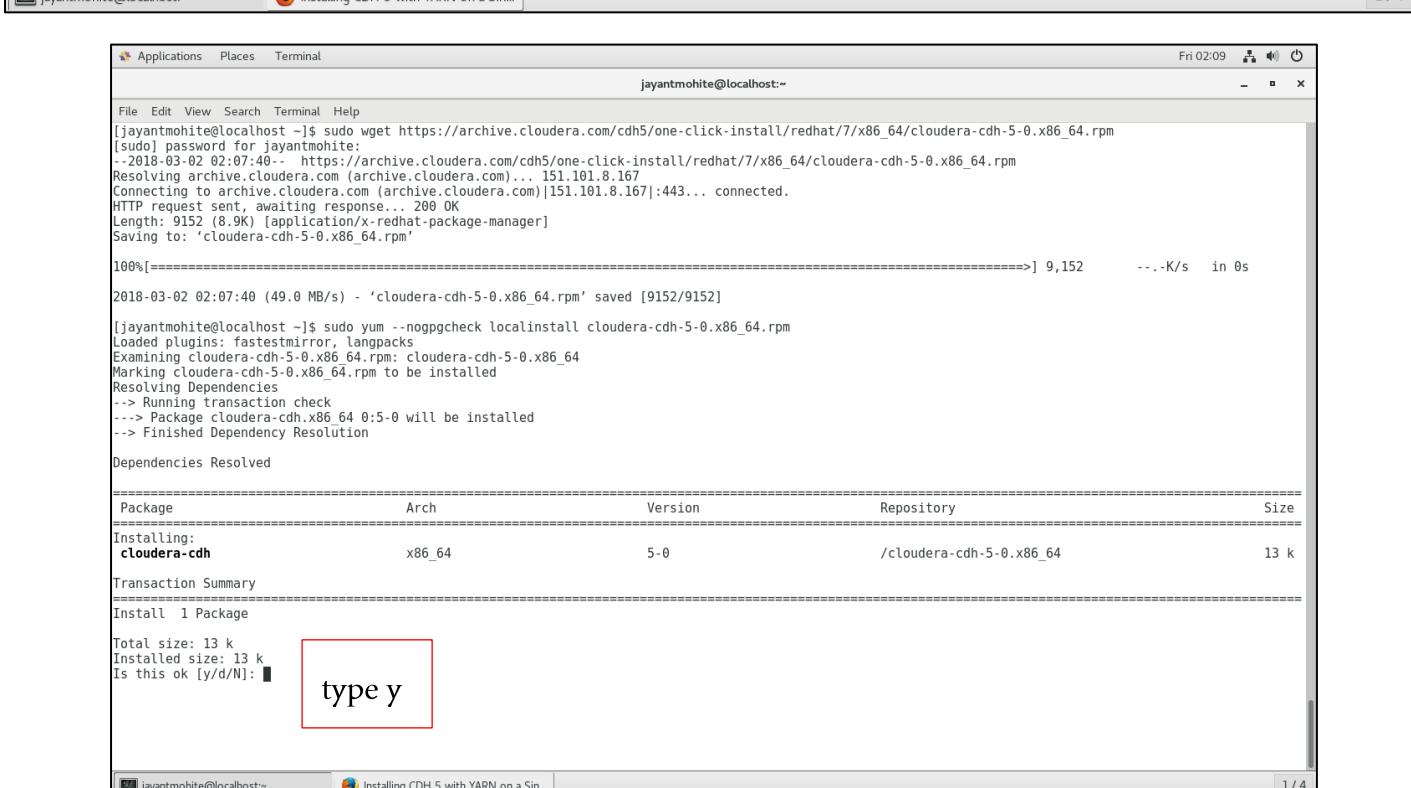
The screenshot shows a terminal window titled 'jayantmohite@localhost:~'. The user has run the command `sudo wget https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm`. The output shows the progress of the download, which completed at 02:07:40 on Friday, March 02, 2018. The file size is 9,152 KB.

```
[jayantmohite@localhost ~]$ sudo wget https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm
[sudo] password for jayantmohite:
--2018-03-02 02:07:40-- https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm
Resolving archive.cloudera.com (archive.cloudera.com)... 151.101.8.167
Connecting to archive.cloudera.com (archive.cloudera.com)|151.101.8.167|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9152 (8.9K) [application/x-redhat-package-manager]
Saving to: 'cloudera-cdh-5-0.x86_64.rpm'

100%[=====] 9,152      --.-K/s   in 0s

2018-03-02 02:07:40 (49.0 MB/s) - 'cloudera-cdh-5-0.x86_64.rpm' saved [9152/9152]

[jayantmohite@localhost ~]$ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
```



The screenshot shows a terminal window titled 'jayantmohite@localhost:~'. The user has run the command `sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm`. The output shows the package being installed, dependencies being resolved, and the transaction summary. A red box highlights the prompt 'Is this ok [y/N]:', with the letter 'y' typed into it.

```
[jayantmohite@localhost ~]$ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
[sudo] password for jayantmohite:
Loaded plugins: fastestmirror, langpacks
Examining cloudera-cdh-5-0.x86_64.rpm: cloudera-cdh-5-0.x86_64
Marking cloudera-cdh-5-0.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
--> Package cloudera-cdh.x86_64 0:5-0 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch       Version        Repository      Size
=====
Installing:
cloudera-cdh   x86_64     5-0           /cloudera-cdh-5-0.x86_64    13 k

Transaction Summary
=====
Install 1 Package

Total size: 13 k
Installed size: 13 k
Is this ok [y/N]: type y
```

Step 3:

Install Hadoop Generation 1 Framework

```
[jayantmohite@localhost ~]$ sudo yum install hadoop-0.20-conf-pseudo
```

```

File Edit View Search Terminal Help
jayantmohite@localhost:~ Fri 02:12
File Edit View Search Terminal Help
jayantmohite@localhost:~ [1 / 4]
jayantmohite@localhost:~ [1 / 4] Installing CDH 5 with YARN on a S... | 1 / 4

Processing Dependency: hadoop-hdfs-secondarynamenode = 2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 for package: hadoop-0.20-conf-pseudo-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Processing Dependency: hadoop = 2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 for package: hadoop-0.20-conf-pseudo-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Processing Dependency: hadoop-0.20-mapreduce-tasktracker = 2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 for package: hadoop-0.20-conf-pseudo-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Processing Dependency: hadoop-0.20-mapreduce-jobtracker = 2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 for package: hadoop-0.20-conf-pseudo-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Processing Dependency: hadoop-hdfs-namenode = 2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 for package: hadoop-0.20-conf-pseudo-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Running transaction check
Package hadoop.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 will be installed
cloudera-cdh5/filelists
Processing Dependency: zookeeper >= 3.4.0 for package: hadoop-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Processing Dependency: bigtop-utils >= 0.7 for package: hadoop-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Processing Dependency: paquet for package: hadoop-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Processing Dependency: /lib/lsb/init-functions for package: hadoop-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
mysql-connectors-community/x86_64/filelists_db
mysql-tools-community/x86_64/filelists_db
mysql5-community/x86_64/filelists_db
Processing Dependency: avro-libs for package: hadoop-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Package hadoop-0.20-mapreduce-jobtracker.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 will be installed
Processing Dependency: hadoop-0.20-mapreduce = 2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 for package: hadoop-0.20-mapreduce-jobtracker-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
15-1.cdh5.14.0.p0.47.el7.x86_64
Package hadoop-0.20-mapreduce-tasktracker.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 will be installed
Package hadoop-hdfs-datanode.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 will be installed
Processing Dependency: hadoop-hdfs = 2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 for package: hadoop-hdfs-datanode-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
17.x86_64
Package hadoop-hdfs-namenode.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 will be installed
Package hadoop-hdfs-secondarynamenode.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 will be installed
Running transaction check
Package avro-libs.noarch 0:1.7.6+cdh5.14.0+137-1.cdh5.14.0.p0.47.el7 will be installed
Package bigtop-utils.noarch 0:0.7.0+cdh5.14.0+0-1.cdh5.14.0.p0.47.el7 will be installed
Package hadoop-0.20-mapreduce.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 will be installed
Package hadoop-hdfs.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 will be installed
Processing Dependency: bigtop-jsvc for package: hadoop-hdfs-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64
Package parquet.noarch 0:1.5.0+cdh5.14.0+192-1.cdh5.14.0.p0.47.el7 will be installed
Processing Dependency: hadoop-client >= 2.6.0+cdh5.4.0 for package: parquet-1.5.0+cdh5.14.0+192-1.cdh5.14.0.p0.47.el7.noarch
Processing Dependency: parquet-format >= 2.1.0 for package: parquet-1.5.0+cdh5.14.0+192-1.cdh5.14.0.p0.47.el7.noarch
Processing Dependency: hadoop-yarn for package: parquet-1.5.0+cdh5.14.0+192-1.cdh5.14.0.p0.47.el7.noarch
Processing Dependency: hadoop-mapreduce for package: parquet-1.5.0+cdh5.14.0+192-1.cdh5.14.0.p0.47.el7.noarch

```

```

Applications Places Terminal Fri 02:12
jayantmohite@localhost:~ - x

File Edit View Search Terminal Help
--> Package parquet-format.noarch 0:2.1.0+cdh5.14.0+19-1.cdh5.14.0.p0.47.el7 will be installed
--> Package redhat-lsb-submod-security.x86_64 0:4.1-27.el7.centos.1 will be installed
--> Package spax.x86_64 0:1.5.2-13.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version          Repository      Size
=====
Installing:
hadoop-0.20-conf-pseudo        x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   9.7 k
Installing for dependencies:
avro-libs                     noarch    1.7.6+cdh5.14.0+137-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   94 M
bigtop-jsvc                   x86_64    0.6+cdh5.14.0+917-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   28 k
bigtop-utils                  noarch    0.7+cdh5.14.0+0-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   11 k
hadoop                        x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   75 M
hadoop-0.20-mapreduce         x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   84 M
hadoop-0.20-mapreduce-jobtracker x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.2 k
hadoop-0.20-mapreduce-tasktracker x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.1 k
hadoop-client                 x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   34 k
hadoop-hdfs                   x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   24 M
hadoop-hdfs-datanode          x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.2 k
hadoop-hdfs-namenode          x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.4 k
hadoop-hdfs-secondarynamenode x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.2 k
hadoop-mapreduce              x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   32 M
hadoop-yarn                   x86_64    2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   15 M
m4                             x86_64    1.4.16-10.el7                                base          256 k
parquet                       noarch    1.5.0+cdh5.14.0+192-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   76 M
parquet-format                noarch    2.1.0+cdh5.14.0+19-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   465 k
redhat-lsb-core                x86_64    4.1-27.el7.centos.1                            base          38 k
redhat-lsb-submod-security     x86_64    4.1-27.el7.centos.1                            base          15 k
spax                           x86_64    1.5.2-13.el7                                  base          260 k
zookeeper                      x86_64    3.4.5+cdh5.14.0+136-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   4.2 M

Transaction Summary
=====
Install 1 Package (+21 Dependent packages)

Total download size: 405 M
Installed size: 488 M
Is this ok [y/d/N]: ■

[jayantmohite@localhost:~] type y 1 with YARN on a SIn... 1 / 4

```

```

Applications Places Terminal Fri 02:14
jayantmohite@localhost:~ - x

File Edit View Search Terminal Help
Transaction Summary
=====
Install 1 Package (+21 Dependent packages)

Total download size: 405 M
Installed size: 488 M
Is this ok [y/d/N]: y
Downloading packages:
warning: /var/cache/yum/x86_64/7/cloudera-cdh5/packages/bigtop-jsvc-0.6.0+cdh5.14.0+917-1.cdh5.14.0.p0.47.el7.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID e8f86acd: NOKEY
Public key for bigtop-jsvc-0.6.0+cdh5.14.0+917-1.cdh5.14.0.p0.47.el7.x86_64.rpm is not installed
(1/22): bigtop-jsvc-0.6.0+cdh5.14.0+917-1.cdh5.14.0.p0.47.el7.noarch.rpm | 28 kB 00:00:00
(2/22): bigtop-utils-0.7.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.noarch.rpm | 11 kB 00:00:00
(3/22): hadoop-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 75 MB 00:00:33
(4/22): hadoop-0.20-conf-pseudo-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 9.7 kB 00:00:00
(5/22): avro-libs-1.7.6+cdh5.14.0+137-1.cdh5.14.0.p0.47.el7.noarch.rpm | 94 MB 00:00:41
(6/22): hadoop-0.20-mapreduce-jobtracker-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 5.2 kB 00:00:00
(7/22): hadoop-0.20-mapreduce-tasktracker-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 5.1 kB 00:00:00
(8/22): hadoop-client-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 34 kB 00:00:00
(9/22): hadoop-hdfs-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 24 kB 00:00:08
(10/22): hadoop-hdfs-datanode-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 5.2 kB 00:00:00
(11/22): hadoop-hdfs-namenode-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 5.4 kB 00:00:00
(12/22): hadoop-hdfs-secondarynamenode-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 5.2 kB 00:00:00
(13/22): hadoop-0.20-mapreduce-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 84 MB 00:00:30
(14/22): m4-1.4.16-10.el7.x86_64.rpm | 256 kB 00:00:01
(15/22): hadoop-mapreduce-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 32 MB 00:00:18
(16/22): hadoop-yarn-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 15 MB 00:00:03
(17/22): redhat-lsb-core-4.1-27.el7.centos.1.x86_64.rpm | 38 kB 00:00:00
(18/22): parquet-format-2.1.0+cdh5.14.0+19-1.cdh5.14.0.p0.47.el7.noarch.rpm | 465 kB 00:00:00
(19/22): spax-1.5.2-13.el7.x86_64.rpm | 260 kB 00:00:00
(20/22): redhat-lsb-submod-security-4.1-27.el7.centos.1.x86_64.rpm | 15 kB 00:00:01
(21/22): zookeeper-3.4.5+cdh5.14.0+136-1.cdh5.14.0.p0.47.el7.x86_64.rpm | 4.2 MB 00:00:01
(22/22): parquet-1.5.0+cdh5.14.0+192-1.cdh5.14.0.p0.47.el7.noarch.rpm | 76 MB 00:00:16

Total                                         4.7 MB/s | 405 MB 00:01:26
Retrieving key from https://archive.cloudera.com/cdh5/redhat/7/x86_64/cdh/RPM-GPG-KEY-cloudera
Importing GPG key 0xE8F86ACD:
Userid : Yun Maintainer <webmaster@cloudera.com>
Fingerprint: 5f14 d39e f068 laca 6f04 a443 f90c 0d8f e8f8 6acd
From : https://archive.cloudera.com/cdh5/redhat/7/x86_64/cdh/RPM-GPG-KEY-cloudera
Is this ok [y/N]: ■

[jayantmohite@localhost:~] type y 1 with YARN on a SIn... 1 / 4

```

```

Applications Places Terminal Fri 02:16 11:40 | 
jayantmohite@localhost:~ - x 
File Edit View Search Terminal Help 
Verifying : hadoop-client-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64 10/22 
Verifying : bigtop-utils-0.7.0+cdh5.14.0+0-1.cdh5.14.0.p0.47.el7.noarch 11/22 
Verifying : hadoop-hdfs-namenode-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64 12/22 
Verifying : hadoop-0.20-mapreduce-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64 13/22 
Verifying : bigtop-jsvc-0.6.0+cdh5.14.0+917-1.cdh5.14.0.p0.47.el7.x86_64 14/22 
Verifying : hadoop-hdfs-secondarynamenode-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64 15/22 
Verifying : hadoop-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.noarch 16/22 
Verifying : parquet-format-2.1.0+cdh5.14.0+19-1.cdh5.14.0.p0.47.el7 17/22 
Verifying : hadoop-hdfs-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64 18/22 
Verifying : zookeeper-3.4.5+cdh5.14.0+136-1.cdh5.14.0.p0.47.el7.x86_64 19/22 
Verifying : spax-1.5.2-13.el7.x86_64 20/22 
Verifying : redhat-lsb-core-4.1-27.el7.centos.1.x86_64 21/22 
Verifying : hadoop-mapreduce-2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7.x86_64 22/22 

Installed: 
hadoop-0.20-conf-pseudo.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 

Dependency Installed: 
avro-libs.noarch 0:1.7.6+cdh5.14.0+137-1.cdh5.14.0.p0.47.el7 
bigtop-jsvc.x86_64 0:0.6.0+cdh5.14.0+917-1.cdh5.14.0.p0.47.el7 
bigtop-utils.noarch 0:0.7.0+cdh5.14.0+0-1.cdh5.14.0.p0.47.el7 
hadoop.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-0.20-mapreduce.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-0.20-mapreduce-jobtracker.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-0.20-mapreduce-tasktracker.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-client.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-hdfs.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-hdfs-datanode.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-hdfs-namenode.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-hdfs-secondarynamenode.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-mapreduce.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
hadoop-yarn.x86_64 0:2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7 
m4.x86_64 0:1.4.16-10.el7 
parquet.noarch 0:1.5.0+cdh5.14.0+192-1.cdh5.14.0.p0.47.el7 
parquet-format.noarch 0:2.1.0+cdh5.14.0+19-1.cdh5.14.0.p0.47.el7 
redhat-lsb-core.x86_64 0:4.1-27.el7.centos.1 
redhat-lsb-submod-security.x86_64 0:4.1-27.el7.centos.1 
spax.x86_64 0:1.5.2-13.el7 
zookeeper.x86_64 0:3.4.5+cdh5.14.0+136-1.cdh5.14.0.p0.47.el7 

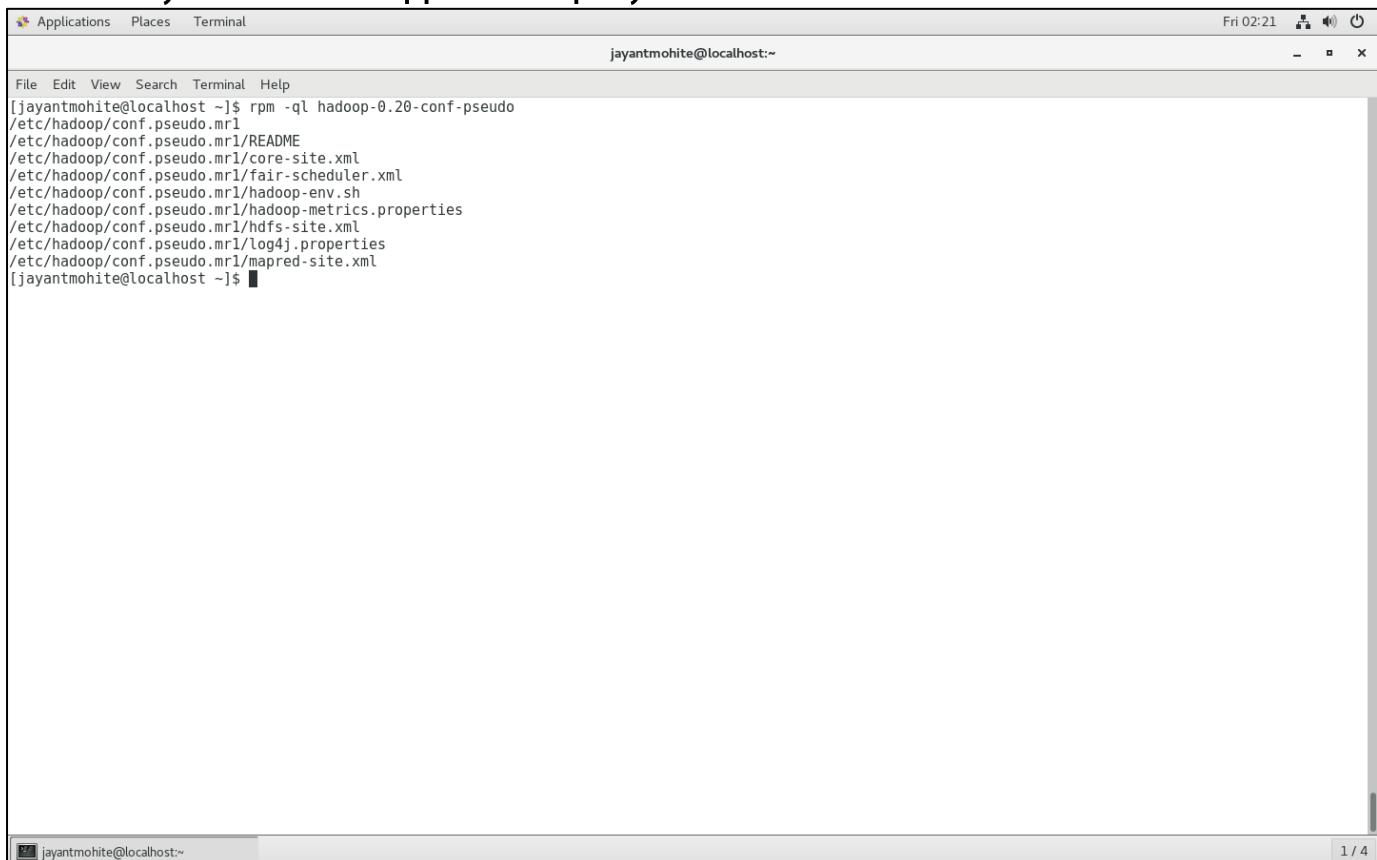
Complete! 
[jayantmohite@localhost ~]$ 

```

jayantmohite@localhost:~ 1 / 4

Step 4:

Verify if Execution Happened Properly



A screenshot of a Linux terminal window titled "Terminal". The window shows the command `rpm -ql hadoop-0.20-conf-pseudo` being run, followed by a list of files and directories extracted from the RPM package. The terminal window has a standard title bar with icons for Applications, Places, Terminal, and Help. The status bar at the bottom right shows the date and time as "Fri 02:21". The bottom right corner of the window frame also displays the number "1 / 4", indicating it's the first page of a multi-page document.

```
File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ rpm -ql hadoop-0.20-conf-pseudo
/etc/hadoop/conf.pseudo.mr1
/etc/hadoop/conf.pseudo.mr1/README
/etc/hadoop/conf.pseudo.mr1/core-site.xml
/etc/hadoop/conf.pseudo.mr1/fair-scheduler.xml
/etc/hadoop/conf.pseudo.mr1/hadoop-env.sh
/etc/hadoop/conf.pseudo.mr1/hadoop-metrics.properties
/etc/hadoop/conf.pseudo.mr1/hdfs-site.xml
/etc/hadoop/conf.pseudo.mr1/log4j.properties
/etc/hadoop/conf.pseudo.mr1/mapred-site.xml
[jayantmohite@localhost ~]$
```

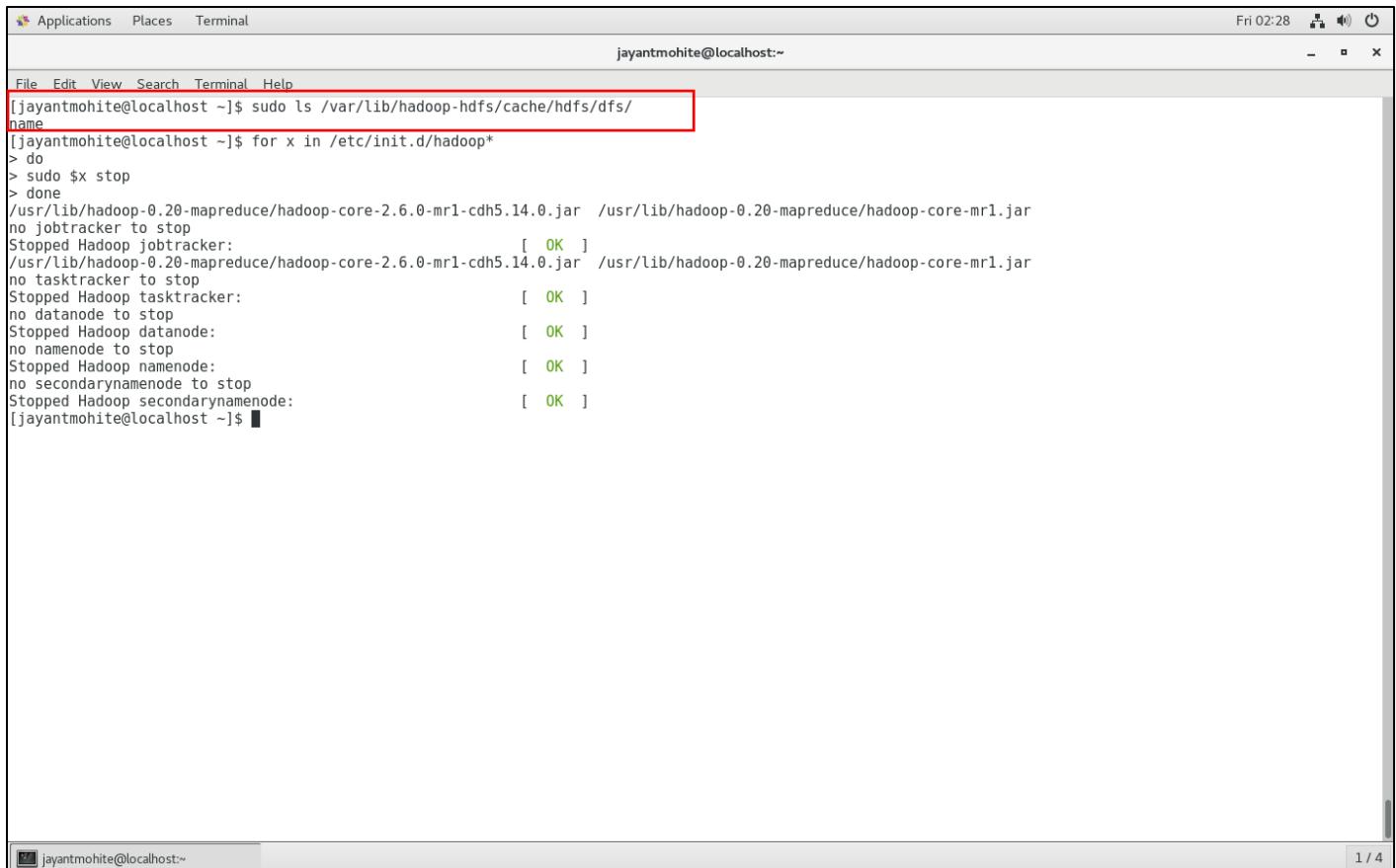
Step 5:

Format Namenode

```
[Applications Places Terminal] Fri 02:25 1/4
jayantmohite@localhost:~/
File Edit View Search Terminal Help
18/03/02 02:24:57 INFO namenode.FSNamesystem: HA Enabled: false
18/03/02 02:24:57 INFO namenode.FSNamesystem: Append Enabled: true
18/03/02 02:24:58 INFO util.GSet: Computing capacity for map INodeMap
18/03/02 02:24:58 INFO util.GSet: VM type      = 64-bit
18/03/02 02:24:58 INFO util.GSet: 1.0% max memory 966.7 MB = 9.7 MB
18/03/02 02:24:58 INFO util.GSet: capacity     = 2^20 = 1048576 entries
18/03/02 02:24:58 INFO namenode.FSDirectory: POSIX ACL inheritance enabled? false
18/03/02 02:24:58 INFO namenode.NameNode: Caching file names occurring more than 10 times
18/03/02 02:24:58 INFO snapshot.SnapshotManager: Loaded config captureOpenFiles: false, skipCaptureAccessTimeOnlyChange: false, snapshotDiffAllowSnapRootDescentant: true
18/03/02 02:24:58 INFO util.GSet: Computing capacity for map cachedBlocks
18/03/02 02:24:58 INFO util.GSet: VM type      = 64-bit
18/03/02 02:24:58 INFO util.GSet: 0.25% max memory 966.7 MB = 2.4 MB
18/03/02 02:24:58 INFO util.GSet: capacity     = 2^18 = 262144 entries
18/03/02 02:24:58 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 0.9990000128746033
18/03/02 02:24:58 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
18/03/02 02:24:58 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension = 0
18/03/02 02:24:58 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.window.num.buckets = 10
18/03/02 02:24:58 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.num.users = 10
18/03/02 02:24:58 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25
18/03/02 02:24:58 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
18/03/02 02:24:58 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
18/03/02 02:24:58 INFO util.GSet: Computing capacity for map NameNodeRetryCache
18/03/02 02:24:58 INFO util.GSet: VM type      = 64-bit
18/03/02 02:24:58 INFO util.GSet: 0.029999999329447746% max memory 966.7 MB = 297.0 KB
18/03/02 02:24:58 INFO util.GSet: capacity     = 2^15 = 32768 entries
18/03/02 02:24:58 INFO namenode.FSNamesystem: ACLs enabled? false
18/03/02 02:24:58 INFO namenode.FSNamesystem: XAttrs enabled? true
18/03/02 02:24:58 INFO namenode.FSNamesystem: Maximum size of an xattr: 16384
18/03/02 02:24:58 INFO namenode.FSImage: Allocated new BlockPoolId: BP-294660007-127.0.0.1-1519986298241
18/03/02 02:24:58 INFO common.Storage: Storage directory /var/lib/hadoop-hdfs/cache/hdfs/dfs/name has been successfully formatted.
18/03/02 02:24:58 INFO namenode.FSImageFormatProtobuf: Saving image file /var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current/fsimage.ckpt_00000000000000000000 using no compression
18/03/02 02:24:58 INFO namenode.FSImageFormatProtobuf: Image file /var/lib/hadoop-hdfs/cache/hdfs/dts/name/current/1simage.ckpt_00000000000000000000 of size 321 bytes saved in 0 seconds.
18/03/02 02:24:58 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
18/03/02 02:24:58 INFO util.ExitUtil: Exiting with status 0
18/03/02 02:24:58 INFO namenode.NameNode: SHUTDOWN_MSG:
*****Shutdown MSG: Shutting down NameNode at localhost/127.0.0.1*****
[jayantmohite@localhost ~]$
```

Step 6:

Stop All Hadoop Services



The screenshot shows a terminal window titled "Terminal" with the user "jayantmohite" at the prompt. The terminal displays a command-line session where the user runs a script to stop Hadoop services. The command is: `sudo ls /var/lib/hadoop-hdfs/cache/dfs/ | name`. This command lists the names of the HDFS data directories. Below this, the user runs a loop to stop each service: `for x in /etc/init.d/hadoop*; do sudo $x stop; done`. The output shows the stopping process for various Hadoop components: jobtracker, tasktracker, datanode, namenode, and secondarynamenode. Each component is shown as stopped with an "[OK]" status message. The session ends with the command `[jayantmohite@localhost ~]$`.

```
Applications Places Terminal Fri 02:28 jayantmohite@localhost:~ File Edit View Search Terminal Help [jayantmohite@localhost ~]$ sudo ls /var/lib/hadoop-hdfs/cache/dfs/ | name [jayantmohite@localhost ~]$ for x in /etc/init.d/hadoop*; do sudo $x stop; done /usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.14.0.jar /usr/lib/hadoop-0.20-mapreduce/hadoop-core-mr1.jar no jobtracker to stop Stopped Hadoop jobtracker: [ OK ] /usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.14.0.jar /usr/lib/hadoop-0.20-mapreduce/hadoop-core-mr1.jar no tasktracker to stop Stopped Hadoop tasktracker: [ OK ] no datanode to stop Stopped Hadoop datanode: [ OK ] no namenode to stop Stopped Hadoop namenode: [ OK ] no secondarynamenode to stop Stopped Hadoop secondarynamenode: [ OK ] [jayantmohite@localhost ~]$ 1 / 4
```

Step 7:

Start Only HDFS Services

```
[jayantmohite@localhost ~]$ for x in /etc/init.d/hadoop-hdfs-*; do sudo $x start; done
starting datanode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-datanode-localhost.localdomain.out
Started Hadoop datanode (hadoop-hdfs-datanode): [ OK ]
starting namenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-namenode-localhost.localdomain.out
Started Hadoop namenode: [ OK ]
starting secondarynamenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-secondarynamenode-localhost.localdomain.out
Started Hadoop secondarynamenode: [ OK ]
[jayantmohite@localhost ~]$
```

Overview 'localhost:8020' (active)

Started:	Fri Mar 02 02:29:08 -0800 2018
Version:	2.6.0-cdh5.14.0, r9b197d35839383c798c618ba917ccaa196a17699
Compiled:	Sat Jan 06 13:38:00 -0800 2018 by jenkins from Unknown
Cluster ID:	CID-9e44b830-795d-458c-80bc-350ffa52b814
Block Pool ID:	BP-294060007-127.0.0.1-1519986298241

Summary

Security is off.
Safemode is off.
1 files and directories, 0 blocks = 1 total filesystem object(s).
Heap Memory used 33.84 MB of 45.46 MB Heap Memory. Max Heap Memory is 966.69 MB.
Non Heap Memory used 42.46 MB of 43.25 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	17.7 GB
DFS Used:	4 KB (0%)

Namenode information - Mozilla Firefox

localhost:50070/dfshealth.html#tab-datanode

Datanode Information

In service: 1 Down: 0 Decommissioned: 0 Decommissioned & dead: 0 In Maintenance & dead: 0

Datanode usage histogram

Disk usage of each DataNode (%)

Bar chart showing disk usage distribution:

Usage Range (%)	Count
0-10	1

In operation

Show 25 entries Search:

Node	Last contact	Capacity	Blocks	Block pool used	Version
localhost (127.0.0.1:50010)	Fri Mar 02 02:32:33 -0800 2018	17.7 GB	0	4 KB (0%)	2.6.0-cdh5.14.0

Showing 1 to 1 of 1 entries Previous 1 Next

jayantmohite@localhost:~ | Namenode information - Mozilla Fir... 1 / 4

Applications Places Terminal Fri 02:33

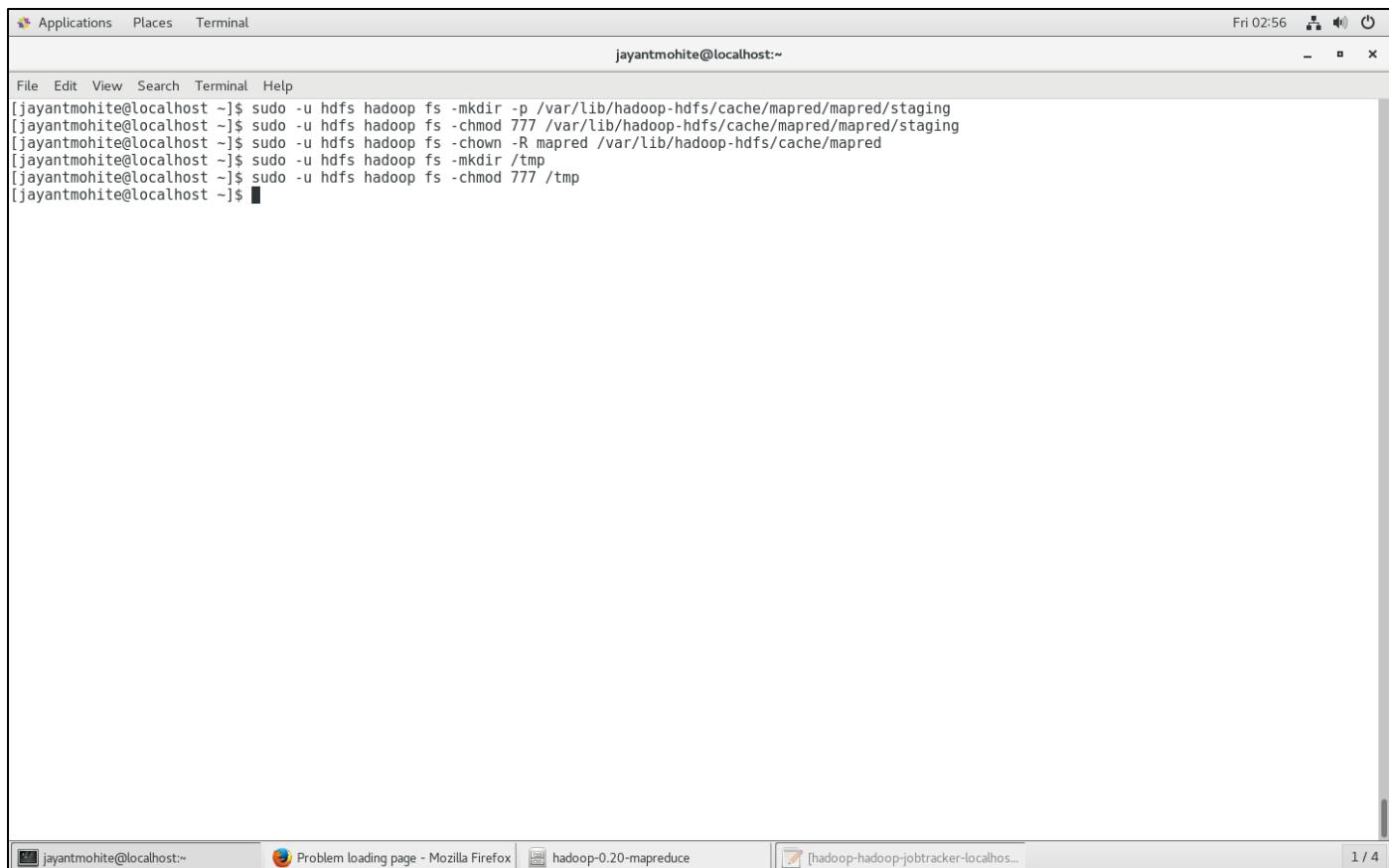
jayantmohite@localhost:~

```
File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ sudo ls /var/lib/hadoop-hdfs/cache/hdfs/dfs/
data name namensecondary
[jayantmohite@localhost ~]$
```

jayantmohite@localhost:~ | Namenode information - Mozilla Fir... 1 / 4

Step 8:

Create the Required Directory Structure with Proper Permissions



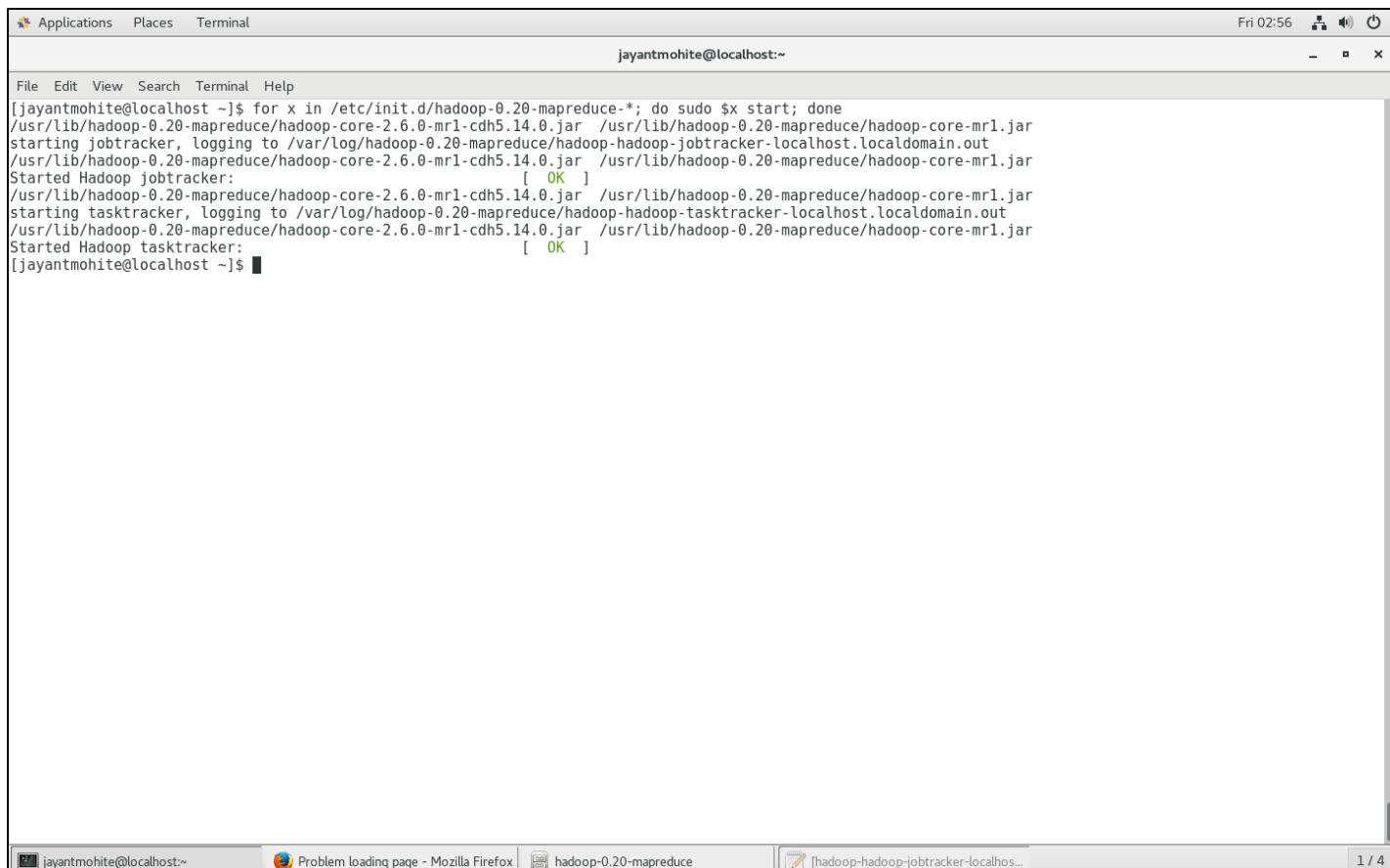
The screenshot shows a terminal window titled "jayantmohite@localhost:~". The user has run several commands to create the required directory structure for HDFS:

```
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -mkdir -p /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -chmod 777 /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -chown -R mapred /var/lib/hadoop-hdfs/cache/mapred
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -mkdir /tmp
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -chmod 777 /tmp
[jayantmohite@localhost ~]$
```

The terminal window is part of a larger interface with tabs at the bottom: "jayantmohite@localhost:~" (active), "Problem loading page - Mozilla Firefox", "hadoop-0.20-mapreduce", and "hadoop-hadoop-jobtracker-localhost...". A page number "1 / 4" is visible in the bottom right corner.

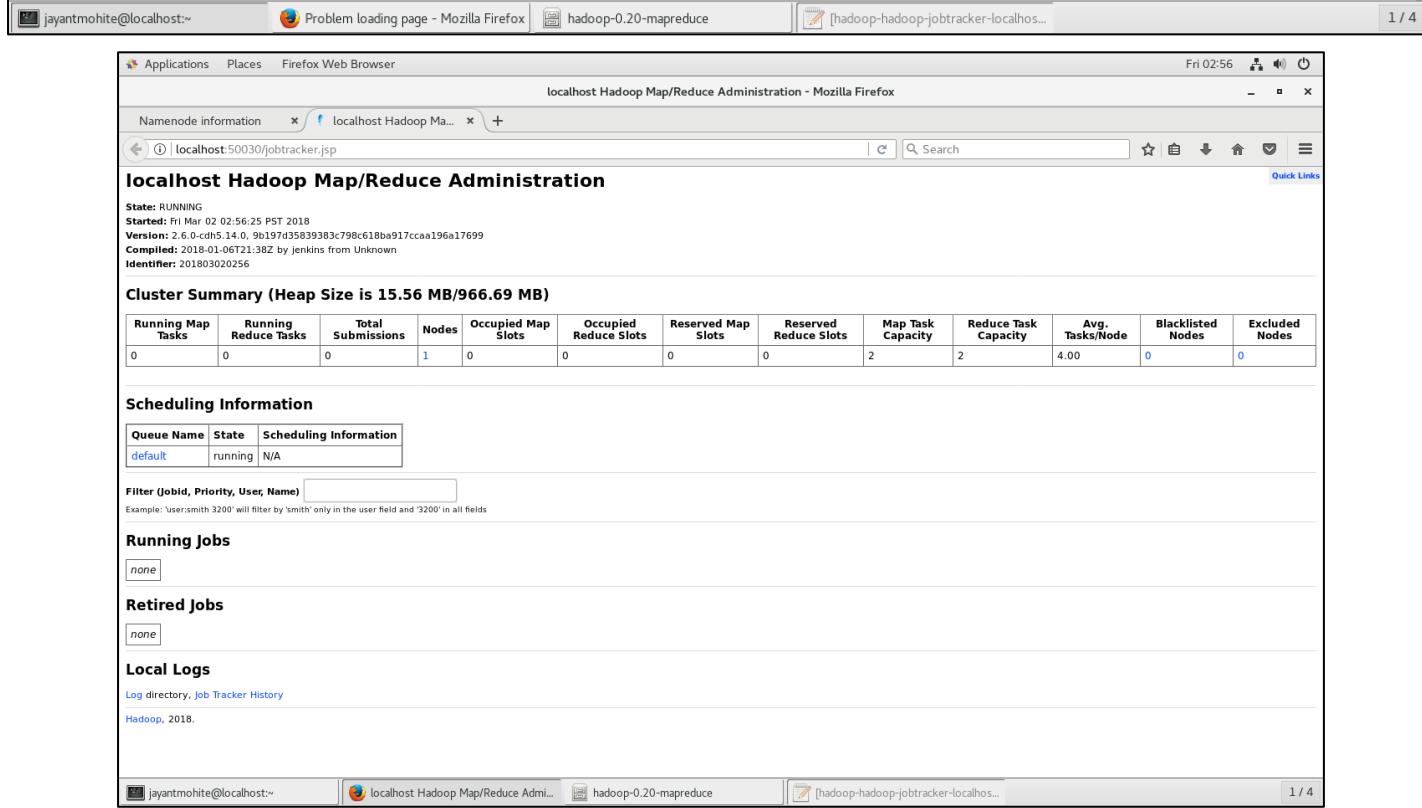
Step 9:

Start All Map Reduce Services



A screenshot of a terminal window titled "jayantmohite@localhost:~". The window shows the command `for x in /etc/init.d/hadoop-0.20-mapreduce-*; do sudo \$x start; done` being run. The output indicates the successful startup of the Jobtracker and Tasktracker services, both returning an "OK" status.

```
[jayantmohite@localhost ~]$ for x in /etc/init.d/hadoop-0.20-mapreduce-*; do sudo $x start; done
/usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.14.0.jar /usr/lib/hadoop-0.20-mapreduce/hadoop-core-mr1.jar
starting jobtracker, logging to /var/log/hadoop-0.20-mapreduce/hadoop-hadoop-jobtracker-localhost.localdomain.out
/usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.14.0.jar /usr/lib/hadoop-0.20-mapreduce/hadoop-core-mr1.jar
Started Hadoop jobtracker: [ OK ]
/usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.14.0.jar /usr/lib/hadoop-0.20-mapreduce/hadoop-core-mr1.jar
starting tasktracker, logging to /var/log/hadoop-0.20-mapreduce/hadoop-hadoop-tasktracker-localhost.localdomain.out
/usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.14.0.jar /usr/lib/hadoop-0.20-mapreduce/hadoop-core-mr1.jar
Started Hadoop tasktracker: [ OK ]
[jayantmohite@localhost ~]$
```



A screenshot of a Firefox browser window titled "localhost Hadoop Map/Reduce Administration - Mozilla Firefox". The page displays cluster summary information, scheduling information, running jobs, retired jobs, and local logs. The cluster summary table shows the following data:

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Excluded Nodes
0	0	0	1	0	0	0	0	2	2	4.00	0	0

The "Running Jobs" section shows a single entry labeled "none". The "Retired Jobs" section also shows a single entry labeled "none".

Step 10:

Create Directory for User with Proper Permissions

Step 11:

Create Sample Input

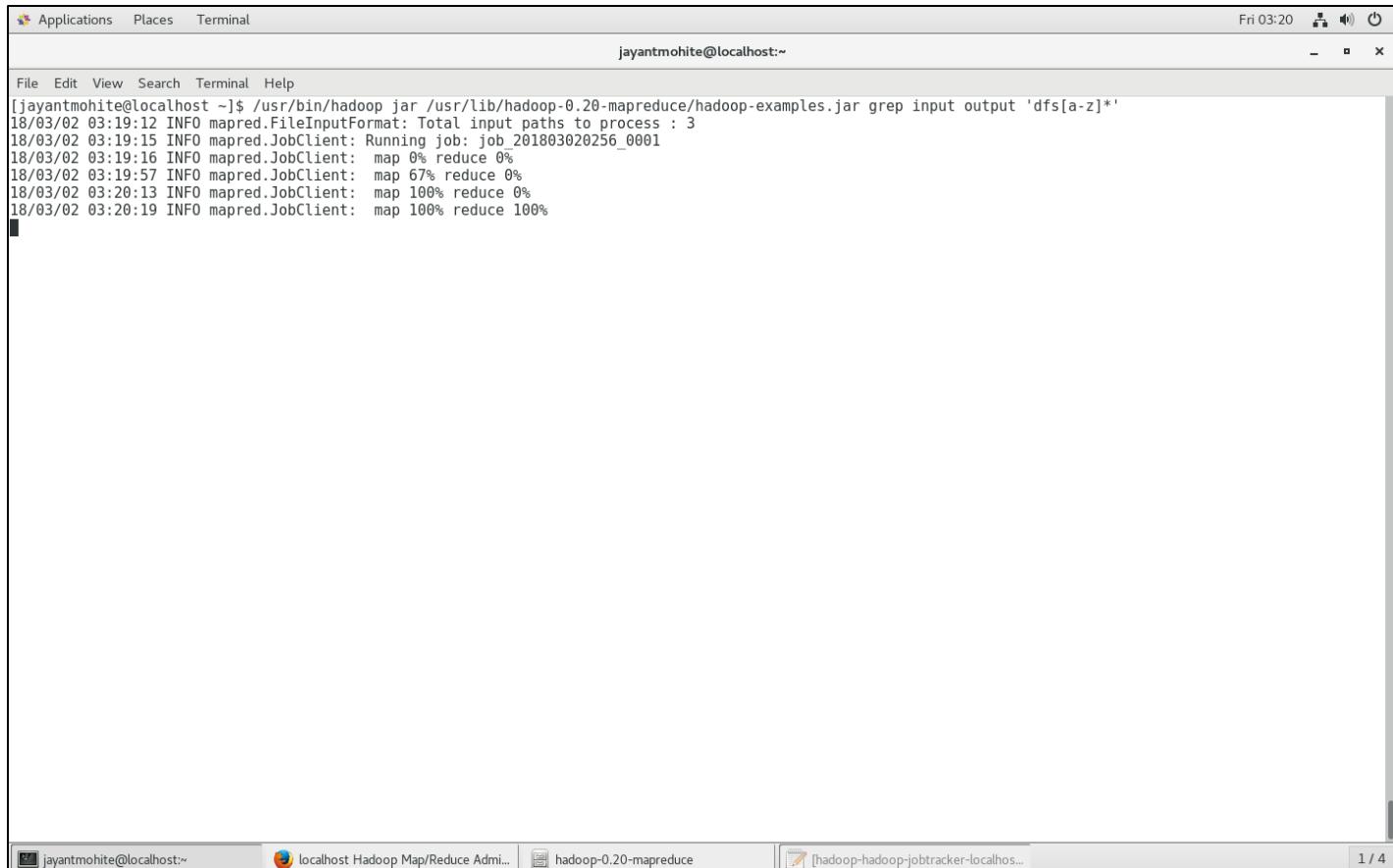
The screenshot shows a terminal window titled "jayantmohite@localhost:~". The terminal displays the following command sequence and output:

```
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -mkdir -p /user/jayantmohite
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -chown jayantmohite /user/jayantmohite
[jayantmohite@localhost ~]$ hadoop fs -mkdir input
[jayantmohite@localhost ~]$ hadoop fs -put /etc/hadoop/conf/*-site.xml input
[jayantmohite@localhost ~]$ hadoop fs -ls
Found 1 items
drwxr-xr-x - jayantmohite supergroup 0 2018-03-02 02:59 input
[jayantmohite@localhost ~]$ hadoop fs -ls input/
Found 3 items
-rw-r--r-- 1 jayantmohite supergroup 2133 2018-03-02 02:59 input/core-site.xml
-rw-r--r-- 1 jayantmohite supergroup 1875 2018-03-02 02:59 input/hdfs-site.xml
-rw-r--r-- 1 jayantmohite supergroup 582 2018-03-02 02:59 input/mapred-site.xml
[jayantmohite@localhost ~]$
```

The terminal window is part of a desktop environment, as evidenced by the application menu bar at the top and the taskbar at the bottom. The taskbar shows other open applications: "localhost Hadoop Map/Reduce Admin...", "hadoop-0.20-mapreduce", and "hadoop-hadoop-jobtracker-localhos...".

Step 12:

Execute Sample Program



A screenshot of a Linux desktop environment showing a terminal window. The terminal title is "jayantmohite@localhost:~". The window contains the following command and its output:

```
[jayantmohite@localhost ~]$ /usr/bin/hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar grep input output 'dfs[a-z]*'
18/03/02 03:19:12 INFO mapred.FileInputFormat: Total input paths to process : 3
18/03/02 03:19:15 INFO mapred.JobClient: Running job: job_201803020256_0001
18/03/02 03:19:16 INFO mapred.JobClient: map 0% reduce 0%
18/03/02 03:19:57 INFO mapred.JobClient: map 67% reduce 0%
18/03/02 03:20:13 INFO mapred.JobClient: map 100% reduce 0%
18/03/02 03:20:19 INFO mapred.JobClient: map 100% reduce 100%
```

The terminal window has a standard Linux interface with a menu bar (Applications, Places, Terminal) and a toolbar. Below the terminal window, the desktop taskbar shows other open applications: "localhost Hadoop Map/Reduce Admin...", "hadoop-0.20-mapreduce", and "[hadoop-hadoop-jobtracker-localhos...]".

Step 13:

Install Sqoop

```

File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ sudo yum install sqoop
[sudo] password for jayantmohite:
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: ftp.iitm.ac.in
 * extras: ftp.iitm.ac.in
 * updates: ftp.iitm.ac.in
Resolving Dependencies
--> Running transaction check
--> Package sqoop.noarch 0:1.4.6+cdh5.14.0+127-1.cdh5.14.0.p0.47.el7 will be installed
--> Processing Dependency: kite for package: sqoop-1.4.6+cdh5.14.0+127-1.cdh5.14.0.p0.47.noarch
--> Running transaction check
--> Package kite.noarch 0:1.0.0+cdh5.14.0+146-1.cdh5.14.0.p0.47.el7 will be installed
--> Processing Dependency: sentry >= 1.3.0+cdh5.1.0 for package: kite-1.0.0+cdh5.14.0+146-1.cdh5.14.0.p0.47.el7.noarch
--> Processing Dependency: solr >= 4.4.0+cdh5.1.0 for package: kite-1.0.0+cdh5.14.0+146-1.cdh5.14.0.p0.47.el7.noarch
--> Running transaction check
--> Package sentry.noarch 0:1.5.1+cdh5.14.0+432-1.cdh5.14.0.p0.47.el7 will be installed
--> Processing Dependency: hive-jdbc >= 1.1.0+cdh5.4.0 for package: sentry-1.5.1+cdh5.14.0+432-1.cdh5.14.0.p0.47.el7.noarch
--> Processing Dependency: hive >= 1.1.0+cdh5.4.0 for package: sentry-1.5.1+cdh5.14.0+432-1.cdh5.14.0.p0.47.el7.noarch
--> Package solr.noarch 0:4.10.3+cdh5.14.0+522-1.cdh5.14.0.p0.47.el7 will be installed
--> Processing Dependency: bigtop-tomcat for package: solr-4.10.3+cdh5.14.0+522-1.cdh5.14.0.p0.47.el7.noarch
--> Running transaction check
--> Package bigtop-tomcat.noarch 0:0.7.0+cdh5.14.0+0-1.cdh5.14.0.p0.47.el7 will be installed
--> Package hive.noarch 0:1.1.0+cdh5.14.0+1330-1.cdh5.14.0.p0.47.el7 will be installed
--> Package hive-jdbc.noarch 0:1.1.0+cdh5.14.0+1330-1.cdh5.14.0.p0.47.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version            Repository      Size
=====
Installing:
sqoop             noarch   1.4.6+cdh5.14.0+127-1.cdh5.14.0.p0.47.el7      cloudera-cdh5    18 M
Installing for dependencies:
bigtop-tomcat     noarch   0.7.0+cdh5.14.0+0-1.cdh5.14.0.p0.47.el7      cloudera-cdh5    7.4 M
hive              noarch   1.1.0+cdh5.14.0+1330-1.cdh5.14.0.p0.47.el7      cloudera-cdh5    47 M
hive-jdbc         noarch   1.1.0+cdh5.14.0+1330-1.cdh5.14.0.p0.47.el7      cloudera-cdh5    45 M
kite              noarch   1.0.0+cdh5.14.0+146-1.cdh5.14.0.p0.47.el7      cloudera-cdh5    104 M
sentry             noarch   1.5.1+cdh5.14.0+432-1.cdh5.14.0.p0.47.el7      cloudera-cdh5    57 M
solr              noarch   4.10.3+cdh5.14.0+522-1.cdh5.14.0.p0.47.el7      cloudera-cdh5    75 M

Transaction Summary
=====
Install 1 Package (+6 Dependent packages)

Total download size: 354 M
Installed size: 447 M
Is this ok [y/d/N]:
```

type y

Step 14:

Install Pig

```
[jayantmohite@localhost ~]$ sudo yum install pig
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: ftp.iitm.ac.in
 * extras: ftp.iitm.ac.in
 * updates: ftp.iitm.ac.in
Resolving Dependencies
--> Running transaction check
--> Package pig.noarch 0:0.12.0+cdh5.14.0+112-1.cdh5.14.0.p0.47.el7 will be installed
--> Processing Dependency: hive-hcatalog for package: pig-0.12.0+cdh5.14.0+112-1.cdh5.14.0.p0.47.el7.noarch
--> Running transaction check
--> Package hive-hcatalog.noarch 0:1.1.0+cdh5.14.0+1330-1.cdh5.14.0.p0.47.el7 will be installed
--> Finished Dependency Resolution
```

```
[jayantmohite@localhost ~]$ sudo yum install pig
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: ftp.iitm.ac.in
 * extras: ftp.iitm.ac.in
 * updates: ftp.iitm.ac.in
Resolving Dependencies
--> Running transaction check
--> Package pig.noarch 0:0.12.0+cdh5.14.0+112-1.cdh5.14.0.p0.47.el7 will be installed
--> Processing Dependency: hive-hcatalog for package: pig-0.12.0+cdh5.14.0+112-1.cdh5.14.0.p0.47.el7.noarch
--> Running transaction check
--> Package hive-hcatalog.noarch 0:1.1.0+cdh5.14.0+1330-1.cdh5.14.0.p0.47.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

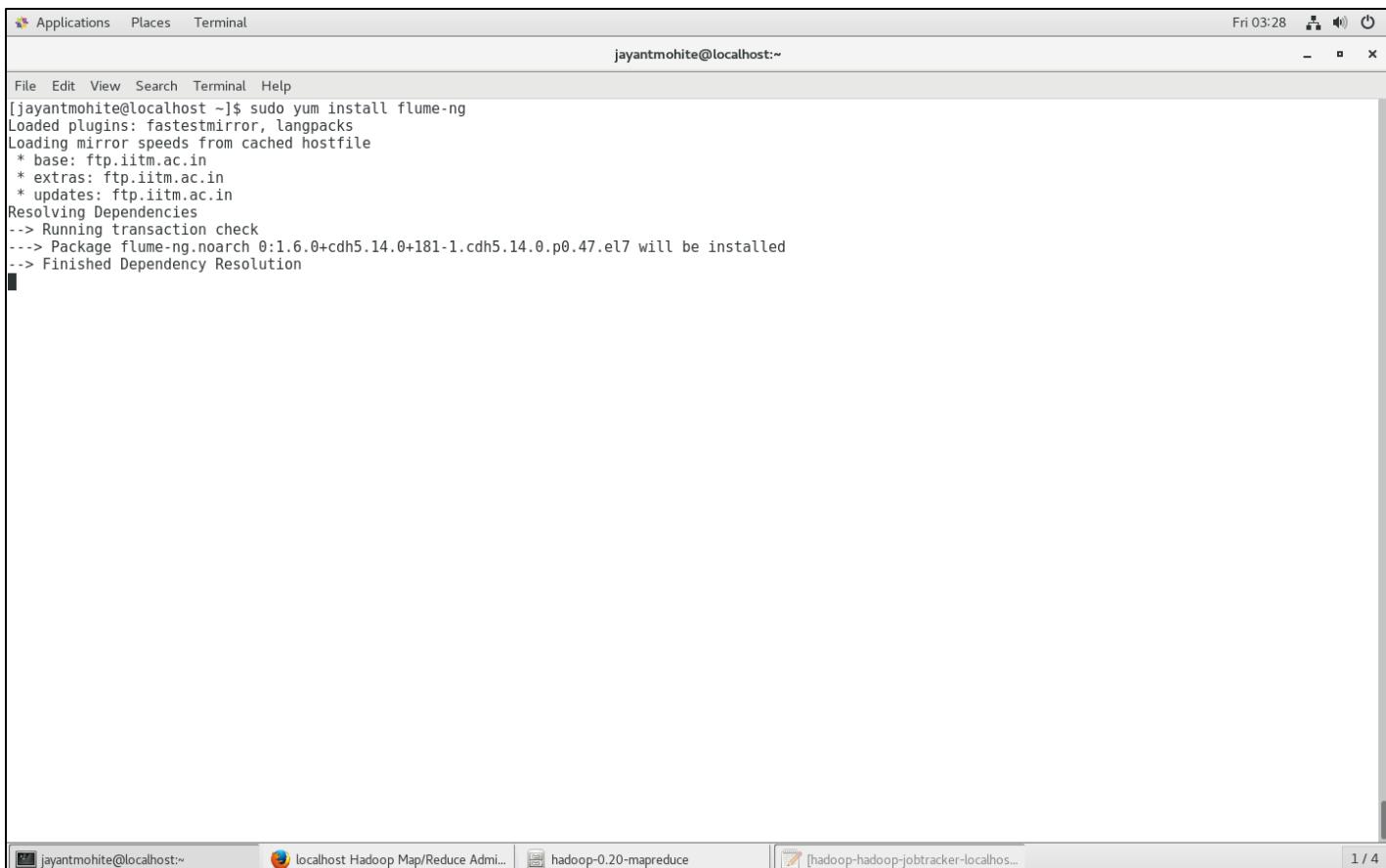
=====
Package           Arch      Version            Repository      Size
=====
Installing:
pig               noarch   0.12.0+cdh5.14.0+112-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   58 M
Installing for dependencies:
hive-hcatalog    noarch   1.1.0+cdh5.14.0+1330-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   393 k

Transaction Summary
=====
Install 1 Package (+1 Dependent package)

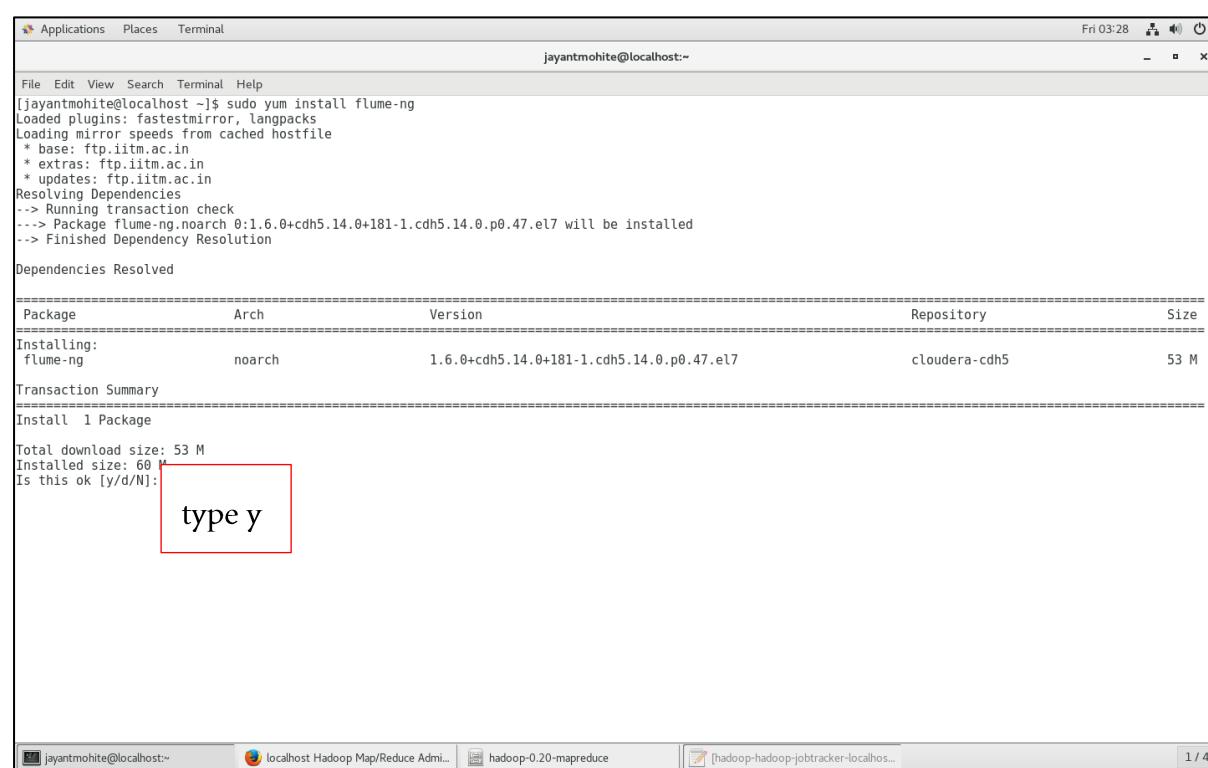
Total download size: 59 M
Installed size: 124 M
Is this ok [y/d/N]: type y
```

Step 15:

Install Flume-ng



```
[jayantmohite@localhost ~]$ sudo yum install flume-ng
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: ftp.iitm.ac.in
 * extras: ftp.iitm.ac.in
 * updates: ftp.iitm.ac.in
Resolving Dependencies
--> Running transaction check
--> Package flume-ng.noarch 0:1.6.0+cdh5.14.0+181-1.cdh5.14.0.p0.47.el7 will be installed
--> Finished Dependency Resolution
```



```
[jayantmohite@localhost ~]$ sudo yum install flume-ng
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: ftp.iitm.ac.in
 * extras: ftp.iitm.ac.in
 * updates: ftp.iitm.ac.in
Resolving Dependencies
--> Running transaction check
--> Package flume-ng.noarch 0:1.6.0+cdh5.14.0+181-1.cdh5.14.0.p0.47.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

---- Package Arch Version Repository Size
Installing:
flume-ng noarch 1.6.0+cdh5.14.0+181-1.cdh5.14.0.p0.47.el7 cloudera-cdh5 53 M

Transaction Summary
---- Install 1 Package

Total download size: 53 M
Installed size: 60 M
Is this ok [y/d/N]: type y
```

Now your Hadoop Cluster is ready and you can start performing your practice tasks on it.

06. Introduction to Hadoop Architecture

As discussed in the previous chapter, Hadoop is a framework built using specifications of the [Google File System](#) to overcome the problems of Big Data in a distributed environment under an open source license of [Apache Foundation](#).

What is a framework?

A framework is a platform for designing applications that gives us four things

- List of functions that can be executed on top of the framework.
- The process followed for execution of each function.
- The Library consisting of all base classes, pre-defined functions and other instructions.
- The list of services.

So, if Hadoop is a framework it should also provide us the four sets mentioned above. Let us try to evaluate Hadoop as a framework in this section.

If we have a look from a high-level perspective at Hadoop as a framework, then this is what we would be able to visualize

List of functionalities that can be performed using this framework:

Hadoop provides us two capabilities, one of which is the ability to store Big Data and the other is the ability to process this stored Big Data.

The process followed under execution:

For storing the data, Hadoop uses something called as **HDFS** and the model used for processing the data is called as **Map Reduce**.

List of library functions:

Hadoop being built in Java, all the library functions are bundled into java jar files and there are three xml files namely **core-site.xml**, **hdfs-site.xml** and **mapred-site.xml**, available for providing user defined values to some configurable properties under the Hadoop stack.

List of services:

There are five services available in Hadoop namely **Namenode**, **Datanode**, **Secondary Namenode**, **Job Tracker** and **Task Tracker**, which take care of providing a background platform for Hadoop users to take advantage of all functionalities offered under the framework.

However, this is not at all giving us a detailed picture, which would help us understand what Hadoop exactly is. So let's deep dive in each of its two functionalities and try to evaluate them as a framework, to understand Hadoop in more details.

Hadoop Distributed File System

Popularly known as the HDFS, the hadoop distributed file system serves as a storage component of Hadoop. Trying to visualize HDFS as an embedded framework in the larger set of the hadoop framework, this is what we can conclude to,

List of functionalities that can be performed using this framework:

HDFS can be visualized as the storage unit of the Hadoop framework, which stores structured as well as unstructured data in text format. For storing data, which is varied in nature, it makes use of flat file system and for handling large amount of data it works on top of distributed environment. Apart from this, in order to maintain the integrity of data, HDFS is designed as the [Write Once Read Many](#) kind of a file system.

The process followed under execution:

For storage, HDFS breaks a file into smaller chunks that are replicated and distributed across multiple machines in order to provide scalability and preserve availability of data in addition to providing a faster parallel processing execution engine.

List of library functions:

Hadoop being built in Java, all the library functions related to HDFS are bundled into java jar files and [hdfs-site.xml](#) is available for providing user defined values to some configurable properties under the Hadoop stack related to storage or you can say related to HDFS.

List of services:

For HDFS, the available services are:

Internal User = **hdfs** (here it is just a name given to the internal user and does not relate to the abbreviation of Hadoop Distributed File System)

Master Service = **Namenode, Secondary Namenode**

Slave Service = **Datanode**

Map Reduce

Map Reduce serves as a **programming model** of Hadoop. Just like there are four basic programming models in mathematics namely **addition, subtraction, multiplication** and **division**; there is only one programming model in hadoop called as Map Reduce. Trying to visualize Map Reduce as an embedded framework in the larger set of the hadoop framework, this is what we can conclude to,

List of functionalities that can be performed using this framework:

Map Reduce can process the data, which is stored in HDFS. Which makes it clear that for any data to be processed under this programming model, it has to be present in HDFS, which is Hadoop' s own storage unit.

The process followed under execution:

Map Reduce works in two stages.

Stage one (**map phase**):

HDFS break an input file into chunks for providing data availability and this is used by Map Reduce to provide parallel processing. In the first phase, a function called as **map ()** is executed independently on every chunk of the input file, where every execution generates an output file stored in local file system on the memory. The user writes the logic in this function. However, this function expects input in key value pair format and provides output in **key value pair** format itself.

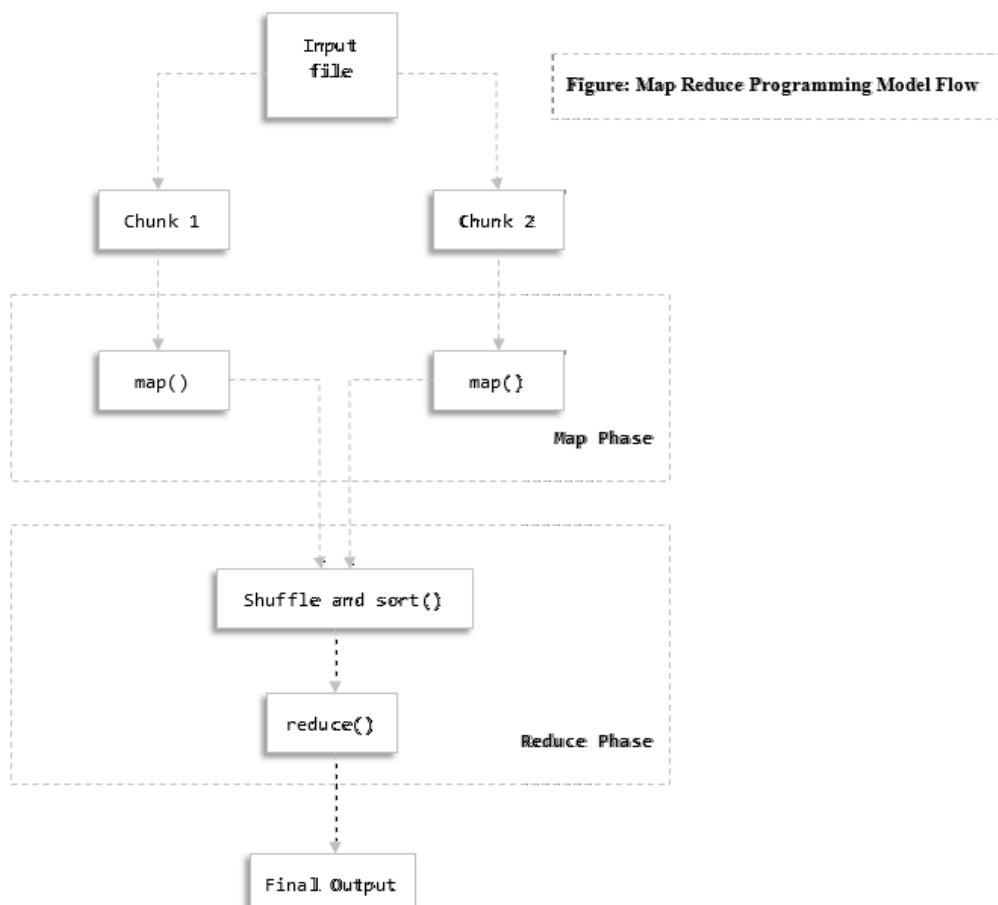
Stage tow (**reduce phase**):

In this phase which is the second and final phase of the Map Reduce programming model, a default function called as **shuffle and sort** combines all output files generated by the map executions on the input file. The resultant output stored in local file system is one file containing a sorted output on the keys with an iterator of all values associated with every key.

The output of the shuffle and sort is then provided internally to a function called as **reduce ()** which does the final execution to get the desired result. The user writes the logic for this function. However, this function expects input in **pairs of key and iterator of values** (output of shuffle and sort function) and generates output in **pairs of key and value**. The output of the reduce () function is stored in HDFS and marks the completion of the execution process.

List of library functions:

Hadoop being built in Java, all the library functions related to Map Reduce are bundled into java jar files and *mapred-site.xml* is available for providing user defined values to some configurable properties under the Hadoop stack related to processing or you can say related to Map Reduce.



List of services:

For Map Reduce, the available services are:

Internal User = [mapred](#)

Master Service = [Job Tracker](#)

Slave Service = [Task Tracker](#)

Apart from all the above stuff there is one more configuration file that does not directly deal with either of the components of Hadoop, rather deals with the entire architecture as a whole and that is called as [core-site.xml](#).

With this, we now have a fair bit of idea of the Hadoop framework let us try to build the block architecture for the same. However, before that let us deal with another important concept in the series of Hadoop generation one which is called as the Hadoop Distributions.

Hadoop Distributions

There are many Distributions of Hadoop available out of which there are many which are customized for specific requirements as well. Few of those, which I recommend or few, which I have been using since a long time, know, are the three top distributions namely [Apache Hadoop](#), [Cloudera Distribution of Hadoop](#) and [Hortonworks Data Platform](#).

When Hadoop first came in, it was a composition of only two components i.e. [HDFS](#) and [Map Reduce](#). However, people in the community soon realized that the Java dependency is turning out to be a constraint, restraining people from using Hadoop and this was a hurdle in solving the problem of Big Data. Therefore, tools were developed which will abstract the complexity of Java at the backend and will give a clean and simple frontend comprised of simple English like commands. These tools were called as [Eco System Tools of Hadoop](#). The invent of this tools significantly reduce the amount of coding required for working with Hadoop and increased its popularity as well. Now 300 lines of code in java Map Reduce was reduced to just around five lines of English like commands. However, the significant drawback of these tools was that they were application specific. Therefore, any company, which was working on a big project using Hadoop had to independently download and manage all tools separately which was kind of increasing the overhead and this, was not desirable. Therefore, few companies like Cloudera came up with packages, which bundled the core components of Hadoop with some popular Eco System tools. These packages became popular due to the ease of user experience they provided and are called as the [Hadoop Distributions](#). There are around [23 Hadoop Distributions](#) available as of date including few

customized one from [IBM](#) and [HP](#). But the most popular ones are from [Apache](#), [Hortonworks](#), [Cloudera](#) and [MapR](#).

It is important to understand that Hadoop is same in all distributions. What is different is the composition of different components together and the user experience. Let us have a brief look at some of few of these tools

Apache Hadoop

Comprises of:

HDFS

Map Reduce

Also called as [Core Hadoop](#)

License: Open Source

Underlying Operating System: Linux

Popular Feature: [First Hadoop Edition](#)

Cloudera's Distribution of Hadoop

Comprises of:

Core Hadoop

Popular Eco System Tools

Proprietary Tools

Cloudera Manager

HUE

Also called as CDH

License: Enterprise Edition

Underlying Operating System: Linux

Popular Feature: **First enterprise edition of Hadoop**

Hortonwork's Data Platform

Comprises of:

Core Hadoop

Popular Eco System Tools

Proprietary Tools

Ambari

HUE Integration

Also called as HDP

License: Enterprise

Underlying Operating System: [Windows](#)

Popular As: First deployment of Hadoop on Windows

Hadoop Modes of Installation

In hadoop terminology and server, which has the installation of Hadoop or has the two components of Hadoop namely HDFS and Map Reduce is called as a **node** and a group of Hadoop Nodes is called as a **Hadoop Cluster**. Based on the type of use case a Hadoop cluster can have from one to 1000s of nodes. Hadoop can be installed in two modes

Single Node Cluster or **Pseudo Distributed Mode**

Multi Node Cluster or **Distributed Mode**

Single Node Cluster

In this mode, all services of the master as well as slave are installed on the same server. It has only one instance of the master as well as the slaves. This installation is good for testing and learning but cannot be used in production environments as it works of vertical scaling. This can be viewed as a simulation model of the actual Hadoop Cluster.

Multi Node Cluster

In this mode, the master services are installed on separate servers. If it is a large deployment, you will have three master servers with separate installations of Namenode, Job Tracker and Secondary Namenode. If it is a medium scale deployment, you will have installations of Namenode and Job Tracker on the same server and installation of Secondary Namenode on a separate server. If it is a small-scale deployment, then you will have installations of all three Namenode, Job Tracker and Secondary Namenode on the same server.

Does not matter which kind of deployment it is you will have multiple slave servers with every server having installations of both Datanode as well as Task Tracker. These two services operate as a pair locally.

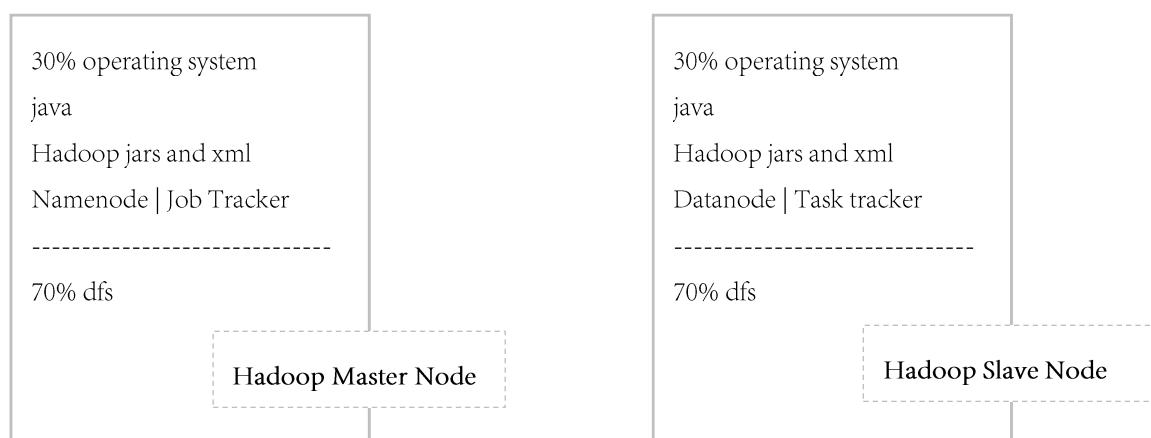
Hadoop Architecture

Now that we have a good knowledge of Hadoop Framework, Its Distributions and Modes of Installations, let us have a look at the architecture. In this section, we will try to logically install a Hadoop Cluster from scratch.

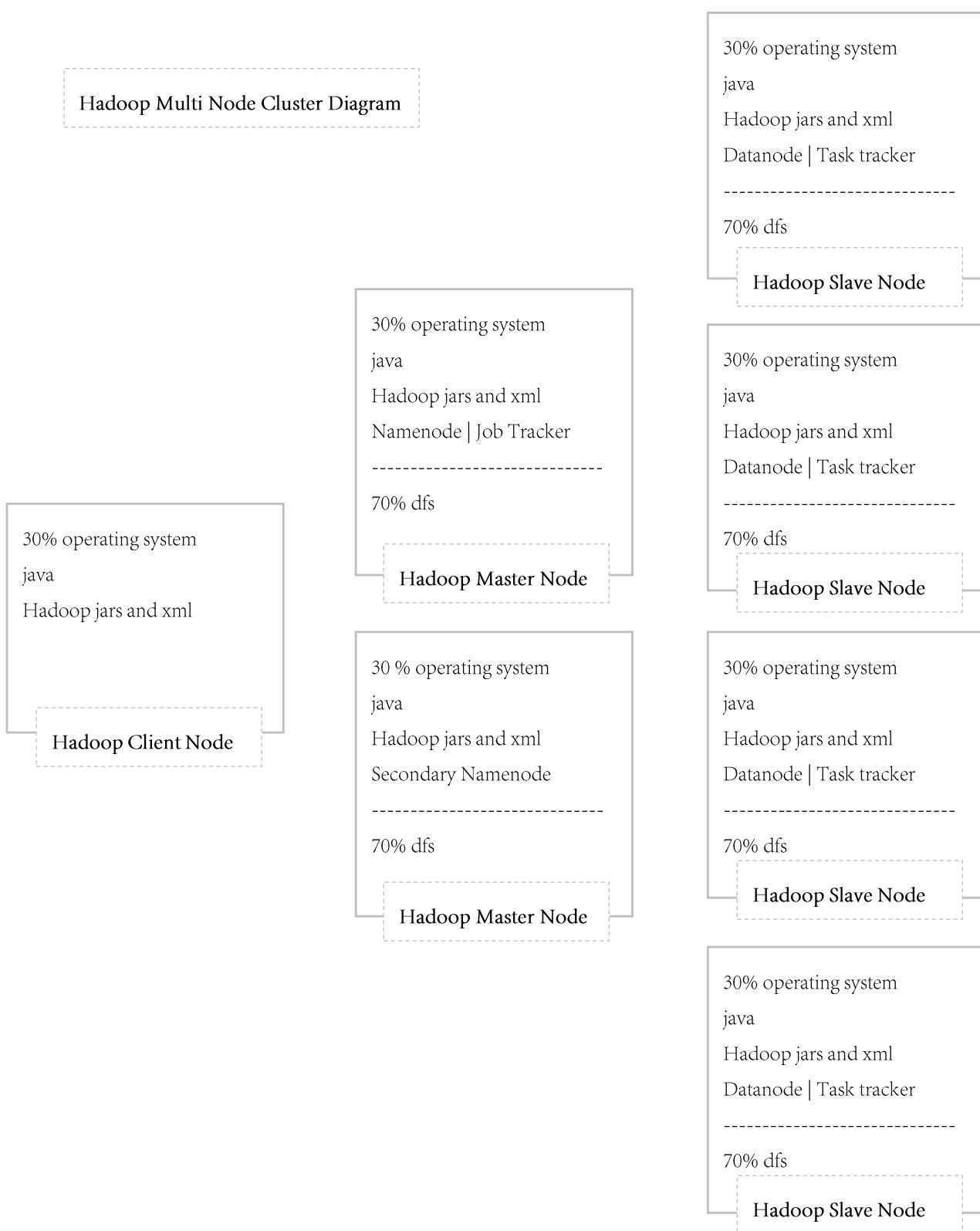
So let us start from buying let us say four servers. Either these servers can be physical servers or cloud servers from any cloud provider like [Amazon AWS](#) or [Virtual Machines](#).

Ideally a dedicated server is machine which has 30% of its storage space reserved for an operating system. In our case we will consider the operating system as [Linux CentOS 7](#). Let us assume we are creating a five node Hadoop Cluster with one master node for Namenode and Job Tracker, one master node for Secondary Namenode, one client machine that will normally be our laptop and four slave nodes.

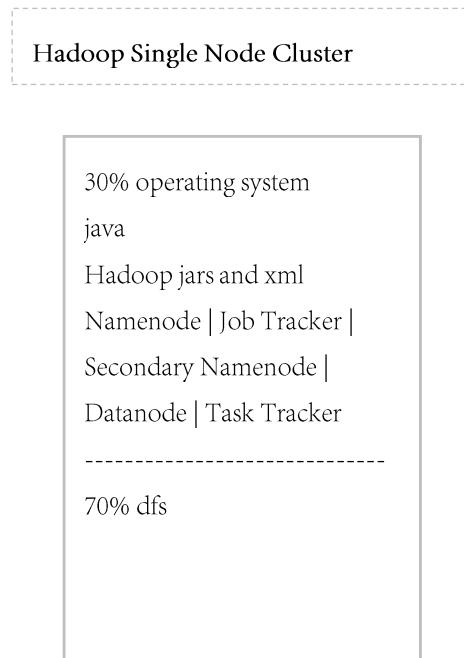
The first step will be to verify that the operating system is properly installed. Then we will do installation of some pre-requisites like Java and MySQL. Considering we are using the distributed package of Cloudera Distribution, we will download and install the Master Package on the master node and slave package on the slave nodes. Once the installations are done our servers should look something like this



Putting all components together, our Hadoop Cluster will look something like this,



If we had done a single node cluster installation, then our cluster would look something like this,



Hadoop Daemons

Now that we have a fair bit of idea of the Hadoop Architecture, let us discuss some more on the different components in the cluster.

Namenode

Namenode is considered to be the brain of the Hadoop Architecture. It acts like the master of all Datanodes in the cluster and is responsible for storing meta-data information of all data stored by the Datanodes. Namenode is the first point of contact for the outer world with the Hadoop cluster. The Namenode listens on [port 8020](#) and is accessible from web [URL port of 50070](#).

Roles of Namenode:

Initiating file Read Write Delete process for the client

Managing all steps involved in File Anatomy of Read and Write

Managing file distribution mappings

Managing and storing meta-data information

Secondary Namenode

Secondary Namenode is considered as the [cloud backup of Namenode](#). As Namenode has a track of all meta-data, the failure of the Namenode will mean your entire cluster is inaccessible. The data will still be there with the Datanode but no one will ever know which chunks belong to which file and hence no one will be able to access anything. Hence, a copy of all meta-data is stored with the Secondary Namenode

Relation between a Namenode and a Secondary Namenode is similar to the relation between your Computer Hard Disk and your Dropbox account where a backup of your Hard Disk is stored. If your Hard Disk crashes, you can use your Dropbox account as your primary Hard Disk. What you need to do is, buy another

Hard Disk, configure it and then you can restore your data from the Dropbox account.

Similarly, if your Namenode goes down, the Secondary Namenode will **NEVER** takes its place. You will have to configure a new Namenode and then you can restore the meta-data information from the backup stored in the Secondary Namenode.

Datanode

Datanode is the slave of Namenode and deals with storing data on the dfs (70% of storage) on a slave node. Separate independent Datanodes manage the dfs of all slaves and together this dfs is called as HDFS and the sum of this capacity is viewed as the entire capacity of the Cluster.

Job Tracker

The Job Tracker is the master of the processing unit of Hadoop and takes care of the Map Reduce programming model with its slaves the Task Trackers. The Job Tracker is the first point of contact for any job submission by the client. Job Tracker listens on port 8021 and is accessible from web URL port of 50030.

Roles of Job Tracker:

Job Analysis

Job Scheduling

Job Monitoring

Task Tracker

A Task Tracker is the slave of the Job Tracker and would be installed in all slave machines. These are the services actually responsible for Task Execution. All data related needs of the Task Tracker can be satisfied by only the Datanode service which is installed in the same server as the Task Tracker. The Task Trackers take

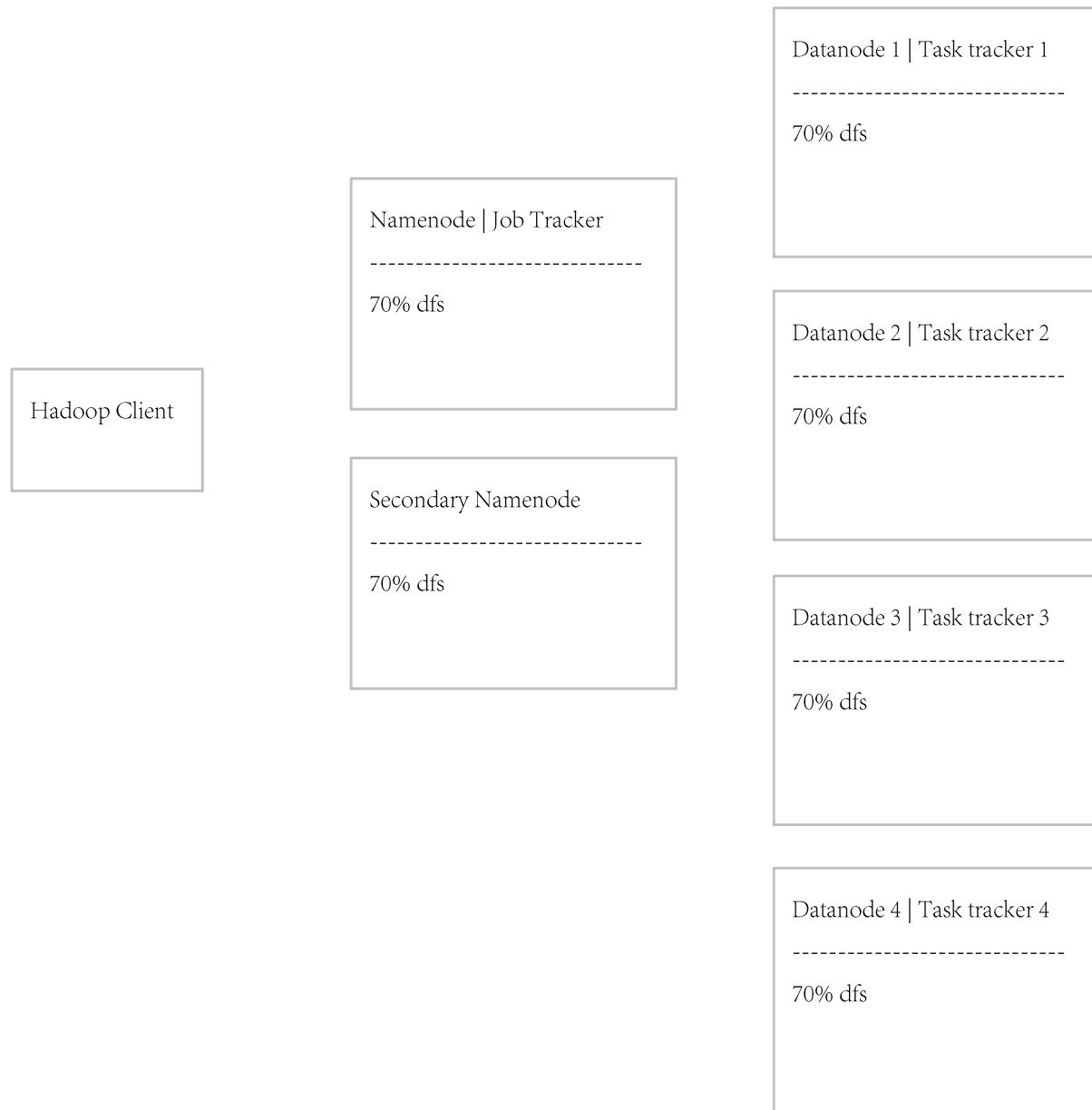
instructions from the Job Tracker and can communicate with the Namenode and Datanodes.

With this, we complete the introduction to the Hadoop Framework and different components involved. In the next chapter, we will have a look at the job process in the Hadoop framework.

07. Hadoop Job Process

In this chapter we will have a look at how different components of the Hadoop Framework come together to get a file stored or processed.

At first sight, this is how our cluster will look like:



Do note that in the above diagram, only the required components are showed. For a detailed architecture, refer to the previous chapter.

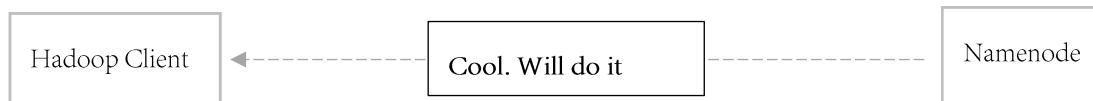
Now, the scenario is that the client has a file named **hello.txt** which is of size **180 MB** which is to be stored in the Hadoop Cluster. For this a series of event will take which are demonstrated below:



The Hadoop Client will send a command to the Namenode on its default listening port of 8020 for storing the file.

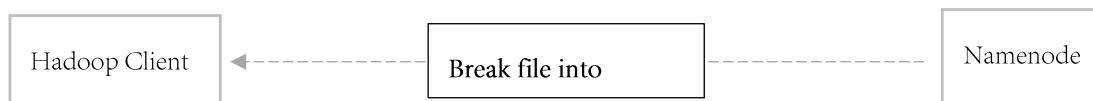
`hadoop fs -put < source as /home/jayantmohite/hello.txt> <destination as /user/jayantmohite>`

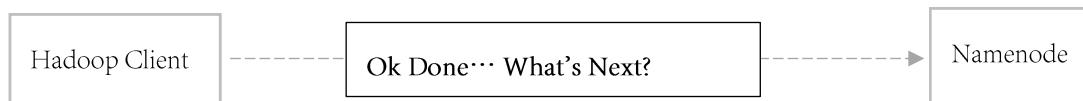
After every step there will be an acknowledgement shared between the involved components without which the next step will not begin.



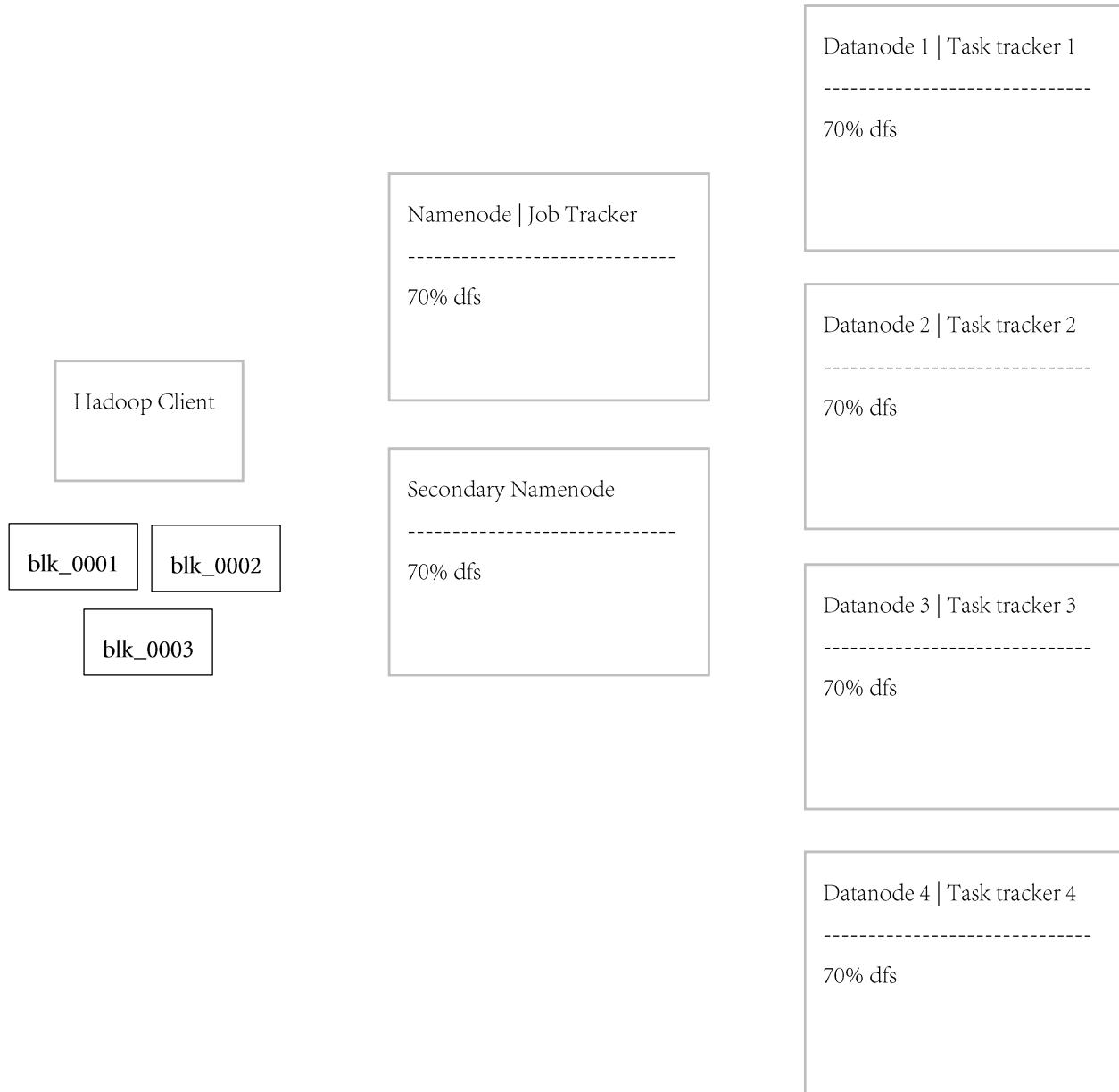
The Namenode will tell the client that Hadoop is a **Distributed Environment** that works in **Parallel Processing Model**. So it cannot accept the input file as a whole. The Namenode will request the client to break the file into smaller chunks with size of each chunk **not greater than 64 MB**.

Why 64 MB? Its because the OS used in production environment is of 64 bits and its been tested and verified that this OS can process a 64 MB file in one thread. This being a Big Data solution, we cannot expect the clients to give us such a small file so its imposed to break it into this dimensions.





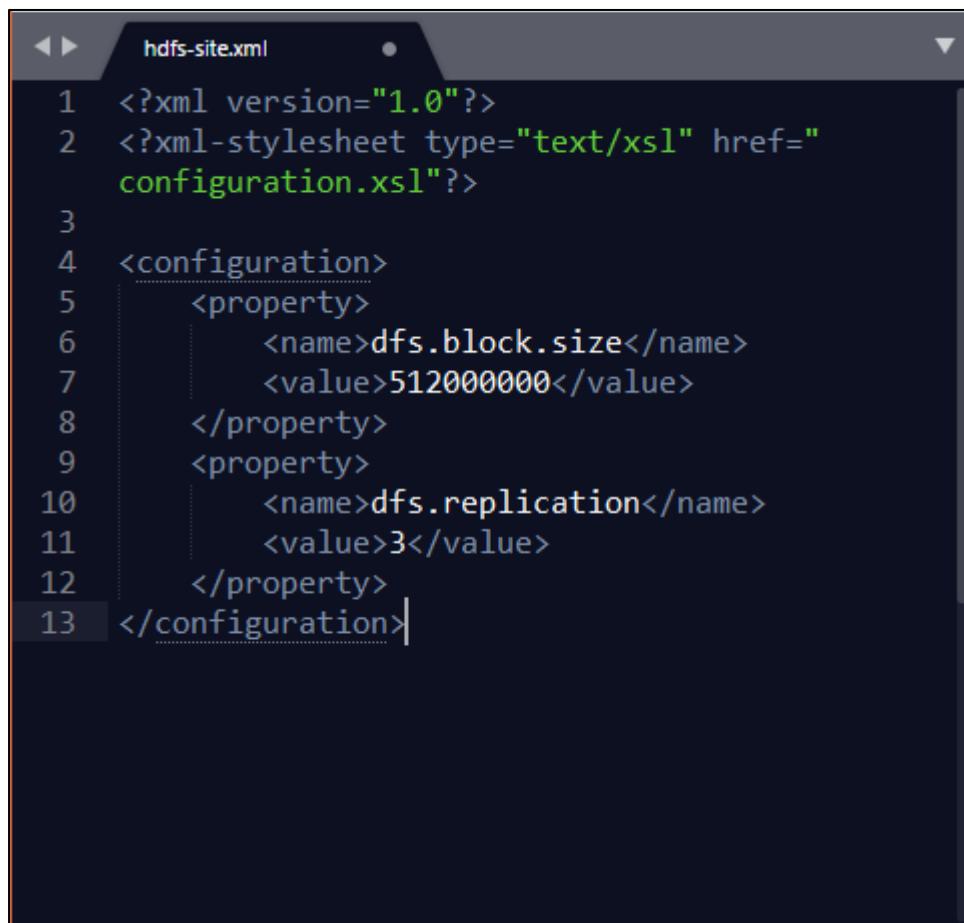
By now this is how the architecture will look like:



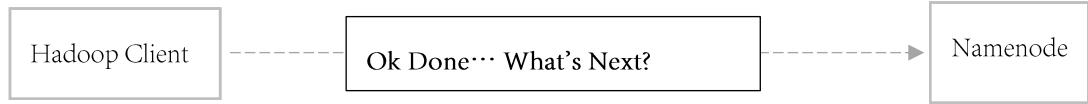
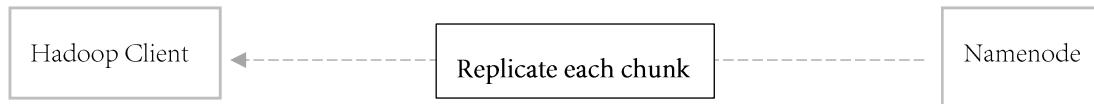
Now the Namenode tells the Client that we are not simply a parallel processing system but we are a reliable parallel processing system. So if the blocks are given to one of the Datanode and the Datanode fails, then we may lose the data which is not reliable.

So the Namenode will request the client to create duplicate copies of each chunk. The number of copies to be created for each chunk is specified by the value of a property called as **Replication Factor**.

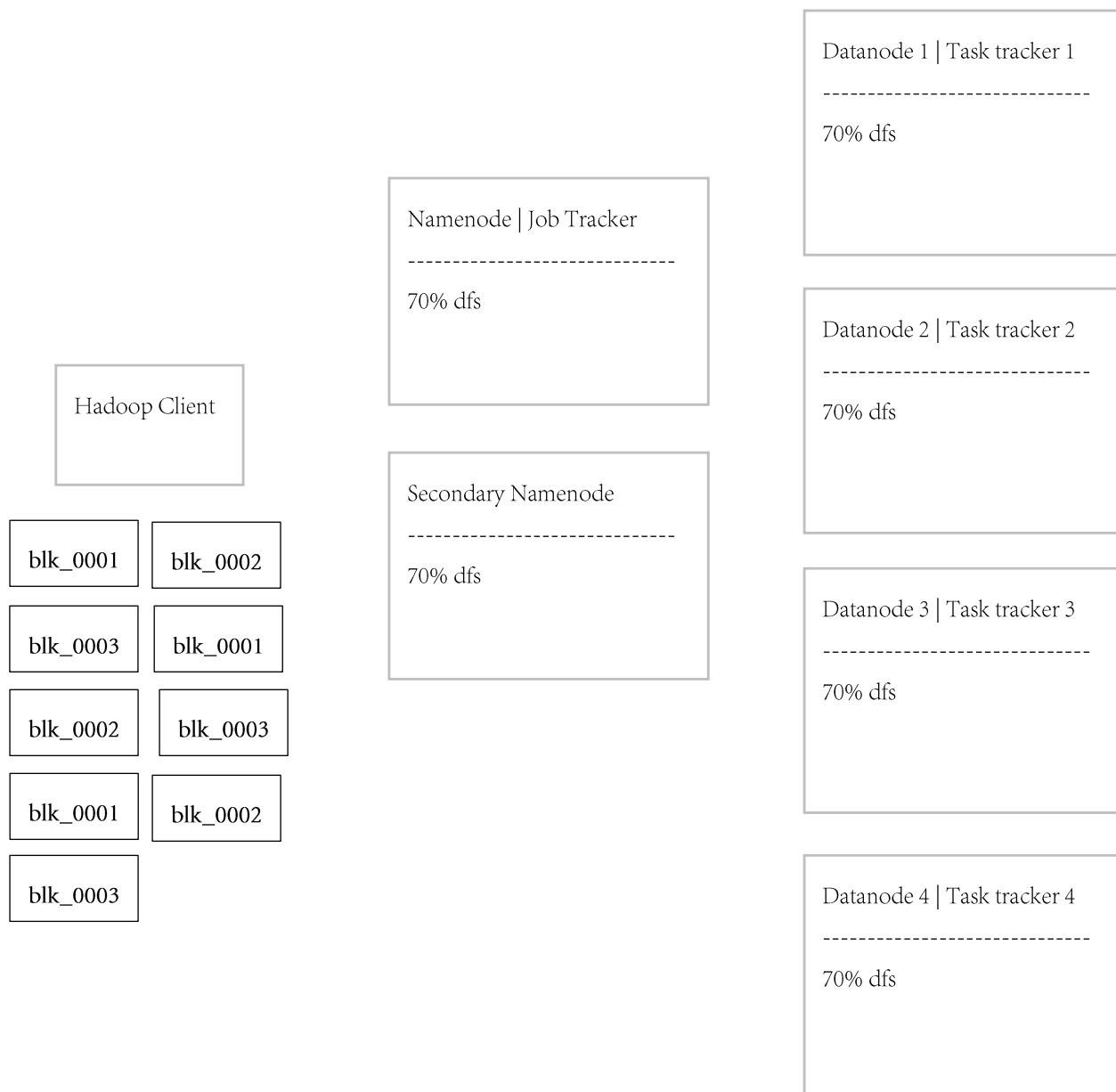
The default value of the **replication factor is 3**, which means there should be a total of 3 copies of each chunk. Based on the reliability of the servers and the criticality of the data, this value can be either reduced to a minimum of 1 or increased to a maximum of number of Datanodes in the target Cluster.



```
hdfs-site.xml
1 <?xml version="1.0"?>
2 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3
4 <configuration>
5   <property>
6     <name>dfs.block.size</name>
7     <value>512000000</value>
8   </property>
9   <property>
10    <name>dfs.replication</name>
11    <value>3</value>
12  </property>
13 </configuration>
```

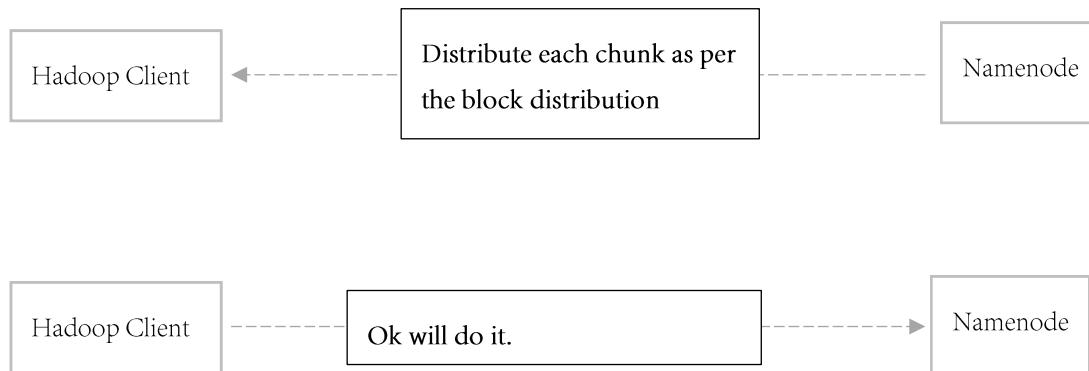


By now this is how the cluster will look like



Now the Namenode will create a plan which will decide which block have to stored on which of the Datanodes. This plan is created by the Namenode on a Random basis keeping only one thing in mind that **no Datanode should get more than one copy of the same block.**

Datanode 1	Datanode 2	Datanode 3	Datanode 4
blk_0001	blk_0002	blk_0003	blk_0001
blk_0002	blk_0003	blk_0001	blk_0002
blk_0003			



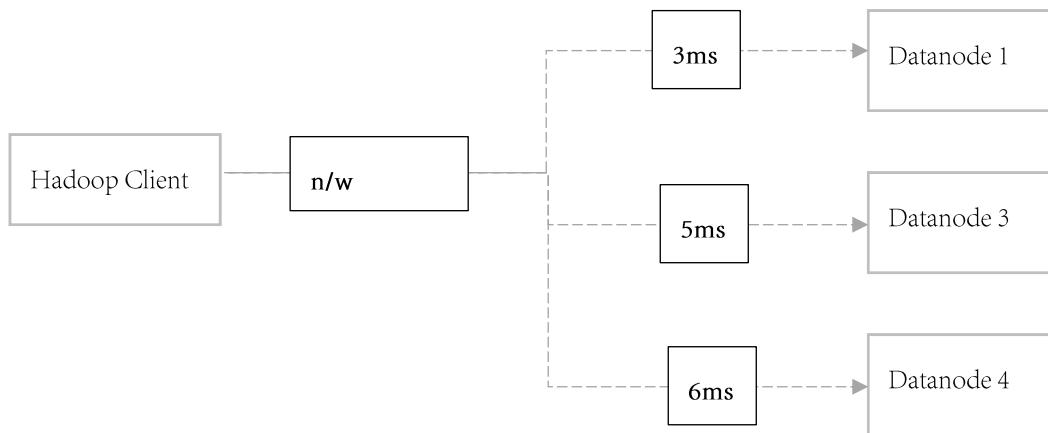
By this time the client has already done a lot of work and is no longer interested to do more work. So now the client will play smart.

The client will start with the first block.

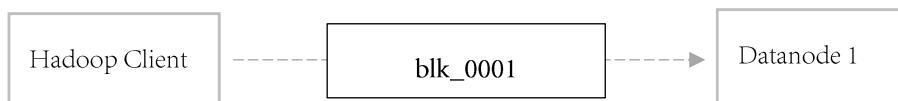
The Client will mark all the Datanodes who are supposed to receive this block.

The Client will then calculate the network distance of itself with all these Datanodes in consideration.

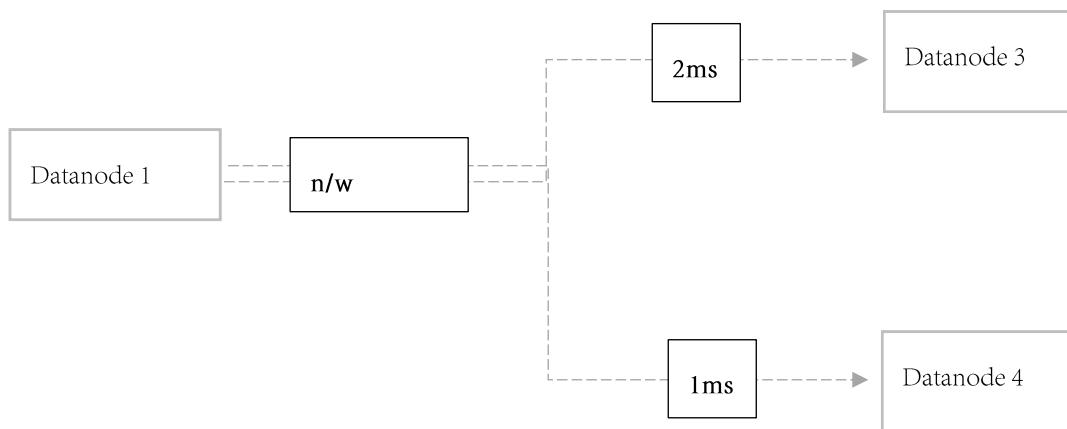
From this the client will identify the nearest node.



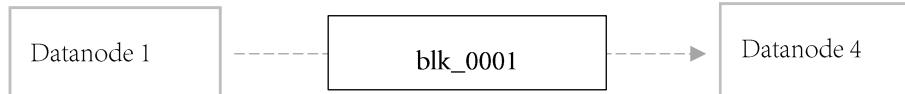
The Client will then share all the copies of the first block to this Datanode and will ask this Datanode to keep one block and distribute the remaining to the other Datanodes



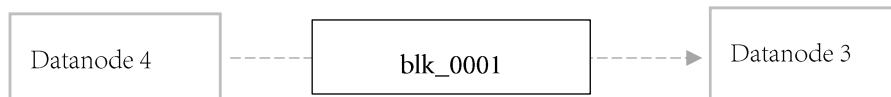
This Datanode will then calculate the distance between himself and the other 2 Datanodes in consideration



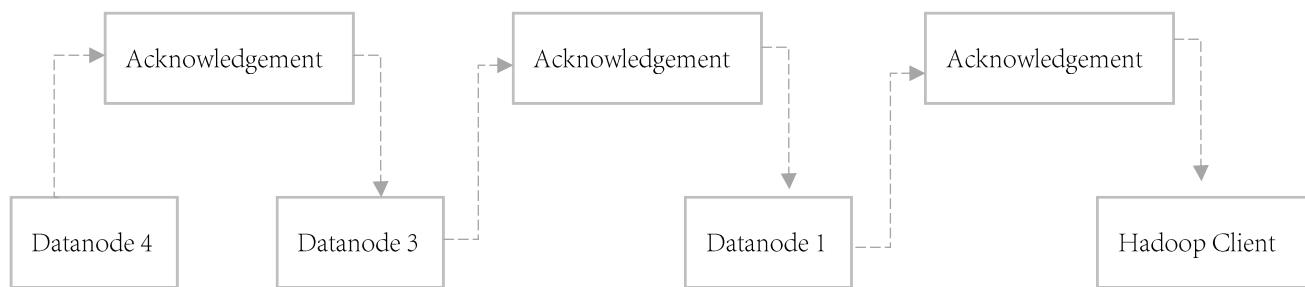
It will then pass on two copies of the block to the nearest Datanode



This Datanode will now keep one of the two copies and pass on one copy to the last Datanode.



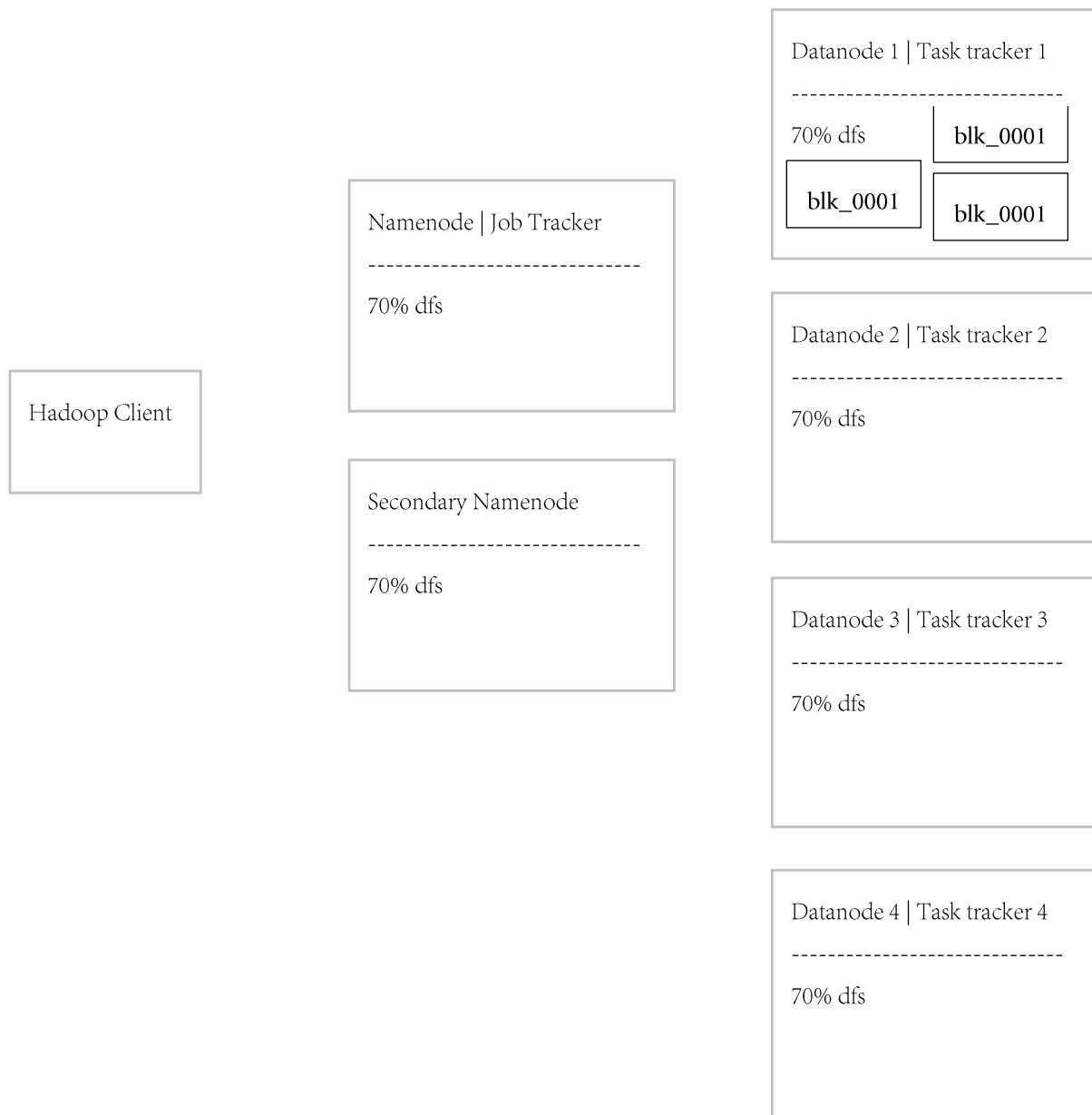
Post this, a series of acknowledgement will finally reach to the client.



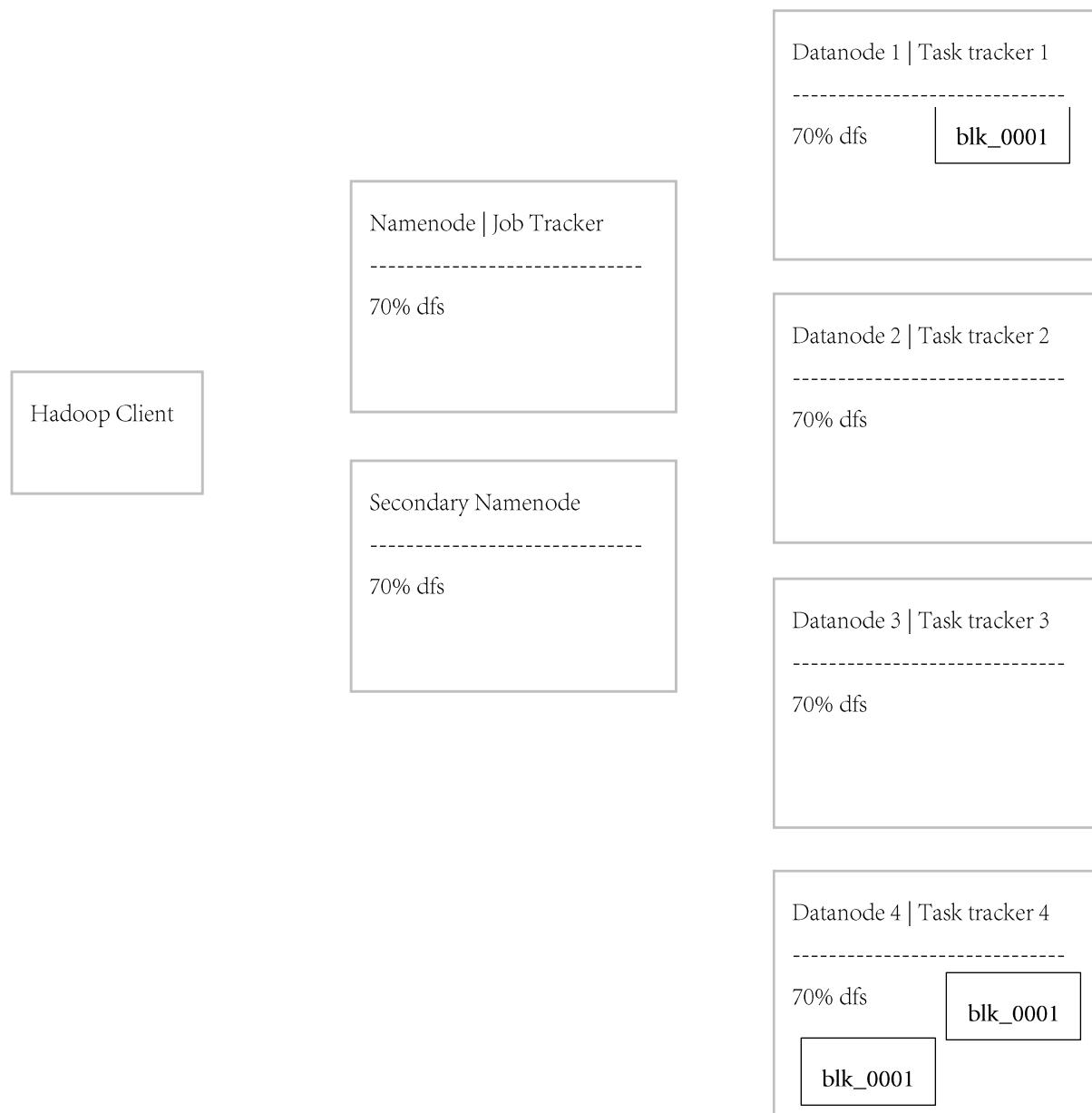
The same step will be repeated for the remaining blocks as well.

Below are some images which will demonstrate the state of the cluster after every iteration explained above

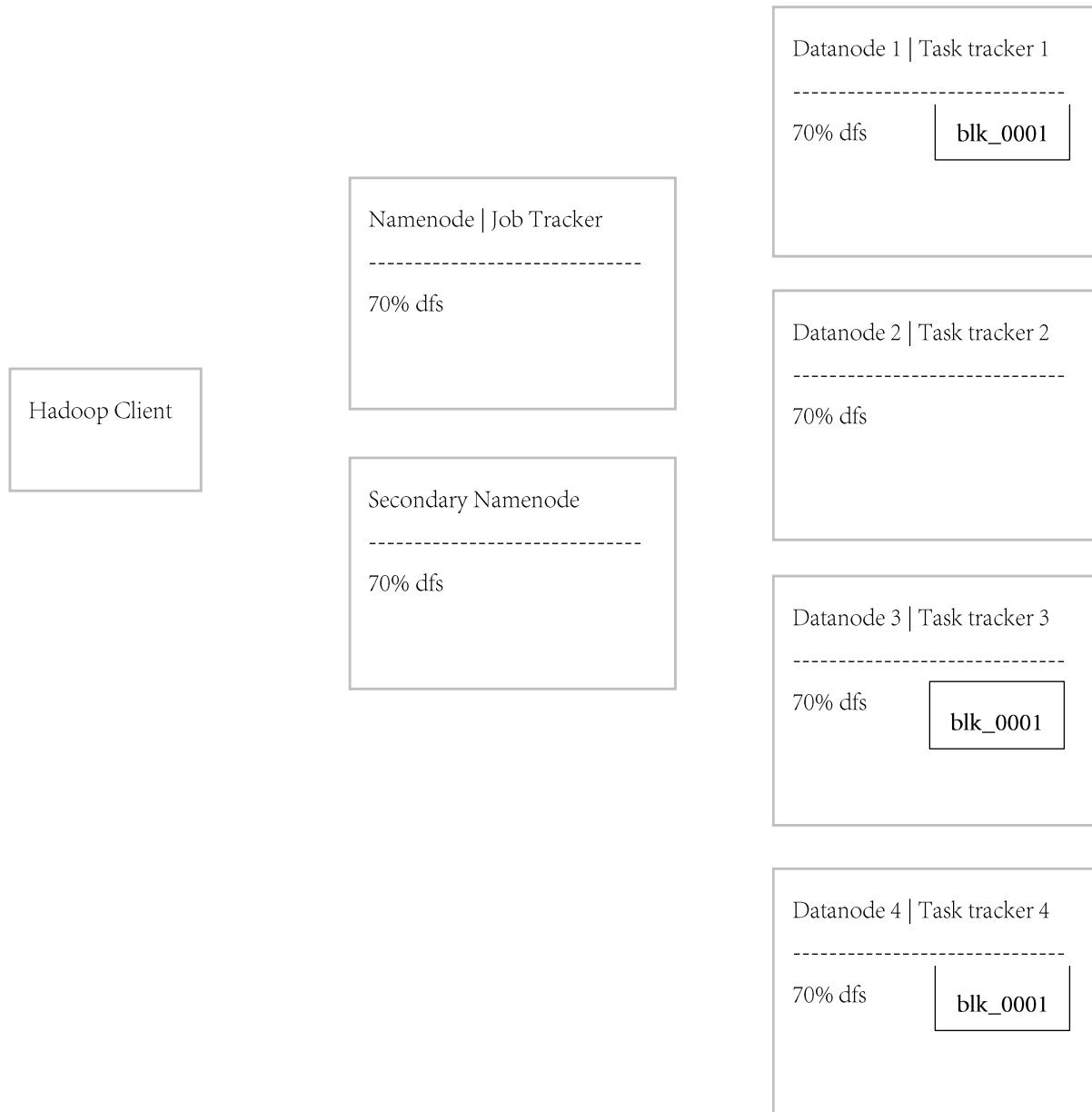
This is the cluster state once all copies of the block are transferred by Client to the nearest Datanode



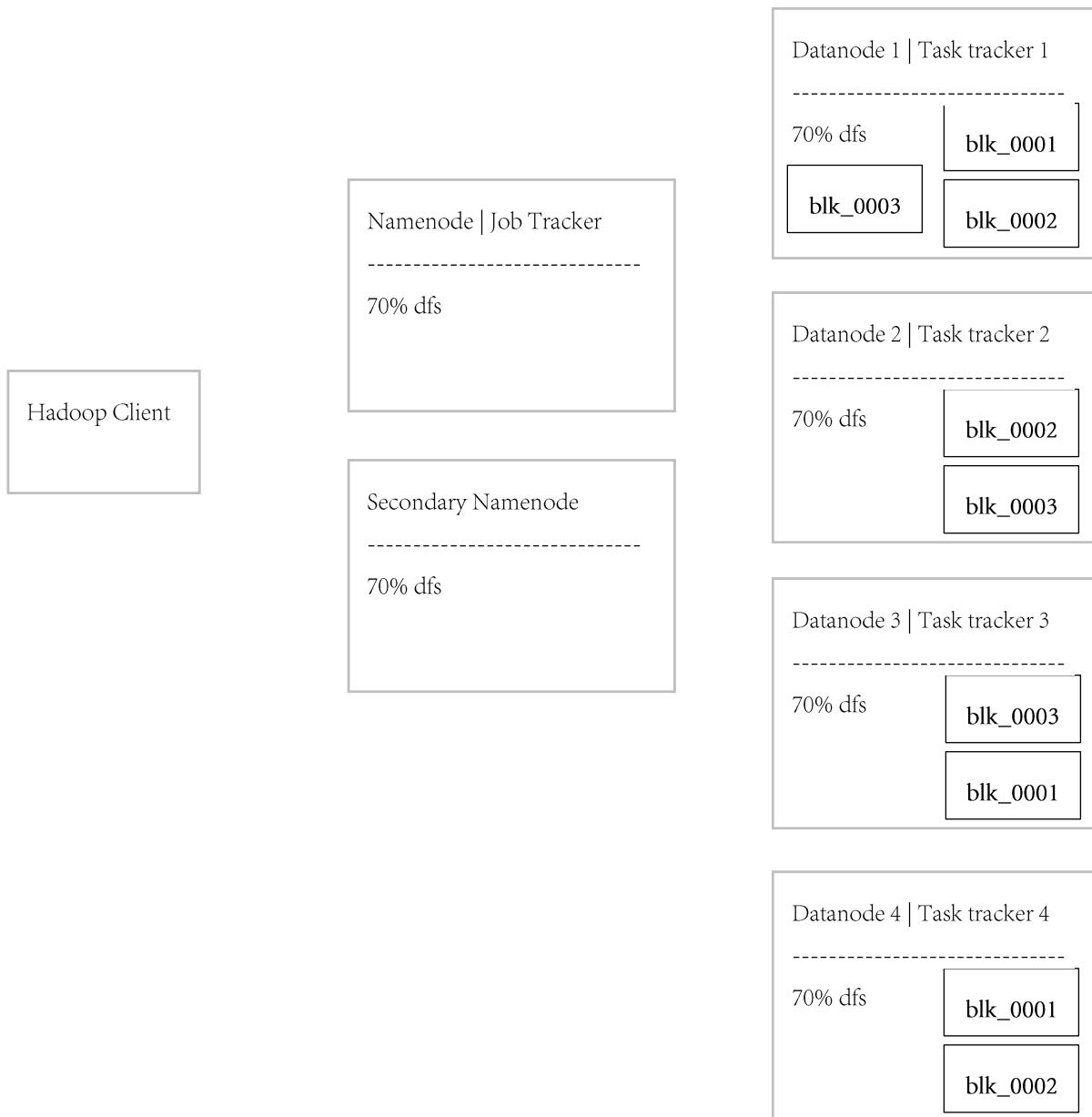
This is the cluster state once two copies of the block are transferred by first Datanode to the nearest second Datanode



This is the cluster state once the last copy of the block is transferred by second Datanode to the third Datanode

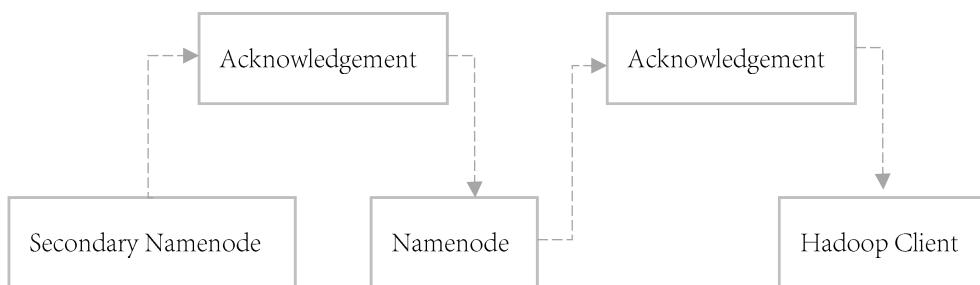
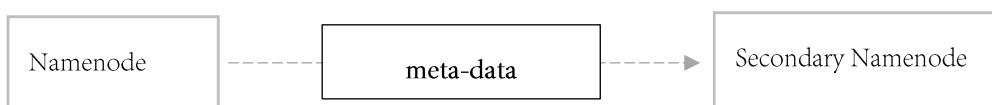
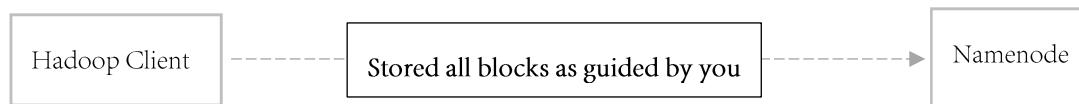


This is how the cluster will look after all the block have been transferred and the client has received an acknowledgement for all.



Post this the client will acknowledge the Namenode to notify him that the block distribution has successfully happened.

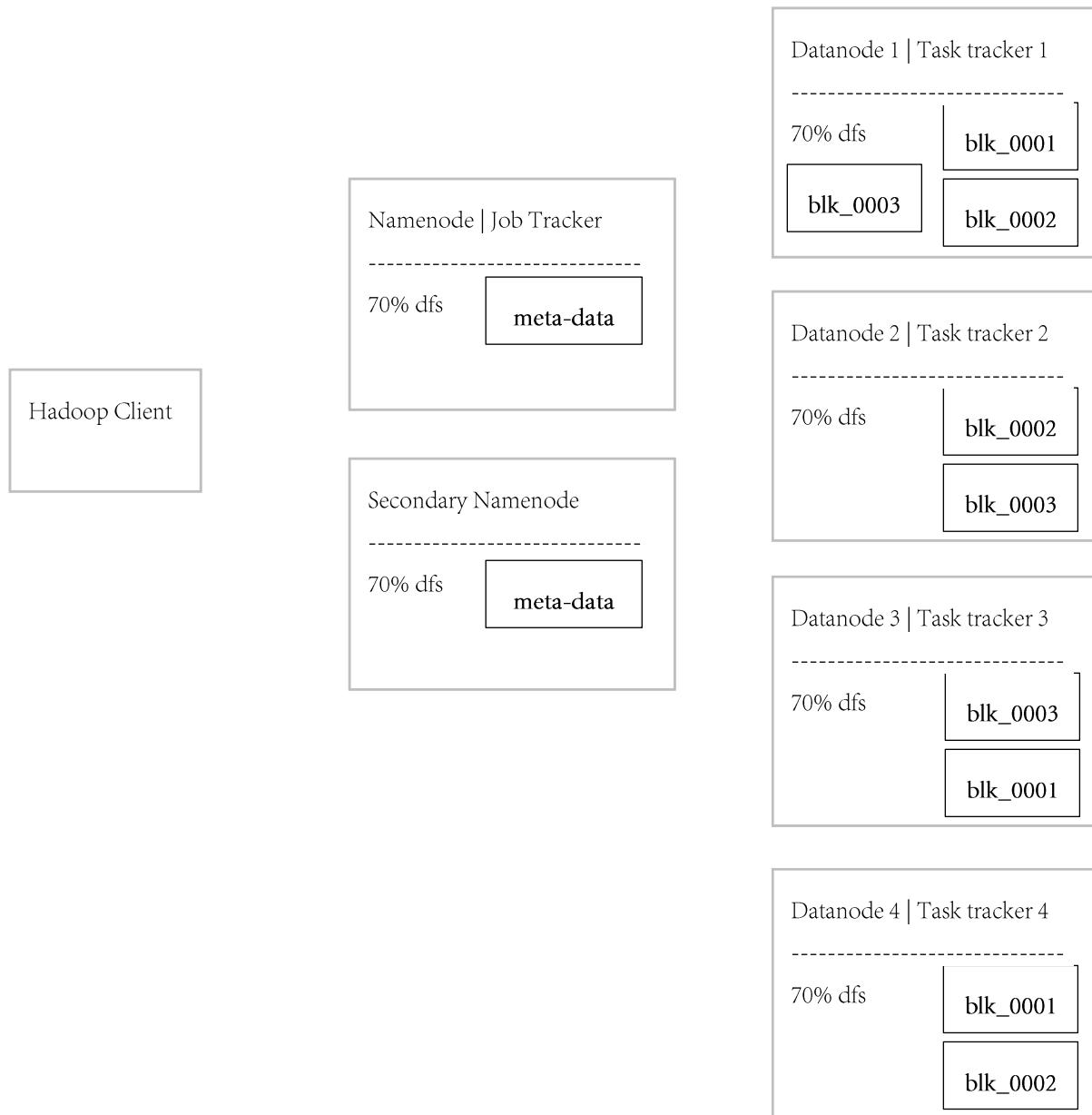
After this the namenode store the distribution mapping in its dfs and will replicated the same on the Secondary Namenode for backup.



Once the Namenode receives an acknowledgement of all block stored and the copy of metadata stored by the Secondary Namenode as well, this is when the entire process is completed and the same is intimated to the Hadoop Client.

This entire process is called as [File Anatomy of Write](#).

The below image show the final state of the Hadoop Cluster after this process is completed.



[File Anatomy of Read](#) is the exact same process in reverse order.

Now that the file has been stored successfully, its time to move on to some processing. A series of events will happen in this process as well which are demonstrated below

The client submits a program called as wordcount.jar to the Job Tracker on port [8021](#) which is the default listening port of the Job Tracker.

This program contains 3 java class files namely

1. Map Class

This contains the [map\(\)](#) which is written by the Client

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WordMapper extends MapReduceBase implements
    Mapper<LongWritable, Text, Text, IntWritable> {

    @Override
    public void map(LongWritable key, Text value,
        OutputCollector<Text, IntWritable> output, Reporter reporter)
        throws IOException {
        String s = value.toString();
        for (String word : s.split("\\W+")) {
            if (word.length() > 0) {
                output.collect(new Text(word), new IntWritable(1));
            }
        }
    }
}
```

2. Reduce Class

This contains the `reduce()` which is written by the Client

```
import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

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

    @Override
    public void reduce(Text key, Iterator<IntWritable> values,
        OutputCollector<Text, IntWritable> output, Reporter reporter)
        throws IOException {

        int wordCount = 0;
        while (values.hasNext()) {
            IntWritable value = values.next();
            wordCount += value.get();
        }
        output.collect(key, new IntWritable(wordCount));
    }
}
```

3. Main Class

This contains the `main()` where the job configuration is defined and the map and reduce classes are initialized

```

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WordCount extends Configured implements Tool {

    @Override
    public int run(String[] args) throws Exception {

        if (args.length != 2) {
            System.out.printf(
                "Usage: %s [generic options] <input dir> <output dir>\n", getClass()
                    .getSimpleName());
            ToolRunner.printGenericCommandUsage(System.out);
            return -1;
        }

        JobConf conf = new JobConf(getConf(), WordCount.class);
        conf.setJobName(this.getClass().getName());

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

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

        conf.setMapOutputKeyClass(Text.class);
        conf.setMapOutputValueClass(IntWritable.class);

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);

        JobClient.runJob(conf);
        return 0;
    }

    public static void main(String[] args) throws Exception {
        int exitCode = ToolRunner.run(new WordCount(), args);
        System.exit(exitCode);
    }
}

```

Upon receiving the job, the Job Tracker will perform a set of responsibilities.

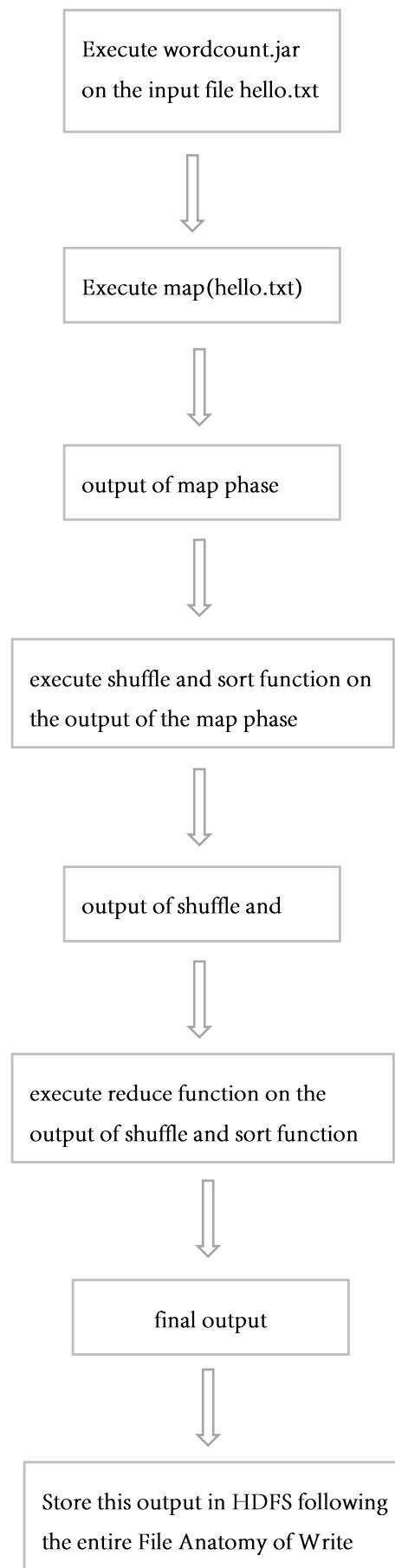
Job Analysis

The Job Tracker will evaluate the job configuration from the main class and will evaluate what is the exact requirement of the program. He will identify that what is to be done is:

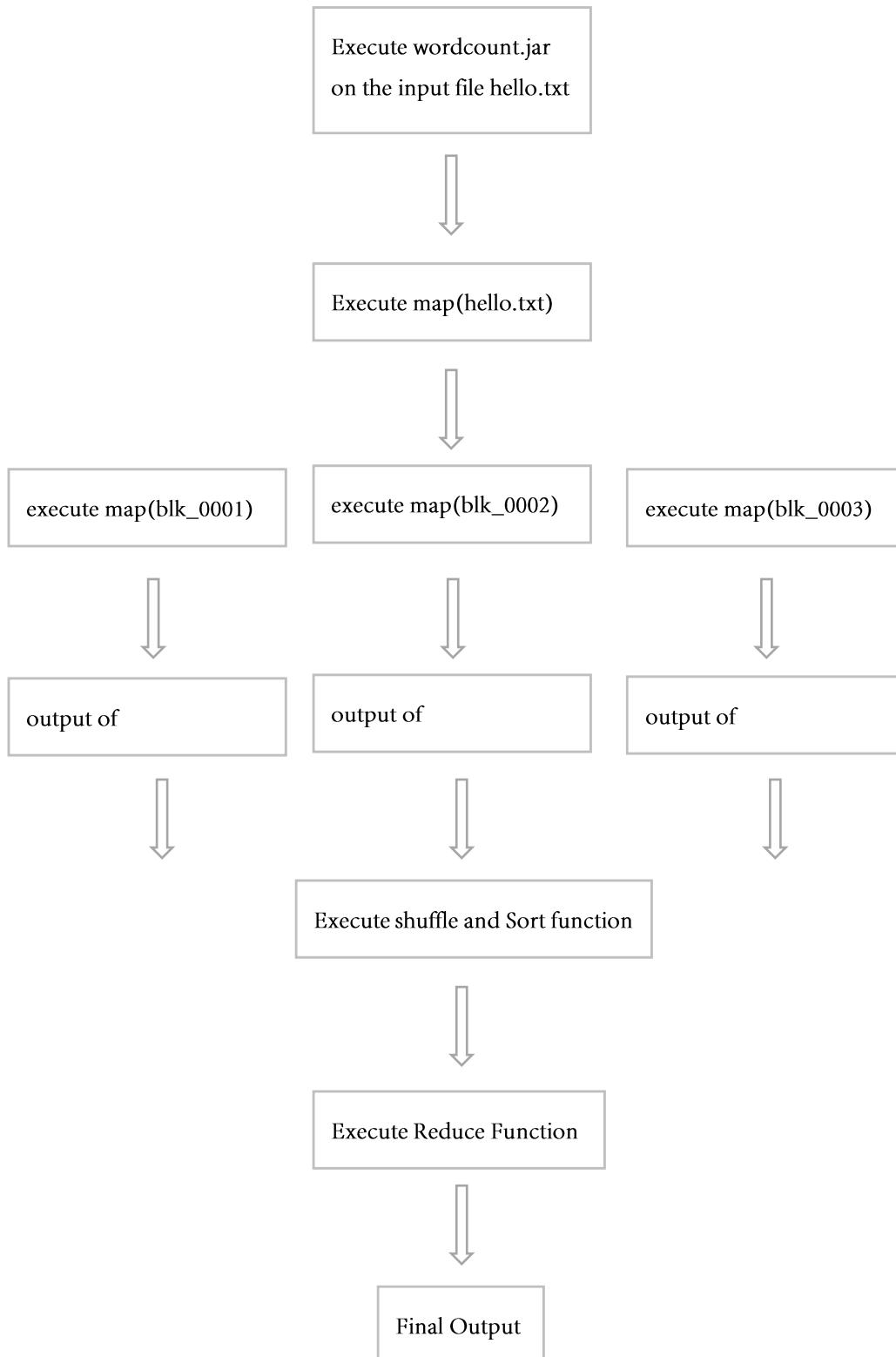
Execute wordcount.jar on
the input file hello.txt

Which implies the execution of the two phases of Map Reduce programming model namely the map phase in which the map function is executed and the reduce phase in which the shuffle and sort function will combine the outputs of all map functions and then the reduce function has to be executed on the combine output

This process is demonstrated in the image below.



But the Job Tracker is not aware of what exactly the file hello.txt is. So he will consult the Namenode and the request for the Block Mapping of hello.txt. Upon receiving this, the process will look like this



Now based on all the Analysis, the Job Tracker will split the Job into multiple Tasks

Task 1 = Execute map() on blk_0001

Task 2 = Execute map() on blk_0002

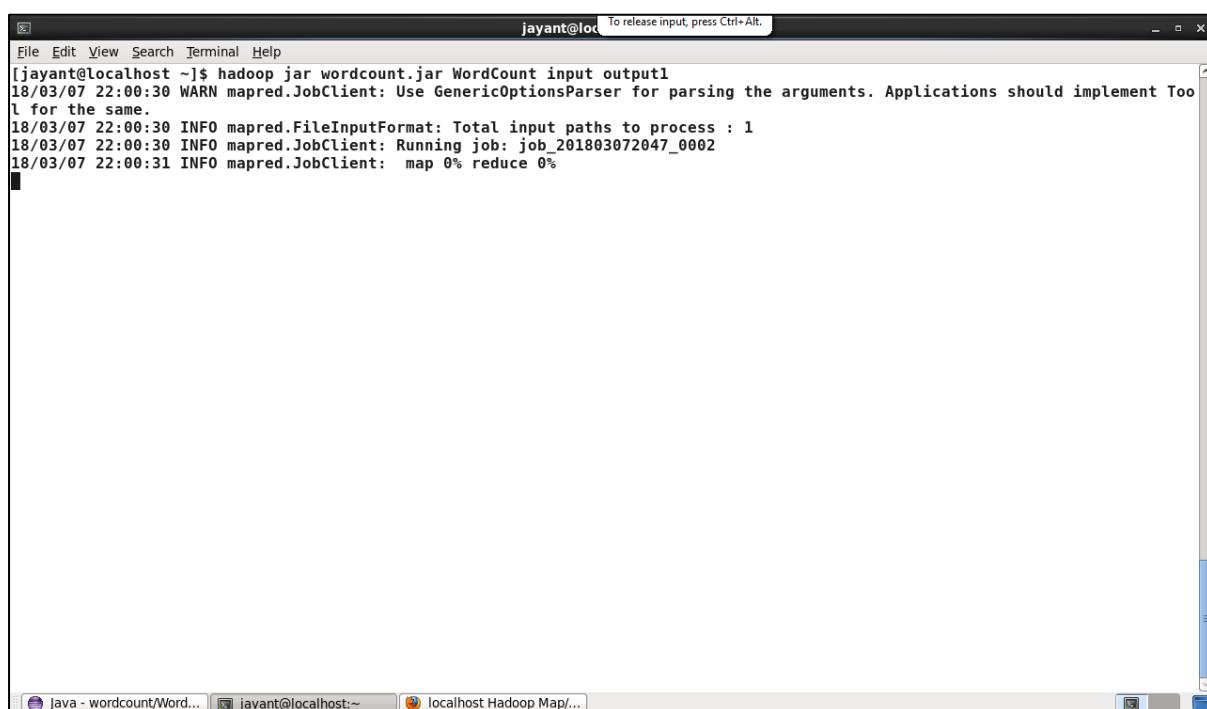
Task 3 = Execute map() on blk_0003

Task 4

Combine output of Task 1, Task 2 and Task 3 by executing the
shuffle_and_sort()

Execute Reduce Function on output of shuffle_and_sort()

Store output of reduce() in HDFS following the File Anatomy of Write



The screenshot shows a terminal window titled "jayant@localhost" with the command "hadoop jar wordcount.jar WordCount input output". The output log shows:

```
[jayant@localhost ~]$ hadoop jar wordcount.jar WordCount input output
18/03/07 22:00:30 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
18/03/07 22:00:30 INFO mapred.FileInputFormat: Total input paths to process : 1
18/03/07 22:00:30 INFO mapred.JobClient: Running job: job_201803072047_0002
18/03/07 22:00:31 INFO mapred.JobClient: map 0% reduce 0%
```

Cluster Summary (Heap Size is 174.38 MB/888.94 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Excluded Nodes
2	0	1	1	2	0	0	0	2	2	4.00	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)

Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201803072047_0001	NORMAL	jayant	WordCount	0.00%	3	0	0.00%	3	0	NA	NA

Retired Jobs

none

Local Logs

Log directory, Job Tracker History
Hadoop, 2018.

Java - Eclipse jayant@localhost:~ localhost Hadoop Map/...

Once the Analysis is done, the Job Tracker will schedule the Tasks. For each scheduling certain criteria is considered which is as follows:

For each Slave to be considered for executing the Task, both services should namely the Datanode and Task Tracker should be up and running

For each Slave passing the above criteria , the block on which the task is to be execute should be available

For each Slave passing the above criteria, free execution slot should be available for scheduling the Task

From each slave passing all the above criteria any slave is randomly selected for the operation

The assigned slaves then start working on their Task and keep the Job Tracker informed about the status of the same. Below are some images that depict the state of the cluster from the browser view.

localhost Hadoop Map/Reduce A To release input, press Ctrl+Alt. refox

[localhost Hadoop Map/Reduce ...](#)

localhost:50030/jobtracker.jsp

2 0 2 1 2 0 0 0 2 2 4.00 0 0 Quick Links

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)

Example: 'users:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201803072047_0002	NORMAL	jayant	WordCount	12.15%	3	0	0.00%	1	0	NA	NA

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201803072047_0001	NORMAL	jayant	WordCount	100.00%	3	3	100.00%	3	3	NA	NA

Retired Jobs

none

Local Logs

Java - wordcount/Word... jayant@localhost:~ localhost Hadoop Map/...

Hadoop map task list for job_201803072047_0002 To release input, press Ctrl+Alt. Mozilla Firefox

[Hadoop map task list for job_201803072047_0002](#)

localhost:50030/jobtasks.jsp?jobid=job_201803072047_0002&type=map&pagenum=1&state=running

Hadoop map task list for job_201803072047_0002 on localhost

Running Tasks

Task	Complete	Status	Start Time	Finish Time	Errors	Counters
task_201803072047_0002_m_000000	68.99%	hdfs://localhost:8020/user/jayant/input/hello.txt:0+67108864	7-Mar-2018 22:00:35			22
task_201803072047_0002_m_000001	69.90%	hdfs://localhost:8020/user/jayant/input/hello.txt:67108864+67108864	7-Mar-2018 22:00:35			22

[Go back to JobTracker](#)

Hadoop, 2018.

Java - wordcount/Word... jayant@localhost:~ Hadoop map task list f...

```

Task Logs: 'attempt_201803072047_0002_m_000001_0' To release input, press Ctrl+Alt+. in Firefox
File Edit View History Bookmarks Tools Help
Task Logs: 'attempt_20180307... +'
localhost:50060/tasklog?attemptid=attempt_201803072047_0002_m_000001_0&all=true
Google

syslog logs

2018-03-07 22:00:38,207 WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2018-03-07 22:00:39,342 WARN org.apache.hadoop.conf.Configuration: session.id is deprecated. Instead, use dfs.metrics.session-id
2018-03-07 22:00:39,345 INFO org.apache.hadoop.metrics.jvm.JvmMetrics: Initializing JVM Metrics with processName=MAP, sessionId=0
2018-03-07 22:00:40,067 INFO org.apache.hadoop.util.ProcessTree: setsid exited with exit code 0
2018-03-07 22:00:40,077 INFO org.apache.hadoop.mapred.Task: Using ResourceCalculatorPlugin : org.apache.hadoop.util.LinuxResourceCalculatorPlugin@848636
2018-03-07 22:00:40,518 INFO org.apache.hadoop.mapred.MapTask: Processing split: hdfs://localhost:8020/user/jayant/input/hello.txt:67108864+67108864
2018-03-07 22:00:40,547 WARN mapreduce.Counters: Counter name MAP_BYTES is deprecated. Use FileInputFormatCounters as group name and BYTES_READ as counter name instead
2018-03-07 22:00:40,555 INFO org.apache.hadoop.mapred.MapTask: numReduceTasks: 1
2018-03-07 22:00:40,563 INFO org.apache.hadoop.mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2018-03-07 22:00:40,571 INFO org.apache.hadoop.mapred.MapTask: io.sort.mb = 100
2018-03-07 22:00:40,858 INFO org.apache.hadoop.mapred.MapTask: data buffer = 79691776/99614720
2018-03-07 22:00:40,858 INFO org.apache.hadoop.mapred.MapTask: record buffer = 262144/327680
2018-03-07 22:00:42,625 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: record full = true
2018-03-07 22:00:42,625 INFO org.apache.hadoop.mapred.MapTask: bufstart = 0; bufend = 2643164; bufvoid = 99614720
2018-03-07 22:00:42,626 INFO org.apache.hadoop.mapred.MapTask: kvstart = 0; kvend = 262144; length = 327680
2018-03-07 22:00:43,379 INFO org.apache.hadoop.mapred.MapTask: Finished spill 0
2018-03-07 22:00:44,169 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: record full = true
2018-03-07 22:00:44,169 INFO org.apache.hadoop.mapred.MapTask: bufstart = 2643164; bufend = 5286285; bufvoid = 99614720
2018-03-07 22:00:44,169 INFO org.apache.hadoop.mapred.MapTask: kvstart = 262144; kvend = 196607; length = 327680
2018-03-07 22:00:44,657 INFO org.apache.hadoop.mapred.MapTask: Finished spill 1
2018-03-07 22:00:44,976 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: record full = true
2018-03-07 22:00:44,976 INFO org.apache.hadoop.mapred.MapTask: bufstart = 5286285; bufend = 7929683; bufvoid = 99614720
2018-03-07 22:00:44,976 INFO org.apache.hadoop.mapred.MapTask: kvstart = 196607; kvend = 131070; length = 327680
2018-03-07 22:00:45,342 INFO org.apache.hadoop.mapred.MapTask: Finished spill 2
2018-03-07 22:00:45,867 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: record full = true
2018-03-07 22:00:45,867 INFO org.apache.hadoop.mapred.MapTask: bufstart = 7929683; bufend = 10572568; bufvoid = 99614720
2018-03-07 22:00:45,868 INFO org.apache.hadoop.mapred.MapTask: kvstart = 131070; kvend = 65533; length = 327680
2018-03-07 22:00:46,470 INFO org.apache.hadoop.mapred.MapTask: Finished spill 3
2018-03-07 22:00:46,791 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: record full = true
2018-03-07 22:00:46,792 INFO org.apache.hadoop.mapred.MapTask: bufstart = 10572568; bufend = 13215990; bufvoid = 99614720
2018-03-07 22:00:46,792 INFO org.apache.hadoop.mapred.MapTask: kvstart = 65533; kvend = 327677; length = 327680
2018-03-07 22:00:47,567 INFO org.apache.hadoop.mapred.MapTask: Finished spill 4
2018-03-07 22:00:47,985 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: record full = true
2018-03-07 22:00:47,985 INFO org.apache.hadoop.mapred.MapTask: bufstart = 13215990; bufend = 15858880; bufvoid = 99614720
2018-03-07 22:00:47,985 INFO org.apache.hadoop.mapred.MapTask: kvstart = 327677; kvend = 262140; length = 327680
2018-03-07 22:00:47,985 INFO org.apache.hadoop.mapred.MapTask: Finished spill 5

```

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Running Jobs

none

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201803072047_0001	NORMAL	jayant	WordCount	100.00%	3	3	100.00%	3	3	NA	NA
job_201803072047_0002	NORMAL	jayant	WordCount	100.00%	3	3	100.00%	1	1	NA	NA

Retired Jobs

none

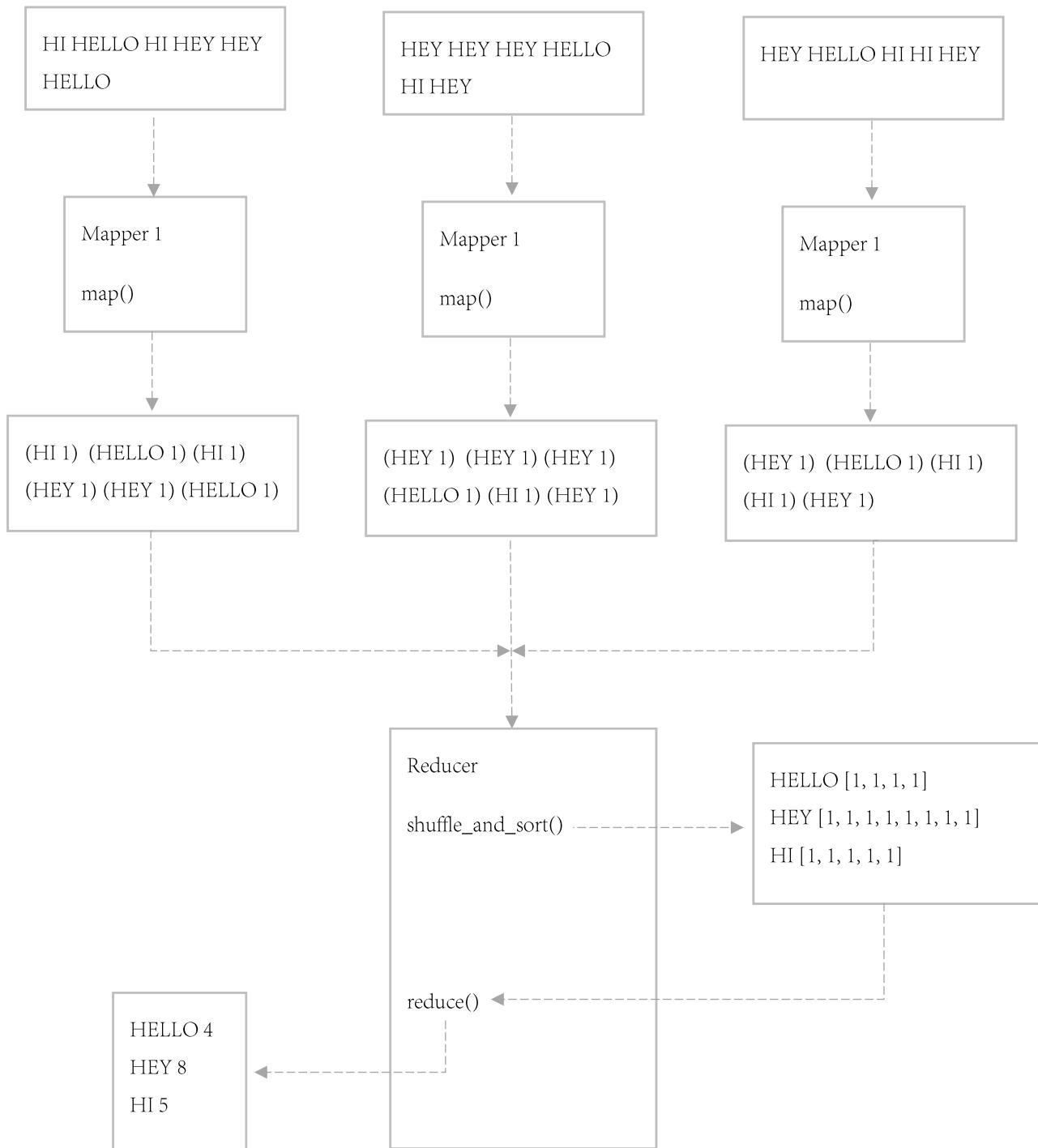
Local Logs

Log directory: Job Tracker History

Hadoop, 2018.

Once the Job Tracker is notified about the completion of all tasks in the Job, the Job Tracker marks the Job as completed and notifies the client about the same.

Map Reduce Programming Model Flow



Map Reduce Sample Execution

Input File

```
Was Where Why How Hi And Hello Now  
And Never Umbrella Was Who Whom And  
Where Why Who With Hey Oragne Was Oragne  
When Where Where How And  
Oragne Umbrella
```

Map Output

```
Was      1  
Where   1  
Why     1  
How     1  
Hi      1  
And     1  
Hello   1  
Now     1  
And     1  
Never   1  
Umbrella    1  
Was      1  
Who     1  
Whom   1  
And     1  
Where   1  
Why     1  
Who     1  
With   1  
Hey    1  
Oragne 1  
Was      1  
Oragne  1
```

Shuffle and Sort Output

```
And      1
And      1
And      1
And      1
Hello    1
Hey      1
Hi       1
How      1
How      1
Never    1
Now      1
Oragne   1
Oragne   1
Oragne   1
Umbrella 1
Umbrella 1
Was      1
Was      1
Was      1
When    1
Where   1
Where   1
Where   1
Where   1
Who     1
Who     1
```

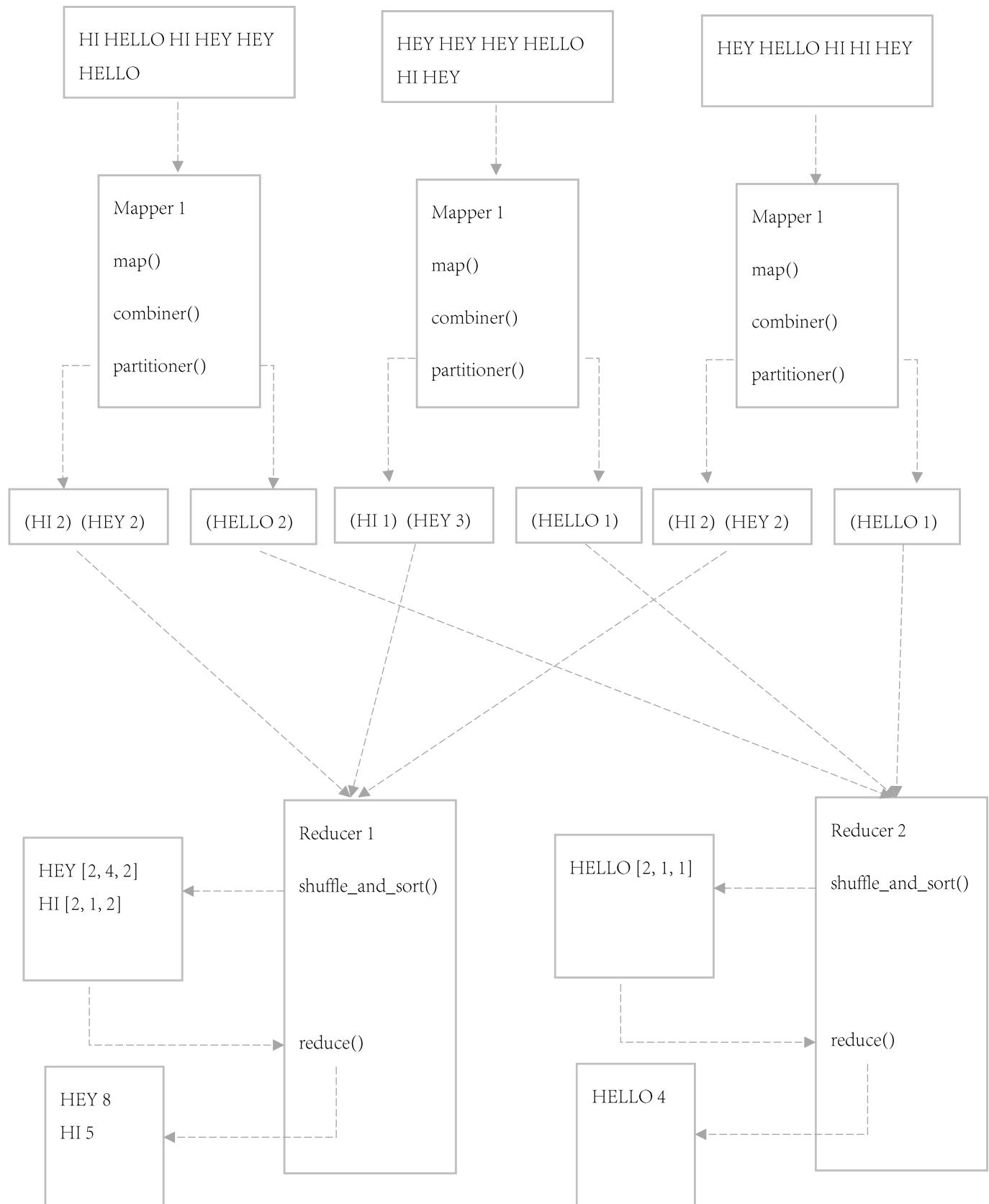
Reducer Output

```
And      4
Hello   1
Hey     1
Hi      1
How     2
Never   1
Now     1
Oragne  3
Umbrella 2
Was     3
When   1
Where   4
Who     2
Whom   1
Why     2
With   1
```

Map Reduce Configuration with Partitioner

```
public class WordCount extends Configured implements Tool {  
  
    @Override  
    public int run(String[] args) throws Exception {  
  
        if (args.length != 2) {  
            System.out.printf(  
                "Usage: %s [generic options] <input dir> <output dir>\n", getClass()  
                    .getSimpleName());  
            ToolRunner.printGenericCommandUsage(System.out);  
            return -1;  
        }  
  
        JobConf conf = new JobConf(getConf(), WordCount.class);  
        conf.setJobName(this.getClass().getName());  
        conf.set("mapred.reduce.tasks", "2");  
  
        FileInputFormat.setInputPaths(conf, new Path(args[0]));  
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));  
  
        conf.setMapperClass(WordMapper.class);  
        conf.setPartitionerClass(VowelPartitioner.class);  
        conf.setReducerClass(SumReducer.class);  
  
        conf.setMapOutputKeyClass(Text.class);  
        conf.setMapOutputValueClass(IntWritable.class);  
  
        conf.setOutputKeyClass(Text.class);  
        conf.setOutputValueClass(IntWritable.class);  
  
        JobClient.runJob(conf);  
        return 0;  
    }  
}
```

Map Reduce With Partitioner Sample Execution



Sample Custom Partitioner Class

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.Partitioner;
public class VowelPartitioner implements Partitioner<Text, IntWritable>{

    @Override
    public int getPartition(Text key, IntWritable value, int numOfPartitions) {
        if (startsWithVowel(key))
            return 0;
        return 1;
    }

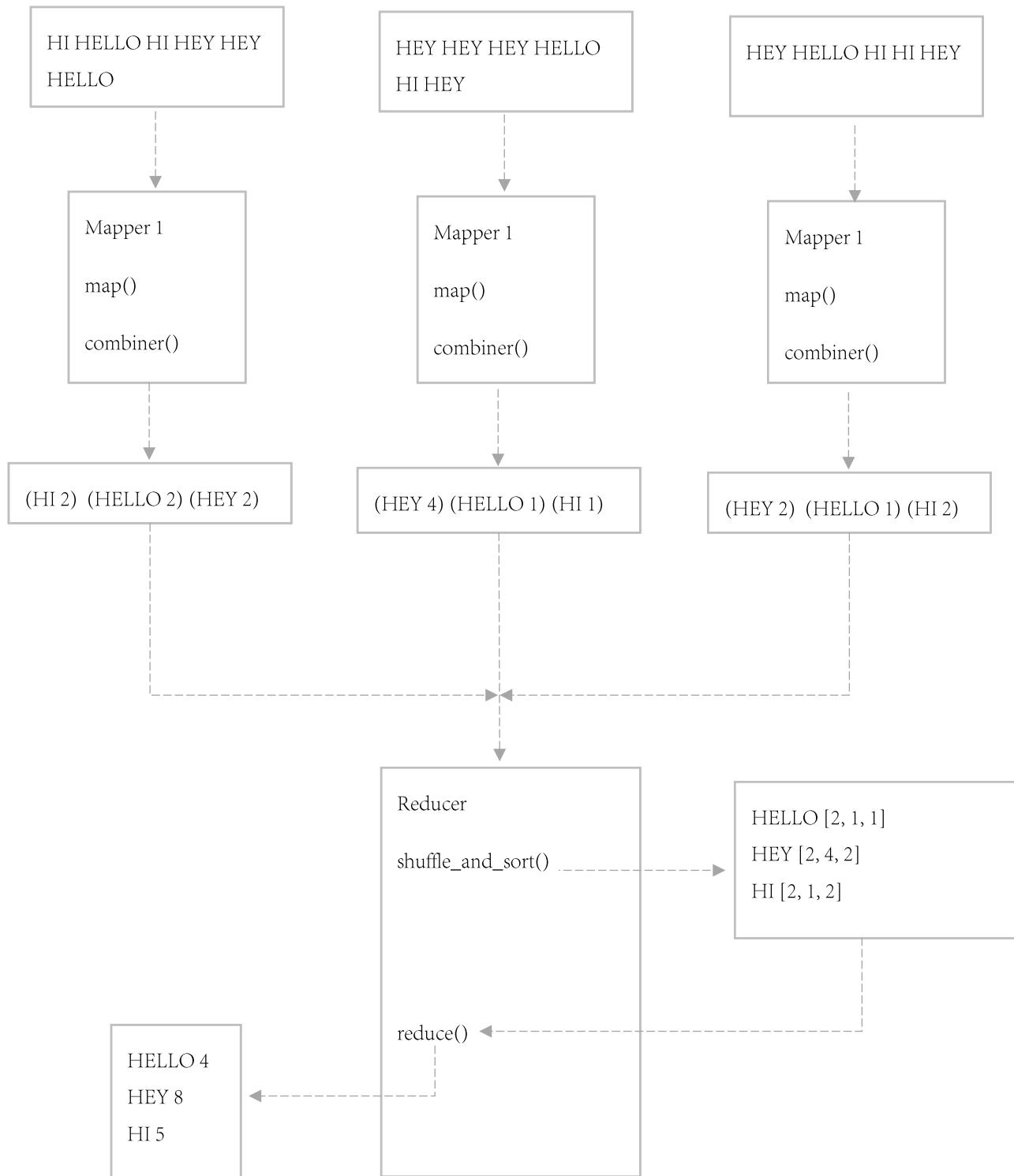
    private boolean startsWithVowel(Text key) {
        if (key == null || key.toString().isEmpty()) return false;
        char c = key.toString().toUpperCase().charAt(0);
        if ((c=='A') || (c=='E') || (c=='I') || (c=='O') || (c=='U')) return true;
        return false;
    }

    @Override
    public void configure(JobConf arg0) {
        // TODO Auto-generated method stub
    }
}
```

Map Reduce Configuration with Combiner

```
public class WordCount extends Configured implements Tool {  
  
    @Override  
    public int run(String[] args) throws Exception {  
  
        if (args.length != 2) {  
            System.out.printf(  
                "Usage: %s [generic options] <input dir> <output dir>\n", getClass()  
                    .getSimpleName());  
            ToolRunner.printGenericCommandUsage(System.out);  
            return -1;  
        }  
  
        JobConf conf = new JobConf(getConf(), WordCount.class);  
        conf.setJobName(this.getClass().getName());  
  
        FileInputFormat.setInputPaths(conf, new Path(args[0]));  
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));  
  
        conf.setMapperClass(WordMapper.class);  
        conf.setCombinerClass(SumReducer.class);  
        conf.setReducerClass(SumReducer.class);  
  
        conf.setMapOutputKeyClass(Text.class);  
        conf.setMapOutputValueClass(IntWritable.class);  
  
        conf.setOutputKeyClass(Text.class);  
        conf.setOutputValueClass(IntWritable.class);  
  
        JobClient.runJob(conf);  
        return 0;  
    }  
}
```

Map Reduce Sample Execution with Combiner



With this we complete the details of the Hadoop Job Process and Map Reduce Programming Model. Now get on to the next chapters and dive deep in to the Hadoop Eco System Tools.

08. Hadoop Eco System Tools

In the last chapter we saw how the basic and only programming model available in Hadoop actually works. But this model had some problems with it when it comes to modelling of real world applications. In this chapter we will run through these disadvantages of Map Reduce and introduce ourselves to its solution.

Why Hadoop?

Hadoop came into existence because we were not able to handle Big Data. And what we do with Big Data is only analysis. So it is very clear that Hadoop was there only for analysis and not for application development.

Hole in Heart of Hadoop Core

One of the core components of Hadoop was Map Reduce. This programming model was entirely and purely Java based. This made understanding and using Map Reduce a tedious affair.

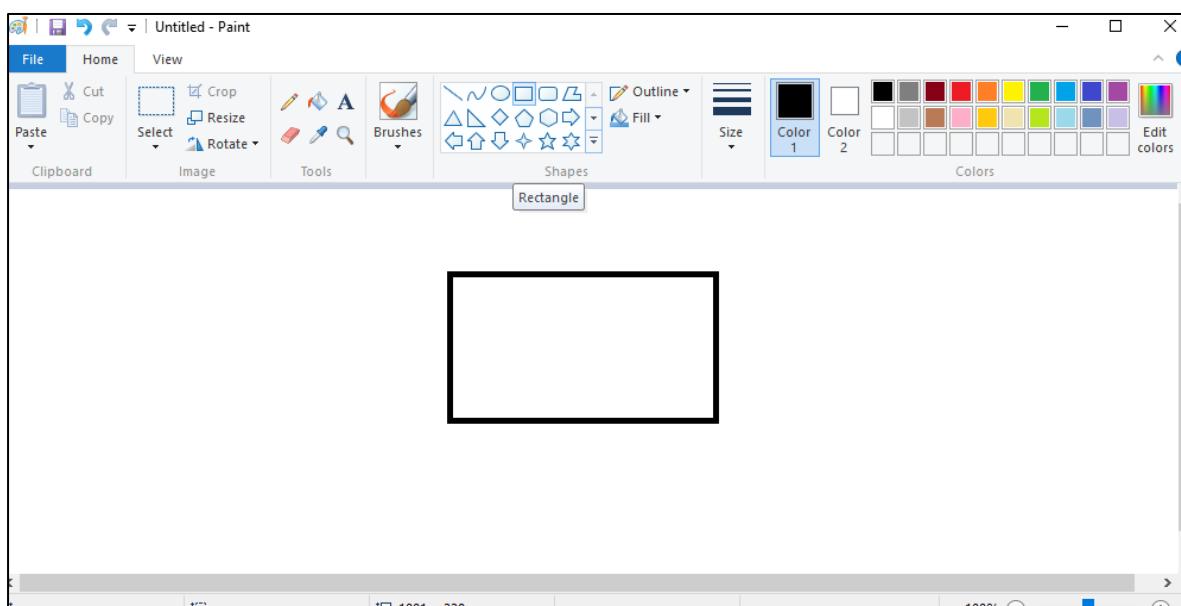
Moreover, as already seen in the previous chapter this model has a default and compulsory phase in between the Map phase and the Reduce phase viz. Shuffle and Sort. The input and output patterns of this internal phase was fixed and the user was not allowed to change any of it nor even its functionality. This means, though we can write code for Map and Reduce functions, still we will always have to restrict their output and input formats in consideration with the Shuffle and Sort phase. This made it even more difficult to define real world problems in the Map Reduce paradigm.

It was also observed that all real world problems cannot be solved through a single Map Reduce program. Sometimes series of Map Reduce programs were needed to be deployed in order to reach to the final solution. This made life with Hadoop even more difficult. With this the Hadoop developers thought that may be

the pre-requisite of Java may actually restrain people from using Hadoop and the problem of Big Data may become even bigger.

Microsoft Paint

What generally happens in [MS Paint](#) if you want to draw a square? You simply click on the square icon in the top panel and then drag, drop and create the square on the canvas. Do you really think that this is the complete process that draws a square? The answer is NO!! At the backend your figure selection triggers some computer graphics like command which captures your clicking coordinates and passes their pixel values as arguments to the corresponding command, upon execution of which the square is drawn for you.



So what have we done in here? We have simply abstracted the details and complexity of computer graphics programs at the backend level and provided a simple user interface at the frontend for ease of use.



Hadoop Eco System

On similar grounds the Hadoop developers thought that why not abstract the complexity of Java at the backend level and give some simple English like commands at the frontend. This is where we came out with a number of tools supporting the Hadoop framework or revolving around the Hadoop cluster by executing Map Reduce programs at the backend and giving the analyst a very simple and/or familiar environment on the frontend.

These tools are called as *Hadoop Ecosystem Projects*. Let's understand certain characteristics of these tools.

Application Specific

Today I had to cook "Chicken Curry". For this I had to do a lot of preparation out of which collecting some spices and churning them into a paste along with coconut and some other ingredients in a mixer grinder was the most time consuming job. I knew that I would be cooking this type of curry a lot of time in future, so in order to save time I created some extra paste for future use. Now whenever I have to cook curry, I don't have to worry about the paste. But the problem that I noticed was, that I cannot use this paste as "Chutney" for "Dosa or Idli". For that I had to make another paste and this paste was not suitable for preparing curry. But from both these paste what I had done was, reduced the preparation time.

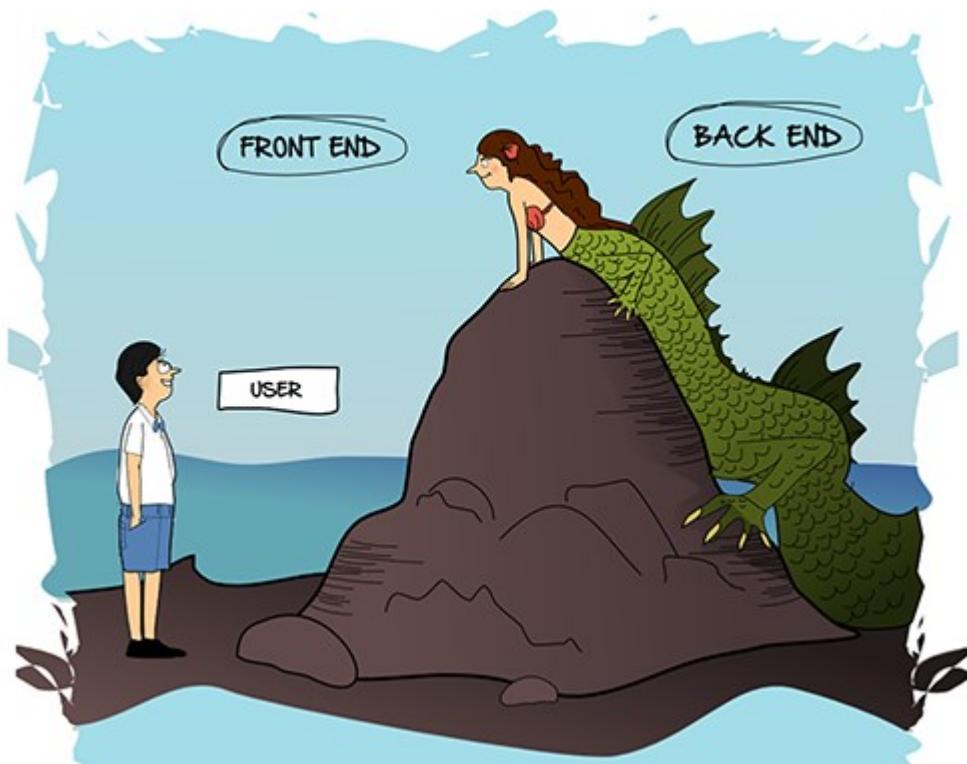
Similarly, companies that used Map Reduce programs to work with several modules of some applications, realized that they would be reusing these modules very often in the future and every time they use these modules they will have to develop these complex Map Reduce programs. Moreover there would be many other companies that may have similar requirements. So to avoid this what they did was bundled all the similar Map Reduce programs together as Library and developed simple English like commands to trigger those library programs from

the frontend. They packaged this along with a processor and execution engine as an [Open Source Hadoop Ecosystem Tool](#).

On similar front, many other developers developed different Ecosystem tools to support different applications that required complex Map Reduce programming, resulting in the evolution of a very large [Hadoop Ecosystem Family](#).

Map Reduce at the Backend

Another feature of these tools is that all of them have MapReduce at the backend. So if you understand the generic backend functionality then it is easy to work with any tool by just understanding the frontend and application.



Client Side Tools

These tools also give ease of deployment as you can directly install them on the Hadoop Client side without the need of playing with your cluster. Only some minor configurations need to be added in your Cluster so as to integrate them together.

Popular Eco System Tools

Generation 1 Tools

Sqoop
(Import Export Tool)

Hive
(SQL on top of Hadoop)

Pig
(PL/SQL on Hadoop)

Flume
(Messaging Service)

Oozie
(Workflow Scheduler)

Generation 2 Tools

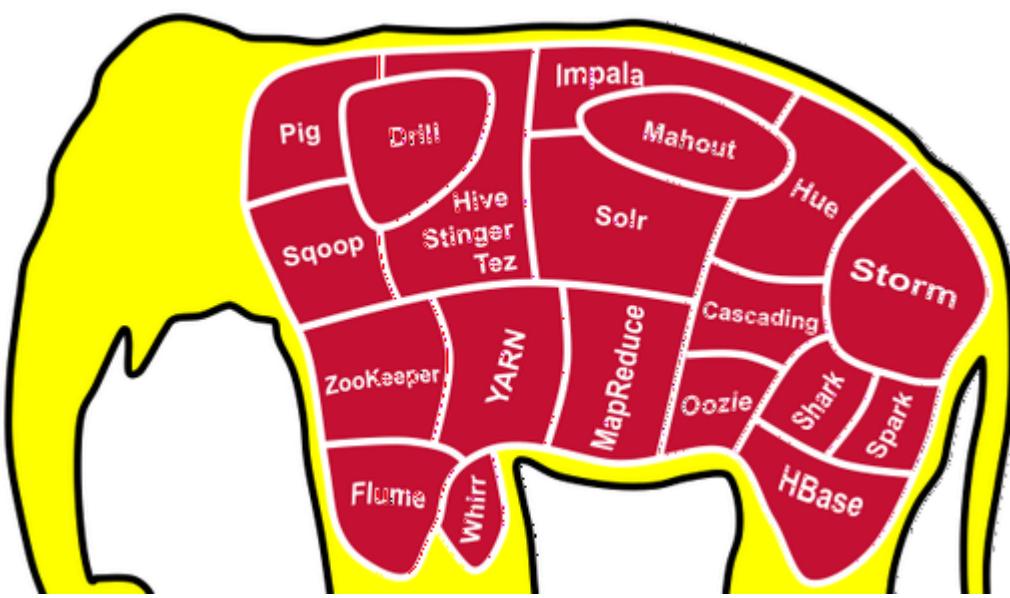
Spark
(Map Reduce + Hive + Pig + Sqoop + Flume + DB + NoSQL)

Kafka
(Messaging Service)

Nifi
(Data Visualization)

Solr
(Search Engine)

Mahout
(Machine Learning)

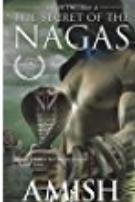


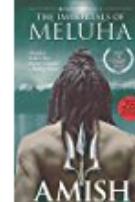
09. Apache Sqoop

Our Web Architecture is basically divided into 2 sections viz. OLTP (Online Transaction Processing) and OLAP (Online Analytical Processing). The Transactional data is stored in the databases and the data required for analysis is stored in Data Warehouses (This is where Hadoop sits). So one of the major tasks in the industry deals in transferring the data between the Database and the Hadoop System.

For instance visualize the working of [Amazon Recommendation Engine](#):

Frequently bought together


+

+


Total price: ₹565.00

[Add all three to Cart](#)

This item: The Secret Of The Nagas (Shiva Trilogy-2) by Amish Tripathi Paperback ₹178.00
 The Oath of the Vayuputras (Shiva Trilogy) by Amish Tripathi Paperback ₹227.00
 The Immortals of Meluha (Shiva Trilogy) by Amish Paperback ₹160.00

When you click on a product you get a list of recommended products. So where do we actually get this from? The Databases, right? The databases are connected to the Website, so whenever we click on a product a query asking for the recommended products is fired on the database and the corresponding list is displayed back on the Website. But how do we conclude the recommendations in the database? What we do is, in the Data Warehouse (Hadoop HDFS) we collect years of data in terms of mapping between a product and its buyer (this becomes the product) and the list of other products that the buyer has brought or searched for (this becomes the recommendation). Then some more analysis is applied and we finally get a list of recommendations for all the products. This list is then uploaded into the Database which fetches the recommendations for us.

One of the traditional approach to handle this requirement is to write MapReduce jobs and get our work done. But this involves a lot of things like establishing a connection, Understanding the Schema, Understanding and defining the Delimiters, Conversion into Text format etc. which makes it a very tedious job. So in order to abstract the complexity and provide a simple interface to work with, Hadoop Ecosystem introduces an Import Export tool on Hadoop called [Sqoop](#).

Insights of Sqoop

Sqoop is an import export tool between HDFS and RDBMS. Almost all the RDBMS make use of [JDBC](#) to communicate with third parties as well as with different database. As Sqoop also uses the same interface for communication, it can definitely communicate with all the RDBMS. The commands for all the databases is the same. You only need to check whether you have the proper drivers installed on your Client machines.

For databases, Sqoop will read the table row-by-row into HDFS. The output of this import process is a set of files containing a copy of the imported table or datasets. The import process is performed in parallel. For this reason, the output will be in multiple files. These files may be [delimited text files, or binary Avro](#) or [Sequence Files](#) containing serialized record data.

After manipulating the imported records you may have a result data set which you can then *export* back to the relational database. Sqoop's export process will read a set of delimited text files from HDFS in parallel, parse them into records, and insert them as new rows in a target database table, for consumption by external applications or users.

Most aspects of the import, code generation, and export processes can be customized. For databases, you can control the specific row range or columns

imported. You can specify particular delimiters and escape characters for the file-based representation of the data, as well as the file format used.

Now lets demonstrate some operations in available in sqoop

Step Description:

Login to the MySQL shell using username as root and password as root. Check for available databases and create a database by name wisdom_db.

Commands:

[jayantmohite@localhost] \$ mysql -u <enter username as root> -p

Enter password: <enter password as root>

(you won't be able to see the password getting typed. But it is recorded by the terminal. So don't worry and simply type without any typing mistake. This will take you to the MySQL client shell.)

mysql> show databases;

(This will give you a list of all available databases in the server)

mysql> create database <enter database name as wisdom_db>;

Step Visualization:

```

Applications Places Terminal Fri 13:48
jayantmohite@localhost:~ - x
File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 5.6.39 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| wisdom |
+-----+
4 rows in set (0.04 sec)

mysql> create database wisdom db;
Query OK, 1 row affected (0.01 sec)

mysql> exit;
Bye
[jayantmohite@localhost ~]$ 

```

Step Description:

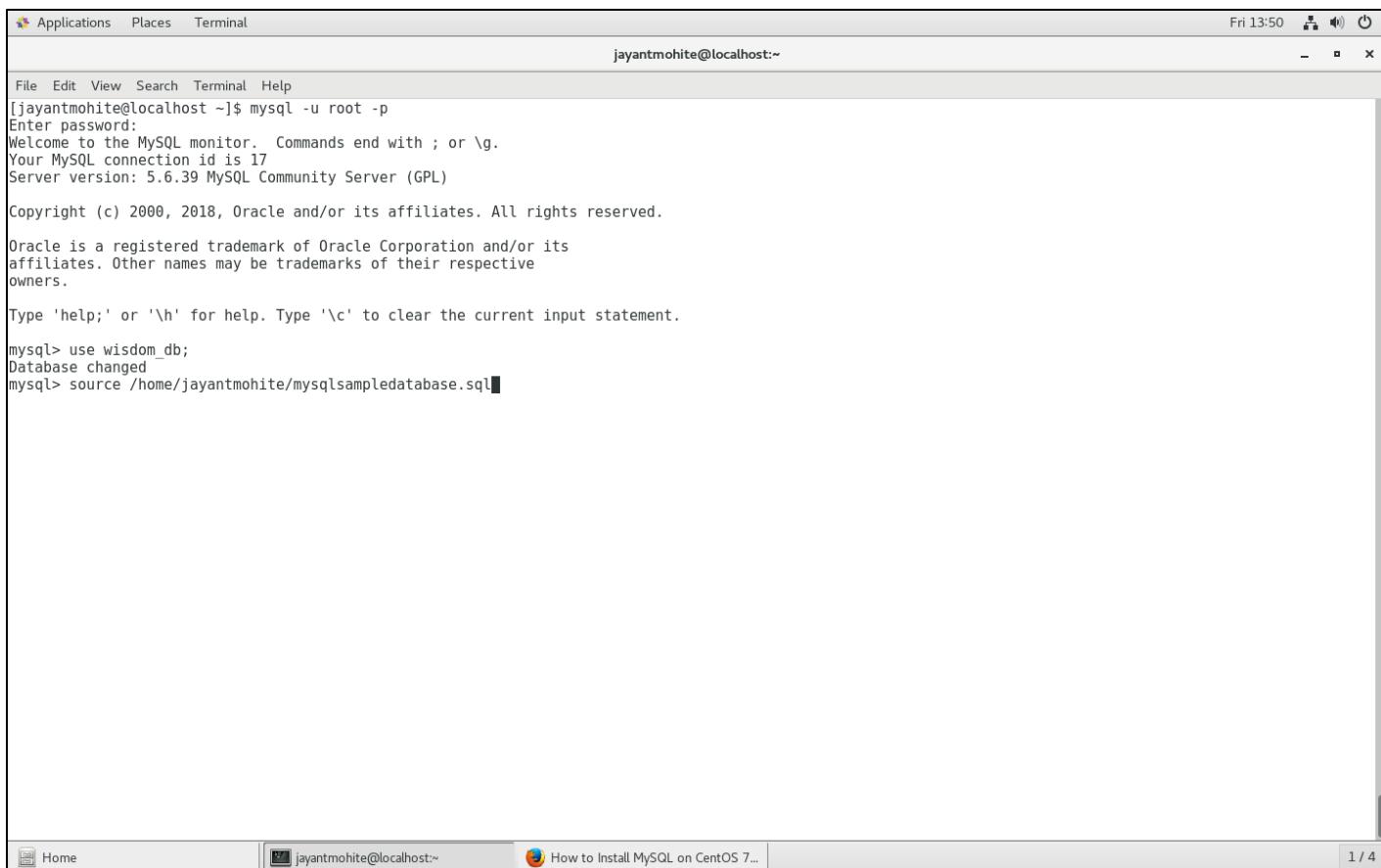
Select the target database as wisdom_db which we created in the above step and import the SQL file by name mysqlsampledatabase.sql (provided at the end of this chapter) to create the sample tables for our use.

Commands:

```
mysql> use <enter database name as wisdom_db>
```

```
mysql> source <file location>/mysqlsampledatabase.sql
```

Step Visualization:



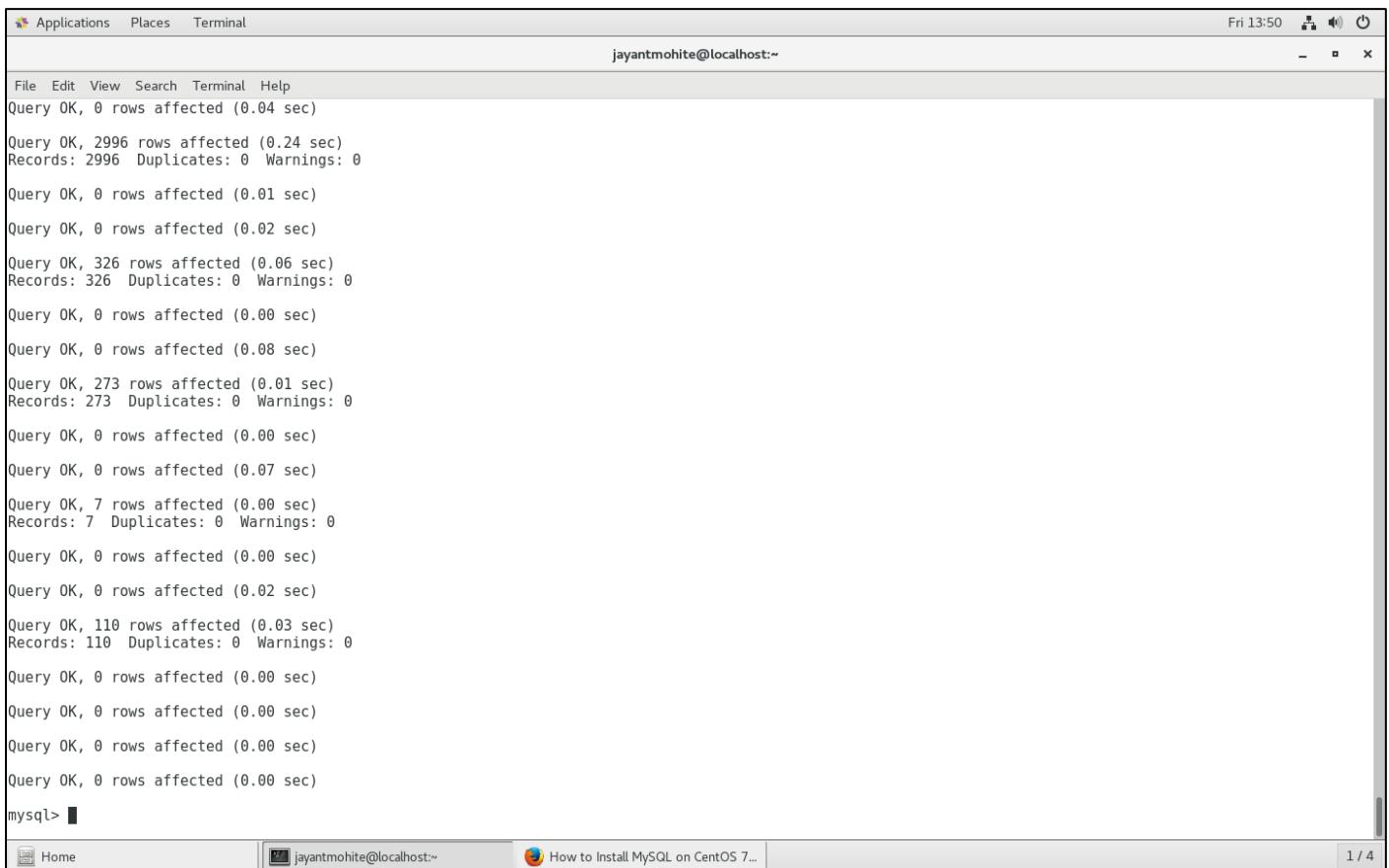
```
[jayantmohite@localhost ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 5.6.39 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use wisdom_db;
Database changed
mysql> source /home/jayantmohite/mysqlsampledatabase.sql
```



The screenshot shows a terminal window with the following details:

- Top bar: Applications, Places, Terminal, Fri 13:50, jayantmohite@localhost:~
- Menu bar: File, Edit, View, Search, Terminal, Help
- Output area:
 - Query OK, 0 rows affected (0.04 sec)
 - Query OK, 2996 rows affected (0.24 sec)
Records: 2996 Duplicates: 0 Warnings: 0
 - Query OK, 0 rows affected (0.01 sec)
 - Query OK, 0 rows affected (0.02 sec)
 - Query OK, 326 rows affected (0.06 sec)
Records: 326 Duplicates: 0 Warnings: 0
 - Query OK, 0 rows affected (0.00 sec)
 - Query OK, 0 rows affected (0.08 sec)
 - Query OK, 273 rows affected (0.01 sec)
Records: 273 Duplicates: 0 Warnings: 0
 - Query OK, 0 rows affected (0.00 sec)
 - Query OK, 0 rows affected (0.07 sec)
 - Query OK, 7 rows affected (0.00 sec)
Records: 7 Duplicates: 0 Warnings: 0
 - Query OK, 0 rows affected (0.00 sec)
 - Query OK, 0 rows affected (0.02 sec)
 - Query OK, 110 rows affected (0.03 sec)
Records: 110 Duplicates: 0 Warnings: 0
 - Query OK, 0 rows affected (0.00 sec)
 - mysql> [REDACTED]
- Bottom bar: Home, jayantmohite@localhost:~, How to Install MySQL on CentOS 7..., 1 / 4

Step Description:

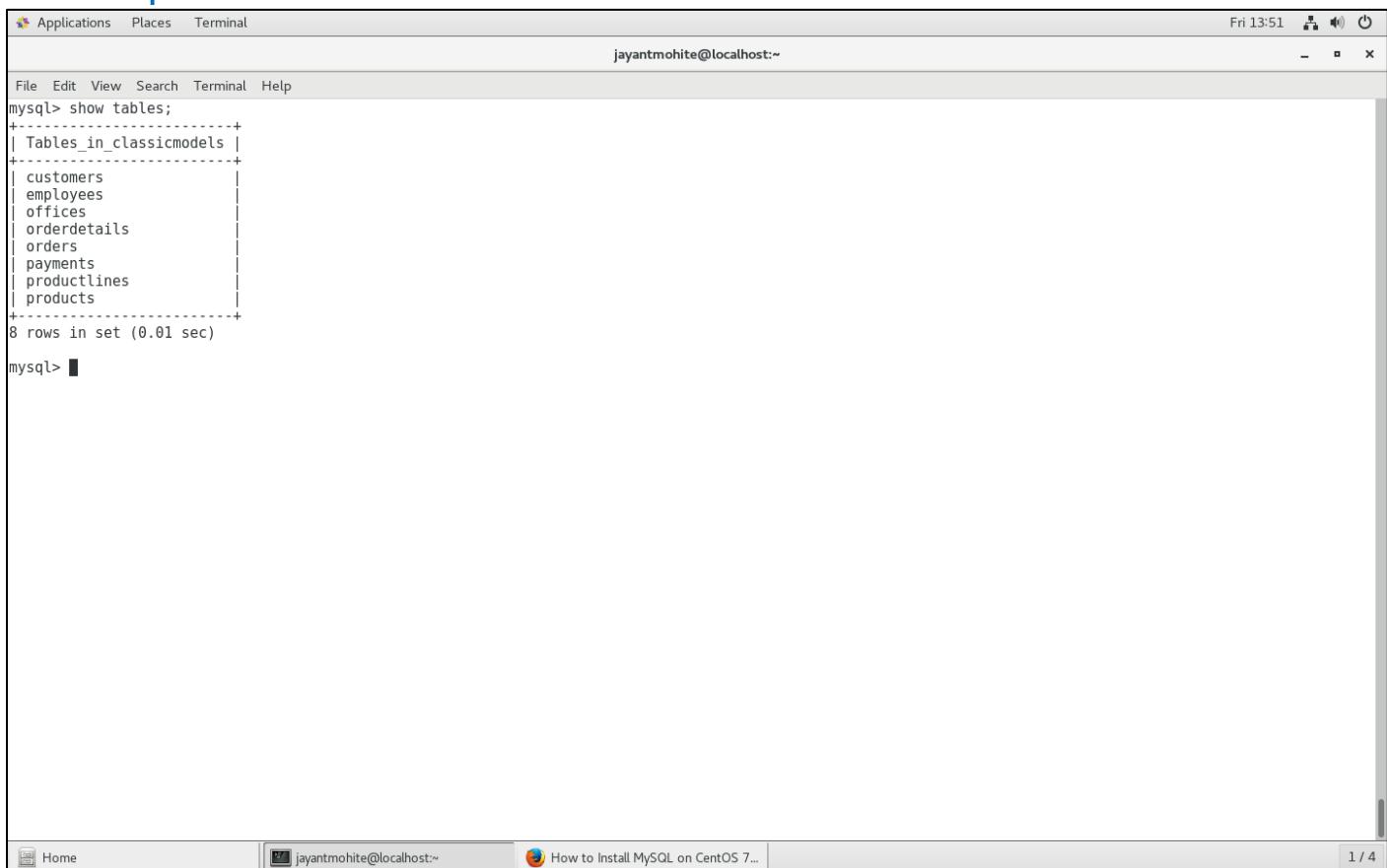
Check if the import was done successfully by listing all the tables. You should be able to see the following tables

- customers
- employees
- offices
- orderdetails
- orders
- payments
- productlines
- products

Commands:

```
mysql> show tables;
```

Step Visualization:



A screenshot of a Linux terminal window titled "Terminal". The window shows the MySQL command-line interface with the user "jayantmohite" connected to "localhost". The command "show tables;" is run, displaying a list of 8 tables in the "classicmodels" database: customers, employees, offices, orderdetails, orders, payments, productlines, and products. The output ends with "8 rows in set (0.01 sec)". The terminal window has a standard Linux desktop interface with icons for Applications, Places, and Terminal at the top, and a date/time stamp "Fri 13:51" in the top right. The bottom of the window shows a navigation bar with "Home", "jayantmohite@localhost:~" (the current tab), and "How to Install MySQL on CentOS 7...". A page number "1 / 4" is visible in the bottom right corner of the navigation bar.

```
Applications Places Terminal
Fri 13:51
jayantmohite@localhost:~
File Edit View Search Terminal Help
mysql> show tables;
+-----+
| Tables_in_classicmodels |
+-----+
| customers
| employees
| offices
| orderdetails
| orders
| payments
| productlines
| products
+-----+
8 rows in set (0.01 sec)

mysql> |
```

Step Description:

You can use different tables and see how different commands in Sqoop work. For this lab session, we will be making use of the table employees. So let's try to get a glimpse of how this table looks.

Commands:

`mysql> select count(*) from <enter table name as employees>;`

(this will give you the count of the total records in the table)

`mysql> describe <enter table name as employees>;`

(this will show you the schema or structure of the table)

`mysql> select * from <enter table name as employees> limit <enter integer value as 5>;`

(this will give you the first 5 records from the employees table)

Step Visualization:

```

Applications Places Terminal Fri 13:52
jayantmohite@localhost:~>

File Edit View Search Terminal Help
mysql> select count(*) from employees;
+-----+
| count(*) |
+-----+
| 23 |
+-----+
1 row in set (0.01 sec)

mysql> describe employees;
+-----+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| employeeNumber | int(11) | NO   | PRI  | NULL    |       |
| lastName     | varchar(50)| NO  |       | NULL    |       |
| firstName    | varchar(50)| NO  |       | NULL    |       |
| extension    | varchar(10) | NO  |       | NULL    |       |
| email        | varchar(100) | NO  |       | NULL    |       |
| officeCode   | varchar(10) | NO  | MUL  | NULL    |       |
| reportsTo    | int(11)    | YES | MUL  | NULL    |       |
| jobTitle     | varchar(50)| NO  |       | NULL    |       |
+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0.03 sec)

mysql> select * from employees limit 5;
+-----+-----+-----+-----+-----+-----+-----+
| employeeNumber | lastName | firstName | extension | email          | officeCode | reportsTo | jobTitle |
+-----+-----+-----+-----+-----+-----+-----+
| 1002 | Murphy  | Diane    | x5800    | dmurphy@classicmodelcars.com | 1          | NULL      | President |
| 1056 | Patterson | Mary     | x4611    | mpatterso@classicmodelcars.com | 1          | 1002      | VP Sales   |
| 1076 | Firrelli  | Jeff     | x9273    | jfirrelli@classicmodelcars.com | 1          | 1002      | VP Marketing |
| 1088 | Patterson | William  | x4871    | wpatterson@classicmodelcars.com | 6          | 1056      | Sales Manager (APAC) |
| 1102 | Bondur   | Gerard   | x5408    | gbondur@classicmodelcars.com | 4          | 1056      | Sale Manager (EMEA) |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> 
```

Step Description:

Browse the HDFS Web Browser. Here you can see that there is no directory by name employees. Rather what we have is the two directories that we created while installing Hadoop in the previous chapters.

This will help you to understand the changes that will be brought to effect by executing the sqoop commands.

In case you have created any more directories, you can always delete them by using the following command:

```
[jayantmohite@localhost] $ hadoop fs -rmdir /user/jayantmohite/*
```

This command will delete all directories in the HDFS user location including the directories listed in the visualization below.

Step Visualization:

The screenshot shows a Firefox browser window titled "Browsing HDFS - Mozilla Firefox". The address bar shows "localhost:50070/explorer.html#/user/jayantmohite". The main content area is titled "Browse Directory" and shows the contents of the "/user/jayantmohite" directory. There are two entries in the table:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	jayantmohite	supergroup	0 B	Fri Mar 02 04:22:34 -0800 2018	0	0 B	input
drwxr-xr-x	jayantmohite	supergroup	0 B	Fri Mar 02 04:27:20 -0800 2018	0	0 B	output23

At the bottom of the page, it says "Hadoop, 2017."

The browser's status bar shows "jayantmohite@localhost:~" and "Browsing HDFS - Mozilla Firefox".

Step Description:

In this step we will be import the table employees from the MySQL database server installed in our client machine. This is called as Sqoop Default Import command in Sqoop terminology. We will use the following parameters:

Database Connector: jdbc (Java DataBase Connector)

Database Server: mysql

Installed on Server (hostname): localhost

Database: classicmodels

Table (to import data from): employees

Username (credentials for MySQL server validation): root

Password (credentials for MySQL server validation): root

Commands:

```
[jayantmohite@localhost] $ sqoop import --connect  
jdbc:mysql://localhost/classicmodels --table employees --username root --  
password root
```

Step Visualization:

```
[jayantmohite@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/classicmodels --table employees --username root --password root
Warning: /usr/lib/sqoop/..hbbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/..accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/03/02 15:12:43 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:12:43 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:12:43 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:12:43 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:12:44 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:12:45 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:12:45 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/052cae84fbb2ab6e726540d693b9cd0f/employees.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:12:55 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/052cae84fbb2ab6e726540d693b9cd0f/employees.jar
18/03/02 15:12:55 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/03/02 15:12:55 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/03/02 15:12:55 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/03/02 15:12:55 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/03/02 15:12:55 INFO mapreduce.ImportJobBase: Beginning import of employees
18/03/02 15:13:12 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:13:12 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`employeeNumber`), MAX(`employeeNumber`) FROM `employees`
18/03/02 15:13:12 INFO db.IntegerSplitter: Split size: 175; Num splits: 4 from: 1002 to: 1702
18/03/02 15:13:17 INFO mapred.JobClient: Running job: job_201803021449_0001
18/03/02 15:13:19 INFO mapred.JobClient: map 0% reduce 0%
```

jayantmohite@localhost:~ Mozilla Firefox Home 1 / 4

Step Observations List:

The given job is split into tasks based on the value of the primary key.

Backend code of the sqoop operation triggers a Map Reduce Program

Number of tasks in which the job is split is 4.

This means 4 mappers will be used in this job.

For Example if the maximum value of the primary key field is 12, then mapper 1 will fetch records having the value of primary key from 1 to 3; the 2nd mapper will fetch values 4 to 6; the 3rd will fetch the values from 7 to 9 and the 4th will fetch values from 10 to 12.

This also means that there is a possibility that the size of output files produced by each mapper may differ. For example there are no records in the range 4 to 6; In this case mapper 2 will produce an empty file.

Step Visualization:

```

Applications Places Terminal Fri 15:14   jayantmohite@localhost:~
File Edit View Search Terminal Help
18/03/02 15:12:55 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/03/02 15:12:55 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/03/02 15:12:55 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/03/02 15:12:55 INFO mapreduce.ImportJobBase: Beginning import of employees
18/03/02 15:13:12 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:13:12 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`employeeNumber`), MAX(`employeeNumber`) FROM `employees`
18/03/02 15:13:12 INFO db.IntegerSplitter: Split size: 175; Num splits: 4 from: 1002 to: 1702
18/03/02 15:13:17 INFO mapred.JobClient: Running job: job_201803021449_0001
18/03/02 15:13:19 INFO mapred.JobClient: map 0% reduce 0%
18/03/02 15:14:09 INFO mapred.JobClient: map 50% reduce 0%
18/03/02 15:14:32 INFO mapred.JobClient: map 100% reduce 0%
18/03/02 15:14:41 INFO mapred.JobClient: Job complete: job_201803021449_0001
18/03/02 15:14:42 INFO mapred.JobClient: Counters: 23
18/03/02 15:14:42 INFO mapred.JobClient: File System Counters
18/03/02 15:14:42 INFO mapred.JobClient: FILE: Number of bytes read=0
18/03/02 15:14:42 INFO mapred.JobClient: FILE: Number of bytes written=1533564
18/03/02 15:14:42 INFO mapred.JobClient: FILE: Number of read operations=0
18/03/02 15:14:42 INFO mapred.JobClient: FILE: Number of large read operations=0
18/03/02 15:14:42 INFO mapred.JobClient: FILE: Number of write operations=0
18/03/02 15:14:42 INFO mapred.JobClient: HDFS: Number of bytes read=513
18/03/02 15:14:42 INFO mapred.JobClient: HDFS: Number of bytes written=1661
18/03/02 15:14:42 INFO mapred.JobClient: HDFS: Number of read operations=4
18/03/02 15:14:42 INFO mapred.JobClient: HDFS: Number of large read operations=0
18/03/02 15:14:42 INFO mapred.JobClient: HDFS: Number of write operations=4
18/03/02 15:14:42 INFO mapred.JobClient: Job Counters
18/03/02 15:14:42 INFO mapred.JobClient: Launched map tasks=4
18/03/02 15:14:42 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=124007
18/03/02 15:14:42 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=0
18/03/02 15:14:42 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0
18/03/02 15:14:42 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0
18/03/02 15:14:42 INFO mapred.JobClient: Map-Reduce Framework
18/03/02 15:14:42 INFO mapred.JobClient: Map input records=23
18/03/02 15:14:42 INFO mapred.JobClient: Map output records=23
18/03/02 15:14:42 INFO mapred.JobClient: Input split bytes=513
18/03/02 15:14:42 INFO mapred.JobClient: Spilled Records=0
18/03/02 15:14:42 INFO mapred.JobClient: CPU time spent (ms)=7590
18/03/02 15:14:42 INFO mapred.JobClient: Physical memory (bytes) snapshot=355659776
18/03/02 15:14:42 INFO mapred.JobClient: Virtual memory (bytes) snapshot=8027168768
18/03/02 15:14:42 INFO mapred.JobClient: Total committed heap usage (bytes)=65273856
18/03/02 15:14:42 INFO mapreduce.ImportJobBase: Transferred 1.6221 KB in 105.1684 seconds (15.7937 bytes/sec)
18/03/02 15:14:42 INFO mapreduce.ImportJobBase: Retrieved 23 records.
[jayantmohite@localhost ~]$ 
```

jayantmohite@localhost:~ Mozilla Firefox Home 1 / 4

Step Observations List:

No Reducer is used in this operation. As it's a simple copy paste job and no aggregation is involved.

Step Visualization:

The screenshot shows a Mozilla Firefox browser window titled "Browsing HDFS - Mozilla Firefox". The address bar displays "localhost:50070/explorer.html#/user/jayantmohite". The main content area is titled "Browse Directory" and shows a single file named "employees" located at "/user/jayantmohite". The file details are as follows:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	jayantmohite	supergroup	0 B	Fri Mar 02 15:14:40 -0800 2018	0	0 B	employees

At the bottom left, the terminal prompt is "jayantmohite@localhost:~". At the bottom right, the status bar shows "1 / 4".

Step Observation List:

By default this sqoop commands create the output directory with the same name as that of the table.

The processing jar file is created default by sqoop with the name same as the name of the table.

The structure of the output directory is same as it used to be in case of the Map Reduce programs.

There are 4 output files (as the number of mappers used were 4 and numbers of reducers used was 0)

Size of all output files varies (reason is explained in previous observations).

Step Visualization:

The screenshot shows the Hadoop Map/Reduce Administration interface on a Firefox browser. The URL is localhost:50030/jobtracker.jsp. The page displays the following information:

- Cluster Summary (Heap Size is 15.56 MB/966.69 MB):**

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Excluded Nodes
0	0	1	1	0	0	0	0	2	2	4.00	0	0
- Scheduling Information:**

Queue Name	State	Scheduling Information
default	running	N/A
- Running Jobs:** A table showing one job entry: **job_201803021449_0001** (NORMAL, jayantmohite, employees.jar, 100.00% complete, 4 map tasks completed, 0 reduce tasks completed, NA diagnostic info).
- Completed Jobs:** A table showing one job entry: **job_201803021449_0001** (NORMAL, jayantmohite, employees.jar, 100.00% complete, 4 map tasks completed, 0 reduce tasks completed, NA diagnostic info).
- Retired Jobs:** A table showing one job entry: **job_201803021449_0001** (NORMAL, jayantmohite, employees.jar, 100.00% complete, 4 map tasks completed, 0 reduce tasks completed, NA diagnostic info).
- Local Logs:** A log entry from jayantmohite@localhost:~ showing the browser title as "localhost Hadoop Map/Reduce Admin...".

Step Visualization:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	jayantmohite	supergroup	0 B	Fri Mar 02 15:14:40 -0800 2018	1	128 MB	_SUCCESS
drwxr-xr-x	jayantmohite	supergroup	0 B	Fri Mar 02 15:13:17 -0800 2018	0	0 B	_logs
-rw-r--r--	jayantmohite	supergroup	614 B	Fri Mar 02 15:14:06 -0800 2018	1	128 MB	part-m-00000
-rw-r--r--	jayantmohite	supergroup	361 B	Fri Mar 02 15:14:06 -0800 2018	1	128 MB	part-m-00001
-rw-r--r--	jayantmohite	supergroup	284 B	Fri Mar 02 15:14:29 -0800 2018	1	128 MB	part-m-00002
-rw-r--r--	jayantmohite	supergroup	402 B	Fri Mar 02 15:14:29 -0800 2018	1	128 MB	part-m-00003

Hadoop, 2017.

Step Visualization:

```

1002,Murphy,Diane,x5800,dmurphy@classicmodelcars.com,1,null,President
1056,Patterson,Mary,x4611,mpatterso@classicmodelcars.com,1,1002,VP Sales
1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,VP Marketing
1088,Patterson,William,x4871,wpatterson@classicmodelcars.com,6,1056,Sales Manager (APAC)
1102,Bondur,Gerard,x5408,gbondur@classicmodelcars.com,4,1056,Sale Manager (EMEA)
1143,Bow,Anthony,x5428,abow@classicmodelcars.com,1,1056,Sales Manager (NA)
1165,Jennings,Leslie,x3291,ljenningss@classicmodelcars.com,1,1143,Sales Rep
1166,Thompson,Leslie,x4065,lthompson@classicmodelcars.com,1,1143,Sales Rep

```

Step Observation List:

If you download either of the output files, you will be able to observe the following

Every output contains some number of records based on certain range

The default delimiter used by which every field is separated from the other is comma.

Step Description:

In this step we will override the default behavior of the sqoop command and will specify our own target directory for dumping the output files.

Commands:

```
[jayantmohite@localhost] $ sqoop import --connect
jdbc:mysql://localhost/classicmodels --table employees --target-dir sqoop1 --
username root -password root
```

New Parameter Used:

--target-directory

(the value of this parameter will become the output directory)

Step Visualization:

```
[jayantmohite@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/classicmodels --table employees --target-dir sqoop1 --username root --password root
Warning: /usr/lib/sqoop/..../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/..../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/03/02 15:18:31 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:18:31 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:18:32 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:18:32 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:18:33 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:18:33 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:18:33 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/0a4f975a0613e0dbc4ba6724c1f4ca0/employees.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:18:43 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/0a4f975a0613e0dbc4ba6724c1f4ca0/employees.jar
18/03/02 15:18:44 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/03/02 15:18:44 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/03/02 15:18:44 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/03/02 15:18:44 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/03/02 15:18:44 INFO mapreduce.ImportJobBase: Beginning import of employees
18/03/02 15:19:02 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:19:02 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`employeeNumber`), MAX(`employeeNumber`) FROM `employees`
18/03/02 15:19:02 INFO db.IntegerSplitter: Split size: 175; Num splits: 4 from: 1002 to: 1702
18/03/02 15:19:03 INFO mapred.JobClient: Running job: job_201803021449_0002
18/03/02 15:19:04 INFO mapred.JobClient: map 0% reduce 0%
```

Browsing HDFS - Mozilla Firefox

localhost Hadoop Ma...

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/user/jayantmohite Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	jayantmohite	supergroup	0 B	Fri Mar 02 15:14:40 -0800 2018	0	0 B	employees
drwxr-xr-x	jayantmohite	supergroup	0 B	Fri Mar 02 15:20:52 -0800 2018	0	0 B	sqoop1

Hadoop, 2017.

jayantmohite@localhost:~ Browsing HDFS - Mozilla Firefox 1 / 4

Step Description:

In this step we will change the default delimiter used and specify a delimiter of our choice. In this case we are specifying the delimiter as tab.

Commands:

```
[jayantmohite@localhost] $ sqoop import --connect
jdbc:mysql://localhost/classicmodels --table employees --target-dir sqoop2 --
fields-terminated-by '\t' --username root -password root
```

New Parameter Used:

--fields-terminated-by

(the value of this parameter will specify the special character that is to be used as a separator between the fields. '\t' represents tab which is equal to 4 spaces)

Step Visualization:

```
jayantmohite@localhost:~$ sqoop import --connect jdbc:mysql://localhost/classicmodels --table employees --target-dir sqoop2 --fields-terminated-by '\t' --username root -password root
Warning: /usr/lib/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/03/02 15:22:07 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:22:07 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:22:07 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:22:07 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:22:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:22:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:22:08 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/ee5da0934ddf64ca4d216252b41be087/employees.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:22:18 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/ee5da0934ddf64ca4d216252b41be087/employees.jar
18/03/02 15:22:18 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/03/02 15:22:18 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/03/02 15:22:18 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/03/02 15:22:18 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/03/02 15:22:18 INFO mapreduce.ImportJobBase: Beginning import of employees
18/03/02 15:22:35 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:22:35 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`employeeNumber`), MAX(`employeeNumber`) FROM `employees`
18/03/02 15:22:35 INFO db.IntegerSplitter: Split size: 175; Num splits: 4 from: 1002 to: 1702
18/03/02 15:22:37 INFO mapred.JobClient: Running job: job_201803021449_0003
18/03/02 15:22:38 INFO mapred.JobClient: map 0% reduce 0%
```

The screenshot shows a terminal window with the following details:

- Top bar: Applications, Places, Text Editor, Fri 15:30, Save, Minimize, Maximize, Close.
- Title bar: part-m-00000(1) (~/Downloads)
- Content area:

ID	First Name	Last Name	Extension	Email	Office ID	Office	Report To	Job Title
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	null	President	
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales	
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing	
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)	
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)	
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)	
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep	
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep	
- Bottom status bar: Plain Text, Tab Width: 8, Ln 1, Col 1, INS, jayantmohite@localhost:~, Browsing HDFS - Mozilla Firefox, part-m-00000(1) (~/Downloads) - ..., 1 / 4

Step Description:

In this step we will make use of only one mapper for performing our task. This is also called as Swoop Sequential Import.

This is one of the options available when you are working with a table that does not have a primary key.

Commands:

```
[jayantmohite@localhost] $ sqoop import --connect
jdbc:mysql://localhost/classicmodels --table employees --target-dir sqoop3 -m 1 --
username root -password root
```

New Parameter Used:

-m

(the value of this parameter specifies the number of mappers or the degree of parallelism to be applied in the job execution. In this case we are specifying it as 1 which is called as sequential import)

Step Visualization:

```
[jayantmohite@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/classicmodels --table employees -m 1 --target-dir sqoop3 --username root --password root
Warning: /usr/lib/sqoop/..hbbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/..accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/03/02 15:31:39 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:31:39 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:31:40 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:31:40 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:31:42 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:31:42 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:31:42 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/f4fc8f8325f087b335ac9210e44e08e9/employees.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:31:50 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/f4fc8f8325f087b335ac9210e44e08e9/employees.jar
18/03/02 15:31:50 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/03/02 15:31:50 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/03/02 15:31:50 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/03/02 15:31:50 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/03/02 15:31:50 INFO mapreduce.ImportJobBase: Beginning import of employees
18/03/02 15:32:08 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:32:10 INFO mapred.JobClient: Running job: job_201803021449_0004
18/03/02 15:32:11 INFO mapred.JobClient: map 0% reduce 0%
18/03/02 15:32:39 INFO mapred.JobClient: map 100% reduce 0%
```

```

File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/classicmodels --table employees -m 1 --target-dir sqoop3 --username root --password root
Warning: /usr/lib/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/03/02 15:31:39 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:31:39 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:31:40 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:31:40 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:31:42 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:31:42 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:31:42 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/f4fc8f8325f087b335ac9210e44e08e9/employees.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:31:50 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/f4fc8f8325f087b335ac9210e44e08e9/employees.jar
18/03/02 15:31:50 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/03/02 15:31:50 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/03/02 15:31:50 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/03/02 15:31:50 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/03/02 15:31:50 INFO mapreduce.ImportJobBase: Beginning import of employees
18/03/02 15:32:08 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:32:10 INFO mapred.JobClient: Running job: job_201803021449_0004
18/03/02 15:32:11 INFO mapred.JobClient: map 0% reduce 0%
18/03/02 15:32:39 INFO mapred.JobClient: map 100% reduce 0%
18/03/02 15:32:48 INFO mapred.JobClient: Job complete: job_201803021449_0004
18/03/02 15:32:48 INFO mapred.JobClient: Counters: 23
18/03/02 15:32:49 INFO mapred.JobClient: File System Counters
18/03/02 15:32:49 INFO mapred.JobClient: FILE: Number of bytes read=0
18/03/02 15:32:49 INFO mapred.JobClient: FILE: Number of bytes written=383932
18/03/02 15:32:49 INFO mapred.JobClient: FILE: Number of read operations=0
18/03/02 15:32:49 INFO mapred.JobClient: FILE: Number of large read operations=0
18/03/02 15:32:49 INFO mapred.JobClient: FILE: Number of write operations=0
18/03/02 15:32:49 INFO mapred.JobClient: HDFS: Number of bytes read=87
18/03/02 15:32:49 INFO mapred.JobClient: HDFS: Number of bytes written=1661
18/03/02 15:32:49 INFO mapred.JobClient: HDFS: Number of read operations=1
18/03/02 15:32:49 INFO mapred.JobClient: HDFS: Number of large read operations=0
18/03/02 15:32:49 INFO mapred.JobClient: HDFS: Number of write operations=1
18/03/02 15:32:49 INFO mapred.JobClient: Job Counters
18/03/02 15:32:49 INFO mapred.JobClient: Launched map tasks=1
18/03/02 15:32:49 INFO mapred.JobClient: total time spent by all maps in occupied slots (ms)=31978
18/03/02 15:32:49 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=0
18/03/02 15:32:49 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0

```

jayantmohite@localhost:~ Browsing HDFS - Mozilla Firefox 1 / 4

Step Description:

In this step we will further use the parameter of specifying the number of mappers to provide our own choice of degree of parallelism or number of mappers.

You can use this option in order to reduce the time consumption for the import in case of tables with large numbers of records.

Command:

```
[jayantmohite@localhost] $ sqoop import --connect
jdbc:mysql://localhost/classicmodels --table employees --target-dir sqoop4 -m 5 --
username root -password root
```

Step Visualization:

```
File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/classicmodels --table employees -m 5 --target-dir sqoop4 --username root --password root
Warning: /usr/lib/sqoop/..hbbase does not exist! HBase imports will fail.
Please set $HBASE HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/..accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO HOME to the root of your Accumulo installation.
18/03/02 15:34:31 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:34:31 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:34:32 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:34:32 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:34:33 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:34:33 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:34:33 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/b17a87fcb6e6fdf1a63aa37ef39529b2/employees.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:34:42 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/b17a87fcb6e6fdf1a63aa37ef39529b2/employees.jar
18/03/02 15:34:42 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/03/02 15:34:42 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/03/02 15:34:42 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/03/02 15:34:42 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/03/02 15:34:42 INFO mapreduce.ImportJobBase: Beginning import of employees
18/03/02 15:35:01 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:35:01 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`employeeNumber`), MAX(`employeeNumber`) FROM `employees`
18/03/02 15:35:01 INFO db.IntegerSplitter: Split size: 140; Num splits: 5 from: 1002 to: 1702
18/03/02 15:35:03 INFO mapred.JobClient: Running job: job_201803021449_0005
18/03/02 15:35:04 INFO mapred.JobClient: map 0% reduce 0%
```

Step Description:

In this step we will specify the fields which should be used instead of the primary key to split the job into multiple tasks.

This is one of the options available when you are working with a table that does not have a primary key field.

Command:

```
[jayantmohite@localhost] $ sqoop import --connect
jdbc:mysql://localhost/classicmodels --table employees --target-dir sqoop5 --split-
by officeCode --username root -password root
```

New Parameter Used:

--split-by

(the value of this field will specify the field that should be used instead of the primary key field or incase of a table which does not contain a primary key field)

Step Visualization:

```
jayantmohite@localhost:~$ sqoop import --connect jdbc:mysql://localhost/classicmodels --table employees --split-by officeCode --target-dir sqoop5 --username root -password root
Warning: /usr/lib/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/03/02 15:38:43 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:38:43 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:38:43 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:38:43 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:38:44 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:38:44 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:38:44 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/e3162ae62d17a2954fc4b642623120d6/employees.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:38:53 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/e3162ae62d17a2954fc4b642623120d6/employees.jar
18/03/02 15:38:53 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/03/02 15:38:53 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/03/02 15:38:53 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/03/02 15:38:53 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/03/02 15:38:53 INFO mapreduce.ImportJobBase: Beginning import of employees
18/03/02 15:39:06 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:39:06 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`officeCode`), MAX(`officeCode`) FROM `employees`
18/03/02 15:39:07 WARN db.TextSplitter: Generating splits for a textual index column.
18/03/02 15:39:07 WARN db.TextSplitter: If your database sorts in a case-insensitive order, this may result in a partial import or duplicate records.
18/03/02 15:39:07 WARN db.TextSplitter: You are strongly encouraged to choose an integral split column.
18/03/02 15:39:09 INFO mapred.JobClient: Running job: job_201803021449_0007
18/03/02 15:39:10 INFO mapred.JobClient: map 0% reduce 0%
```

Step Description:

In this step, instead of importing the complete data set of the table, we will specify a criteria based on which the records should be fetched.

You can also specify the projection which will limit the fields that are displayed in the output.

Commands:

```
[jayantmohite@localhost] $ sqoop import --connect  
jdbc:mysql://localhost/classicmodels --table employees --target-dir sqoop6 --  
where 'officeCode=1' --columns lastName,firstName --username root -password  
root
```

New Parameters Used:

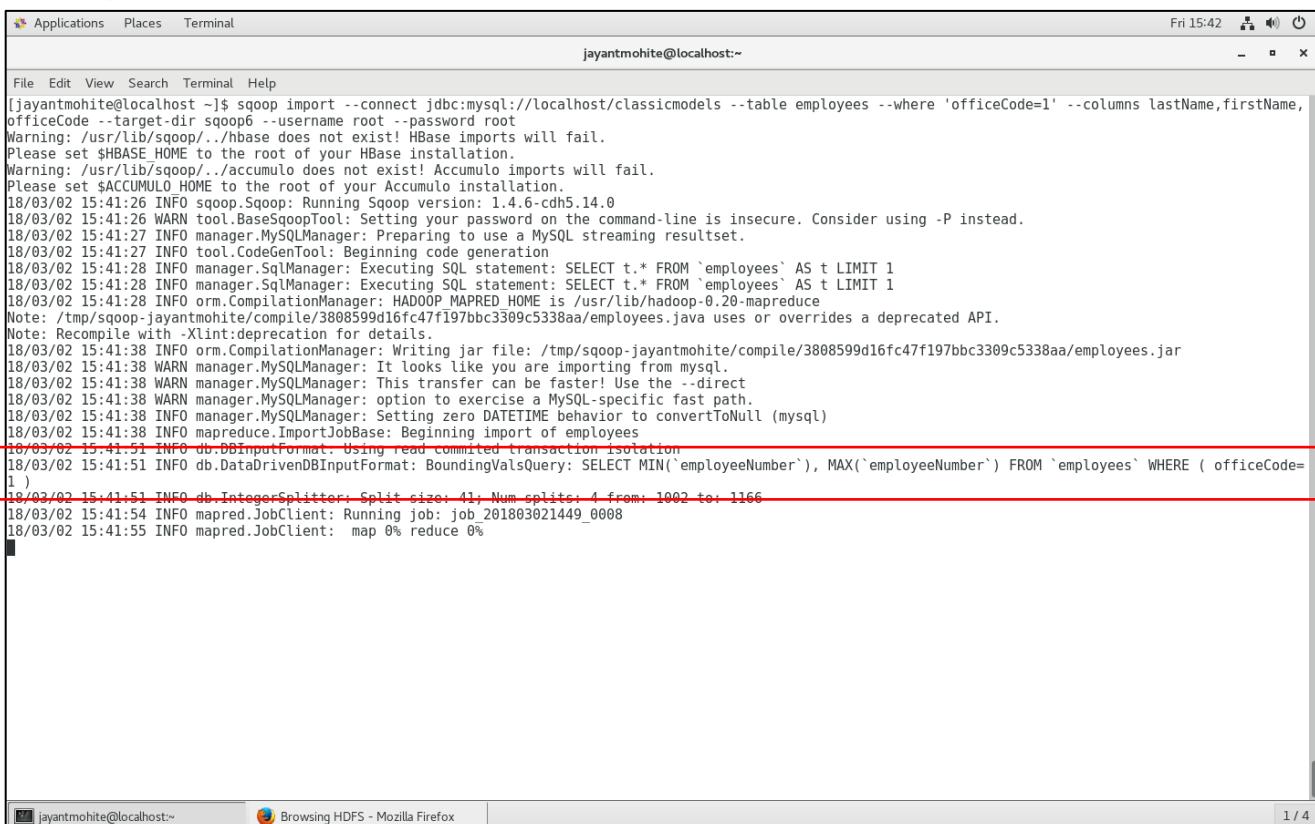
--where

(this is the condition that needs to be executed in order to filter the output records)

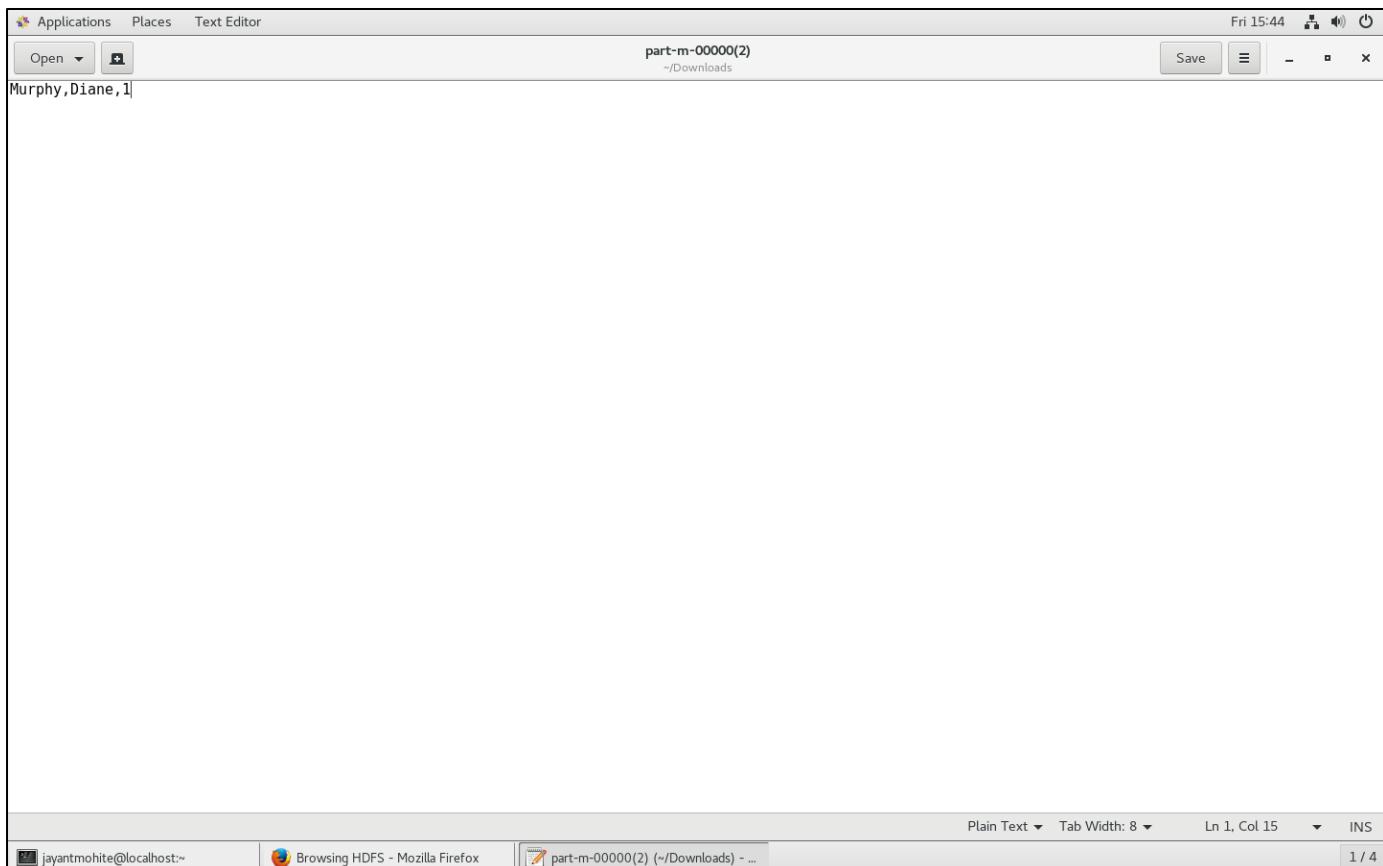
--columns

(this is a comma separated list of columns that should be a part of the output)

Step Visualization:



```
[jayantmohite@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/classicmodels --table employees --where 'officeCode=1' --columns lastName,firstName,officeCode -target-dir sqoop6 --username root --password root
Warning: /usr/lib/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/03/02 15:41:26 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:41:26 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:41:27 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:41:27 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:41:28 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:41:28 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employees` AS t LIMIT 1
18/03/02 15:41:28 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/3808599d16fc47f197bbc3309c5338aa/employees.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:41:38 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/3808599d16fc47f197bbc3309c5338aa/employees.jar
18/03/02 15:41:38 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/03/02 15:41:38 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/03/02 15:41:38 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/03/02 15:41:38 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/03/02 15:41:38 INFO mapreduce.ImportJobBase: Beginning import of employees
18/03/02 15:41:51 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:41:51 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`employeeNumber`), MAX(`employeeNumber`) FROM `employees` WHERE ( officeCode=1 )
18/03/02 15:41:51 INFO db.IntegerSplitter: Split size: 4; Num splits: 4 from: 1002 to: 1166
18/03/02 15:41:54 INFO mapred.JobClient: Running job: job_201803021449_0008
18/03/02 15:41:55 INFO mapred.JobClient: map 0% reduce 0%
```



```
part-m-00000(2)
~/Downloads
Murphy,Diane,1
Murphy,Diane,1
```

Step Description:

In this step we will specify a query and the import will be the records fetched by the execution of this query. It can be a simple query, a query with criteria and projection, a query with joins, a sub query or a nested query as well.

By far you might have observed one thing that whenever you need to do something extra than what the default Sqoop command provides or if your need to override any default behavior of Sqoop, you need to introduce a new parameter.

If we consider a query as a parameter, then the query itself can be of different types and after debugging may produce multiple parameters with different values like `table_name`, `criteria`, `projections`, `join tables`, `join conditions`, etc.

If the user has to manage this and specify this then it would turn out to be an overhead and then Sqoop instead of reducing workload will rather turn out increasing it.

So in order to avoid this situation, sqoop manages the debugging of the query on its end so that the user does not have to get into these complications.

So no matter what sort of query it is, you need to compulsory have a `where` clause in the query with the variable `$CONDITIONS`.

Basically this variable is used by sqoop to store the parameters and their values that would be generated after debugging the query.

Command:

```
[jayantmohite@localhost] $ sqoop import --connect  
jdbc:mysql://localhost/classicmodels --table employees --target-dir sqoop7 --  
username root -password root
```

New Parameter used:

--query

(this has to be the exact query you want to execute. Any query that produces a valid result on the database can be used here.)

```
[jayantmohite@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/classicmodels --query 'select firstName,lastName from employees where officeCode=1 and $CONDITIONS' -m 1 --target-dir sqoop7 --username root --password root
Warning: /usr/lib/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/03/02 15:46:38 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:46:39 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:46:39 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:46:39 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:46:41 INFO manager.SqlManager: Executing SQL statement: select firstName,lastName from employees where officeCode=1 and (1 = 0)
18/03/02 15:46:41 INFO manager.SqlManager: Executing SQL statement: select firstName,lastName from employees where officeCode=1 and (1 = 0)
18/03/02 15:46:41 INFO manager.SqlManager: Executing SQL statement: select firstName,lastName from employees where officeCode=1 and (1 = 0)
18/03/02 15:46:41 INFO orm.CompilationManager: HADOOP MAPRED HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/c1da43f4f7e0fa92433d4e9a18a026a6/QueryResult.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:46:50 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/c1da43f4f7e0fa92433d4e9a18a026a6/QueryResult.jar
18/03/02 15:46:50 INFO mapreduce.ImportJobBase: Beginning query import.
18/03/02 15:47:06 INFO db.DBInputFormat: Using read committed transaction isolation
18/03/02 15:47:08 INFO mapred.JobClient: Running job: job_201803021449_0009
18/03/02 15:47:09 INFO mapred.JobClient: map 0% reduce 0%
```

jayantmohite@localhost:~ [Browsing HDFS - Mozilla Firefox] 1 / 4

Step Description:

In this step we will export create a table in MySQL database by name jayant_table. Our aim is to add data to this table from a text file which is saved in HDFS.

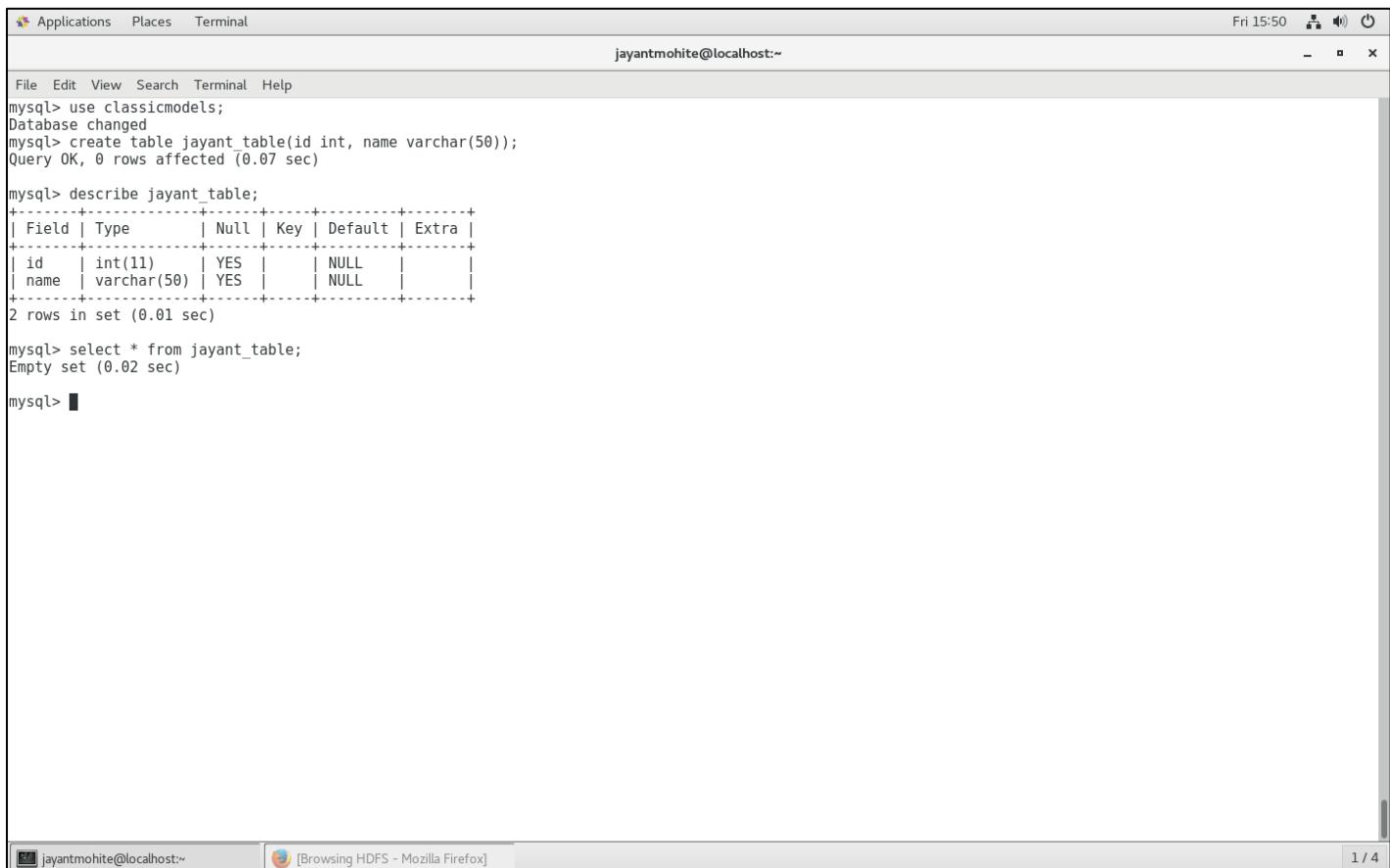
The schema of this table will resemble the input file which is store in HDFS.

Commands:

```
mysql> use classicmodels;
```

```
mysql> create table jayant_table(id int, name varchar(50))
```

Step Visualization:



```
Applications Places Terminal Fri 15:50
jayantmohite@localhost:~ - x

File Edit View Search Terminal Help
mysql> use classicmodels;
Database changed
mysql> create table jayant_table(id int, name varchar(50));
Query OK, 0 rows affected (0.07 sec)

mysql> describe jayant_table;
+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| id    | int(11) | YES  |     | NULL    |       |
| name  | varchar(50)| YES |     | NULL    |       |
+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> select * from jayant_table;
Empty set (0.02 sec)

mysql> ■
```

Step Description:

In this step we will create our input file by name jayant_file.txt using the file editor called as gedit available in CentOS. We will store this file in our home location on the Linux File System.

We will create a directory by name jayant_data in our HDFS and store the file in this directory

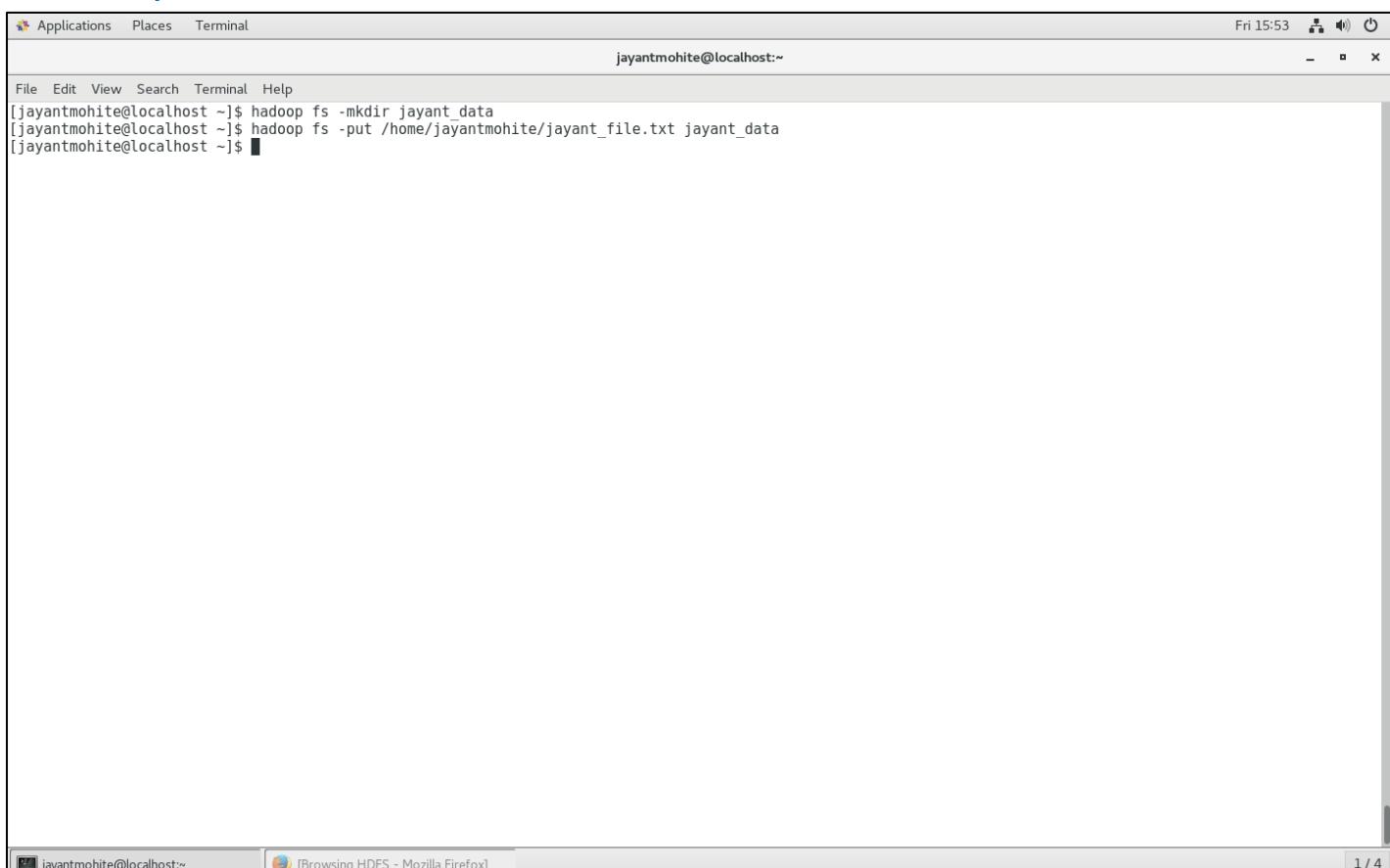
Commands:

[jayantmohite@localhost] \$ hadoop fs -mkdir <enter name of directory to be created as jayant_data>

(this will create the directory under the location /user/jayantmohite/)

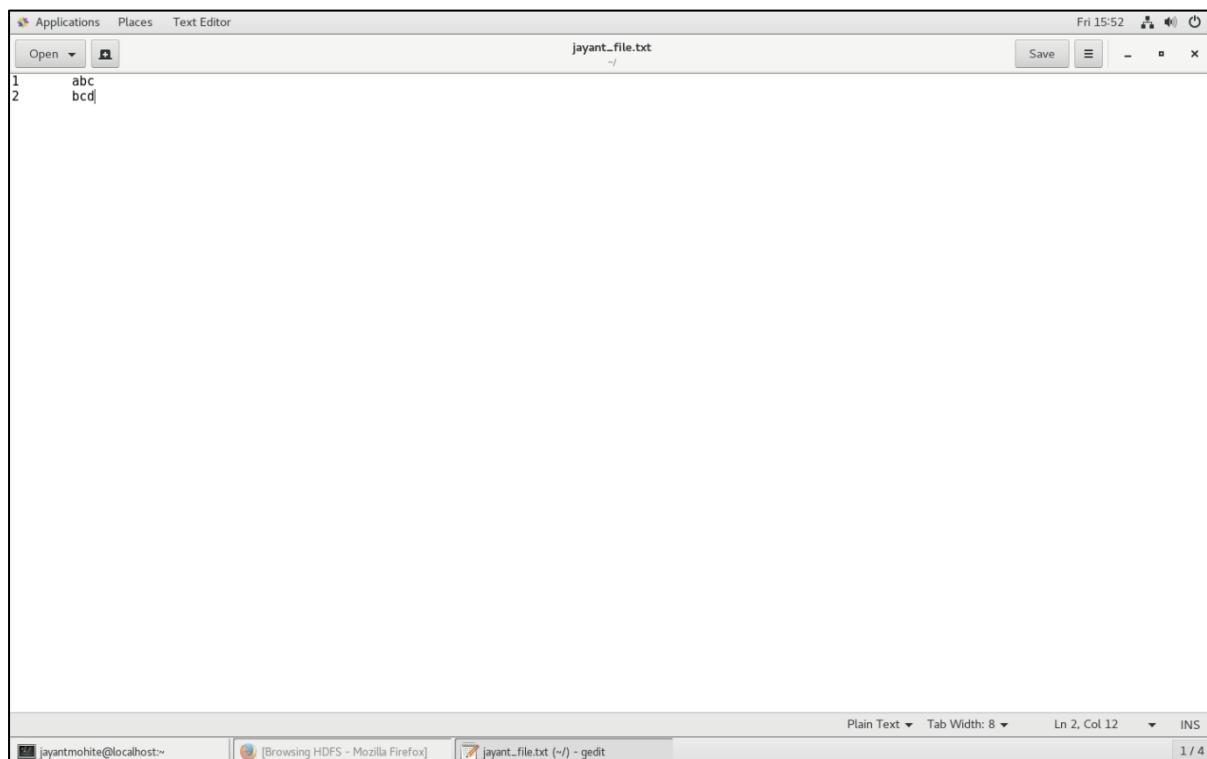
[jayantmohite@localhost] \$ hadoop fs -put <enter source location as /home/jayantmohite/jayant_file.txt> <enter destination location as jayant_data>

Step Visualization:



The screenshot shows a terminal window titled 'jayantmohite@localhost:~'. The window has a standard Linux desktop interface with icons for Applications, Places, and Terminal at the top. The title bar also shows the date and time as 'Fri 15:53'. The terminal window contains the following command history:

```
File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ hadoop fs -mkdir jayant_data
[jayantmohite@localhost ~]$ hadoop fs -put /home/jayantmohite/jayant_file.txt jayant_data
[jayantmohite@localhost ~]$
```



Step Description:

In this step we will load the data from the file named jayant_file.txt stored in hdfs location of /user/jayantmohite/jayant_data into the table jayant_table that we created earlier in MySQL database.

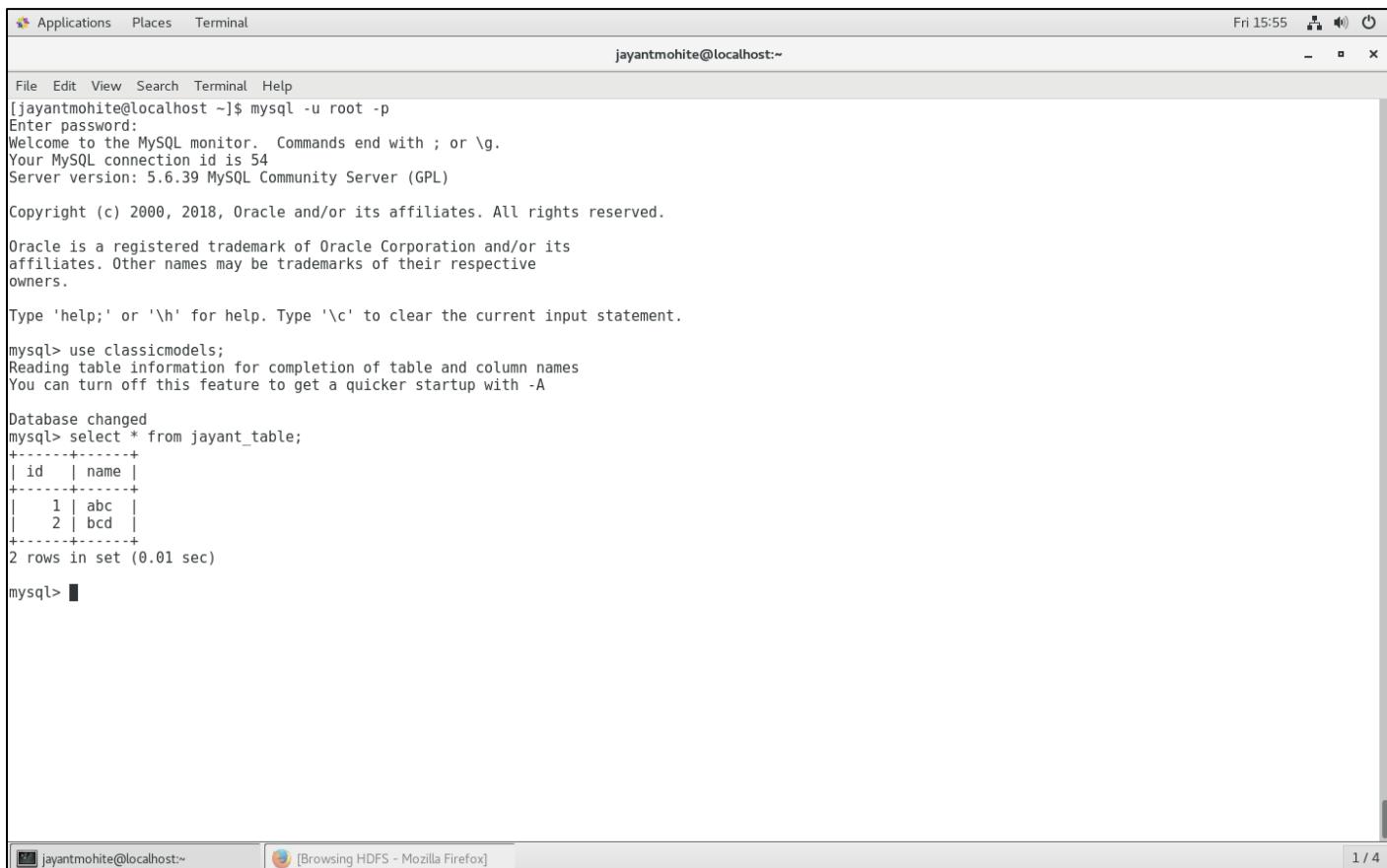
Commands:

```
[jayantmohite@localhost] $ sqoop export --connect
jdbc:mysql://localhost/classicmodels --table jayant_table --export-dir jayant_data -
-fields-terminated by '\t' -m 1 --username root -password root
```

Step Visualization:

```
[jayantmohite@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/classicmodels --table jayant_table -m 1 --export-dir jayant_data --fields-terminated-by '\t' --username root --password root
Warning: /usr/lib/sqoop/..hbbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/..accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/03/02 15:53:30 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
18/03/02 15:53:30 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/03/02 15:53:30 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/03/02 15:53:30 INFO tool.CodeGenTool: Beginning code generation
18/03/02 15:53:31 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `jayant_table` AS t LIMIT 1
18/03/02 15:53:31 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `jayant_table` AS t LIMIT 1
18/03/02 15:53:31 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-0.20-mapreduce
Note: /tmp/sqoop-jayantmohite/compile/df01ed374de39db76d09f3348ecc1054/jayant_table.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/03/02 15:53:41 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-jayantmohite/compile/df01ed374de39db76d09f3348ecc1054/jayant_table.jar
18/03/02 15:53:41 INFO mapreduce.ExportJobBase: Beginning export of jayant_table
18/03/02 15:53:59 INFO input.FileInputFormat: Total input paths to process : 1
18/03/02 15:53:59 INFO input.FileInputFormat: Total input paths to process : 1
18/03/02 15:54:03 INFO mapred.JobClient: Running job: job_201803021449_0010
18/03/02 15:54:04 INFO mapred.JobClient: map 0% reduce 0%
```

Now login to your MySQL shell and verify the result



```
[jayantmohite@localhost ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 54
Server version: 5.6.39 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use classicmodels;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from jayant_table;
+----+-----+
| id | name |
+----+-----+
| 1  | abc  |
| 2  | bcd  |
+----+-----+
2 rows in set (0.01 sec)

mysql>
```

With this we complete the Sqoop Tutorial. Now you can go ahead and explore more options available and explained in the theory section of Sqoop.

10. Apache Hive

Talking about analysis the main task always remain is [Data Mining](#). Data Mining means deriving useful information from large chunks of data. The simplest way of doing this has always been asking question to the data. This was a very easy task when we were dealing with the database. Using the Structured Query Language ([SQL](#)) we were able to query the data easily.

When it came to Hadoop, this task became tedious as Hadoop does not support SQL. We had to write complex Map Reduce programs to do those functions that were once done by simple SELECT queries. In order to ease this task we came up with a tool that worked as a Map Reduce SQL abstraction or SQL on top of Hadoop. This tool is called as [Hive](#).

When Hive is installed on the client machine we get a Hive metastore that is configured on the client side and on the cluster we need to build a Hive warehouse. The Hive metastore is a place where the Hive query engine keeps all the Schema related information or the so called Meta Data. The Hive warehouse inside the HDFS is where the actual Hive table data is stored. You can easily visualize the warehouse as a part of HDFS distributed across all the Data Nodes.

Now let us have a look at some Hive operations to get a more detailed understanding of this tool.

Step Description:

Whenever we talk about a table, there are two things involved. One is the meta data of the table or the schema of the table and the second thing is the data itself.

Hive stores the meta data of a table in the metastore located in the Linux File System which is created when we install Hive and stores the data in the warehouse location which we create in the HDFS.

In this step we will do some additional configurations required to setup the Hive metastore and the warehouse and log in to the Hive Shell.

Commands:

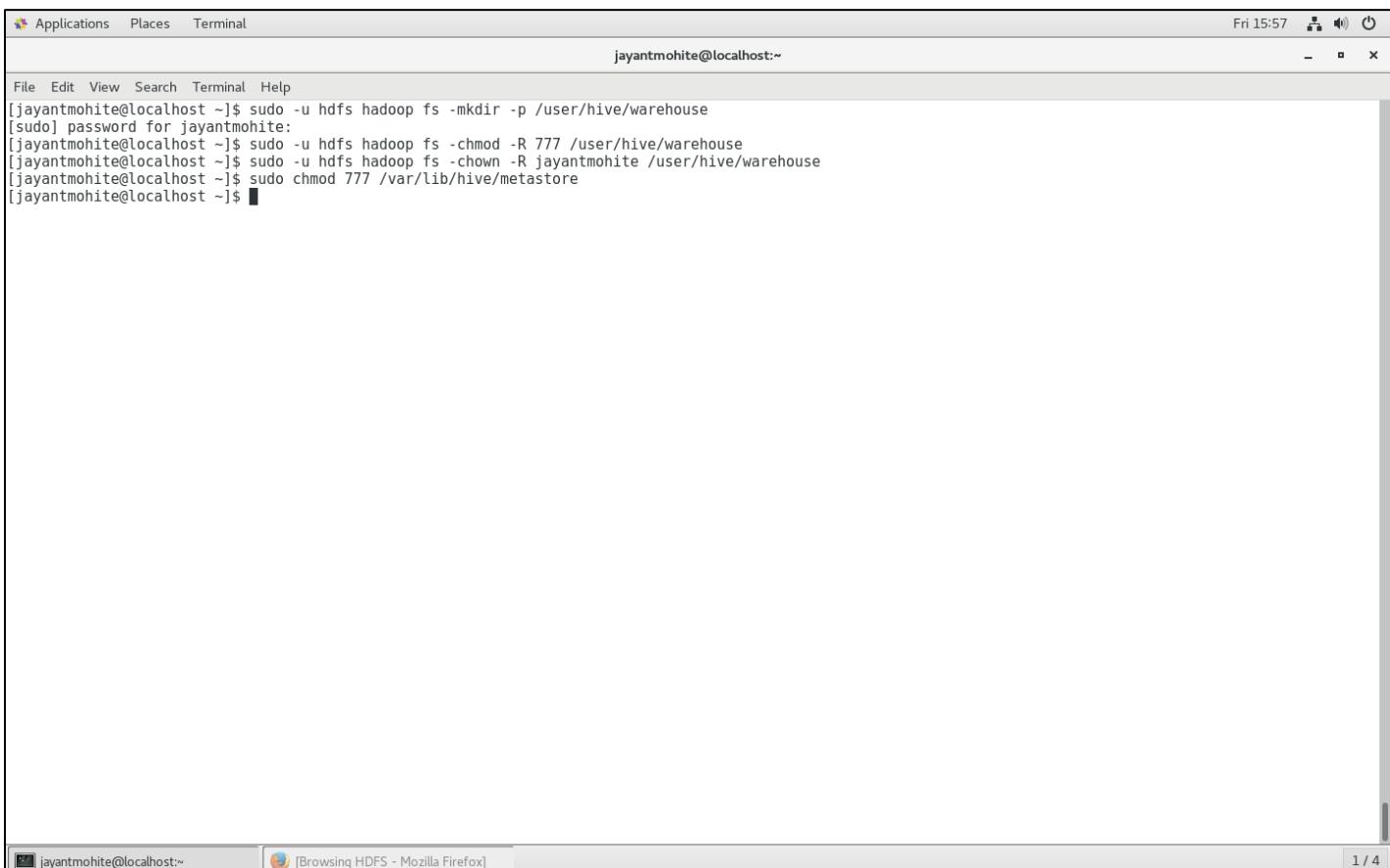
```
[jayantmohite@localhost] $ sudo -u hdfs hadoop fs -mkdir -p  
/user/hive/warehouse
```

```
[jayantmohite@localhost] $ sudo -u hdfs hadoop fs -chmod -R 777  
/user/hive/warehouse
```

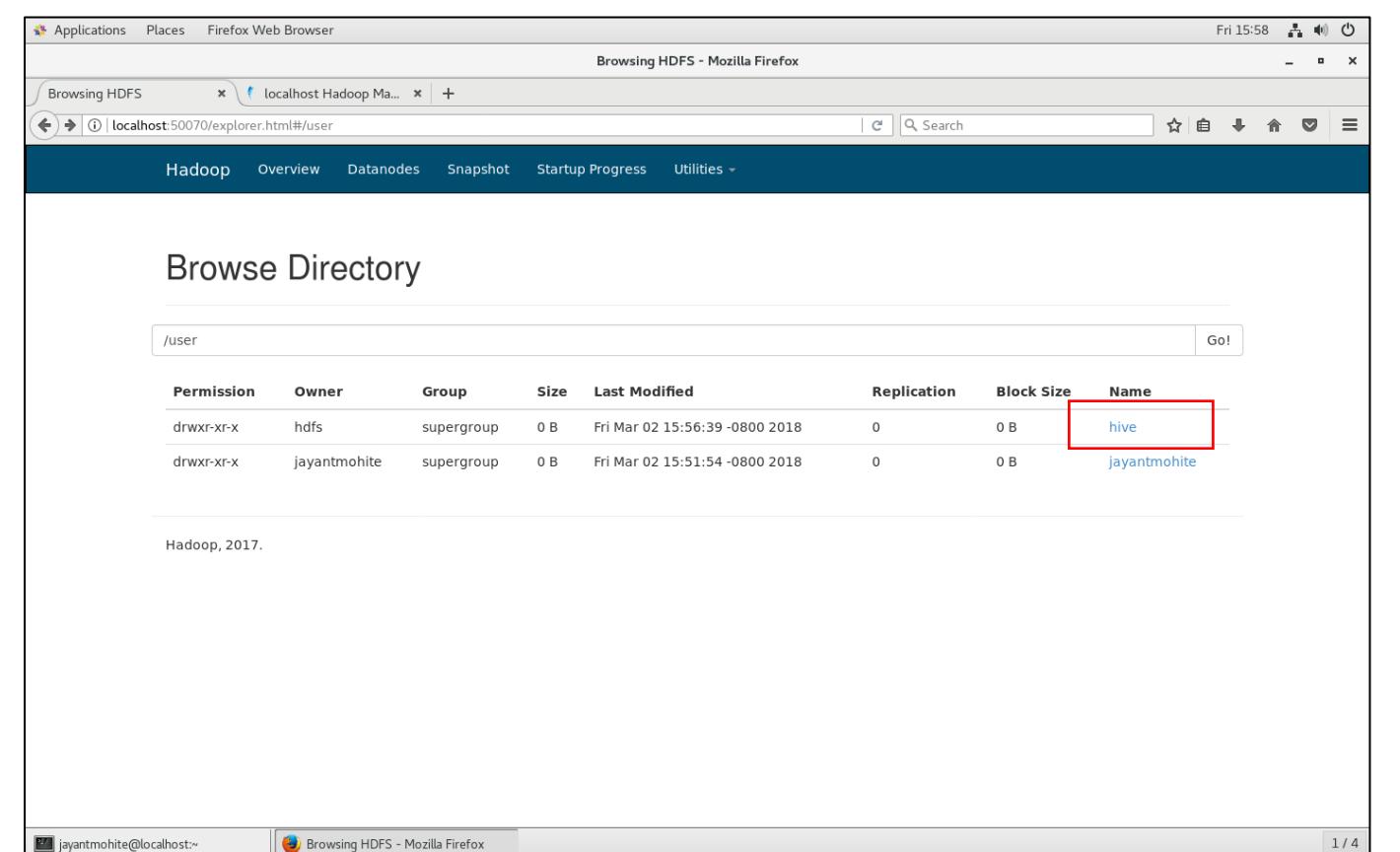
```
[jayantmohite@localhost] $ sudo -u hdfs hadoop fs -chown -R jayantmohite  
/user/hive/warehouse
```

```
[jayantmohite@localhost] $ sudo chmod 777 /var/lib/hive/metastore
```

Step Visualization:



```
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -mkdir -p /user/hive/warehouse
[jayantmohite@localhost ~]$ sudo password for jayantmohite:
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -chmod -R 777 /user/hive/warehouse
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -chown -R jayantmohite /user/hive/warehouse
[jayantmohite@localhost ~]$ sudo chmod 777 /var/lib/hive/metastore
[jayantmohite@localhost ~]$
```



Browsing HDFS - Mozilla Firefox

localhost:5070/explorer.html#/user

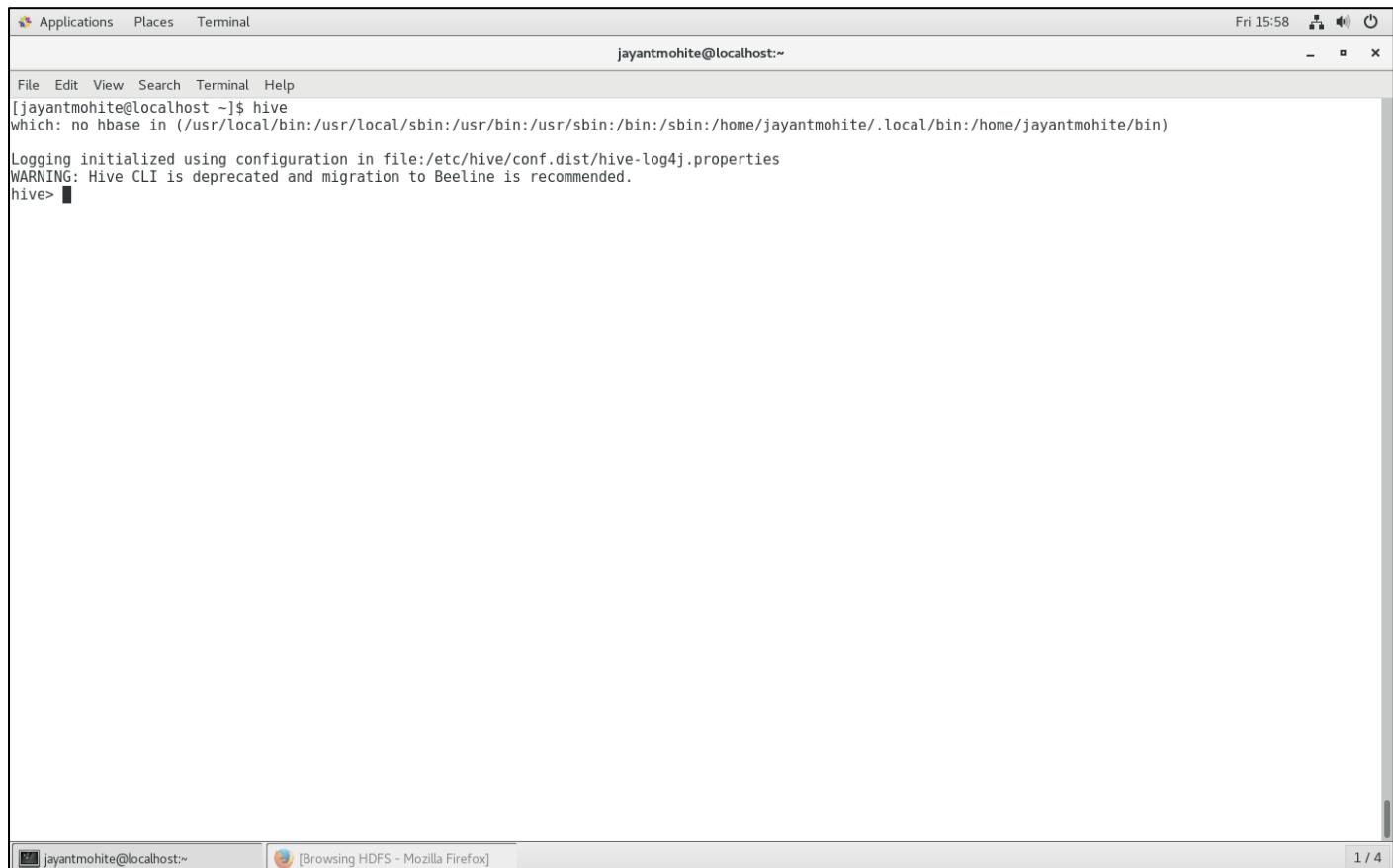
Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/user

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hdfs	supergroup	0 B	Fri Mar 02 15:56:39 -0800 2018	0	0 B	hive
drwxr-xr-x	jayantmohite	supergroup	0 B	Fri Mar 02 15:51:54 -0800 2018	0	0 B	jayantmohite

Hadoop, 2017.



A screenshot of a Linux terminal window titled "jayantmohite@localhost:~". The window shows the following text:

```
File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ hive
which: no hbase in (/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/jayantmohite/.local/bin:/home/jayantmohite/bin)
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> █
```

The terminal window has a standard title bar with icons for Applications, Places, and Terminal. The status bar at the bottom shows the user's name and the current directory. The bottom of the window also displays the taskbar with other open applications.

Understanding Working of Hive

Browsing HDFS - Mozilla Firefox

localhost:50070/explorer.html#/user/hive/warehouse

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

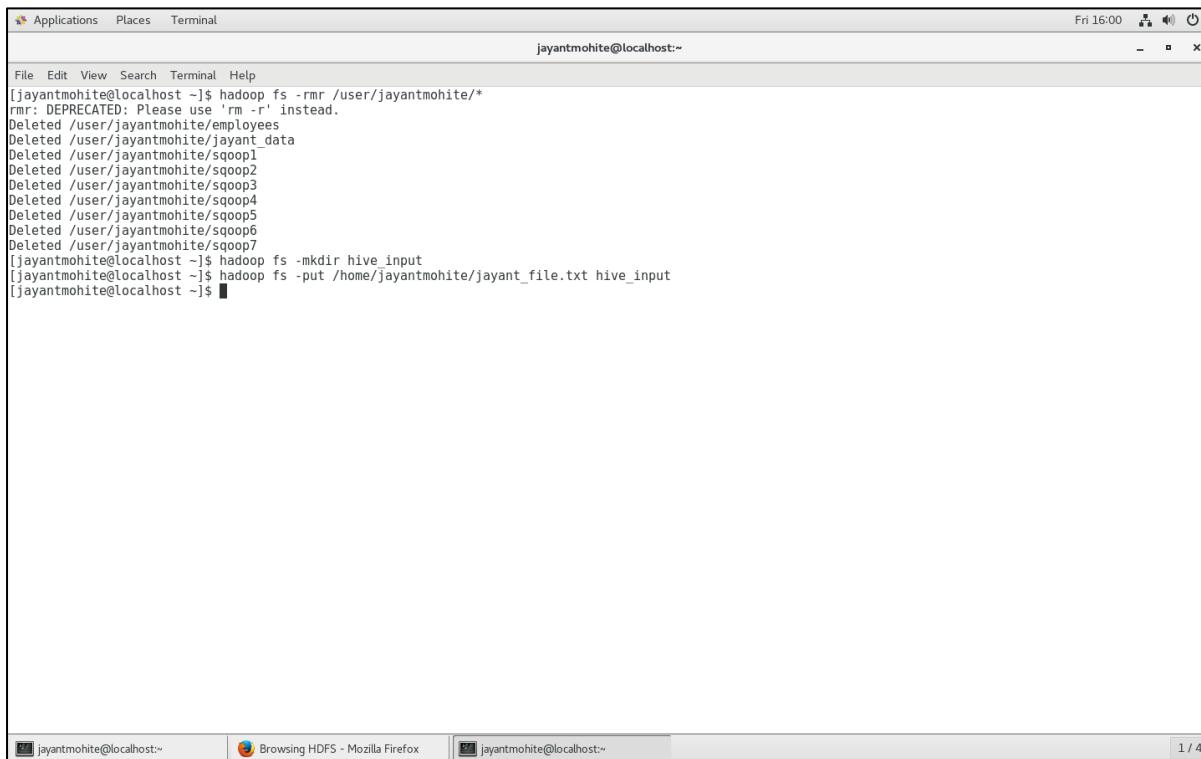
Browse Directory

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
							/user/hive/warehouse

Hadoop, 2017.

jayantmohite@localhost:~ | Browsing HDFS - Mozilla Firefox | 1 / 4

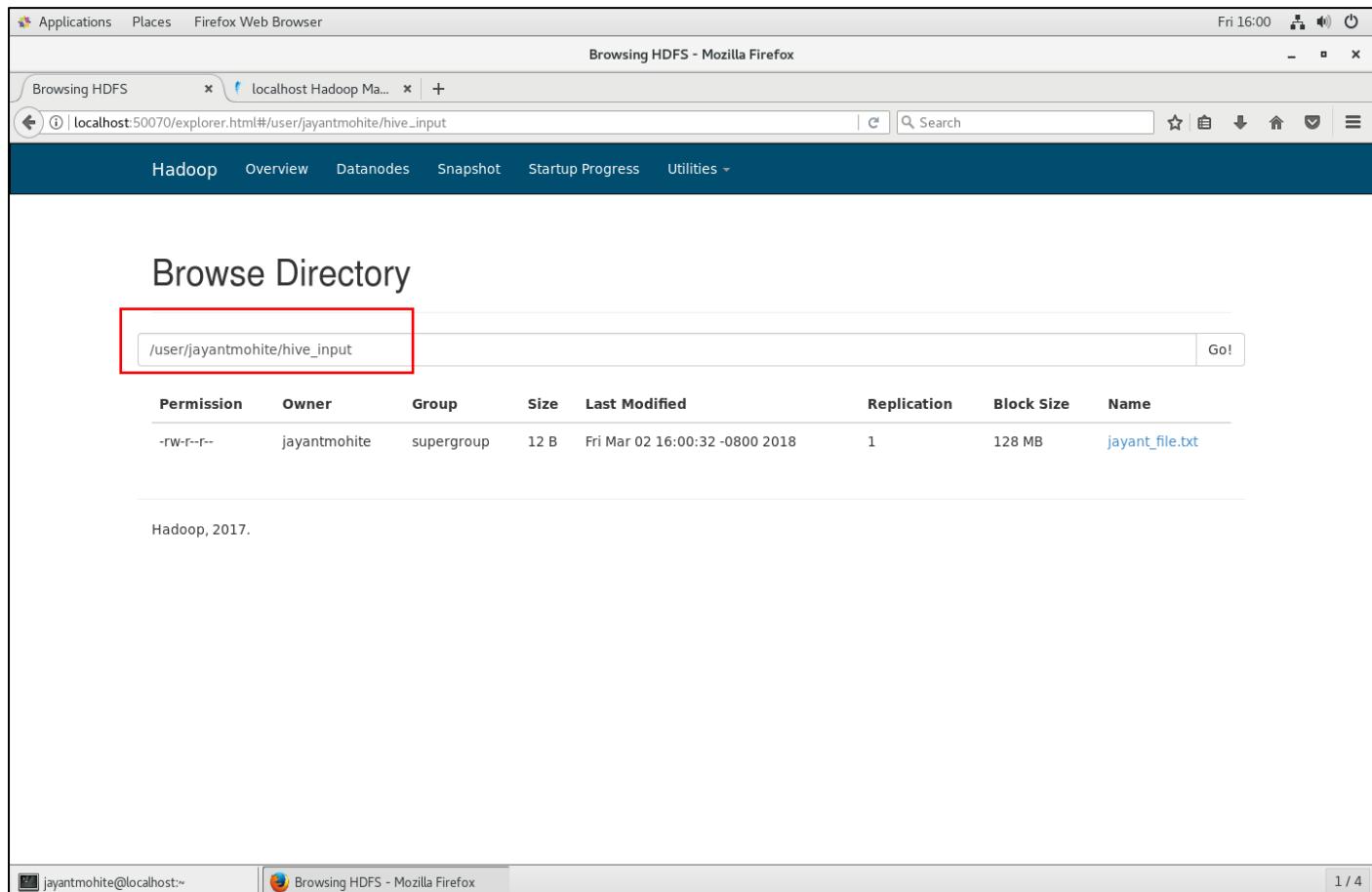
You can see that at this point of time the, warehouse is empty as we have not started working with Hive yet. Now, we will use the file jayant_file.txt as input for our initial operations.



A screenshot of a Linux terminal window titled "Terminal". The window shows a command-line session with the user "jayantmohite" running commands on a local host. The session starts with removing a directory, then deleting several files and directories under "/user/jayantmohite". It then creates a new directory "hive_input" and puts a file named "jayant_file.txt" into it. The terminal window has a standard title bar with icons for Applications, Places, and Terminal, and a status bar at the bottom showing the current tab and page number (1 / 4).

```
[jayantmohite@localhost ~]$ hadoop fs -rmdir /user/jayantmohite/*
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted /user/jayantmohite/employees
Deleted /user/jayantmohite/jayant_data
Deleted /user/jayantmohite/sqoop1
Deleted /user/jayantmohite/sqoop2
Deleted /user/jayantmohite/sqoop3
Deleted /user/jayantmohite/sqoop4
Deleted /user/jayantmohite/sqoop5
Deleted /user/jayantmohite/sqoop6
Deleted /user/jayantmohite/sqoop7
[jayantmohite@localhost ~]$ hadoop fs -mkdir hive_input
[jayantmohite@localhost ~]$ hadoop fs -put /home/jayantmohite/jayant_file.txt hive_input
[jayantmohite@localhost ~]$
```

Observe that the file is now available at HDFS location
`/user/jayantmohite/hive_input`



Browsing HDFS - Mozilla Firefox

localhost:50070/explorer.html#/user/jayantmohite/hive_input

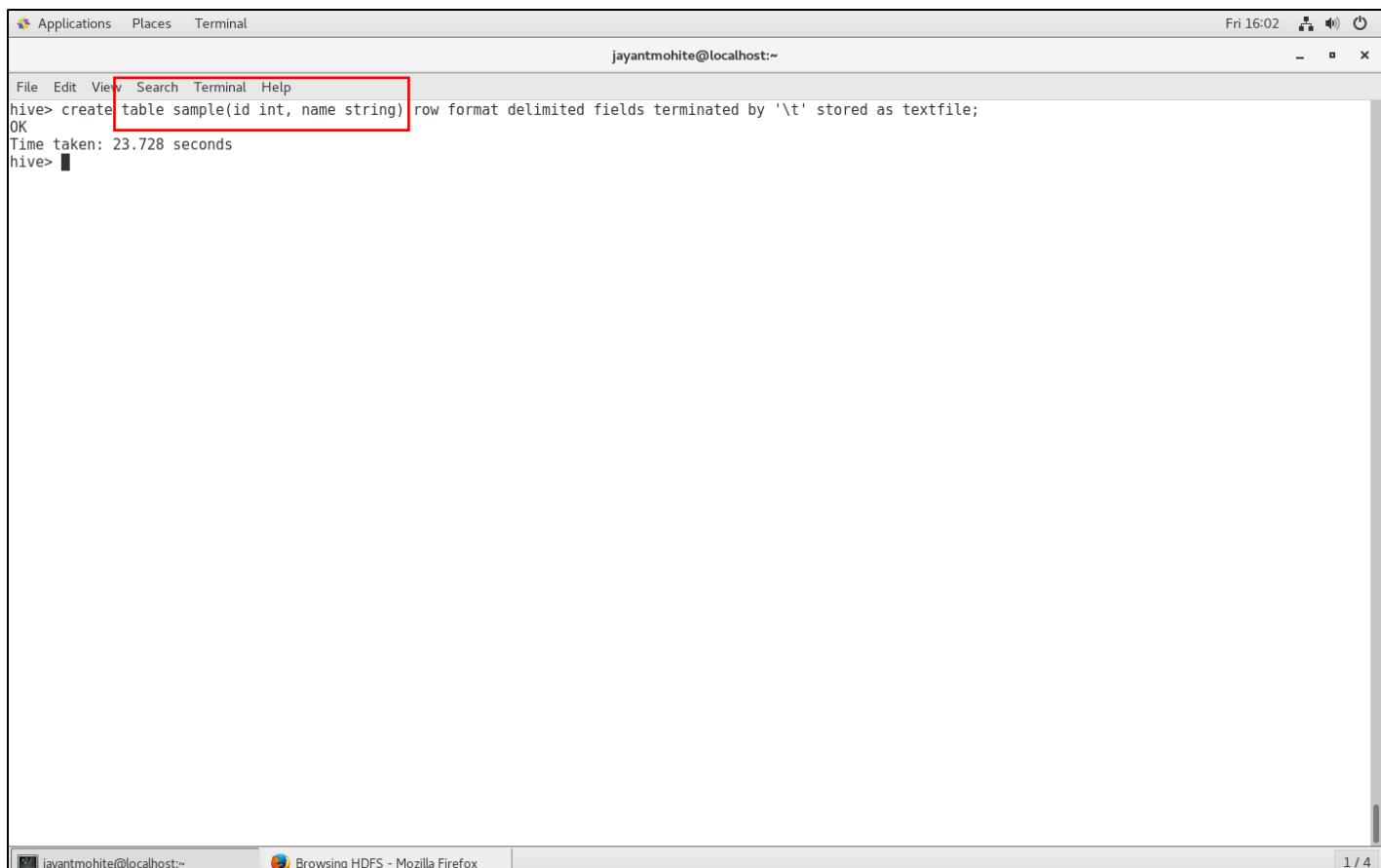
Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	jayantmohite	supergroup	12 B	Fri Mar 02 16:00:32 -0800 2018	1	128 MB	jayant_file.txt

Hadoop, 2017.

We will now enter our hive shell and create a table as we would do in any RDBMS.



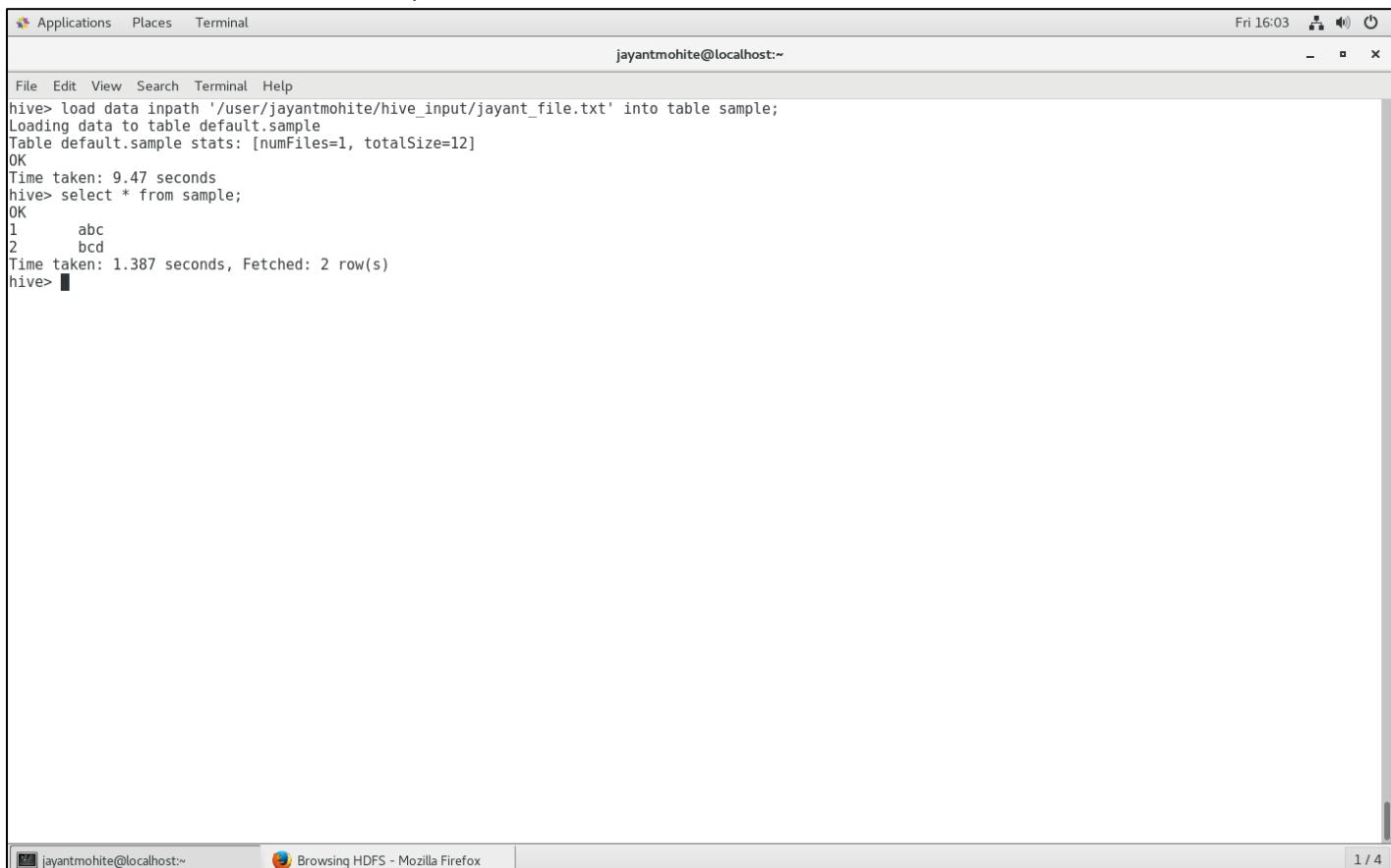
A screenshot of a Linux terminal window titled "Terminal". The window shows the command "hive> create table sample(id int, name string) row format delimited fields terminated by '\t' stored as textfile;" followed by "OK" and "Time taken: 23.728 seconds". The "Search" menu item in the top bar is highlighted with a red box. The bottom status bar shows "jayantmohite@localhost:~" and "Browsing HDFS - Mozilla Firefox". The window title bar says "jayantmohite@localhost:~". The top right corner shows the date "Fri 16:02" and system icons.

```
File Edit View Search Terminal Help
hive> create table sample(id int, name string) row format delimited fields terminated by '\t' stored as textfile;
OK
Time taken: 23.728 seconds
hive>
```

With this we expect that a table should be created but what actually happens is a directory by the name of the table is created in the warehouse.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	jayantmohite	supergroup	0 B	Fri Mar 02 16:02:15 -0800 2018	0	0 B	sample

You can observe that at this very point the directory created is empty. Now we will load data in our newly created table.

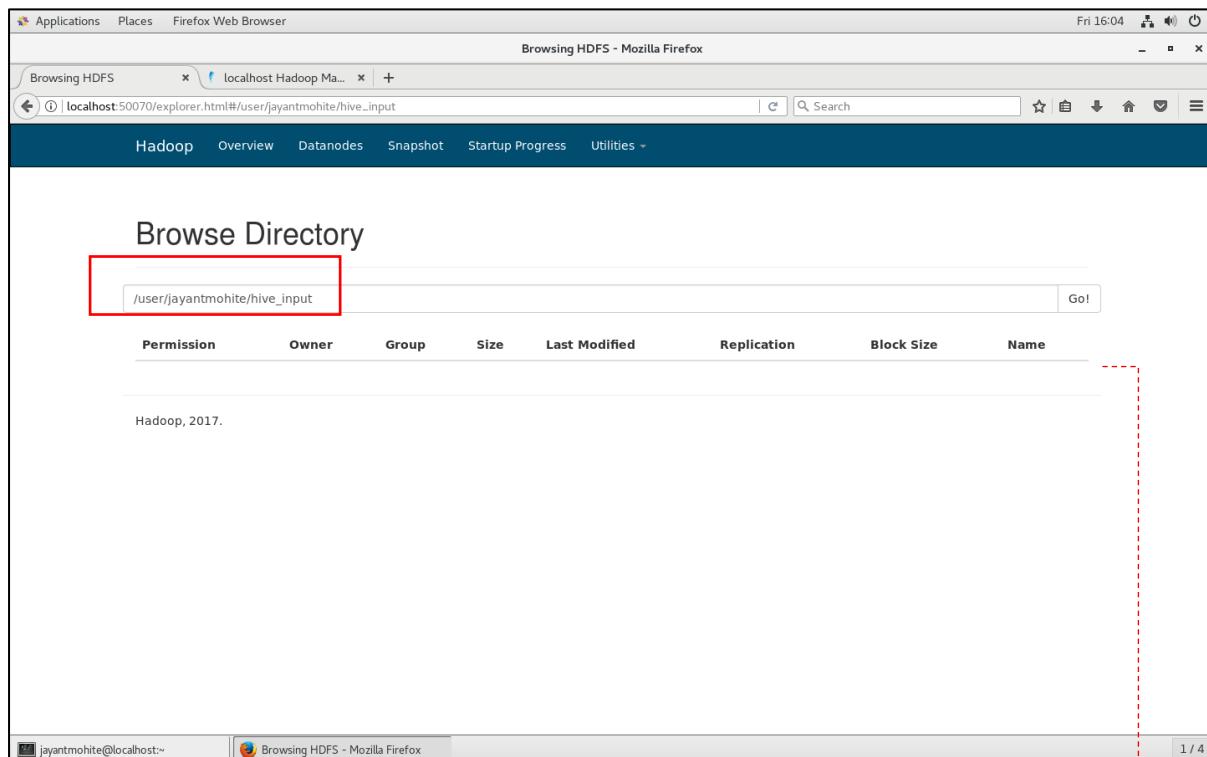


A screenshot of a Linux terminal window titled "Terminal". The window shows the following Hive session:

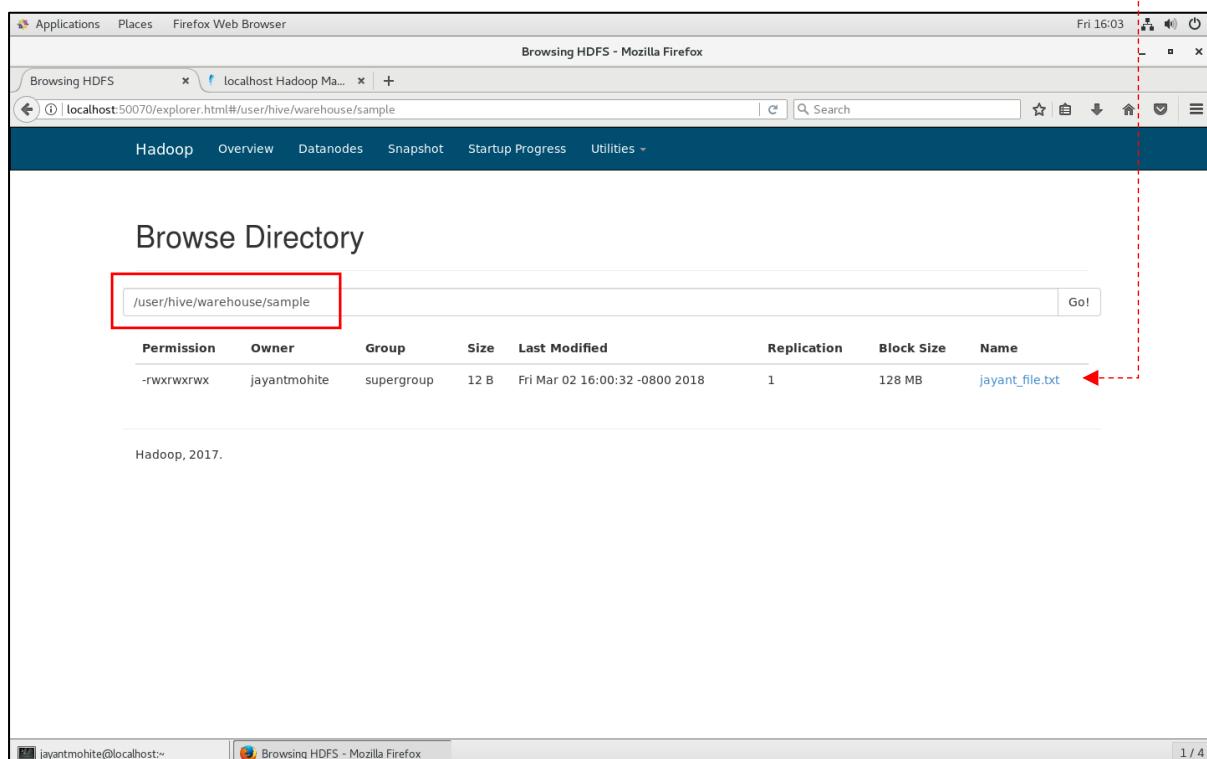
```
File Edit View Search Terminal Help
hive> load data inpath '/user/jayantmohite/hive_input/jayant_file.txt' into table sample;
Loading data to table default.sample
Table default.sample stats: [numFiles=1, totalSize=12]
OK
Time taken: 9.47 seconds
hive> select * from sample;
OK
1     abc
2     bcd
Time taken: 1.387 seconds, Fetched: 2 row(s)
hive>
```

The terminal window has a title bar with "Applications", "Places", and "Terminal". The status bar at the bottom shows "jayantmohite@localhost:~". Below the terminal window, there is a taskbar with icons for "jayantmohite@localhost:~" and "Browsing HDFS - Mozilla Firefox".

But what actually happens is data is cut from the source location and is pasted in the hive warehouse location.

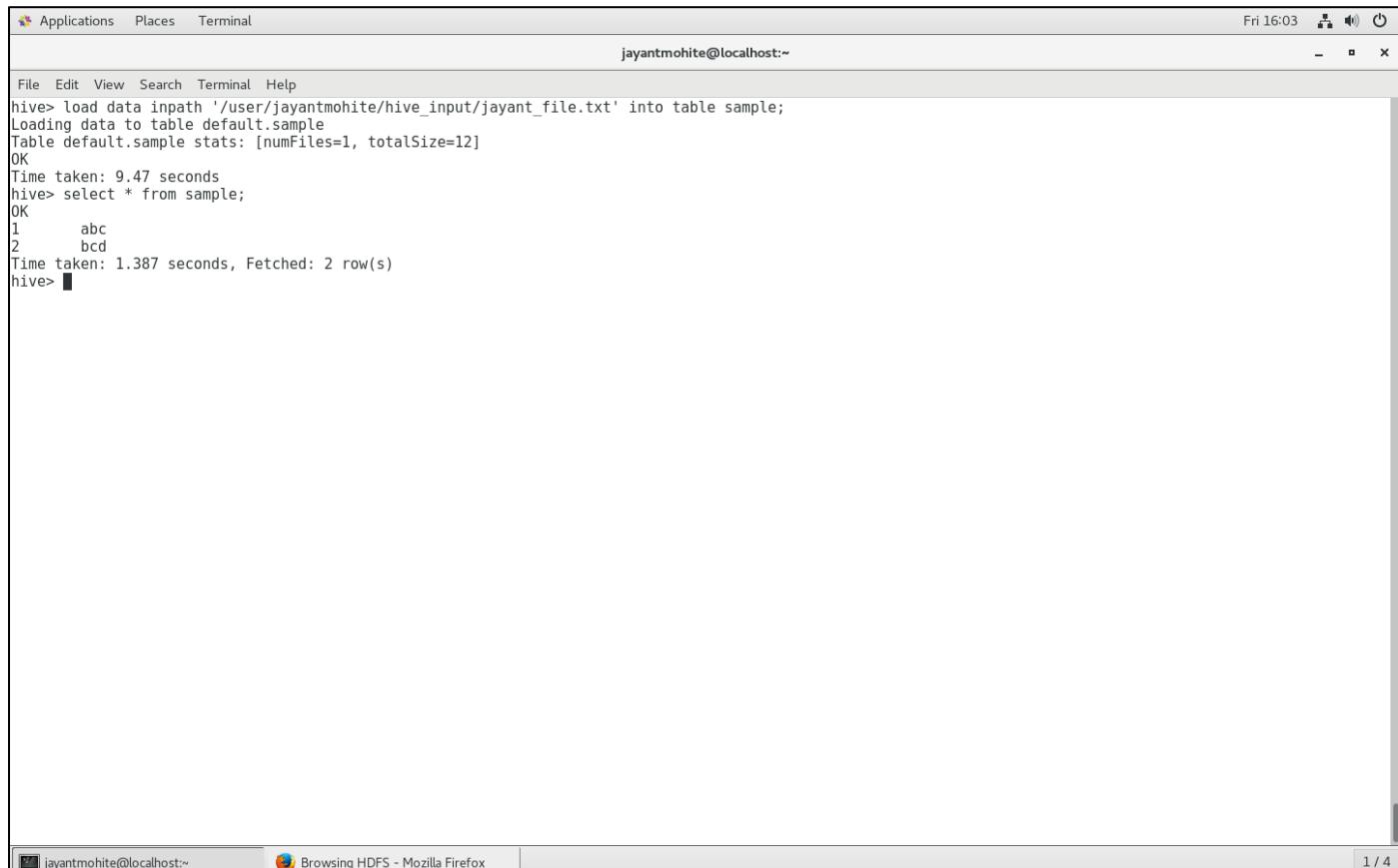


The screenshot shows a Firefox browser window titled "Browsing HDFS - Mozilla Firefox". The address bar displays "localhost Hadoop Ma... | localhost:50070/explorer.html#/user/jayantmohite/hive_input". The main content area is titled "Browse Directory" and shows a table with one row. The table has columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The single row contains: -rwxrwxrwx, jayantmohite, supergroup, 12 B, Fri Mar 02 16:00:32 -0800 2018, 1, 128 MB, jayant_file.txt. A red box highlights the path "/user/jayantmohite/hive_input" in the address bar. A red dashed arrow points from the "Name" column of the table towards the right edge of the screen.



The screenshot shows a Firefox browser window titled "Browsing HDFS - Mozilla Firefox". The address bar displays "localhost Hadoop Ma... | localhost:50070/explorer.html#/user/hive/warehouse/sample". The main content area is titled "Browse Directory" and shows a table with one row. The table has columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The single row contains: -rwxrwxrwx, jayantmohite, supergroup, 12 B, Fri Mar 02 16:00:32 -0800 2018, 1, 128 MB, jayant_file.txt. A red box highlights the path "/user/hive/warehouse/sample" in the address bar. A red dashed arrow points from the "Name" column of the table towards the right edge of the screen.

Now you can use the table and execute any SQL queries you wish. But what will actually happen at the backend is, when you execute a query on the table, a Map Reduce program will be executed on the directory in the warehouse location.



```

Applications Places Terminal Fri 16:03
jayantmohite@localhost:~ - x P

File Edit View Search Terminal Help
hive> load data inpath '/user/jayantmohite/hive_input/jayant_file.txt' into table sample;
Loading data to table default.sample
Table default.sample stats: [numFiles=1, totalSize=12]
OK
Time taken: 9.47 seconds
hive> select * from sample;
OK
1    abc
2    bcd
Time taken: 1.387 seconds, Fetched: 2 row(s)
hive> ■

```

jayantmohite@localhost:~ Browsing HDFS - Mozilla Firefox 1 / 4

Commands:

hive> create table <enter table name as sample> (id int, name string) row format
delimited fields terminated by '\t' stored as textfile;
(in this command, row format delimited fields terminated by '\t' refer to the
delimiter options used in our input file. Hive supports multiple file formats for
internally handling the storage of the table, one of which is textfile.)

hive> load data inpath <enter input data location as
/user/jayantmohite/hive_input/jayant_file.txt> into table <enter table name as
sample>;

(this will cut data from source location /user/jayantmohite/hive_input and will paste it in destination location /user/hive/warehouse/sample)

Hive supports multiple file formats for internally handling the storage of the table's data. These file formats are

1. Text File
2. Sequence File
3. RCFile
4. ORCFile
5. Avro Files
6. Parquet

Now lets consider we have the following type of data

1	abc	100
2	bcd	200
3	cde	300
4	def	400
5	efg	500

Text File Representation

[1,abc,100]
[2,bcd,200]
[3,cde,300]
[4,def,400]
[5,efg,500]

RCFile Representation

[1,2,3,4,5]
[abc, bcd, cde, def, efg]
[100, 200, 300, 400, 500]

Sequence File Representation

[1, abc, 100, 2, bcd, 200, 3, cde, 300, 4, def, 400, 5, efg, 500]

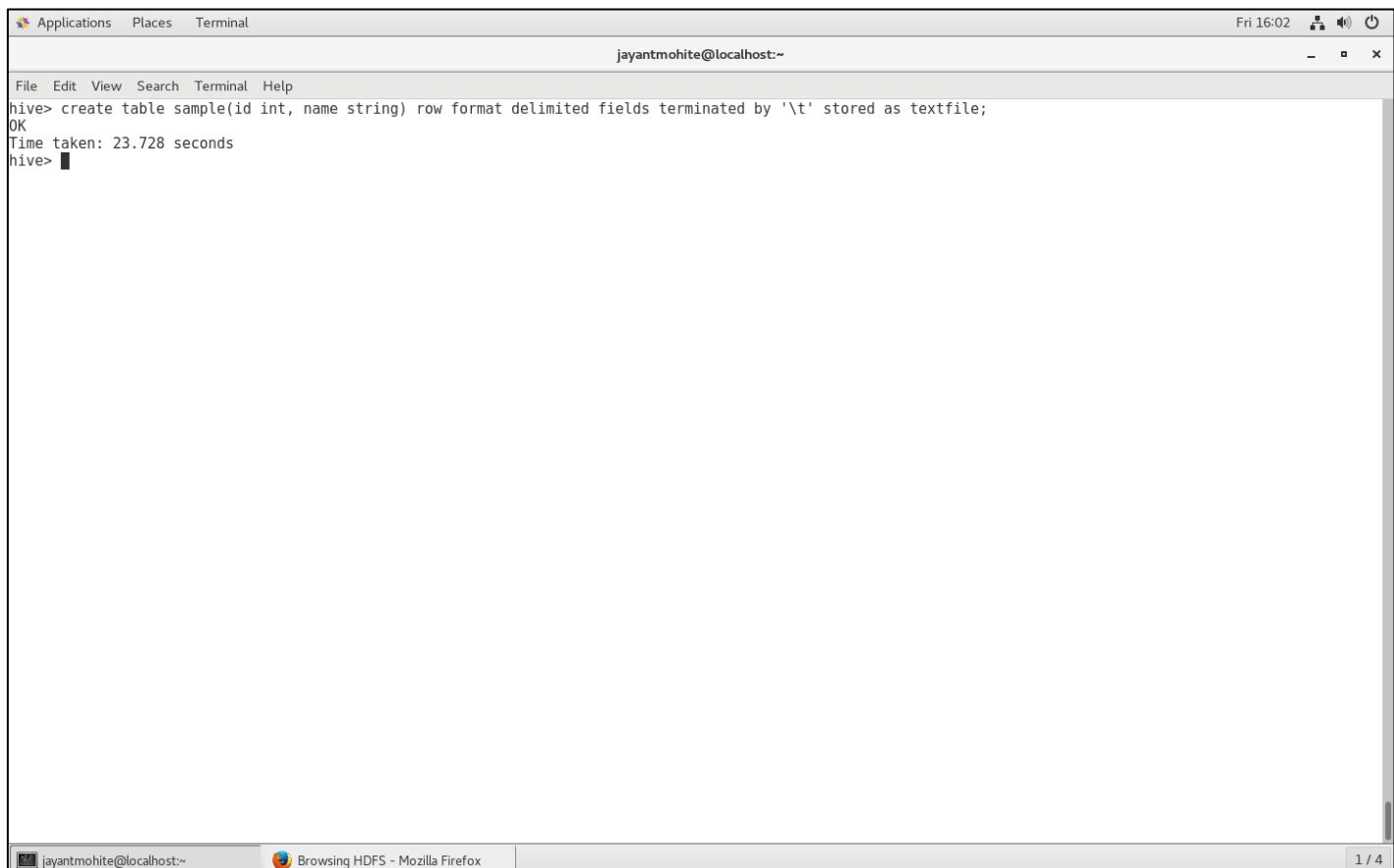
Sample Sequence File representation for more understanding

```
ID AB000263 standard; RNA; PRI; 368 BP.
XX
AC AB000263;
XX
DE Homo sapiens mRNA for prepro cortistatin like peptide, complete cds.
XX
SQ Sequence 368 BP;
AB000263 Length: 368 Check: 4514 ..
   1 acaagatgcc attgtccccc ggccctcctgc tgctgctgct ctccggggcc acggccaccc
   61 ctgcccctgccc cctggagggt ggccccacccg gcccggacacag cgagcatatgc caggaagcg
  121 caggaataag gaaaaggcagc ctcctgactt tcctcgcttg gtggttttag tggacctccc
  181 agggccagtgc cggggcccttc ataggagagg aagctcggttgg ggtggccagg cggcaggaaag
  241 ggcgcacccccc ccagcaatcc gcgcgcggg acagaatgcc ctgcaggaac ttcttcgttgg
  301 agaccttctc ctccctgcaaa taaaacctca cccatgaatg ctcacgcaag tttattaca
  361 gacctgaa
```

Working with textfile format

```
hive> create table sample(id int, name string) row format delimited fields  
terminated by '\t' stored as textfile;
```

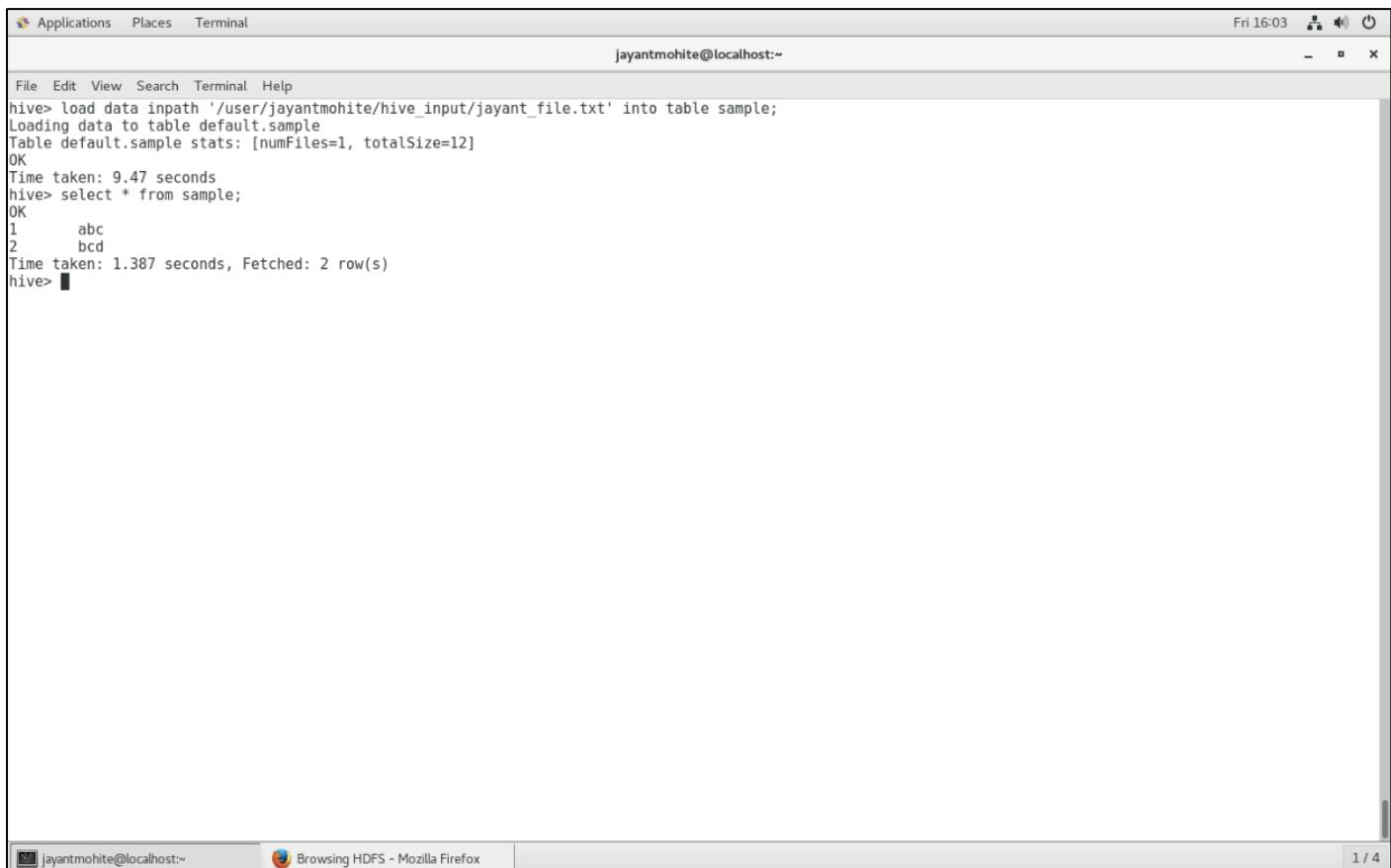
```
hive> load data inpath '/user/jayantmohite/hive_input/jayant_file.txt' into table  
sample;
```



A screenshot of a Linux terminal window titled "Terminal". The window shows the following command being run:

```
File Edit View Search Terminal Help  
hive> create table sample(id int, name string) row format delimited fields terminated by '\t' stored as textfile;  
OK  
Time taken: 23.728 seconds  
hive> █
```

The terminal window has a standard Linux desktop interface with a menu bar at the top and a scroll bar on the right. The bottom of the window shows the user's session information and the Firefox browser icon.



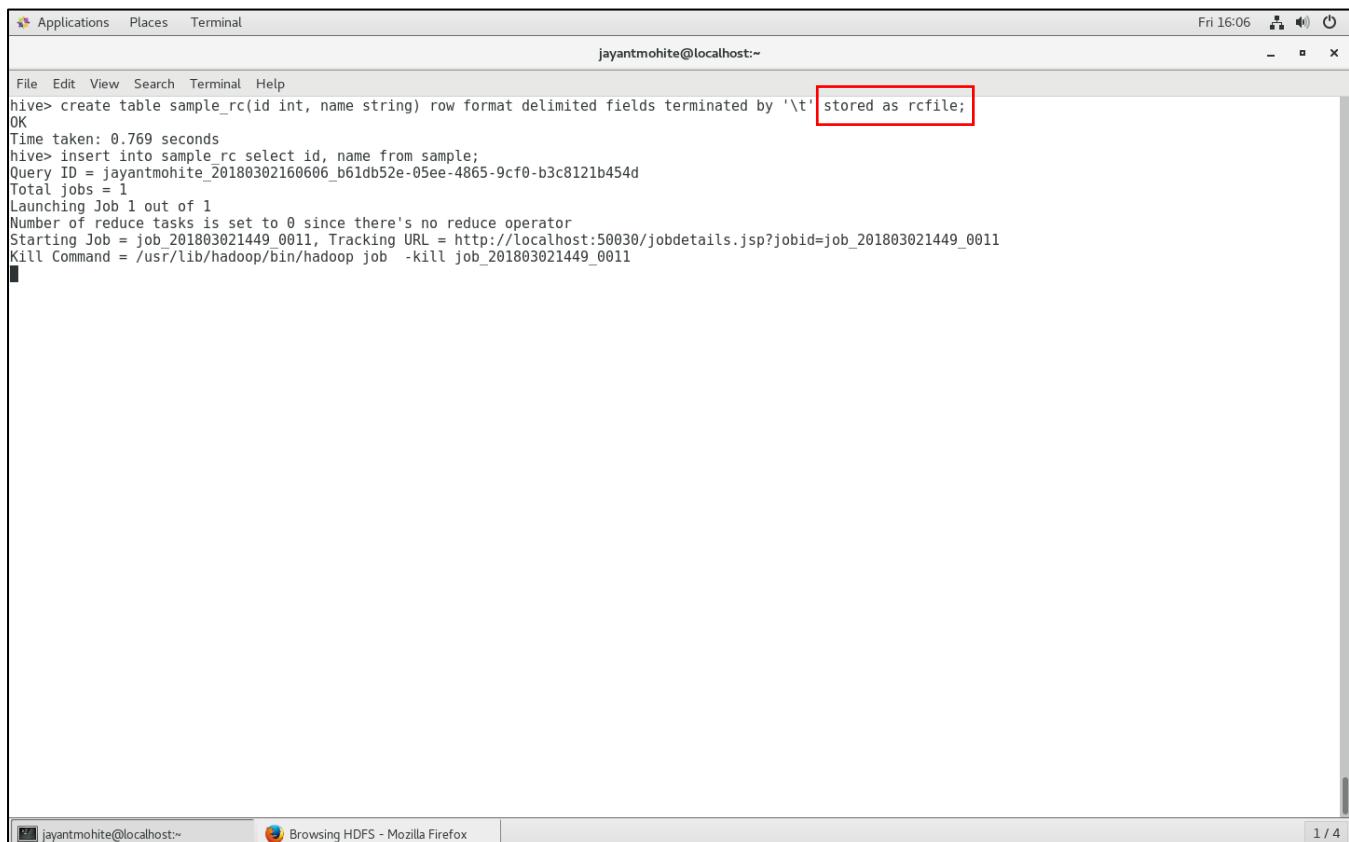
A screenshot of a Linux terminal window titled "jayantmohite@localhost:~". The window shows the following Hive session:

```
hive> load data inpath '/user/jayantmohite/hive_input/jayant_file.txt' into table sample;
Loading data to table default.sample
Table default.sample stats: [numFiles=1, totalSize=12]
OK
Time taken: 9.47 seconds
hive> select * from sample;
OK
1      abc
2      bcd
Time taken: 1.387 seconds, Fetched: 2 row(s)
hive>
```

The terminal window has a title bar with "Applications", "Places", and "Terminal" and a status bar at the bottom showing "jayantmohite@localhost:~" and "Browsing HDFS - Mozilla Firefox". A vertical scroll bar is visible on the right side of the terminal window.

Working with other file formats is slightly different than working with the textfile format. You cannot load data directly from a input file into these tables. You need to have a textfile format table and then you can load data from this table to your table of any other file format

Example RCFile Table

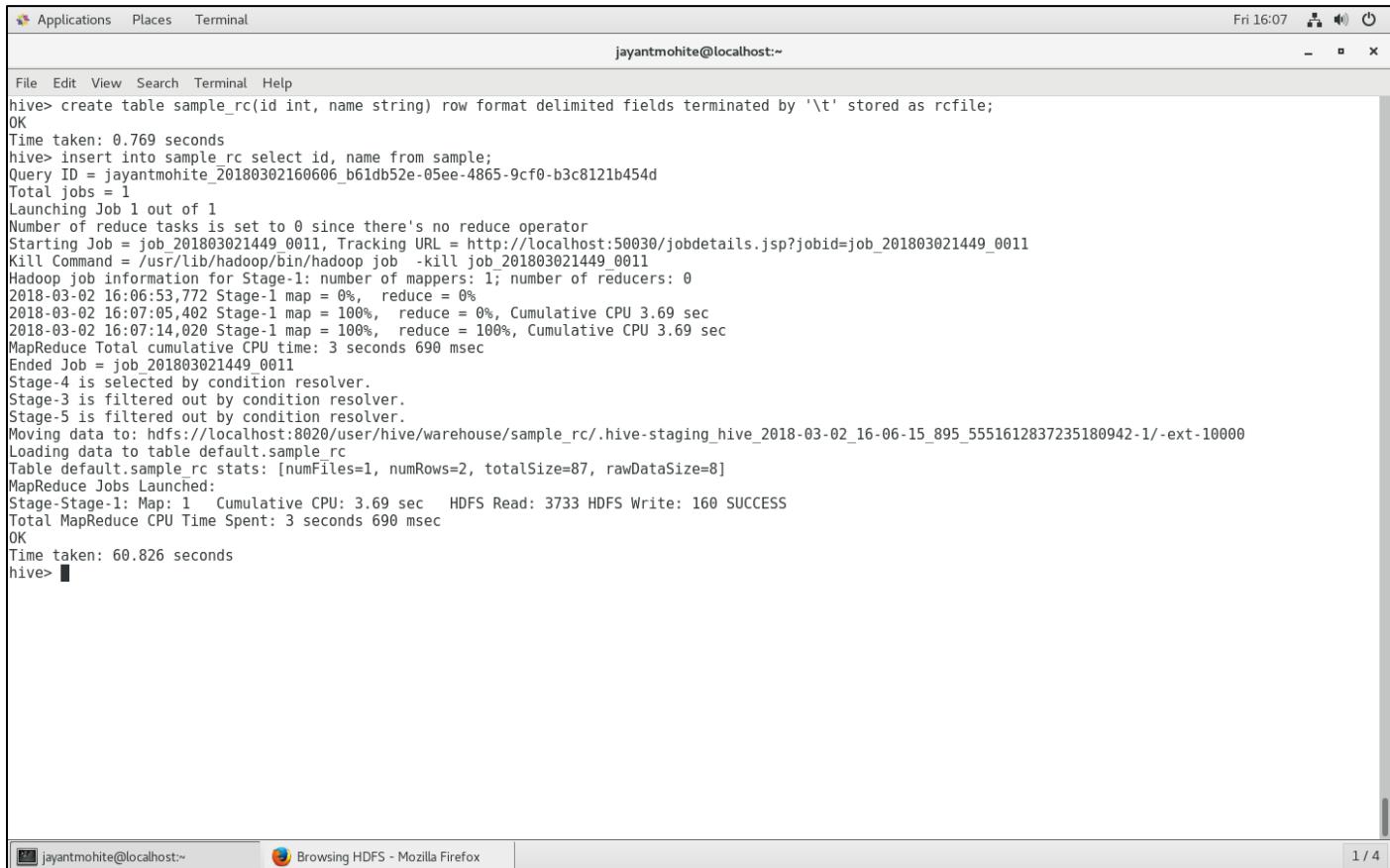


The screenshot shows a terminal window titled 'jayantmohite@localhost:~'. The window contains the following Hive session:

```
hive> create table sample_rc(id int, name string) row format delimited fields terminated by '\t' stored as rcfile;
OK
Time taken: 0.769 seconds
hive> insert into sample_rc select id, name from sample;
Query ID = jayantmohite_20180302160606_b61db52e-05ee-4865-9cf0-b3c8121b454d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201803021449_0011, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201803021449_0011
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_201803021449_0011
```

hive> create table sample_rc(id int, name string) row format delimited fields terminated by '\t' stored as RCFile;
(do note that everything else in the syntax is same. What has changed is only the file format.)

hive> insert into table <enter your RCFile table name as sample_rc> select id, name from <enter your textfile table name as sample>;
(this will load data from the textfile table into the newly created RCFile table)

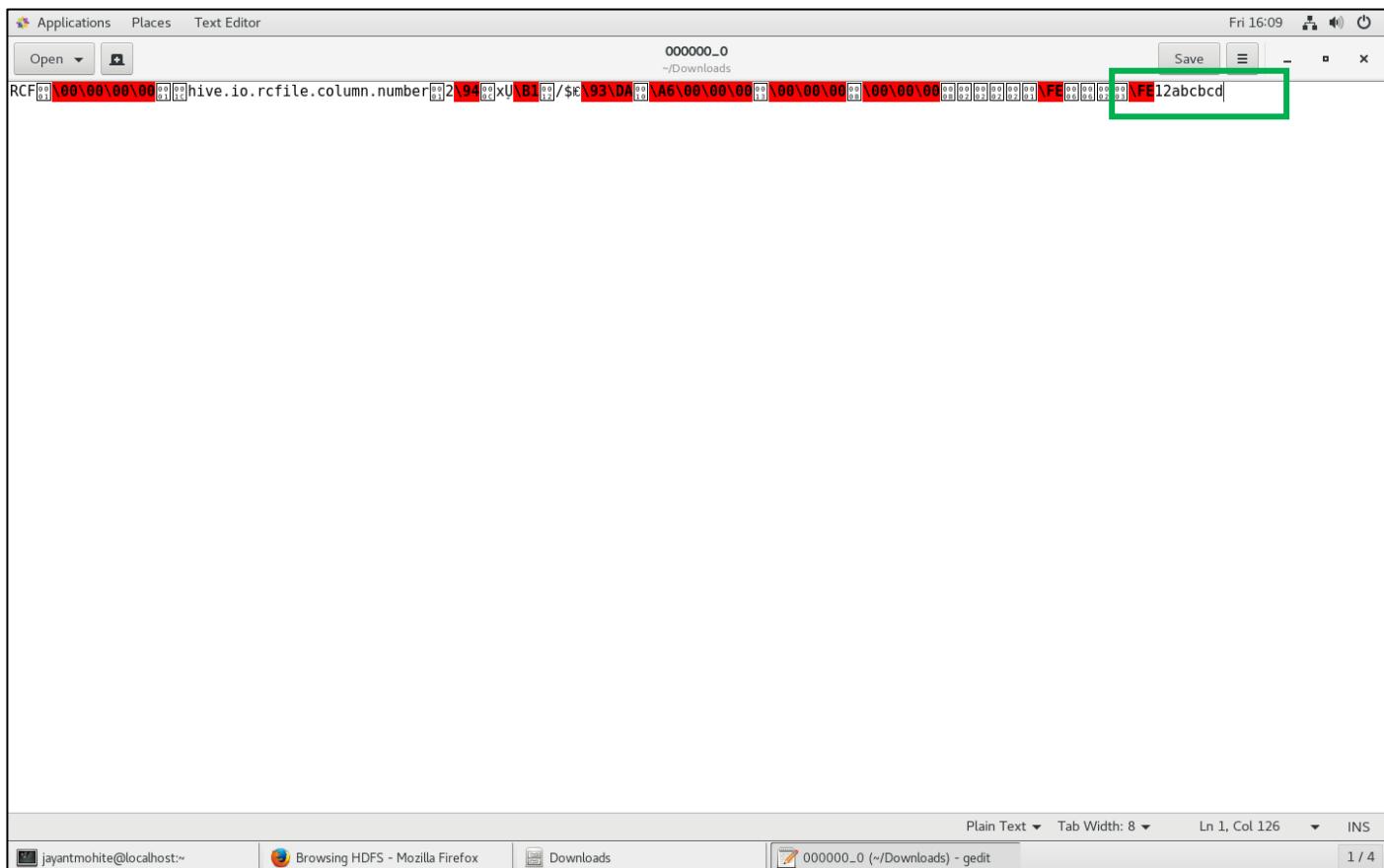


```

Applications Places Terminal Fri 16:07
jayantmohite@localhost:~ File Edit View Search Terminal Help
hive> create table sample_rc(id int, name string) row format delimited fields terminated by '\t' stored as rcfile;
OK
Time taken: 0.769 seconds
hive> insert into sample_rc select id, name from sample;
Query ID = jayantmohite_20180302160606_b61db52e-05ee-4865-9cf0-b3c8121b454d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201803021449_0011, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201803021449_0011
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_201803021449_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-03-02 16:06:53,772 Stage-1 map = 0%, reduce = 0%
2018-03-02 16:07:05,402 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.69 sec
2018-03-02 16:07:14,020 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.69 sec
MapReduce Total cumulative CPU time: 3 seconds 690 msec
Ended Job = job_201803021449_0011
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:8020/user/hive/warehouse/sample_rc/.hive-staging_hive_2018-03-02_16-06-15_895_5551612837235180942-1/-ext-10000
Loading data to table default.sample_rc
Table default.sample_rc stats: [numFiles=1, numRows=2, totalSize=87, rawDataSize=8]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 3.69 sec HDFS Read: 3733 HDFS Write: 160 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 690 msec
OK
Time taken: 60.826 seconds
hive> 

```

The terminal window shows the execution of a Hive command to create a table and insert data. It includes logs from the MapReduce job, such as mapper and reducer progress, cumulative CPU times, and HDFS read/write statistics.



```

Plain Text ▾ Tab Width: 8 ▾ Ln 1, Col 126 ▾ INS
jayantmohite@localhost:~ | Browsing HDFS - Mozilla Firefox | Downloads | 000000_0 (~/Downloads) - gedit | 1 / 4

```

The text editor displays the raw hex dump of an RC file named '000000_0'. The file content is shown in two columns: hex values and ASCII characters. A green box highlights the last few bytes of the file, which are 'FE12abcbcd'. The file is located in the 'Downloads' directory.

```

Applications Places Terminal Fri 16:10
jayantmohite@localhost:~ - x

File Edit View Search Terminal Help
hive> create table sample_orc(id int, name string) row format delimited fields terminated by '\t' stored as orcfile;
OK
Time taken: 3.537 seconds
hive> insert into sample_orc select id, name from sample;
Query ID = jayantmohite_20180302160909_7ca2b54a-f8df-416a-ab84-e6b6a39f6fc
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201803021449_0012, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201803021449_0012
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_201803021449_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-03-02 16:18:26,629 Stage-1 map 0%, reduce = 0%
2018-03-02 16:18:41,411 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.64 sec
2018-03-02 16:18:49,868 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.64 sec
MapReduce Total cumulative CPU time: 3 seconds 640 msec
Ended Job = job_201803021449_0012
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:8020/user/hive/warehouse/sample_orc/.hive-staging_hive_2018-03-02_16-09-50_717_4658142098497828555-1/-ext-10000
Loading data to table default.sample_orc
Table default.sample_orc stats: [numFiles=1, numRows=2, totalSize=295, rawDataSize=182]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 3.64 sec   HDFS Read: 3796 HDFS Write: 370 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 640 msec
OK
Time taken: 61.353 seconds
hive> 

```

jayantmohite@localhost:~ Browsing HDFS - Mozilla Firefox Downloads 1 / 4

Commands:

hive> create table sample_orc(id int, name string) row format delimited fields terminated by '\t' stored as ORCFile;
(do note that everything else in the syntax is same. What has changed is only the file format.)

hive> insert into table <enter your ORCFile table name as sample_orc> select id, name from <enter your textfile table name as sample>;
(this will load data from the textfile table into the newly created ORCFile table)

```
000000_0(1)
~/Downloads
```

The screenshot shows a terminal window with a text editor open. The title bar says '000000_0(1)' and '~/Downloads'. The editor interface includes 'Open', 'Save', and other standard window controls. The main pane displays a large amount of binary data as hex values. The data starts with '000000_00' and continues with various patterns of hex digits. The bottom of the terminal window shows a tab bar with 'Plain Text', 'Browsing HDFS - Mozilla Firefox', 'Downloads', and the current file '000000_0(1) (~/Downloads) - gedit'. There is also a status bar at the bottom right indicating 'Ln 1, Col 1' and 'INS'.

Example of Create Table As Select

In this kind of operation we will be creating a new table from the result set of a query

```

File Edit View Search Terminal Help
jayantmohite@localhost:~
hive> create table sample_ctas row format delimited fields terminated by '\t' stored as textfile as select id,name from sample where id=1;
Query ID = jayantmohite_20180302161414_67f8d14d-874d-4139-a175-46ff82fe0d44
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201803021449_0013, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201803021449_0013
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_201803021449_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-03-02 16:14:52,295 Stage-1 map = 0%, reduce = 0%
2018-03-02 16:15:09,323 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.7 sec
2018-03-02 16:15:17,570 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.7 sec
MapReduce Total cumulative CPU time: 4 seconds 700 msec
Ended Job = job_201803021449_0013
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:8020/user/hive/warehouse/.hive-staging_hive_2018-03-02_16-14-20_588_4052884696955713042-1/-ext-10001
Moving data to: hdfs://localhost:8020/user/hive/warehouse/sample_ctas
Table default.sample_ctas stats: [numFiles=1, numRows=1, totalSize=6, rawDataSize=5]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 4.7 sec HDFS Read: 3816 HDFS Write: 81 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 700 msec
OK
Time taken: 58.334 seconds
hive> select * from sample_ctas;
OK
1          abc
Time taken: 0.307 seconds, Fetched: 1 row(s)
hive>

```

Commands:

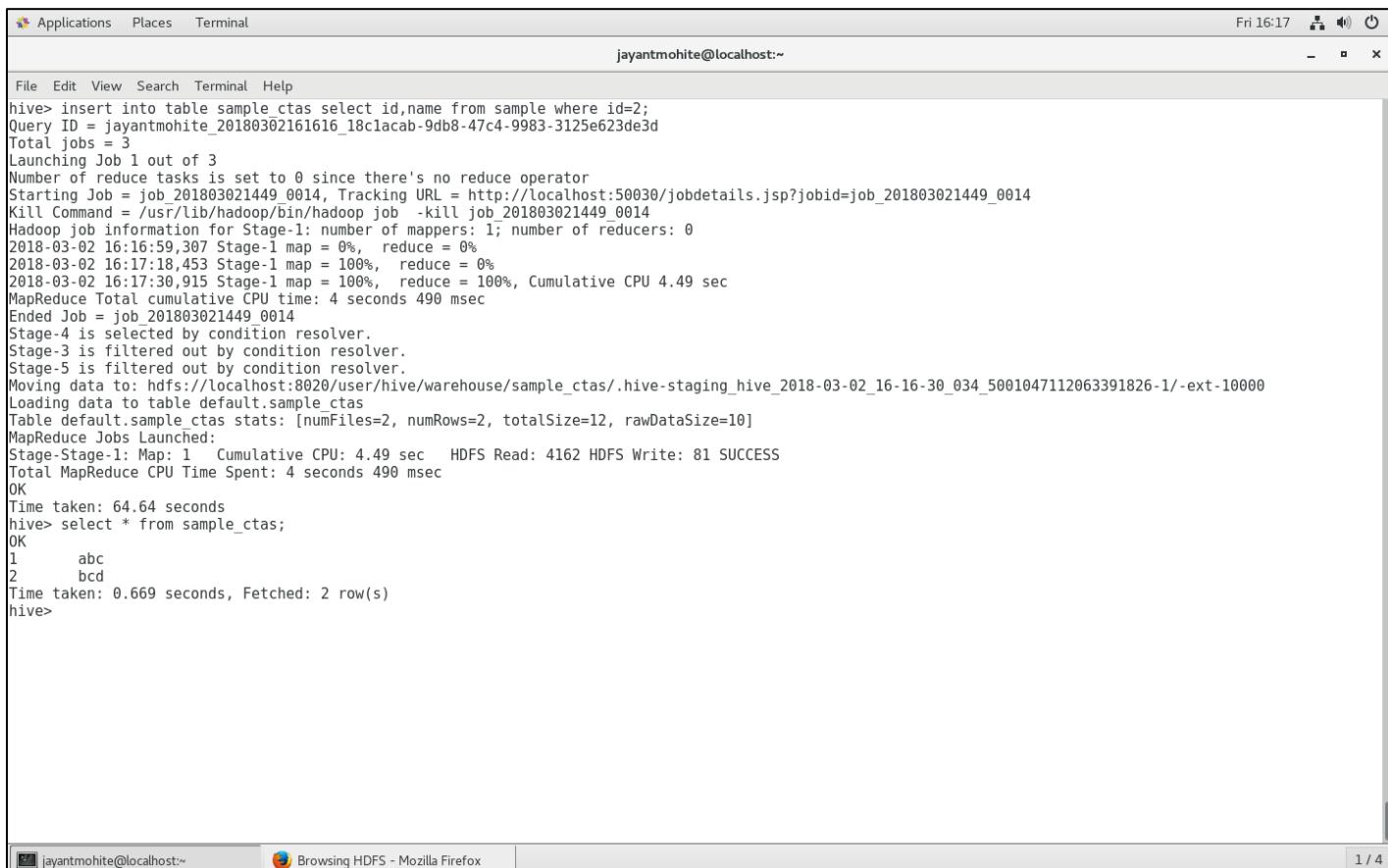
`hive> create table sample_ctas row format delimited fields terminated by '\t' stored as textfile as select id, name from sample where id = 1;`

In this command the query `select id, name from sample where id = 1` will be execute first. The schema that this query returns will become the schema of the new table and the data that this query returns will become the data of the new table.

In this command both create table and load data commands are combined together.

Example of Hive Insert Into

In this example, we will append some data to an existing table from another existing table.



A screenshot of a Linux terminal window titled "jayantmohite@localhost:~". The terminal shows the following Hive session:

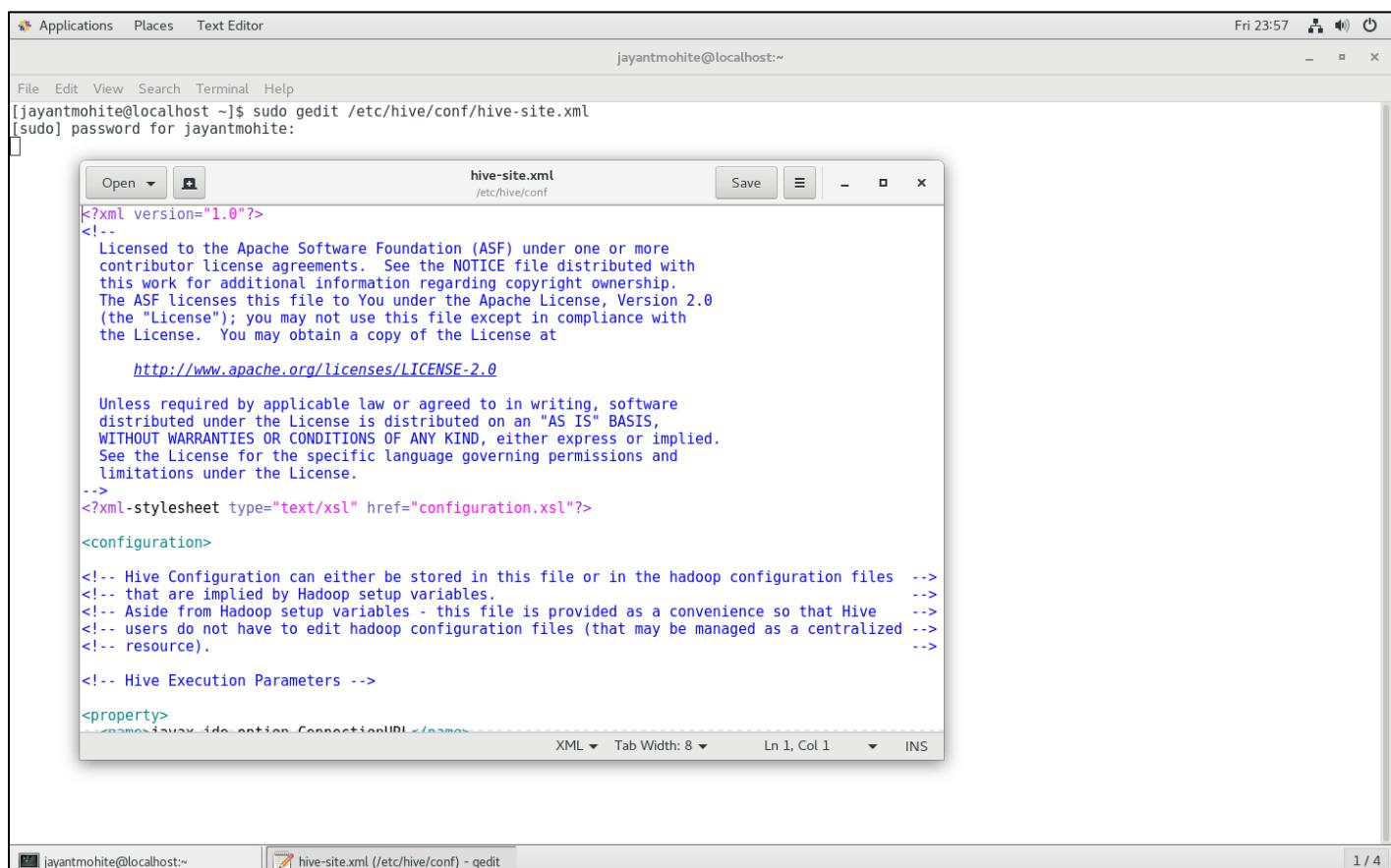
```
File Edit View Search Terminal Help
hive> insert into table sample_ctas select id,name from sample where id=2;
Query ID = jayantmohite_20180302161616_18c1acab-9db8-47c4-9983-3125e623de3d
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201803021449_0014, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201803021449_0014
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_201803021449_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-03-02 16:16:59,307 Stage-1 map = 0%,  reduce = 0%
2018-03-02 16:17:18,453 Stage-1 map = 100%,  reduce = 0%
2018-03-02 16:17:30,915 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.49 sec
MapReduce Total cumulative CPU time: 4 seconds 490 msec
Ended Job = job_201803021449_0014
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:8020/user/hive/warehouse/sample_ctas/.hive-staging_hive_2018-03-02_16-16-30_034_5001047112063391826-1/-ext-10000
Loading data to table default.sample_ctas
Table default.sample_ctas stats: [numFiles=2, numRows=2, totalSize=12, rawDataSize=10]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Cumulative CPU: 4.49 sec   HDFS Read: 4162 HDFS Write: 81 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 490 msec
OK
Time taken: 64.64 seconds
hive> select * from sample_ctas;
OK
1      abc
2      bcd
Time taken: 0.669 seconds, Fetched: 2 row(s)
hive>
```

Commands:

```
hive> insert into table sample_ctas select id, name from sample where id = 2;
```

In this command the query select id, name from sample where id = 2 will be executed first. The schema of the target table should match the schema of the query result to avoid bad data. The data returned by the query will be appended to the existing data in the target table.

Now before moving into other type of operations in Hive, lets define some properties in the hive configuration file named `hive-site.xml` located at location `/etc/hive/conf/hive-site.xml`. Also lets create some input files that we will use in the upcoming examples.



```
[jayantmohite@localhost ~]$ sudo gedit /etc/hive/conf/hive-site.xml
[jayantmohite@localhost ~]$
```

The screenshot shows a terminal window with the command `sudo gedit /etc/hive/conf/hive-site.xml` run. The file content is as follows:

```
<?xml version="1.0"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements. See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License. You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<xmlesstylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
  <!-- Hive Configuration can either be stored in this file or in the hadoop configuration files -->
  <!-- that are implied by Hadoop setup variables. -->
  <!-- Aside from Hadoop setup variables - this file is provided as a convenience so that Hive -->
  <!-- users do not have to edit hadoop configuration files (that may be managed as a centralized -->
  <!-- resource). -->
  <!-- Hive Execution Parameters -->
<property>
```

Applications Places Text Editor *hive-site.xml /etc/hive/conf Sat 00:08 Save - x

```

</property>
<property>
  <name>hive.support.concurrency</name>
  <value>true</value>
</property>

<property>
  <name>hive.enforce.bucketing</name>
  <value>true</value>
</property>

<property>
  <name>hive.exec.dynamic.partition</name>
  <value>true</value>
</property>

<property>
  <name>hive.exec.dynamic.partition.mode</name>
  <value>nonstrict</value>
</property>

<property>
  <name>hive.txn.manager</name>
  <value>org.apache.hadoop.hive.ql.lockmgr.DbTxnManager</value>
</property>

<property>
  <name>hive.compactor.initiator.on</name>
  <value>true</value>
</property>

<property>
  <name>hive.compactor.worker.threads</name>
  <value>1</value>
</property>

<property>
  <name>hive.in.test</name>
  <value>true</value>
</property>
</configuration>

```

for transaction support

for bucketing support

for dynamic partition support

for transaction support

XML Tab Width: 8 Ln 66, Col 1 INS

jayanmohite@localhost:~ *hive-site.xml (/etc/hive/conf) - gedit 1 / 4

We will be creating 3 input files with almost same data. Major difference between these file will be that one of them will have four columns including one column as the country and other two won't have the country column.

For assumption we consider that we have collected one data set from a global domain so we need have the country column into it and rest we have collected from our India and US servers so we know the country and hence that column is not required.

In hive the Map Reduce program is executed on the entire directory of the table. Lets assume that there are 10 file with total of 10 million records. Out of which only 100 records belong to India. If we try to fire a query over this table, still it will query all 10 files as at the backend what happens is simply a Map Reduce program on the entire directory. So the time required for this operation will not be optimized. But what if we are able to created sub directories under our table directory and every sub directory contains data of distinct countries. In this case whenever we execute a query with a where clause on the country column, the operation will be more optimized as the operation will execute on the countries specific sub directory rather than executing on the complete directory. This is called as [Partitioning](#).

But there can be two cases in which we can get the data.

Case 1: The value of the partition column is available in the data itself. As in case of our first input file with the country column.

In this case we switch to a type of partitioning which automatically identifies the partitions and is called as Dynamic Partitioning

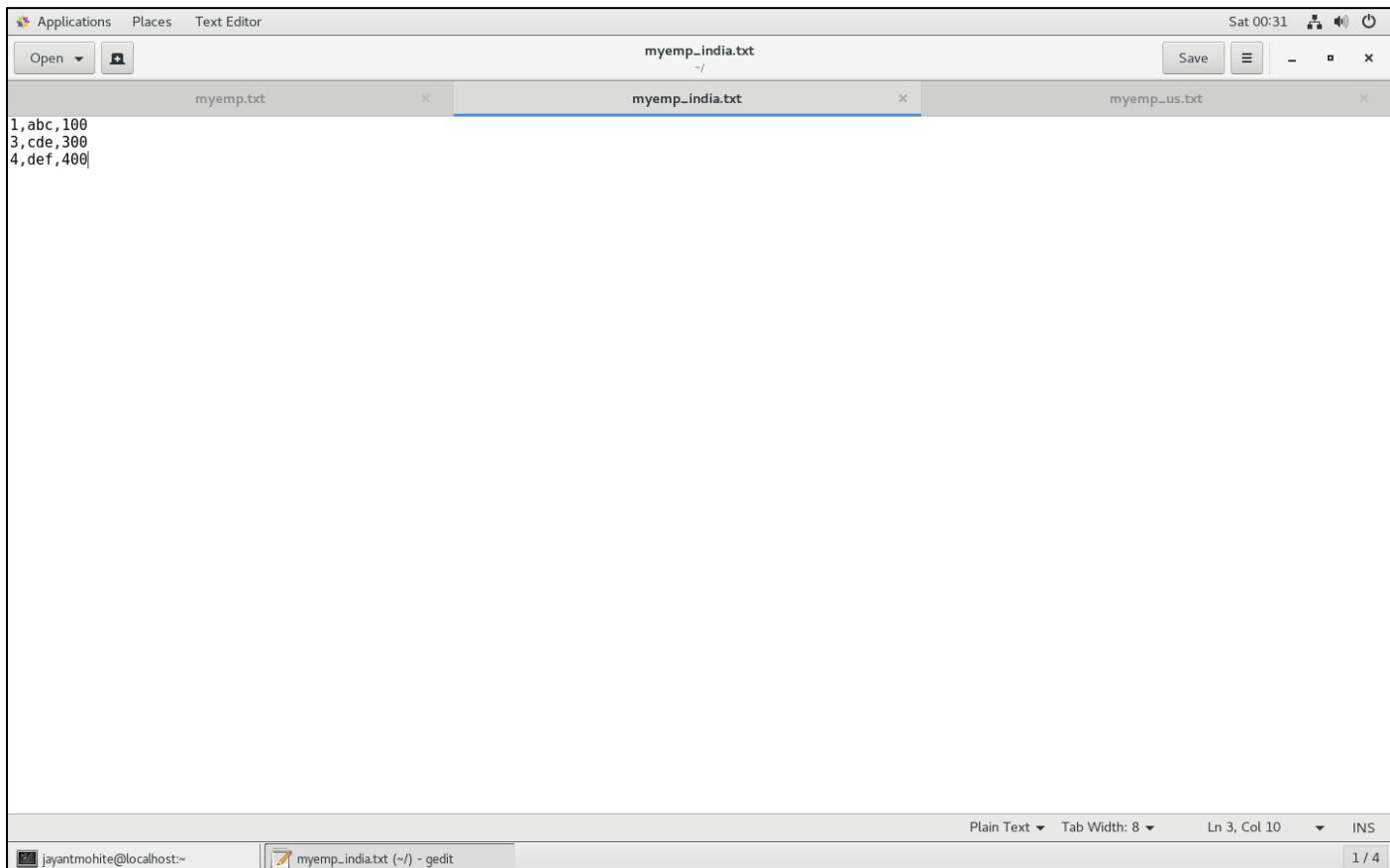
Case 2: The value of the partition column is not available in the data but we have separate inputs for each value as in case of our remaining two input files.

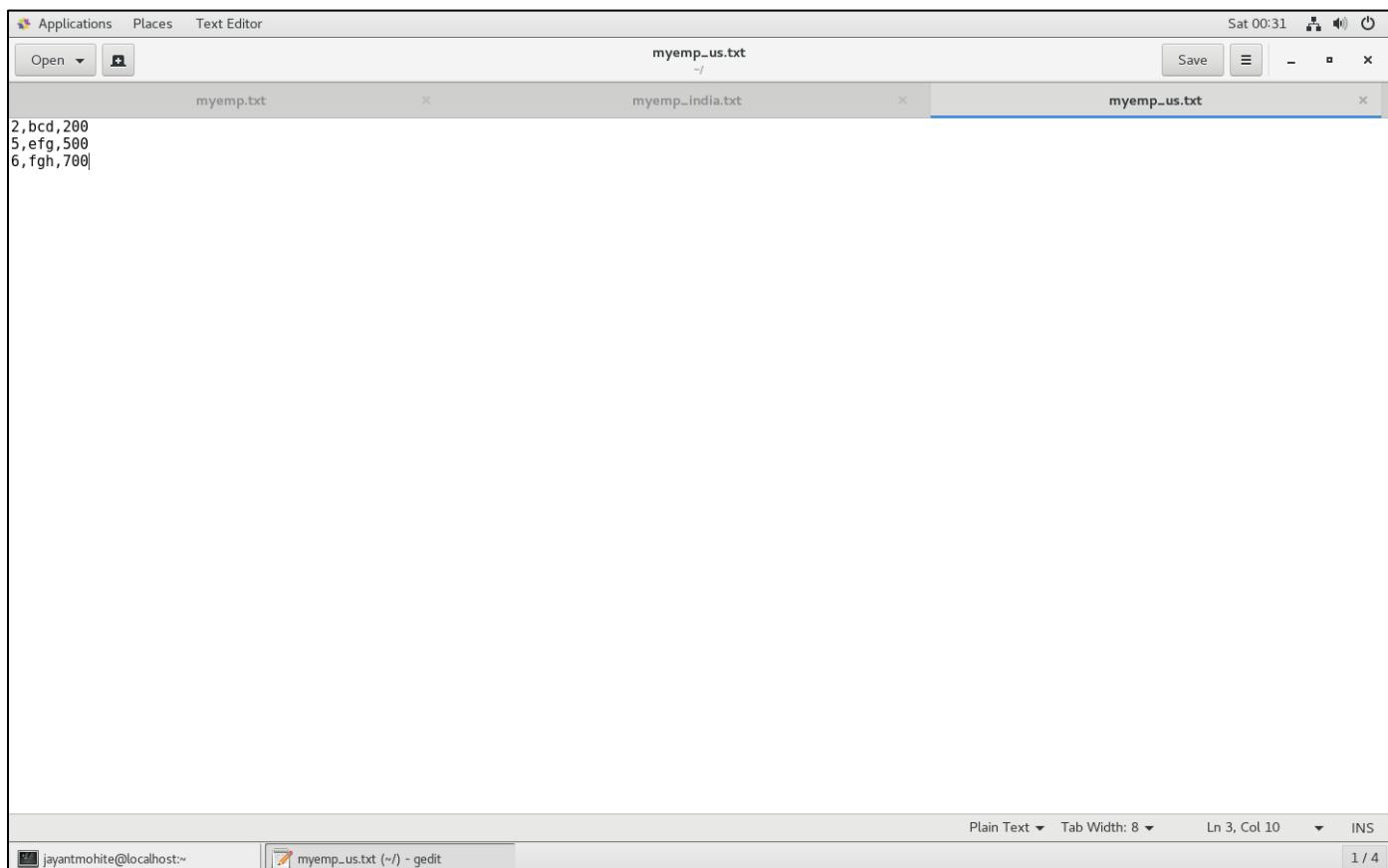
In this case we switch to a type of partitioning in which we manually specify the value of each partition and this is called as static partitioning.

The screenshot shows a Linux desktop interface. At the top, there is a menu bar with 'Applications', 'Places', and 'Text Editor'. On the right side of the screen, there is a system tray showing the date 'Sat 00:27' and icons for battery, signal strength, and power. In the center, there is a terminal window titled 'jayantmohite@localhost:~' and a text editor window titled 'myemp.txt (~) - gedit'. The text editor contains the following data:

```
1,abc,100,india
2,bcd,200,us
3,cde,300,india
4,def,400,india
5,efg,500,us
6,fgh,700,aus|
```

The terminal window shows the command 'Plain Text' selected, a tab width of 8, and line 6, column 14. The status bar at the bottom indicates '1 / 4'.





Now lets create our primary table.

The screenshot shows a terminal window with the following session:

```
Applications Places Terminal Sat 00:35 jayantmohite@localhost:~  
File Edit View Search Terminal Help  
hive> create table myemp(id int, name string, salary int, country string) row format delimited fields terminated by ',' stored as textfile;  
OK  
Time taken: 23.978 seconds  
hive> load data local inpath '/home/jayantmohite/myemp.txt' into table myemp;  
Loading data to table default.myemp  
Table default.myemp stats: [numFiles=1, totalSize=88]  
OK  
Time taken: 3.56 seconds  
hive> select * from myemp;  
OK  
1 abc 100 india  
2 bcd 200 us  
3 cde 300 india  
4 def 400 india  
5 efg 500 us  
6 fgh 700 aus  
Time taken: 3.344 seconds, Fetched: 6 row(s)  
hive>
```

The terminal window has a title bar "jayantmohite@localhost:~". The bottom status bar shows "jayantmohite@localhost:~" and "*Untitled Document 1 - gedit". A page number "1 / 4" is visible in the bottom right corner.

Command for creating a partitioned table

```
hive> create table myemp_partitioned(id int, name string, salary int) partitioned by (country string) row format delimited fields terminated by ',' stored as textfile;  
(In this command we are creating a table that has 4 columns and the country column is the one on which the table is partitioned)
```

Loading data in Static Partitioning

```
hive> load data local inpath '/home/jayantmohite/myemp_india.txt' into table myemp_partitioned partition(country = "india");  
(In this command we load data from a file which has 3 columns. The value of the 4th column which is the partition column is provided manually by us.)
```

```
hive> load data local inpath '/home/jayantmohite/myemp_us.txt' into table myemp_partitioned partition(country = "us");
```

The screenshot shows a terminal window titled 'jayantmohite@localhost:~'. The window contains the following Hive session:

```
File Edit View Search Terminal Help  
jayantmohite@localhost:~  
hive> create table myemp_partitioned(id int, name string, salary int) partitioned by (country string) row format delimited fields terminated by ',' stored as textfile;  
OK  
Time taken: 0.338 seconds  
hive> load data local inpath '/home/jayantmohite/myemp_india.txt' into table myemp_partitioned partition(country="india");  
Loading data to table default.myemp_partitioned partition (country=india)  
Partition default.myemp_partitioned{country=india} stats: [numFiles=1, numRows=0, totalSize=30, rawDataSize=0]  
OK  
Time taken: 2.058 seconds  
hive>
```

The screenshot shows a Firefox browser window titled "Browsing HDFS - Mozilla Firefox". The address bar displays "localhost:50070/explorer.html#/user/hive/warehouse/myemp_partitioned". The main content area is titled "Browse Directory" and shows a table of file metadata for a single file named "country=india". The table columns are: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The file details are: drwxrwxrwx, jayantmohite, supergroup, 0 B, Sat Mar 03 00:37:15 -0800 2018, 0, 0 B, country=india. Below the table, the text "Hadoop, 2017." is visible. The bottom of the browser window shows the status bar with "jayantmohite@localhost:~" and "[*Untitled Document 1 - gedit]".

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	jayantmohite	supergroup	0 B	Sat Mar 03 00:37:15 -0800 2018	0	0 B	country=india

Hadoop, 2017.

```

File Edit View Search Terminal Help
hive> load data local inpath '/home/jayantmohite/myemp_india.txt' into table myemp_partitioned partition(country="india");
Loading data to table default.myemp_partitioned partition (country=india)
Partition default.myemp_partitioned{country=india} stats: [numFiles=1, numRows=0, totalSize=30, rawDataSize=0]
OK
Time taken: 2.058 seconds
hive> load data local inpath '/home/jayantmohite/myemp_us.txt' into table myemp_partitioned partition(country="us");
Loading data to table default.myemp_partitioned partition (country=us)
Partition default.myemp_partitioned{country=us} stats: [numFiles=1, numRows=0, totalSize=30, rawDataSize=0]
OK
Time taken: 3.173 seconds
hive> insert into table myemp_partitioned partition(country) select id,name,salary,country from myemp;
Query ID = jayantmohite_20180303004242_b408c31f-a3d7-43a5-9174-3159ef3b96fa
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201803022342_0001, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201803022342_0001
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_201803022342_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-03-03 00:42:48,431 Stage-1 map = 0%, reduce = 0%
2018-03-03 00:43:04,640 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.84 sec
2018-03-03 00:43:14,996 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 2.84 sec
MapReduce Total cumulative CPU time: 2 seconds 840 msec
Ended Job = job_201803022342_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:8020/user/hive/warehouse/myemp_partitioned/.hive-staging_hive_2018-03-03_00-42-11_437_7671890836849167558-1/-ext-10000
Loading data to table default.myemp_partitioned partition (country=null)
    Time taken for load dynamic partitions : 1664
    Loading partition {country=aus}
    Loading partition {country=us}
    Loading partition {country=india}
    Time taken for adding to write entity : 15
Partition default.myemp_partitioned{country=aus} stats: [numFiles=1, numRows=1, totalSize=10, rawDataSize=9]
Partition default.myemp_partitioned{country=india} stats: [numFiles=2, numRows=3, totalSize=60, rawDataSize=27]
Partition default.myemp_partitioned{country=us} stats: [numFiles=2, numRows=2, totalSize=50, rawDataSize=18]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 2.84 sec   HDFS Read: 4184 HDFS Write: 266 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 840 msec
OK
Time taken: 69.896 seconds
hive> 
```

The terminal window has a title bar with 'Applications', 'Places', 'Terminal', 'Sat 00:43', and system icons. The bottom status bar shows 'jayantmohite@localhost:~', 'Untitled Document 1 - edit', 'Browsing HDFS - Mozilla Firefox', and '1 / 4'.

Loading data in Dynamic Partitioning

```
hive> insert into table myemp_partitioned partition (country) select id, name, salary, country from myemp;
```

In this command we load data into the partitioned table by name myemp_partitioned from a non-partitioned table by name myemp which we have created previously in this chapter.

Dynamic partitioning will automatically sense the distinct values of the partitioned columns and will create the required sub folders

In case the partition already exists, the data will be appended to the partition and if it does not exist, a new partition will be created.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	jayantmohite	supergroup	0 B	Sat Mar 03 00:43:03 -0800 2018	0	0 B	country=aus
drwxrwxrwx	jayantmohite	supergroup	0 B	Sat Mar 03 00:43:18 -0800 2018	0	0 B	country=india
drwxrwxrwx	jayantmohite	supergroup	0 B	Sat Mar 03 00:43:18 -0800 2018	0	0 B	country=us

Hadoop, 2017.

Hive Bucketing Example

Bucketing in Hive can be viewed as partitions created within partitions which is actually unconditional clustering based on number of buckets specified by the user.

Commands:

```
hive> create table myemp_bucket(id int, name string, salary int) partitioned by (country string) clustered by (id) into 3 buckets row format delimited fields terminated by ',' stored as textfile;
```

So basically this is the same syntax as for the table partitioning with only difference that we are further dividing all partitions into 3 sub-sections called as buckets. So data in every partition will be further split into 3 parts.

Browsing HDFS - Mozilla Firefox

localhost:50070/explorer.html#/user/hive/warehouse/myemp_bucket/country=india

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxrwxrwx	jayantmohite	supergroup	10 B	Sat Mar 03 00:48:44 -0800 2018	1	128 MB	00000_0
-rwxrwxrwx	jayantmohite	supergroup	20 B	Sat Mar 03 00:48:43 -0800 2018	1	128 MB	00001_0
-rwxrwxrwx	jayantmohite	supergroup	0 B	Sat Mar 03 00:49:09 -0800 2018	1	128 MB	00002_0

Go!

Hadoop, 2017.

jayantmohite@localhost:~ *Untitled Document 1 - gedit Browsing HDFS - Mozilla Firefox 1 / 4

Hive Transactions

Like any other RDBMS, even Hive supports transactional queries like insert, update and delete.

```

Applications Places Terminal Sat 00:57
jayantmohite@localhost:~>

File Edit View Search Terminal Help
hive> create table students_table(name varchar(64),age int, gpa decimal(3,2)) clustered by (age) into 2 buckets stored as orc tblproperties('transactional'=true');
OK
Time taken: 1.541 seconds
hive> insert into table students table values ('fred',35,1.28),('barney',32,2.32);
Query ID = jayantmohite_20180303005656_767952d8-6071-4cd4-922e-d72dd0925ff4
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 2
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201803022342_0003, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201803022342_0003
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_201803022342_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 2
2018-03-03 00:56:31,584 Stage-1 map = 0%, reduce = 0%
2018-03-03 00:56:49,269 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.06 sec
2018-03-03 00:57:16,077 Stage-1 map = 100%, reduce = 67%, Cumulative CPU 5.06 sec
2018-03-03 00:57:21,773 Stage-1 map = 100%, reduce = 83%, Cumulative CPU 5.06 sec
2018-03-03 00:57:22,814 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.06 sec
MapReduce Total cumulative CPU time: 14 seconds 310 msec
Ended Job = job_201803022342_0003
Loading data to table default.students_table
Table default.students_table stats: [numFiles=2, numRows=2, totalSize=1458, rawDataSize=0]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 2   Cumulative CPU: 14.31 sec   HDFS Read: 13228 HDFS Write: 1622 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 310 msec
OK
Time taken: 90.024 seconds
hive> select * from students_table;
OK
barney 32      2.32
fred    35      1.28
Time taken: 1.566 seconds, Fetched: 2 row(s)
hive> 
```

1 / 4

Applications Places Terminal Sat 01:01 jayantmohite@localhost:~

```

File Edit View Search Terminal Help
hive> update students table set name='jayant' where gpa >= 2.0;
Query ID = jayantmohite_20180303005959_1b941305-befb-44a0-bfcf-9e52ff5af117
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 2
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201803022342_0004, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201803022342_0004
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_201803022342_0004
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 2
2018-03-03 00:59:24,780 Stage-1 map = 0%, reduce = 0%
2018-03-03 00:59:57,391 Stage-1 map = 50%, reduce = 0%, Cumulative CPU 4.81 sec
2018-03-03 00:59:58,758 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 9.29 sec
2018-03-03 01:00:22,946 Stage-1 map = 100%, reduce = 33%, Cumulative CPU 9.29 sec
2018-03-03 01:00:24,021 Stage-1 map = 100%, reduce = 67%, Cumulative CPU 9.29 sec
2018-03-03 01:00:30,374 Stage-1 map = 100%, reduce = 83%, Cumulative CPU 13.9 sec
2018-03-03 01:00:31,400 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 19.02 sec
MapReduce Total cumulative CPU time: 19 seconds 20 msec
Ended Job = job_201803022342_0004
Loading data to table default.students_table
Table default.students table stats: [numFiles=3, numRows=2, totalSize=2196, rawDataSize=0]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2 Reduce: 2 Cumulative CPU: 19.02 sec HDFS Read: 23204 HDFS Write: 868 SUCCESS
Total MapReduce CPU Time Spent: 19 seconds 20 msec
OK
Time taken: 100.095 seconds
hive> select * from students_table;
OK
jayant 32 2.32
fred 35 1.28
Time taken: 1.005 seconds, Fetched: 2 row(s)
hive> 
```

jayantmohite@localhost:~ *Untitled Document 1 - gedit Browsing HDFS - Mozilla Firefox 1 / 4

A screenshot of a Linux terminal window titled "Terminal". The window shows a session for user "jayantmohite" at "localhost". The terminal displays the following Hive command and its execution:

```
hive> explain select * from students_table;
OK
STAGE DEPENDENCIES:
 Stage-0 is a root stage

STAGE PLANS:
 Stage: Stage-0
 Fetch Operator
 limit: -1
 Processor Tree:
  TableScan
   alias: students_table
   Statistics: Num rows: 2 Data size: 2196 Basic stats: COMPLETE Column stats: NONE
   Select Operator
    expressions: name (type: varchar(64)), age (type: int), gpa (type: decimal(3,2))
    outputColumnNames: _col0, _col1, _col2
    Statistics: Num rows: 2 Data size: 2196 Basic stats: COMPLETE Column stats: NONE
    ListSink

Time taken: 1.54 seconds, Fetched: 17 row(s)
hive>
```

The terminal window has a red box highlighting the "STAGE DEPENDENCIES" and "STAGE PLANS" sections. At the bottom of the terminal window, there are several icons for file management and system monitoring.

In this way we can enjoy the features of a traditional database on the distributed data storage of Hadoop without much complexity. This is comparatively slower than the traditional databases but that's ok as long as we are concerned only about data analysis. The advantage is that we can now work on a very large scale of data in a single go.

11. Apache Pig

If you have some data of a Car Showroom and if I tell you to calculate the number of eligible people that have applied for a loan to buy a Ferrari in the year 2015, by evaluating certain criteria like, if the buyer is a male then, check his wife's Pan Card details if he is married else check his mother's Pan Card details. If any of them (Mother, Wife) falls under the Second Taxation Slab then calculate the minimum EMI that he can pay and if the EMI is below 2,00,000 INR then check the current Salary of that person, if it is more than 3,00,000 INR, then mark him as eligible else discard his application. And do the same kind of calculation if the buyer is a woman.

Would you be able to do this using SQL? The answer simply is a big NO. Of course if you are a SQL expert then you can definitely do it but still the query will have many Joins and Nested queries which will make the query very slow and make this approach unfeasible. So how do we do achieve this then? We have a simple approach of doing this which is an alternate to or the advanced version of SQL i.e. [PL/SQL](#). By using PL/SQL we can easily use the conditional and looping statements and solve our problem.

Similarly if there are any operations that cannot be solved using [HIVE](#) or some tasks that may turn out to be tedious if done using HIVE then to make life easy, we have an interface of PL/SQL on Hadoop, and the topic of this chapter that is [PIG](#).

Introduction

PIG is referred to as a PL/SQL interface on top of Hadoop. We can use PIG and implement different programming aspects like loops and conditions while working out with our tabular data.

```
[wisdom@localhost ~]$ pig
2015-10-01 03:56:26,697 [main] INFO org.apache.pig.Main Apache Pig version 0.11.0-cdh4.6.0 (r: unknown) compiled Feb 26 2014, 03:02:34
2015-10-01 03:56:26,698 [main] INFO org.apache.pig.Main - Logging error messages to: /home/wisdom/pig_1443686186695.log
2015-10-01 03:56:26,743 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/wisdom/.pigbootup not found
2015-10-01 03:56:27,005 [main] WARN org.apache.hadoop.conf.Configuration - fs.default.name is deprecated. Instead, use fs.defaultFS
2015-10-01 03:56:27,005 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:8020
2015-10-01 03:56:27,754 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to map-reduce job tracker at: localhost:8021
2015-10-01 03:56:27,756 [main] WARN org.apache.hadoop.conf.Configuration - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> ■
```

PIG is a Scripting Language that can explore huge data sets of size TB and PB easily. It provides a higher level language called as **PIG Latin** which can be executed in the PIG shell called **Grunt**.

Unlike HIVE, PIG stores the data and the schema of the data in a single unit at the same place inside the HDFS. In PIG, a Single data element is called as **atom**, collection of atoms is called as **tuple**, and collection of tuples is called as **BAG**. Typically, PIG would load a dataset in a bag and then create new bag by modifying the existing one.

In the example below, the *movies_this_week* file is loaded in *movies* bag, then a new BAG called *hit_movies* is created for movies with ratings more than 4. Later a new bag *top_movies* is created and finally the results are stored in a file called *must_watch_movies*.

Every time we process the data inside a bag a new bag is created, as a result of the operation. In order to avoid the unnecessary additional replication of data, which results out of this bag creation of PIG, it does not implicitly store the bag. A bag is alive only for the current session in which it is created. If the session is

terminated, the bag is lost. In order to store the bag for future use, we need to explicitly store that bag.

```
PIG BAG  
movies = LOAD 'movies_this_week.txt'  
AS (id, name, rating, year);  
  
New PIG BAG  
hit_movies = FILTER movies BY rating > 4;  
  
Transformation on existing BAG  
  
top_movies = ORDER hit_movies BY rating DESC;  
  
PIG Explicit Storage  
STORE top_movies INTO 'must_watch_movies'
```

Data types in Pig

Data Type	Description
Int	32 Bit Integer
Long	64 Bit Integer
Float	32 Bit Floating Point Number
Double	64 Bit Floating Point Number
Chararray	String
Byte array	Blob

Basic Transformations in Pig

Operator	Description
Filter	Selects a set of tuples from a relation based on a condition.
For Each	Iterates the tuples of a relation, generating a data transformation.
Group	Groups the data in one or more relations.
Join	Joins two or more relations
Load	Loads data from the file system.
Store	Stores data in the file system.

Pig Configuration to avoid displaying logs

We will modify the logging properties of Pig and set it to a mode where only fatal error are displayed. For that we will have to make changes in log4j.properties file available in the configuration directory of Pig.

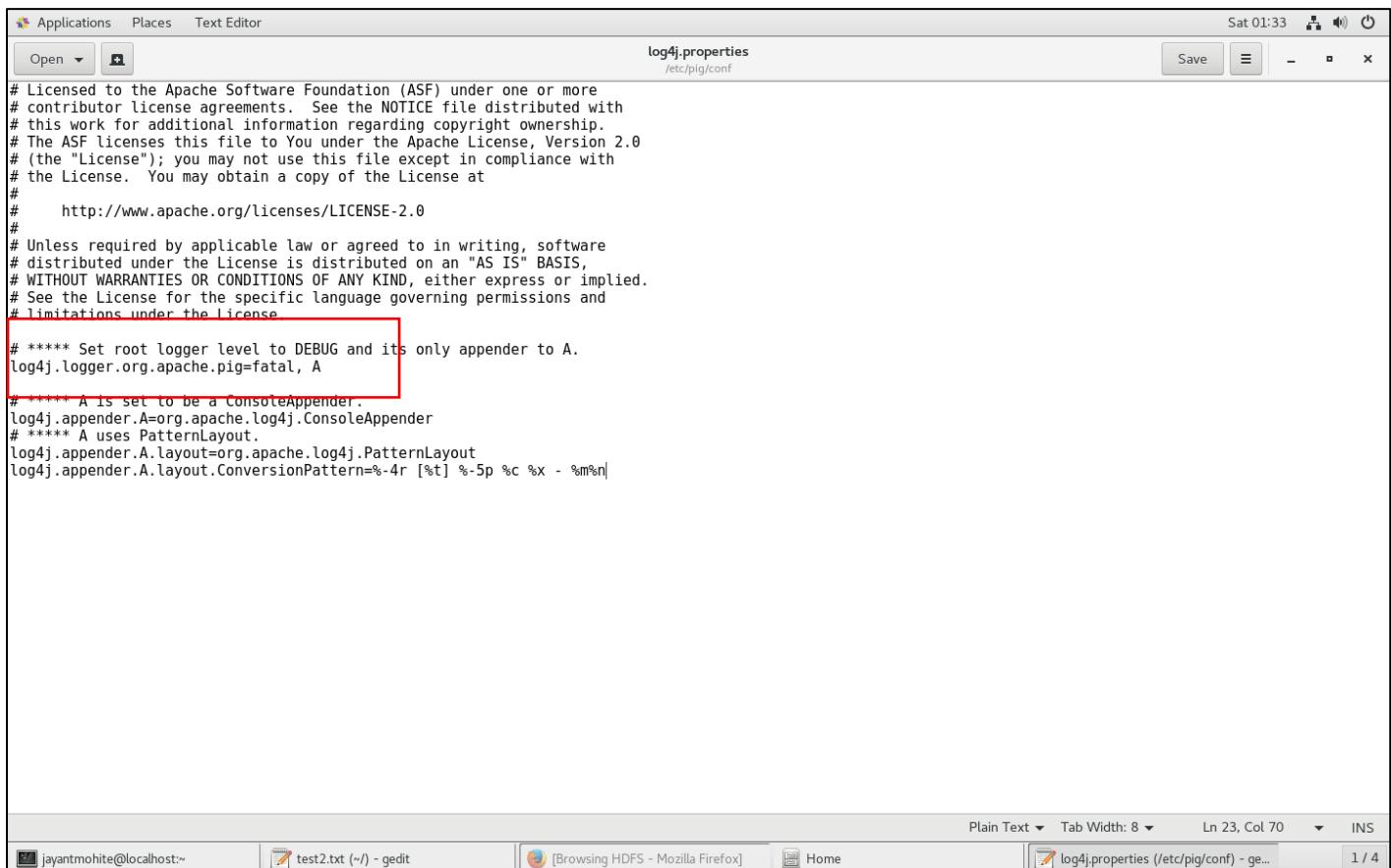
The screenshot shows a terminal window titled 'jayantmohite@localhost:~'. The user has run several commands to navigate to the configuration directory of Pig:

```
[jayantmohite@localhost ~]$ ls /etc/pig  
conf  conf.dist  
[jayantmohite@localhost ~]$ ls /etc/pig/conf  
build.properties  log4j.properties  pig.properties  
[jayantmohite@localhost ~]$ sudo gedit /etc/pig/conf/log4j.properties
```

A red box highlights the command 'sudo gedit /etc/pig/conf/log4j.properties'. After running this command, the terminal displays several 'WARNING' messages from the gedit application:

```
** (gedit:8899): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-language not supported  
** (gedit:8899): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported  
** (gedit:8899): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-position not supported
```

The terminal prompt '[jayantmohite@localhost ~]\$' is visible at the bottom.



```

Applications Places Text Editor
Open Save Sat 01:33
log4j.properties /etc/pig/conf
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License
#
# ***** Set root logger level to DEBUG and its only appender to A.
log4j.logger.org.apache.pig=fatal, A
#
# ***** A is set to be a ConsoleAppender.
log4j.appender.A=org.apache.log4j.ConsoleAppender
# ***** A uses PatternLayout.
log4j.appender.A.layout=org.apache.log4j.PatternLayout
log4j.appender.A.layout.ConversionPattern=%-4r [%t] %-5p %c %x - %m%n

```

Plain Text Tab Width: 8 Ln 23, Col 70 INS

jayantmohite@localhost:~ test2.txt (~) - gedit [Browsing HDFS - Mozilla Firefox] Home log4j.properties (/etc/pig/conf) - gedit 1 / 4

Steps:

Open the pig log properties by using the command

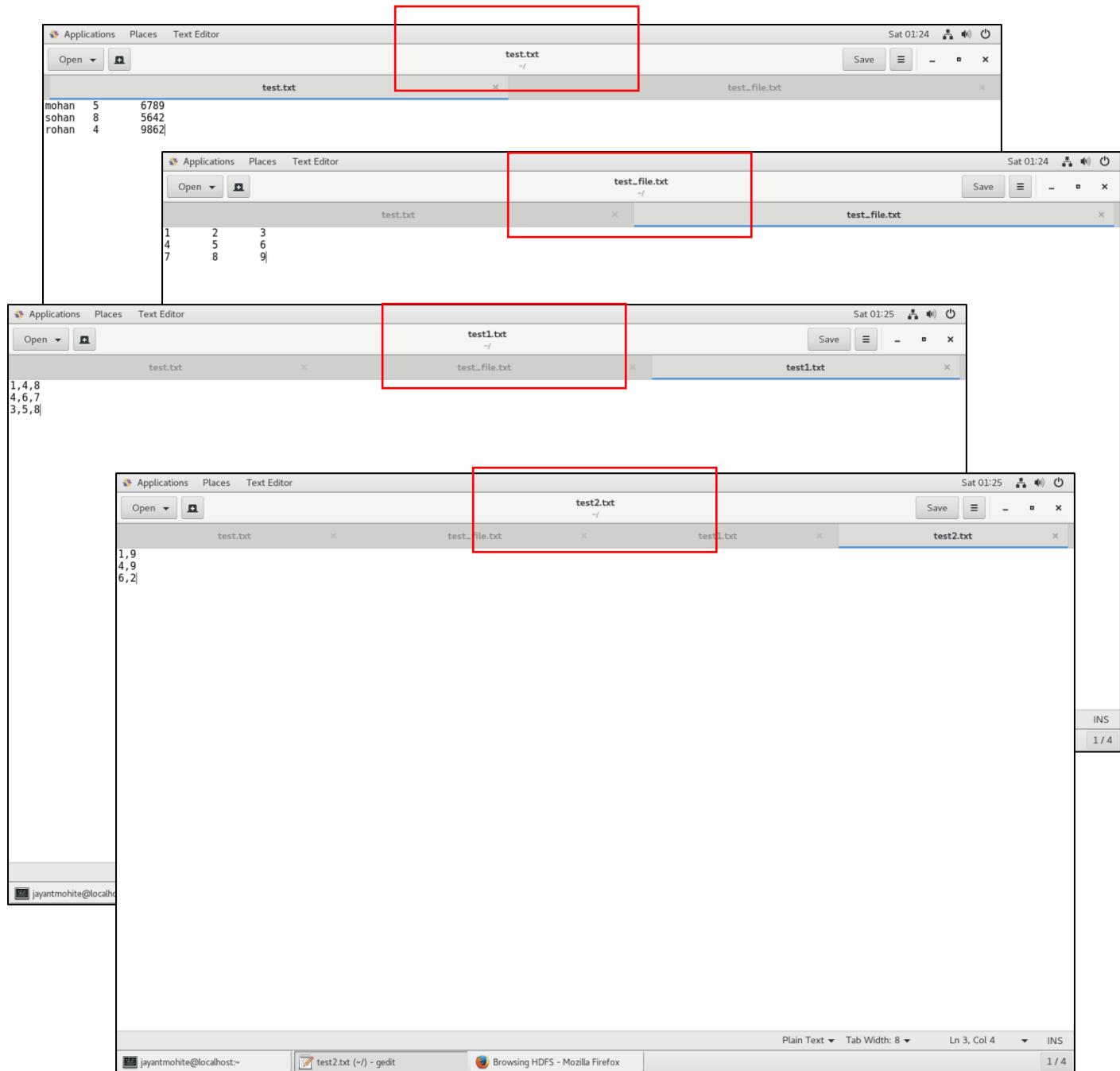
`sudo gedit /etc/pig/conf/log4j.properties`

and the to it a line which states

`log4j.logger.org.apache.pig=fatal, A`

For this lab session we will be using Pig in local mode.

Sample files used:



Step Description:

In this step we will load the input file test.txt located at /home/jayantmohite/ with a schema of 3 columns namely name, emp_num and salary.

In this case we are not specifying any delimiter as the delimiter used in the file test.txt is tab and in Pig tab (\t) is default delimiter.

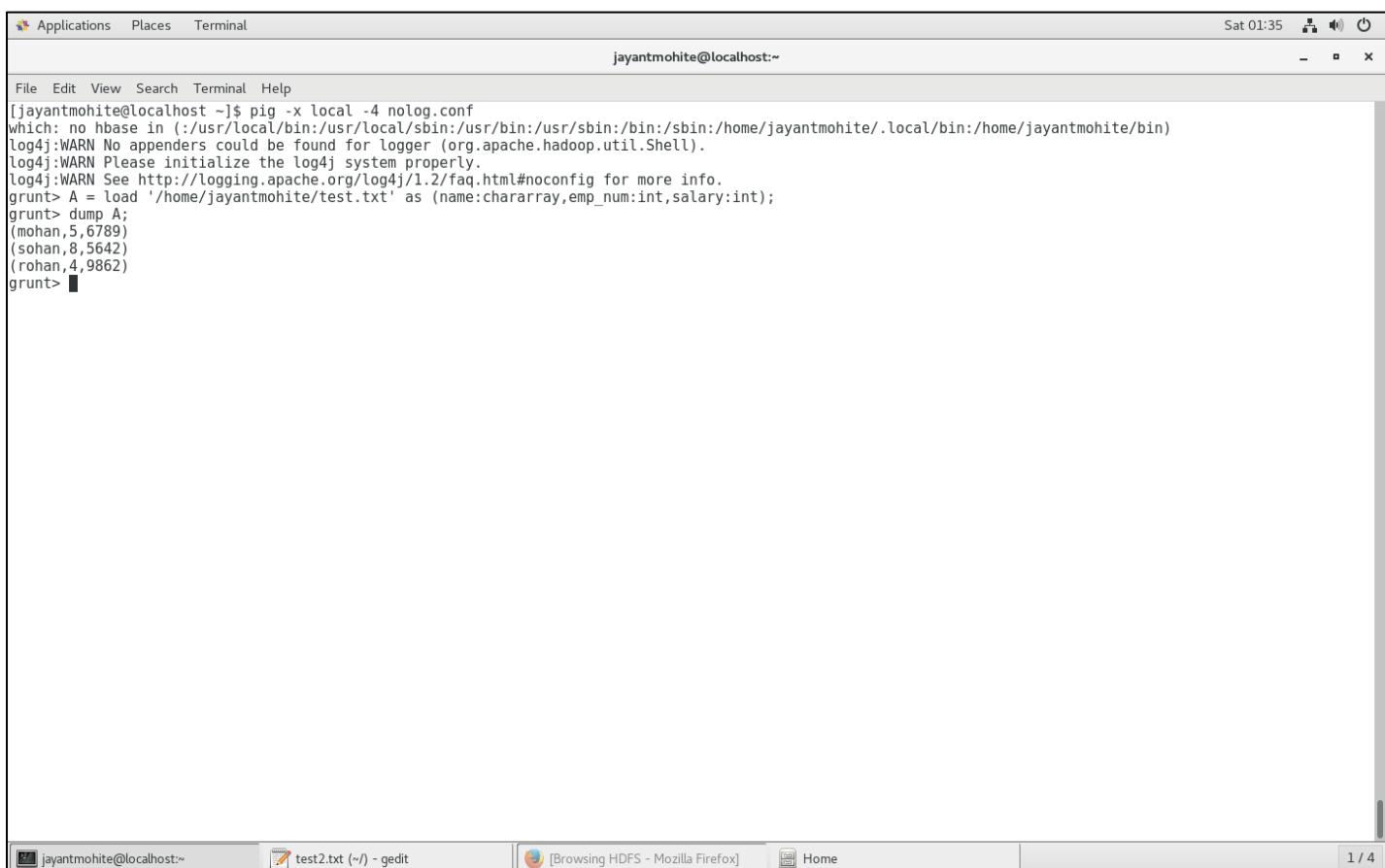
Commands:

```
grunt> A = load '/home/jayantmohite/test.txt' as (name:chararray, emp_num:int, salary:int);
```

(In this command, A is the bag that we are creating)

```
grunt> dump A;
```

Step Visualization:



The screenshot shows a terminal window with the following session:

```
[jayantmohite@localhost ~]$ pig -x local -4 nolog.conf
which: no hbase in (:/usr/local/bin:/usr/local/sbin:/usr/bin:/sbin:/home/jayantmohite/.local/bin:/home/jayantmohite/bin)
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
grunt> A = load '/home/jayantmohite/test.txt' as (name:chararray,emp_num:int,salary:int);
grunt> dump A;
(mohan,5,6789)
(sohan,8,5642)
(rohan,4,9862)
grunt> ■
```

The terminal window has a title bar "jayantmohite@localhost:~". The bottom status bar shows the current directory as "jayantmohite@localhost:~" and other open applications like "test2.txt (~) - gedit" and "[Browsing HDFS - Mozilla Firefox]".

Do note that all labs in this chapter are performed in the Pig Local Mode.

Step Description

In this step we will create a bag B couple of time with the same input file name test_file.txt. The difference in both these execution will be that first execution, we will not specify any schema but in second execution we will specify the schema.

In case where we don't specify the schema, we can reference columns by their positional parameters like column 1 is referenced as \$0; column 2 is referenced as \$1 and so on.

Commands:

grunt> B = load '/home/jayantmohite/test_file.txt' using PigStorage('\t');

(In this command, though not required but we have specified the delimiter used in the file by using the function PigStorage. We are not specifying any schema in this command.)

grunt> B = load '/home/jayantmohite/test_file.txt' using PigStorage('\t') as (f1:int, f2:int, f3:int);

(In this command, we are specifying the schema for the bag which can be checked by the Describe command.)

Step Visualization:

A screenshot of a Linux desktop environment. At the top, there's a menu bar with 'Applications', 'Places', and 'Terminal'. The terminal window title is 'jayantmohite@localhost:~'. The terminal content shows the following Pig Latin script and its execution:

```
File Edit View Search Terminal Help
jayantmohite@localhost:~
grunt> B = load '/home/jayantmohite/test file.txt' using PigStorage('\t');
grunt> dump B;
(1,2,3)
(4,5,6)
(7,8,9)
grunt> B = load '/home/jayantmohite/test file.txt' using PigStorage('\t') as (f1:int, f2:int, f3:int);
grunt> dump B;
(1,2,3)
(4,5,6)
(7,8,9)
grunt> describe B;
B: {f1: int,f2: int,f3: int}
grunt> ■
```

The terminal window has a red box highlighting the first two lines of the command and its output. Below the terminal, the desktop taskbar shows other open applications: 'jayantmohite@localhost:~' (terminal), '[test2.txt (~) - gedit]' (text editor), and '[Browsing HDFS - Mozilla Firefox]' (web browser). A status bar at the bottom right indicates '1 / 4'.

Step Description:

In this step we will have a look at some operators in Pig like filter, foreach and generate.

Commands:

```
grunt> B = load '/home/jayantmohite/test_file.txt' using PigStorage('\t') as (f1:int, f2:int, f3:int);
```

```
grunt> X = filter B by f3 == 3;
```

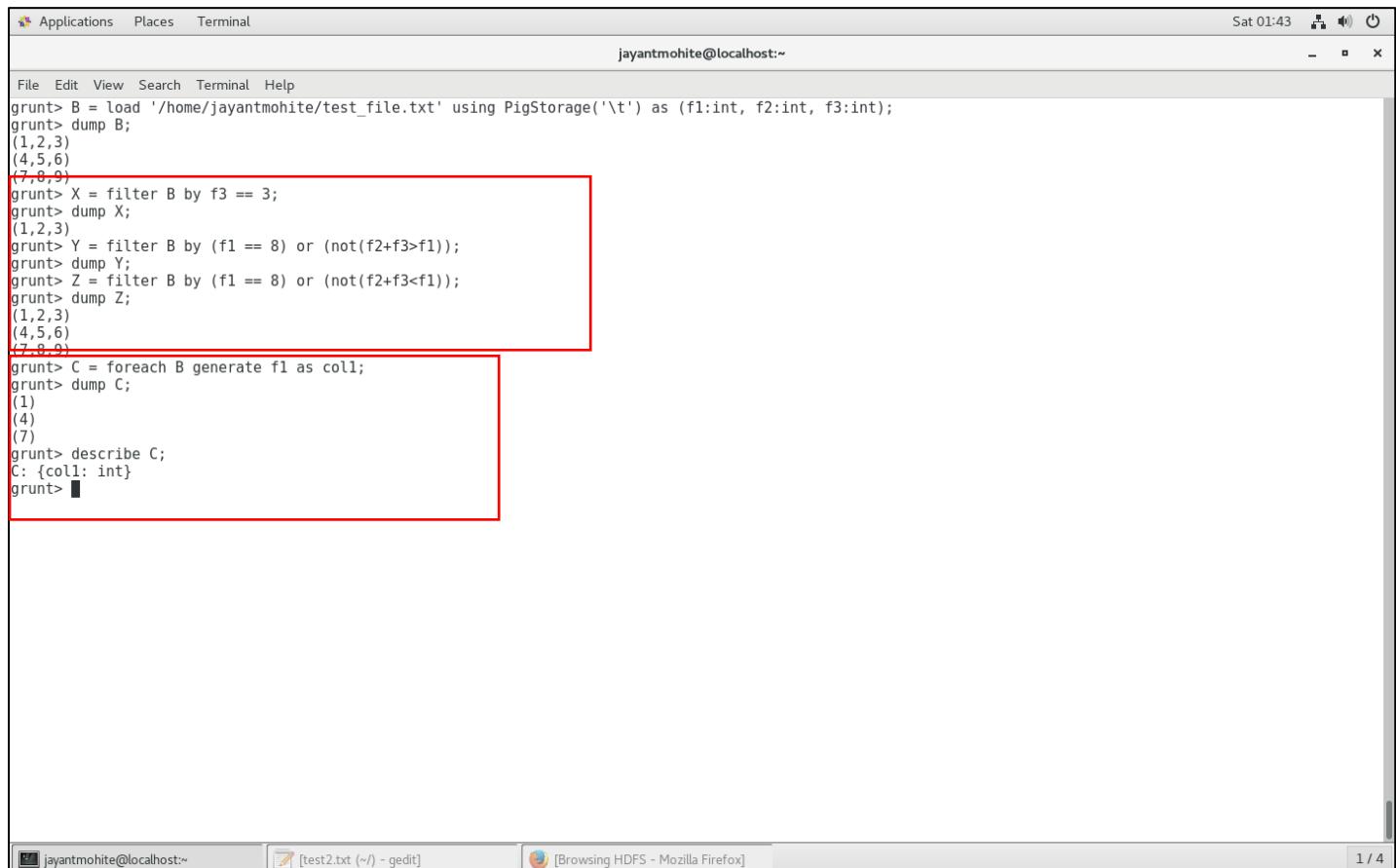
(In this command we are creating a bag by name X which will contains records from the bag B where the column f3 has a value equal to 3)

```
grunt> Z = filter B by (f1 == 8) or (not(f2 + f3 < f1));
```

```
grunt> C = foreach B generate f1 as col1;
```

(In this command we are creating a bag by name C, which will have data of all records only from column f1 of bag B. This new bag C will have a schema where the only column it has will have a name as col1)

Step Visualization:



```
Applications Places Terminal Sat 01:43 jayantmohite@localhost:~  
File Edit View Search Terminal Help  
grunt> B = load '/home/jayantmohite/test_file.txt' using PigStorage('\t') as (f1:int, f2:int, f3:int);  
grunt> dump B;  
(1,2,3)  
(4,5,6)  
(7,8,9)  
grunt> X = filter B by f3 == 3;  
grunt> dump X;  
(1,2,3)  
grunt> Y = filter B by (f1 == 8) or (not(f2+f3>f1));  
grunt> dump Y;  
grunt> Z = filter B by (f1 == 8) or (not(f2+f3<f1));  
grunt> dump Z;  
(1,2,3)  
(4,5,6)  
(7,8,9)  
grunt> C = foreach B generate f1 as coll;  
grunt> dump C;  
(1)  
(4)  
(7)  
grunt> describe C;  
C: {coll: int}  
grunt>
```

Step Description:

In this step we will see how Group By works in Pig

Commands:

```
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int, name:chararray, salary:int, country:chararray);
```

(for this you can use the same input file that we have used in Hive Partitions)

```
grunt> B = group A by country;
```

```
grunt> describe B;
```

(this command will show the schema of the bag B which contains records from bag A grouped by country. In this bag, every tuple will have 2 atoms. The 1st atom will be the key group and the 2nd atom will be the elements of bag A that belong to this group.)

Step Visualization:

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "jayantmohite@localhost:~". The terminal content displays the following Pig Latin script and its execution results:

```
File Edit View Search Terminal Help
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int,name:chararray,salary:int,country:chararray);
grunt> dump A;
(1,abc,100,india)
(2,bcd,200,us)
(3,cde,300,india)
(4,def,400,india)
(5,efg,500,us)
(6,fg,h,700,aus)
grunt> B = group A by country;
grunt> describe B;
B: {group: chararray,A: {(id: int,name: chararray,salary: int,country: chararray)}}
(grunt> dump B;
(us,{(2,bcd,200,us),(5,efg,500,us)})
(aus,{(6,fg,h,700,aus)})
(india,{(1,abc,100,india),(3,cde,300,india),(4,def,400,india)})}
```

The terminal window is part of a desktop interface with other application icons visible in the dock at the bottom.

Step Description:

In this step we will have a demonstration of the join and union commands of pig wherein we will be working with two bags in the same command.

Command:

```
grunt> A = load '/home/jayantmohite/test1.txt' using PigStorage(',') as (a1:int, a2:int, a3:int);
```

```
grunt> B = load '/home/jayantmohite/test2.txt' using PigStorage(',') as (b1:int, b2:int);
```

```
grunt> X = join A by a1, B by b1;
```

(the data that we are loading in bag X can be visualized by the SQL query as
select * from A,B where A.a1 = B.b1)

```
grunt> Y = union A, B;
```

Step Visualization:

```
Applications Places Terminal Sat 01:49 jayantmohite@localhost:~  
File Edit View Search Terminal Help  
grunt> A = load '/home/jayantmohite/test1.txt' using PigStorage(',') as (a1:int, a2:int, a3:int);  
grunt> dump A;  
(1,4,8)  
(4,6,7)  
(3,5,8)  
grunt> B = load '/home/jayantmohite/test2.txt' using PigStorage(',') as (b1:int, b2:int);  
grunt> dump B;  
(1,9)  
(4,9)  
(6,2)  
grunt> X = join A by a1, B by b1;  
grunt> dump X;  
(1,4,8,1,9)  
(4,6,7,4,9)  
grunt> Y = union A,B;  
grunt> dump Y;  
(1,4,8)  
(4,6,7)  
(3,5,8)  
(1,9)  
(4,9)  
(6,2)  
grunt>
```

jayantmohite@localhost:~ myemp.txt (~) - gedit [Browsing HDFS - Mozilla Firefox] Home 1 / 4

Step Description:

In this step we will have a demonstration of the Sample command. Lets say we have an input data with 10 million records and we need to run some pig analytics on to it. And we are not sure if our program will yield the correct result. In order to verify this, it would be great if we could test our program on a subset of the actual data.

This is exactly what sample command does.

Command:

```
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int, name:chararray, salary:int, country: chararray);
```

```
grunt> X = sample A 0.5;
```

(in here 0.5 is like 50% of the total input data)

Step Visualization:

A screenshot of a Linux desktop environment. At the top, there is a horizontal menu bar with 'Applications', 'Places', and 'Terminal'. The system tray shows the date 'Sat 02:10' and icons for battery, signal strength, and power. Below the menu bar is a terminal window titled 'jayantmohite@localhost:~'. The terminal displays a session of the Apache Pig command-line interface (grun). The user loads a file 'myemp.txt' into a relation 'A' using the command 'load ... using PigStorage(',') as (id:int, name:chararray, salary:int, country:chararray);'. Then, they dump the contents of 'A' to the screen, showing six tuples: (1,abc,100,india), (2,bcd,200,us), (3,cde,300,india), (4,def,400,india), (5,efg,500,us), and (6,fgh,700,aus). Next, they sample 'A' with a probability of 0.5, resulting in a dump of three tuples: (2,bcd,200,us), (5,efg,500,us), and (6,fgh,700,aus). The terminal prompt 'grunt>' appears at the end. At the bottom of the screen, there is a dock with several icons: a terminal icon labeled 'jayantmohite@localhost:~', a gedit icon labeled 'myemp.txt (~/) - gedit', a Firefox icon labeled '[Browsing HDFS - Mozilla Firefox]', and a 'Home' icon. On the far right of the dock, it says '1 / 4'.

```
File Edit View Search Terminal Help
jayantmohite@localhost:~
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int, name:chararray, salary:int, country:chararray);
grunt> dump A;
(1,abc,100,india)
(2,bcd,200,us)
(3,cde,300,india)
(4,def,400,india)
(5,efg,500,us)
(6,fgh,700,aus)
grunt> X = sample A 0.5;
grunt> dump X;
(2,bcd,200,us)
(5,efg,500,us)
(6,fgh,700,aus)
grunt>
```

Step Description:

In this step we will have a look at the split command which allows you to create multiple bags from an existing bag in the same command based on some criteria

Commands:

```
grunt> A = load '/home/jayantmohite/test1.txt' using PigStorage(',') as (f1:int, f2:int, f3:int);
```

```
grunt> split A into x if f1 < 7, y if f2 == 5, z if (f3 < 6 or f3 > 6);
```

(this command will create the following 3 bags from data in bag A

bag x = all records from A that have the value of 1st column less than 7

bag y = all records from A that have value of 2nd column equal to 5

bag z = all records from A that have value of 3rd column not equal to 6)

Step Visualization:

A screenshot of a Linux desktop environment. At the top, there's a horizontal menu bar with 'Applications', 'Places', and 'Terminal'. The system tray shows the date 'Sat 02:13' and icons for signal strength, battery, and power. Below the menu is a terminal window titled 'jayantmohite@localhost:~'. The terminal contains the following Pig Latin script:

```
File Edit View Search Terminal Help
grunt> A = load '/home/jayantmohite/test1.txt' using PigStorage(',') as (f1:int, f2:int, f3:int);
grunt> dump A;
(1,4,8)
(4,6,7)
(3,5,8)
grunt> split A into x if f1<7, y if f2==5, z if(f3<6 or f3>6);
grunt> dump x;
(1,4,8)
(4,6,7)
(3,5,8)
grunt> dump y;
(3,5,8)
grunt> dump z;
(1,4,8)
(4,6,7)
(3,5,8)
grunt> █
```

The terminal window has a dark background and light-colored text. Below the terminal is a docked application bar with several icons: a terminal icon labeled 'jayantmohite@localhost:~', a file icon labeled 'myemp.txt (~) - gedit', a browser icon labeled '[Browsing HDFS - Mozilla Firefox]', and a home icon labeled 'Home'. On the far right of the bar, it says '1 / 4'.

Step Description:

In this step we will demonstrate some aggregate functions like Min, Max and Count

Commands:

```
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int, name:chararray, salary:int, country:chararray);
```

```
grunt> B = group A by country;
```

```
grunt> C = foreach B generate A.country, AVG(A.salary);
```

```
grunt> D = foreach B generate group, MAX(A.salary);
```

```
grunt> E = foreach B generate group, MIN(A.salary);
```

```
grunt> F = foreach B generate COUNT(A);
```

Step Visualization:

The screenshot shows a Linux desktop interface. At the top, there's a panel with icons for Applications, Places, Terminal, and user information (jayantmohite@localhost). The main area is a terminal window titled 'jayantmohite@localhost:~'. It contains the following Pig Latin script:

```
File Edit View Search Terminal Help
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int,name:chararray,salary:int,country:chararray);
grunt> dump A;
(1,abc,100,india)
(2,bcd,200,us)
(3,cde,300,india)
(4,def,400,india)
(5,efg,500,us)
(6,fgh,700,aus)
grunt> B = group A by country;
grunt> dump B;
(us,{(2,bcd,200,us),(5,efg,500,us)})
(aus,{(6,fgh,700,aus)})
(india,{(1,abc,100,india),(3,cde,300,india),(4,def,400,india)})
grunt> C = foreach B generate A.country, AVG(A.salary)
>> ;
grunt> dump C;
({(us),(us)},350.0)
({(aus)},700.0)
({(india),(india),(india)},266.666666666667)
grunt> D = foreach B generate group, MAX(A.salary);
grunt> dump D;
(us,500)
(aus,700)
(india,400)
grunt> E = foreach B generate group, MIN(A.salary);
grunt> dump E;
(us,200)
(aus,700)
(india,100)
grunt> F = foreach B generate COUNT(A);
grunt> dump F;
(2)
(1)
(3)
grunt>
```

Below the terminal, there's a dock with several icons: a terminal icon labeled 'jayantmohite@localhost:~', a file icon labeled 'myemp.txt (~) - gedit', a browser icon labeled '[Browsing HDFS - Mozilla Firefox]', and a home icon labeled 'Home'. On the far right of the dock, it says '1 / 4'.

In this way by using Pig you can easily transform your data using conditions and loops.

12. Apache Flume

One of the most important component of the Web Architecture is the [Web Server](#). Not only because it provides service for our Web Site but also because we can fetch a lot of useful information out of it. Information like number of online users, pages most visited, pages least visited, time slot of more user traffic, number of error, etc. This can be further used to do a lot of analysis for deriving business information.

For example, if an e-commerce company has introduced a new module on their Web Site and want to track the popularity and efficiency of that module. In this case they are required to constantly monitor the user activity on their Website. So how can they achieve this?

The answer is simple. What they need to do is analyse their Web Server [logs](#) and from that they would be able to track the user activity as well as trace any error generated. But now the challenge is that, the logs on the server is an ever incrementing file and if we talk about companies like Amazon, then their Web Servers generate around PB of data almost every hour. So how to analyse such large amount of data.

As a solution to this problem we have a Hadoop Ecosystem tool that acts like a [messenger](#) and will keep a track of all updates happening in the logs file and will reliably dump those into the HDFS. Then using tools like [HIVE](#) and [PIG](#) you can easily analyse it and perform your desired calculations.

This tool is called as [FLUME](#) and this is what we would be talking around in this chapter.

Introduction

So what is Flume exactly all about? Flume is a simple messaging service that can monitor any ever incrementing file and dump the updates in the HDFS. Flume is generally a separate server altogether and is neither installed on Client nor on the Cluster. One instance of the Flume server is called as a Flume Agent. One Flume agent can be used to monitor a single ever incrementing file. All you need to do is, tell the [Flume Agent](#) the location of the Source file which it has to monitor, the destination directory in HDFS where it has to dump the updates and the communication channel that is to be used for this operation. Once you configure and start the Flume agent, this continuous service will keep on monitoring the changes in your source file and will keep on dumping it in the HDFS. Optionally you can also configure the time interval after which the updates are to be dumped. BY default it will dump the updates as and when they happen.

The screenshot shows a terminal window titled "wisdom : bash". The command entered is:

```
[wisdom@localhost ~]$ flume-ng agent --conf-file /usr/lib/flume-ng/conf/flume.conf --name agent -Dflume.root.logger=INFO,console
```

Two parts of the command are highlighted with red boxes and arrows pointing down to explanatory text:

- The first part, `--conf-file /usr/lib/flume-ng/conf/flume.conf`, is highlighted and points to the text: "Configuration file where the source, destination and channel are specified".
- The second part, `--name agent`, is highlighted and points to the text: "Defining the level of logging".

Working of Flume Agent

So now let's assume that we have a Web Server log file that is to be analysed. For this reason what we will do is configure the Flume agent and give it a destination directory inside the HDFS to dump the updates on the log file.

```
wisdom : sudo
File Edit View Scrollback Bookmarks Settings Help
# under the License.

# The configuration file needs to define the sources,
# the channels and the sinks.
# Sources, channels and sinks are defined per agent,
# in this case called 'agent' → Configuring the
# Flume Agent
agent.sources = seqGenSrc
agent.channels = memoryChannel
agent.sinks = loggerSink

# For each one of the sources, the type is defined
agent.sources.seqGenSrc.type = exec
agent.sources.seqGenSrc.command = tail -F /home/wisdom/Documents/log.txt
[redacted]
-- INSERT --
[redacted]
wisdom : sudo
```

Once the Flume agent is started it will start monitoring the log file. Whenever any updates take place it will create something called as events out of it. We can visualize events as a temporary file. The Agent will keep on doing this for a specified interval of time, after which it will bundle all the events together and dump it in the HDFS. By the time this happens there are still updates happening in the log file which are still monitored by the Agent.

```
wisdom : sudo
File Edit View Scrollback Bookmarks Settings Help
agent.channels = memoryChannel
agent.sinks = loggerSink

# For each one of the sources, the type is defined
agent.sources.seqGenSrc.type = exec
agent.sources.seqGenSrc.command = tail -F /home/wisdom/Documents/log.txt

# The channel can be defined as follows.
agent.sources.seqGenSrc.channels = memoryChannel → Defining the
# communication channel
agent.sinks.loggerSink.type = logger

#Specify the channel the sink should use
agent.sinks.loggerSink.channel = memoryChannel
[redacted]
-- INSERT --
[redacted]
wisdom : sudo
```

```
wisdom : sudo
File Edit View Scrollback Bookmarks Settings Help
agent.sinks.loggerSink.type = logger
#Specify the channel the sink should use
agent.sinks.loggerSink.channel = memoryChannel
# Each channel's type is defined.
agent.channels.memoryChannel.type = memory
# Other config values specific to each type of channel(sink or source)
# can be defined as well
# In this case, it specifies the capacity of the memory channel
agent.channels.memoryChannel.capacity = 100
agent.sinks.loggerSink.type = hdfs
agent.sinks.loggerSink.hdfs.path = hdfs://localhost/user/wisdom/logs
agent.sinks.loggerSink.hdfs.fileType = DataStream
-- INSERT --
wisdom : sudo
```

Defining the HDFS Destination

```
wisdom : flume-ng
File Edit View Scrollback Bookmarks Settings Help
15/10/02 11:57:40 INFO instrumentation.MonitoringCounterGroup: Component type: SOURCE, name: seqGenSrc started
15/10/02 11:57:44 INFO hdfs.HDFSDataStream: Serializer = TEXT, UseRawLocalFileSystem = false
15/10/02 11:57:45 INFO hdfs.BucketWriter: Creating hdfs://localhost/user/wisdom/Log_dir/FlumeData.1443801464990.tmp → Event Creation
15/10/02 11:57:46 INFO hdfs.BucketWriter: Renaming hdfs://localhost/user/wisdom/Log_dir/FlumeData.1443801464990.tmp to hdfs://localhost/user/wisdom/log_dir/FlumeData.1443801464990
15/10/02 11:57:46 INFO hdfs.BucketWriter: Creating hdfs://localhost/user/wisdom/log_dir/FlumeData.1443801464991.tmp
Dumping file in HDFS
wisdom : flume-ng
```

Contents of directory [/user/wisdom/log_dir](#)

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
FlumeData.1443801464990	file	90 B	1	64 MB	2015-10-02 11:57	rw-r--r--	wisdom	supergroup
FlumeData.1443801464991	file	51 B	1	64 MB	2015-10-02 11:58	rw-r--r--	wisdom	supergroup

[Go back to DFS home](#)

This process is continuous and will carry on till you explicitly terminate the FLUME agent. In this way we can have all the contents of the log file as and when they are generated.

13. Apache Oozie

I work as a Freelancer developer and Trainer. Sometimes I do get simultaneous assignments with same deadlines in both the streams. In such situations it becomes very difficult to manage everything.

One day what happened was very interesting. I had to deliver a full day training to some client in their office premises and on the same day I had to deliver 3 development assignments of 3 different clients. The development assignments included executing around 10 commands that were designed as a pipeline. Means output of one operation was used as the input of the next operation. So it was not possible for me to initiate all assignments and leave for the training and hope that they get completed by the time I come back. I needed someone to stay there and execute these commands one after the other and in case of errors suspend the process. Writing a script was also not a feasible solution.

So what I did was, I told my younger brother that he needs to help me today and told him that I would be giving him 3 pieces of paper with 10 commands each and all his has to do is write those commands in the terminal in a sequential manner and in case of error intimate me in the evening.

Assignment 1

```
<name>command 1.1</name>
<action>action 1.1</action>
<name>command 1.2</name>
<action>action 1.2</action>
<name>command 1.3</name>
<action>action 1.3</action>
<name>command 1.4</name>
<action>action 1.4</action>
```

Assignment 2

```
<name>command 2.1</name>
<action>action 2.1</action>
<name>command 2.2</name>
<action>action 2.2</action>
<name>command 2.3</name>
<action>action 2.3</action>
<name>command 2.4</name>
<action>action 2.4</action>
```

Assignment 3

```
<name>command 3.1</name>
<action>action 3.1</action>
<name>command 3.2</name>
<action>action 3.2</action>
<name>command 3.3</name>
<action>action 3.3</action>
<name>command 3.4</name>
<action>action 3.4</action>
```

Though he was not aware about what actually the actions were or what was the outcome of what he is doing, he still completed all 3 assignments by the time I came back. It was a big help and this solution really worked nice for me.

Was this incident interesting? Now let's relate it to what we have to understand in this chapter.

In the above incident I represent any [Hadoop Client](#) that has multiple interdependent jobs to be executed on the Hadoop Cluster. My younger brother is the Hadoop Ecosystem tool called [OOZIE](#). The pieces of paper are xml documents called as [OOZIE workflows](#) inside which the name represents the [Control Flow Nodes](#) and the action represents the [Action Nodes](#).

Introduction

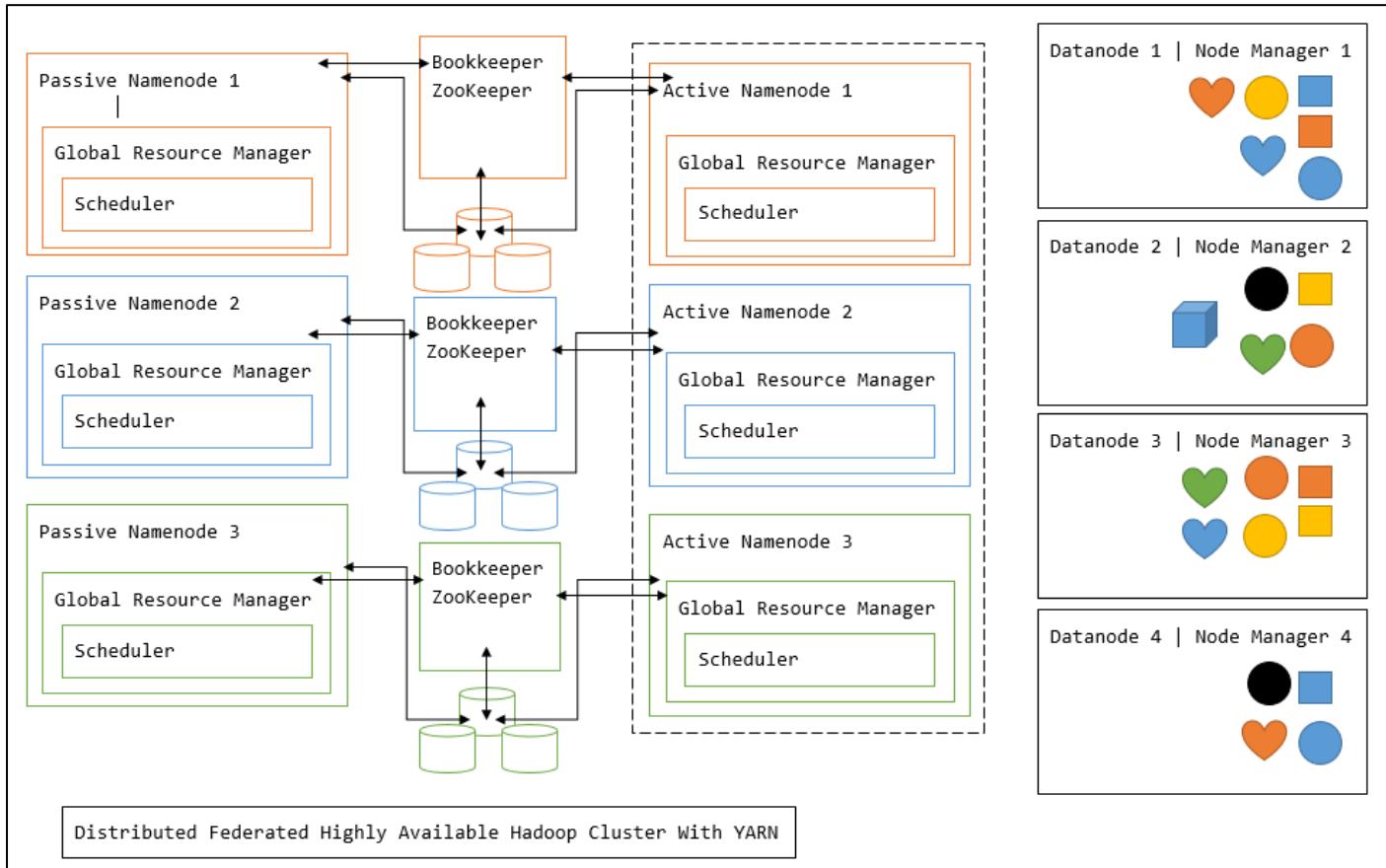
Oozie is a Hadoop Ecosystem tool which works as a Workflow scheduler. What Oozie can do is simply schedule your jobs and execute them in a sequential manner. Oozie is not concerned whether it is a Map reduce job or a Pig job or a Hive job or any other Hadoop Job. It is only concerned with a xml file which called as Oozie Workflow that defines the actions to be performed by the engine in a sequential manner.

If a PIG command is to be executed, the Oozie engine will simply submit that to the PIG engine and will make sure that it gets executed and the output and/or error are given back as the result.

Apart from this, Oozie can also work like a [CRON](#) job. You can decide a schedule for your jobs and Oozie will execute the job whenever the time triggers.

14. Hadoop Generation 2

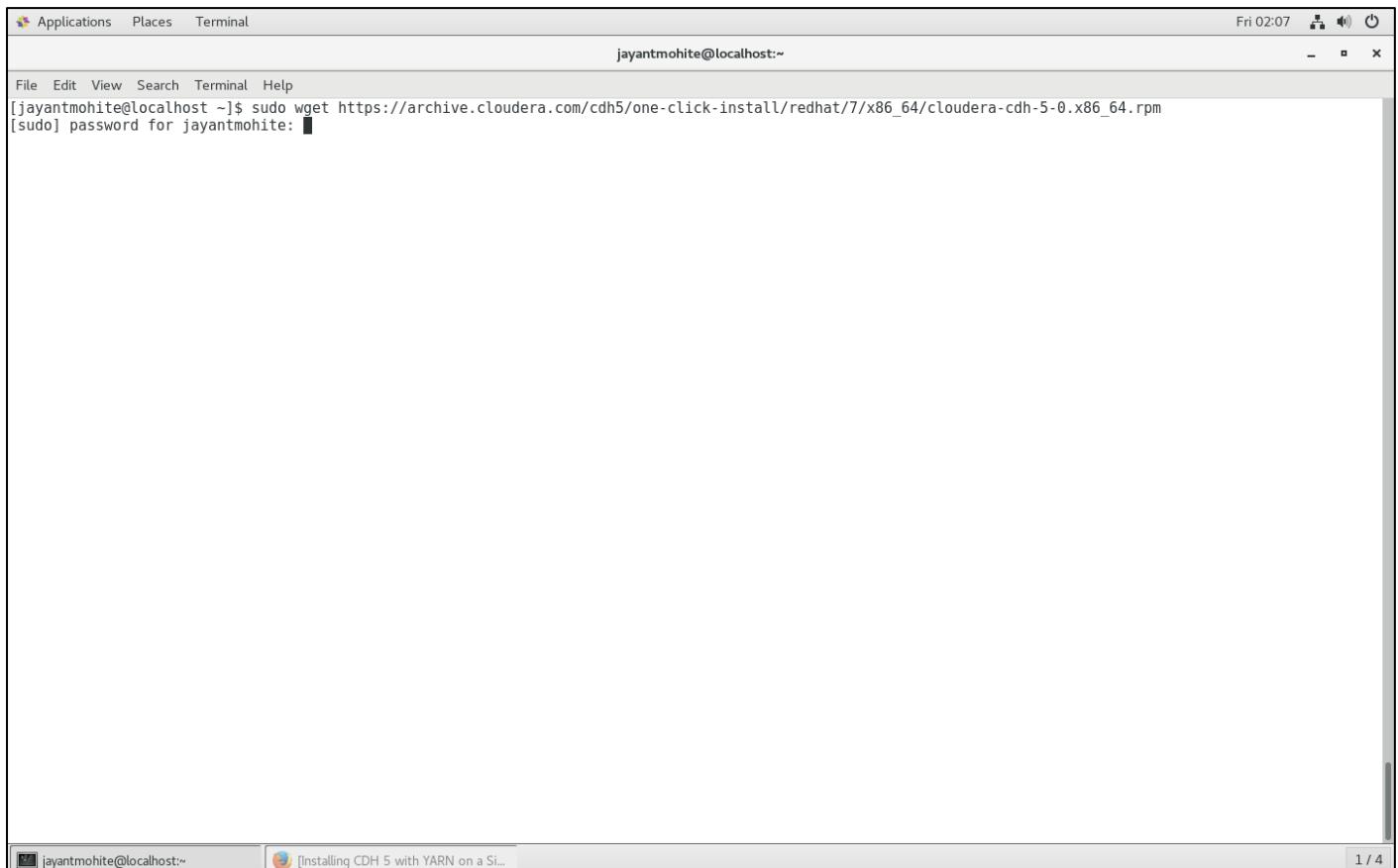
Hadoop Generation 2 Architecture



Hadoop Generation 2 Installation

Step1:

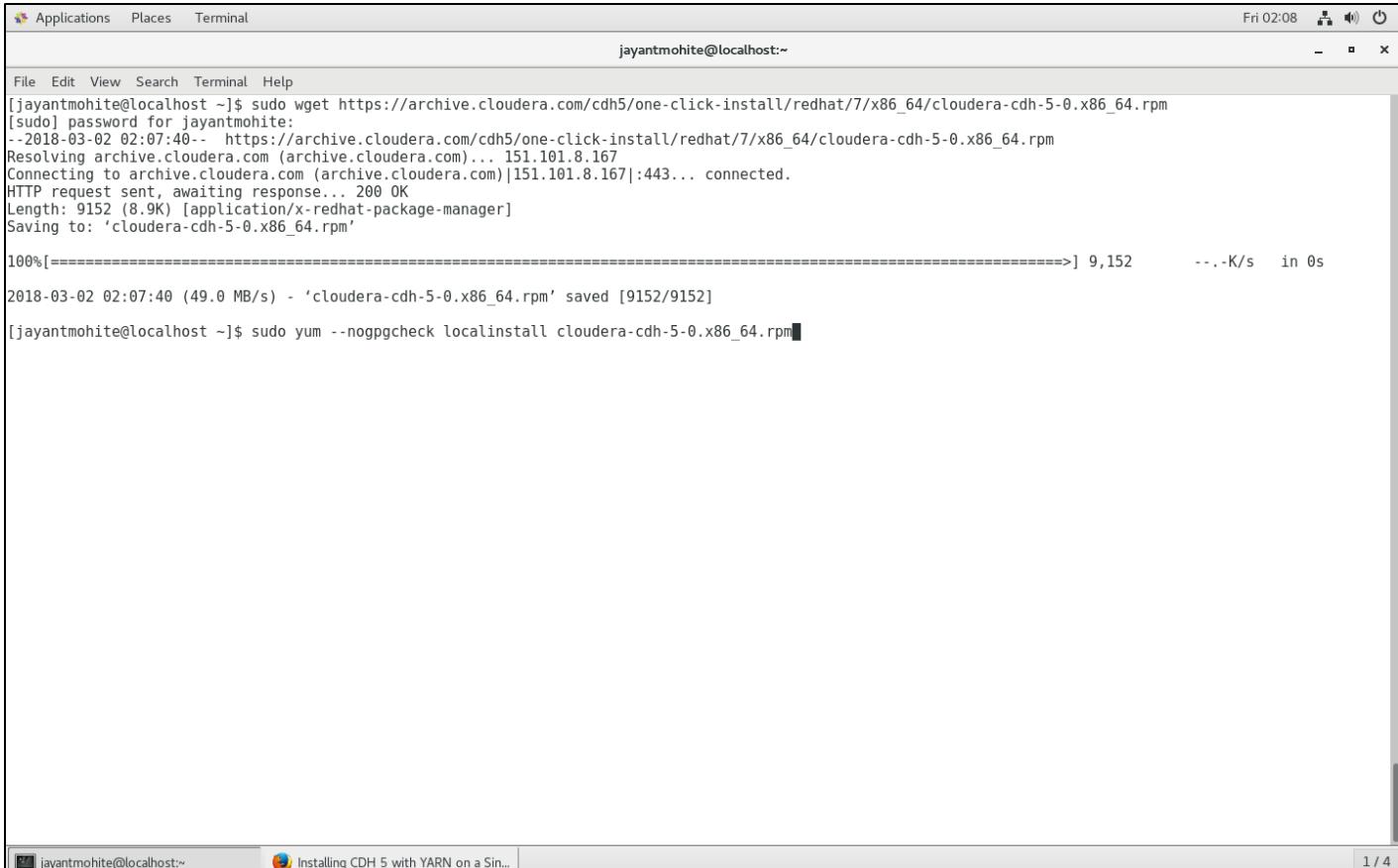
Download the Cloudera Package of Hadoop Version 5



A screenshot of a Linux terminal window titled "Terminal". The window shows the command `sudo wget https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm` being run by the user "jayantmohite". The terminal interface includes a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The status bar at the bottom shows the user's name "jayantmohite@localhost" and the file path "...". A progress bar at the bottom indicates the download is 1 / 4 complete.

Step 2:

Install the Downloaded Package

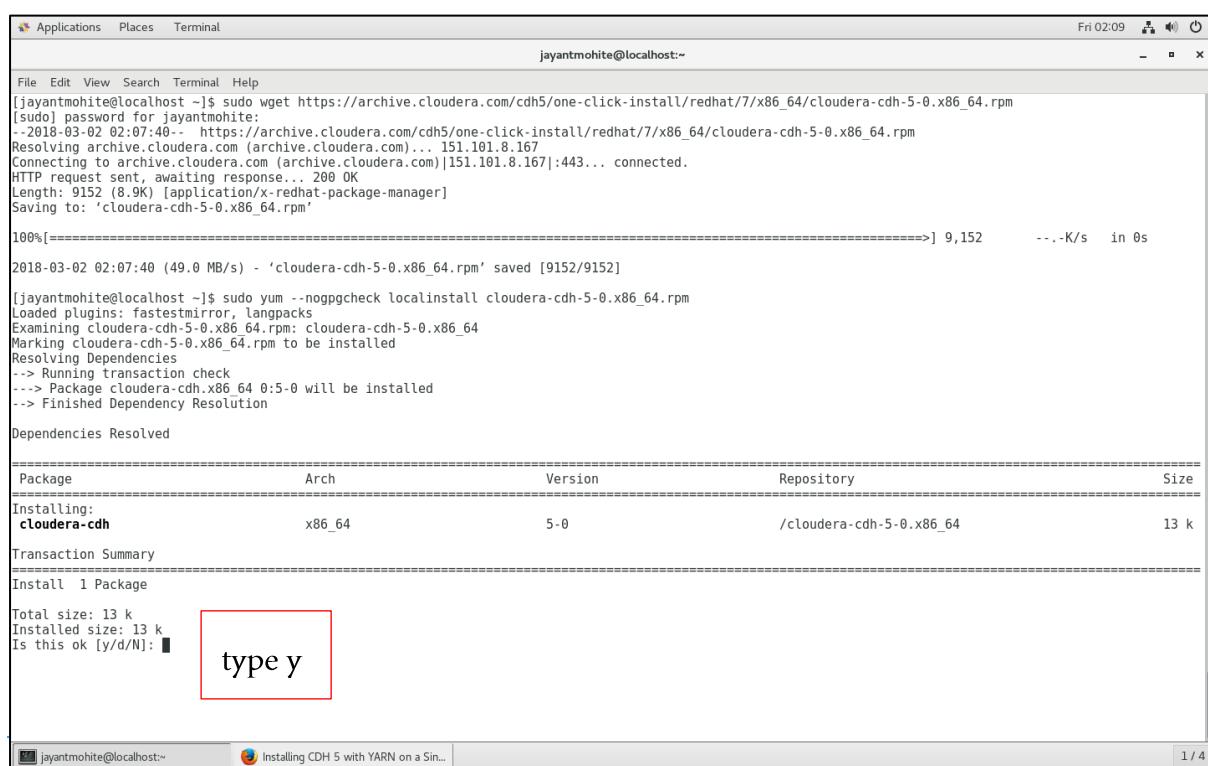


```

Applications Places Terminal Fri 02:08
jayantmohite@localhost:~ File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ sudo wget https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm
[sudo] password for jayantmohite:
--2018-03-02 02:07:40-- https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm
Resolving archive.cloudera.com (archive.cloudera.com)... 151.101.8.167
Connecting to archive.cloudera.com (archive.cloudera.com)|151.101.8.167|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9152 (8.9K) [application/x-redhat-package-manager]
Saving to: 'cloudera-cdh-5-0.x86_64.rpm'

100%[=====] 9,152      --.-K/s   in 0s
2018-03-02 02:07:40 (49.0 MB/s) - 'cloudera-cdh-5-0.x86_64.rpm' saved [9152/9152]
[jayantmohite@localhost ~]$ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm

```



```

Applications Places Terminal Fri 02:09
jayantmohite@localhost:~ File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ sudo wget https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm
[sudo] password for jayantmohite:
--2018-03-02 02:07:40-- https://archive.cloudera.com/cdh5/one-click-install/redhat/7/x86_64/cloudera-cdh-5-0.x86_64.rpm
Resolving archive.cloudera.com (archive.cloudera.com)... 151.101.8.167
Connecting to archive.cloudera.com (archive.cloudera.com)|151.101.8.167|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9152 (8.9K) [application/x-redhat-package-manager]
Saving to: 'cloudera-cdh-5-0.x86_64.rpm'

100%[=====] 9,152      --.-K/s   in 0s
2018-03-02 02:07:40 (49.0 MB/s) - 'cloudera-cdh-5-0.x86_64.rpm' saved [9152/9152]
[jayantmohite@localhost ~]$ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
Loaded plugins: fastestmirror, langpacks
Examining cloudera-cdh-5-0.x86_64.rpm: cloudera-cdh-5-0.x86_64
Marking cloudera-cdh-5-0.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
--> Package cloudera-cdh.x86_64 0:5-0 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch       Version          Repository        Size
=====
Installing:
cloudera-cdh   x86_64    5-0              /cloudera-cdh-5-0.x86_64  13 k

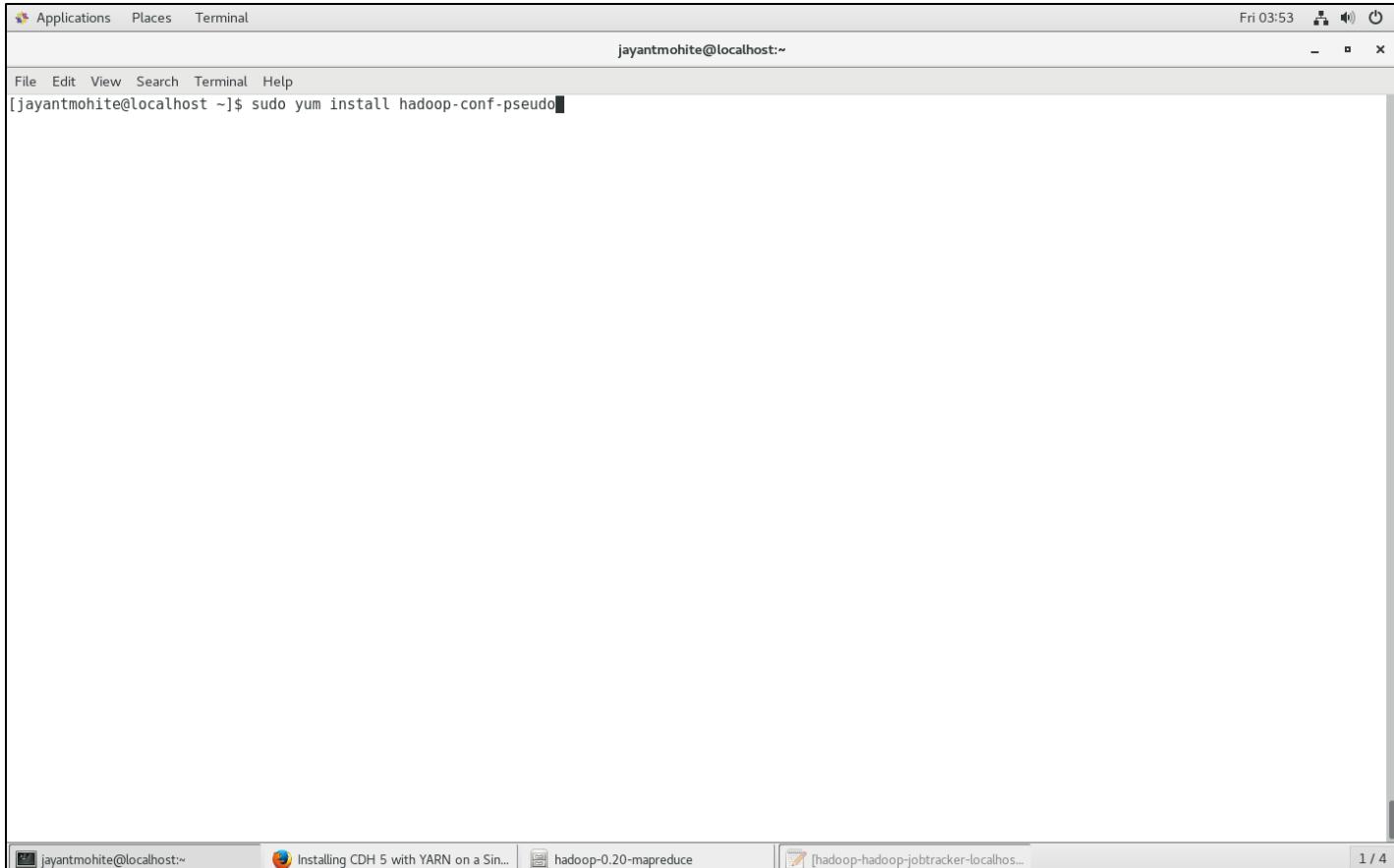
Transaction Summary
=====
Install 1 Package

Total size: 13 k
Installed size: 13 k
Is this ok [y/d/N]: type y

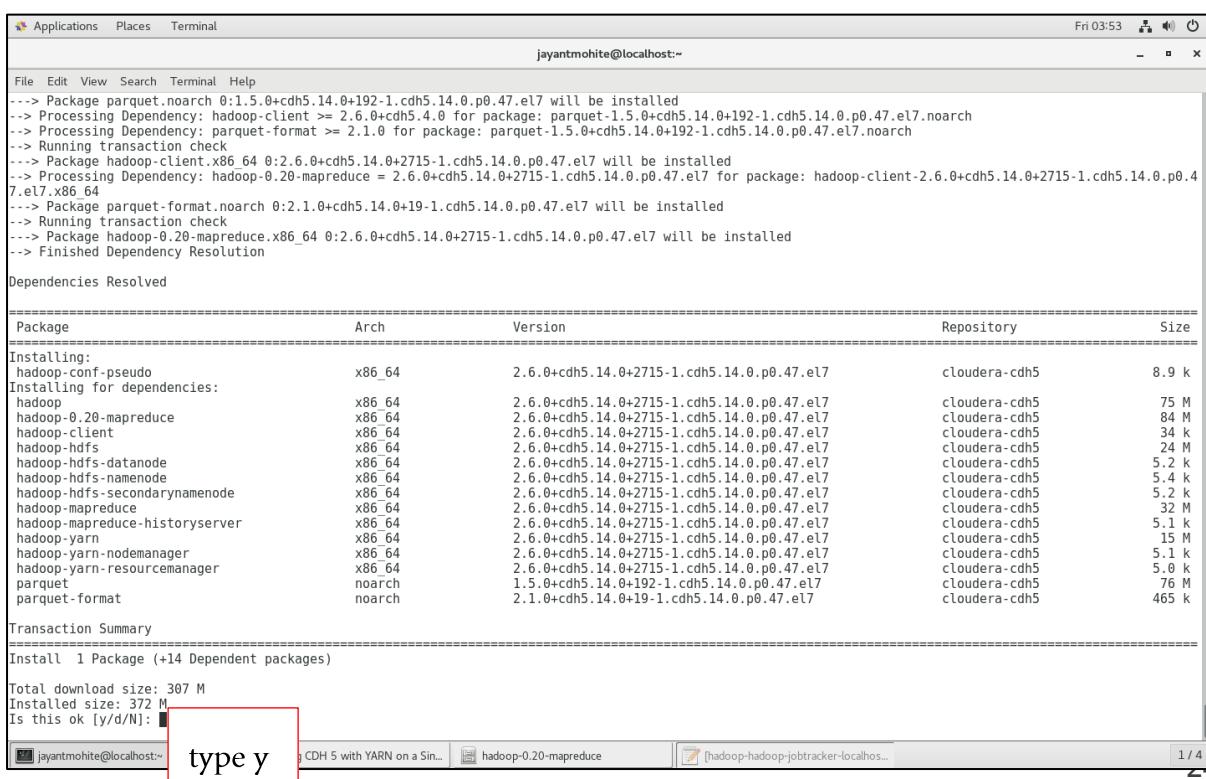
```

Step 3:

Install Hadoop Generation 2 Framework



```
[jayantmohite@localhost ~]$ sudo yum install hadoop-conf-pseudo
```

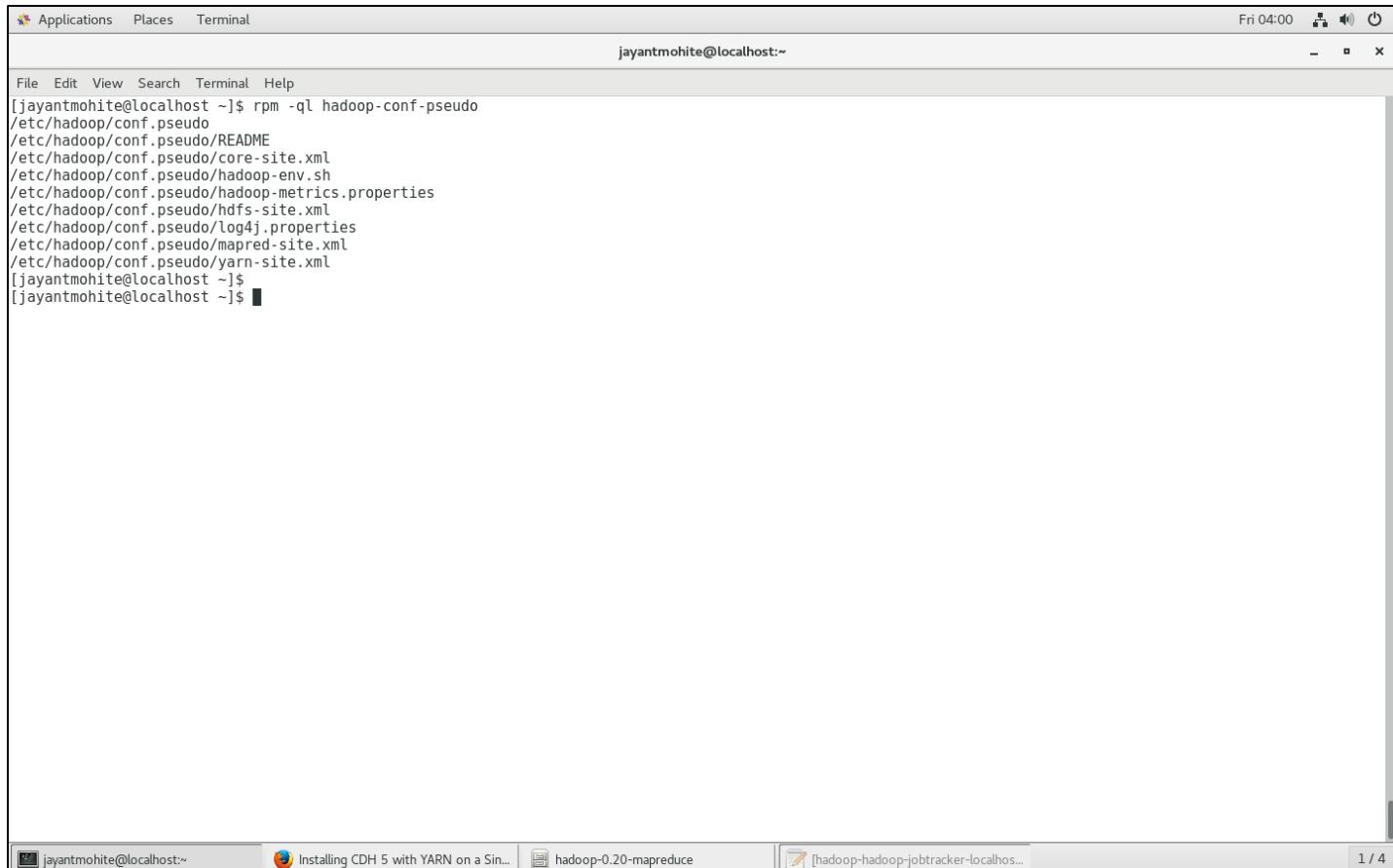


```
jayantmohite@localhost:~$ sudo yum install hadoop-conf-pseudo
...
Dependencies Resolved
=====
Package           Arch      Version            Repository        Size
=====
Installing:
hadoop-conf-pseudo          x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   8.9 k
Installing for dependencies:
hadoop                  x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   75 M
hadoop-0.20-mapreduce       x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   84 M
hadoop-client              x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   34 k
hadoop-hdfs                x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   24 M
hadoop-hdfs-datanode        x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.2 k
hadoop-hdfs-namenode        x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.4 k
hadoop-hdfs-secondarynamenode x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.2 k
hadoop-mapreduce             x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   32 M
hadoop-mapreduce-historyserver x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.1 k
hadoop-yarn                  x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   15 M
hadoop-yarn-nodemanager       x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.1 k
hadoop-yarn-resourcemanager    x86_64   2.6.0+cdh5.14.0+2715-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   5.0 k
parquet                     noarch   1.5.0+cdh5.14.0+192-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   76 M
parquet-format                noarch   2.1.0+cdh5.14.0+19-1.cdh5.14.0.p0.47.el7    cloudera-cdh5   465 k
Transaction Summary
=====
Install 1 Package (+14 Dependent packages)

Total download size: 307 M
Installed size: 372 M
Is this ok [y/d/N]: type y
```

Step 4:

Verify if the installation happened properly



A screenshot of a Linux terminal window titled "Terminal". The window shows the command `rpm -ql hadoop-conf-pseudo` being run, which lists several configuration files under the `/etc/hadoop/conf/pseudo` directory. The terminal window has a standard title bar with "Applications", "Places", "Terminal", and "File Edit View Search Terminal Help". The status bar at the bottom shows the user "jayantmohite" and the command "Installing CDH 5 with YARN on a Sin...". The window is set to "Fri 04:00".

```
[jayantmohite@localhost ~]$ rpm -ql hadoop-conf-pseudo
/etc/hadoop/conf/pseudo
/etc/hadoop/conf/pseudo/README
/etc/hadoop/conf/pseudo/core-site.xml
/etc/hadoop/conf/pseudo/hadoop-env.sh
/etc/hadoop/conf/pseudo/hadoop-metrics.properties
/etc/hadoop/conf/pseudo/hdfs-site.xml
/etc/hadoop/conf/pseudo/log4j.properties
/etc/hadoop/conf/pseudo/mapred-site.xml
/etc/hadoop/conf/pseudo/yarn-site.xml
[jayantmohite@localhost ~]$ [jayantmohite@localhost ~]$
```

Step 5:

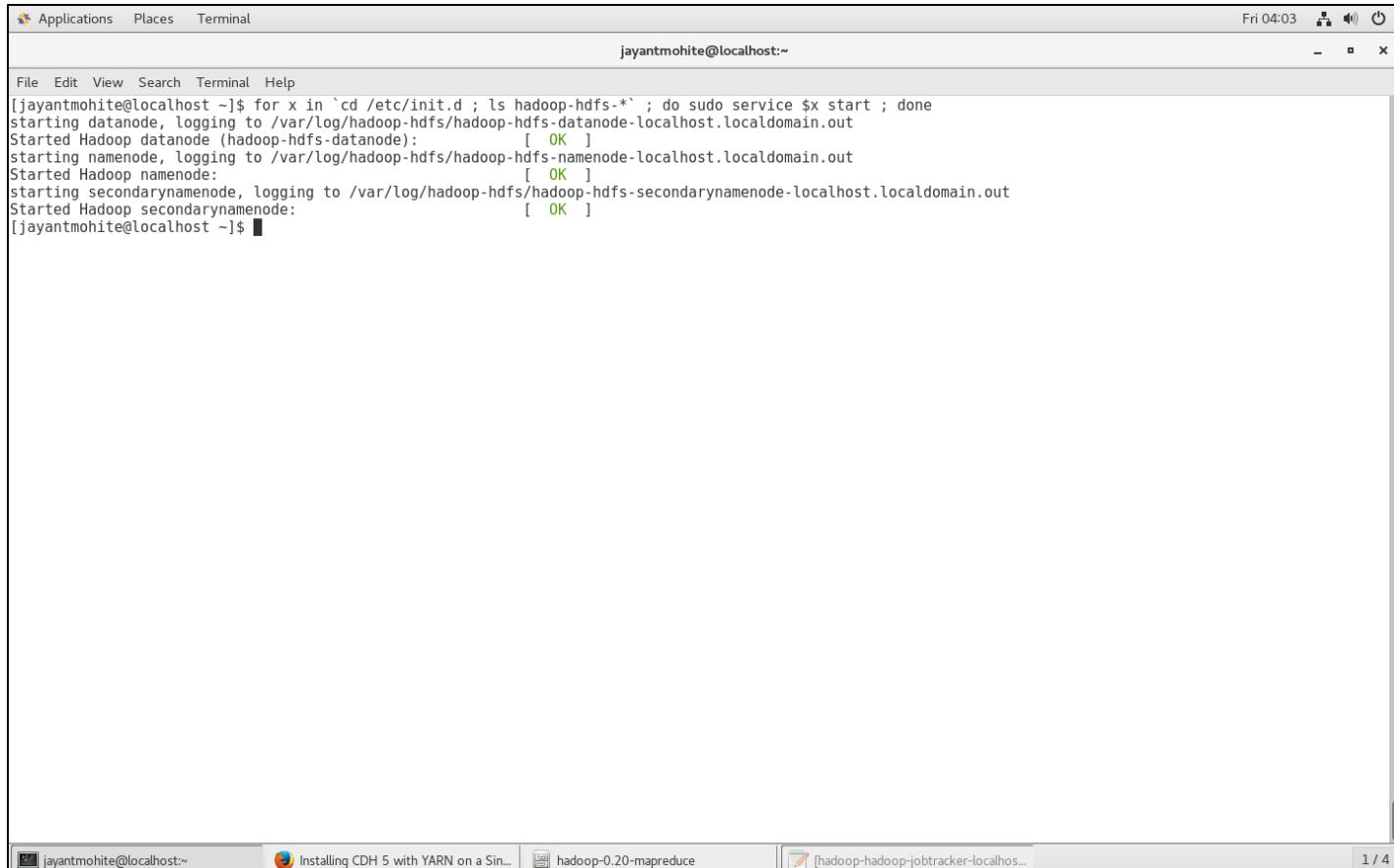
Format Namenode

```
[jayantmohite@localhost ~]$ sudo -u hdfs hdfs namenode -format
18/03/02 02:24:55 INFO namenode.NameNode: STARTUP_MSG:
*****STARTUP_MSG: Starting NameNode
STARTUP_MSG: user = hdfs
STARTUP_MSG: host = localhost/127.0.0.1
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 2.6.0-cdh5.14.0
STARTUP_MSG: classpath = /etc/hadoop/conf:/usr/lib/hadoop/lib/activation-1.1.jar:/usr/lib/hadoop/lib/junit-4.11.jar:/usr/lib/hadoop/lib/apacheds-i18n-2.0.0-M15.jar:/usr/lib/hadoop/lib/jets3t-0.9.0.jar:/usr/lib/hadoop/lib/apacheds-kerberos-codec-2.0.0-M15.jar:/usr/lib/hadoop/lib/log4j-1.2.17.jar:/usr/lib/hadoop/lib/api-asn1-api-1.0.0-M20.jar:/usr/lib/hadoop/lib/api-util-1.0.0-M20.jar:/usr/lib/hadoop/lib/asm-3.2.jar:/usr/lib/hadoop/lib/avro.jar:/usr/lib/hadoop/lib/jettison-1.1.jar:/usr/lib/hadoop/lib/aws-java-sdk-bundle-1.11.134.jar:/usr/lib/hadoop/lib/jetty-6.1.26.cloudera.4.jar:/usr/lib/hadoop/lib/azure-data-lake-store-sdk-2.2.3.jar:/usr/lib/hadoop/lib/slf4j-log4j12.jar:/usr/lib/hadoop/lib/commons-beanutils-1.9.2.jar:/usr/lib/hadoop/lib/jetty-util-6.1.26.cloudera.4.jar:/usr/lib/hadoop/lib/commons-beanutils-core-1.8.0.jar:/usr/lib/hadoop/lib/commons-cli-1.2.jar:/usr/lib/hadoop/lib/snappy-java-1.0.4.1.jar:/usr/lib/hadoop/lib/commons-codec-1.4.jar:/usr/lib/hadoop/lib/jsch-0.1.42.jar:/usr/lib/hadoop/lib/commons-collections-3.2.2.jar:/usr/lib/hadoop/lib/stax-api-1.0.2.jar:/usr/lib/hadoop/lib/commons-compress-1.4.1.jar:/usr/lib/hadoop/lib/jsp-api-2.1.jar:/usr/lib/hadoop/lib/commons-configuration-1.6.jar:/usr/lib/hadoop/lib/xmlenc-0.52.jar:/usr/lib/hadoop/lib/commons-digester-1.8.jar:/usr/lib/hadoop/lib/commons-el-1.0.jar:/usr/lib/hadoop/lib/xz-1.0.jar:/usr/lib/hadoop/lib/commons-httpclient-3.1.jar:/usr/lib/hadoop/lib/commons-io-2.4.jar:/usr/lib/hadoop/lib/commons-lang-2.6.jar:/usr/lib/hadoop/lib/zookeeper.jar:/usr/lib/hadoop/lib/commons-logging-1.1.3.jar:/usr/lib/hadoop/lib/commons-math3-3.1.1.jar:/usr/lib/hadoop/lib/commons-net-3.1.jar:/usr/lib/hadoop/lib/commons-pool2-2.5.4.jar:/usr/lib/hadoop/lib/curator-client-2.7.1.jar:/usr/lib/hadoop/lib/curator-framework-2.7.1.jar:/usr/lib/hadoop/lib/curator-recipes-2.7.1.jar:/usr/lib/hadoop/lib/gson-2.2.4.jar:/usr/lib/hadoop/lib/guava-11.0.2.jar:/usr/lib/hadoop/lib/hamcrest-core-1.3.jar:/usr/lib/hadoop/lib/servlet-api-2.5.jar:/usr/lib/hadoop/lib/htrace-core-4.0.1-incubating.jar:/usr/lib/hadoop/lib/httpclient-4.2.5.jar:/usr/lib/hadoop/lib/httpcore-4.2.5.jar:/usr/lib/hadoop/lib/jackson-jaxrs-1.8.8.jar:/usr/lib/hadoop/lib/jackson-xc-1.8.8.jar:/usr/lib/hadoop/lib/jasper-compiler-5.5.23.jar:/usr/lib/hadoop/lib/jasper-runtime-5.5.23.jar:/usr/lib/hadoop/lib/java-xmlbuilder-0.4.jar:/usr/lib/hadoop/lib/jaxb-api-2.2.2.jar:/usr/lib/hadoop/lib/jaxb-impl-2.2.3-1.jar:/usr/lib/hadoop/lib/jersey-core-1.9.jar:/usr/lib/hadoop/lib/jersey-json-1.9.jar:/usr/lib/hadoop/lib/jersey-server-1.9.jar:/usr/lib/hadoop/lib/jsr305-3.0.0.jar:/usr/lib/hadoop/lib/legredactor-1.0.3.jar:/usr/lib/hadoop/lib/mockito-all-1.8.5.jar:/usr/lib/hadoop/lib/netty-3.10.5.Final.jar:/usr/lib/hadoop/lib/paranamer-2.3.jar:/usr/lib/hadoop/lib/protoBuf-java-2.5.0.jar:/usr/lib/hadoop//parquet-format:javadoc.jar:/usr/lib/hadoop//parquet-format-sources.jar:/usr/lib/hadoop//parquet-format.jar:/usr/lib/hadoop//parquet-avro.jar:/usr/lib/hadoop//parquet-cascading.jar:/usr/lib/hadoop//parquet-column.jar:/usr/lib/hadoop//parquet-common.jar:/usr/lib/hadoop//parquet-encoding.jar:/usr/lib/hadoop//parquet-generator.jar:/usr/lib/hadoop//parquet-hadoop-bundle.jar:/usr/lib/hadoop//parquet-hadoop.jar:/usr/lib/hadoop//parquet-jackson.jar:/usr/lib/hadoop//parquet-pig-bundle.jar:/usr/lib/hadoop//parquet-pig.jar:/usr/lib/hadoop//parquet-protobuf.jar:/usr/lib/hadoop//parquet-scala-2.10.jar:/usr/lib/hadoop//parquet-scrooge-2.10.jar:/usr/lib/hadoop//parquet-test-hadoop2.jar:/usr/lib/hadoop//parquet-thrift.jar:/usr/lib/hadoop//parquet-tools.jar:/usr/lib/hadoop//hadoop-anotations-2.6.0-cdh5.14.0.jar:/usr/lib/hadoop//hadoop-annotations.jar:/usr/lib/hadoop//hadoop-auth-2.6.0-cdh5.14.0.jar:/usr/lib/hadoop//hadoop-auth.jar:/usr/lib/hadoop//hadoop-aws-2.6.0-cdh5.14.0.jar:/usr/lib/hadoop//hadoop-aws.jar:/usr/lib/hadoop//hadoop-azure-datalake-2.6.0-cdh5.14.0.jar:/usr/lib/hadoop//hadoop-azure-datalake.jar:/usr/lib/hadoop//hadoop-common-2.6.0-cdh5.14.0-tests.jar:/usr/lib/hadoop//hadoop-common-2.6.0-cdh5.14.0.jar:/usr/lib/hadoop//hadoop-common-tests.jar:/usr/lib/hadoop//hadoop-common.jar:/usr/lib/hadoop//hadoop-nfs-2.6.0-cdh5.14.0.jar:/usr/lib/hadoop//hadoop-nfs.jar:/usr/lib/hadoop-hdfs://:/usr/lib/hadoop-hdfs/lib/asm-3.2.jar:/usr/lib/hadoop-hdfs/lib/commons-cli-1.2.jar:/usr/lib/hadoop-hdfs/lib/commons-codec-1.4.jar:/usr/lib/hadoop-hdfs/lib/commons-daemon-1.0.13.jar:/usr/lib/hadoop-hdfs/lib/commons-el-1.0.jar:/usr/lib/hadoop-hdfs/lib/commons-io-2.4.jar:/usr/lib/hadoop-hdfs/lib/commons-lang-2.6.jar:/usr/lib/hadoop-hdfs/lib/commons-logging-1.1.3.jar:/usr/lib/hadoop-hdfs/lib/guava-11.0.2.jar:/usr/lib/hadoop-hdfs/lib/htrace-core-4.0.1-incubating.jar:/usr/lib/hadoop-hdfs/lib/jackson-core-asl-1.8.8.jar:/usr/lib/hadoop-hdfs/lib/jackson-mapper-asl-1.8.8.jar:/usr/lib/hadoop-hdfs/lib/jasper-runtime-5.5.23.jar:/usr/lib/hadoop-hdfs/lib/jersey-core-1.9.jar:/usr/lib/hadoop-hdfs/lib/jersey-server-1.9.jar:/usr/lib/hadoop-hdfs/lib/jsp-api-2.1.jar:/usr/lib/hadoop-hdfs/lib/jsr305-3.0.0.jar:/usr/lib/hadoop-hdfs/lib/leveldbjni-all-1.8.jar:/usr/lib/hadoop-hdfs/lib/log4j-2.1.27.jar:/usr/lib/hadoop-hdfs/lib/netty-3.10.5.Final.jar:/usr/lib/hadoop-hdfs/lib/protoBuf-java-2.5
```

```
[jayantmohite@localhost ~]$ hdfs namenode -format
18/03/02 02:24:57 INFO namenode.FSNamesystem: HA Enabled: false
18/03/02 02:24:57 INFO namenode.FSNamesystem: Append Enabled: true
18/03/02 02:24:58 INFO util.GSet: Computing capacity for map INodeMap
18/03/02 02:24:58 INFO util.GSet: VM type = 64-bit
18/03/02 02:24:58 INFO util.GSet: 1.0% max memory 966.7 MB = 9.7 MB
18/03/02 02:24:58 INFO util.GSet: capacity = 2^20 = 1048576 entries
18/03/02 02:24:58 INFO namenode.FSDirectory: POSIX ACL inheritance enabled? false
18/03/02 02:24:58 INFO namenode.NameNode: Caching file names occurring more than 10 times
18/03/02 02:24:58 INFO snapshot.SnapshotManager: Loaded config captureOpenFiles: false, skipCaptureAccessTimeOnlyChange: false, snapshotDiffAllowSnapRootDesendant: true
18/03/02 02:24:58 INFO util.GSet: Computing capacity for map cachedBlocks
18/03/02 02:24:58 INFO util.GSet: VM type = 64-bit
18/03/02 02:24:58 INFO util.GSet: 0.25% max memory 966.7 MB = 2.4 MB
18/03/02 02:24:58 INFO util.GSet: capacity = 2^18 = 262144 entries
18/03/02 02:24:58 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 0.9990000128746033
18/03/02 02:24:58 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
18/03/02 02:24:58 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension = 0
18/03/02 02:24:58 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.window.num.buckets = 10
18/03/02 02:24:58 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.num.users = 10
18/03/02 02:24:58 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25
18/03/02 02:24:58 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
18/03/02 02:24:58 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
18/03/02 02:24:58 INFO util.GSet: Computing capacity for map NameNodeRetryCache
18/03/02 02:24:58 INFO util.GSet: VM type = 64-bit
18/03/02 02:24:58 INFO util.GSet: 0.29999999329447746% max memory 966.7 MB = 297.0 KB
18/03/02 02:24:58 INFO util.GSet: capacity = 2^15 = 32768 entries
18/03/02 02:24:58 INFO namenode.FSNamesystem: ACLs enabled? false
18/03/02 02:24:58 INFO namenode.FSNamesystem: XAttrs enabled? true
18/03/02 02:24:58 INFO namenode.FSNamesystem: Maximum size of an xattr: 16384
18/03/02 02:24:58 INFO namenode.FSImage: Allocated new BlockPoolId: BP-294060007-127.0.0.1-1519986298241
18/03/02 02:24:58 INFO common.Storage: Storage directory /var/lib/hadoop-hdfs/cache/hdfs/dfs/name has been successfully formatted.
18/03/02 02:24:58 INFO namenode.FSImageFormatProtobuf: Saving image file /var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current/fsimage.ckpt_00000000000000000000 using no compression
18/03/02 02:24:58 INFO namenode.FSImageFormatProtobuf: Image file /var/lib/hadoop-hdfs/cache/dfs/name/current/fsimage.ckpt_00000000000000000000 of size 321 bytes saved in 0 seconds.
18/03/02 02:24:58 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
18/03/02 02:24:58 INFO util.ExitUtil: Exiting with status 0
18/03/02 02:24:58 INFO namenode.NameNode: SHUTDOWN_MSG:
*****SHUTDOWN_MSG: Shutting down NameNode at localhost/127.0.0.1
*****[jayantmohite@localhost ~]$
```

Step 6:

Start only HDFS services



A screenshot of a Linux terminal window titled "jayantmohite@localhost:~". The window shows the command: `[jayantmohite@localhost ~]$ for x in `cd /etc/init.d ; ls hadoop-hdfs-*` ; do sudo service $x start ; done`. The output indicates that the datanode, namenode, and secondarynamenode services were started successfully, each with an "[OK]" message. The terminal window has a standard title bar with icons for Applications, Places, Terminal, and Help, and a status bar at the bottom showing the user's name and the date/time.

```
[jayantmohite@localhost ~]$ for x in `cd /etc/init.d ; ls hadoop-hdfs-*` ; do sudo service $x start ; done
starting datanode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-datanode-localhost.localdomain.out
Started Hadoop datanode (hadoop-hdfs-datanode): [ OK ]
starting namenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-namenode-localhost.localdomain.out
Started Hadoop namenode: [ OK ]
starting secondarynamenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-secondarynamenode-localhost.localdomain.out
Started Hadoop secondarynamenode: [ OK ]
[jayantmohite@localhost ~]$
```

Step 7:

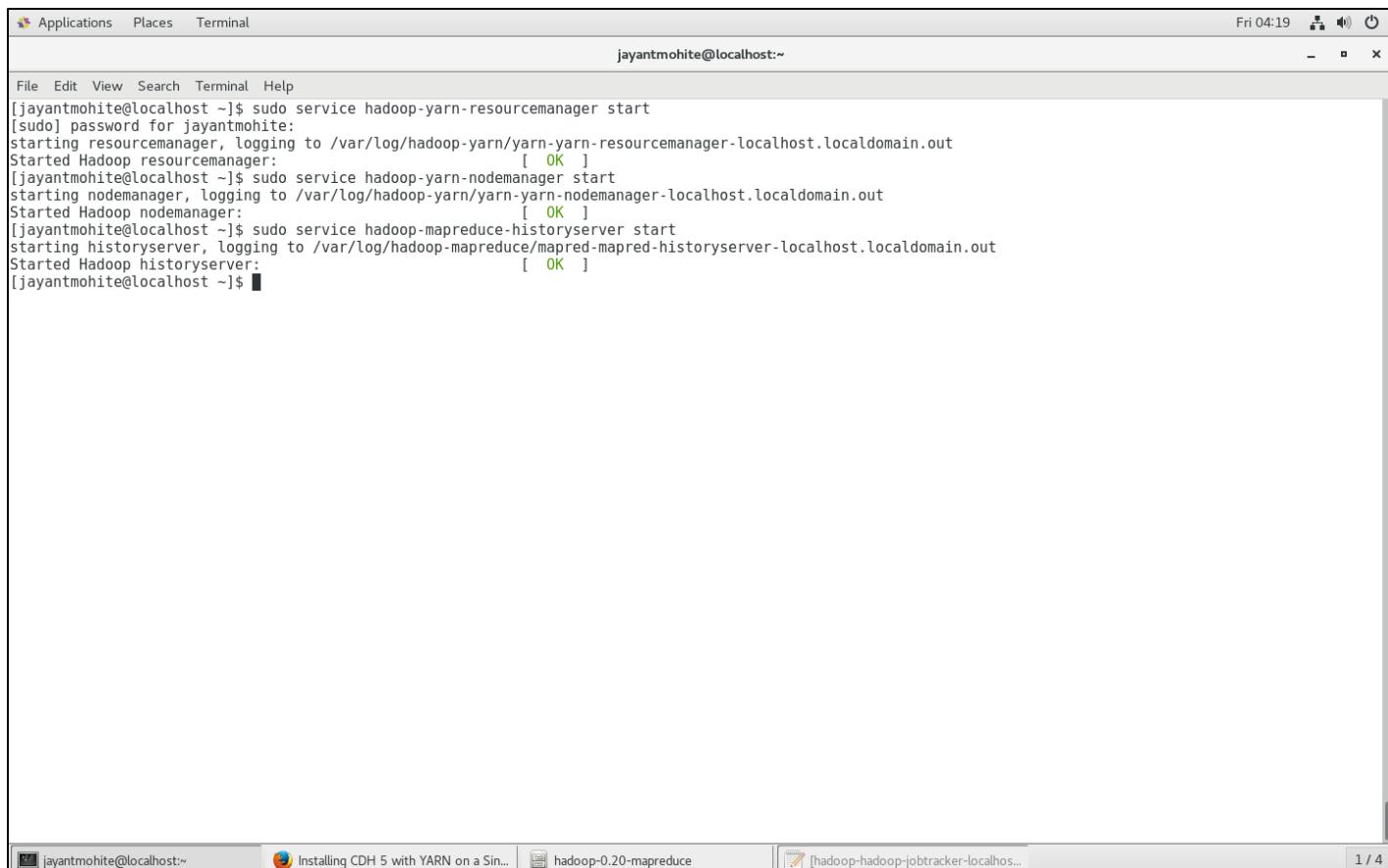
Run the init-hdfs Shell script to create the required directory structure

```
[jayantmohite@localhost ~]$ sudo /usr/lib/hadoop/libexec/init-hdfs.sh
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /tmp'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chmod -R 1777 /tmp'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /var'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /var/log'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chmod -R 1775 /var/log'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown yarn:mapred /var/log'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /tmp/hadoop-yarn'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown -R mapred:mapred /tmp/hadoop-yarn'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /tmp/hadoop-yarn/staging/history/done_intermediate'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown -R mapred:mapred /tmp/hadoop-yarn/staging'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chmod -R 1777 /tmp'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /var/log/hadoop-yarn/apps'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chmod -R 1777 /var/log/hadoop-yarn/apps'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown yarn:mapred /var/log/hadoop-yarn/apps'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /hbase'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown hbase /hbase'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /benchmarks'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/history'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown mapred /user/history'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/jenkins'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chmod -R 777 /user/jenkins'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown jenkins /user/jenkins'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/hive'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chmod -R 777 /user/hive'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown hive /user/hive'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/root'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chmod -R 777 /user/root'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown root /user/root'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/hue'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown -R 777 /user/hue'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown hue /user/hue'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/oozie'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/oozie/share'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/oozie/share/lib'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/oozie/share/lib/hive'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/oozie/share/lib/mapreduce-streaming'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/oozie/share/lib/distcp'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /user/oozie/share/lib/pig'
```

```
[jayantmohite@localhost ~]$ ls /usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.14.0.jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar
+ ls /usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.14.0.jar /user/oozie/share/lib/mapreduce-streaming
+ cp /usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.14.0.jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar
+ cp /usr/lib/hadoop-mapreduce/hadoop-streaming.jar /user/oozie/share/lib/mapreduce-streaming
+ cp /usr/lib/hadoop-mapreduce/hadoop-distcp-2.6.0-cdh5.14.0.jar /usr/lib/hadoop-mapreduce/hadoop-distcp.jar
+ cp /usr/lib/hadoop-mapreduce/hadoop-distcp*.jar /user/oozie/share/lib/distcp
+ cp /usr/lib/hadoop-mapreduce/hadoop-distcp*.jar /user/oozie/share/lib/distcp
+ cp /usr/lib/pig/lib/*.jar /usr/lib/pig/*.jar
+ cp /usr/lib/sqoop/lib/*.jar /usr/lib/sqoop/*.jar
+ chmod -R 777 /user/oozie
+ chown -R oozie /user/oozie
+ mkdir -p /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
```

Step 8:

Start Yarn Services



A screenshot of a terminal window titled "jayantmohite@localhost:~". The window shows the command-line interface for starting Hadoop YARN services. The user runs "sudo service hadoop-yarn-resourcemanager start" and "sudo service hadoop-yarn-nodemanager start", both of which succeed with "[OK]". Then, the user runs "sudo service hadoop-mapreduce-historyserver start", which also succeeds with "[OK]". The terminal window has a standard Linux desktop interface at the top, including application icons and a clock. At the bottom, there are tabs for other open windows: "jayantmohite@localhost:~" (active), "Installing CDH 5 with YARN on a Sin...", "hadoop-0.20-mapreduce", and "[hadoop-hadoop-jobtracker-localhos...]".

```
[jayantmohite@localhost ~]$ sudo service hadoop-yarn-resourcemanager start
[sudo] password for jayantmohite:
starting resourcemanager, logging to /var/log/hadoop-yarn/yarn-yarn-resourcemanager-localhost.localdomain.out
Started Hadoop resourcemanager: [ OK ]
[jayantmohite@localhost ~]$ sudo service hadoop-yarn-nodemanager start
starting nodemanager, logging to /var/log/hadoop-yarn/yarn-yarn-nodemanager-localhost.localdomain.out
Started Hadoop nodemanager: [ OK ]
[jayantmohite@localhost ~]$ sudo service hadoop-mapreduce-historyserver start
starting historyserver, logging to /var/log/hadoop-mapreduce/mapred-mapred-historyserver-localhost.localdomain.out
Started Hadoop historyserver: [ OK ]
[jayantmohite@localhost ~]$
```

Step 9:

Create Directory for User and Sample Input

The screenshot shows a terminal window titled "jayantmohite@localhost:~". The terminal session is as follows:

```
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -mkdir /user/jayantmohite
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -chown jayantmohite /user/jayantmohite
[jayantmohite@localhost ~]$ hadoop fs -mkdir input
[jayantmohite@localhost ~]$ hadoop fs -put /etc/hadoop/conf/*.xml input
[jayantmohite@localhost ~]$ hadoop fs -ls input
Found 4 items
-rw-r--r-- 1 jayantmohite supergroup      2133 2018-03-02 04:22 input/core-site.xml
-rw-r--r-- 1 jayantmohite supergroup      2324 2018-03-02 04:22 input/dfs-site.xml
-rw-r--r-- 1 jayantmohite supergroup      1549 2018-03-02 04:22 input/mapred-site.xml
-rw-r--r-- 1 jayantmohite supergroup      2375 2018-03-02 04:22 input/yarn-site.xml
[jayantmohite@localhost ~]$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
[jayantmohite@localhost ~]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input output23 'dfs[a-z.]+'
```

The terminal window has a title bar with "Applications", "Places", "Terminal", and a date/time "Fri 04:23". The bottom status bar shows the user "jayantmohite@localhost:~" and three tabs: "Installing CDH 5 with YARN on a Sin...", "hadoop-0.20-mapreduce", and "[hadoop-hadoop-jobtracker-localhos...]".

Step 10:

Execute Sample Program

```

Applications Places Terminal jayantmohite@localhost:~ Fri 04:24
File Edit View Search Terminal Help
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -mkdir /user/jayantmohite
[jayantmohite@localhost ~]$ sudo -u hdfs hadoop fs -chown jayantmohite /user/jayantmohite
[jayantmohite@localhost ~]$ hadoop fs -mkdir input
[jayantmohite@localhost ~]$ hadoop fs -put /etc/hadoop/conf/*.xml input
[jayantmohite@localhost ~]$ hadoop fs -ls input
Found 4 items
-rw-r--r-- 1 jayantmohite supergroup 2133 2018-03-02 04:22 input/core-site.xml
-rw-r--r-- 1 jayantmohite supergroup 2324 2018-03-02 04:22 input/hdfs-site.xml
-rw-r--r-- 1 jayantmohite supergroup 1549 2018-03-02 04:22 input/mapred-site.xml
-rw-r--r-- 1 jayantmohite supergroup 2375 2018-03-02 04:22 input/yarn-site.xml
[jayantmohite@localhost ~]$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
[jayantmohite@localhost ~]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input output23 'dfs[a-z.]+'
18/03/02 04:24:04 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
18/03/02 04:24:06 WARN mapreduce.JobResourceUploader: No job jar file set. User classes may not be found. See Job or Job#setJar(string).
18/03/02 04:24:07 INFO input.FileInputFormat: Total input paths to process : 4
18/03/02 04:24:09 INFO mapreduce.JobSubmitter: number of splits:4
18/03/02 04:24:10 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1519993085263_0001
18/03/02 04:24:10 INFO mapred.YARNRunner: Job jar is not present. Not adding any jar to the list of resources.
18/03/02 04:24:11 INFO impl.YarnClientImpl: Submitted application application_1519993085263_0001
18/03/02 04:24:11 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1519993085263_0001/
18/03/02 04:24:11 INFO mapreduce.Job: Running job: job_1519993085263_0001

```

jayantmohite@localhost:~ Installing CDH 5 with YARN on a Sin... hadoop-0.20-mapreduce [hadoop-hadoop-jobtracker-localhos...]

```

Applications Places Terminal jayantmohite@localhost:~ Fri 04:27
File Edit View Search Terminal Help
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=13379
Total time spent by all reduces in occupied slots (ms)=11964
Total time spent by all map tasks (ms)=13379
Total time spent by all reduce tasks (ms)=11964
Total vcore-milliseconds taken by all map tasks=13379
Total vcore-milliseconds taken by all reduce tasks=11964
Total megabyte-milliseconds taken by all map tasks=13700096
Total megabyte-milliseconds taken by all reduce tasks=12251136
Map-Reduce Framework
Map input records=10
Map output records=10
Map output bytes=304
Map output materialized bytes=330
Input split bytes=138
Combine input records=0
Combine output records=0
Reduce input groups=1
Reduce shuffle bytes=330
Reduce input records=10
Reduce output records=10
Spilled Records=20
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=274
CPU time spent (ms)=2350
Physical memory (bytes) snapshot=269791232
Virtual memory (bytes) snapshot=5107773440
Total committed heap usage (bytes)=137498624
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=470
File Output Format Counters
Bytes Written=244
[jayantmohite@localhost ~]$ 

```

jayantmohite@localhost:~ Installing CDH 5 with YARN on a Sin... hadoop-0.20-mapreduce [hadoop-hadoop-jobtracker-localhos...]

Sample View of YARN browser

The screenshot shows the 'All Applications' page of the YARN browser. The left sidebar includes a cluster navigation menu with options like 'About Nodes Applications', 'Scheduler', and 'Tools'. The main content area displays 'Cluster Metrics' and 'Cluster Nodes Metrics' tables. Below these are sections for 'User Metrics for dr.who' and a detailed table of application logs. The table lists two completed applications:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores
application_1519993085263_0002	jayantmohite	grep-sort	MAPREDUCE	root.jayantmohite	Fri Mar 2 04:26:29 2018	Fri Mar 2 04:27:21 -0800 2018	FINISHED	SUCCEEDED	N/A	N/A
application_1519993085263_0001	jayantmohite	grep-search	MAPREDUCE	root.jayantmohite	Fri Mar 2 04:24:11 2018	Fri Mar 2 04:26:26 -0800 2018	FINISHED	SUCCEEDED	N/A	N/A

At the bottom, there is a footer bar with links to 'All Applications - Mozilla Firefox', 'hadoop-0.20-mapreduce', and 'hadoop-hadoop-jobtracker-localhost...'.

The screenshot shows the 'About the Cluster' page of the YARN browser. The left sidebar includes a cluster navigation menu with options like 'About Nodes Applications', 'Scheduler', and 'Tools'. The main content area displays 'Cluster Metrics' and 'Cluster Nodes Metrics' tables. Below these are sections for 'User Metrics for dr.who' and a detailed table of cluster configuration parameters. The table includes fields such as Cluster ID, ResourceManager state, ResourceManager HA state, ResourceManager HA connection state, ResourceManager RMStateStore, ResourceManager started on, ResourceManager version, and Hadoop version. The 'Cluster overview' section at the bottom contains a summary of these configurations.

Cluster ID:	1519993085263
ResourceManager state:	STARTED
ResourceManager HA state:	active
ResourceManager HA connection state:	ResourceManager HA is not enabled.
ResourceManager RMStateStore:	org.apache.hadoop.yarn.server.resourcemanager.recovery.NullRMStateStore
ResourceManager started on:	Fri Mar 02 04:18:05 -0800 2018
ResourceManager version:	2.6.0-cdh5.14.0 from 9b197d35839383c798c618ba917ccaa196a17699 by jenkins source checksum 97b766559320ef98d44e45c06ccb5f66 on 2018-01-06T21:47Z
Hadoop version:	2.6.0-cdh5.14.0 from 9b197d35839383c798c618ba917ccaa196a17699 by jenkins source checksum f4ddee45985a34faa91db2aad8731f on 2018-01-06T21:38Z

At the bottom, there is a footer bar with links to 'About the Cluster - Mozilla Firefox', 'hadoop-0.20-mapreduce', and 'hadoop-hadoop-jobtracker-localhost...'.

Applications Places Firefox Web Browser Fri 04:29

Browsing HDFS NEW,NEW_SAVING,SUBMITTED,ACCEPTED,RUNNING Applications - Mozilla Firefox

localhost:8088/cluster/scheduler

hadoop NEW,NEW_SAVING,SUBMITTED,ACCEPTED,RUNNING Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
2	0	0	2	0	0 B	8 GB	0 B	0	8	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0

Application Queues

Legend: Steady Fair Share Instantaneous Fair Share Used Used (over fair share) Max Capacity

- root
 - + root.default
 - + root.jayantmohite

Show 20 entries Search:

ID	User	Name	Application Type	Queue	Fair Share	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Reserved CPU Vcores	Reserved Memory MB	Progress	Track UI
No data available in table																

Showing 0 to 0 of 0 entries First Previous Next Last

1 / 4

Applications Places Firefox Web Browser Fri 04:29

Browsing HDFS Nodes of the cluster - Mozilla Firefox

localhost:8088/cluster/nodes

hadoop Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
2	0	0	2	0	0 B	8 GB	0 B	0	8	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
/default-rack	RUNNING	localhost:46221	localhost:8042	Fri Mar 02 04:28:27 -0800 2018	0	0 B	8 GB	0	8	2.6.0-cdh5.14.0		

Showing 1 to 1 of 1 entries First Previous 1 Next Last

1 / 4

15. Introduction to NoSQL Databases

Before talking about NoSQL let's try to list down some advantages of SQL and see the reasons why they were so popular from the last 3 decades.

Advantages of SQL Databases:

1. Schema

- a. They can handle Structured Data in a neat and organised way
- b. Organised Schema gives rise to a uniform application cycle in which Database Models are designed first and later applications. This provides stability as well.

2. Scaling

- a. We can Scale Vertically to meet the growing needs of Storage Space

3. SQL

- a. SQL provides an easy and faster way of querying data.
- b. Not all the Tables are standalone as some do hold relations with other tables. This can be realized using Joins.

4. Properties

- a. ACID properties that governed these Databases, assured that these Databases provide high level of Atomicity, Consistency, Isolation and Durability.
- b. To understand ACID, consider the following scenario
- c. For example, "customer A" has INR 3,500 in his account and "Customer B" has INR 4,000 in his account. Now "Customer A" wants to transfer INR 1000 to "Customer B".
 - i. **Atomicity** defines that, "Customer A" will initiate the transaction. Money will get debited from account balance of "Customer A" and will get credited to account balance of "Customer B". This process will complete the Transaction.

- ii. **Consistency** defines that, if “Customer A” attempts to transfer INR 1000 to the account of “Customer B”, then only INR 1000 will be debited from account balance of “Customer A” and his final balance will become INR 2,500 and only INR 1000 will get credited to account balance of “Customer B” and his final balance will become INR 5,000. In this way the total of both accounts, before and after transactions will also be the same i.e. INR 7,500.
- iii. **Isolation** defines that, if “Customer A” wants to make another transaction of transferring INR 3,000 to “Customer C”, then first Consistency and Atomicity will be completed for the previous transactions. Only then if he still has the required balance, he can make out the next transaction.
- iv. **Durability** defines that, once the Transaction is complete and the data related to that is changed, it remains in the same changed state forever unless any other transaction does not alter it.

5. Searching, Sorting and Functions

- a. Aggregation Functions that provide an easy way to apply complex operations of values.
- b. Indexing that makes searching and Sorting Easier.

Now let's try to compare, what it needs to handle Big Data and what our SQL Databases offer

	Big Data Storage Needs	SQL Offerings
Schema	Unstructured Data with lot of variety in Data Types and Structures	Can handle only Structured Data
Scaling	Unlimited Scaling with Sever Always available as data grows every minute	Supports only Vertical Scaling that involves Upper Limit and Downtime
Querying	Anything that can work on distributed Data Stored in Flat Files	Provides SQL that can work only on Tables stored in one Server
Application Development	Expects Applications to be developed and modified at a fast rate to match up to the growing competition.	Follows a Slow process in which Database Design restricts the scope of Development and Upgrade in Applications.

One thing is clear from the above table that, all the advantages of SQL Databases actually turns out to be limitations hindering their capabilities to handle Big Data. From all this discussion we can certainly conclude on one thing that to handle Big Data, we really need some Databases that **do not**:

1. Only support Structured Data
2. Have a Schema
3. Enforce dependency on database for application design
4. Support Vertical Scaling
5. Support SQL
6. Support Relations and JOINs
7. Support ACID

So we came up with a family of databases that

1. Supports Structured as well as Unstructured Data.
2. Stores data in Flat File System
 - a. Data is stored in Binary Format
3. Can support anything as long as your application can understand it
 - a. For Example, you can write the value of the field "AGE" as either 24 or "twenty-four" or "20 + 4"
4. Does not have support to the concept of Data Types
5. Can scale Horizontally
6. Use functions to query the data. So that working with Flat Files becomes easy.
7. Will embed all properties of an entity with one object itself i.e. have embedded objects. This eradicates the need of maintaining Joins as Joins become expensive with the increase in number of Tables.

These Databases are called as [NoSQL Databases](#), popularly called as [Not Only SQL Databases](#). These Databases do not follow any properties of the Traditional RDBMS. Based on how they store and Retrieve Data, they can be broadly classified as

1. Key Value Pair Store Databases
 - a. Popular in this category is [Amazon's Dynamo](#)
2. Columnar Store Databases
 - a. Popular in this category are [Google's Big Table](#) and [Facebook's Cassandra](#).
3. Document Store Databases
 - a. Popular in this category is [10 Gen's \[now known as MongoDB University\] MongoDB](#)
4. Graph Based Databases.
 - a. Popular in this category is [Neo4j](#)

While learning about NoSQL databases, it is really important to understand about the BASE properties that govern these databases and [the CAP' s Theorem](#) which justified these properties.

[CAP' s Theorem:](#)

[Cap Brewer](#) defined that, any distributed system can be defined by 3 of its characteristics namely [Consistency](#), [Availability](#) and [Partition Tolerance](#).

To understand Partition Tolerance, consider “[Zee TV](#)” has a central broadcasting unit in Delhi location of India. Now it provides Services to Mumbai as well as Bangalore. This is called as Partition. Now, if there is a problem in the link between Mumbai and Delhi and if the Mumbai people are not able to view Zee TV, then this does not affect the Bangalore people. The Bangalore guys will continue their viewing as their connection with Delhi is not hampered. This is called as Partition Tolerance.

CAP' s theorem further states that, no distributed system on the planet can guarantee all these 3 properties together. They need to compromise on one of these properties so that they can possess the other 2 in abundance.

With this evolved **the BASE properties** that govern the architecture of all NoSQL databases. BASE is defined into 3 characteristics

1. Basically Available
2. Soft state
3. Eventual consistency

Basically Available assures that the server will atleast seem to be running all the time in any prevailing conditions. For example, assume, it would be a situation where you are in an Online Class and you can hear what the Trainer is talking but you cannot see his shared Screen. Means atleast something is happening.

Soft State and **Eventual Consistency** go hand in hand where Soft State identifies the state of Database to be inconsistent sometimes, which is guaranteed by Eventual Consistency to become consistent after a while. To understand this, consider the famous example of Booking a Ticket on Book My Show wherein you select a Movie, Theatre, Show Data and Times, Seats and you proceed for payment. But when it was time for you to get an OTP for completing the payment, then you realise that there is no network in your cell phone and you cancel the transaction. Soon you realise that network is back and you again try to book the same tickets. So you select the same Movie, same Theatre, same Show Dates and Time but you realise that now the seats that you wanted are booked. If you wait for another 15-20 minutes and then try again, you will find those seats available. This means that the Database was not consistent immediately but got consistent after some updates.

This completely your introduction to NoSQL databases and you can now get ahead and start you journey with MongoDB.

16. Apache Hbase

Since [1970](#), RDBMS is the solution for data storage and maintenance related problems. After the advent of big data, companies realized the benefit of processing big data and started opting for solutions like Hadoop.

Hadoop uses distributed file system for storing big data, and MapReduce to process it. Hadoop excels in storing and processing of huge data of various formats such as arbitrary, semi-, or even unstructured.

Limitations of Hadoop

Hadoop can perform only batch processing, and data will be accessed only in a sequential manner. That means one has to search the entire dataset even for the simplest of jobs.

A huge dataset when processed results in another huge data set, which should also be processed sequentially. At this point, a new solution is needed to access any point of data in a single unit of time ([random access](#)).

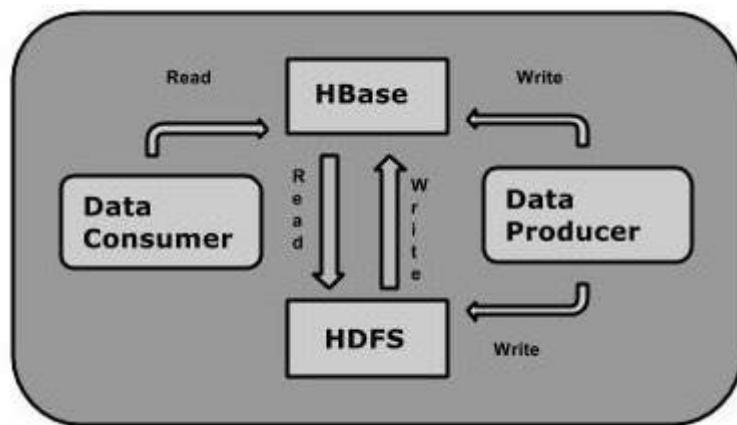
Hadoop Random Access Databases

Applications such as [HBase](#), [Cassandra](#), [couchDB](#), [Dynamo](#), and [MongoDB](#) are some of the databases that store huge amounts of data and access the data in a random manner.

What is HBase?

HBase is a [distributed column-oriented database](#) built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. HBase is a data model that is similar to [Google's big table](#) designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System. One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.



HBase and HDFS

HDFS	HBase
HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
It provides high latency batch processing; no concept of batch processing.	It provides low latency access to single rows from billions of records (Random access).
It provides only sequential access of data.	HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups.

Storage Mechanism in HBase

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

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

Given below is an example schema of table in HBase.

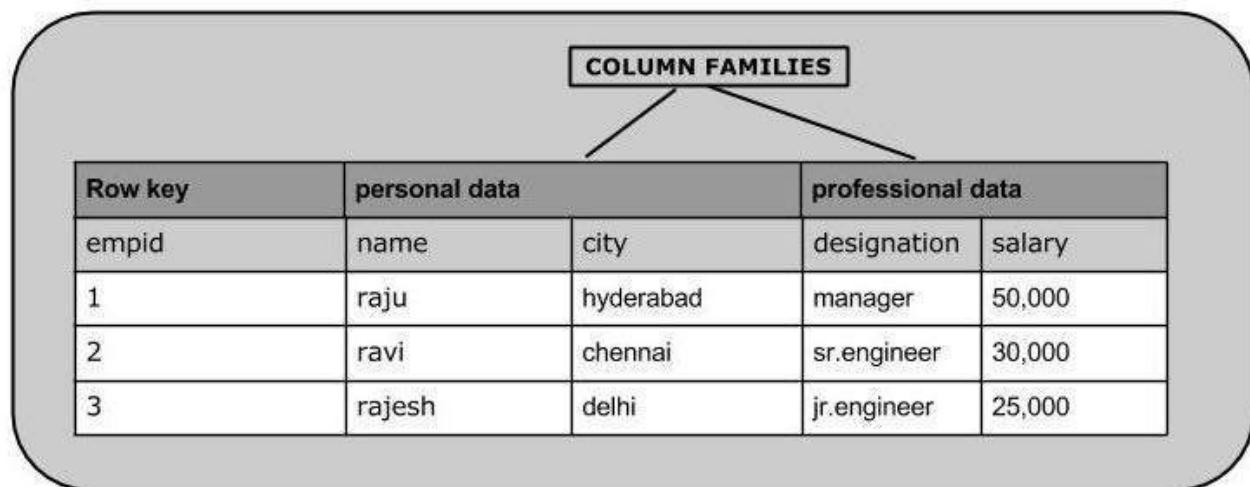
Rowid	Column Family											
	col1	col2	col3									
1												
2												
3												

Column Oriented and Row Oriented

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

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

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



HBase and RDBMS

HBase	RDBMS
HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families.	An RDBMS is governed by its schema, which describes the whole structure of tables.
It is built for wide tables. HBase is horizontally scalable.	It is thin and built for small tables. Hard to scale.
No transactions are there in HBase.	RDBMS is transactional.
It has de-normalized data.	It will have normalized data.
It is good for semi-structured as well as structured data.	It is good for structured data.

Features of HBase

- HBase is linearly scalable.
- It has automatic failure support.
- It provides consistent read and writes.
- It integrates with Hadoop, both as a source and a destination.
- It has easy java API for client.
- It provides data replication across clusters.

Where to Use HBase

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

Applications of HBase

- It is used whenever there is a need to write heavy applications.
- HBase is used whenever we need to provide fast random access to available data.
- Companies such as Facebook, Twitter, Yahoo, and Adobe use HBase internally.

HBase History

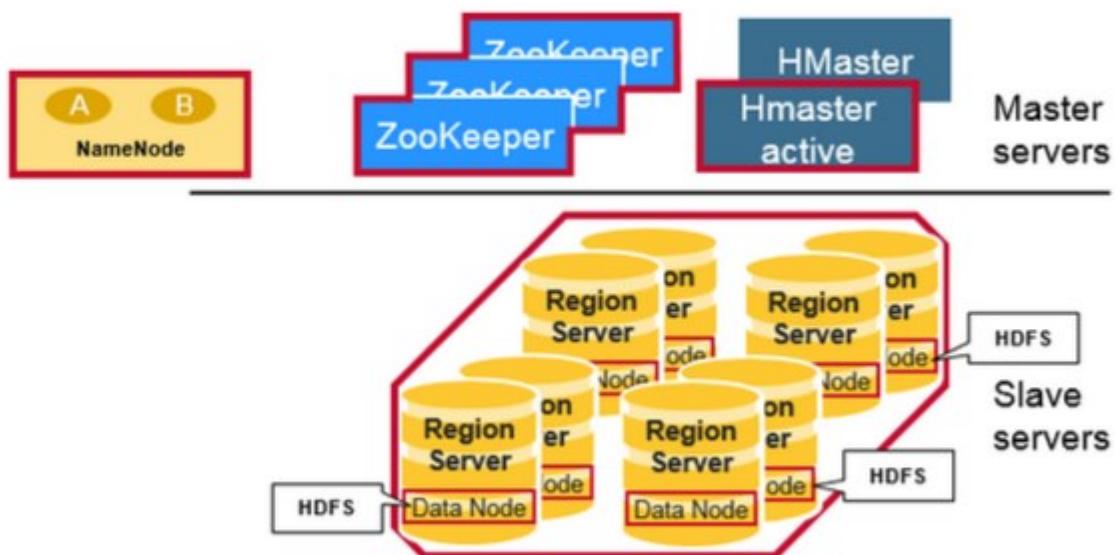
Year	Event
Nov 2006	Google released the paper on BigTable.
Feb 2007	Initial HBase prototype was created as a Hadoop contribution.
Oct 2007	The first usable HBase along with Hadoop 0.15.0 was released.
Jan 2008	HBase became the sub project of Hadoop.
Sept 2009	HBase 0.20.0 was released.
May 2010	HBase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

HBase Architectural Components

Physically, HBase is composed of three types of servers in a master slave type of architecture. Region servers serve data for reads and writes. When accessing data, clients communicate with HBase RegionServers directly. Region assignment, DDL (create, delete tables) operations are handled by the HBase Master process. Zookeeper, which is part of HDFS, maintains a live cluster state.

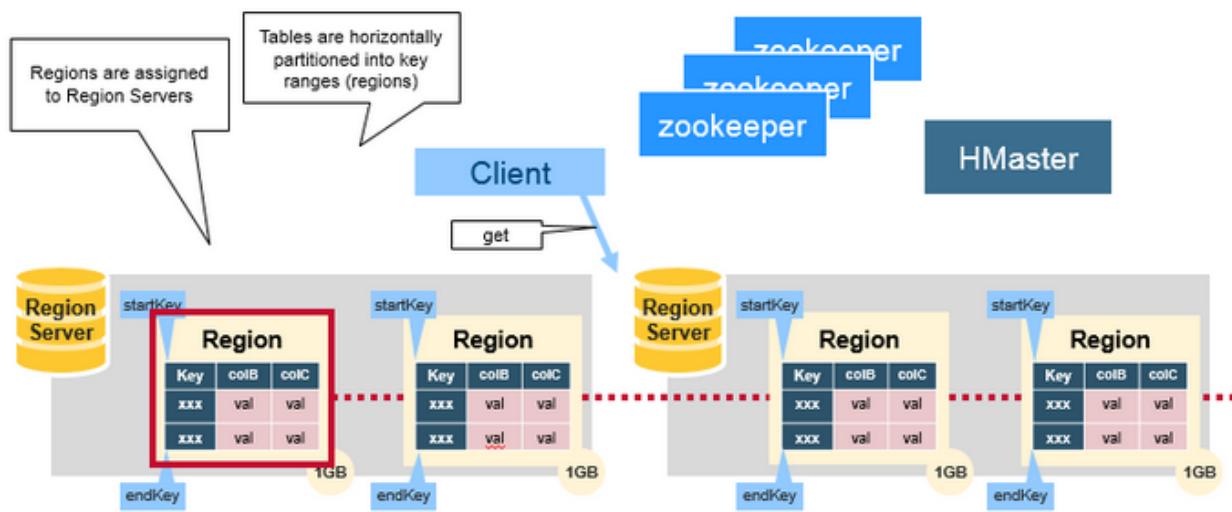
The Hadoop DataNode stores the data that the Region Server is managing. All HBase data is stored in HDFS files. Region Servers are collocated with the HDFS DataNodes, which enable data locality (putting the data close to where it is needed) for the data served by the RegionServers. HBase data is local when it is written, but when a region is moved, it is not local until compaction.

The NameNode maintains metadata information for all the physical data blocks that comprise the files.



Regions

HBase Tables are divided horizontally by row key range into “Regions.” A region contains all rows in the table between the region’s start key and end key. Regions are assigned to the nodes in the cluster, called “Region Servers,” and these serve data for reads and writes. A region server can serve about 1,000 regions.

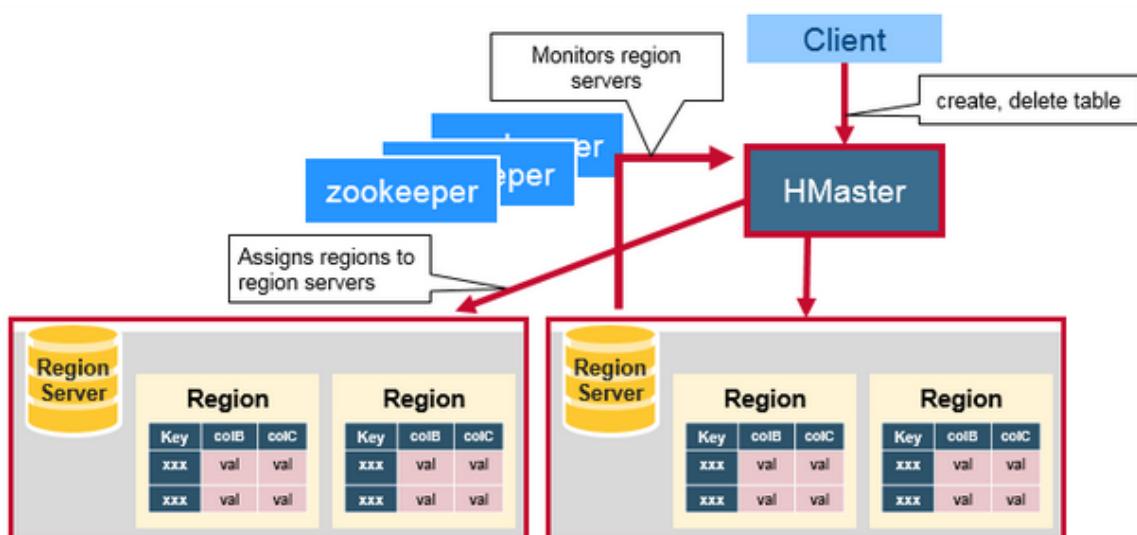


HBase HMaster

Region assignment, DDL (create, delete tables) operations are handled by the HBase Master.

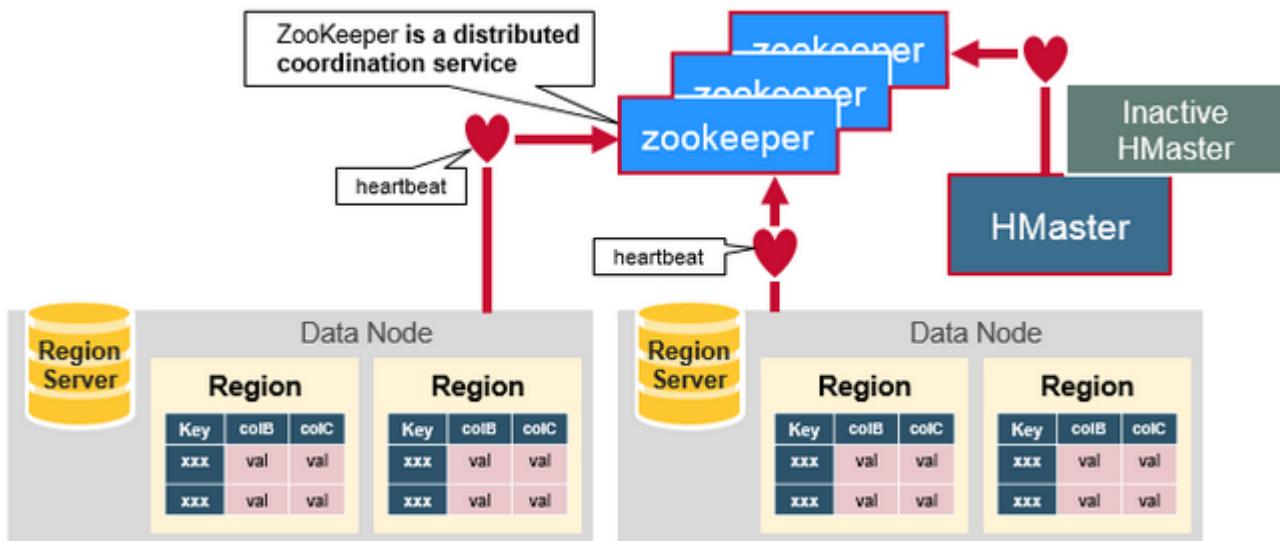
A master is responsible for:

- Coordinating the region servers
 - Assigning regions on startup , re-assigning regions for recovery or load balancing
 - Monitoring all RegionServer instances in the cluster (listens for notifications from zookeeper)
- Admin functions
 - Interface for creating, deleting, updating tables



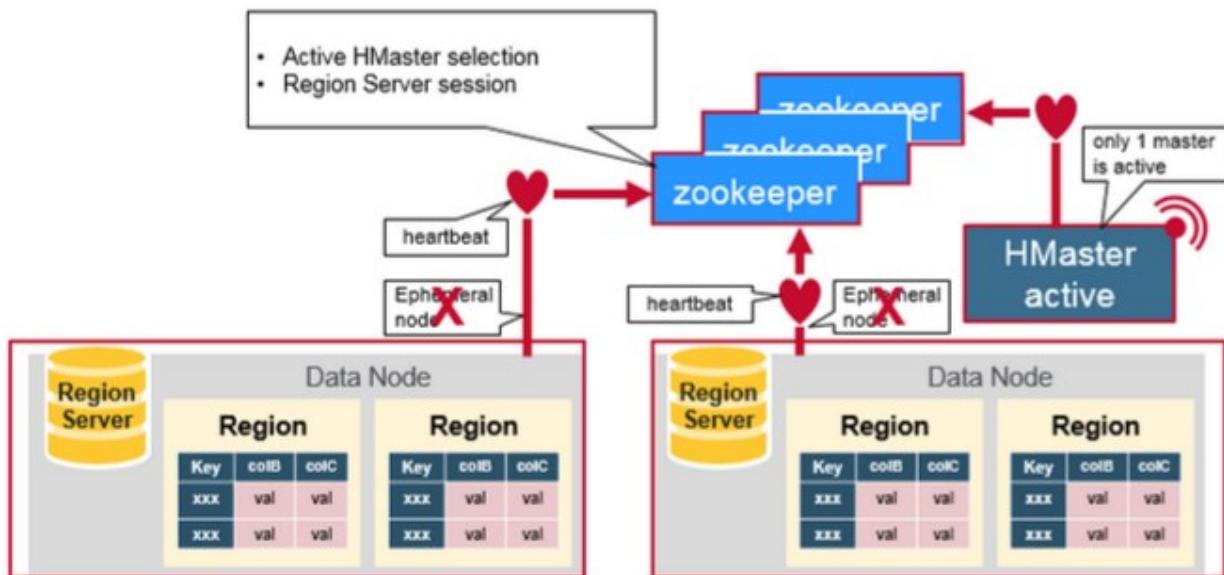
ZooKeeper: The Coordinator

HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster. Zookeeper maintains which servers are alive and available, and provides server failure notification. Zookeeper uses consensus to guarantee common shared state. Note that there should be three or five machines for consensus.



How the Components Work Together

Zookeeper is used to coordinate shared state information for members of distributed systems. Region servers and the active HMaster connect with a session to ZooKeeper. The ZooKeeper maintains ephemeral nodes for active sessions via heartbeats.



Each Region Server creates an ephemeral node. The HMaster monitors these nodes to discover available region servers, and it also monitors these nodes for server failures. HMasters vie to create an ephemeral node. Zookeeper determines the first one and uses it to make sure that only one master is active. The active HMaster sends heartbeats to Zookeeper, and the inactive HMaster listens for notifications of the active HMaster failure.

If a region server or the active HMaster fails to send a heartbeat, the session is expired and the corresponding ephemeral node is deleted. Listeners for updates will be notified of the deleted nodes. The active HMaster listens for region servers, and will recover region servers on failure. The Inactive HMaster listens for active HMaster failure, and if an active HMaster fails, the inactive HMaster becomes active.

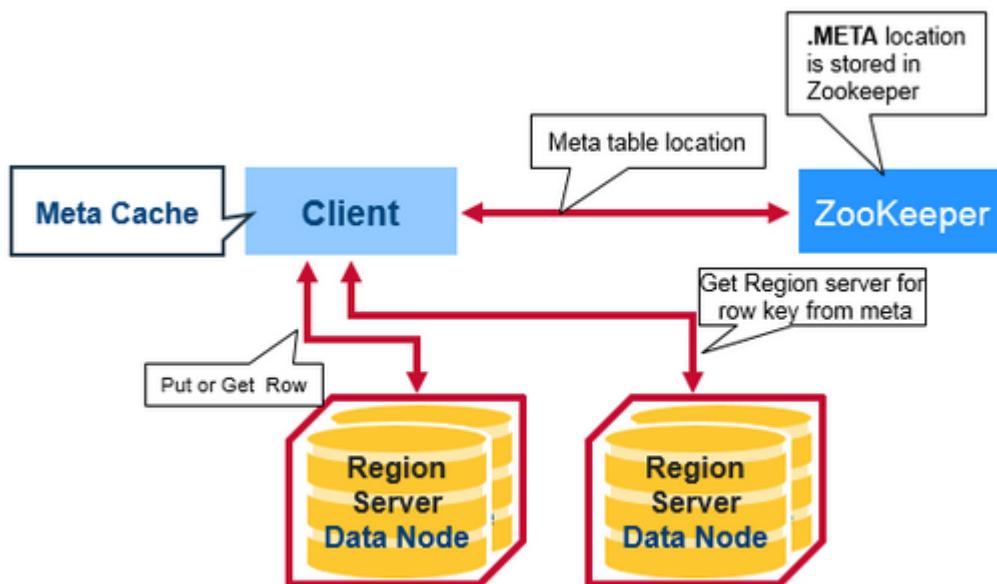
HBase First Read or Write

There is a special HBase Catalog table called the META table, which holds the location of the regions in the cluster. ZooKeeper stores the location of the META table.

This is what happens the first time a client reads or writes to HBase:

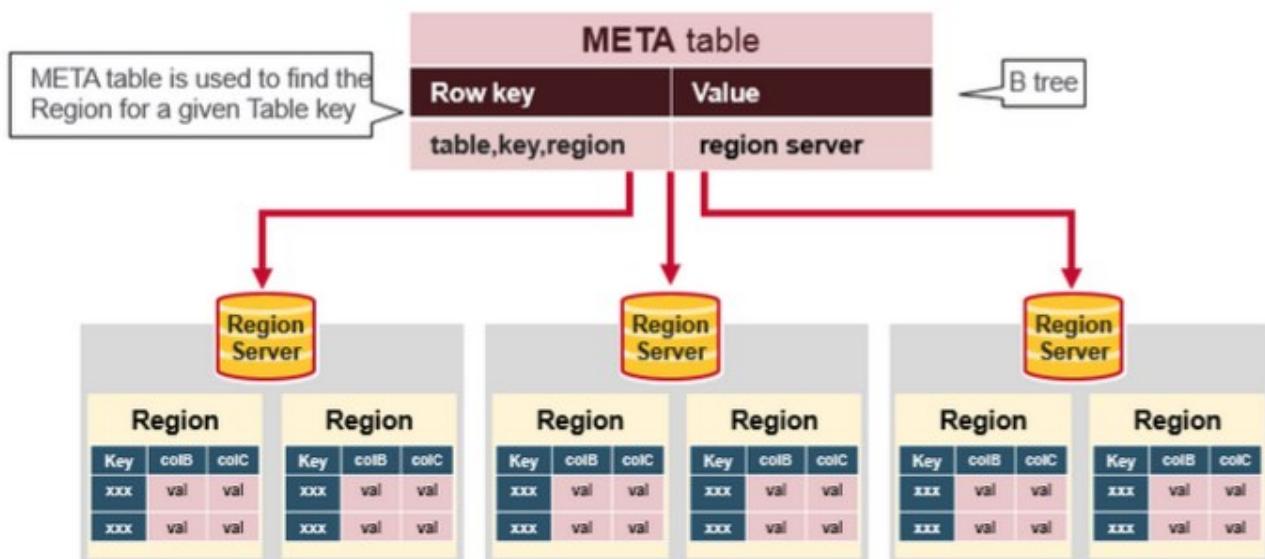
1. The client gets the Region server that hosts the META table from ZooKeeper.
2. The client will query the .META. server to get the region server corresponding to the row key it wants to access. The client caches this information along with the META table location.
3. It will get the Row from the corresponding Region Server.

For future reads, the client uses the cache to retrieve the META location and previously read row keys. Over time, it does not need to query the META table, unless there is a miss because a region has moved; then it will re-query and update the cache.



HBase Meta Table

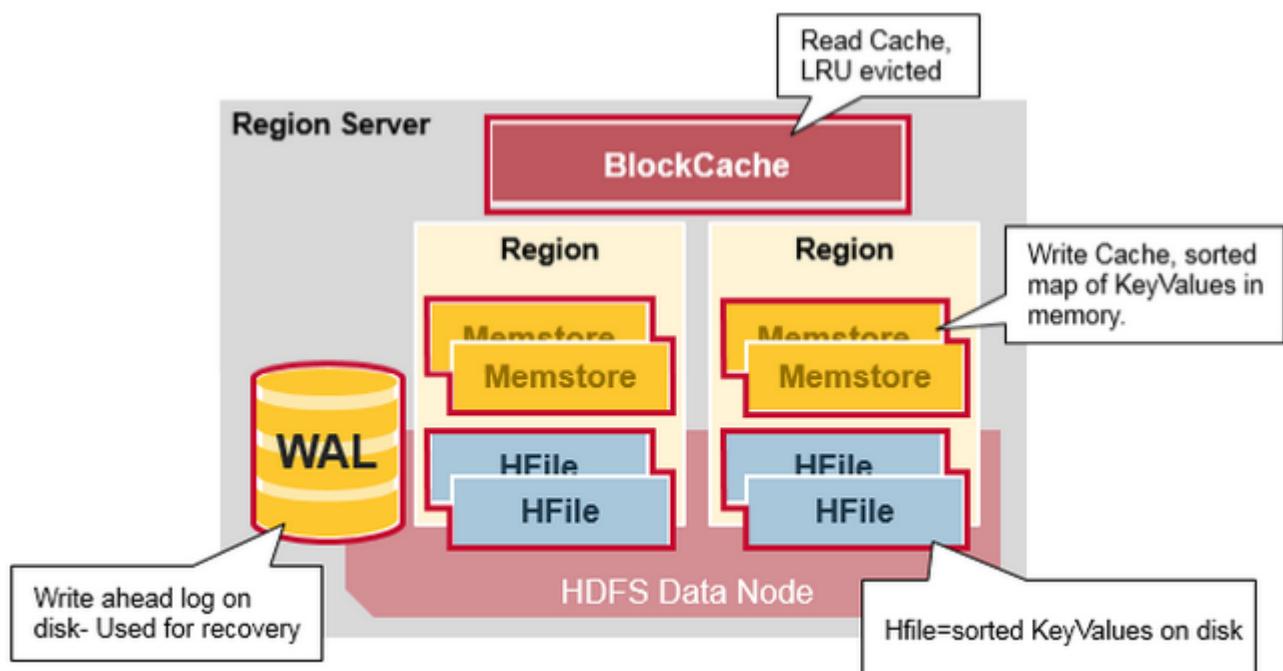
- This META table is an HBase table that keeps a list of all regions in the system.
- The .META. table is like a b tree.
- The .META. table structure is as follows:
 - Key: region start key,region id
 - Values: RegionServer



Region Server Components

A Region Server runs on an HDFS data node and has the following components:

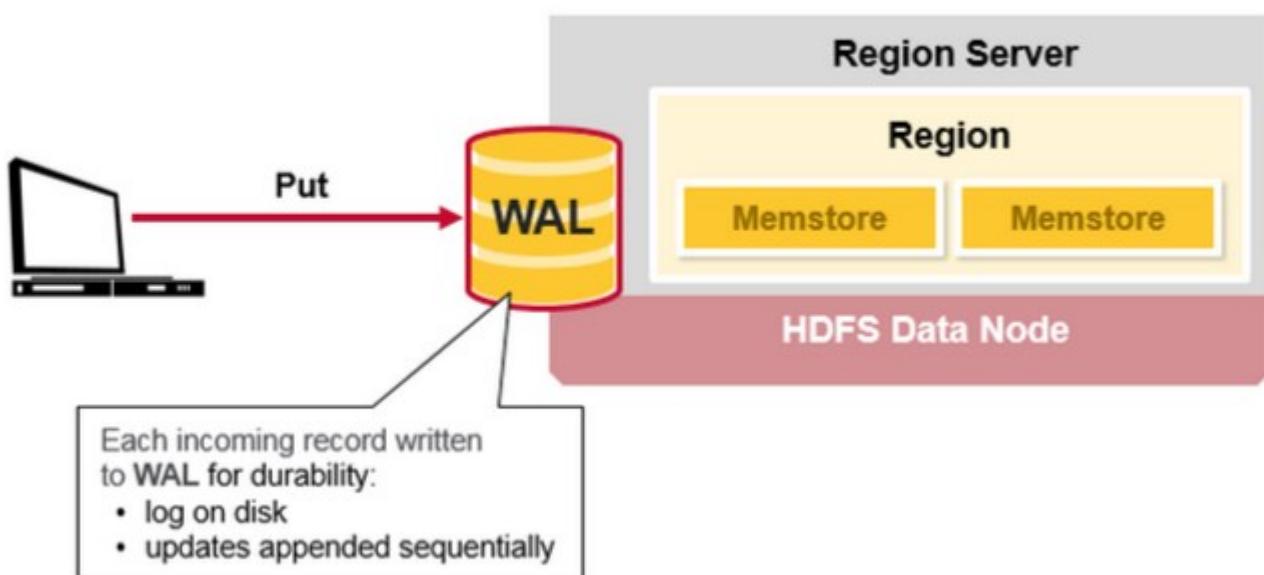
- WAL: Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't yet been persisted to permanent storage; it is used for recovery in the case of failure.
- BlockCache: is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when full.
- MemStore: is the write cache. It stores new data which has not yet been written to disk. It is sorted before writing to disk. There is one MemStore per column family per region.
- Hfiles store the rows as sorted KeyValues on disk.



HBase Write Steps (1)

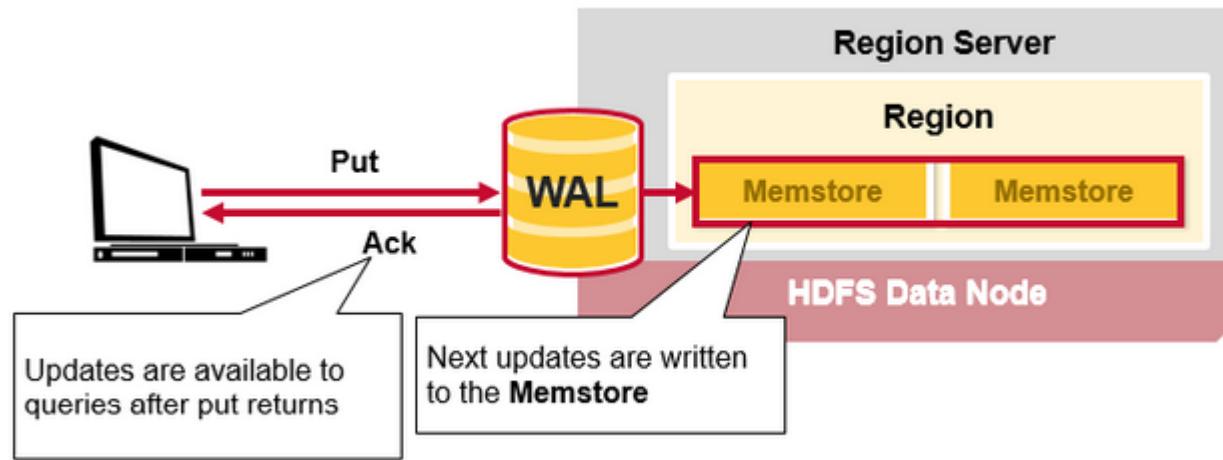
When the client issues a Put request, the first step is to write the data to the write-ahead log, the WAL:

- Edits are appended to the end of the WAL file that is stored on disk.
- The WAL is used to recover not-yet-persisted data in case a server crashes.



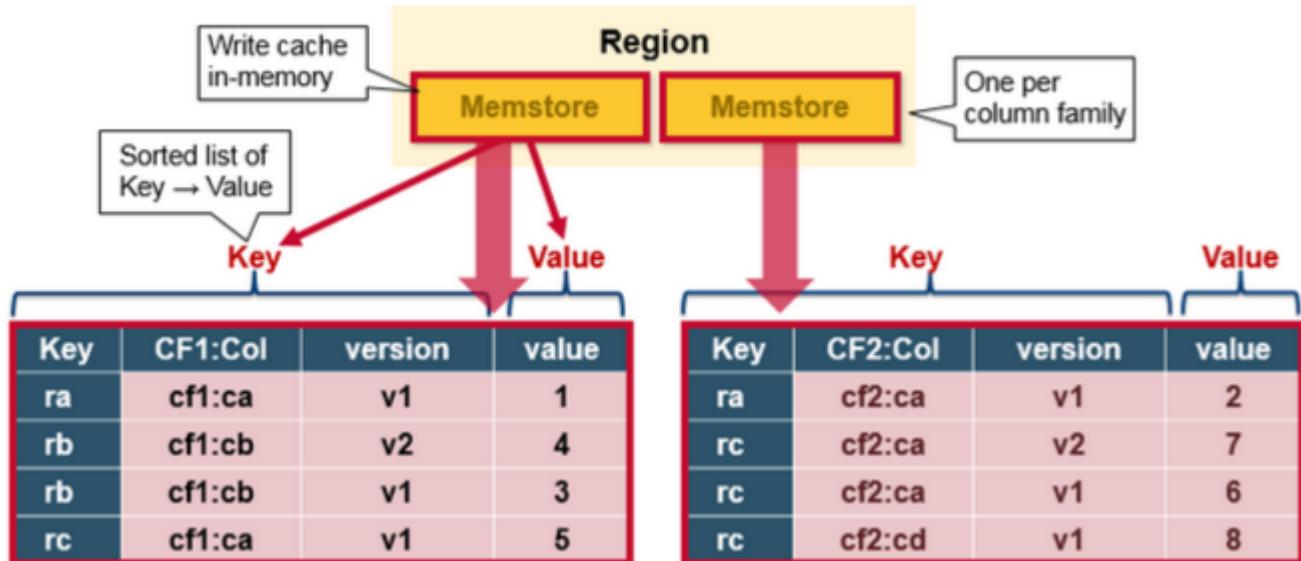
HBase Write Steps (2)

Once the data is written to the WAL, it is placed in the MemStore. Then, the put request acknowledgement returns to the client.



HBase MemStore

The MemStore stores updates in memory as sorted KeyValues, the same as it would be stored in an HFile. There is one MemStore per column family. The updates are sorted per column family.

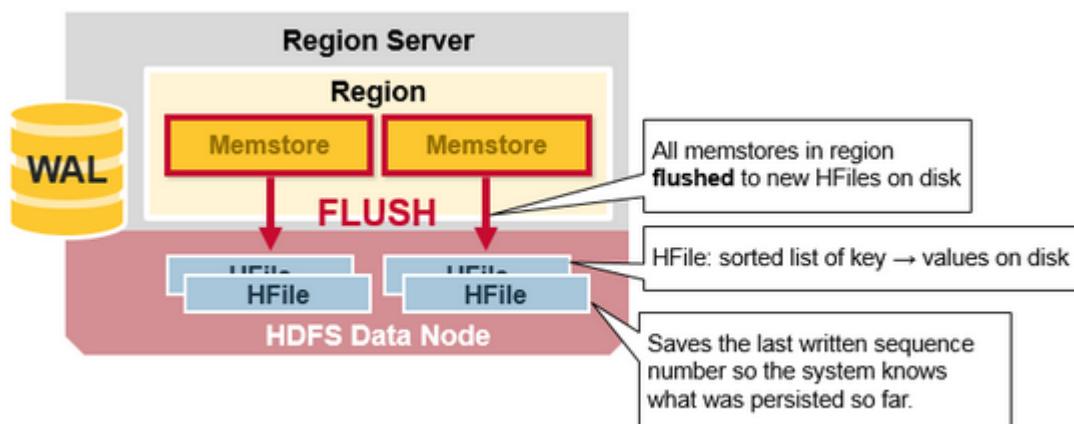


HBase Region Flush

When the MemStore accumulates enough data, the entire sorted set is written to a new HFile in HDFS. HBase uses multiple HFiles per column family, which contain the actual cells, or KeyValue instances. These files are created over time as KeyValue edits sorted in the MemStores are flushed as files to disk.

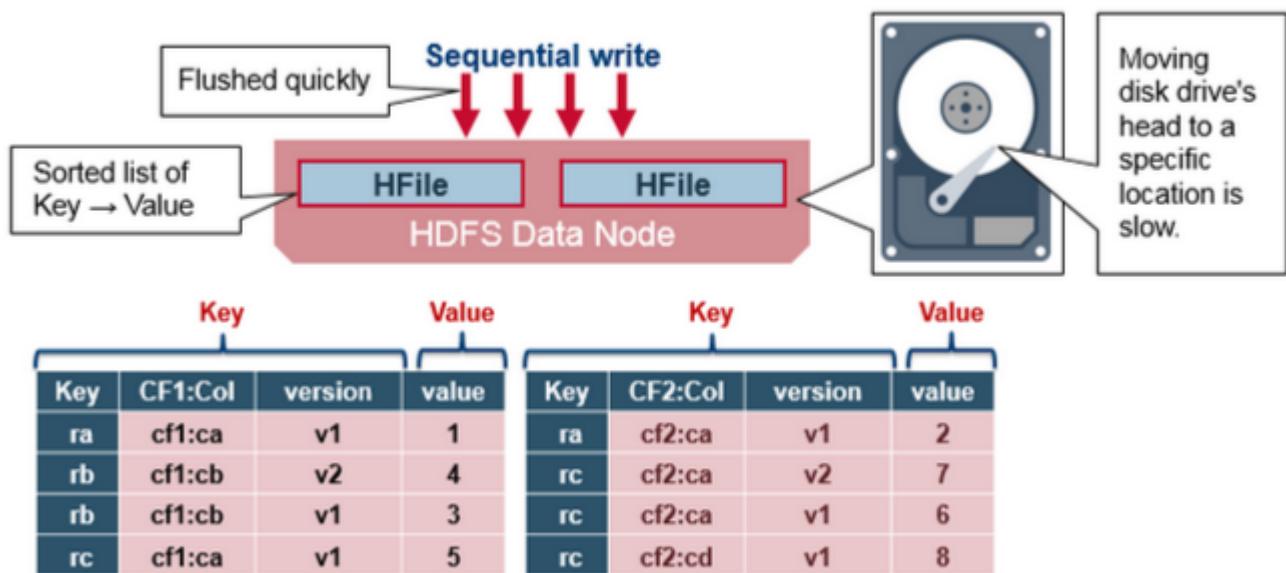
Note that this is one reason why there is a limit to the number of column families in HBase. There is one MemStore per CF; when one is full, they all flush. It also saves the last written sequence number so the system knows what was persisted so far.

The highest sequence number is stored as a meta field in each HFile, to reflect where persisting has ended and where to continue. On region startup, the sequence number is read, and the highest is used as the sequence number for new edits.



HBase HFile

Data is stored in an HFile which contains sorted key/values. When the MemStore accumulates enough data, the entire sorted KeyValue set is written to a new HFile in HDFS. This is a sequential write. It is very fast, as it avoids moving the disk drive head.

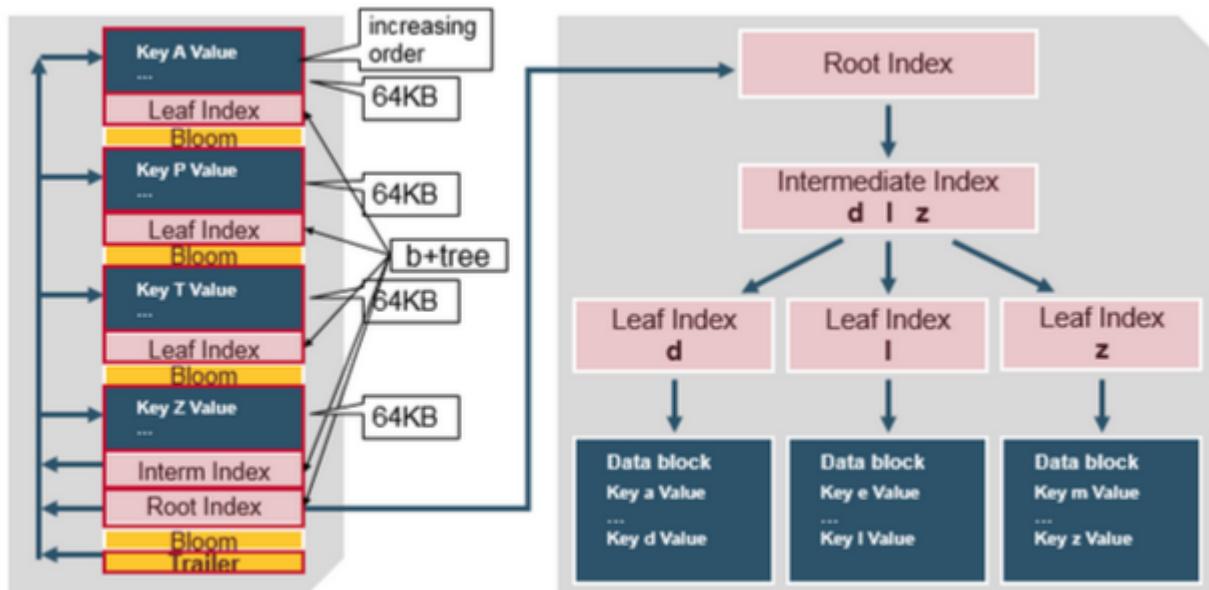


HBase HFile Structure

An HFile contains a multi-layered index which allows HBase to seek to the data without having to read the whole file. The multi-level index is like a b+tree:

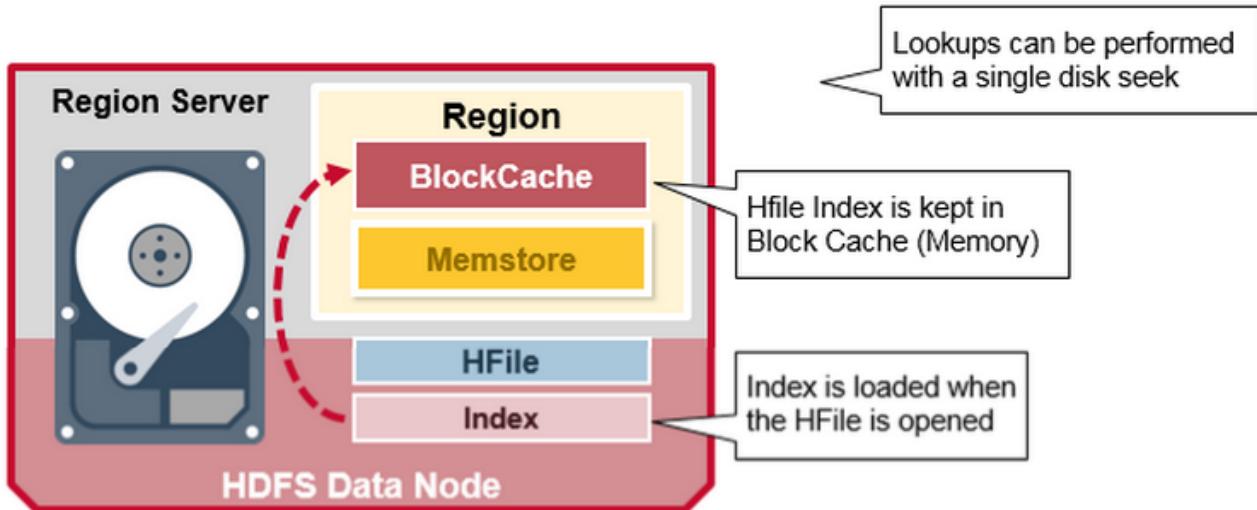
- Key value pairs are stored in increasing order
- Indexes point by row key to the key value data in 64KB “blocks”
- Each block has its own leaf-index
- The last key of each block is put in the intermediate index
- The root index points to the intermediate index

The trailer points to the meta blocks, and is written at the end of persisting the data to the file. The trailer also has information like bloom filters and time range info. Bloom filters help to skip files that do not contain a certain row key. The time range info is useful for skipping the file if it is not in the time range the read is looking for.



HFile Index

The index, which we just discussed, is loaded when the HFile is opened and kept in memory. This allows lookups to be performed with a single disk seek.

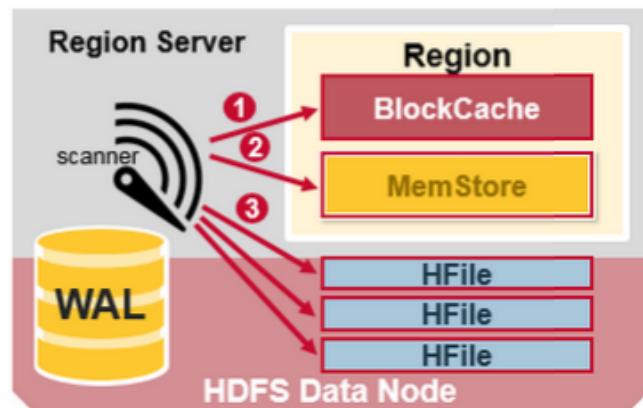


HBase Read Merge

We have seen that the KeyValue cells corresponding to one row can be in multiple places, row cells already persisted are in Hfiles, recently updated cells are in the MemStore, and recently read cells are in the Block cache. So when you read a row, how does the system get the corresponding cells to return? A Read merges Key Values from the block cache, MemStore, and HFiles in the following steps:

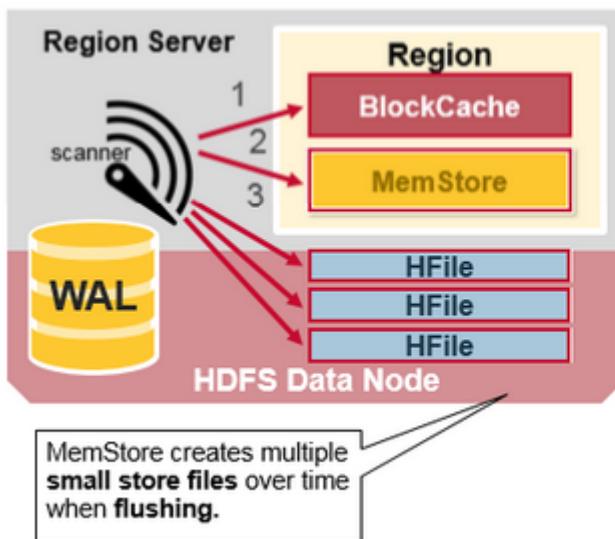
1. First, the scanner looks for the Row cells in the Block cache - the read cache. Recently Read Key Values are cached here, and Least Recently Used are evicted when memory is needed.
2. Next, the scanner looks in the MemStore, the write cache in memory containing the most recent writes.
3. If the scanner does not find all of the row cells in the MemStore and Block Cache, then HBase will use the Block Cache indexes and bloom filters to load HFiles into memory, which may contain the target row cells.

- 1 First the scanner looks for the Row KeyValues in the Block cache
- 2 Next the scanner looks in the MemStore
- 3 If all row cells not in MemStore or blockCache, look in HFiles



HBase Read Merge

As discussed earlier, there may be many HFiles per MemStore, which means for a read, multiple files may have to be examined, which can affect the performance. This is called read amplification.

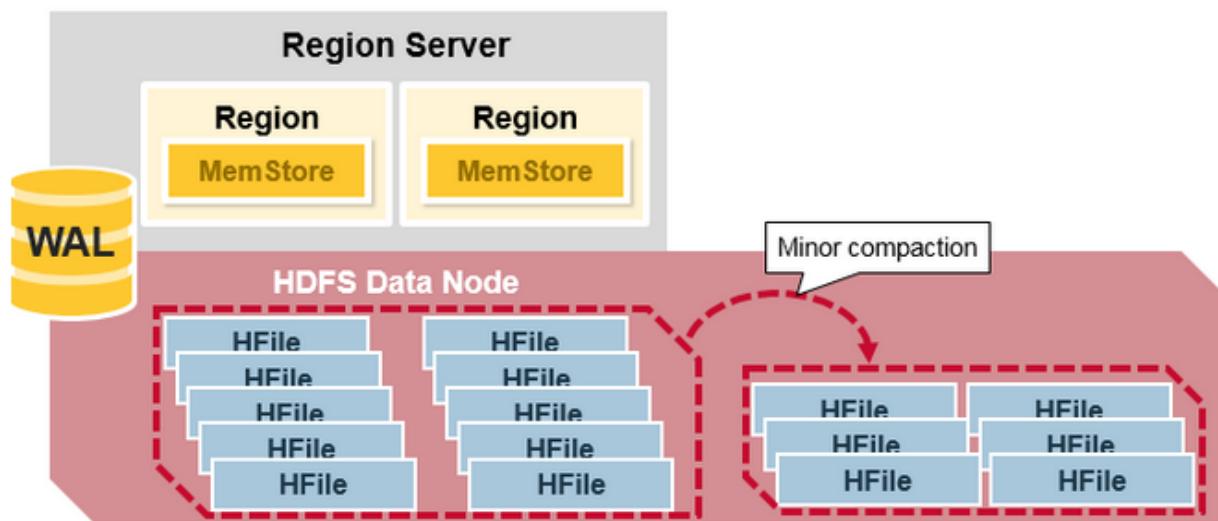


Read Amplification

- multiple files have to be examined

HBase Minor Compaction

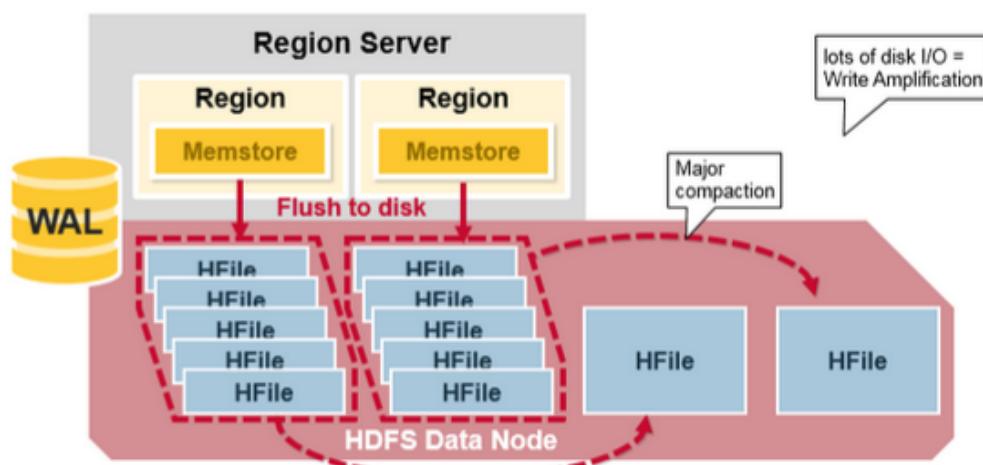
HBase will automatically pick some smaller HFiles and rewrite them into fewer bigger Hfiles. This process is called minor compaction. Minor compaction reduces the number of storage files by rewriting smaller files into fewer but larger ones, performing a merge sort.



HBase Major Compaction

Major compaction merges and rewrites all the HFiles in a region to one HFile per column family, and in the process, drops deleted or expired cells. This improves read performance; however, since major compaction rewrites all of the files, lots of disk I/O and network traffic might occur during the process. This is called write amplification.

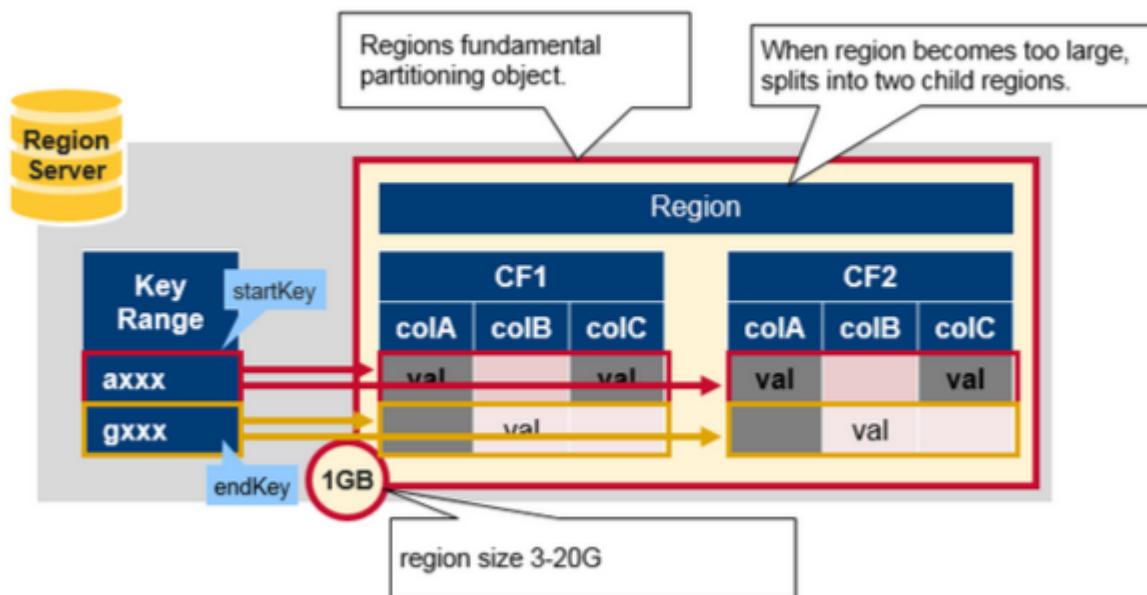
Major compactions can be scheduled to run automatically. Due to write amplification, major compactions are usually scheduled for weekends or evenings. Note that MapR-DB has made improvements and does not need to do compactions. A major compaction also makes any data files that were remote, due to server failure or load balancing, local to the region server.



Region = Contiguous Keys

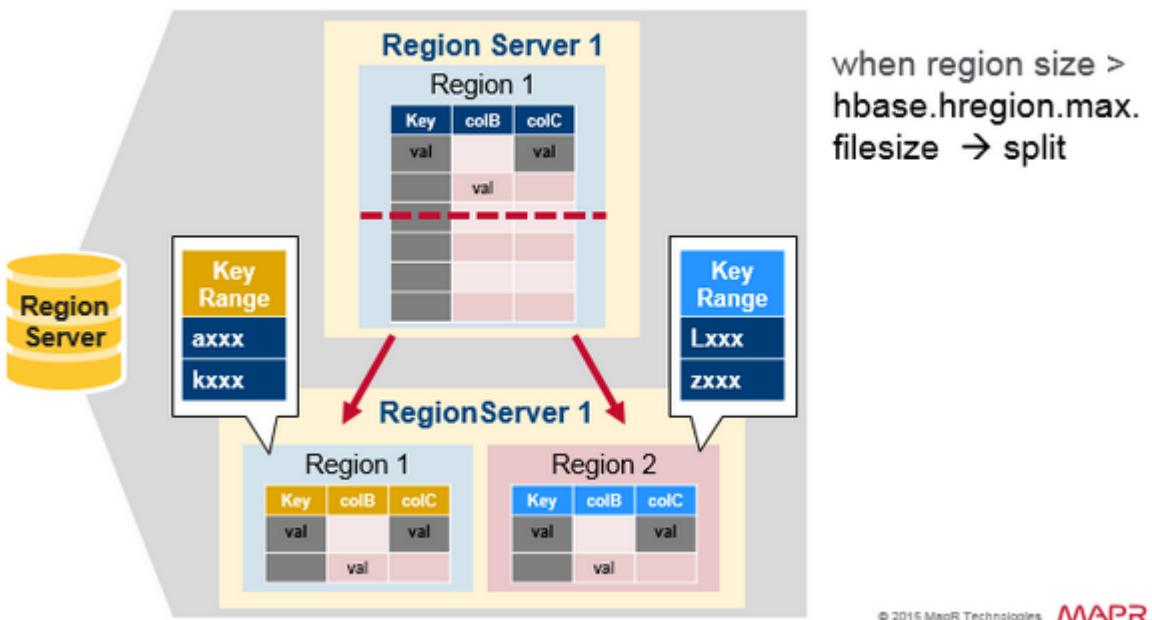
Let's do a quick review of regions:

- A table can be divided horizontally into one or more regions. A region contains a contiguous, sorted range of rows between a start key and an end key
- Each region is 1GB in size (default)
- A region of a table is served to the client by a RegionServer
- A region server can serve about 1,000 regions (which may belong to the same table or different tables)



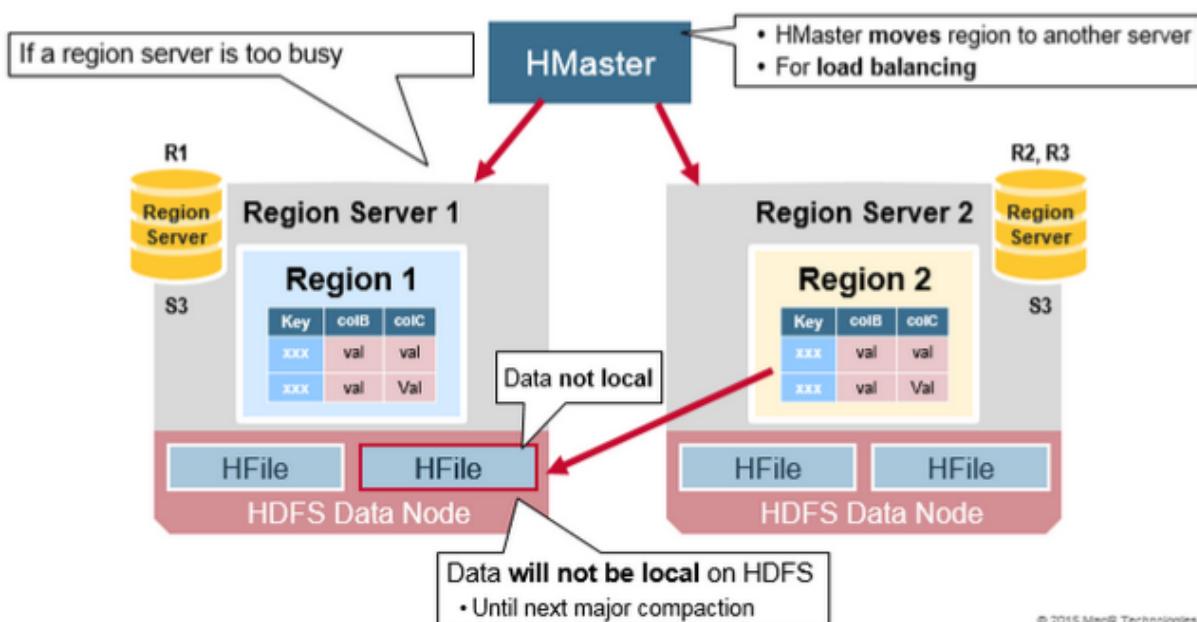
Region Split

Initially there is one region per table. When a region grows too large, it splits into two child regions. Both child regions, representing one-half of the original region, are opened in parallel on the same Region server, and then the split is reported to the HMaster. For load balancing reasons, the HMaster may schedule for new regions to be moved off to other servers.



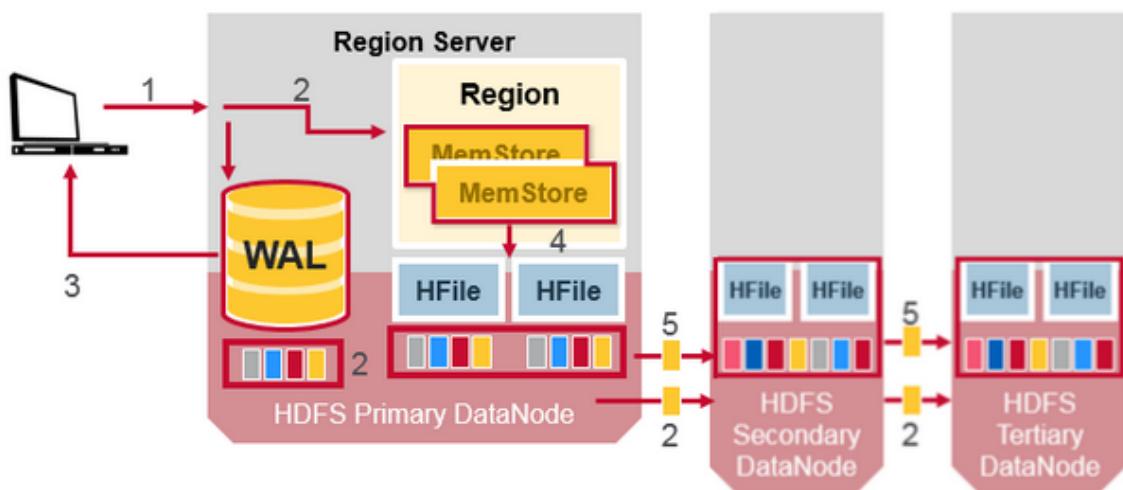
Read Load Balancing

Splitting happens initially on the same region server, but for load balancing reasons, the HMaster may schedule for new regions to be moved off to other servers. This results in the new Region server serving data from a remote HDFS node until a major compaction moves the data files to the Regions server's local node. HBase data is local when it is written, but when a region is moved (for load balancing or recovery), it is not local until major compaction.



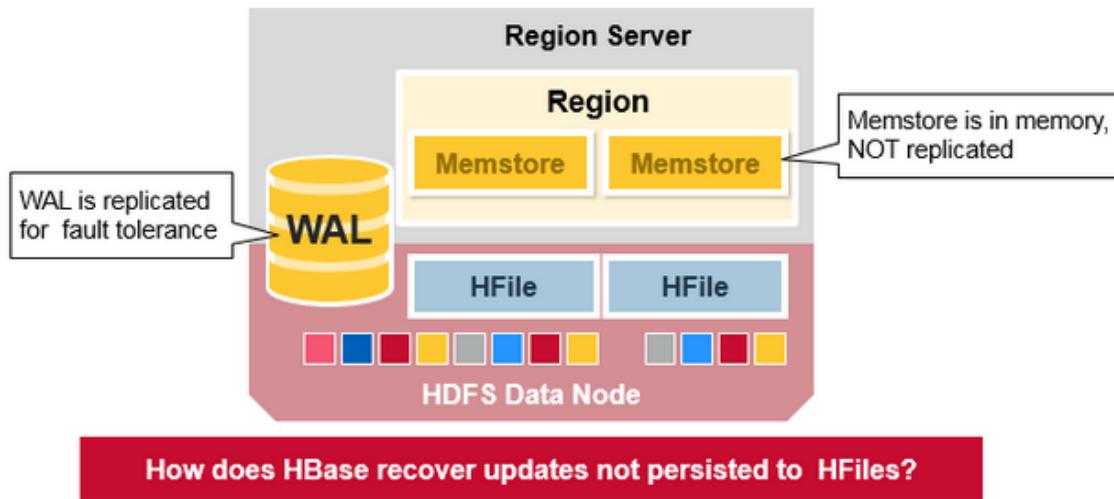
HDFS Data Replication

All writes and Reads are to/from the primary node. HDFS replicates the WAL and HFile blocks. HFile block replication happens automatically. HBase relies on HDFS to provide the data safety as it stores its files. When data is written in HDFS, one copy is written locally, and then it is replicated to a secondary node, and a third copy is written to a tertiary node.



HDFS Data Replication (2)

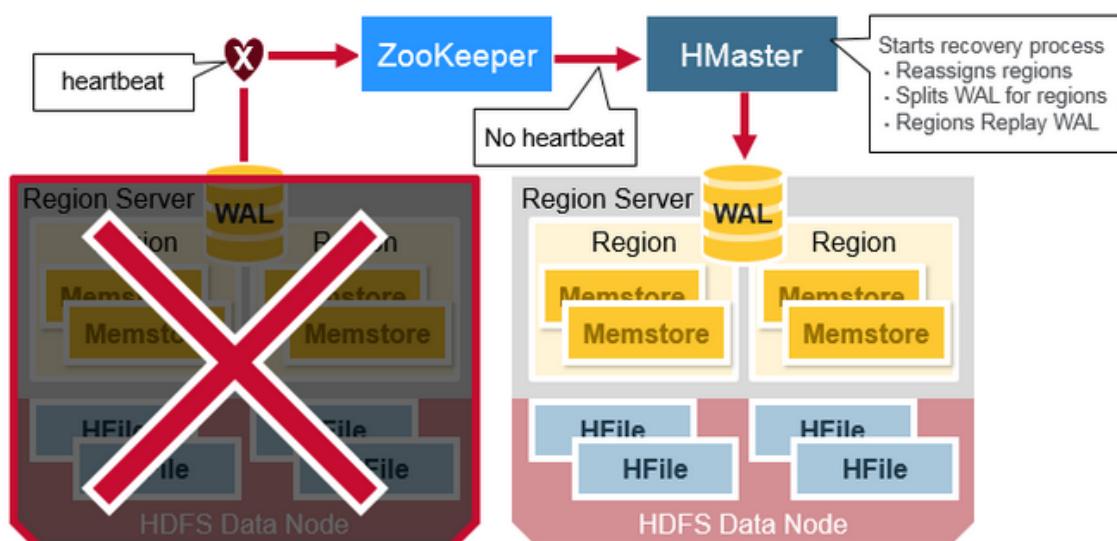
The WAL file and the Hfiles are persisted on disk and replicated, so how does HBase recover the MemStore updates not persisted to HFiles? See the next section for the answer.



HBase Crash Recovery

When a RegionServer fails, Crashed Regions are unavailable until detection and recovery steps have happened. Zookeeper will determine Node failure when it loses region server heart beats. The HMaster will then be notified that the Region Server has failed.

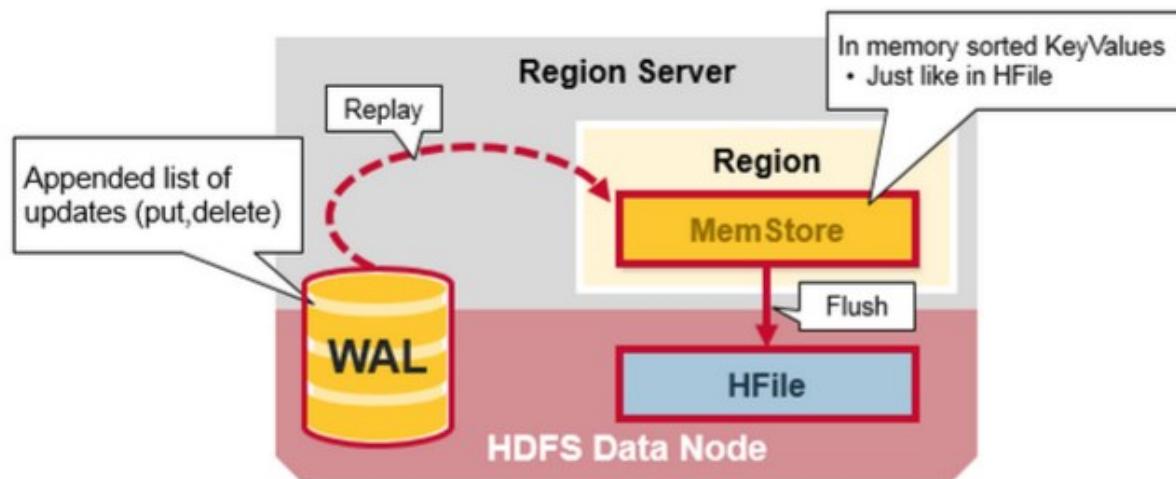
When the HMaster detects that a region server has crashed, the HMaster reassigns the regions from the crashed server to active Region servers. In order to recover the crashed region server's memstore edits that were not flushed to disk. The HMaster splits the WAL belonging to the crashed region server into separate files and stores these file in the new region servers' data nodes. Each Region Server then replays the WAL from the respective split WAL, to rebuild the memstore for that region.



Data Recovery

WAL files contain a list of edits, with one edit representing a single put or delete. Edits are written chronologically, so, for persistence, additions are appended to the end of the WAL file that is stored on disk.

What happens if there is a failure when the data is still in memory and not persisted to an HFile? The WAL is replayed. Replaying a WAL is done by reading the WAL, adding and sorting the contained edits to the current MemStore. At the end, the MemStore is flushed to write changes to an HFile.



Apache HBase Architecture Benefits

HBase provides the following benefits:

- **Strong consistency model**
 - When a write returns, all readers will see same value
- **Scales automatically**
 - Regions split when data grows too large
 - Uses HDFS to spread and replicate data
- **Built-in recovery**
 - Using Write Ahead Log (similar to journaling on file system)
- **Integrated with Hadoop**
 - MapReduce on HBase is straightforward

17. Apache Spark

What is Apache Spark?

Before we learn about Apache Spark or its use cases or how we use it, let's see the reason behind its invention.

- *Exploding Data*

We are aware that today we have huge data being generated everywhere from various sources. This data is either being stored intentionally in a structured way or getting generated by machines. But data is of no use until we mine it and try to do some kind of analysis on it, in order to come up with actions based on the analysis outcomes. The act of gathering and storing information for eventual analysis is ages old but it had never been based on such a large amount of data, which is there today. There is a specific term for such voluminous data i.e. "Big Data" .

Big data is a term that describes the huge volume of data which can be structured and unstructured or semi-structured. But it's not the amount of data which is a concern for the organizations since it is just a storage problem which can be easily addressed by the cheap storage available today. It's what Business get out of the data matters. Big data can be analyzed for insights that lead to better decisions and strategic business moves.

This problem can be solved if we have a framework which not only gives a solution to store all kinds of data (structured, semi-structured or unstructured) but an efficient way of analyzing it according to business needs. One of such framework which is widely used is known as Hadoop. But Hadoop has several limitation (which will be discussed in later sections), because of which Apache Spark was created.

- *Data Manipulation speed*

Now we have a solution for our storage and also an efficient way of analyzing data of any size. That means we can make business decisions after analyzing data. But there is another challenge, which is that the decision based on the analysis insight on a huge data might not be relevant after some time. Any decision is useful if we take the action on time, but doing any analysis on huge data takes time, sometimes more than the deadline on which that action had to be performed. So in such cases such insights are of no use because the deadline for action has passed.

Processing larger scale of data with Hadoop' s processing framework i.e. MapReduce (MR) is far better than our traditional system but still not good enough for organizations to take all its decision on time, because Hadoop operates on batch processing of data leading to high latency.

Several other shortcomings of Hadoop are:

- Adherence to its MapReduce programming model
- Limited programming language API options
- Not a good fit for iterative algorithms like Machine Learning Algorithms
- Pipelining of tasks is not easy

Apache spark was developed as a solution to the above mentioned limitations of Hadoop.

What is Spark

Apache Spark is an open source data processing framework for performing Big data analytics on distributed computing cluster.

Spark was initially started by Matei Zaharia at UC Berkeley's AMPLab in 2009. It was an academic project in UC Berkley. Initially the idea was to build a cluster management tool, which can support different kind of cluster computing systems. The cluster management tool which was built as a result is Mesos. After Mesos was created, developers built a cluster computing framework on top of it, resulting in the creating of Spark.

Spark was meant to target interactive iterative computations like machine learning. In the year 2013, the spark project was passed on to the Apache Software Foundation.

Spark Features

Spark has several advantages when compared to other big data and MapReduce technologies like Hadoop and Storm. Spark is faster than MapReduce and offers low latency due to reduced disk input and output operation. Spark has the capability of in memory computation and operations, which makes the data processing really fast than other MapReduce.

Unlike Hadoop spark maintains the intermediate results in memory rather than writing every intermediate output to disk. This hugely cuts down the execution time of the operation, resulting in faster execution of task, as more as 100X time a standard MapReduce job. Apache Spark can also hold data onto the disk. When data crosses the threshold of the memory storage it is spilled to the disk. This way spark acts as an extension of MapReduce. Spark doesn't execute the tasks immediately but maintains a chain of operations as meta-data of the job called DAG. The action on the DAG happens only when an action operation is called on to the transformation DAG. This process is called as lazy evaluation. This allows optimized execution of the queries on Big Data.

Apache Spark has other features, such as:

- Supports wide variety of operations, compared to Map and Reduce functions.
- Provides concise and consistent APIs in Scala, Java and Python.
- Spark is written in Scala Programming Language and runs in JVM.
- It currently has support in the following programming languages to develop applications in Spark:
 - Scala
 - Java
 - Python
 - R
- Features interactive shell for Scala and Python. This is not available in Java yet.
- It leverages the distributed cluster memory for doing computations for increased speed and data processing.
- Spark enables applications in Hadoop clusters to run up to as much as 100 times faster in memory and 10 times faster even when running in disk.
- It is most suitable for real time decision making with big data.
- It runs on top of existing Hadoop cluster and access Hadoop data store (HDFS), it can also process data stored by HBase structure. It can also run without Hadoop with apache Mesos or alone in standalone mode.
- Apache Spark can be integrated with various data sources like SQL, NoSQL, S3, HDFS, local file system etc.
- Good fit for iterative tasks like Machine Learning (ML) algorithms.

- In addition to Map and Reduce operations, it supports SQL like queries, streaming data, machine learning and data processing in terms of graph.

Hadoop and Apache Spark

Hadoop as a big data processing technology has proven to be the go to solution for processing large data sets. MapReduce is a great solution for computations, which needs one-pass to complete, but not very efficient for use cases that require multi-pass for computations and algorithms. Each stage in the data processing workflow has one Map and one Reduce phase .To leverage MapReduce solution we need to convert our use case into MapReduce pattern. The Job's output data between each step has to be stored in the file system before the next step can begin. Hence, this approach is slow, due to replication & disk Input/output operations. Also, Hadoop ecosystem doesn' t have every component to complete a big data use case. It also requires the integration of several other tools for different big data use cases (like Mahout for Machine Learning and Storm for streaming data processing, Flume for log aggregation).

If you want to do an iterative job, you would have to stitch together a sequence of MapReduce jobs and execute them in sequence. Each of those jobs has high-latency, and each depends upon the completion of previous stages. Spark allows programmers to employ complex, multi-step data pipelines using directed acyclic graph (DAG) pattern. It allows in-memory data sharing across DAGs, so that different jobs can work with the same data without going to disk.

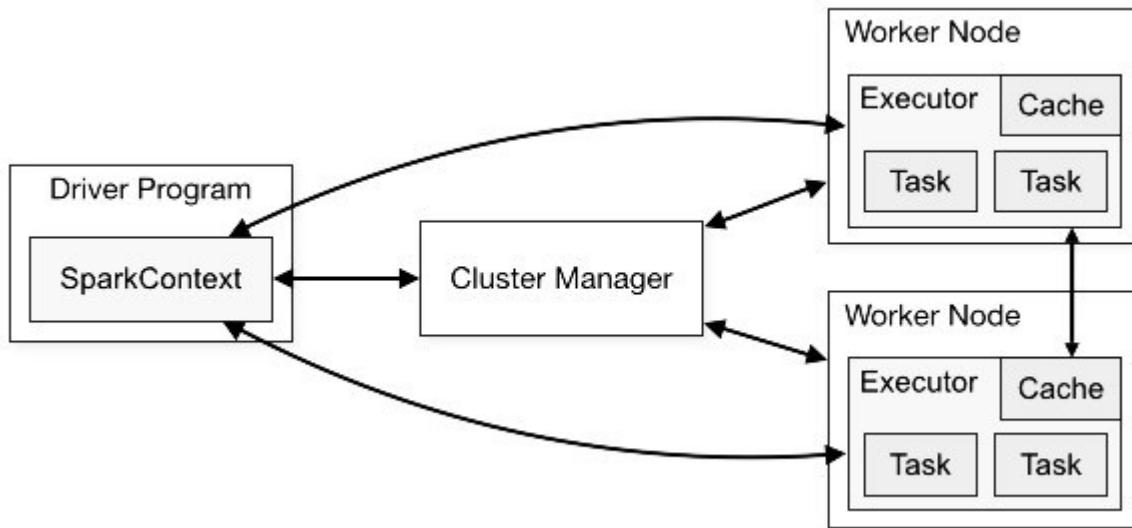
Spark can run on top of Hadoop' s distributed file system Hadoop Distributed File System (HDFS) to leverage the distributed replicated storage. Spark can be used along with MapReduce in the same Hadoop cluster or can be used alone as a processing framework. Apache Spark is an alternative to Hadoop MapReduce rather than a replacement of Hadoop. It' s not intended to replace Hadoop but it can regarded as an extension to it. In many use cases MapReduce and Spark can be

used together where MapReduce job can be used for batch processing and spark can be used for real-time processing.

Apache Spark Components and Architecture

SparkContext is an independent process through which spark application runs over a cluster. It gives the handle to the distributed mechanism/cluster so that you may use the resources of the distributed machines in your job. Your application program which will use SparkContext object would be known as driver program. Specifically, to run on a cluster, the SparkContext connects to several types of cluster managers (like Spark' s own standalone cluster manager, apache Mesos or Hadoop's YARN), which allocate resources across applications. Once connected, Spark takes over executors on distributed nodes in the cluster, which are processes in the distributed nodes that run computations and store data for your application. Next, it sends your application code to the executors through SparkContext. Finally tasks are sent to the executors to run and complete it.

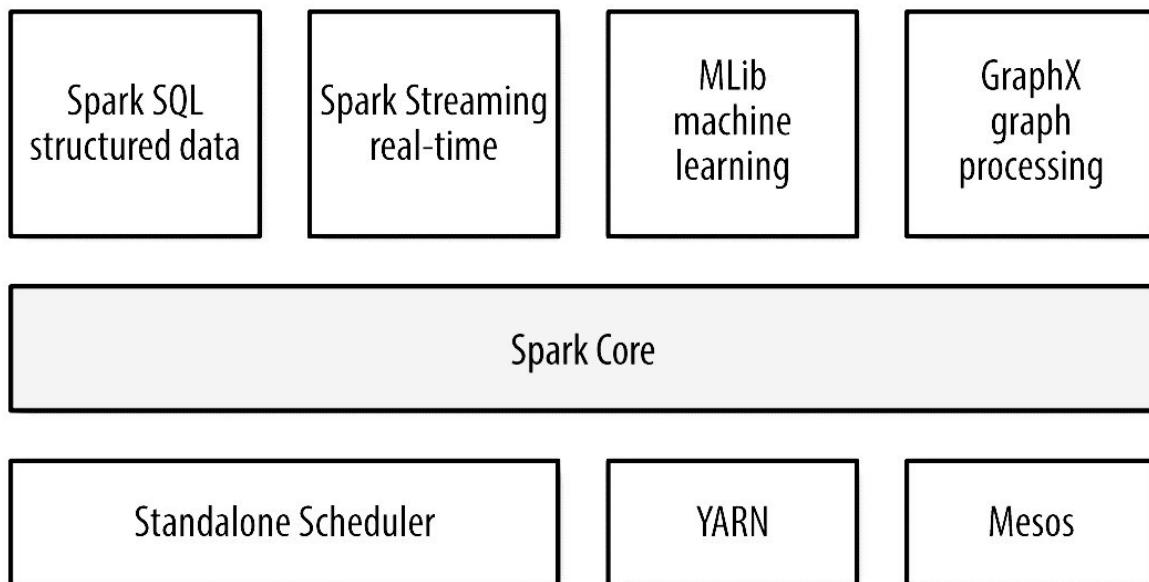
Cluster overview



Following are most important takeaways of the architecture:

- Each application gets its own executor processes, which remains in memory up to the duration of the complete application and run tasks in multiple threads. This means each application is independent from the other, on both the scheduling side since each driver schedules its own tasks and executor side as tasks from different applications run in different JVMs.
- Spark is independent of cluster managers that implies, it can be coupled with any cluster manager and then leverage that cluster.
- Because the driver schedules tasks on the cluster, it should be run as close to the worker nodes as possible.

Spark Eco-system components



Spark Core

Spark Core is the base of an overall spark project. It is responsible for distributed task dispatching, parallelism, scheduling, and basic I/O functionalities. All the basic functionality of spark core are exposed through an API (for Java, Python, Scala, and R) centered on the RDD abstraction. A ‘driver’ program starts parallel operations such as map, filter or reduce on any RDD by passing a function to SparkCore, which further schedules the function’s execution in parallel on the cluster.

Other than Spark Core API, there are additional useful and powerful libraries that are part of the Spark ecosystem and adds powerful capabilities in Big Data analytics and Machine Learning areas. These libraries include:

- **Spark Streaming**

Spark Streaming is a useful addition to the core Spark API. It enables high-throughput, fault-tolerant stream processing of live data streams. It is used for processing real-time streaming data. This is based on micro batch style of computing and processing. The fundamental stream unit is DStream. DStream is basically a series of RDDs, to process the real-time data.



Spark SQL, DataFrames and Datasets:

- ***SQL***

Spark SQL exposes spark APIs to run SQL query like computation on large data. A spark user can perform ad-hoc query and perform near real time ETL on a different types of data like (like JSON, Parquet, Database).

- ***DataFrames***

A DataFrame can be considered as a distributed set of data which has been organized into many named columns. It can be compared with a relational table, CSV file or a data frame in R or Python. The DataFrame functionality is made available as API in Scala, Java, Python, and R.

- ***Dataset***

A Dataset is a new addition in the list of spark libraries. It is an experimental interface added in Spark 1.6 that tries to provide the benefits of RDDs with the benefits of Spark SQL' s optimized execution engine.

- ***Spark MLlib And ML***

MLlib is collective bunch few handy and useful machine learning algorithms and data cleaning and processing approaches which includes classification, clustering, regression, feature extraction, dimensionality reduction, etc. as well as underlying optimization primitives like SGD and BFGS.

- ***Spark GraphX***

GraphX is the Spark API for graphs and graph-parallel computation. GraphX enhances the Spark RDD by introducing the Resilient Distributed Property Graph.

A RDD property graph is a directed multi-graph with properties attached with each of its vertex and edge. GraphX has a set of basic operators (like subgraph, joinVertices, aggregateMessages, etc.).Along with operators it has an optimized variant of the Pregel API. GraphX is still under development and many developers are working towards simplification of graph related tasks.

18. Hadoop Administration

Planning deployment of a Hadoop Cluster involves analysing and deciding on a number of crucial aspects like version and distribution, Size of Cluster, Hardware Configurations, Ports and Properties, Job Configurations etc. In this chapter we will have a brief look into these aspects from a Hadoop Administrator' s point of view.

Capacity Planning

The very first decision will remain on how big your Hadoop Cluster should be. This decision will be influenced by the amount of data that your company handles.

For example:

Your company collects around 2 TB of data every month. With default replication factor, you would be dumping around 6 TB of data every month in your Hadoop Cluster. Additionally not all storage space is allotted to the HDFS. So every month your company will be needing approximately 8 TB of storage.

Considering you buy servers with storage capacity of 4 TB, then you will need around 2 servers in a month which approximates to a total of around 24 Data Nodes in a year. Additionally your Cluster will have the master servers.

Depending on the company' s budget, you can either have the Name Node, Job Tracker and Secondary Name Node on separate servers, which makes a total of $24 + 3 = 27$ Nodes or you may opt to have the Name Node and Job Tracker on same servers and the Secondary Name Node on separate servers, which makes a total of $24 + 2 = 26$ Nodes.

Hardware Selection

There are generally 2 types of nodes in a Hadoop Cluster viz. Master Nodes and Slave Nodes.

Master nodes are crucial in the performance of the Hadoop Cluster and are the single point of failures. Name Node going down will leave the entire cluster inaccessible and if the Job Tracker goes down you will have to resubmit all the running and queued jobs. So make sure that you purchase best in class servers with at least the following specifications

- Quad Core CPU
- 32 GB RAM
- Raided Hard Disk Drives
- Dual Power Supplies
- Dual Ethernet Cards

Slave Nodes are the actual working Nodes. In order to achieve higher rate of parallel processing, it is suggested to trust in quantity instead of quality. Means if you can either buy 2 best quality servers or buy 5 less efficient servers, then it is always better to go with more number of nodes.

Typically the hardware configurations for a slave Node may fall under the ratio of 1:2:6-8. Means if you have 1 TB of Storage, then you may have atleast 2 cores of CPU and atleast 6 to 8 GB of RAM. The number of simultaneous Map and Reduce tasks is equal to 1.5 times of the number of CPU cores in the slave server. Means if you have around 8 cores of CPU then that machine can perform around 12 simultaneous tasks.

OS Selection

Hadoop can be installed on any Linux Operating system (Windows only if Hortonworks is used) like RHEL, Fedora, Ubuntu, Centos etc. Normally in a production deployment, RHEL is used on the master servers and Centos is used on the slave nodes. Feel free to choose an OS that your company is comfortable administering.

Java Selection

Being written in JAVA, this becomes one of the major factors to think upon. Depending on how you are going to write, run and execute the programs, you can choose between JDK and JRE. Hadoop supports both. As Hadoop is way too complex in itself, it is recommended to choose the most stable version of JAVA, to avoid additional complexity out of the bugs in unstable versions. You can download JAVA from Oracle or Sun's official sites.

Hadoop Selection

Selecting the version and distribution of Hadoop depends on a number of factors but what is more important is being able to map your requirements with the available features and investing in only what is required.

Hadoop History

Deploying the recent most stable version is what any company would like to do. This will surely give a lot of features but initially it is better to select a stable version based on requirement so that you don't increase the complexity with unnecessary implementations.

Hadoop has grown from a fundamental HDFS with Map Reduce to a large Ecosystem over a period of 10 years now. The below tables will help you in understanding the different releases of Hadoop

Version	Features
4 September, 2007 : release 0.14.1	Better checksums in HDFS. Introduced C++ API for MapReduce. Eclipse Plugin, including HDFS browsing, job monitoring, etc.
29 October 2007: release 0.15.0	contains the first working version of Hbase
24 February, 2009: release 0.19.1	Issues related to Data Loss were resolved
23 August, 2010: release 0.21.0	Stable version for Production deployments
11 May, 2011: release 0.20.203.0	First version to be successfully deployed on 4,500 machine cluster
5 Sep, 2011: release 0.20.204.0	RPMs and DEBs were introduced for the first time

Version	Features
11 Nov, 2011: release 0.23.0	Contains HDFS Federation and YARN
10 December, 2011: release 0.22.0	<p>Some new features were added like</p> <ol style="list-style-type: none"> 1. HBase support with hflush and hsync. 2. New implementation of file append. 3. BackupNode and CheckpointNode. 4. Hierarchical job queues. 5. Job limits per queue/pool. 6. Dynamically stop/start job queues. 7. Advances in new mapreduce API: Input/Output formats, ChainMapper/Reducer. 8. TaskTracker blacklisting. 9. DistributedCache sharing.
27 December, 2011: release 1.0.0	<p>Provides major features likes:</p> <ol style="list-style-type: none"> 1. Security 2. HBase (append/hsynch/hflush, and security) 3. Webhdfs (with full support for security) 4. Performance enhanced access to local files for HBase
10 Mar, 2012: Release 1.0.1 available	Provided better compatibility with Ganglia, HBase, and Sqoop
9 October, 2012: Release 2.0.2-alpha	<p>This delivers significant enhancements to HDFS HA.</p> <p>Also it has a significantly more stable version of YARN</p>

Version	Features
14 February, 2013: Release 2.0.3-alpha	<p>This version introduced improved features like:</p> <ol style="list-style-type: none"> 1. QJM for HDFS HA for NameNode 2. Multi-resource scheduling (CPU and memory) for YARN 3. YARN ResourceManager Restart 4. Significant stability at scale for YARN (over 30,000 nodes and 14 million)
13 May, 2013: Release 1.2.0	<p>This release delivers over 200 enhancements and bug-fixes.</p> <p>Major enhancements include:</p> <ol style="list-style-type: none"> 1. Web services for JobTracker 2. WebHDFS enhancements 3. More robust Namenode in case of edit log corruption 4. Add NodeGroups level to NetworkTopology
25 August, 2013: Release 2.1.0-beta	<p>This release has significant changes like:</p> <ol style="list-style-type: none"> 1. Improved HDFS Snapshots 2. Support for running Hadoop on Microsoft Windows 3. YARN API stabilization 4. Binary Compatibility for MapReduce applications built on hadoop-1.x 5. Substantial amount of integration testing with rest of projects in the ecosystem

15 October, 2013:
Release 2.2.0

Apache Hadoop 2.2.0 is the GA release of Apache Hadoop 2.x. This release has a number of significant highlights compared to Hadoop 1.x:

1. YARN - A general purpose resource management system for Hadoop to allow MapReduce and other other data processing frameworks and services
2. High Availability for HDFS
3. HDFS Federation
4. HDFS Snapshots
5. NFSv3 access to data in HDFS
6. Support for running Hadoop on Microsoft Windows
7. Binary Compatibility for MapReduce applications built on hadoop-1.x
8. Substantial amount of integration testing with rest of projects in the ecosystem

<p>20 February, 2014: Release 2.3.0</p>	<p>Support for Heterogeneous Storage hierarchy in HDFS. In-memory cache for HDFS data with centralized administration and management. Simplified distribution of MapReduce binaries via HDFS in YARN Distributed Cache.</p>
<p>18 November, 2014: Release 2.6.0</p>	<p>Apache Hadoop 2.6.0 contains a number of significant enhancements such as:</p> <ul style="list-style-type: none"> 1. Hadoop Common <ul style="list-style-type: none"> a. Key management server (beta) b. Credential provider (beta) 2. Hadoop HDFS <ul style="list-style-type: none"> a. Support for Archival Storage b. Transparent data at rest encryption (beta) c. Operating secure DataNode without requiring root access d. Hot swap drive: support add/remove data node volumes without restarting data node (beta) 3. Hadoop YARN <ul style="list-style-type: none"> a. Support for long running services in YARN <ul style="list-style-type: none"> i. Service Registry for applications b. Support for rolling upgrades <ul style="list-style-type: none"> i. Work-preserving restarts of ResourceManager ii. Container-preserving restart of NodeManager c. Support node labels during scheduling d. Support for time-based resource reservations in Capacity Scheduler (beta)

	<p>Major changes include</p> <ol style="list-style-type: none">1. HADOOP<ol style="list-style-type: none">a. Move to JDK8+b. Classpath isolation on by defaultc. Shell script rewrite2. HDFS<ol style="list-style-type: none">a. Removal of hftp in favor of webhdfsb. Support for Erasure Codes in HDFS3. YARN MAPREDUCE<ol style="list-style-type: none">a. Derive heap size or mapreduce.*.memory.mb automatically
hadoop-3.0	

Evolution of Hadoop Ecosystem

Developments did took place not only within Hadoop but also around it.

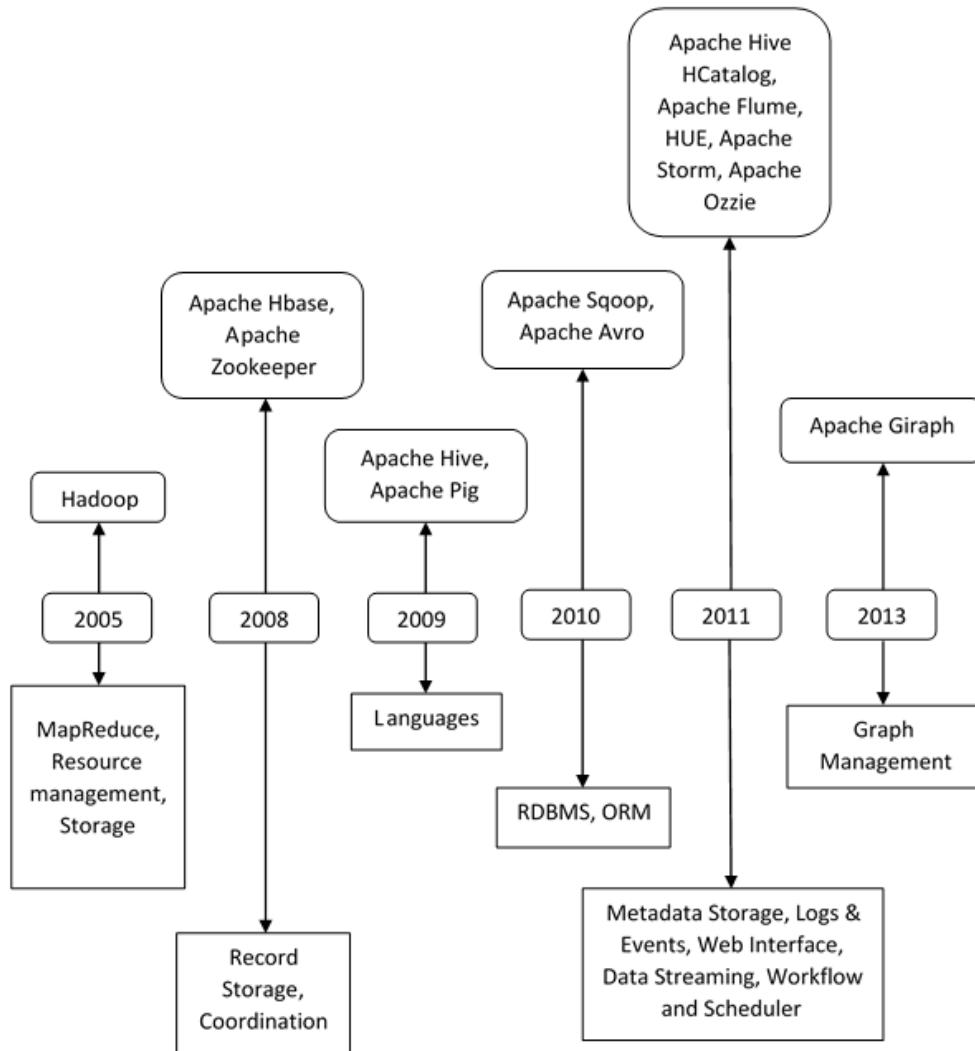


Fig 5: Hadoop Evolution

All these Ecosystem tools were to be installed and managed individually. Because of this, companies working on large applications with Hadoop started to face difficulties giving rise to Hadoop Distributions.

Hadoop Distributions

Hadoop Distributions include organisations that provide packaged Hadoop Ecosystem, support etc. These organisations have created a platform where most of the popular Hadoop Ecosystem tools are integrated together along with the core functionalities to provide a single interface for installing and managing the vast range of Ecosystem tools. The distributions have not only made deployment of large applications around Hadoop easy but also provide immediate support over bugs and problems to companies that use their platforms.

Out of the many distributions available Cloudera, Hortonworks and MapR are the most popular ones. All these distributions have their own characteristics in the number and type of tools bundled in the package, the support, the new releases etc. Based on the requirements of the company, these characteristics influence the decision of selecting one of these distributions.

Cloudera's Distribution of Hadoop

Cloudera Inc. is an American-based software company that provides Apache Hadoop-based software, support and services, and training to business customers.

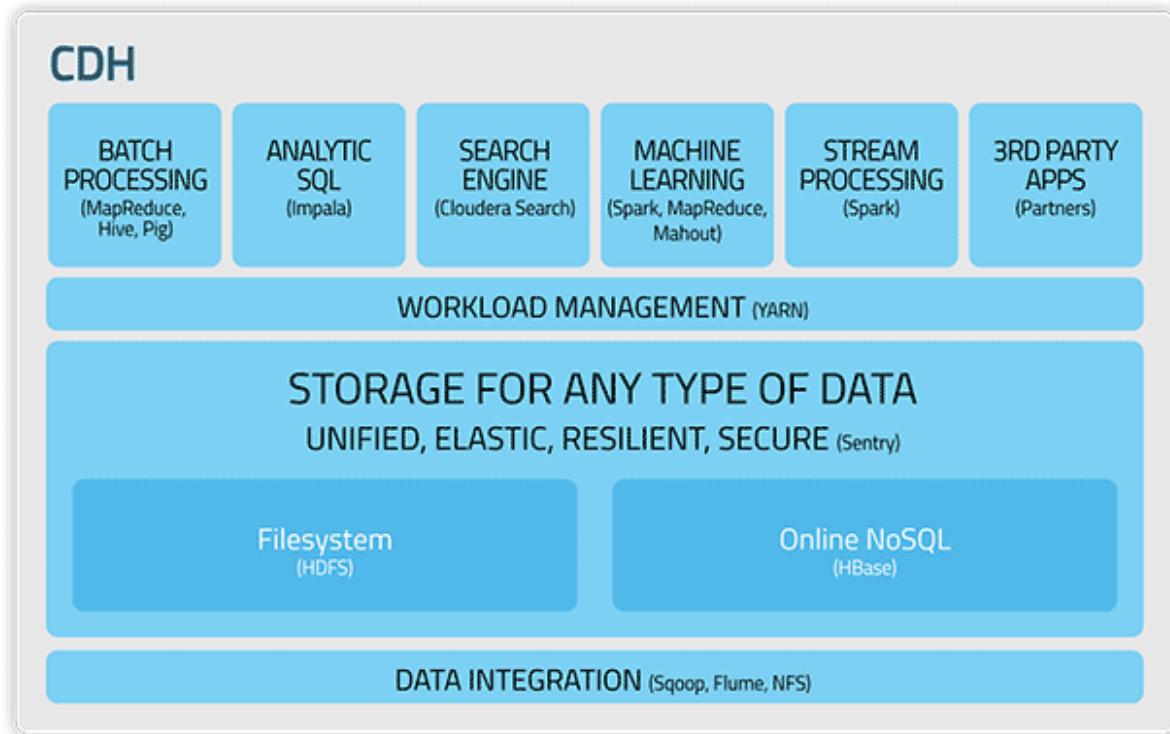
Cloudera's open-source Apache Hadoop distribution, CDH (Cloudera Distribution Including Apache Hadoop), targets enterprise-class deployments of that technology. Cloudera says that more than 50% of its engineering output is donated upstream to the various Apache-licensed open source projects (Apache Hive, Apache Avro, Apache Hbase, and so on) that combine to form the Hadoop platform. Cloudera is also a sponsor of the Apache Software Foundation.

Cloudera offers software, services and support in three different bundles:

Cloudera Enterprise includes CDH and an annual subscription license (per node) to Cloudera Manager and technical support. It comes in three editions: Basic, Flex, and Data Hub.

Cloudera Express includes CDH and a version of Cloudera Manager lacking enterprise features such as rolling upgrades and backup/disaster recovery.

CDH may be downloaded from Cloudera's website at no charge, but with no technical support nor Cloudera Manager.



Hortonworks Data Platform

Hortonworks is a business computer software company based in Santa Clara, California. The company focuses on the development and support of Apache Hadoop, a framework that allows for the distributed processing of large data sets across clusters of computers.

Hortonworks employs contributors to the open source software project Apache Hadoop. The company was named after Horton the Elephant of the *Horton Hears a Who!* Book.

Hortonworks is a sponsor of the Apache Software Foundation.

Hortonworks' product named Hortonworks Data Platform (HDP) includes Apache Hadoop and is used for storing, processing, and analysing large volumes of data. The platform is designed to deal with data from many sources and formats. The platform includes various Apache Hadoop projects including the Hadoop Distributed File System, MapReduce, Pig, Hive, Hbase and Zookeeper and additional components.

In October 2011 Hortonworks announced Microsoft would collaborate on a Hadoop distribution for Microsoft Azure and Windows Server. On February 25,

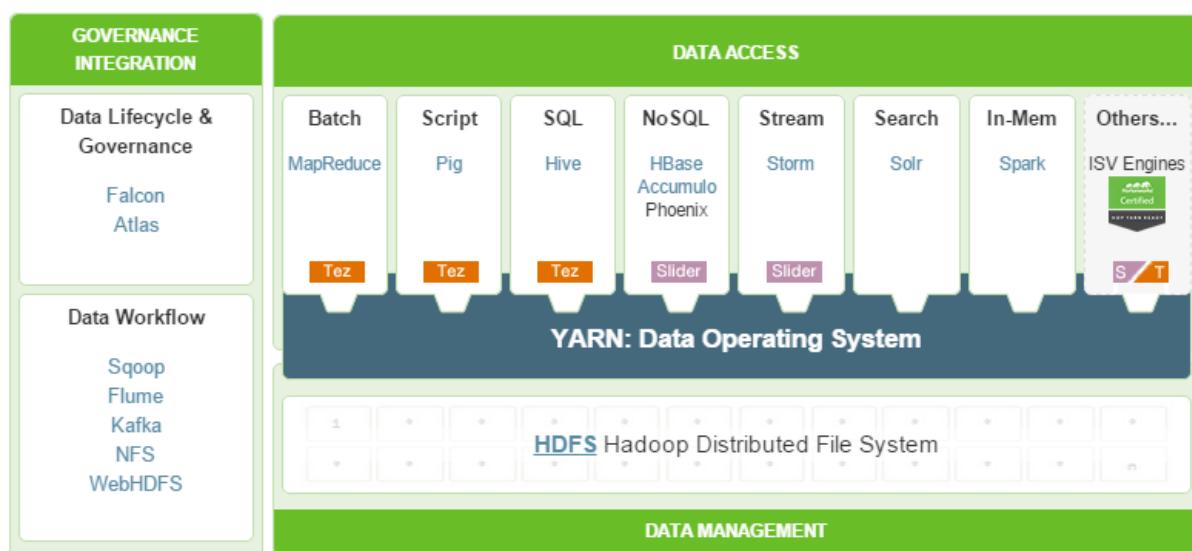


Fig 7: HDP Governance & Data Access

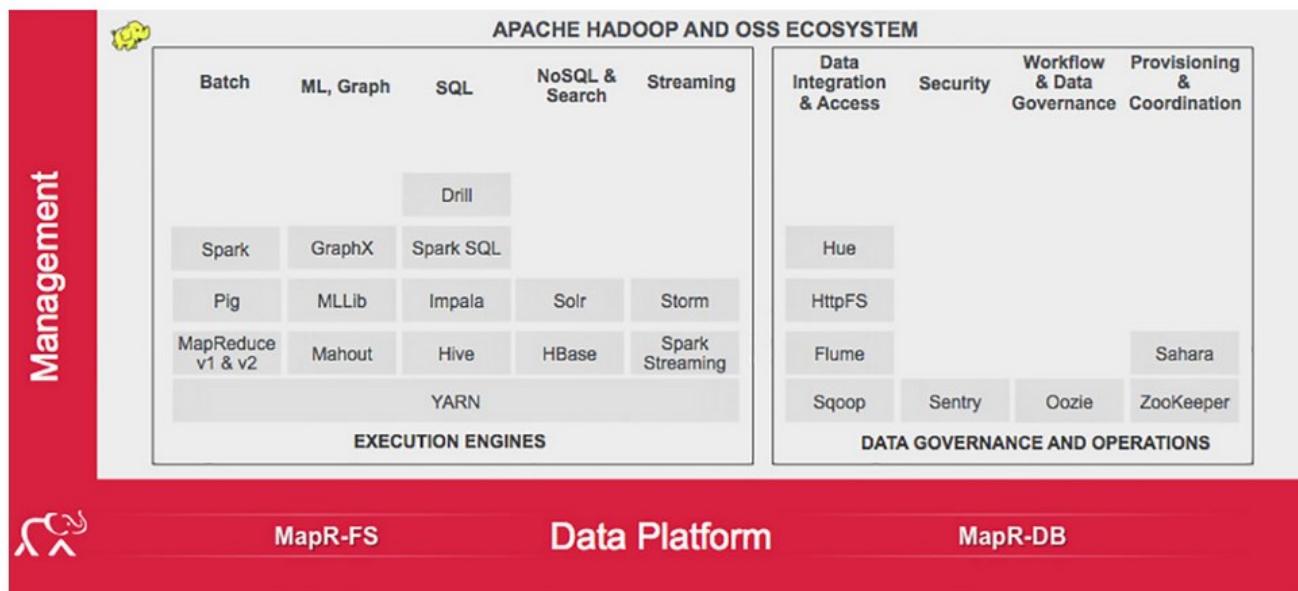
2013, Hortonworks announced availability of a beta version of the Hortonworks Data Platform for Windows.



MapR

MapR is a San Jose, California-based enterprise software company that develops and sells Apache Hadoop-derived software. The company contributes to Apache Hadoop projects like Hbase, Pig (programming language), Apache Hive, and Apache ZooKeeper. MapR's Apache Hadoop distribution claims to provide full data protection, no single points of failure, improved performance, and dramatic ease of use advantages. MapR entered a technology licensing agreement with EMC Corporation on 25 May 2011, supporting an EMC-specific distribution of Apache Hadoop. MapR was selected by Amazon to provide an upgraded version of Amazon's Elastic Map Reduce (EMR) service. MapR has also been selected by Google as a technology partner. MapR was able to break the minute sort speed record on Google's compute platform.

MapR provides three versions of their product known as M3, M5 and M7. M3 is a free version of the M5 product with degraded availability features. M7 is like M5, but adds a purpose built rewrite of Hbase that implements the Hbase API directly in the file-system layer.



Other Hadoop Distributions include

- [Datameer](#)

Datameer Analytics Solution (DAS) is a Hadoop-based solution for big data analytics that includes data source integration, storage, an analytics engine and visualization.

- [IBM](#)

IBM InfoSphere BigInsights brings the power of Apache Hadoop to the enterprise. BigInsights Enterprise Edition builds on Apache Hadoop with capabilities to withstand the demands of an enterprise including:

Advanced Text Analytics

Performance Optimizations

Workload Management & Scheduling

Professional-Grade Visualization & Developer Tooling

Enterprise Integration & Security

- [Amazon Web Services](#)

Amazon offers a version of Apache Hadoop on their EC2 infrastructure, sold as Amazon Elastic MapReduce.

- [Apache Bigtop](#)

Apache Bigtop is a project for the development of packaging and tests of the Apache Hadoop ecosystem. This includes testing at various levels (packaging, platform, runtime, upgrade, etc...) developed by a community with a focus on the system as a whole, rather than individual projects.

- [Cloudspace](#)

Cloudspace is a web technology consulting company, since 1996. Cloudspace uses Apache Hadoop to scale client and internal projects on Amazon's EC2 and bare metal architectures.

[Cascading](#)

- Cascading is a popular feature-rich API for defining and executing complex and fault tolerant data processing workflows on a Apache Hadoop cluster.
Cascading 2.0 is Apache-licensed.

- [DataStax](#)

DataStax provides a product which fully integrates Apache Hadoop with Apache Cassandra and Apache Solr in its DataStax Enterprise platform. DataStaxEnterprise is completely free to use for development environments with no restrictions.

- [Pentaho – Open Source Business Intelligence](#)

Pentaho provides a complete, end-to-end open-source BI alternative to proprietary offerings like Oracle, SAP and IBM.

- [VMware](#)

Initiate Open Source project and product to enable easily and efficiently deploy and use Hadoop on virtual infrastructure. HVE - Hadoop Virtual Extensions to make Hadoop virtualization-aware. Serengeti - enable the rapid deployment of an Apache Hadoop cluster

Hadoop Installation Modes

Hadoop can be installed and configured in 2 modes

- **Pseudo Distributed Mode**

In this mode all the Hadoop services viz. masters as well as slaves are configured and started in one single machine. This is also called as a Single Node Hadoop Cluster. This mode is useful for testing and learning purpose but won't help you in dealing with Big Data.

- **Distributed Mode**

This is the mode for which Hadoop was actually built. This is also called as a Multi Node Cluster. This Shared Nothing architecture makes sure that you master and slaves are installed and configured on different servers. Normally Name Node and Job Tracker are installed on one server and Data Node and Task Trackers are installed on the slave servers.

Important configuration files

In the Hadoop installation, the most important configuration files are

- [core-site.xml](#)

In this file you will specify the changes that deal with the cluster information.

For example the details of the master node will be specified in this file. The configuration of client side ecosystem tools can also be specified in this file.

- [hdfs-site.xml](#)

In this file you will specify configurations related to HDFS data storage and retrieval. Configurations like the Replication Factor, Block Size etc. can be modified from this file.

- [mapred-site.xml](#)

In this file you can play with the default configurations for executing the Map Reduce programs. Like if you want to have 4 Mappers for all your operations then you can specify it from this file.

The settings in these files are called as the configurations on the cluster and has the least priority if the same configurations are specified from elsewhere. There are 3 ways in which you can specify or modify the configurations.

- From the Program (**highest priority**)
- From the Client (**second highest priority**)
- From the Cluster (**lowest priority**)

After analysing all these aspects in the most appropriate way, you can possibly end up setting a cluster with a high cost/performance ratio.

19. Labs Section

Lab 01: Configure Hadoop in Pseudo Distributed Mode

1. Step 1: Verify if the package is installed properly
2. rpm -ql hadoop-0.20-conf-pseudo
- 3.
4. Expected Output
5. /etc/hadoop/conf.pseudo.mr1
6. /etc/hadoop/conf.pseudo.mr1/README
7. /etc/hadoop/conf.pseudo.mr1/core-site.xml
8. /etc/hadoop/conf.pseudo.mr1/hadoop-metrics.properties
9. /etc/hadoop/conf.pseudo.mr1/hdfs-site.xml
10. /etc/hadoop/conf.pseudo.mr1/log4j.properties
11. /etc/hadoop/conf.pseudo.mr1/mapred-site.xml
- 12.
- 13.
14. Step 2: Format the NameNode Service
15. sudo -u hdfs hdfs namenode -format
- 16.
17. Expected Output
18. 14/12/03 15:35:54 INFO namenode.NameNode: STARTUP_MSG:
19. ****
20. STARTUP_MSG: Starting NameNode
21. STARTUP_MSG: host = localhost.localdomain/127.0.0.1
22. STARTUP_MSG: args = [-format]
23. STARTUP_MSG: version = 2.0.0-cdh4.4.0
24. STARTUP_MSG: classpath = /etc/hadoop/conf:/usr/lib/hadoop/lib/activation-
25.
26. #A lot of class path files would then follow
27.
28. 14/12/03 15:35:55 INFO namenode.NNStorageRetentionManager: Going to retain 1 images
with txid >= 0
29. 14/12/03 15:35:55 INFO util.ExitUtil: Exiting with status 0
30. 14/12/03 15:35:55 INFO namenode.NameNode: SHUTDOWN_MSG:
31. ****
32. SHUTDOWN_MSG: Shutting down NameNode at localhost.localdomain/127.0.0.1
33. ****/
34.
35.
36. Step 3: Stop all Hadoop Services

```
37. for service in /etc/init.d/hadoop*
38. do
39. sudo $service stop
40. done
41.
42. Expected Output
43. Stopping Hadoop jobtracker: [OK]
44. stopping jobtracker
45. Stopping Hadoop tasktracker: [OK]
46. stopping tasktracker
47. Stopping Hadoop datanode: [OK]
48. stopping datanode
49. Stopping Hadoop namenode: [OK]
50. no namenode to stop
51. Stopping Hadoop secondarynamenode: [OK]
52. stopping secondarynamenode
53.
54.
55. Step 4: Start all Hadoop HDFS Services
56. for service in /etc/init.d/hadoop-hdfs-*
57. do
58. sudo $service start
59. done
60.
61. Expected Output
62. Starting Hadoop datanode: [OK]
63. starting datanode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-datanode-
localhost.localdomain.out
64. Starting Hadoop namenode: [OK]
65. starting namenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-namenode-
localhost.localdomain.out
66. Starting Hadoop secondarynamenode: [OK]
67. starting secondarynamenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-
secondarynamenode-localhost.localdomain.out
68.
69.
70. Step 5: Verify from the terminal whether the Hadoop HDFS have started properly
71. sudo /usr/java/latest/bin/jps
72.
73. Expected Output
74. 3181 DataNode
```

75. **3377** SecondaryNameNode
76. **2277** HMaster
77. **3822** Jps
78. **3275** NameNode
- 79.
80. Step 6: Verify from the browser whether the HDFS services have started properly
81. url: localhost:50070
82. Expected Output
- 83.
- 84.
85. Step 7: Create the /tmp directory and manage its permissions
86. sudo -u hdfs hadoop fs -mkdir /tmp
87. sudo -u hdfs hadoop fs -chmod -R **1777** /tmp
- 88.
89. Expected Output
90. Nothing
- 91.
- 92.
93. Step 8: Create MapReduce Var directories and manage their permissions & ownership
94. sudo -u hdfs hadoop fs - mkdir -p /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
95. sudo -u hdfs hadoop fs -chmod **1777** /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
96. sudo -u hdfs hadoop fs -chown -R mapred /var/lib/hadoop-hdfs/cache/mapred
- 97.
98. Expected Output
99. Nothing
- 100.
- 101.
102. Step 9: Verify the newly created directory structure
103. sudo -u hdfs hadoop fs -ls -R /
- 104.
105. Expected Output
106. drwxrwxrwt - hdfs supergroup **0 2014-12-03 15:39** /tmp
- 107.
108. drwxr-xr-x - hdfs supergroup **0 2014-12-03 15:41** /var
- 109.
110. drwxr-xr-x - hdfs supergroup **0 2014-12-03 15:41** /var/lib
- 111.
112. drwxr-xr-x - hdfs supergroup **0 2014-12-03 15:42** /var/lib/hadoop-hdfs
- 113.
114. drwxr-xr-x - hdfs supergroup **0 2014-12-03 15:42** /var/lib/hadoop-hdfs/cache
- 115.

```
116. drwxr-xr-x - mapred supergroup 0 2014-12-03 15:42 /var/lib/hadoop-
      hdfs/cache/mapred
117.
118. drwxr-xr-x - mapred supergroup 0 2014-12-03 15:42 /var/lib/hadoop-
      hdfs/cache/mapred/mapred
119.
120. drwxrwxrwt - mapred supergroup 0 2014-12-03 15:42 /var/lib/hadoop-
      hdfs/cache/mapred/mapred/staging
121.
122.
123. Step 10: Start services related to MapReduce
124. for service in /etc/init.d/hadoop-0.20-mapreduce-*
125. do
126. sudo $service start
127. done
128.
129. Expected Output
130. Starting Hadoop jobtracker: [OK]
131. starting jobtracker, logging to /var/log/hadoop-0.20-mapreduce/hadoop-hadoop-
      jobtracker-localhost.localdomain.out
132.
133. Starting Hadoop tasktracker: [OK]
134. starting tasktracker, logging to /var/log/hadoop-0.20-mapreduce/hadoop-hadoop-
      tasktracker-localhost.localdomain.out
135.
136.
137. Step 11: Verify from the terminal whether the Hadoop MapReduce services have sta-
      rted properly
138. sudo /usr/java/latest/bin/jps
139.
140. Expected Output
141. 3960 JobTracker
142. 3181 DataNode
143. 4150 Jps
144. 3377 SecondaryNameNode
145. 2277 HMaster
146. 3275 NameNode
147. 4078 TaskTracker
148.
149.
```

150. Step 12: Verify from the browser whether the Hadoop MapReduce services have started properly
151. url: localhost:50030
152.
153. Expected Output
154.
155.
156. Step 13: Create a home directory on HDFS for the Client and manage its ownership
157. sudo -u hdfs hadoop fs -mkdir /user/Jayant
158. sudo -u hdfs hadoop fs -chown Jayant /user/Jayant
159.
160. Expected Output
161. Nothing
162.
163.
164. Step 14: Make directory in HDFS called Input and copy a sample text file
165. hadoop fs -mkdir first_input
166. hadoop fs -put /home/Jayant/Documents/test.txt first_input
167. hadoop fs -ls input
168.
169. Expected Output
170. Found 1 items
171. -rw-r--r-- 1 Jayant supergroup 125 2014-12-03 15:50 first_input/test.txt
172.
173.
174. Step 15: Run the example MapReduce program
175. /usr/bin/hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar grep first_input first_output 'W[a-z]+'
176.
177. Expected Output
178. 14/12/03 15:52:20 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
179. 14/12/03 15:52:20 INFO mapred.FileInputFormat: Total input paths to process : 1
180.
181. 14/12/03 15:52:21 INFO mapred.JobClient: Running job: job_201412031546_0001
182.
183. Then a lot of things will show up
184.
185. (bytes)=176492544
186. 14/12/03 15:52:45 INFO mapred.JobClient: org.apache.hadoop.mapreduce.lib.input.FileInputFormatCounter

- 187.
188. **14/12/03 15:52:45 INFO mapred.JobClient: BYTES_READ=106**
- 189.
190. The above line confirms successful execution of the program
- 191.
- 192.
193. **Step 16: Verify the output**
194. **hadoop fs -cat first_output/part-00000**
- 195.
196. **Expected Output**
197. **Actual Output**

Lab 02: Importing Data from MySQL and querying using HIVE

1. Step 1: Start the MySQL Server
2. sudo service mysqld start
- 3.
4. Expected Output
5. Starting mysqld: [OK]
- 6.
- 7.
8. Step 2: Verify the databases available
9. sqoop list-databases --connect jdbc:mysql://localhost/wisdom_db --username root --password root
- 10.
11. Expected Output
12. information_schema
13. mysql
14. test
15. wisdom_db
- 16.
- 17.
18. Step 3: Verify the tables available
19. sqoop list-tables --connect jdbc:mysql://localhost/wisdom_db --username root --password root
- 20.
21. Expected Output
22. server_log
- 23.
- 24.
25. Step 4: Perform the import of table server_log
26. sqoop import --connect jdbc:mysql://localhost/wisdom_db --table server_log --fields-terminated-by '\t' -m 1 --username root --password root
- 27.
28. Expected Output
29. 14/12/09 22:10:19 INFO mapred.JobClient: Running job: job_201412092203_0002
30. 14/12/09 22:10:20 INFO mapred.JobClient: map 0% reduce 0%
31. 14/12/09 22:10:29 INFO mapred.JobClient: map 100% reduce 0%
32. 14/12/09 22:10:30 INFO mapred.JobClient: Job complete: job_201412092203_0002
33. 14/12/09 22:10:30 INFO mapred.JobClient: Counters: 23
34. 14/12/09 22:10:30 INFO mapred.JobClient: File System Counters
35. 14/12/09 22:10:30 INFO mapred.JobClient: FILE: Number of bytes read=0
36. 14/12/09 22:10:30 INFO mapred.JobClient: FILE: Number of bytes written=215224

37. 14/12/09 22:10:30 INFO mapred.JobClient: FILE: Number of read operations=0
38. 14/12/09 22:10:30 INFO mapred.JobClient: FILE: Number of large read operations=0
39. 14/12/09 22:10:30 INFO mapred.JobClient: FILE: Number of write operations=0
40. 14/12/09 22:10:30 INFO mapred.JobClient: HDFS: Number of bytes read=87
41. 14/12/09 22:10:30 INFO mapred.JobClient: HDFS: Number of bytes written=989999
42. 14/12/09 22:10:30 INFO mapred.JobClient: HDFS: Number of read operations=1
43. 14/12/09 22:10:30 INFO mapred.JobClient: HDFS: Number of large read operations=0
44. 14/12/09 22:10:30 INFO mapred.JobClient: HDFS: Number of write operations=1
45. 14/12/09 22:10:30 INFO mapred.JobClient: Job Counters
46. 14/12/09 22:10:30 INFO mapred.JobClient: Launched map tasks=1
47. 14/12/09 22:10:30 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=7223
48. 14/12/09 22:10:30 INFO mapred.JobClient: Total time spent by all reduces in occupied slot s (ms)=0
49. 14/12/09 22:10:30 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0
50. 14/12/09 22:10:30 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0
51. 14/12/09 22:10:30 INFO mapred.JobClient: Map-Reduce Framework
52. 14/12/09 22:10:30 INFO mapred.JobClient: Map input records=50000
53. 14/12/09 22:10:30 INFO mapred.JobClient: Map output records=50000
54. 14/12/09 22:10:30 INFO mapred.JobClient: Input split bytes=87
55. 14/12/09 22:10:30 INFO mapred.JobClient: Spilled Records=0
56. 14/12/09 22:10:30 INFO mapred.JobClient: CPU time spent (ms)=870
57. 14/12/09 22:10:30 INFO mapred.JobClient: Physical memory (bytes) snapshot=57729024
58. 14/12/09 22:10:30 INFO mapred.JobClient: Virtual memory (bytes) snapshot=391020544
59. 14/12/09 22:10:30 INFO mapred.JobClient: Total committed heap usage (bytes)=15990784
60. 14/12/09 22:10:30 INFO mapreduce.ImportJobBase: Transferred 966.7959 KB in 12.2144 seconds (79.1522 KB/sec)
61. 14/12/09 22:10:30 INFO mapreduce.ImportJobBase: Retrieved 50000 records.
- 62.
- 63.
64. Step 5: Configure Hive Metastore and Warehouse
65. sudo -u hdfs hadoop fs -mkdir /user/hive/warehouse
66. hadoop fs -chmod g+w /tmp
67. sudo -u hdfs hadoop fs -chmod g+w /user/hive/warehouse
68. sudo -u hdfs hadoop fs -chown -R wisdom /user/hive/warehouse
69. sudo chmod 777 /var/lib/hive/metastore
- 70.
71. Expected Output

72. Nothing

73.

74.

75. Step 6: Create the table Metadata

76. `create table server_log_table (country STRING, ip_address STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;`

77.

78. Expected Output

79. Nothing

80.

81.

82. Step 7: Load data imported using Sqoop into the newly created table

83. `LOAD DATA INPATH "/user/wisdom/server_log/part-m-00000" INTO TABLE server_log_table;`

84.

85. Expected Output

86. Nothing

87.

88.

89. Step 8: Run a sample Select query

90. `SELECT country,count(1) from server_log_table group by country;`

91.

92. Expected Output

93. MapReduce Job Execution details will be loaded and then you will see the following result.

94. Australia 10000

95. India 10000

96. Newzeland 10000

97. UK 10000

98. US 10000

Lab 03: Configuring Hadoop in Distributed Mode

1. Step 1: Stop all services from all the Machines
2. `for service in /etc/init.d/hadoop*`
3. `do`
4. `sudo $service stop`
5. `done`
- 6.
- 7.
8. Step 2: Create a copy of the configuration template files from all the Machines
9. `sudo cp -r /etc/hadoop/conf.empty /etc/hadoop/conf.wisdom`
- 10.
- 11.
12. Step 3: Install the newly created configurations from all the Machines
13. `sudo /usr/sbin/alternatives --install /etc/hadoop/conf hadoop-conf /etc/hadoop/conf.wisdom 99`
- 14.
- 15.
16. Step 4: Configure Hadoop to point to the newly installed configuration from all the Machines
17. `/usr/sbin/update-alternatives --display hadoop-conf`
- 18.
- 19.
20. Step 5: Check the ip_address of all your machines
21. `/sbin/ifconfig`
- 22.
- 23.
24. Step 6: Configure the hosts file from all the Machines
25. Using the Vi editor, open the /etc/hosts file. Delete all the existing content and add all the members of your group (including yourself) in the following way
26. `ip_address_of_member host_name_of_member`
27. For example
28. `192.168.1.235 member1`
29. `192.168.1.236 member2`
- 30.
- 31.
32. Step 7: Configure the Network file from all the Machines
33. Using the Vi editor, open the /etc/sysconfig/network file. Update the Hostname field with the name that you have decided for your machine. For example
34. `Hostname = member1`
- 35.

- 36.
37. Step 8: Configure the new set of configuration files from all the Machines (here host name of Name Node machine is considered as master)
38. Name Value
39. Filename: /etc/hadoop/conf.wisdom/core-site.xml
40. fs.default.name hdfs://master:8020
- 41.
42. Filename: /etc/hadoop/conf.wisdom/dfs-site.xml
43. dfs.name.dir /home/disk1/dfs/name_space,/home/disk2/dfs/name_space
44. dfs.data.dir /home/disk1/dfs/data_space,/home/disk2/dfs/data_space
45. dfs.http.address master:50070
- 46.
47. Filename: /etc/hadoop/conf.wisdom/mapred-site.xml
48. mapred.local.dir /home/disk1/mapred/local,/home/disk2/mapred/local
49. mapred.job.tracker master:8021
50. mapred.jobtracker.staging.root.dir /user
- 51.
52. Step 9: Create the directories mentioned in the configuration files from all the Machines
53. sudo mkdir -p /home/disk1/dfs/name_space
54. sudo mkdir -p /home/disk2/dfs/name_space
55. sudo mkdir -p /home/disk1/dfs/data_space
56. sudo mkdir -p /home/disk2/dfs/data_space
57. sudo mkdir -p /home/disk1/mapred/local
58. sudo mkdir -p /home/disk2/mapred/local
- 59.
- 60.
61. Step 10: Configure permissions to the newly created directories from all the Machines
62. sudo chown -R hdfs:hadoop /home/disk1/dfs/name_space
63. sudo chown -R hdfs:hadoop /home/disk2/dfs/name_space
64. sudo chown -R hdfs:hadoop /home/disk1/dfs/data_space
65. sudo chown -R hdfs:hadoop /home/disk2/dfs/data_space
66. sudo chown -R mapred:hadoop /home/disk1/mapred/local
67. sudo chown -R mapred:hadoop /home/disk2/mapred/local
- 68.
- 69.
70. Step 11: Format the Name Node from the Master Machine
71. sudo -u hdfs hadoop namenode -format
- 72.
- 73.
74. Step 12: Start the HDFS master services from the Master Machine
75. sudo /etc/init.d/hadoop-hdfs-namenode start

76. `sudo /etc/init.d/hadoop-hdfs-secondarynamenode start`
- 77.
- 78.
79. **Step 13:** Start the HDFS slave service from the Slave Machines
80. `sudo /etc/init.d/hadoop-hdfs-datanode start`
- 81.
- 82.
- 83.
84. **Step 14:** Create HDFS directory **for** the client from any **1** Machine
85. `sudo -u hdfs hadoop fs -mkdir /user/wisdom`
86. `sudo -u hdfs hadoop fs -chown wisdom /user/wisdom`
- 87.
- 88.
89. **Step 15:** Create MapReduce Specific directories from any **1** Machine
90. `sudo -u hdfs hadoop fs -mkdir /tmp`
91. `sudo -u hdfs hadoop fs -mkdir /mapred/system`
92. `sudo -u hdfs hadoop fs -chmod -R 1777 /tmp`
93. `sudo -u hdfs hadoop fs -chmod -R 1777 /mapred/system`
94. `sudo -u hdfs hadoop fs -chown -R mapred /mapred/system`
- 95.
- 96.
97. **Step 16:** Start the MapReduce Master service from the Master machine
98. `sudo /etc/init.d/hadoop-0.20-mapreduce-jobtracker start`
- 99.
- 100.
101. **Step 17:** Start the MapReduce Slave service from the Slave Machines
102. `sudo /etc/init.d/hadoop-0.20-mapreduce-tasktracker start`
- 103.
- 104.
105. **Step 18:** Verify your HDFS Cluster
106. Go to url: master:**50070**
- 107.
- 108.
109. **Step 19:** Verify your MapReduce Cluster
110. Go to url: master:**50030**

Lab 04: Data Analysis using Pig

1. Step 1: Start Pig in Local Mode
2. `pig -x local`
- 3.
4. Expected Output
5. `2014-12-09 01:15:00,156 [main] INFO org.apache.pig.Main - Apache Pig version 0.11.0-
cdh4.6.0 (r: unknown) compiled Feb 26 2014, 03:02:34`
6. `2014-12-`
`09 01:15:00,156 [main] INFO org.apache.pig.Main - Logging error messages to: /home/training/pig_1418105700155.log`
7. `2014-12-`
`09 01:15:00,185 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/training/.pigbootup not found`
8. `2014-12-`
`09 01:15:00,310 [main] WARN org.apache.hadoop.conf.Configuration - fs.default.name is deprecated. Instead, use fs.defaultFS`
9. `2014-12-`
`09 01:15:00,311 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecution
Engine - Connecting to hadoop file system at: file:///`
10. `2014-12-`
`09 01:15:00,489 [main] WARN org.apache.hadoop.conf.Configuration - io.bytes.per.checksum
is deprecated. Instead, use dfs.bytes-per-checksum`
11. `2014-12-`
`09 01:15:00,489 [main] WARN org.apache.hadoop.conf.Configuration - fs.default.name is depre
cated. Instead, use fs.defaultFS`
12. `grunt>`
- 13.
- 14.
15. Step 2: Load the data from the Linux File System into a Pig Bag
16. `grunt> movies = LOAD '/home/wisdom/movies_data.csv' USING PigStorage(',') AS (id:int,na
me:chararray,year:int,rating:double,duration:int);`
- 17.
- 18.
19. Step 3: list the contents of the newly created Bag
20. `grunt> DUMP movies;`
- 21.
22. Expected Output
23. `(1,The Nightmare Before Christmas,1993,3.9,4568)`
24. `(2,The Mummy,1932,3.5,4388)`
25. `(3,Orphans of the Storm,1921,3.2,9062)`

26. (4,The Object of Beauty,1991,2.8,6150)
27. (5,Night Tide,1963,2.8,5126)
28. (6,One Magic Christmas,1985,3.8,5333)
29. (7,Muriel's Wedding,1994,3.5,6323)
30. (8,Mother's Boys,1994,3.4,5733)
31. (9,Nosferatu:Original Version,1929,3.5,5651)
32. (10,Nick of Time,1995,3.4,5333)
33. (,,,,)
- 34.
- 35.
36. Step 4: Filter movies with rating more than 3.5
37. hit_movies = FILTER movies BY (float) rating > 3.5;
- 38.
39. Expected Output
40. (1,The Nightmare Before Christmas,1993,3.9,4568)
41. (6,One Magic Christmas,1985,3.8,5333)
- 42.
- 43.
44. Step 5: Store the result
45. grunt> STORE hit_movies INTO '/home/wisdom/hit_movies';
- 46.
47. Expected Output
48. Success!
49. Job Stats (time in seconds):
50. JobId Alias Feature Outputs
51. job_local1485562464_0005 hit_movies,movies MAP_ONLY /home/wisdom/hit_movies,
52. Input(s):
53. Successfully read records from: "/home/wisdom/movies_data.csv"
54. Output(s):
55. Successfully stored records in: "/home/wisdom/hit_movies"
56. Job DAG:
57. job_local1485562464_0005
- 58.
- 59.
60. Step 6: Filter movies released between year 1920 and 1930
61. grunt> classical_movies = FILTER movies by year>1920 and year<1930;
- 62.
63. Expected Output
64. (3,Orphans of the Storm,1921,3.2,9062)
65. (9,Nosferatu:Original Version,1929,3.5,5651)

66.
67.
68. Step 7: Filter movies starting by "The"
69. grunt> movies_starting_with_The = FILTER movies BY name matches 'The.*';
70.
71. Expected Output
72. (1,The Nightmare Before Christmas,1993,3.9,4568)
73. (2,The Mummy,1932,3.5,4388)
74. (4,The Object of Beauty,1991,2.8,6150)
75.
76.
77. Step 8: Filter movies with duration less than 5000 seconds
78. grunt> small_movies = FILTER movies BY duration < 5000;
79.
80. Expected Output
81. (1,The Nightmare Before Christmas,1993,3.9,4568)
82. (2,The Mummy,1932,3.5,4388)
83.
84.
85. Step 9: For each movie find the number of minutes
86. grunt> movie_duration = FOREACH movies GENERATE name,(double)(duration/60);
87.
88. Expected Output
89. (The Nightmare Before Christmas,76.0)
90. (The Mummy,73.0)
91. (Orphans of the Storm,151.0)
92. (The Object of Beauty,102.0)
93. (Night Tide,85.0)
94. (One Magic Christmas,88.0)
95. (Muriel's Wedding,105.0)
96. (Mother's Boys,95.0)
97. (Nosferatu:Original Version,94.0)
98. (Nick of Time,88.0)
99.
100.
101. Step 10: Find the number of movies released every year
102. grunt> grouped_by_year = GROUP movies BY year;
103. grunt> count_by_year = FOREACH grouped_by_year GENERATE
104. group,COUNT(movies);
105. grunt> DUMP count_by_year;
106.

107. Expected Output
108. (1921,1)
109. (1929,1)
110. (1932,1)
111. (1963,1)
112. (1985,1)
113. (1991,1)
114. (1993,1)
115. (1994,2)
116. (1995,1)
117. (0)
118.
119.
120. Step 11: Get the latest movies
121. grunt> latest_movies = ORDER movies BY year DESC;
122. grunt> DUMP latest_movies;
123.
124. Expected Output
125. (10,Nick of Time,1995,3.4,5333)
126. (7,Muriel's Wedding,1994,3.5,6323)
127. (8,Mother's Boys,1994,3.4,5733)
128. (1,The Nightmare Before Christmas,1993,3.9,4568)
129. (4,The Object of Beauty,1991,2.8,6150)
130. (6,One Magic Christmas,1985,3.8,5333)
131. (5,Night Tide,1963,2.8,5126)
132. (2,The Mummy,1932,3.5,4388)
133. (9,Nosferatu:Original Version,1929,3.5,5651)
134. (3,Orphans of the Storm,1921,3.2,9062)
135. (,,,)
136.
137.
138. Step 12: List the first 3 movies
139. grunt> top_3_movies = LIMIT movies 3;
140. grunt> DUMP top_3_movies;
141.
142. Expected Output
143. (1,The Nightmare Before Christmas,1993,3.9,4568)
144. (2,The Mummy,1932,3.5,4388)
145. (3,Orphans of the Storm,1921,3.2,9062)

Lab 05: Hive Insights

1. Command 1: Start Hive
2. \$ hive
- 3.
4. Expected Output
5. Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
6. Hive history file=/tmp/wisdom/hive_job_log_5be691a2-d3a1-4d44-88d4-5b42ac1c3439_1305503489.txt
7. hive>
- 8.
- 9.
10. Command 2: Check Hive Version
11. hive> set system:sun.java.command;
- 12.
13. Expected Output
14. system:sun.java.command=org.apache.hadoop.util.RunJar /usr/lib/hive/lib/hive-cl~~i-0.10.0~~-cdh4.4.0.jar org.apache.hadoop.hive.cli.CliDriver
- 15.
- 16.
17. Command 3: List all databases
18. hive> SHOW DATABASES;
- 19.
20. Expected Output
21. OK
22. default
23. Time taken: ~~5.255~~ seconds
- 24.
- 25.
26. Command 4: Create database
27. hive> CREATE DATABASE IF NOT EXISTS financials;
- 28.
29. Expected Output
30. OK
31. Time taken: ~~0.836~~ seconds
- 32.
- 33.
34. Command 5: Create Table
35. hive> CREATE DATABASE human_resources;
36. Expected Output
37. OK

38. Time taken: 0.051 seconds

39.

40.

41.

42. Command 6: Show databases

43. hive> SHOW DATABASES;

44.

45. Expected Output

46. OK

47. default

48. financials

49. human_resources

50. Time taken: 0.098 seconds

51.

52.

53. Command 7: Using Regex

54. hive> SHOW DATABASES LIKE 'h.*';

55.

56. Expected Output

57. OK

58. human_resources

59. Time taken: 0.14 seconds

60.

61.

62. Command 8: Describe database

63. hive> DESCRIBE DATABASE financials;

64.

65. Expected Output

66. OK

67. financials hdfs://localhost:8020/user/hive/warehouse/financials.db

68. Time taken: 0.128 seconds

69.

70.

71. Command 9: Use a database

72. hive> USE financials;

73.

74. Expected Output

75. OK

76. Time taken: 0.035 seconds

77.

78.

79. Command 10: Show name of database in command line options
80. hive> set hive.cli.print.current.db=true;
81.
82. Expected Output
83. hive (financials)>
84.
85.
86. Command 11: Stop name display and drop database
87. hive (financials)> USE default;
88. hive> set hive.cli.print.current.db=false;
89. hive> DROP DATABASE IF EXISTS human_resources CASCADE;
90.
91. Expected Output
92. OK
93. Time taken: 0.765 seconds
94.
95.
96. Command 12: Show tables
97. hive> show tables;
98.
99. Expected Output
100. OK
101. user_log
102. user_log1
103. Time taken: 5.271 seconds
104.
105.
106. Command 13: Create table
107. hive> create table test_emp (name STRING, country STRING, salary INT, position STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;
108.
109. Expected Output
110. OK
111. Time taken: 1.09 seconds
112.
113.
114. Command 14: Drop Table
115. hive> DROP TABLE test_emp;
116.
117. Expected Output
118. OK

119. Time taken: 1.355 seconds
120.
121.
122. Command 15: Create Table
123. hive> create table emp (name STRING, country STRING, salary INT, position STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
124.
125. Expected Output
126. OK
127. Time taken: 0.124 seconds
128.
129.
130. Command 16: Load Data
131. hive> LOAD DATA INPATH "/user/wisdom/emp" INTO TABLE emp;
132. Expected Output
133. Loading data to table default.emp
134. Table default.emp stats: [num_partitions: 0, num_files: 1, num_rows: 0, total_size: 2 36, raw_data_size: 0]
135. OK
136. Time taken: 0.332 seconds
137.
138.
139. Command 17: List all records
140. hive> select * from emp;
141.
142. Expected Output
143. OK
144. Alice US 2000 Dev
145. Bob AUS 3000 Dev
146. Ram India 1500 Dev
147. Laxmi India 2456 QA
148. Shanti India 7666 TL
149. Laxman India 50000 Manager
150. Catty US 4000 Qa
151. Sam US 5000 TL
152. Pat US 8000 Manager
153. Baiely AUS 10000 QA
154. Glen AUS 123466 TL
155. Ricky AUS 9000 Manager
156. Time taken: 0.242 seconds

157.
158.
159. Command 18: Select all content in limit
160. hive> select * from emp LIMIT 5;

161.
162. Expected Output
163. Alice US 2000 Dev
164. Bob AUS 3000 Dev
165. Ram India 1500 Dev
166. Laxmi India 2456 QA
167. Shanti India 7666 TL
168. Time taken: 0.185 seconds

169.
170.
171. Command 19: Specify search criteria
172. hive> select * from emp where country = "India";

173.
174. Expected Output
175. Ram India 1500 Dev
176. Laxmi India 2456 QA
177. Shanti India 7666 TL
178. Laxman India 50000 Manager
179. Time taken: 20.133 seconds

180.
181.
182. Command 20: Distinct
183. hive> select DISTINCT country from emp;

184.
185. Expected Output
186. AUS
187. India
188. US
189. Time taken: 16.66 seconds

190.
191.
192. Command 21: Count
193. hive> select count(*) from emp;
194.
195. Expected Output
196. 12
197. Time taken: 12.994 seconds

```
198.  
199.  
200. Command 22: Group By  
201. hive> select country, count(name) from emp GROUP BY country;  
202.  
203. Expected Output  
204. AUS 4  
205. India 4  
206. US 4  
207. Time taken: 13.052 seconds  
208.  
209.  
210. Command 23: Having  
211. hive> select country, count(name) from emp GROUP BY country HAVING SUM(salar  
y) > 40000;  
212.  
213. Expected Output  
214. AUS 4  
215. India 4  
216. Time taken: 14.027 seconds  
217.  
218.  
219. Command 24: Describe  
220. hive> DESCRIBE emp;  
221.  
222. Expected Output  
223. OK  
224. name string  
225. country string  
226. salary int  
227. position string  
228. Time taken: 0.276 seconds  
229.  
230.  
231. Command 25: Insert  
232. hive> FROM emp e INSERT OVERWRITE TABLE empl SELECT e.name,e.country,e.sala  
ry,e.position;  
233.  
234. Expected Output  
235. Total MapReduce jobs = 3  
236. Launching Job 1 out of 3
```

237. Number of reduce tasks is set to 0 since there's no reduce operator
238. Starting Job = job_201412050226_0009, Tracking URL = http://localhost:50030/job_details.jsp?jobid=job_201412050226_0009
239. Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_201412050226_0009
240. Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
241. 2014-12-05 04:21:56,437 Stage-1 map = 0%, reduce = 0%
242. 2014-12-05 04:21:59,455 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.76 sec
243. 2014-12-05 04:22:00,462 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.76 sec
244. 2014-12-05 04:22:01,469 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.76 sec
245. 2014-12-05 04:22:02,481 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 0.76 sec
246. MapReduce Total cumulative CPU time: 760 msec
247. Ended Job = job_201412050226_0009
248. Ended Job = -605431950, job is filtered out (removed at runtime).
249. Ended Job = 1240269776, job is filtered out (removed at runtime).
250. Moving data to: hdfs://localhost:8020/tmp/hive-wisdom/hive_2014-12-05_04-21-51_661_6120076174992428081-1/-ext-10000
251. Loading data to table default.empl
252. Table default.empl stats: [num_partitions: 0, num_files: 1, num_rows: 0, total_size: 236, raw_data_size: 0]
253. 12 Rows loaded to empl
254. MapReduce Jobs Launched:
255. Job 0: Map: 1 Cumulative CPU: 0.76 sec HDFS Read: 439 HDFS Write: 236 SUCCESS
256. Total MapReduce CPU Time Spent: 760 msec
257. OK
258. Time taken: 11.392 seconds

Lab 06: Map Reduce Example: Word Count

Data Set

1. The CentOS Project is a community-driven free software effort focused on delivering a robust
2. open source ecosystem. For users, we offer a consistent manageable platform that suits a wide variety of deployments.
3. For open source communities, we offer a solid, predictable base to
4. build upon, along with extensive resources to build, test, release, and maintain their code.
- 5.
6. We're also expanding the availability of CentOS images across a number of vendors,
7. providing official images for Amazon, Google, and more.
8. For self-hosted cloud, we also provide a generic cloud-init enabled image.
- 9.
10. For more information about updates and improvements in CentOS 7,
11. please check out the release notes or the release announcement in the mailing list archive.

Problem Statement

To calculate number of words in an input file

Source Code

```
1. import java.io.IOException;
2. import java.util.*;
3. import org.apache.hadoop.fs.Path;
4. import org.apache.hadoop.conf.*;
5. import org.apache.hadoop.io.*;
6. import org.apache.hadoop.mapreduce.*;
7. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
8. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
9. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
10. import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
11.
12. public class Wordcount {
13.     public static class Map extends
14.         Mapper<LongWritable, Text, Text, IntWritable> {
15.         private final static IntWritable one = new IntWritable(1);
16.         private Text word = new Text();
```

```
17. public void map(LongWritable key, Text value, Context context)
18. throws IOException, InterruptedException {
19.     String line = value.toString();
20.     StringTokenizer tokenizer = new StringTokenizer(line);
21.     while (tokenizer.hasMoreTokens()) {
22.         word.set(tokenizer.nextToken());
23.         context.write(word, one);
24.     }
25. }
26. }
27.
28. public static class Reduce extends
29. Reducer<Text, IntWritable, Text, IntWritable> {
30.     public void reduce(Text key, Iterable<IntWritable> values,
31. Context context) throws IOException, InterruptedException {
32.         int sum = 0;
33.         for (IntWritable val : values) {
34.             sum = sum + val.get();
35.         }
36.         context.write(key, new IntWritable(sum));
37.     }
38. }
39.
40. public static void main(String[] args) throws Exception {
41.     Configuration conf = new Configuration();
42.     Job job = new Job(conf, "wordcount");
43.     job.setJarByClass(Wordcount.class);
44.     job.setOutputKeyClass(Text.class);
45.     job.setOutputValueClass(IntWritable.class);
46.     job.setMapperClass(Map.class);
47.     job.setReducerClass(Reduce.class);
48.
49.     job.setInputFormatClassTextInputFormat.class);
50.     job.setOutputFormatClass(TextOutputFormat.class);
51.
52.     FileInputFormat.addInputPath(job, new Path(args[0]));
53.     FileOutputFormat.setOutputPath(job,new
54.     Path(args[1]));
55.     job.waitForCompletion(true);
56. }
57. }
```

Lab 07. Map Reduce Example: Call Records Analysis

Data Set

```
1. 9665128505|8983006310|2015-03-01 09:08:10|2015-03-01 10:12:15|1
2. 9665128505|8983006310|2015-03-01 07:08:10|2015-03-01 08:12:15|0
3. 9665128505|8983006310|2015-03-01 09:08:10|2015-03-01 09:12:15|1
4. 9665128505|8983006310|2015-03-01 09:08:10|2015-03-01 10:12:15|0
5. 9665128506|8983006310|2015-03-01 09:08:10|2015-03-01 10:12:15|1
6. 9665128507|8983006310|2015-03-01 09:08:10|2015-03-01 10:12:15|1
```

Problem Statement

Write a Map Reduce Program to find out all phone numbers who are making more than 60 mins of STD calls.

Source Code

```
1. package in.co.hadoop.tutorials;
2. import java.io.IOException;
3. import java.text.ParseException;
4. import java.text.SimpleDateFormat;
5. import java.util.Date;
6. import org.apache.hadoop.conf.Configuration;
7. import org.apache.hadoop.fs.Path;
8. import org.apache.hadoop.io.LongWritable;
9. import org.apache.hadoop.io.Text;
10. import org.apache.hadoop.mapreduce.Job;
11. import org.apache.hadoop.mapreduce.Mapper;
12. import org.apache.hadoop.mapreduce.Reducer;
13. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
14. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
15. public class CallRecord {
16.     public static void main(String[] args) throws Exception {
17.         Configuration conf = new Configuration();
18.         if(args.length != 2) {
19.             System.err.println("Usage: stdsubscriber <in> <out>");
20.             System.exit(2);
21.         }
22.         Job job = new Job(conf, "STD Subscribers");
```

```
23. job.setJarByClass(CallRecord.class);
24. job.setMapperClass(TokenizerMapper.class);
25. job.setCombinerClass(SumReducer.class);
26. job.setReducerClass(SumReducer.class);
27. job.setOutputKeyClass(Text.class);
28. job.setOutputValueClass(LongWritable.class);
29. FileInputFormat.addInputPath(job, new Path(args[0]));
30. FileOutputFormat.setOutputPath(job, new Path(args[1]));
31. System.exit(job.waitForCompletion(true) ? 0 : 1);
32. }
33.
34. public static class SumReducer extends
35. Reducer<Text, LongWritable, Text, LongWritable> {
36. private LongWritable result = new LongWritable();
37.
38. public void reduce(Text key, Iterable<LongWritable> values,
39. Reducer<Text, LongWritable, Text, LongWritable>.Context context)
40. throws IOException, InterruptedException {
41. long sum = 0;
42. for (LongWritable val : values) {
43. sum += val.get();
44. }
45. this.result.set(sum);
46. if (sum >= 60) {
47. context.write(key, this.result);
48. }
49.
50. }
51. }
52.
53. public static class TokenizerMapper extends
54. Mapper<Object, Text, Text, LongWritable> {
55.
56. Text phoneNumber = new Text();
57. LongWritable durationInMinutes = new LongWritable();
58.
59. public void map(Object key, Text value,
60. Mapper<Object, Text, Text, LongWritable>.Context context)
61. throws IOException, InterruptedException {
62. String[] parts = value.toString().split("[\\s]+");
63. if (parts[CDRConstants.STDFlag].equalsIgnoreCase("1")) {
```

```
64.  
65. phoneNumber.set(parts[CDRConstants.fromPhoneNumber]);  
66. String callEndTime = parts[CDRConstants.callEndTime];  
67. String callStartTime = parts[CDRConstants.callStartTime];  
68. long duration = toMillis(callEndTime) - toMillis(callStartTime);  
69. durationInMinutes.set(duration / (1000 * 60));  
70. context.write(phoneNumber, durationInMinutes);  
71. }  
72. }  
73.  
74. private long toMillis(String date) {  
75.  
76. SimpleDateFormat format = new SimpleDateFormat(  
77. "yyyy-MM-dd HH:mm:ss");  
78. Date dateFrm = null;  
79. try {  
80. dateFrm = format.parse(date);  
81.  
82. } catch (ParseException e) {  
83.  
84. e.printStackTrace();  
85. }  
86. return dateFrm.getTime();  
87. }  
88. }  
89.  
90. }  
91.  
92. class CDRConstants {  
93.  
94. public static int fromPhoneNumber = 0;  
95. public static int toPhoneNumber = 1;  
96. public static int callStartTime = 2;  
97. public static int callEndTime = 3;  
98. public static int STDFlag = 4;  
99.  
100. }
```

Lab 08. Map Reduce Example: Temperature Analysis

Data Set

1. 006701199099991950051507004...999999N9+00001+99999999999
2. 004301199099991950051512004...999999N9+00221+99999999999
3. 004301199099991950051518004...999999N9-00111+99999999999
4. 004301265099991949032412004...0500001N9+01111+99999999999
5. 004301265099991949032418004...0500001N9+00781+99999999999

Problem Statement

To calculate maximum temperature from the weather dataset.

Source Code

```
1. import org.apache.hadoop.io.IntWritable;
2. import org.apache.hadoop.io.LongWritable;
3. import org.apache.hadoop.io.Text;
4. import org.apache.hadoop.mapreduce.Mapper;
5.
6. import java.io.IOException;
7.
8. import org.apache.hadoop.io.IntWritable;
9. import org.apache.hadoop.io.Text;
10. import org.apache.hadoop.mapreduce.Reducer;
11. import org.apache.hadoop.conf.Configuration;
12. import org.apache.hadoop.fs.Path;
13. import org.apache.hadoop.io.IntWritable;
14. import org.apache.hadoop.io.Text;
15. import org.apache.hadoop.mapreduce.Job;
16. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
17. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
18.
19. public class Maxtemp{
20.     public static class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text, IntW
ritable> {
21.         private static final int MISSING = 9999;
22.         @Override
23.         public void map(LongWritable key, Text value, Context context)
```

```
24. throws IOException, InterruptedException {  
25.     String line = value.toString();  
26.     if(line.length()>0){  
27.         String year = line.substring(15, 19);  
28.         int airTemperature;  
29.         if (line.charAt(40) == '+') {  
30.             airTemperature = Integer.parseInt(line.substring(41, 45));  
31.         } else {  
32.             airTemperature = Integer.parseInt(line.substring(40, 45));  
33.         }  
34.     }  
35.     if (airTemperature != MISSING) {  
36.         context.write(new Text(year), new IntWritable(airTemperature));  
37.     }  
38. }  
39. }  
40. }  
41.  
42. public static class MaxTemperatureReducer  
43. extends Reducer<Text, IntWritable, Text, IntWritable> {  
44.     @Override  
45.     public void reduce(Text key, Iterable<IntWritable> values, Context context)  
46.             throws IOException, InterruptedException {  
47.         int maxValue = Integer.MIN_VALUE;  
48.         for (IntWritable value : values) {  
49.             maxValue = Math.max(maxValue, value.get());  
50.         }  
51.         context.write(key, new IntWritable(maxValue));  
52.     }  
53. }  
54. public static void main(String[] args) throws Exception {  
55.     Configuration conf=new Configuration();  
56.     Job job = new Job();  
57.     job.setJarByClass(Maxtemp.class);  
58.     job.setJobName("Max temperature");  
59.     FileInputFormat.addInputPath(job, new Path(args[0]));  
60.     FileOutputFormat.setOutputPath(job, new Path(args[1]));  
61.     job.setMapperClass(MaxTemperatureMapper.class);  
62.     job.setReducerClass(MaxTemperatureReducer.class);  
63.     job.setOutputKeyClass(Text.class);  
64.     job.setOutputValueClass(IntWritable.class);
```

```
65. Path out=new Path(args[1]);  
66. out.getFileSystem(conf).delete(out);  
67. System.exit(job.waitForCompletion(true) ? 0 : 1);  
68. }  
69. }
```

Lab 09. Map Reduce with Distributed Cache

Customers Data

Cust ID	Name	City
100	Valentine	Fort Resolution
101	Stephen	Portland
102	Howard	Masullas
103	Conan	Bosa
104	Phillip	Langley
105	Kelly	St. Johann in Tirol
106	Prescott	Leverkusen
107	Ahmed	Lauder
108	Otto	Maunath Bhanjan
109	Akeem	Hamilton
110	Garrison	Samsun
111	Nero	Canberra
112	Nissim	Tranent
113	Ciaran	Sant'Elpidio a Mare
114	Blake	Oyo
115	Erasmus	Amroha
116	Addison	Guadalupe
117	Xanthus	Heestert

Transactions Data

100	500	03/07/2015	[43;56;366;96]
101	501	07/17/15	[93;46;36;96]
100	502	11/28/15	[943;56;325]
102	503	08/25/15	[543;56;39;96;25]
104	504	08/26/16	[64;56;56;96;25;3]
105	505	09/14/15	[94;46;36;96]
106	506	10/20/14	[63;46;36;96]
105	507	05/08/2015	[943;46;36;96]
102	508	10/09/2014	[943;56]
109	509	08/27/15	[64;56;56;96;25;3]
110	510	06/26/15	[64;56;56;96;25;3]
111	511	03/02/2016	[93;46;36;96]
112	512	05/28/15	[93;46;36;96]
113	513	12/20/15	[93;46;36;96]

Employee Driver Code

```
1. package dist_cache;
2.
3. import java.net.URI;
4.
5. import org.apache.hadoop.conf.Configuration;
6. import org.apache.hadoop.conf.Configured;
7. import org.apache.hadoop.filecache.DistributedCache;
8. import org.apache.hadoop.fs.Path;
9. import org.apache.hadoop.io.Text;
10. import org.apache.hadoop.mapreduce.Job;
11. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
13. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14. import org.apache.hadoop.util.Tool;
15. import org.apache.hadoop.util.ToolRunner;
16.
17.
18. public class EmployeeDriver extends Configured implements Tool{
19.
20.     public static void main(String[] args) throws Exception
21.     {
22.         ToolRunner.run(new EmployeeDriver(),args);
23.     }
24.
25.     @Override
26.     public int run(String[] args) throws Exception {
27.
28.         Configuration config = new Configuration();
29.         /*
30.          * Adding the file into cache i.e data node's main memory
31.          */
32.         DistributedCache.addCacheFile(new URI("/Customers.csv"),config);
33.
34.         Job job = new Job(config,"Distributed Cache");
35.         job.setJarByClass(EmployeeDriver.class);
36.
37.
38.         job.setMapperClass(EmployeeMapper.class);
39.
```

```
40.     job.setInputFormatClass(TextInputFormat.class);
41.
42.     job.setMapOutputKeyClass(Text.class);
43.     job.setMapOutputValueClass(Text.class);
44. //Number of reduce tasks are zero since we are doing map side join no reducers are required
45.     job.setNumReduceTasks(0);
46. Path out=new Path(args[1]);
47. out.getFileSystem(config).delete(out);
48.     FileInputFormat.addInputPath(job, new Path(args[0]));
49.     FileOutputFormat.setOutputPath(job, new Path(args[1]));
50.
51.     job.waitForCompletion(true);
52.
53.
54.
55.     return 0;
56. }
57.
58.
59. }
```

Employee Mapper Code

```
1. package dist_cache;
2.
3. import java.io.BufferedReader;
4. import java.io.IOException;
5. import java.io.InputStreamReader;
6. import java.net.URI;
7. import java.util.HashMap;
8.
9. import org.apache.hadoop.filecache.DistributedCache;
10. import org.apache.hadoop.fs.FSDatalInputStream;
11. import org.apache.hadoop.fs.FileSystem;
12. import org.apache.hadoop.fs.Path;
13. import org.apache.hadoop.io.LongWritable;
14. import org.apache.hadoop.io.Text;
15. import org.apache.hadoop.mapreduce.Mapper;
```

```
16.  
17.  
18. public class EmployeeMapper extends Mapper<LongWritable,Text,Text,Text>{  
19.  
20.     private URI[] files;  
21.     /*  
22.      * Declare Transaction Map to store all the records from department mapper as  
23.      * key value pairs  
24.      */  
25.     private HashMap<String,String> CustMap = new HashMap<String,String>();  
26.  
27.     /*  
28.      * Setup method will be executed once per input split , this method  
29.      * is executed before map method  
30.      */  
31.     @Override  
32.     public void setup(Context context) throws IOException  
33.     {  
34.         // Access the cache files  
35.         files = DistributedCache.getCacheFiles(context.getConfiguration());  
36.         System.out.println("files:"+ files);  
37.         Path path = new Path(files[0]);  
38.  
39.         FileSystem fs = FileSystem.get(context.getConfiguration());  
40.         /*  
41.          * Create the input stream and read the file from the cache and store  
42.          * in Hash map  
43.          */  
44.         FSDataInputStream in = fs.open(path);  
45.         BufferedReader br = new BufferedReader(new InputStreamReader(in));  
46.         String line="";  
47.         while((line = br.readLine())!=null)  
48.         {  
49.             String splits[] = line.split(",");  
50.             //splits[0] is the CustID no and splits[1 ] is the Transactions  
51.             CustMap.put(splits[0], splits[1]);  
52.  
53.         }  
54.  
55.         br.close();  
56.         in.close();
```

```
57.
58.
59. }
60. /*
61. * (non-Javadoc)
62. * @see org.apache.hadoop.mapreduce.Mapper#map(KEYIN, VALUEIN, org.apache.hadoop.mapreduce.Mapper.Context)
63. *
64. * Map method will be executed after setup method, and map method will execute per every
65. * record from the input file
66. */
67. @Override
68. public void map(LongWritable key, Text val, Context context) throws IOException, InterruptedException
69. {
70.
71.     String[] splits = val.toString().split(",");
72.     /*
73.      * Department Id is joining key in this data sets,
74.      * join the data based on the department Id
75.     */
76.     int count=0;
77.     long sum=0;
78.     if(CustMap.containsKey(splits[0]))
79.     {
80.         String Transitions=splits[3].replaceAll("\n|\r", "");
81.
82.         String []trans=Transitions.split(";");
83.         for(int i=0;i<trans.length;i++)
84.             count++;
85.         sum+=Integer.parseInt(trans[i].trim());
86.     }
87.     context.write(new Text(CustMap.get(splits[0])), new Text("Number of Trans:"+count+
88.         ", "+ "Total Amount:"+sum));
89. }
90.
91.
92.
93. }
```

94.

95. }

Lab 10. Map Reduce with Reduce Side Joins

user_logs data

1. jim logout 93.24.237.12
2. mike new_tweet 87.124.79.252
3. bob new_tweet 58.133.120.100
4. mike logout 55.237.104.36
5. jim new_tweet 93.24.237.12
6. marie view_user 122.158.130.90

users data

1. anne 22 NY
2. joe 39 CO
3. alison 35 NY
4. mike 69 VA
5. marie 27 OR
6. jim 21 OR
7. bob 71 CA
8. mary 53 NY
9. dave 36 VA
10. dude 50 CA

Reduce Side Join Code

```
1. import java.io.IOException;
2.
3. import org.apache.hadoop.conf.Configuration;
4. import org.apache.hadoop.fs.Path;
5. import org.apache.hadoop.io.LongWritable;
6. import org.apache.hadoop.io.Text;
7. import org.apache.hadoop.mapreduce.Job;
8. import org.apache.hadoop.mapreduce.Mapper;
9. import org.apache.hadoop.mapreduce.Reducer;
10. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
11. import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
12. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
13. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14. import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
15.
16.
17. public class ReduceSideJoin {
18.
19.     public static class UserMap extends Mapper<LongWritable,Text,Text,Text>{
20.
21.         @Override
22.         protected void map(LongWritable key, Text value,Context context)
23.             throws IOException, InterruptedException {
24.
25.             String line=value.toString();
26.             String tokens[]=line.split(" ");
27.             String name=tokens[0];
28.             String city=tokens[2];
29.             context.write(new Text(name), new Text("User"+"\t"+city));
30.         }
31.
32.     }
33.     public static class LogsMap extends Mapper<LongWritable,Text,Text,Text>{
34.
35.         @Override
36.         protected void map(LongWritable key, Text value,
37.             Context context)
38.             throws IOException, InterruptedException {
39.
```

```
40.     String line=value.toString();
41.     String tokens[]=line.split(" ");
42.     String name=tokens[0];
43.     String ip=tokens[2];
44.     context.write(new Text(name), new Text("Logs"+"\t"+ip));
45.   }
46.
47.
48. }
49. public static class Reduce extends Reducer<Text,Text,Text,Text>{
50.
51.   @Override
52.   protected void reduce(Text key, Iterable<Text> values,
53.                         Context context)
54.     throws IOException, InterruptedException {
55.   String city=null;
56.   String ip=null;
57.   for(Text t:values){
58.     String parts[]=t.toString().split("\t");
59.     if(parts[0].equals("User")){
60.       city=parts[1];
61.     }
62.     if(parts[0].equals("Logs")){
63.       ip=parts[1];
64.     }else{
65.       ip="IP not found";
66.     }
67.   }
68.
69.   context.write(key, new Text(city+"\t"+ip));
70. }
71.
72. }
73. public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
74.   Configuration conf=new Configuration();
75.   Job job=new Job();
76.   job.setJarByClass(ReduceSideJoin.class);
77.   job.setOutputKeyClass(Text.class);
78.   job.setReducerClass(Reduce.class);
79.   job.setOutputValueClass(Text.class);
```

```
80.    job.setInputFormatClass(TextInputFormat.class);
81.    job.setOutputFormatClass(TextOutputFormat.class);
82.    MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class,UserMap.cl
     ass);
83.    MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,LogsMap.cl
     ass);
84.    FileOutputFormat.setOutputPath(job, new Path(args[2]));
85.    Path out=new Path(args[2]);
86.    out.getFileSystem(conf).delete(out);
87.    System.exit(job.waitForCompletion(true)?0:1);
88. }
89. }
90.
91.
```

Lab 11. Map Reduce with Custom Input Format

Attributes.java

```
1. package com.iris;
2.
3. import java.io.DataInput;
4. import java.io.DataOutput;
5. import java.io.IOException;
6.
7. import org.apache.hadoop.io.Writable;
8. import org.apache.hadoop.io.WritableComparable;
9.
10.
11. public class Attributes implements WritableComparable<Attributes> {
12.     private float sepalLength;
13.     private float sepalWidth;
14.     private float petalLength;
15.     private float petalWidth;
16.
17.     public Attributes(){}
18.     public Attributes(float sepalLength, float sepalWidth, float petalLength,
19.                       float petalWidth) {
20.         super();
21.         this.sepalLength = sepalLength;
22.         this.sepalWidth = sepalWidth;
23.         this.petalLength = petalLength;
24.         this.petalWidth = petalWidth;
25.     }
26.
27.     public float getSepalLength() {
28.         return sepalLength;
29.     }
30.
31.     public void setSepalLength(float sepalLength) {
32.         this.sepalLength = sepalLength;
33.     }
34.
35.     public float getSepalWidth() {
36.         return sepalWidth;
37.     }
```

```
38.  
39. public void setSepalWidth(float sepalWidth) {  
40.     this.sepalWidth = sepalWidth;  
41. }  
42.  
43. public float getPetalLength() {  
44.     return petalLength;  
45. }  
46.  
47. public void setPetalLength(float petalLength) {  
48.     this.petalLength = petalLength;  
49. }  
50.  
51. public float getPetalWidth() {  
52.     return petalWidth;  
53. }  
54.  
55. public void setPetalWidth(float petalWidth) {  
56.     this.petalWidth = petalWidth;  
57. }  
58.  
59. @Override  
60. public void readFields(DataInput in) throws IOException {  
61.  
62.     sepalLength = in.readFloat();  
63.     sepalWidth = in.readFloat();  
64.     petalLength = in.readFloat();  
65.     petalWidth = in.readFloat();  
66. }  
67.  
68. @Override  
69. public void write(DataOutput out) throws IOException {  
70.     out.writeFloat(sepalLength);  
71.     out.writeFloat(sepalWidth);  
72.     out.writeFloat(petalLength);  
73.     out.writeFloat(petalWidth);  
74. }  
75.  
76. @Override  
77. public int compareTo(Attributes o) {  
78.     // TODO Auto-generated method stub
```

```
79.     return 0;
80. }
81.
82.
83. @Override
84. public String toString() {
85. // TODO Auto-generated method stub
86. return this.getSepalLength()+" "+this.getSepalWidth()+" "+this.getPetalLength()+" "+this
     .getPetalWidth();
87. }
88. }
```

Driver.java

```
1. package com.iris;
2.
3. import java.io.IOException;
4. import java.text.AttributedCharacterIterator.Attribute;
5. import org.apache.hadoop.conf.Configuration;
6. import org.apache.hadoop.fs.Path;
7. import org.apache.hadoop.io.Text;
8. import org.apache.hadoop.mapreduce.Job;
9. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
10. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
11. import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
12.
13. public class Driver {
14.     public static void main(String[] args) throws IOException, ClassNotFoundException, Interr
     uptedException {
15.         Configuration conf=new Configuration();
16.         Job job=new Job();
17.         job.setJarByClass(Driver.class);
18.         job.setMapperClass(IrisMapper.class);
19.         job.setOutputKeyClass(Text.class);
20.         job.setOutputValueClass(Attributes.class);
21.         job.setInputFormatClass(MyInputFormat.class);
22.         job.setOutputFormatClass(TextOutputFormat.class);
23.         Path out=new Path(args[1]);
24.         out.getFileSystem(conf).delete(out);
25.         FileInputFormat.addInputPath(job,new Path( args[0]));
26.         FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

```
27.    job.waitForCompletion(true);  
28. }  
29.  
30. }
```

IrisMapper.java

```
1. package com.iris;  
2.  
3. import java.io.IOException;  
4.  
5. import org.apache.hadoop.io.Text;  
6. import org.apache.hadoop.mapreduce.Mapper;  
7.  
8. public class IrisMapper extends Mapper<Text,Attributes,Text,Attributes> {  
9.  
10.    @Override  
11.    protected void map(Text key, Attributes value,  
12.                      Context context)  
13.        throws IOException, InterruptedException {  
14.            if(value.getPetalLength()>4.0 &&value.getSepalWidth()>3.5)  
15.                context.write(key, value);  
16.        }  
17.  
18. }
```

MyInputFormat.java

```
1. package com.iris;  
2.  
3.  
4. import java.io.IOException;  
5.  
6. import org.apache.hadoop.io.Text;  
7. import org.apache.hadoop.mapreduce.InputSplit;  
8. import org.apache.hadoop.mapreduce.RecordReader;  
9. import org.apache.hadoop.mapreduce.TaskAttemptContext;  
10. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
11.  
12. public class MyInputFormat extends FileInputFormat<Text,Attributes> {  
13. }
```

```
14.  
15. @Override  
16. public RecordReader<Text, Attributes> createRecordReader(InputSplit arg0,  
17.     TaskAttemptContext arg1) throws IOException, InterruptedException {  
18.     return new MyRecordReader();  
19. }  
20. }
```

MyRecordReader.java

```
1. package com.iris;  
2.  
3.  
4. import java.io.IOException;  
5.  
6. import org.apache.hadoop.io.Text;  
7. import org.apache.hadoop.mapreduce.InputSplit;  
8. import org.apache.hadoop.mapreduce.RecordReader;  
9. import org.apache.hadoop.mapreduce.TaskAttemptContext;  
10. import org.apache.hadoop.mapreduce.lib.input.LineRecordReader;  
11.  
12.  
13. public class MyRecordReader extends RecordReader<Text,Attributes> {  
14.  
15.     private Text key;  
16.     private Attributes value;  
17.     private LineRecordReader reader = new LineRecordReader();  
18.     @Override  
19.     public void close() throws IOException {  
20.         // TODO Auto-generated method stub  
21.         reader.close();  
22.     }  
23.  
24.     @Override  
25.     public Text getCurrentKey() throws IOException, InterruptedException {  
26.         // TODO Auto-generated method stub  
27.         return key;  
28.     }  
29.  
30.     @Override  
31.     public Attributes getCurrentValue() throws IOException, InterruptedException {
```

```
32. // TODO Auto-generated method stub
33. return value;
34. }
35.
36. @Override
37. public float getProgress() throws IOException, InterruptedException {
38. // TODO Auto-generated method stub
39. return reader.getProgress();
40. }
41.
42. @Override
43. public void initialize(InputSplit is, TaskAttemptContext tac)
44. throws IOException, InterruptedException {
45. reader.initialize(is, tac);
46.
47. }
48.
49. @Override
50. public boolean nextKeyValue() throws IOException, InterruptedException {
51. // TODO Auto-generated method stub
52.
53. boolean gotNextKeyValue = reader.nextKeyValue();
54. if(gotNextKeyValue){
55. if(key==null){
56. key = new Text();
57. }
58. if(value == null){
59. value = new Attributes();
60. }
61. Text line = reader.getCurrentValue();
62. String[] tokens = line.toString().split(",");
63. key.set(new Text(tokens[4]));
64. value.setSepalLength(new Float(Float.parseFloat(tokens[0])));
65. value.setSepalWidth(new Float(Float.parseFloat(tokens[1])));
66. value.setPetalLength(new Float(Float.parseFloat(tokens[2])));
67. value.setPetalWidth(new Float(Float.parseFloat(tokens[3])));
68.
69. }
70. else {
71. key = null;
72. value = null;
```

```
73.    }
74.    return gotNextKeyValue;
75. }
76.
77. }
78.
```

Lab 12. Map Reduce With Binary To Sequence

BinaryFilesToHadoopSequenceFile.java

```
1. import java.io.ByteArrayOutputStream;
2. import java.io.IOException;
3. import java.net.URI;
4.
5. import org.apache.hadoop.conf.Configuration;
6. import org.apache.hadoop.fs.FSDataInputStream;
7. import org.apache.hadoop.fs.FileSystem;
8. import org.apache.hadoop.fs.Path;
9. import org.apache.hadoop.io.BytesWritable;
10. import org.apache.hadoop.io.IOUtils;
11. import org.apache.hadoop.io.Text;
12. import org.apache.hadoop.mapreduce.Job;
13. import org.apache.hadoop.mapreduce.Mapper;
14. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
15. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
16. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
17. import org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat;
18. import org.apache.hadoop.util.GenericOptionsParser;
19. import org.apache.commons.*;
20.
21. /**
22. * Class BinaryFilesToHadoopSequenceFile
23. *This program is to convert many small binary files(images) into one sequence file of decent
24. * size
25.
26.
27.
28. public class BinaryFilesToHadoopSequenceFile {
29.
30. //Log4JLogger logger = new Log4JLogger().getLogger();
31. public static class BinaryFilesToHadoopSequenceFileMapper extends Mapper<Object, Text,
32. Text, BytesWritable> {
33. /**
34. * @method map
```

```
35. *The Mapper implementation,TextInputFormat takes the offset value as key and entire path  
   of the image line by line as value  
36. *Entire value is read in buffer and written back to bytearray output stream  
37. *and emits the path of the images as key and entire file contained in bytearray output str  
   eam as value  
38. *Output :Key :Path of the image value :Entire file consisting of all images  
39. */  
40.  
41.  
42. public void map(Object key, Text value, Context context)  
43. throws IOException, InterruptedException {  
44.  
45. //logger.info("map method called..");  
46.  
47. String uri = value.toString();  
48. Configuration conf = new Configuration();  
49. FileSystem fs = FileSystem.get(URI.create(uri), conf);  
50. FSDataInputStream in = null;  
51. try {  
52. in = fs.open(new Path(uri));  
53. java.io.ByteArrayOutputStream bout = new ByteArrayOutputStream();  
54. byte buffer[] = new byte[1024 * 1024];  
55.  
56. while( in.read(buffer, 0, buffer.length) >= 0 ) {  
57. bout.write(buffer);  
58. }  
59. context.write(value, new BytesWritable(bout.toByteArray()));  
60. }finally {  
61. IOUtils.closeStream(in);  
62. }  
63. }  
64.  
65. }  
66. /**  
67. * @method main  
68. * In the main method we define sequenceformat class which takes the key and values fro  
   m mapper and stores them as a sequence file with all the pairs of keys and values in hdfs  
69. */  
70.  
71.  
72. public static void main(String[] args) throws Exception {
```

```
73. Configuration conf = new Configuration();
74. //String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
75. // if (otherArgs.length != 2) {
76. // System.err.println("Usage: BinaryFilesToHadoopSequenceFile <in Path for url file> <out p
    at for sequence file>");
77. // System.exit(2);
78. //}
79.
80. Job job = new Job(conf, "BinaryFilesToHadoopSequenceFile");
81. job.setJarByClass(BinaryFilesToHadoopSequenceFile.class);
82. job.setMapperClass(BinaryFilesToHadoopSequenceFileMapper.class);
83. job.setOutputKeyClass(Text.class);
84. job.setOutputValueClass(BytesWritable.class);
85. job.setInputFormatClass(TextInputFormat.class);
86. job.setOutputFormatClass(SequenceFileOutputFormat.class);
87. FileInputFormat.addInputPath(job, new Path(args[0]));
88. FileOutputFormat.setOutputPath(job, new Path(args[1]));
89. System.exit(job.waitForCompletion(true) ? 0 : 1);
90. }
91.
92.
93. }
```

convert_to_sequence.java

```
1. import java.io.ByteArrayOutputStream;
2. import java.io.IOException;
3. import java.net.URI;
4.
5. import org.apache.hadoop.conf.Configuration;
6. import org.apache.hadoop.fs.FSDataInputStream;
7. import org.apache.hadoop.fs.FileSystem;
8. import org.apache.hadoop.fs.Path;
9. import org.apache.hadoop.io.BytesWritable;
10. import org.apache.hadoop.io.IOUtils;
11. import org.apache.hadoop.io.Text;
12. import org.apache.hadoop.mapreduce.Job;
13. import org.apache.hadoop.mapreduce.Mapper;
14. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
15. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
16. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
17. import org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat;
18. import org.apache.hadoop.util.GenericOptionsParser;
19. import org.apache.commons.*;
20. //import org.apache.commons.logging.impl.Log4JLogger;
21.
22.
23. public class convert_to_sequence {
24.
25.     //Log4JLogger logger = new Log4JLogger().getLogger();
26.
27.     public static class BinaryFilesToHadoopSequenceFileMapper extends Mapper<Object, Te
xt, Text, BytesWritable> {
28.
29.         public void map(Object key, Text value, Context context)
30.             throws IOException, InterruptedException {
31.
32.             //logger.info("map method called..");
33.
34.             String uri = value.toString();
35.             Configuration conf = new Configuration();
36.             FileSystem fs = FileSystem.get(URI.create(uri), conf);
37.             FSDataInputStream in = null;
38.             try {
39.                 in = fs.open(new Path(uri));
40.                 java.io.ByteArrayOutputStream bout = new ByteArrayOutputStream();
41.                 byte buffer[] = new byte[1024 * 1024];
42.
43.                 while( in.read(buffer, 0, buffer.length) >= 0 ) {
44.                     bout.write(buffer);
45.                 }
46.                 context.write(value, new BytesWritable(bout.toByteArray()));
47.             } finally {
48.                 IOUtils.closeStream(in);
49.             }
50.         }
51.
52.     }
53.
54.
55.     public static void main(String[] args) throws Exception {
56.         Configuration conf = new Configuration();
```

```
57. //String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
58. // if (otherArgs.length != 2) {
59. //     System.err.println("Usage: BinaryFilesToHadoopSequenceFile <in Path for url file> <
      out pat for sequence file>");
60. //     System.exit(2);
61. // }
62.
63.
64. Job job = new Job(conf, "convert_to_sequence");
65. job.setJarByClass(convert_to_sequence.class);
66. job.setMapperClass(BinaryFilesToHadoopSequenceFileMapper.class);
67. job.setOutputKeyClass(Text.class);
68. job.setOutputValueClass(BytesWritable.class);
69. job.setInputFormatClass(TextInputFormat.class);
70. job.setOutputFormatClass(SequenceFileOutputFormat.class);
71.
72. FileInputFormat.addInputPath(job, new Path(args[0]));
73. FileOutputFormat.setOutputPath(job, new Path(args[1]));
74. System.exit(job.waitForCompletion(true) ? 0 : 1);
75. }
76.
77.
78. }
```

Lab 13. Map Reduce with Duplicate Remover

ImageDriver.java

```
1. import org.apache.hadoop.fs.Path;
2. import org.apache.hadoop.io.IntWritable;
3. import org.apache.hadoop.io.Text;
4. import org.apache.hadoop.conf.Configuration;
5. import org.apache.hadoop.conf.Configured;
6. import org.apache.hadoop.mapred.TextInputFormat;
7. //import org.apache.hadoop.mapreduce.lib.input.NLineInputFormat;
8. import org.apache.hadoop.mapreduce.Job;
9. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
10. import org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat;
11. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
12. import org.apache.hadoop.util.Tool;
13. import org.apache.hadoop.util.ToolRunner;
14.
15.
16.
17.
18.
19. public class ImageDriver extends Configured implements Tool
20. {
21.
22.     public int run(String[] args) throws Exception
23.     {
24.         //getting configuration object and setting job name
25.         Configuration conf = getConf();
26.         Job job = new Job(conf, "Word Count hadoop-0.20");
27.
28.         //setting the class names
29.         job.setJarByClass(ImageDriver.class);
30.         job.setMapperClass(ImageDuplicatesMapper.class);
31.         job.setInputFormatClass(SequenceFileInputFormat.class);
32.         //job.setCombinerClass(WordCountReducer.class);
33.         job.setReducerClass(ImageDupsReducer.class);
34.
35.
36.         //setting the output data type classes
37.         job.setOutputKeyClass(Text.class);
```

```
38. job.setOutputValueClass(Text.class);
39.
40. //to accept the hdfs input and output dir at run time
41. FileInputFormat.addInputPath(job, new Path(args[0]));
42. FileOutputFormat.setOutputPath(job, new Path(args[1]));
43.
44. return job.waitForCompletion(true) ? 0 : 1;
45. }
46. public static void main(String[] args) throws Exception {
47.     int res = ToolRunner.run(new Configuration(), new ImageDriver(), args);
48.     System.exit(res);
49. }
50. }
```

ImageDuplicatesMapper.java

```
1. import java.io.IOException;
2. import java.security.MessageDigest;
3. import java.security.NoSuchAlgorithmException;
4. import org.apache.hadoop.conf.Configuration;
5. import org.apache.hadoop.fs.Path;
6. import org.apache.hadoop.io.BytesWritable;
7. import org.apache.hadoop.io.Text;
8. import org.apache.hadoop.mapreduce.Job;
9. import org.apache.hadoop.mapreduce.Mapper;
10. import org.apache.hadoop.mapreduce.Reducer;
11. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12. import org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat;
13. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14. import org.apache.hadoop.util.GenericOptionsParser;
15.
16.
17. public class ImageDuplicatesMapper extends Mapper<Text, BytesWritable, Text, Text>{
18.
19.
20.
21.     public void map(Text key, BytesWritable value, Context context) throws IOException,
InterruptedException {
22.
23.         //get the md5 for this specific file
```

```
24.  
25.     String md5Str;  
26.  
27.     try {  
28.  
29.         md5Str = calculateMd5(value.getBytes());  
30.  
31.     } catch (NoSuchAlgorithmException e) {  
32.  
33.         e.printStackTrace();  
34.  
35.         context.setStatus("Internal error - can't find the algorithm for calculating th  
e md5");  
36.  
37.         return;  
38.  
39.     }  
40.  
41.     Text md5Text = new Text(md5Str);  
42.  
43.  
44.  
45.     //put the file in the map where the md5 is the key, so duplicates will  
46.  
47.     // be grouped together for the reduce function  
48.  
49.     context.write(md5Text, key);  
50.  
51. }
```

52.

53.

54.

55.

56.

```
57. static String calculateMd5(byte[] imageData) throws NoSuchAlgorithmException {  
58.  
59.     //get the md5 for this specific data  
60.  
61.     MessageDigest md = MessageDigest.getInstance("MD5");  
62.  
63.     md.update(imageData);
```

```
64.  
65.     byte[] hash = md.digest();  
66.  
67.  
68.  
69.     // Below code of converting Byte Array to hex  
70.  
71.     String hexString = new String();  
72.  
73.     for (int i=0; i < hash.length; i++) {  
74.  
75.         hexString += Integer.toHexString( ( hash[i] & 0xff ) + 0x100, 16).substring( 1 );  
76.  
77.     }  
78.  
79.     return hexString;  
80.  
81. }
```

ImageDupsReducer.java

```
1. import java.io.IOException;  
2.  
3. import org.apache.hadoop.io.Text;  
4. import org.apache.hadoop.mapreduce.Reducer;  
5.  
6.  
7. public class ImageDupsReducer extends Reducer<Text,Text,Text,Text> {  
8.  
9.  
10.  
11.    public void reduce(Text key, Iterable<Text> values, Context context)  
12.        throws IOException, InterruptedException {  
13.  
14.  
15.        //Key here is the md5 hash while the values are all the image files that  
16.
```

```
17.      // are associated with it. for each md5 value we need to take only
18.
19.      // one file (the first)
20.
21.      Text imageFilePath = null;
22.
23.      for (Text filePath : values) {
24.
25.          imageFilePath = filePath;
26.
27.          break; //only the first one
28.
29.      }
30.
31.      // In the result file the key will be again the image file path.
32.
33.      context.write(imageFilePath, key);
34.
35.  }
36.
37. }
```

Lab 14. Pig Scripts

pig_with_schema.pig

```
1. weblogs = LOAD '/user/training/weblogs/web*' USING PigStorage('\t')
2. AS (
3.   client_ip:chararray,
4.   full_request_date:chararray,
5.   day:int,
6.   month:chararray,
7.   month_num:int,
8.   year:int,
9.   hour:int,
10. minute:int,
11. second:int,
12. timezone:chararray,
13. http_verb:chararray,
14. uri:chararray,
15. http_status_code:chararray,
16. bytes_returned:chararray,
17. referrer:chararray,
18. user_agent:chararray
19. );
20.
21. DUMP weblogs;
```

pig_with_schema_and_describe.pig

```
1. weblogs = LOAD '/user/training/weblogs/web*' USING PigStorage('\t')
2.     AS (
3.     client_ip:chararray,
4.     full_request_date:chararray,
5.     day:int,
6.     month:chararray,
7.     month_num:int,
8.     year:int,
9.     hour:int,
10.    minute:int,
11.    second:int,
12.    timezone:chararray,
13.    http_verb:chararray,
14.    uri:chararray,
15.    http_status_code:chararray,
16.    bytes_returned:chararray,
17.    referrer:chararray,
18.    user_agent:chararray
19. );
20.
21. DESCRIBE weblogs;
```

pig_with_schema_and_explain.pig

```
1. weblogs = LOAD '/user/training/weblogs/web*' USING PigStorage('\t')
2. AS (
3. client_ip:chararray,
4. full_request_date:chararray,
5. day:int,
6. month:chararray,
7. month_num:int,
8. year:int,
9. hour:int,
10. minute:int,
11. second:int,
12. timezone:chararray,
13. http_verb:chararray,
14. uri:chararray,
15. http_status_code:chararray,
16. bytes_returned:chararray,
17. referrer:chararray,
18. user_agent:chararray
19. );
20. b = FILTER weblogs BY uri == '/careers';
21. EXPLAIN b;
```

pig_with_schema_and_filter.pig

```
1. weblogs = LOAD '/user/training/weblogs/web*' USING PigStorage('\t')
2.     AS (
3.     client_ip:chararray,
4.     full_request_date:chararray,
5.     day:int,
6.     month:chararray,
7.     month_num:int,
8.     year:int,
9.     hour:int,
10.    minute:int,
11.    second:int,
12.    timezone:chararray,
13.    http_verb:chararray,
14.    uri:chararray,
15.    http_status_code:chararray,
16.    bytes_returned:chararray,
17.    referrer:chararray,
18.    user_agent:chararray
19. );
20. careers_visitors = FILTER weblogs BY uri == '/careers';
21. DUMP careers_visitors;
```

pig_with_schema_and.foreach.pig

```
1. weblogs = LOAD '/user/training/weblogs/web*' USING PigStorage('\t')
2.     AS (
3.     client_ip:chararray,
4.     full_request_date:chararray,
5.     day:int,
6.     month:chararray,
7.     month_num:int,
8.     year:int,
9.     hour:int,
10.    minute:int,
11.    second:int,
12.    timezone:chararray,
13.    http_verb:chararray,
14.    uri:chararray,
15.    http_status_code:chararray,
16.    bytes_returned:chararray,
17.    referrer:chararray,
18.    user_agent:chararray
19. );
20. careers_visitors = FILTER weblogs BY uri == '/careers';
21. select_fields = FOREACH careers_visitors GENERATE client_ip;
22. DUMP select_fields;
```

pig_with_schema_and_groupby.pig

```
1. weblogs = LOAD '/user/training/weblogs/web*' USING PigStorage('\t')
2.     AS (
3.     client_ip:chararray,
4.     full_request_date:chararray,
5.     day:int,
6.     month:chararray,
7.     month_num:int,
8.     year:int,
9.     hour:int,
10.    minute:int,
11.    second:int,
12.    timezone:chararray,
13.    http_verb:chararray,
14.    uri:chararray,
15.    http_status_code:chararray,
16.    bytes_returned:chararray,
17.    referrer:chararray,
18.    user_agent:chararray
19. );
20. groups_fields = GROUP weblogs BY http_verb;
21. count = FOREACH groups_fields GENERATE group,COUNT(weblogs);
22. DUMP count;
```

Lab 15. Hive Titanic Data Analysis

Problem Statement

1. Find number of female passengers who survived the crash
- 2.
3. Find the average age of the passengers who survived the crash
- 4.
5. Find number of passengers Embarked in Queenstown and who survived the crash
- 6.
7. Find names of the passengers who embarked in Cherbourg
- 8.
9. Find names of the passengers that died travelling in passenger **class 1**
- 10.
11. Find number of passengers survived who boarded in all the Embarked ports
- 12.
13. Find number of passengers who survived in each **class** and sort them in descending order.
- 14.
15. Find the average fare of each passenger **class**.
- 16.
17. Find the number of passengers survived in each **class** who have Embarked in Southampton port

Data Set Link

1. <https://drive.google.com/open?id=1oTh1pXy71a2KBChSuJ4gDEPpQHmdPSS4>

Table Creation

1. `hive> create table Titanic`
2. `(PassengerId String,Survived int,Pclass int,Name String,`
3. `Sex String,Age int,SibSp int,Parch int,Ticket String,`
4. `Fare float,Cabin String,Embarked String)`
5. `row format delimited fields terminated by ','`
6. `tblproperties ("skip.header.line.count"="1");`
- 7.
8. `hive> load data inpath '/home/wisdom/TitanicData/' into table Titanic;`

Solution

1. Problem: Find number of female passengers who survived the crash
2. Solution: hive> SELECT count (*) from Titanic where Survived == 1 and Sex == 'female';
- 3.
4. Problem: Find the average age the passengers who survived the crash
5. Solution: hive> SELECT avg(Age) from Titanic where Survived == 1;
- 6.
7. Problem: Find number of passengers Embarked in Queenstown and who survived the crash
8. Solution: hive> SELECT Count (*) from Titanic where Embarked = 'Q';
- 9.
10. Problem: Find names of the passengers who embarked in Cherbourg
11. Solution: hive> SELECT Name from Titanic where Embarked = 'C';
- 12.
13. Problem: Find names of the passengers died travelling in passenger class 1
14. Solution: hive> SELECT Name from Titanic where Pclass = "1" and Survived == 0;
- 15.
16. Problem: Find number of passengers survived who boarded in all the Embarked ports
17. Solution: hive> SELECT Embarked, count (*) as ct from Titanic where Survived == 1 and Embarked is not null group by Embarked sort by ct asc;
- 18.
19. Problem: Find number of passengers who survived in each class and sort them in descending order.
20. Solution: hive> SELECT Pclass , count(*) as ct from Titanic where Survived == 1 group by Pclass sort by ct desc;
- 21.
22. Problem: Find the average fare of each passenger class.
23. Solution: hive> SELECT Pclass , avg(fare) from Titanic group by Pclass;
- 24.
25. Problem: Find number of passengers survived in each class who have Embarked in Southampton port
26. Solution: hive> SELECT Pclass , Embarked ,count(*) from Titanic where Survived == 1 and Embarked == 'S' group by Pclass , Embarked;

Lab 16. Consumer Complaint Analysis

Problem Statement

1. Problem Statement 1
2. Write a pig script to find no of complaints which got timely response
- 3.
4. Problem Statement 2
5. Write a pig script to find no of complaints where consumer forum forwarded the complaint same day they received to respective company
- 6.
- 7.
8. Problem Statement 3
9. Write a pig script to find list of companies topping in complaint chart (companies with maximum number of complaints)
- 10.
- 11.
12. Problem Statement 4
13. Write a pig script to find no of complaints filed with product type has "Debt collection" for the year 2015

Data Set Link

1. https://drive.google.com/open?id=1fUWC_5vCMyyyNfgBLXipjyd3_iVrtccW

Data Set Description

1. Column heading index Description
2. Date received 0 date on which consumer filed the complaint
- 3.
4. Product 1 Type of the product
5. Sub-product 2 Sub product type
6. Issue 3 Issue faced by the consumer
7. Sub-issue 4 Any sub issues if exists
8. Consumer complaint narrative 5 Detailed description of complaint
9. Company public response 6 Company's public response to the complaint
10. Company 7 Name of the company
11. State 8 State from which consumer filed the complaint
12. ZIP code 9 Zip code
13. Submitted via 10 Channel from which complaint was submitted

14. Date sent to company **11** Date on which consumer forum forwarded the complaint to company
15. Company response to consumer **12** Company's response to the consumer
16. Timely response? **13**
17. Consumer disputed? **14**
18. Complaint ID **15** Unique complaint id
- 19.
- 20.
21. This data is comma delimited. bl1Cm3UH3e
22. hJ9Ms8CW4i
23. In some rows there are few columns which are enclosed in **double quotes** and have many commas and due to **this** the same column gets spitted in to many columns **for example:**
- 24.

Sample record:

- 25.
26. **10/16/2015**, Debt collection, "Other (phone, health club, etc.)", **Cont'd attempts collect debt not owed, Debt was discharged in bankruptcy,,,"Convergent Resources, Inc."**, OH,438XX, Web,**10/16/2015**,Closed with explanation, Yes,,**1612132**
- 27.
28. This is the entire column "**Other (phone, health club, etc.)**" should be product but, **if** we split **this** file based on comma then **this** column will be spitted into **3** columns which will result in wrong outputs.
- 29.
30. In order to tackle **this** we should remove the comma's present only inside **double quotes**.
- 31.
32. Since Hadoop is used to handle big data it's recommended to use java map reduce to remove unnecessary commas.
- 33.
34. Note: work on **this** problem statements after doing the data cleaning as mentioned above.

Cleandata.java

```
1. import java.io.IOException;
2.
3. import org.apache.hadoop.fs.Path;
4. import org.apache.hadoop.io.LongWritable;
5. import org.apache.hadoop.io.NullWritable;
6. import org.apache.hadoop.io.Text;
7. import org.apache.hadoop.mapreduce.Job;
8. import org.apache.hadoop.mapreduce.Mapper;
9. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
10. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
11. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
12. import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
13.
14.
15.
16. public class Cleandata {
17.
18.     public static class Map extends Mapper<LongWritable, Text, Text, NullWritable> {
19.
20.         @Override
21.         protected void map(LongWritable key, Text value, Context context)
22.             throws IOException, InterruptedException {
23.
24.             String s = value.toString();
25.             String s2="";
26.             String s1[] = s.split(",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)\"");
27.             for(int i=0;i<s1.length;i++){
28.                 if(s1[i].startsWith("\\"))s1[i]=s1[i].replace("\\","");
29.
30.                 s2=s2+s1[i]+ ",";
31.             }
32.
33.
34.             context.write(new Text(s2), NullWritable.get());
35.         }
36.
37.     }
38.
39.
```

```
40.  
41. public static void main(String[] args) throws Exception {  
42.  
43.  
44.     //JobConf conf = new JobConf(Cleandata.class);  
45.     Job job=new Job();  
46.     job.setJobName("wordcount");  
47.     job.setJarByClass(Cleandata.class);  
48.     job.setOutputKeyClass(Text.class);  
49.     job.setOutputValueClass(NullWritable.class);  
50.     job.setMapOutputKeyClass(Text.class);  
51.     job.setMapOutputValueClass(NullWritable.class);  
52.  
53.     job.setMapperClass(Map.class);  
54.     job.setNumReduceTasks(0);  
55.     // conf.setReducerClass(Reduce.class);  
56.  
57.     job.setInputFormatClass(TextInputFormat.class);  
58.     job.setOutputFormatClass(TextOutputFormat.class);  
59.  
60.  
61.     FileInputFormat.setInputPaths(job, new Path(args[0]));  
62.     FileOutputFormat.setOutputPath(job, new Path(args[1]));  
63.  
64.  
65.     job.waitForCompletion(true);  
66.  
67. }  
68. }
```

Pig Solution 1

```
1. --Write a Pig Script to find no of complaints which got timely response
2.
3. consumer_data = load '/home/wisdom/consumer/part-m-00000' using PigStorage(',');
4.
5. timely_response = filter consumer_data by $13 == 'No';
6.
7. grouped_response = group timely_response ALL;
8.
9. final_result = foreach grouped_response generate COUNT (timely_response);
10.
11. dump final_result;
```

Pig Solution 2

1. --
Write a Pig Script to find no of complaints where consumer forum forwarded the complaint same day they received to respective company
- 2.
3. consumer_data = load '/home/wisdom/consumer/part-m-00000' using PigStorage (',');
- 4.
5. forwarding_date = filter consumer_data by \$0 == \$11;
- 6.
7. grouped_forwarding_date = group forwarding_date ALL;
- 8.
9. final_result = foreach grouped_forwarding_date generate COUNT (forwarding_date);
- 10.
11. dump final_result

Pig Solution 3

```
1. --
   Write a Pig Script to find list of companies toping in complaint chart (companies with maximum number of complaints)
2.
3. consumer_data = load '/home/wisdom/consumer/part-m-00000' using PigStorage(',');
4.
5. companies = filter consumer_data by $7 is not null;
6.
7. B = group companies by $7;
8.
9. grouped_companies = foreach B generate group, COUNT(companies.$15) as ct;
10.
11. ordered_grouped_companies = order grouped_companies by ct desc;
12.
13. final_result = limit ordered_grouped_companies 10;
14.
15. dump final_result;
```

Pig Solution 4

```
1. --
   Write a Pig Script to find the number of complaints filed on with product type has "Debt collection" for the year 2015
2.
3. consumer_data = load '/home/wisdom/consumer/part-m-00000' using PigStorage(',');
4.
5. debt_data = filter consumer_data by SUBSTRING ($0, 6, 10) == '2015' and $1 == 'Debt collection';
6.
7. grouped_debt_data = group debt_data ALL;
8.
9. final_result = foreach grouped_debt_data generate COUNT(debt_data);
10.
11. dump final_result
```

Lab 17. Crime Data Analysis

Dataset Link

1. <https://drive.google.com/open?id=1iu5dPzly0-TbzH-J4-FSk-Rk6wR30qt5>

Problem Statement

1. **Problem Statement 1**
2. Write a mapreduce program to calculate the number of cases investigated under each FBI code
- 3.
4. **Problem Statement 2**
5. Write a mapreduce program to calculate the number of cases investigated under FBI code **3 2**.
- 6.
7. **Problem Statement 3**
8. Write a mapreduce program to calculate the number of arrests in theft district wise.
- 9.
10. **Problem Statement 4**
11. Write a mapreduce program to calculate the number of arrests done between October **2014** and October **2015**.

Solution 1

District_wise_Crimes.java

```
1. import org.apache.hadoop.fs.Path;
2. import org.apache.hadoop.conf.*;
3. import org.apache.hadoop.io.*;
4. import org.apache.hadoop.mapreduce.*;
5. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
6. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
7. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
8. import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
9. public class District_wise_Crimes
10. {
11.     public static void main(String[] args) throws Exception
12.     {
13.         Configuration conf = new Configuration();
14.         Job job = new Job(conf, "wordcount");
15.         job.setJarByClass(District_wise_Crimes.class);
16.         job.setMapOutputKeyClass(Text.class);
17.         job.setMapOutputValueClass(Text.class);
18.         job.setOutputKeyClass(Text.class);
19.         job.setOutputValueClass(IntWritable.class);
20.         job.setMapperClass(Map.class);
21.         //job.setNumReduceTasks(0);
22.         job.setReducerClass(Reduce.class);
23.
24.         job.setInputFormatClass(TextInputFormat.class);
25.         job.setOutputFormatClass(TextOutputFormat.class);
26.
27.         FileInputFormat.addInputPath(job, new Path(args[0]));
28.         FileOutputFormat.setOutputPath(job,new Path(args[1]));
29.         //Path out=new Path(args[1]);
30.         //out.getFileSystem(conf).delete(out);
31.         job.waitForCompletion(true);
32.     }
33. }
```

Map.java

```
1. import java.io.IOException;
2.
3. import org.apache.hadoop.io.*;
4. import org.apache.hadoop.mapreduce.*;
5.
6. public class Map extends Mapper<LongWritable, Text, Text, Text>
7. {
8.
9.     public void map(LongWritable key, Text value, Context context)
10. throws IOException, InterruptedException
11. {
12.     String line=value.toString();
13.     if(line.length()>0 && line!=null)
14.     {
15.         String tokens[]=line.split(",");
16.         if(tokens.length>11)
17.         {
18.             String district=tokens[11];
19.             String Crime_type=tokens[5];
20.             String arrest=tokens[8];
21.
22.             context.write(new Text(district), new Text(Crime_type+","+arrest));
23.         }
24.     }
25. }
26. }
```

Reduce.java

```
1. import java.io.IOException;
2. import org.apache.hadoop.io.*;
3. import org.apache.hadoop.mapreduce.*;
4. public class Reduce extends Reducer<Text, Text, Text, IntWritable> {
5.     public void reduce(Text key, Iterable<Text> values,
6. Context context) throws IOException, InterruptedException {
7.     int count=0;
8.     for(Text record:values){
9.         String line=record.toString();
10.        String parts[]=line.split(",");
11.        if(parts[0].contains("THEFT")&&parts[1].equals("true")){
12.            count++;
13.        }
14.    }
15. }
16. context.write(new Text("Total arrest for theft in:"+key), new IntWritable(count));
17.
18. }
19. }
```

Solution 2

District_wise_Crimes.java

```
1. import org.apache.hadoop.fs.Path;
2. import org.apache.hadoop.conf.*;
3. import org.apache.hadoop.io.*;
4. import org.apache.hadoop.mapreduce.*;
5. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
6. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
7. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
8. import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
9. public class District_wise_Crimes
10. {
11.     public static void main(String[] args) throws Exception {
12.         Configuration conf = new Configuration();
13.         Job job = new Job(conf);
14.         job.setJarByClass(District_wise_Crimes.class);
15.
16.         job.setMapOutputKeyClass(Text.class);
17.         job.setMapOutputValueClass(IntWritable.class);
18.         job.setOutputKeyClass(Text.class);
19.         job.setOutputValueClass(IntWritable.class);
20.
21.         job.setMapperClass(Map.class);
22.         job.setReducerClass(Reduce.class);
23.
24.         job.setInputFormatClass(TextInputFormat.class);
25.         job.setOutputFormatClass(TextOutputFormat.class);
26.
27.         FileInputFormat.addInputPath(job, new Path(args[0]));
28.         FileOutputFormat.setOutputPath(job, new Path(args[1]));
29.         job.waitForCompletion(true);
30.     }
31. }
```

Map.java

```
1. import java.io.IOException;
2. import org.apache.hadoop.io.*;
3. import org.apache.hadoop.mapreduce.*;
4.
5.
6. public class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
7.
8.     public void map(LongWritable key, Text value, Context context)
9.         throws IOException, InterruptedException {
10.        String line=value.toString();
11.        if(line.length()>0 && line!=null){
12.            String tokens[]=line.split(",");
13.            if(tokens.length>11){
14.                String Fbicode=tokens[14];
15.                //if(Fbicode.equals("32"))
16.                context.write(new Text(Fbicode), new IntWritable(1));
17.            }
18.        }
19.    }
20. }
21. }
22. }
```

Reduce.java

```
1. import java.io.IOException;
2. import org.apache.hadoop.io.*;
3. import org.apache.hadoop.mapreduce.*;
4.
5. public class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
6.     public void reduce(Text key, Iterable<IntWritable> values,
7. Context context)
8. throws IOException, InterruptedException {
9.     int count=0;
10.    for(IntWritable record:values)
11.    {
12.        count++;
13.    }
14. }
15.
16. }
17. context.write(new Text("Number of crimes investigated for FBI code:"+key+" is"), new Int
Writable(count));
18.
19. }
20. }
```

Solution 3

District_wise_Crimes.java

```
1. import org.apache.hadoop.fs.Path;
2. import org.apache.hadoop.conf.*;
3. import org.apache.hadoop.io.*;
4. import org.apache.hadoop.mapreduce.*;
5. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
6. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
7. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
8. import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
9. public class District_wise_Crimes {
10. public static void main(String[] args) throws Exception {
11. Configuration conf = new Configuration();
12. Job job = new Job(conf, "wordcount");
13. job.setJarByClass(District_wise_Crimes.class);
14. job.setMapOutputKeyClass(Text.class);
15. job.setMapOutputValueClass(IntWritable.class);
16. job.setOutputKeyClass(Text.class);
17. job.setOutputValueClass(IntWritable.class);
18. job.setMapperClass(Map.class);
19. //job.setNumReduceTasks(0);
20. job.setReducerClass(Reduce.class);
21.
22. job.setInputFormatClass(TextInputFormat.class);
23. job.setOutputFormatClass(TextOutputFormat.class);
24.
25. FileInputFormat.addInputPath(job, new Path(args[0]));
26. FileOutputFormat.setOutputPath(job,new Path(args[1]));
27. //Path out=new Path(args[1]);
28. //out.getFileSystem(conf).delete(out);
29. job.waitForCompletion(true);
30. }
31. }
```

Map.java

```
1. import java.io.IOException;
2. import org.apache.hadoop.io.*;
3. import org.apache.hadoop.mapreduce.*;
4.
5.
6. public class Map extends Mapper<LongWritable, Text, Text, IntWritable>
7. {
8.
9.     public void map(LongWritable key, Text value, Context context)
10. throws IOException, InterruptedException
11. {
12.     String line=value.toString();
13.     if(line.length()>0 && line!=null)
14.     {
15.         String tokens[]=line.split(",");
16.         if(tokens.length>11)
17.         {
18.
19.             String Fbicode=tokens[14];
20.
21.             if(Fbicode.equals("32"))
22.                 context.write(new Text(Fbicode), new IntWritable(1));
23.         }
24.     }
25. }
26. }
```

Reduce.java

```
1. import java.io.IOException;
2. import org.apache.hadoop.io.*;
3. import org.apache.hadoop.mapreduce.*;
4.
5. public class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
6.     public void reduce(Text key, Iterable<IntWritable> values,
7. Context context) throws IOException, InterruptedException {
8.     int count=0;
9.     for(IntWritable record:values){
10.
11.         count++;
12.
13.
14.     }
15.     context.write(new Text("Number of crimes investigated for FBI code:"+key+" is"), new Int
Writable(count));
16.
17. }
18. }
```

Lab 18. Pig UDF

1. Executing Pig UDF
- 2.
3. Step 1: From Terminal
4. Create a comma separated file in Linux FS by name "employee.txt"
5. Copy this file in HDFS under a newly created folder
6. "employee_input"
- 7.
8. Step 2: From Eclipse
9. Create a project by name "uppercase".
10. Under the "src" folder of the project create a class by name
11. "UpperCase" with package as "default".
12. Paste the given code inside the class.
- 13.
14. Step 3: To Remove Errors
15. Go to Properties => Build Path => Libraries => Add External Jars.
- 16.
17. Go to location /usr/lib/pig and add the available jar files.
- 18.
19. Export the project as "ucase.jar".
- 20.
- 21.
22. Step 4: In Pig type the following commands
23. register /home/wisdom/ucase.jar;
- 24.
25. emp_data = LOAD '/user/wisdom/employee_input/employee.txt' using
26. PigStorage(',') AS (emp_name:chararray, emp_id:
27. int, salary:float);
- 28.
29. upper_case_name = FOREACH emp_data GENERATE UpperCase(emp_name);
- 30.
31. dump upper_case_name;

Source Code

```
1. package udf_proj;
2. import java.io.IOException;
3. import org.apache.pig.EvalFunc;
4. import org.apache.pig.data.Tuple;
5.
6. public class Upper extends EvalFunc<String>
7. {
8.     public String exec(Tuple input) throws IOException {
9.         if (input==null || input.size()==0)
10.             return null;
11.         try{
12.             String str = (String)input.get(0);
13.             return str.toUpperCase();
14.         }catch(Exception e){
15.             throw new IOException("Caught exception processing input row ", e);
16.         }
17.     }
18. }
```

Lab 19. Parsing XML File using Pig

pig_xml.jar location

1. https://drive.google.com/open?id=1NliSHI0pVm0adUgLlgv-cBWwzVaVc_9t

Procedure

1. Preparing the Input
- 2.
3. Take the below sample lines from core-site.xml as input file and Copy this file to hdfs
- 4.
5. `<configuration>`
6. `<property>`
7. `<name>fs.default.name</name>`
8. `<value>hdfs://localhost:8020</value>`
9. `</property>`
10. `</configuration>`
- 11.
- 12.
13. Execute the XML Loader
14. `register /home/wisdom/pig_xml.jar`
- 15.
16. `xml_input_data = load '/home/wisdom/core_site_xml' using pig.XML.newloader('property') as (doc:chararray);`
- 17.
18. `data_from_xml_ip = foreach a GENERATE FLATTEN(REGEX_EXTRACT_ALL(doc,'<property>\s*<name>(.*)</name>\s*<value>(.*)</value>\s*</property>'));`
19. Output
- 20.
21. `(fs.defaultFS,hdfs://localhost:8020)`

Lab 20. HBase CRUD Operations

create_delete_table.java

```
1. package com.javaapi;
2.
3. import java.io.IOException;
4.
5. import org.apache.hadoop.conf.Configuration;
6. import org.apache.hadoop.hbase.HBaseConfiguration;
7. import org.apache.hadoop.hbase.HColumnDescriptor;
8. import org.apache.hadoop.hbase.HTableDescriptor;
9. import org.apache.hadoop.hbase.MasterNotRunningException;
10. import org.apache.hadoop.hbase.ZooKeeperConnectionException;
11. import org.apache.hadoop.hbase.client.HBaseAdmin;
12. import org.apache.hadoop.hbase.util.Bytes;
13.
14. public class Creat_Delete_Table {
15.     Configuration conf=HBaseConfiguration.create();
16.
17.     public void createtable(String name,String[] colfamily) throws MasterNotRunningException,
18.         n,ZooKeeperConnectionException, IOException{
19.
20.     HBaseAdmin admin=new HBaseAdmin(conf);
21.     HTableDescriptor des=new HTableDescriptor(Bytes.toBytes(name));
22.
23.     for (int i = 0; i < colfamily.length; i++) {
24.         des.addFamily(new HColumnDescriptor(colfamily[i]));
25.     }
26.
27.
28.
29.     if(admin.tableExists(name)){
30.
31.         System.out.println("Table Already exists");
32.     }
33.     else{
34.
35.         admin.createTable(des);
36.     }
```

```
37. System.out.println("Table:" + name + " sucessfully created");
38. }
39.
40. }
41.
42. public void deleteTable(String tablename) throws MasterNotRunningException, ZooKeeperException, IOException{
43. HBaseAdmin admin=new HBaseAdmin(conf);
44. if(admin.tableExists(tablename)){
45. admin.disableTable(tablename);
46. admin.deleteTable(tablename);
47. System.out.println("Table: " +tablename+" deleted");
48. }
49. else{
50. System.out.println("Table dosenot exists");
51. }
52. }
53. public static void main(String[] args) throws MasterNotRunningException, ZooKeeperConnectionException, IOException {
54. TableOps op=new TableOps();
55. String tablename = "driver";
56. String[] familys = { "trip", "customer" };
57. op.createtable(tablename, familys);
58. //op.deleteTable(tablename);
59. }
60. }
```

[put_get_example.java](#)

```
1. package com.javaapi;
2.
3. import java.io.IOException;
4. import java.util.ArrayList;
5. import java.util.List;
6.
7. import org.apache.hadoop.conf.Configuration;
8. import org.apache.hadoop.hbase.HBaseConfiguration;
9. import org.apache.hadoop.hbase.KeyValue;
10. import org.apache.hadoop.hbase.client.Delete;
11. import org.apache.hadoop.hbase.client.Get;
12. import org.apache.hadoop.hbase.client.HTable;
13. import org.apache.hadoop.hbase.client.Put;
14. import org.apache.hadoop.hbase.client.Result;
15. import org.apache.hadoop.hbase.client.ResultScanner;
16. import org.apache.hadoop.hbase.client.Scan;
17. import org.apache.hadoop.hbase.util.Bytes;
18.
19. public class Put_Get_example {
20.     Configuration conf=HBaseConfiguration.create();
21.     public void addRecord(String tableName, String rowKey,
22.             String family, String qualifier, String value) throws Exception {
23.         try {
24.             HTable table = new HTable(conf, tableName);
25.             Put put = new Put(Bytes.toBytes(rowKey));
26.             put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes
27.                     .toBytes(value));
28.             table.put(put);
29.             // System.out.println("insert record " + rowKey + " to table "
30.             // + tableName + " ok.");
31.         } catch (IOException e) {
32.             e.printStackTrace();
33.         }
34.     }
35. }
36. public void getonerecord(String tablename, String rowkey) throws IOException{
37.     HTable table=new HTable(conf, tablename);
38.     Get get=new Get(Bytes.toBytes(rowkey));
39.     Result rs=table.get(get);
```

```

40.    for(KeyValue kv : rs.raw()){
41.        System.out.print(new String(kv.getRow()) + " ");
42.        System.out.print(new String(kv.getFamily()) + ":" );
43.        System.out.print(new String(kv.getQualifier()) + " ");
44.        System.out.print(kv.getTimestamp() + " ");
45.        System.out.println(new String(kv.getValue()));
46.    }
47. }
48. public void delRecord(String tableName, String rowKey)
49.     throws IOException {
50.     HTable table = new HTable(conf, tableName);
51.     List<Delete> list = new ArrayList<Delete>();
52.     Delete del = new Delete(rowKey.getBytes());
53.     list.add(del);
54.     table.delete(list);
55.     System.out.println("del record " + rowKey + " ok.");
56. }
57. public void getAllRecords(String tablename) throws IOException{
58.     HTable table=new HTable(conf,tablename);
59.     Scan sc=new Scan();
60.     ResultScanner rs=table.getScanner(sc);
61.     System.out.println("\n\nGet all records");
62.     for(Result r:rs){
63.         for(KeyValue kv : r.raw()){
64.             System.out.print(new String(kv.getRow()) + " ");
65.             System.out.print(new String(kv.getFamily()) + ":" );
66.             System.out.print(new String(kv.getQualifier()) + " ");
67.             System.out.print(kv.getTimestamp() + " ");
68.             System.out.println(new String(kv.getValue()));
69.         }
70.     }
71. }
72. }
73. public static void main(String[] args) throws Exception {
74.     Put_Get_example op=new Put_Get_example();
75.     String tablename="driver";
76.
77.     //first record
78.     op.addRecord(tablename, "d1", "trip", "id", "102");
79.     op.addRecord(tablename, "d1", "trip", "x", "-12");
80.     op.addRecord(tablename, "d1", "customer", "name", "Pradeep");

```

```
81.    op.addRecord(tablename, "d1", "customer", "phone", "190208020");
82.    //second record
83.    op.addRecord(tablename, "d2", "trip", "id", "103");
84.    op.addRecord(tablename, "d2", "trip", "x", "-15");
85.    op.addRecord(tablename, "d2", "customer", "name", "Kiran");
86.    op.addRecord(tablename, "d2", "customer", "phone", "190208020");
87.    //display one record
88.    // op.getOneRecord("driver", "d1");
89.
90.    //get all records.
91.    //op.getAllRecords(tablename);
92.    op.delRecord(tablename, "d1");
93. }
94. }
```

[tableOps.java](#)

```
1. package com.javaapi;
2.
3. import java.io.IOException;
4.
5. import org.apache.hadoop.conf.Configuration;
6. import org.apache.hadoop.hbase.HBaseConfiguration;
7. import org.apache.hadoop.hbase.HColumnDescriptor;
8. import org.apache.hadoop.hbase.HTableDescriptor;
9. import org.apache.hadoop.hbase.KeyValue;
10. import org.apache.hadoop.hbase.MasterNotRunningException;
11. import org.apache.hadoop.hbase.ZooKeeperConnectionException;
12. import org.apache.hadoop.hbase.client.Get;
13. import org.apache.hadoop.hbase.client.HBaseAdmin;
14. import org.apache.hadoop.hbase.client.HTable;
15. import org.apache.hadoop.hbase.client.Put;
16. import org.apache.hadoop.hbase.client.Result;
17. import org.apache.hadoop.hbase.client.ResultScanner;
18. import org.apache.hadoop.hbase.client.Scan;
19. import org.apache.hadoop.hbase.util.Bytes;
20.
21. public class TableOps {
22.     Configuration conf=HBaseConfiguration.create();
23.
24.     public void createtable(String name,String[] colfamily) throws MasterNotRunningException,
25.         ZooKeeperConnectionException, IOException{
26.
27.         HBaseAdmin admin=new HBaseAdmin(conf);
28.         HTableDescriptor des=new HTableDescriptor(Bytes.toBytes(name));
29.
30.         for (int i = 0; i < colfamily.length; i++) {
31.             des.addFamily(new HColumnDescriptor(colfamily[i]));
32.         }
33.
34.         if(admin.tableExists(name)){
35.
36.             System.out.println("Table Already exists");
37.
38.         }
39.     }
40. }
```

```
39.    }
40.  else{
41.
42.    admin.createTable(des);
43.
44.    System.out.println("Table:" + name + " sucessfully created");
45.  }
46.
47. }
48. public void addRecord(String tableName, String rowKey,
49.   String family, String qualifier, String value) throws Exception {
50. try{
51.   HTable table = new HTable(conf, tableName);
52.   Put put = new Put(Bytes.toBytes(rowKey));
53.   put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes
54.     .toBytes(value));
55.   table.put(put);
56.   System.out.println("insert record " + rowKey + " to table "
57.     + tableName + " ok.");
58. } catch (IOException e) {
59.   e.printStackTrace();
60. }
61.
62. }
63. public void deleteTable(String tablename) throws MasterNotRunningException, ZooKeeper
rConnectionException, IOException{
64. HBaseAdmin admin=new HBaseAdmin(conf);
65. if(admin.tableExists(tablename)){
66. admin.disableTable(tablename);
67. admin.deleteTable(tablename);
68. System.out.println("Table: "+tablename+" deleted");
69. }
70. else{
71.   System.out.println("Table does not exists");
72. }
73. }
74. public void getonerecord(String tablename, String rowkey) throws IOException{
75. HTable table=new HTable(conf, tablename);
76. Get get=new Get(Bytes.toBytes(rowkey));
77. Result rs=table.get(get);
78. for(KeyValue kv : rs.raw()){


```

```

79.     System.out.print(new String(kv.getRow()) + " ");
80.     System.out.print(new String(kv.getFamily()) + ":" );
81.     System.out.print(new String(kv.getQualifier()) + " ");
82.     System.out.print(kv.getTimestamp() + " ");
83.     System.out.println(new String(kv.getValue()));
84. }
85. }
86. public void getAllRecords(String tablename) throws IOException{
87.     HTable table=new HTable(conf,tablename);
88.     Scan sc=new Scan();
89.     ResultScanner rs=table.getScanner(sc);
90.     for(Result r:rs){
91.         for(KeyValue kv : r.raw()){
92.             System.out.print(new String(kv.getRow()) + " ");
93.             System.out.print(new String(kv.getFamily()) + ":" );
94.             System.out.print(new String(kv.getQualifier()) + " ");
95.             System.out.print(kv.getTimestamp() + " ");
96.             System.out.println(new String(kv.getValue()));
97.         }
98.     }
99. }
100. }
101. public static void main(String[] args) throws Exception {
102.
103.     TableOps op=new TableOps();
104.     String tablename = "driver";
105.     String[] familys = { "trip", "customer" };
106.     op.createtable(tablename, familys);
107.
108.
109.     /*
110.      //first record
111.      op.addRecord(tablename, "d1", "trip", "id", "102");
112.      op.addRecord(tablename, "d1", "trip", "x", "-12");
113.      op.addRecord(tablename, "d1", "customer", "name", "Pradeep");
114.      op.addRecord(tablename, "d1", "customer", "phone", "190208020");
115.      //second record
116.      op.addRecord(tablename, "d2", "trip", "id", "103");
117.      op.addRecord(tablename, "d2", "trip", "x", "-15");
118.      op.addRecord(tablename, "d2", "customer", "name", "Kiran");
119.      op.addRecord(tablename, "d2", "customer", "phone", "190208020");*/

```

```
120.     // op.getonerecord("driver", "d1");  
121.  
122.     op.deleteTable("driver");  
123.     // op.getAllRecords(tablename);  
124.  
125. }  
126. }
```

Lab 21. Hbase Thrift server using python

1. Install python dependencies **for centos**
- 2.
3. `sudo yum install boost-devel php-devel pcre-devel automake libtool flex bison pkgconfig gcc-c++ boost-devel libevent-devel zlib-devel python-devel ruby-devel libtool*`
- 4.
- 5.
6. Download and install thrift server
- 7.
8. Wget <https://archive.apache.org/dist/thrift/0.6.0/thrift-0.6.0.tar.gz>
9. `$ tar xfz thrift-0.6.1.tar.gz`
10. `$ cd thrift-0.6.1`
11. `$./configure`
12. `$ make`
13. `$ make install`
14. `$ cd ..`
- 15.
16. After performance of a configuration you should see what languages at you to be supported Thrift th:
 - 17. Building C++ Library : no
 - 18. Building C (GLib) Library : no
 - 19. Building Java Library : no
 - 20. Building C# Library : no
 - 21. Building Python Library : yes
 - 22. Building Ruby Library : no
 - 23. Building Haskell Library : no
 - 24. Building Perl Library : no
 - 25. Building PHP Library : yes
 - 26. Building Erlang Library : yes
- 27.
- 28.
29. Generate HBase thrift python module
30. Once **this** is done, you should have `thrift` in your path.
31. `thrift --gen py Hbase.thrift`
32. This command will create `gen-py` folder in your `thrift` folder
- 33.
34. **5)** Add HBase thrift python module to `pythonpath`
35. `export PYTHONPATH=$PYTHONPATH:/<path to gen-py>`
- 36.

- 37.
38. **6) Start HBase thrift server**
39. You can simply start the thrift server by executing the following command:
40. `$HBASE_HOME/bin/hbase thrift start`
41. This will start HBase thrift server on port **9090** (**default** port).
- 42.
- 43.
44. **7) Use the client!**
45. Here is a sample code that will print all the table names on your HBase server:
46. `from thrift.transport.TSocket import TSocket`
47. `from thrift.transport.TTransport import TBufferedTransport`
48. `from thrift.protocol import TBinaryProtocol`
49. `from hbase import Hbase`
- 50.
51. `transport = TBufferedTransport(TSocket('localhost', 9090))`
52. `transport.open()`
53. `protocol = TBinaryProtocol.TBinaryProtocol(transport)`
54. `client = Hbase.Client(protocol)`
55. `print(client.getTableNames())`

Lab 22. Scala Installation

1. \$sudo wget <http://www.scala-lang.org/files/archive/scala-2.10.4.tgz>
2. \$sudo tar xvf scala-**2.10.4**.tgz
3. \$sudo vi ~/.bashrc
4. //Add the following lines
5. export SCALA_HOME=/home/jayantmohite/scala-**2.10.4**
6. export PATH=\$PATH:\$SCALA_HOME/bin
7. \$source ~/.bashrc
8. \$scala -version

Lab 23. Spark Installation

1. \$sudo wget <https://archive.apache.org/dist/spark/spark-2.0.0/spark-2.0.0-bin-hadoop2.6.tgz>
2. \$tar xvf spark-2.0.0-bin-hadoop2.6.tgz
3. \$sudo vi ~/.bashrc
4. //Add the following lines
5. export SPARK_HOME=/home/jayantmohite/spark-2.0.0-bin-hadoop2.6/
6. export PATH=\$PATH:\$SPARK_HOME/bin
7. \$source ~/.bashrc
8. \$spark-shell

Lab 24. Spark RDD Operations

```
1. ****
2. Create RDD
3. ****
4. scala> val file = sc.textFile("/home/jayantmohite/sample.txt")
5.
6. scala> val centosLines = file.filter(line => line.contains("CentOS"))
7.
8. scala> centosLines.count
9.
10. scala> centosLines.first
11.
12. scala> centosLines.take(2)
13.
14. scala> centosLines.saveAsTextFile("/home/jayantmohite/results.txt")
15.
16. ****
17. Map Functions
18. ****
19. scala> sc.parallelize(List(1,2,3)).map(x=>List(x,x,x)).collect
20.
21. scala> sc.parallelize(List(1,2,3)).flatMap(x=>List(x,x,x)).collect
22.
23. scala> val parallel = sc.parallelize(1 to 9, 3)
24.
25. scala> parallel.mapPartitions( x => List(x.next).iterator).collect
26.
27. scala> parallel.mapPartitionsWithIndex( (index: Int, it: Iterator[Int])
28.     | => it.toList.map(x=> index + "|" +x).iterator).collect
29.
30. ****
31. Sample
32. ****
33.
34. scala> parallel.sample(true,0.2).collect
35.
36. scala> parallel.sample(false,0.2).collect
37.
38. scala> parallel.takeSample(true,2)
39.
```

```
40. scala> parallel.takeSample(false,2)
41.
42. ****
43. Union
44. ****
45. scala> val parallel = sc.parallelize(1 to 9)
46.
47. scala> val par2 = sc.parallelize(5 to 15)
48.
49. scala> parallel.union(par2).collect
50. scala> parallel.intersection(par2).collect
51. scala> parallel.union(par2).distinct.collect
52.
53.
54. ****
55. Intersection
56. ****
57. scala> val parallel = sc.parallelize(1 to 9)
58.
59. scala> val par2 = sc.parallelize(5 to 15)
60.
61. scala> parallel.intersection(par2).collect
62.
63. ****
64. Distinct
65. ****
66. scala> val parallel = sc.parallelize(1 to 9)
67.
68. scala> val par2 = sc.parallelize(5 to 15)
69.
70. scala> parallel.union(par2).distinct.collect
71.
72. ****
73. Group By, Reduce By, Sort By Keys
74. ****
75. scala> val empNames = sc.textFile("/home/jayantmohite/employees.csv")
76.
77. scala> val rows = empNames.map(line => line.split(","))
78.
79. scala> val countriesToName = rows.map(x => (x(3),x(1)))
80.
```

```
81. scala> countriesToName.groupByKey.collect
82.
83. scala> rows.map(x => (x(3),x(2).toInt)).reduceByKey((v1,v2) => v1 + v2).collect
84.
85. scala> rows.map(x => (x(3),x(1))).sortByKey().collect
86.
87. scala> rows.map(x => (x(3),x(1))).sortByKey(false).collect
88.
89. ****
90. Joins on Duplicate Fields
91. ****
92. scala> val rdd1 = sc.parallelize(Seq(
93.   |   ("math", 55),
94.   |   ("math", 56),
95.   |   ("english", 57),
96.   |   ("english", 58),
97.   |   ("science", 59),
98.   |   ("science", 54)))
99.
100.    scala> val rdd2 = sc.parallelize(Seq(
101.      |   ("math", 60),
102.      |   ("math", 65),
103.      |   ("science", 61),
104.      |   ("science", 62),
105.      |   ("history", 63),
106.      |   ("history", 64)))
107.
108.    scala> val joined = rdd1.join(rdd2)
109.
110.    scala> joined.collect()
111.
112.    scala> val leftJoined = rdd1.leftOuterJoin(rdd2)
113.
114.    scala> leftJoined.collect()
115.
116.    scala> val rightJoined = rdd1.rightOuterJoin(rdd2)
117.
118.    scala> rightJoined.collect()
119.
120. ****
121. Joins on Non Duplicate Fields
```

```
122. Perform As Root
123. *****/
124. scala> val left=Seq(("bob", "2015-01-13", 4), ("alice", "2015-04-
23", 10)).toDF("name","date","duration")
125.
126. scala> val right=Seq(("alice", 100), ("bob", 23)).toDF("name","upload")
127.
128. scala> val df = left.join(right, left.col("name") === right.col("name"))
129.
130. scala> df.collect
```

Lab 25. Spark SQL

1. Create a file by name employee.json with the following content
2.

```
{"name":"John", "age":28}
```
3.

```
{"name":"Andrew", "age":36}
```
4.

```
{"name":"Clarke", "age":22}
```
5.

```
{"name":"Kevin", "age":42}
```
6.

```
{"name":"Richard", "age":51}"
```
- 7.
8.

```
*****
```
9. **SPARK SQL as Data Frame**
10.

```
*****
```
11.

```
scala> import org.apache.spark.sql.SparkSession
```
12.

```
scala> val spark = SparkSession.builder().appName("Spark SQL basic example").config("spark.some.config.option", "some-value").getOrCreate()
```
13.

```
scala> import spark.implicits._
```
14.

```
scala> val df = spark.read.json("/home/jayantmohite/employee.json")
```
15.

```
scala> df.show()
```
16.

```
scala> df.printSchema()
```
17.

```
scala> df.select("name").show()
```
18.

```
scala> df.select($"name", $"age" + 2).show()
```
19.

```
scala> df.filter($"age" > 30).show()
```
20.

```
scala> df.groupBy("age").count().show()
```
- 21.
- 22.
23.

```
*****
```
24. **SPARK SQL as HQL**
25.

```
*****
```
26.

```
scala> import org.apache.spark.sql.Row
```
27.

```
scala> import org.apache.spark.sql.SparkSession
```
28.

```
scala> case class Record(key: Int, value: String)
```
29.

```
scala> val warehouseLocation = "spark-warehouse"
```
30.

```
scala> val spark = SparkSession.builder().appName("Spark Hive Example").config("spark.sql.warehouse.dir", warehouseLocation).enableHiveSupport().getOrCreate()
```
31.

```
scala> import spark.implicits._
```
32.

```
scala> import spark.sql
```
33.

```
scala> sql("CREATE TABLE IF NOT EXISTS student (id INT, name STRING) row format delimited fields terminated by '\t' stored as textfile")
```
34.

```
scala> sql("LOAD DATA LOCAL INPATH '/home/jayantmohite/student.txt' INTO TABLE student")
```
35.

```
scala> sql("SELECT * FROM student").show()
```

Lab 26. Spark Hive Configuration

1. \$sudo cp /home/jayantmohite/spark-**2.0.0**-bin-hadoop**2.6**/conf/spark-env.sh.template /home/jayantmohite/spark-**2.0.0**-bin-hadoop**2.6**/conf/spark-env.sh
- 2.
3. \$sudo gedit /home/jayantmohite/spark-**2.0.0**-bin-hadoop**2.6**/conf/spark-env.sh
4. At the end of **this** file add the following
5. export HIVE_HOME=/usr/bin/hive
- 6.
7. \$sudo cp /usr/lib/hive/conf/hive-site.xml /home/jayantmohite/spark-**2.0.0**-bin-hadoop**2.6**/conf/
- 8.
9. \$sudo yum install hive-metastore
10. \$sudo service hive-metastore start
- 11.
12. \$su
13. #cd /home/jayantmohite/spark-**2.0.0**-bin-hadoop**2.6**/sbin
14. #./start-all.sh
15. #cd
16. #cd /home/jayantmohite/spark-**2.0.0**-bin-hadoop**2.6**/bin
17. #spark-shell

Lab 27. Spark HQL

```
1. ****
2. SPARK HIVE
3. ****
4. scala> import org.apache.spark.sql.hive.HiveContext
5. scala> val hiveContext = new org.apache.spark.sql.hive.HiveContext(sc)
6. scala> import hiveContext._
7. scala> val hive_tables=hiveContext.sql("show tables")
8. scala> hive_tables.collect
9. scala> val limited_rows=hiveContext.sql("select * from emp limit 2")
```

20. Cheat Sheets

Pig Cheat Sheet

Basic Operators

Operator	Description	Example
Arithmetic Operators	+, -, *, /, %, ?:	X = FOREACH A GENERATE f1, f2, f1%f2; X = FOREACH A GENERATE f2, (f2==1?1:COUNT(B));
Boolean Operators	and, or, not	X = FILTER A BY (f1==8) OR (NOT (f2+f3 > f1));
Cast Operators	Casting from one datatype to another	B = FOREACH A GENERATE (int)\$0 + 1; B = FOREACH A GENERATE \$0 + 1, \$1 + 1.0
Comparison Operators	==, !=, >, <, >=, <=, matches	X = FILTER A BY (f1 == 8); X = FILTER A BY (f2 == 'apache'); X = FILTER A BY (f1 matches '.*apache.*');
Construction Operators	Used to construct tuple (), bag {} and map []	B = foreach A generate (name, age); B = foreach A generate {(name, age)}, {name, age}; B = foreach A generate [name, gpa];
Dereference Operators	dereference tuples (tuple.id or tuple.(id, …)), bags (bag.id or bag.(id, …)) and maps (map#‘ key’)	X = FOREACH A GENERATE f2.t1,f2.t3 (dereferencing is used to retrieve two fields from tuple f2)
Disambiguate Operator	(::) used to identify field names after JOIN, COGROUP, CROSS, or FLATTEN operators	A = load 'data1' as (x, y); B = load 'data2' as (x, y, z); C = join A by x, B by x; D = foreach C generate A::y;

Flatten Operator	Flatten un-nests tuples as well as bags	consider a relation that has a tuple of the form (a, (b, c)). The expression GENERATE \$0, flatten(\$1), will cause that tuple to become (a, b, c).
Null Operator	is null, is not null	X = FILTER A BY f1 is not null;
Sign Operators	+ -> has no effect, - -> changes the sign of a positive/negative number	A = LOAD 'data' as (x, y, z); B = FOREACH A GENERATE -x, y;

Relational Operators

Operator	Description	Example
COGROUP/GROUP	Groups the data in one or more relations. The COGROUP operator groups together tuples that have the same group key (key field)	A = load 'student' AS (name:chararray,age:int,gpa:float); B = GROUP A BY age;
CROSS	Computes the cross product of two or more relations	X = CROSS A,B A = (1, 2, 3) B = (2, 4) DUMP X; (4, 2, 1) (8, 9) (1,2,3,2,4) (1, 3) (1,2,3,8,9) (1,2,3,1,3) (4,2,1,2,4) (4,2,1,8,9) (4,2,1,1,3)
DEFINE	Assigns an alias to a UDF or streaming command.	DEFINE CMD 'perl PigStreaming.pl - nameMap' input(stdin using PigStreaming('')) output(stdout using PigStreaming('')); A = LOAD 'file' ; B = STREAM B THROUGH CMD;
DISTINCT	Removes duplicate tuples in a relation.	X = DISTINCT A; A = (8,3,4) DUMP X; (1,2,3) (4,3,3) (4,3,3) (4,3,3) (8,3,4) (1,2,3)
FILTER	Selects tuples from a relation based on some condition.	X = FILTER A BY f3 == 3; A = (1,2,3) DUMP X; (4,5,6) (1,2,3) (7,8,9)

		(4,3,3) (4,3,3) (8,4,3) (8,4,3)
FOREACH	Generates transformation of data for each row as specified	X = FOREACH A GENERATE a1, a2; A = (1,2,3) DUMP X; (4,2,5) (1,2) (8,3,6) (4,2) (8,3)

IMPORT	Import macros defined in a separate file.	<pre>/* myscript.pig */ IMPORT 'my_macro.pig' ;</pre>
JOIN	Performs an inner join of two or more relations based on common field values.	<pre>X = JOIN A BY a1, B BY b1; DUMP X (1,2,1,3) A = (1,2) B = (1,3) (1,2,1,2) (4,5) (1,2) (4,5,4,7) (4,7)</pre>
LOAD	Loads data from the file system.	<pre>A = LOAD 'myfile.txt' ; LOAD 'myfile.txt' AS (f1:int, f2:int, f3:int);</pre>
MAPREDUCE	Executes native MapReduce jobs inside a Pig script.	<pre>A = LOAD 'WordcountInput.txt' ; B = MAPREDUCE 'wordcount.jar' STORE A INTO 'inputDir' LOAD 'outputDir' AS (word:chararray, count: int) 'org.myorg.WordCount inputDir outputDir';</pre>
ORDERBY	Sorts a relation based on one or more fields.	<pre>A = LOAD 'mydata' AS (x: int, y: map[]); B = ORDER A BY x;</pre>
SAMPLE	Partitions a relation into two or more relations, selects a random data sample with the stated sample size.	<p>Relation X will contain 1% of the data in relation A.</p> <pre>A = LOAD 'data' AS (f1:int,f2:int,f3:int); X = SAMPLE A 0.01;</pre>
SPLIT	Partitions a relation into two or more relations based on some expression.	<pre>SPLIT input_var INTO output_var IF (field1 is not null), ignored_var IF (field1 is null);</pre>
STORE	Stores or saves results to the file system.	<pre>STORE A INTO 'myoutput' USING PigStorage ('*');</pre>

		1*2*3 4*2*1
STREAM	Sends data to an external script or program	A = LOAD 'data' ; B = STREAM A THROUGH 'stream.pl -n 5';
UNION	Computes the union of two or more relations. (Does not preserve the order of tuples)	X = UNION A, B; A = (1,2,3) B = (2,4) DUMP X; (4,2,1) (8,9) (1,2,3) (1,3) (4,2,1) (2,4) (8,9) (1,3)

Functions

Function	Syntax	Description
AVG	AVG(expression)	Computes the average of the numeric values in a single-column bag.
CONCAT	CONCAT (expression, expression)	Concatenates two expressions of identical type.
COUNT	COUNT(expression)	Computes the number of elements in a bag, it ignores null.
COUNT_STAR	COUNT_STAR(expression)	Computes the number of elements in a bag, it includes null.
DIFF	DIFF (expression, expression)	Compares two fields in a tuple, any tuples that are in one bag but not the other are returned in a bag.
DIFF	DIFF (expression, expression)	Compares two fields in a tuple, any tuples that are in one bag but not the other are returned in a bag.
IsEmpty	IsEmpty(expression)	Checks if a bag or map is empty.
MAX	MAX(expression)	Computes the maximum of the numeric values or chararrays in a single-column bag
MIN	MIN(expression)	Computes the minimum of the numeric values or chararrays in a single-column bag.
SIZE	SIZE(expression)	Computes the number of elements based on any Pig data type. SIZE includes NULL values in the size computation

SUM	SUM(expression)	Computes the sum of the numeric values in a single-column bag.
TOKENIZE	TOKENIZE(expression [, 'field_delimiter'])	Splits a string and outputs a bag of words.

Load/Store Functions

FUnction	Syntax	Description
Handling Compression	A = load 'myinput.gz' ; store A into 'myoutput.gz' ;	PigStorage and TextLoader support gzip and bzip compression for both read (load) and write (store). BinStorage does not support compression.
BinStorage	A = LOAD 'data' USING BinStorage();	Loads and stores data in machine-readable format.
JsonLoader, JsonStorage	A = load 'a.json' using JsonLoader();	Load or store JSON data.
PigDump	STORE X INTO 'output' USING PigDump();	Stores data in UTF-8 format.
PigStorage	A = LOAD 'student' USING PigStorage('t') AS (name: chararray, age:int, gpa: float);	Loads and stores data as structured text files.
TextLoader	A = LOAD 'data' USING TextLoader();	Loads unstructured data in UTF-8 format.

Math Functions

Operator	Description	Example
ABS	ABS(expression)	Returns the absolute value of an expression. If the result is not negative ($x \geq 0$), the result is returned. If the result is negative ($x < 0$), the negation of the result is returned.
ACOS	ACOS(expression)	Returns the arc cosine of an expression.
ASIN	ASIN(expression)	Returns the arc sine of an expression.
ATAN	ATAN(expression)	Returns the arc tangent of an expression.
CBRT	CBRT(expression)	Returns the cube root of an expression.
CEIL	CEIL(expression)	Returns the value of an expression rounded up to the nearest integer. This function never decreases the result value.
COS	COS(expression)	Returns the trigonometric cosine of an expression.
COSH	COSH(expression)	Returns the hyperbolic cosine of an expression.
EXP	EXP(expression)	Returns Euler's number e raised to the power of x.
FLOOR	FLOOR(expression)	Returns the value of an expression rounded down to the nearest integer. This function never increases the result value.
LOG	LOG(expression)	Returns the natural logarithm (base e) of an expression.
LOG10	LOG10(expression)	Returns the base 10 logarithm of an expression.
RANDOM	RANDOM()	Returns a pseudo random number (type double) greater than or equal to 0.0 and less than 1.0.

ROUND	ROUND(expression)	Returns the value of an expression rounded to an integer (if the result type is float) or rounded to a long (if the result type is double).
SIN	SIN(expression)	Returns the sine of an expression.
SINH	SINH(expression)	Returns the hyperbolic sine of an expression.
SQRT	SQRT(expression)	Returns the positive square root of an expression.
TAN	TAN(expression)	Returns the trigonometric tangent of an angle.
TANH	TANH(expression)	Returns the hyperbolic tangent of an expression.

String Functions

Operator	Description	Example
INDEXOF	INDEXOF(string, 'character' , startIndex)	Returns the index of the first occurrence of a character in a string, searching forward from a start index.
LAST_INDEX	LAST_INDEX_OF(expression)	Returns the index of the last occurrence of a character in a string, searching backward from a start index.
LCFIRST	LCFIRST(expression)	Converts the first character in a string to lower case.
LOWER	LOWER(expression)	Converts all characters in a string to lower case.
REGEX_EXTRACT	REGEX_EXTRACT (string, regex, index)	Performs regular expression matching and extracts the matched group defined by an index parameter. The function uses Java regular expression form.
REGEX_EXTRACT_ALL	REGEX_EXTRACT (string, regex)	Performs regular expression matching and extracts all matched groups. The function uses Java regular expression form.
REPLACE	REPLACE(string, 'oldChar' , 'newChar');	Replaces existing characters in a string with new characters.
STRSPLIT	STRSPLIT(string, regex, limit)	Splits a string around matches of a given regular expression.

SUBSTRING	SUBSTRING(string, startIndex, stopIndex)	Returns a substring from a given string.
TRIM	TRIM(expression)	Returns a copy of a string with leading and trailing white space removed.
UCFIRST	UCFIRST(expression)	Returns a string with the first character converted to upper case.
UPPER	UPPER(expression)	Returns a string converted to upper case.

Tuple, Bag, Map Functions

Operator	Description	Example
TOTUPLE	TOTUPLE(expression [, expression …])	Converts one or more expressions to type tuple.
TOBAG	TOBAG(expression [, expression …])	Converts one or more expressions to individual tuples which are then placed in a bag.
TOMAP	TOMAP(key-expression, value-expression [, key-expression, value-expression …])	Converts key/value expression pairs into a map. Needs an even number of expressions as parameters. The elements must comply with map type rules.
TOP	TOP(topN,column,relation)	Returns the top-n tuples from a bag of tuples.

Data Types

SIMPLE TYPES

Operator	Description	Example
int	Signed 32-bit integer	10
long	Signed 64-bit integer	Data: 10L or 10l Display: 10L
float	32-bit floating point	Data: 10.5F or 10.5f or 10.5e2f or 10.5E2F Display: 10.5F or 1050.0F
double	64-bit floating point	Data: 10.5 or 10.5e2 or 10.5E2 Display: 10.5 or 1050.0
chararray	Character array (string) in Unicode UTF-8 format	hello world
bytearray	Byte array (blob)	
boolean	boolean	true/false (case insensitive)

Complex Types

Operator	Description	Example
tuple	An ordered set of fields.	(19,2)
bag	An collection of tuples.	{(19,2), (18,1)}
map	A set of key value pairs.	[name#John,phone#5551212]

Hive Cheat Sheet

Date Functions

The following built-in date functions are supported in Hive:

Return Type	Name(Signature)	Example
string	from_unixtime(bigint unixtime[, string format])	Converts the number of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00"
bigint	unix_timestamp()	Gets current time stamp using the default time zone.
bigint	unix_timestamp(string date)	Converts time string in format yyyy-MM-dd HH:mm:ss to Unix time stamp, return 0 if fail: unix_timestamp('2009-03-20 11:30:01') = 1237573801
bigint	unix_timestamp(string date, string pattern)	Convert time string with given pattern to Unix time stamp, return 0 if fail: unix_timestamp('2009-03-20' , 'yyyy-MM-dd') = 1237532400
string (pre- 2.1.0) date (2.1.0 on)	to_date(string timestamp)	Returns the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01"
int	year(string date)	Returns the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970

int	month(string date)	Returns the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11
int	day(string date) dayofmonth(date)	Return the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1
int	hour(string date)	Returns the hour of the timestamp: hour('2009-07-30 12:58:59') = 12, hour(' 12:58:59') = 12
int	minute(string date)	Returns the minute of the timestamp
int	second(string date)	Returns the second of the timestamp
int	weekofyear(string date)	Return the week number of a timestamp string: weekofyear("1970-11-01 00:00:00") = 44, weekofyear("1970-11-01") = 44
int	datediff(string enddate, string startdate)	Return the number of days from startdate to enddate: datediff('2009-03-01' , '2009-02-27') = 2
string (pre- 2.1.0) date (2.1.0 on)	date_add(date/timestamp/ string startdate, tinyint/smallint/int days)	Add a number of days to startdate: date_add('2008-12-31' , 1) = '2009-01-01'
string (pre- 2.1.0) date (2.1.0 on)	date_sub(date/timestamp/ string startdate, tinyint/smallint/int days)	Subtract a number of days to startdate: date_sub('2008-12-31' , 1) = '2008-12-30'

timest amp	from_utc_timestamp({any primitive type}*, string timezone)	Assumes given timestamp is UTC and converts to given timezone (as of Hive 0.8.0)
timest amp	to_utc_timestamp({any primitive type}*, string timezone)	Assumes given timestamp is in given timezone and converts to UTC (as of Hive 0.8.0)
date	current_date	Returns the current date at the start of query evaluation (as of Hive 1.2.0). All calls of current_date within the same query return the same value.
timest amp	current_timestamp	Returns the current timestamp at the start of query evaluation (as of Hive 1.2.0). All calls of current_timestamp within the same query return the same value.
string	add_months(string start_date, int num_months)	Returns the date that is num_months after start_date (as of Hive 1.1.0). start_date is a string, date or timestamp. num_months is an integer. The time part of start_date is ignored. If start_date is the last day of the month or if the resulting month has fewer days than the day component of start_date, then the result is the last day of the resulting month. Otherwise, the result has the same day component as start_date.
string	last_day(string date)	Returns the last day of the month which the date belongs to (as of Hive 1.1.0). date is a string in the format 'yyyy-MM-dd HH:mm:ss' or 'yyyy-MM-dd'. The time part of date is ignore
string	next_day(string start_date, string day_of_week)	Returns the first date which is later than start_date and named as day_of_week (as of Hive 1.2.0). start_date is a string/date/timestamp. day_of_week is 2 letters, 3 letters or full name of the day of the week (e.g. Mo, tue,

		FRIDAY). The time part of start_date is ignored. Example: next_day('2015-01-14' , 'TU') = 2015-01-20.
string	trunc(string date, string format)	Returns date truncated to the unit specified by the format (as of Hive <u>1.2.0</u>). Supported formats: MONTH/MON/MM, YEAR/YYYY/YY. Example: trunc('2015-03-17' , 'MM') = 2015-03-01.
double	months_between(date1, date2)	Returns number of months between dates date1 and date2 (as of Hive <u>1.2.0</u>). If date1 is later than date2, then the result is positive. If date1 is earlier than date2, then the result is negative. If date1 and date2 are either the same days of the month or both last days of months, then the result is always an integer. Otherwise the UDF calculates the fractional portion of the result based on a 31-day month and considers the difference in time components date1 and date2. date1 and date2 type can be date, timestamp or string in the format 'yyyy-MM-dd' or 'yyyy-MM-dd HH:mm:ss'. The result is rounded to 8 decimal places. Example: months_between('1997-02-28 10:30:00' , '1996-10-30') = 3.94959677
string	date_format(date/timestamp ts, string fmt)	Converts a date/timestamp/string to a value of string in the format specified by the date format fmt (as of Hive <u>1.2.0</u>). Supported formats are Java SimpleDateFormat formats – https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html . The second argument fmt should be constant. Example: date_format('2015-04-08' , 'y') = '2015'. date_format can be used to implement other UDFs, e.g.: dayname(date) is date_format(date, 'EEEE'); dayofyear(date) is date_format(date, 'D')

Mathematical Functions

The following built-in mathematical functions are supported in Hive; most return NULL when the argument(s) are NULL:

Return Type	Name(Signature)	Example
BIGINT	round(double a)	Returns the rounded BIGINT value of the double
DOUBLE	round(double a, int d)	Returns the double rounded to d decimal places
BIGINT	floor(double a)	Returns the maximum BIGINT value that is equal or less than the double
BIGINT	ceil(double a), ceiling(double a)	Returns the minimum BIGINT value that is equal or greater than the double
double	rand(), rand(int seed)	Returns a random number (that changes from row to row) that is distributed uniformly from 0 to 1. Specifying the seed will make sure the generated random number sequence is deterministic.
double	exp(double a), exp(DECIMAL a)	Returns e^a where e is the base of the natural logarithm
double	ln(double a), ln(DECIMAL a)	Returns the natural logarithm of the argument
double	log10(double a), log10(DECIMAL a)	Returns the base-10 logarithm of the argument
double	log2(double a), log2(DECIMAL a)	Returns the base-2 logarithm of the argument

double	<code>log(double base, double a), log(DECIMAL base, DECIMAL a)</code>	Return the base “base” logarithm of the argument
double	<code>pow(double a, double p), power(double a, double p)</code>	Return a^p
double	<code>sqrt(double a), sqrt(DECIMAL a)</code>	Returns the square root of a
string	<code>bin(BIGINT a)</code>	Returns the number in binary format
string	<code>hex(BIGINT a) hex(string a) hex(BINARY a)</code>	If the argument is an int, hex returns the number as a string in hex format. Otherwise if the number is a string, it converts each character into its hex representation and returns the resulting string.
string	<code>unhex(string a)</code>	Inverse of hex. Interprets each pair of characters as a hexadecimal number and converts to the character represented by the number.
string	<code>conv(BIGINT num, int from_base, int to_base), conv(STRING num, int from_base, int to_base)</code>	Converts a number from a given base to another
double	<code>abs(double a)</code>	Returns the absolute value
int double	<code>pmod(int a, int b) pmod(double a, double b)</code>	Returns the positive value of a mod b
double	<code>sin(double a), sin(DECIMAL a)</code>	Returns the sine of a (a is in radians)
double	<code>asin(double a), asin(DECIMAL a)</code>	Returns the arc sin of x if $-1 \leq a \leq 1$ or null otherwise

double	<code>cos(double a), cos(DECIMAL a)</code>	Returns the cosine of a (a is in radians)
double	<code>acos(double a), acos(DECIMAL a)</code>	Returns the arc cosine of x if $-1 \leq a \leq 1$ or null otherwise
<code>tan(double a)</code>	<code>tan(double a), tan(DECIMAL a)</code>	Returns the tangent of a (a is in radians)
double	<code>atan(double a), atan(DECIMAL a)</code>	Returns the arctangent of a
double	<code>degrees(double a), degrees(DECIMAL a)</code>	Converts value of a from radians to degrees
double	<code>radians(double a)</code>	Converts value of a from degrees to radians
int double	<code>positive(int a), positive(double a)</code>	Returns a
int double	<code>negative(int a), negative(double a)</code>	Returns -a
float	<code>sign(double a), sign(DECIMAL a)</code>	Returns the sign of a as '1.0' or '-1.0'
double	<code>e()</code>	Returns the value of e
double	<code>pi()</code>	Returns the value of pi

String Functions

The following are built-in String functions are supported in hive:

Return Type	Name(Signature)	Example
int	ascii(string str)	Returns the numeric value of the first character of str
string	base64(binary bin)	Converts the argument from binary to a base- 64 string (as of 0.12.0)
string	chr(bigint double A)	Returns the ASCII character having the binary equivalent to A (as of Hive 1.3.0 and 2.1.0). If A is larger than 256 the result is equivalent to chr(A % 256). Example: select chr(88); returns "X" .string
string	concat(string binary A, string binary B...)	Returns the string or bytes resulting from concatenating the strings or bytes passed in as parameters in order. e.g. concat('foo' , 'bar') results in 'foobar' . Note that this function can take any number of input strings.
array<struct<string, double>>	context_ngrams(array<array>, array, int K, int pf)	Returns the top-k contextual N-grams from a set of tokenized sentences, given a string of "context" . See StatisticsAndDataMining for more information.
string	concat_ws(string SEP, string A, string B...)	Like concat() above, but with custom separator SEP.
string	concat_ws(string SEP, array<string>)	Like concat_ws() above, but taking an array of strings. (as of Hive 0.9.0)

string	decode(binary bin, string charset)	Decodes the first argument into a String using the provided character set (one of ‘US-ASCII’ , ‘ISO-8859-1’ , ‘UTF-8’ , ‘UTF-16BE’ , ‘UTF-16LE’ , ‘UTF-16’). If either argument is null, the result will also be null. (As of Hive 0.12.0.)
string	elt(N int,str1 string,str2 string,str3 string,⋯)	Return string at index number. For example elt(2,’ hello’ , ‘ world’) returns ‘world’ . Returns NULL if N is less than 1 or greater than the number of arguments. (See https://dev.mysql.com/doc/refman/5.7/en/string-functions.html#function_elt)
binary	encode(string src, string charset)	Encodes the first argument into a BINARY using the provided character set (one of ‘US-ASCII’ , ‘ISO-8859-1’ , ‘UTF-8’ , ‘UTF-16BE’ , ‘UTF-16LE’ , ‘UTF-16’). If either argument is null, the result will also be null. (As of Hive 0.12.0.)
int	field(val T,val1 T,val2 T,val3 T,⋯)	Returns the index of val in the val1,val2,val3,⋯ list or 0 if not found. For example field(‘world’ , ‘ say’ , ‘ hello’ , ‘ world’) returns 3. All primitive types are supported, arguments are compared using str.equals(x). If val is NULL, the return value is 0. (See https://dev.mysql.com/doc/refman/5.7/en/string-functions.html#function_field)
int	find_in_set(string str, string strList)	Returns the first occurrence of str in strList where strList is a comma-delimited string. Returns null if either argument is null. Returns 0 if the first

		argument contains any commas. e.g. find_in_set('ab' , 'abc,b,ab,c,def') returns 3
string	format_number(number x, int d)	Formats the number X to a format like '#,###,##.##' , rounded to D decimal places, and returns the result as a string. If D is 0, the result has no decimal point or fractional part. (as of Hive 0.10.0)
string	get_json_object(string json_string, string path)	Extract json object from a json string based on json path specified, and return json string of the extracted json object. It will return null if the input json string is invalid. NOTE: The json path can only have the characters [0-9a-z_], i.e., no upper-case or special characters. Also, the keys *cannot start with numbers.* This is due to restrictions on Hive column names.
boolean	in_file(string str, string filename)	Returns true if the string str appears as an entire line in filename.
string	initcap(string A)	Returns string, with the first letter of each word in uppercase, all other letters in lowercase. Words are delimited by whitespace. (As of Hive 1.1.0.)
int	instr(string str, string substr)	Returns the position of the first occurrence of substr in str
int	length(string A)	Returns the length of the string
int	levenshtein(string A, string B)	Returns the Levenshtein distance between two strings (as of Hive 1.2.0). For example, levenshtein('kitten' , 'sitting') results in 3.

int	locate(string substr, string str[, int pos])	Returns the position of the first occurrence of substr in str after position pos
string	lower(string A) lcase(string A)	Returns the string resulting from converting all characters of A to lower case. For example, lower('fOoBaR') results in 'foobar' .
string	lpad(string str, int len, string pad)	Returns str, left-padded with pad to a length of len. If str is longer than len, the return value is shortened to len characters. In case of empty pad string, the return value is null.
string	ltrim(string A)	Returns the string resulting from trimming spaces from the beginning(left hand side) of A e.g. ltrim(' foobar ') results in 'foobar '
array<struct<string, double>>	ngrams(array<array >, int N, int K, int pf)	Returns the top-k N-grams from a set of tokenized sentences, such as those returned by the sentences() UDAF. See StatisticsAndDataMining for more information.
string	parse_url(string urlString, string partToExtract [, string keyToExtract])	Returns the specified part from the URL. Valid values for partToExtract include HOST, PATH, QUERY, REF, PROTOCOL, AUTHORITY, FILE, and USERINFO. e.g. parse_url('http://facebook.com/path1/p.php?k1=v1&k2=v2#Ref1' , 'HOST') returns 'facebook.com' . Also a value of a particular key in QUERY can be extracted by providing the key as the third argument, e.g. parse_url('http://facebook.com/path1/p.php?k1=v1&k2=v2#Ref1' , 'QUERY' , 'k1') returns 'v1' .

string	<code>printf(String format, Obj... args)</code>	Returns the input formatted according do printf-style format strings (as of Hive 0.9.0)
string	<code>regexp_extract(string subject, string pattern, int index)</code>	Returns the string extracted using the pattern. e.g. <code>regexp_extract('foothebar' , 'foo.(.*?)(bar)' , 2)</code> returns 'bar.' Note that some care is necessary in using predefined character classes: using '\s' as the second argument will match the letter s; 's' is necessary to match whitespace, etc. The 'index' parameter is the Java regex Matcher group() method index. See docs/api/java/util/regex/Matcher.html for more information on the 'index' or Java regex group() method.
string	<code>regexp_replace(string INITIAL_STRING, string PATTERN, string REPLACEMENT)</code>	Returns the string resulting from replacing all substrings in INITIAL_STRING that match the java regular expression syntax defined in PATTERN with instances of REPLACEMENT, e.g. <code>regexp_replace("foobar" , "oo ar" , "")</code> returns 'fb.' Note that some care is necessary in using predefined character classes: using '\s' as the second argument will match the letter s; 's' is necessary to match whitespace, etc.
string	<code>repeat(string str, int n)</code>	Repeat str n times
string	<code>replace(string A, string OLD, string NEW)</code>	Returns the string A with all non-overlapping occurrences of OLD replaced with NEW (as of Hive 1.3.0 and 2.1.0). Example: <code>select replace("ababab" , "abab" , "Z");</code> returns "Zab" .

string	reverse(string A)	Returns the reversed string
string	rpad(string str, int len, string pad)	Returns str, right-padded with pad to a length of len. If str is longer than len, the return value is shortened to len characters. In case of empty pad string, the return value is null.
string	rtrim(string A)	Returns the string resulting from trimming spaces from the end(right hand side) of A e.g. rtrim('foobar ') results in 'foobar'
array<array>	sentences(string str, string lang, string locale)	Tokenizes a string of natural language text into words and sentences, where each sentence is broken at the appropriate sentence boundary and returned as an array of words. The 'lang' and 'locale' are optional arguments. e.g. sentences('Hello there! How are you?') returns (("Hello" , "there"), ("How" , "are" , "you"))
string	soundex(string A)	Returns soundex code of the string (as of Hive 1.2.0). For example, soundex('Miller') results in M460.
string	space(int n)	Return a string of n spaces
array	split(string str, string pat)	Split str around pat (pat is a regular expression)
map<string,string>	str_to_map(text[, delimiter1, delimiter2])	Splits text into key-value pairs using two delimiters. Delimiter1 separates text into K-V pairs, and Delimiter2 splits each K-V pair. Default delimiters are ';' for delimiter1 and '=' for delimiter2.

string	<code>substr(string binary A, int start)</code> <code>substring(string binary A, int start)</code>	Returns the substring or slice of the byte array of A starting from start position till the end of string A e.g. <code>substr('foobar' , 4)</code> results in 'bar'
string	<code>substr(string binary A, int start, int len)</code> <code>substring(string binary A, int start, int len)</code>	Returns the substring or slice of the byte array of A starting from start position with length len e.g. <code>substr('foobar' , 4, 1)</code> results in 'b'
string	<code>translate(string char varchar input, string char varchar from, string char varchar to)</code>	Translates the input string by replacing the characters present in the from string with the corresponding characters in the to string. This is similar to the translate function in PostgreSQL. If any of the parameters to this UDF are NULL, the result is NULL as well (available as of Hive 0.10.0; char/varchar support added as of Hive 0.14.0.)
string	<code>trim(string A)</code>	Returns the string resulting from trimming spaces from both ends of A e.g. <code>trim('foobar')</code> results in 'foobar'
string	<code>unbase64(string str)</code>	Converts the argument from a base 64 string to BINARY. (As of Hive 0.12.0.)
string	<code>upper(string A)</code> <code>ucase(string A)</code>	Returns the string resulting from converting all characters of A to upper case e.g. <code>upper('fOoBaR')</code> results in 'FOOBAR'

Collection Functions

The following built-in collection functions are supported in hive:

Return Type	Name(Signature)	Example
int	size(Map)	Returns the number of elements in the map type
int	size(Array)	Returns the number of elements in the array type
array	map_keys(Map)	Returns an unordered array containing the keys of the input map
array	map_values(Map)	Returns an unordered array containing the values of the input map
boolean	array_contains(Array, value)	Returns TRUE if the array contains value
array	sort_array(Array)	Sorts the input array in ascending order according to the natural ordering of the array elements and returns it (as of version 0.9.0)

Built-in Aggregate Functions (UDAF)

The following are built-in aggregate functions supported in Hive:

Return Type	Name(Signature)	Example
bigint	count(*), count(expr), count(DISTINCT expr[, expr_...])	count(*) – Returns the total number of retrieved rows, including rows containing NULL values; count(expr) – Returns the number of rows for which the supplied expression is non-NULL; count(DISTINCT expr[, expr_...]) – Returns the number of rows for which the supplied expression(s) are unique and non-NULL.
double	sum(col), sum(DISTINCT col)	Returns the sum of the elements in the group or the sum of the distinct values of the column in the group
double	avg(col), avg(DISTINCT col)	Returns the average of the elements in the group or the average of the distinct values of the column in the group
double	min(col)	Returns the minimum of the column in the group
double	max(col)	Returns the maximum value of the column in the group
double	variance(col), var_pop(col)	Returns the variance of a numeric column in the group
double	var_samp(col)	Returns the unbiased sample variance of a numeric column in the group

double	<code>stddev_pop(col)</code>	Returns the standard deviation of a numeric column in the group
double	<code>stddev_samp(col)</code>	Returns the unbiased sample standard deviation of a numeric column in the group
double	<code>covar_pop(col1, col2)</code>	Returns the population covariance of a pair of numeric columns in the group
double	<code>covar_samp(col1, col2)</code>	Returns the sample covariance of a pair of a numeric columns in the group
double	<code>corr(col1, col2)</code>	Returns the Pearson coefficient of correlation of a pair of a numeric columns in the group
double	<code>percentile(BIGINT col, p)</code>	Returns the exact p^{th} percentile of a column in the group (does not work with floating point types). p must be between 0 and 1. NOTE: A true percentile can only be computed for integer values. Use <code>PERCENTILE_APPROX</code> if your input is non-integral.
array<double>	<code>percentile(BIGINT col, array(p1 [, p2]…))</code>	Returns the exact percentiles p_1, p_2, \dots of a column in the group (does not work with floating point types). p_i must be between 0 and 1. NOTE: A true percentile can only be computed for integer values. Use <code>PERCENTILE_APPROX</code> if your input is non-integral.
double	<code>percentile_approx(DOUBLE col, p [, B])</code>	Returns an approximate p^{th} percentile of a numeric column (including floating point types) in the group. The B parameter controls approximation accuracy at the cost of memory. Higher values yield better approximations, and the default is 10,000.

		When the number of distinct values in col is smaller than B, this gives an exact percentile value.
array	percentile_approx(DOUBLE col, array(p1 [, p2]… [, B]))	Same as above, but accepts and returns an array of percentile values instead of a single one.
array<struct { 'x' , 'y' }>	histogram_numeric(col, b)	Computes a histogram of a numeric column in the group using b non-uniformly spaced bins. The output is an array of size b of double-valued (x,y) coordinates that represent the bin centers and heights
array	collect_set(col)	Returns a set of objects with duplicate elements eliminated
array	collect_list(col)	Returns a list of objects with duplicates. (As of Hive 0.13.0.)
integer	ntile(INTEGER x)	Divides an ordered partition into x groups called buckets and assigns a bucket number to each row in the partition. This allows easy calculation of tertiles, quartiles, deciles, percentiles and other common summary statistics. (As of Hive 0.11.0.)

Built-in Table-Generating Functions (UDTF)

Normal user-defined functions, such as concat(), take in a single input row and output a single output row. In contrast, table-generating functions transform a single input row to multiple output rows.

Return Type	Name(Signature)	Example
inline(ARRAY<STRUCT[,STRUCT]>)	Explodes an array of structs into a table (as of Hive 0.10)	
explode(ARRAY<T> a)	explode() takes in an array as an input and outputs the elements of the array as separate rows. UDTF's can be used in the SELECT expression list and as a part of LATERAL VIEW.	

Conditional Functions

Return Type	Name(Signature)	Example
T	if(boolean testCondition, T valueTrue, T valueFalseOrNull)	Return valueTrue when testCondition is true, returns valueFalseOrNull otherwise
T	COALESCE(T v1, T v2, ...)	Return the first v that is not NULL, or NULL if all v's are NULL
T	CASE a WHEN b THEN c [WHEN d THEN e]* [ELSE f] END	When a = b, returns c; when a = d, return e; else return f
T	CASE WHEN a THEN b [WHEN c THEN d]* [ELSE e] END	When a = true, returns b; when c = true, return d; else return e

Functions for Text Analytics

Return Type	Name(Signature)	Example
array<struct<string,double>>	context_ngrams(array<array>, array, int K, int pf)	<p>Returns the top-k contextual N-grams from a set of tokenized sentences, given a string of "context".</p> <p>See StatisticsAndDataMining for more information.N-grams are subsequences of length N drawn from a longer sequence.</p> <p>The purpose of the ngrams() UDAF is to find the k most frequent n-grams from one or more sequences. It can be used in conjunction with the sentences() UDF to analyze unstructured natural language text, or the collect() function to analyze more general string data.</p>
array<struct<string,double>>	ngrams(array<array>, int N, int K, int pf)	<p>Returns the top-k N-grams from a set of tokenized sentences, such as those returned by the sentences() UDAF.</p> <p>See StatisticsAndDataMining for more information.Contextual n-grams are similar to n-grams, but allow you to specify a 'context' string around which n-grams are to be estimated. For example, you can specify that you're only interested in finding the most</p>

		<p>common two-word phrases in text that follow the context “I love” . You could achieve the same result by manually stripping sentences of non-contextual content and then passing them to ngrams(), but context_ngrams() makes it much easier.</p>
--	--	---

Apache Spark Cheat Sheet

Transformations (return new RDDs – Lazy)

Where	Function	DStream API	Description
RDD	map(function)	Yes	Return a new distributed dataset formed by passing each element of the source through a function.
RDD	filter(function)	Yes	Return a new dataset formed by selecting those elements of the source on which function returns true.
OrderedRDD Functions	filterByRange(lower, upper)	No	Returns an RDD containing only the elements in the the inclusive range lower to upper.
RDD	flatMap(function)	Yes	Similar to map, but each input item can be mapped to 0 or more output items (so function should return a Seq rather than a single item).

RDD	<code>mapPartitions(function)</code>	Yes	Similar to map, but runs separately on each partition of the RDD.
RDD	<code>mapPartitionsWithIndex(function)</code>	No	Similar to mapPartitions, but also provides function with an integer value representing the index of the partition.
RDD	<code>sample(withReplacement, fraction, seed)</code>	No	Sample a fraction of the data, with or without replacement, using a given random number generator seed.
RDD	<code>union(otherDataset)</code>	Yes	Return a new dataset that contains the union of the elements in the datasets.
RDD	<code>intersection(otherDataset)</code>	No	Return a new RDD that contains the intersection of elements in the datasets.
RDD	<code>distinct([numTasks])</code>	No	Return a new dataset that

			contains the distinct elements of the source dataset.
PairRDD Functions	<code>groupByKey([numTasks])</code>	Yes	Returns a dataset of (K, Iterable<V>) pairs. Use <code>reduceByKey</code> or <code>aggregateByKey</code> to perform an aggregation (such as a sum or average).
PairRDD Functions	<code>reduceByKey(function, [numTasks])</code>	Yes	Returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function.
PairRDD Functions	<code>aggregateByKey(zeroValue)(seqOp, combOp, [numTasks])</code>	No	Returns a dataset of (K, U) pairs where the values for each key are aggregated using the given combine functions and a neutral "zero" value. Allows an aggregated value type that is different than the input value type.
OrderedRDD Functions	<code>sortByKey([ascending], [numTasks])</code>	No	Returns a dataset of (K, V) pairs sorted

			by keys in ascending or descending order, as specified in the boolean ascending argument.
PairRDD Functions	join(otherDataset, [numTasks])	Yes	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.
PairRDD Functions	cogroup(otherDataset, [numTasks])	Yes	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (Iterable<V>, Iterable<W>)) tuples.
RDD	cartesian(otherDataset)	No	When called on datasets of types T and U, returns a dataset of (T, U) pairs (all pairs of elements).

RDD	<code>pipe(command, [envVars])</code>	No	Pipe each partition of the RDD through a shell command, e.g. a Perl or bash script.
RDD	<code>coalesce(numPartitions)</code>	No	Decrease the number of partitions in the RDD to numPartitions. Useful for running operations more efficiently after filtering down a large dataset.
RDD	<code>repartition(numPartitions)</code>	Yes	Reshuffle the data in the RDD randomly to create either more or fewer partitions and balance it across them. This always shuffles all data over the network.
OrderedRDD Functions	<code>repartitionAndSortWithinPartitions(partitioner)</code>	No	Repartition the RDD according to the given partitioner and, within each resulting partition, sort records by their keys. More efficient than calling

		<p>repartition and then sorting.</p>
--	--	---

Actions (return values – NOT Lazy)

Where	Function	DStream API	Description
RDD	reduce(function)	Yes	Aggregate the elements of the dataset using a function (which takes two arguments and returns one).
RDD	collect()	No	Return all the elements of the dataset as an array at the driver program. Best used on sufficiently small subsets of data.
RDD	count()	Yes	Return the number of elements in the dataset.
RDD	countByValue()	Yes	Return the count of each unique value in this RDD as a local map of (value, count) pairs.
RDD	first()	No	Return the first element of the dataset (similar to take(1)).
RDD	take(n)	No	Return an array with the first n elements of the dataset.
RDD	takeSample(withReplacement, num, [seed])	No	Return an array with a random sample of num elements of the dataset.
RDD	takeOrdered(n, [ordering])	No	Return the first n elements of the RDD using either their

			natural order or a custom comparator.
RDD	saveAsTextFile(path)	Yes	Write the elements of the dataset as a text. Spark will call <code>toString</code> on each element to convert it to a line of text in the file.
SequenceFileRDD Functions	saveAsSequenceFile(path) (Java and Scala)	No	Write the elements of the dataset as a Hadoop SequenceFile in a given path. For RDDs of key-value pairs that use Hadoop's Writable interface.
RDD	saveAsObjectFile(path) (Java and Scala)	Yes	Write the elements of the dataset in a simple format using Java serialization, which can then be loaded using <code>SparkContext.objectFile()</code> .
PairRDD Functions	countByKey()	No	Only available on RDDs of type <code>(K, V)</code> . Returns a hashmap of <code>(K, Int)</code> pairs with the count of each key.
RDD	foreach(function)	Yes	Run a function on each element of the dataset. This is usually done for side effects such as updating an Accumulator.

Persistence Methods

Where	Function	DStream API	Description
RDD	cache()	Yes	Don't be afraid to call cache on RDDs to avoid unnecessary recomputation. NOTE: This is the same as persist(MEMORY_ONLY).
RDD	persist([Storage Level])	Yes	Persist this RDD with the default storage level.
RDD	unpersist()	No	Mark the RDD as non-persistent, and remove its blocks from memory and disk.
RDD	checkpoint()	Yes	Save to a file inside the checkpoint directory and all references to its parent RDDs will be removed.

Additional Transformation and Actions

Where	Function	Description
SparkContext	doubleRDDToDoubleRDDFunctions	Extra functions available on RDDs of Doubles
SparkContext	numericRDDToDoubleRDDFunctions	Extra functions available on RDDs of Doubles
SparkContext	rddToPairRDDFunctions	Extra functions available on RDDs of (key, value) pairs
SparkContext	hadoopFile()	Get an RDD for a Hadoop file with an arbitrary InputFormat
SparkContext	hadoopRDD()	Get an RDD for a Hadoop file with an arbitrary InputFormat
SparkContext	makeRDD()	Distribute a local Scala collection to form an RDD
SparkContext	parallelize()	Distribute a local Scala collection to form an RDD
SparkContext	textFile()	Read a text file from a file system URI
SparkContext	wholeTextFiles()	Read a directory of text files from a file system URI

Extended RDDs w/ Custom Transformations and Actions

RDD Name	Description
CoGroupedRDD	A RDD that cogsroups its parents. For each key k in parent RDDs, the resulting RDD contains a tuple with the list of values for that key.
EdgeRDD	Storing the edges in columnar format on each partition for performance. It may additionally store the vertex attributes associated with each edge.
JdbcRDD	An RDD that executes an SQL query on a JDBC connection and reads results. For usage example, see test case JdbcRDDSuite.
ShuffledRDD	The resulting RDD from a shuffle.
VertexRDD	Ensures that there is only one entry for each vertex and by pre-indexing the entries for fast, efficient joins.

Streaming Transformations

Where	Function	Description
DStream	<code>window(windowLength, slideInterval)</code>	Return a new DStream which is computed based on windowed batches of the source DStream.
DStream	<code>countByWindow(windowLength, slideInterval)</code>	Return a sliding window count of elements in the stream.
DStream	<code>reduceByWindow(function, windowLength, slideInterval)</code>	Return a new single-element stream, created by aggregating elements in the stream over a sliding interval using function.
PairDStream Functions	<code>reduceByKeyAndWindow(function, windowLength, slideInterval, [numTasks])</code>	Returns a new DStream of (K, V) pairs where the values for each key are aggregated using the given reduce function over batches in a sliding window.
PairDStream Functions	<code>reduceByKeyAndWindow(function, invFunc, windowLength, slideInterval, [numTasks])</code>	A more efficient version of the above <code>reduceByKeyAndWindow()</code> . Only applicable to those reduce functions which have a corresponding "inverse reduce" function. Checkpointing must be enabled for using this operation.
DStream	<code>countByValueAndWindow(windowLength, slideInterval, [numTasks])</code>	Returns a new DStream of (K, Long) pairs where the value of each key is its frequency within a sliding window.
DStream	<code>transform(function)</code>	The transform operation (along with its variations like

		transformWith) allows arbitrary RDD-to-RDD functions to be applied on a Dstream.
PairDStream Functions	updateStateByKey(function)	The updateStateByKey operation allows you to maintain arbitrary state while continuously updating it with new information.

RDD Persistence

Storage Level	Meaning
MEMORY_ONLY (default level)	Store RDD as deserialized Java objects. If the RDD does not fit in memory, some partitions will not be cached and will be recomputed on the fly when needed.
MEMORY_AND_DISK	Store RDD as deserialized Java objects. If the RDD does not fit in memory, store the partitions that don't fit on disk, and load them when they're needed.
MEMORY_ONLY_SER	Store RDD as serialized Java objects. Generally more space-efficient than deserialized objects, but more CPU-intensive to read.
MEMORY_AND_DISK_SER	Similar to MEMORY_ONLY_SER, but spill partitions that don't fit in memory to disk instead of recomputing them on the fly each time they're needed.
DISK_ONLY	Store the RDD partitions only on disk.
MEMORY_ONLY_2, MEMORY_AND_DISK_2, etc...	Same as the levels above, but replicate each partition on two cluster nodes.

Shared Data

Broadcast Variables Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks.

Language	Create, Evaluate
Scala	<code>val broadcastVar = sc.broadcast(Array(1, 2, 3))</code>
	<code>broadcastVar.value</code>
Java	<code>Broadcast<int[]> broadcastVar = sc.broadcast(new int[] {1, 2, 3});</code>
	<code>broadcastVar.value();</code>
Python	<code>broadcastVar = sc.broadcast([1, 2, 3])</code>
	<code>broadcastVar.value</code>

Accumulators are variables that are only "added" to through an associative operation and can therefore be efficiently supported in parallel.

Language	Create, Add, Evaluate
Scala	<code>val accum = sc.accumulator(0, My Accumulator)</code>
	<code>sc.parallelize(Array(1, 2, 3, 4)).foreach(x => accum += x)</code>
	<code>accum.value</code>
Java	<code>Accumulator<Integer> accum = sc.accumulator(0);</code>
	<code>sc.parallelize(Arrays.asList(1, 2, 3, 4)).foreach(x -> accum.add(x))</code>
	<code>accum.value();</code>
Python	<code>accum = sc.accumulator(0)</code>

MLlib Reference

Topic	Description
Data types	Vectors, points, matrices.
Basic Statistics	Summary, correlations, sampling, testing and random data.
Classification and regression	Includes SVMs, decision trees, naïve Bayes, etc...
Collaborative filtering	Commonly used for recommender systems.
Clustering	Clustering is an unsupervised learning approach.
Dimensionality reduction	Dimensionality reduction is the process of reducing the number of variables under consideration.
Feature extraction and transformation	Used in selecting a subset of relevant features (variables, predictors) for use in model construction.
Frequent pattern mining	Mining is usually among the first steps to analyze a large-scale dataset.
Optimization	Different optimization methods can have different convergence guarantees.
PMML model export	MLlib supports model export to Predictive Model Markup Language.

Hadoop Shell Cheat Sheet

1. hdfs rename / move file or directory
2. hdfs dfs -mv /old/file /new/file
- 3.
4. hdfs **get** admin report / cluster status
5. hdfs dfsadmin -report
- 6.
7. hdfs **get** rack awareness and number of under replicated blocks
8. hadoop fsck / -locations -blocks -files | grep -i -C6 miss
- 9.
10. hdfs print a compressed file uncompress a file
11. hdfs dfs -text /path/to/compressed_file
- 12.
13. hdfs merge all files **in** a folder to one file
14. hdfs dfs -getmerge /path/to/folder /path/to/file
- 15.
16. hdfs rm remove file
17. hdfs dfs -rm /user/hive/warehouse/blah/blah
- 18.
19. hdfs kill yarn job
20. hadoop job -kill job_id
- 21.
22. hdfs empty trash
23. hdfs dfs -expunge
- 24.
25. hdfs stream result of pipe stdin to a hadoop file
26. cat somefile.tsv | hdfs dfs -put - /file/on/cluster.tsv
- 27.
28. hdfs put copy file from local to cluster
29. hdfs dfs -put localfile /user/hadoop/hadoopfile
- 30.
31. hdfs create touch a file
32. hdfs dfs -touchz /path/to/myfile.txt
- 33.
34. hdfs tail a file
35. hdfs dfs -tail /my_file
- 36.
37. hdfs **return** stat information of the path (basic file info)
38. hdfs dfs -stat /my_file
- 39.

40. hdfs delete folder
41. hdfs dfs -rmr /folder/to/delete
- 42.
43. hdfs delete file
44. hdfs dfs -rm /file/to/delete
- 45.
46. hdfs list all files **in** a directory
47. hdfs dfs -ls /user/dir1
- 48.
49. hdfs **get** access control list ACL of files and directories
50. hdfs dfs -getfacl /file
- 51.
52. hdfs display size of files and directories
53. hdfs dfs -du /user/hadoop/dir1
- 54.
55. hdfs copy file within the cluster
56. hdfs dfs -cp /source_file /dest_file
- 57.
58. hdfs count number of directories, files and bytes
59. hdfs dfs -count /hdfs/folder/file
- 60.
61. hdfs copy from local machine to cluster with overwrite
62. hdfs dfs -copyFromLocal -f /local/file.csv /hdfs/folder/
- 63.
64. hdfs copy from remote cluster to local machine
65. hdfs dfs -copyToLocal /hdfs/folder/file.csv /local/folder/
- 66.
67. hdfs copy from local machine to cluster
68. hdfs dfs -copyFromLocal /local/file.csv /hdfs/folder/
- 69.
70. hdfs stream append to a file **using** unix pipe stdin
71. cat somefile | hdfs dfs -appendToFile - /hdfs/my_file.txt
- 72.
73. hdfs append to a file
74. hdfs dfs -appendToFile /local_file.txt /hdfs/my_file.txt
- 75.
76. hdfs print a file
77. hdfs dfs -cat /my_file
- 78.
79. hdfs chgrp change group of file
80. hdfs dfs -chgrp group_name /my_file

- 81.
82. `hdfs chmod change file permissions`
83. `hdfs dfs -chmod 1755 /my_file`
- 84.
85. `hdfs chown change owner of file/directory`
86. `hdfs dfs -chown user:group /my_hdfs/file`
- 87.
88. `hdfs set replication factor recursive`
89. `hdfs dfs -setrep -R -w 1 /my_hdfs/folder`
- 90.
91. `hdfs change replication factor of a file`
92. `hdfs dfs -setrep -w 2 /my_hdfs/file`
- 93.
94. `hdfs Copy Files from one cluster to other`
95. `hadoop distcp -pb -`
`overwrite hftp://hdfssource:50070/file/to/copy hdfs://hdfsdest:8020/user/test`
- 96.
97. `print sequence file`
98. `hdfs dfs -text /path/to/file/hdfs`
- 99.
100. `create directory`
101. `hdfs dfs -mkdir /path/to/directory`
- 102.
103. `hadoop streaming`
104. `hadoop jar $HADOOP_HOME/hadoop-streaming.jar \`
`-input myInputDirs \`
105. `-output myOutputDir \`
106. `-mapper myPythonScript.py \`
107. `-reducer /bin/wc \`
108. `-file myPythonScript.py`
- 109.
- 110.
111. `set sticky bit (preventing anyone except the superuser, owner from deleting or moving the files within the directory)`
112. `sudo -u hdfs hadoop fs -chmod 1777 /tmp`

Hadoop Admin Cheat Sheet

Hadoop Namenode Commands

Command	Description
hadoop namenode -format	Format HDFS filesystem from Namenode
hadoop namenode -upgrade	Upgrade the NameNode
start-dfs.sh	Start HDFS Daemons
stop-dfs.sh	Stop HDFS Daemons
start-mapred.sh	Start MapReduce Daemons
stop-mapred.sh	Stop MapReduce Daemons
hadoop namenode -recover -force	Recover namenode metadata after a cluster failure (may lose data)

Hadoop fsck Commands

Command	Description
hadoop fsck /	Filesystem check on HDFS
hadoop fsck / -files	Display files during check
hadoop fsck / -files -blocks	Display files and blocks during check
hadoop fsck / -files -blocks -locations	Display files, blocks and its location during check
hadoop fsck / -files -blocks -locations -racks	Display network topology for data-node locations
hadoop fsck -delete	Delete corrupted files
hadoop fsck -move	Move corrupted files to /lost+found directory

Hadoop Job Commands

Command	Description
hadoop job -submit <job-file>	Submit the job
hadoop job -status <job-id>	Print job status completion percentage
hadoop job -list all	List all jobs
hadoop job -list-active-trackers	List all available TaskTrackers
hadoop job -set-priority <job-id> <priority>	Set priority for a job. Valid priorities: VERY_HIGH, HIGH, NORMAL, LOW, VERY_LOW
hadoop job -kill-task <task-id>	Kill a task
hadoop job -history	Display job history including job details, failed and killed jobs

Hadoop dfs admin Commands

Command	Description
hadoop dfsadmin -report	Report filesystem info and statistics
hadoop dfsadmin -metasave file.txt	Save namenode's primary data structures to file.txt
hadoop dfsadmin -setQuota 10 /quotatest	Set Hadoop directory quota to only 10 files
hadoop dfsadmin -clrQuota /quotatest	Clear Hadoop directory quota
hadoop dfsadmin -refreshNodes	Read hosts and exclude files to update datanodes that are allowed to connect to namenode. Mostly used to commission or decommission nodes
hadoop fs -count -q /mydir	Check quota space on directory /mydir
hadoop dfsadmin -setSpaceQuota /mydir 100M	Set quota to 100M on hdfs directory named /mydir
hadoop dfsadmin -clrSpaceQuota /mydir	Clear quota on a HDFS directory
hadoop dfsadmin -saveNameSpace	Backup Metadata (fsimage & edits). Put cluster in safe mode before this command.

Hadoop Safe Mode (Maintenance Mode) Commands

The following dfs admin commands helps the cluster to enter or leave safe mode, which is also called as maintenance mode. In this mode, Namenode does not accept any changes to the name space, it does not replicate or delete blocks.

Command	Description
hadoop dfsadmin -safemode enter	Enter safe mode
hadoop dfsadmin -safemode leave	Leave safe mode
hadoop dfsadmin -safemode get	Get the status of mode
hadoop dfsadmin -safemode wait	Wait until HDFS finishes data block replication

Hadoop Configuration Files

File	Description
hadoop-env.sh	Sets ENV variables for Hadoop
core-site.xml	Parameters for entire Hadoop cluster
hdfs-site.xml	Parameters for HDFS and its clients
mapred-site.xml	Parameters for MapReduce and its clients
masters	Host machines for secondary Namenode
slaves	List of slave hosts

Hadoop mradmin Commands

Command	Description
hadoop mradmin -safemode get	Check Job tracker status
hadoop mradmin -refreshQueues	Reload mapreduce configuration
hadoop mradmin -refreshNodes	Reload active TaskTrackers
hadoop mradmin -refreshServiceAcl	Force Jobtracker to reload service ACL
hadoop mradmin -refreshUserToGroupsMappings	Force jobtracker to reload user group mappings

Hadoop Balancer Commands

Command	Description
<code>start-balancer.sh</code>	Balance the cluster
<code>hadoop dfsadmin -setBalancerBandwidth <bandwidthinbytes></code>	Adjust bandwidth used by the balancer
<code>hadoop balancer -threshold 20</code>	Limit balancing to only 20% resources in the cluster

Hadoop Filesystem Commands

Command	Description
hadoop fs -mkdir mydir	Create a directory (mydir) in HDFS
hadoop fs -ls	List files and directories in HDFS
hadoop fs -cat myfile	View a file content
hadoop fs -du	Check disk space usage in HDFS
hadoop fs -expunge	Empty trash on HDFS
hadoop fs -chgrp hadoop file1	Change group membership of a file
hadoop fs -chown huser file1	Change file ownership
hadoop fs -rm file1	Delete a file in HDFS
hadoop fs -touchz file2	Create an empty file
hadoop fs -stat file1	Check the status of a file
hadoop fs -test -e file1	Check if file exists on HDFS
hadoop fs -test -z file1	Check if file is empty on HDFS
hadoop fs -test -d file1	Check if file1 is a directory on HDFS

Additional Hadoop Filesystem Commands

Command	Description
<code>hadoop fs -copyFromLocal <source> <destination></code>	Copy from local filesystem to HDFS
<code>hadoop fs -copyFromLocal file1 data</code>	e.g: Copies file1 from local FS to data dir in HDFS
<code>hadoop fs -copyToLocal <source> <destination></code>	copy from hdfs to local filesystem
<code>hadoop fs -copyToLocal data/file1 /var/tmp</code>	e.g: Copies file1 from HDFS data directory to /var/tmp on local FS
<code>hadoop fs -put <source> <destination></code>	Copy from remote location to HDFS
<code>hadoop fs -get <source> <destination></code>	Copy from HDFS to remote directory
<code>hadoop distcp hdfs://192.168.0.8:8020/input hdfs://192.168.0.8:8020/output</code>	Copy data from one cluster to another using the cluster URL
<code>hadoop fs -mv file:///data/datafile /user/hduser/data</code>	Move data file from the local directory to HDFS
<code>hadoop fs -setrep -w 3 file1</code>	Set the replication factor for file1 to 3
<code>hadoop fs -getmerge mydir bigfile</code>	Merge files in mydir directory and download it as one big file

21. Hadoop Test

Hadoop MCQ Quiz 1

1. 1. _____ is a platform **for** constructing data flows **for** extract, transform, and load (ETL) processing and analysis of large datasets.
 2. a. Pig Latin
 3. b. Oozie
 4. c. Pig
 5. d. Hive
 - 6.
7. 2. Point **out** the correct statement:
 8. a. Hive **is** not a relational database, but a query engine that supports the parts of SQL specific to querying data
 9. b. Hive **is** a relational database with SQL support
 10. c. Pig **is** a relational database with SQL support
 11. d. All of the mentioned
 - 12.
13. 3. _____ **is** general-purpose computing model and runtime system **for** distributed data analytics.
 14. a. Mapreduce
 15. b. Drill
 16. c. Oozie
 17. d. None of the mentioned
 - 18.
19. 4. The Pig Latin scripting language **is** not only a higher-level data flow language but also has operators similar to:
 20. a. SQL
 21. b. JSON
 22. c. XML
 23. d. All of the mentioned
 - 24.
25. 5. _____ Jobs are optimized **for** scalability but not latency.
 26. a. Mapreduce
 27. b. Drill
 28. c. Oozie
 29. d. Hive
 - 30.
31. 6. Which of the following platforms does Hadoop run on?
 32. a. Bare metal
 33. b. Debian

34. c. Cross-platform
35. d. UNIX-like
36.
37. 7. Hadoop achieves reliability by replicating the data across multiple hosts, and hence does not require _____ storage on hosts
38. a. RAID
39. b. Standard RAID levels
40. c. ZFS
41. d. Operating system
42.
43. 8. Above the file systems comes the _____ engine, which consists of one Job Tracker, to which client applications submit MapReduce jobs.
44. a. MapReduce
45. b. Google
46. c. Functional programming
47. d. Facebook
48.
49. 9. According to analysts, **for** what can traditional IT systems provide a foundation when they're integrated with big data technologies like Hadoop?
50. a. Big data management and data mining
51. b. Data warehousing and business intelligence
52. c. Management of Hadoop clusters
53. d. Collecting and storing unstructured data
54.
55.
56. 10. Hadoop **is** a framework that works with a variety of related tools. Common cohorts include:
57. a. MapReduce, Hive and Hbase
58. b. MapReduce, MySQL and Google Apps
59. c. MapReduce, Hummer and Iguana
60. d. MapReduce, Heron and Trumpet
61.
62. 11. Point **out** the wrong statement:
63. a. Hardtop's processing capabilities are huge and its real advantage lies **in** the ability to process terabytes & petabytes of data
64. b. Hadoop uses a programming model called "MapReduce", all the programs should conform to **this** model **in** order to work on Hadoop platform
65. c. The programming model, MapReduce, used by Hadoop **is** difficult to write and test
66. d. All of the mentioned
67.
68.

- 69.
- 70.
71. 12. All of the following accurately describe Hadoop, EXCEPT:
- Open source
 - Real-time
 - Java-based
 - Distributed computing approach
- 76.
77. 13. _____ has the world's largest Hadoop cluster.
- Apple
 - Datamatics
 - Facebook
 - None of the mentioned
- 82.
- 83.
84. 14. Point out the correct statement:
- Applications can use the Reporter to report progress
 - The Hadoop MapReduce framework spawns one map task **for** each InputSplit generated by the InputFormat **for** the job
 - The intermediate, sorted outputs are always stored **in** a simple (key-len, key, value-len, value) format
 - All of the mentioned
- 89.
90. 15. Point out the wrong statement:
- Reducer has 2 primary phases
 - increasing the number of reduces increases the framework overhead, but increases load balancing and lowers the cost of failures
 - It is legal to set the number of reduce-tasks to zero **if** no reduction is desired
 - The framework groups Reducer inputs by keys (since different mappers may have output the same key) **in** sort stage
- 95.
96. 16. _____ **is** a generalization of the facility provided by the MapReduce framework to collect data output by the Mapper or the Reducer
- Partitioner
 - OutputCollector
 - Reporter
 - All of the mentioned
- 101.
102. 17. _____ **is** the primary **interface for** a user to describe a MapReduce job to the Hadoop framework **for** execution.
103. a. Map Parameters

104. b. JobConf
105. c. MemoryConf
106. d. None of the mentioned
107.
108.
109. 18. Point out the correct statement:
110. a. MapReduce tries to place the data and the compute as close as possible
111. b. Map Task in MapReduce is performed using the Mapper() function
112. c. Reduce Task in MapReduce is performed using the Map() function
113. d. All of the mentioned
114.
115.
116.
117. 19. _____ Part of the MapReduce is responsible for processing one or more chunks of data and producing the output results.
118. a. Maptask
119. b. Mapper
120. c. Task execution
121. d. All of the mentioned
122.
123. 20. Point out the wrong statement:
124. a. A MapReduce job usually splits the input data-
set into independent chunks which are processed by the map tasks in a completely parallel manner
125. b. The MapReduce framework operates exclusively on pairs
126. c. Applications typically implement the Mapper and Reducer interfaces to provide the map and reduce methods
127. d. None of the mentioned
128.
129. 21. _____ Maps input key/value pairs to a set of intermediate key/value pairs.
130. a. Mapper
131. b. Reducer
132. c. Both Mapper and Reducer
133. d. None of the mentioned
134.
135. 22. The number of maps is usually driven by the total size of:
136. a. inputs
137. b. outputs
138. c. tasks
139. d. None of the mentioned

- 140.
141. 23. _____ is the **default** Partitioner **for** partitioning key space.
a. HashPar
b. Partitioner
c. HashPartitioner
d. None of the mentioned
- 142.
- 143.
- 144.
- 145.
- 146.
147. 24. _____ gives site-specific configuration **for** a given hadoop installation.
a. core-default.xml
b. core-site.xml
c. coredefault.xml
d. All of the mentioned
- 148.
- 149.
- 150.
- 151.
- 152.
153. 25. Administrators typically define parameters **as final** **in** _____ **for** values that user applications may not alter
a. core-default.xml
b. core-site.xml
c. coredefault.xml
d. All of the mentioned
- 154.
- 155.
- 156.
- 157.
- 158.
159. 26. _____ is a framework **for** collecting and storing script-level statistics **for** Pig Latin.
a. Pig Stats
b. PStatistics
c. Pig Statistics
d. None of the mentioned
- 160.
- 161.
- 162.
- 163.
- 164.
- 165.
166. 27. PigUnit runs **in** Pig's _____ mode by **default**.
a. local
b. tez
c. mapreduce
d. None of the mentioned
- 167.
- 168.
- 169.
- 170.
- 171.
172. 28. Pig operates **in** mainly how many modes?
a. Two
b. Three
c. Four
d. Five
- 173.
- 174.
- 175.
- 176.
- 177.
178. 29. Point **out** the correct statement:

179. a. You can run Pig **in** either mode **using** the "pig" command
180. b. You can run Pig **in** batch mode **using** the Grunt shell
181. c. You can run Pig **in** interactive mode **using** the FS shell
182. d. None of the mentioned
- 183.
184. 30. You can run Pig **in** batch mode **using** _____.
185. a. Pig shell command
186. b. Pig scripts
187. c. Pig options
188. d. All of the mentioned
- 189.
- 190.
- 191.
- 192.
193. 31. Pig Latin statements are generally organized **in** one of the following ways:
194. a. A LOAD statement to read data from the file system
195. b. A series of "transformation" statements to process the data
196. c. A DUMP statement to view results or a STORE statement to save the results
197. d. All of the mentioned
- 198.
199. 32. Point **out** the wrong statement
200. a. To run Pig **in** local mode, you need access to a single machine
201. b. The DISPLAY **operator** will display the results to your terminal screen
202. c. To run Pig **in** mapreduce mode, you need access to a Hadoop cluster and HDFSins
tallation
203. d. All of the mentioned
- 204.
205. 33. Which of the following function **is** used to read data **in** PIG?
206. a. WRITE
207. b. READ
208. c. LOAD
209. d. None of the mentioned
- 210.
211. 34. You can run Pig **in** interactive mode **using** the _____ shell.
212. a. Grunt
213. b. FS
214. c. HDFS
215. d. None of the mentioned
- 216.
217. 35. Which of the following **is** the **default** mode **in** Pig?
218. a. Mapreduce

219. b. Tez
220. c. Local
221. d. All of the mentioned
222.
223. 36. Which of the following will run pig **in** local mode?
a. \$ pig -x local ...
b. \$ pig -x tez_local ...
c. \$ pig ...
d. None of the mentioned
228.
229. 37. \$ pig -x tez_local ... will enable _____ mode **in** Pig.
a. Mapreduce
b. Tez
c. Local
d. None of the mentioned
234.
235. 38. Records are terminated by a _____ character.
a. RECORD_DELIMITER
b. FIELD_DELIMITER
c. FIELD_LIMITER
d. None of the mentioned
240.
241.
242.
243.
244. 39. Point **out** the correct statement:
a. Data locality means movement of algorithm to the data instead of data to algorithm
b. When the processing **is** done on the data algorithm **is** moved across the Action Nodes rather than data to the algorithm
c. Moving Computation **is** expensive than Moving Data
d. None of the mentioned
249.
250. 40. The daemons associated with the MapReduce phase are -_____ and task-trackers.
a. job-tracker
b. map-tracker
c. reduce-tracker
d. All of the mentioned
255.

256. 41. The JobTracker pushes work **out** to available _____ nodes **in** the cluster, striving to keep the work **as** close to the data **as** possible
257. a. DataNodes
258. b. TaskTracker
259. c. ActionNodes
260. d. All of the mentioned
- 261.
262. 42. Point **out** the wrong statement:
263. a. The map function **in** Hadoop MapReduce have the following general form: map :(K1, V1) → list (K2, V2)
264. b. The reduce function **in** Hadoop MapReduce have the following general form: reduce: (K2, list (V2)) → list (K3, V3)
265. c. MapReduce has a complex model of data processing: inputs and outputs **for** the map and reduce functions are key-value pairs
266. d. None of the mentioned
- 267.
268. 43. _____ controls the partitioning of the keys of the intermediate map-outputs.
269. a. Collector
270. b. Partitioner
271. c. InputFormat
272. d. None of the mentioned
- 273.
274. 44. Output of the mapper **is** first written on the local disk **for** sorting and _____ process.
275. a. shuffling
276. b. secondary sorting
277. c. forking
278. d. reducing
- 279.
280. 45. A _____ serves **as** the master and there **is** only one NameNode per cluster.
281. a. Data Node
282. b. NameNode
283. c. Data block
284. d. Replication
- 285.
- 286.
- 287.
- 288.
- 289.
- 290.

291. 46. Point out the correct statement:
292. a. DataNode is the slave/worker node and holds the user data in the form of Data Blocks
293. b. Each incoming file is broken into 32 MB by default
294. c. Data blocks are replicated across different nodes in the cluster to ensure a low degree of fault tolerance
295. d. None of the mentioned
- 296.
297. 47. HDFS works in a _____ fashion.
298. a. master-worker
299. b. master-slave
300. c. worker/slave.
301. d. All of the mentioned
- 302.
303. 48. _____ NameNode is used when the Primary NameNode goes down.
304. a. Rack
305. b. Data
306. c. Secondary
307. d. None of the mentioned
- 308.
309. 49. Which of the following scenario may not be a good fit for HDFS?
310. a. HDFS is not suitable for scenarios requiring multiple/simultaneous writes to the same file
311. b. HDFS is suitable for storing data related to applications requiring low latency data access
312. c. HDFS is suitable for storing data related to applications requiring low latency data access
313. d. None of the mentioned
314. 50. The need for data replication can arise in various scenarios like:
315. a. Replication Factor is changed
316. b. DataNode goes down
317. c. Data Blocks get corrupted
318. d. All of the mentioned
- 319.
320. 51. _____ is the slave/worker node and holds the user data in the form of Data Blocks.
321. a. DataNode
322. b. NameNode
323. c. Data block
324. d. Replication
- 325.

326. 52. HDFS provides a command line **interface** called _____ used to interact with HDFS.
327. a. "HDFS Shell"
328. b. "FS Shell"
329. c. "DFS Shell"
330. d. None of the mentioned
- 331.
332. 53. HDFS **is** implemented in _____ programming language.
333. a. C++
334. b. Java
335. c. Scala
336. d. None of the mentioned
- 337.
338. 54. In order to read any file **in** HDFS, instance of _____ **is** required.
339. a. filesystem
340. b. DataStream
341. c. outstream
342. d. inputstream
- 343.
344. 55. Point **out** the wrong statement:
345. a. The framework calls reduce method **for** each pair **in** the grouped inputs
346. b. The output of the Reducer **is** re-sorted
347. c. reduce method reduces values **for** a given key
348. d. None of the mentioned
- 349.
350. 56. Interface _____ reduces a **set** of intermediate values which share a key to a smaller **set** of values.
351. a. Mapper
352. b. Reducer
353. c. Writable
354. d. Readable
- 355.
356. 57. Reducer' s input **is** the grouped output of a:
357. a. Mapper
358. b. Reducer
359. c. Writable
360. d. Readable
- 361.
362. 58. The output of the reduce task **is** typically written to the Filesystem via:
363. a. OutputCollector
364. b. InputCollector

365. c. OutputCollect

366. d. All of the mentioned

Hadoop MCQ Quiz 2

1. 1. Due to Hadoop's ability to manage unstructured and semi-structured data and because of its scale-out support for handling ever-growing quantities of data, many experts view it as a replacement for the enterprise data warehouse.
2. a. True
3. b. False
4.
5. 2. Flume server resides inside HDFS
6. a. True
7. b. False
8.
9. 3. Flume and Sqoop do the same job with only difference that flume works with txt files and Sqoop works with databases
10. a. True
11. b. False
12.
13. 4. Hadoop cannot store Structured data
14. a. True
15. b. False
16.
17. 5. If there are 2 Shuffles and Sorts then there are 2 mappers and 1 reducer
18. a. True
19. b. False
20.
21. 6. We cannot configure the Source and Sink of Flume
22. a. True
23. b. False
24.
25. 7. Combiner is also called as Mini Mapper
26. a. True
27. b. False
28.
29. 8. We can use Partitioner even if there is only one Reducer.
30. a. True
31. b. False
32.
33.
34. 9. In case of failure of Namenode, the Secondary Namenode can take its place
35. a. True

36. b. False
- 37.
38. 10. Hadoop cannot work with commodity Hardware
39. a. True
40. b. False
- 41.
42. 11. Hive and Pig are exactly the same
43. a. True
44. b. False
- 45.
46. 12. Number of Mappers are 7 and number of Reducers is 0 then number of output files is 7.
47. a. True
48. b. False
- 49.
50. 13. Flume sits between Webserver and Hadoop cluster.
51. a. True
52. b. False
- 53.
54. 14. MapReduce can process data stored either in filesystem (unstructured) or in a database (structured).
55. a. True
56. b. False
- 57.
58. 15. Hadoop is a replacement to RDBMS.
59. a. True
60. b. False
- 61.
62. 16. All MapReduce implementations implement exactly same algorithm.
63. a. True
64. b. False
- 65.
- 66.
67. 17. For using Hadoop to process your data, the data has to be moved/ingested into HDFS.
68. a. True
69. b. False
- 70.
71. 18. Use of commodity hardware is suitable for slave nodes.
72. a. True
73. b. False
- 74.
75. 19. Partitioner reduces traffic from Reducer to Mapper.

76. a. True
77. b. False
78.
79. 20. Hadoop runs the jobs by dividing it into Map tasks and Reduce tasks.
80. a. True
81. b. False
82.
83. 21. Hadoop can handle petabytes of data which are stored across hundreds or thousands of nodes and physical storage servers.
84. a. True
85. b. False
86.
87. 22. HDFS instances are classified into two components: the nameNode, which maintains metadata to keep the track on placement of physical data across the Hadoop instance and dataNodes, which store the data.
88. a. True
89. b. False
90.
91. 23. Hadoop is a framework that works with a variety of related tools. Common stack includes:
92. a. MapReduce, Hive and HBase
93. b. MapReduce, MySQL and Google Apps
94. c. MapReduce, Hummer and Iguana
95. d. MapReduce, Heron and Trumpet
96.
97.
98.
99. 24. The MapReduce programming model widely used in analytics was developed at _____.
100. a. Apache Foundation
101. b. Google
102. c. Microsoft Research
103. d. Apple
104.
105. 25. What was Hadoop named after?
106. a. Creator Doug Cutting's favorite circus act
107. b. Cutting's high school rock band
108. c. The toy elephant of Cutting's son
109. d. A sound Cutting's laptop made during Hadoop's development
110.
111. 26. Flume is a _____.

112. a. Import Export tool
113. b. Messaging Service
114. c. Scheduling and Workflow utility
115. d. Data flow language
116.
117. **27.** Which file is used **for** updating mapreduce settings?
118. a. core-site
119. b. mapred-site
120. c. hdfs-site
121. d. hadoop-env.sh
122.
123. **28.** Which programming model is also used as the associated implementation **for** processing and generating large data sets with a parallel, distributed algorithm on a cluster in Hadoop?
124. a. MapReduce
125. b. Pig
126. c. Hive
127. d. SQL
128.
129. **29.** _____ is an optimized file system **for** distributed processing of very large datasets on commodity hardware.
130. a. RDBMS
131. b. Flat File System
132. c. HDFS File System
133. d. MapReduce
134.
135. **30.** Which one of the following is an example of unstructured data?
136. a. call detail records
137. b. web server logs
138. c. sensory data
139. d. Voicemail
140.
141. **31.** Which of the following is semi-structured format?
142. a. email
143. b. accord
144. c. hippa
145. d. word document
146.
147. **32.** If number of Reducers is **0** then Output of Mapper is stored in _____.
148. a. Hadoop File system

149. b. Linux File system
150. c. Mapper Filesystem
151. d. Partitioner Filesystem
152.
153. **33.** What is the **default** Block size?
154. a. 128Mb
155. b. 64Mb
156. c. 256Mb
157. d. 512Mb
158.
159. **34.** Which file is used **for** updating mapreduce settings?
160. a. core-site
161. b. mapred-site
162. c. hdfs-site
163. d. hadoop-env.sh
164.
165. **35.** Partitioner is used to
166. a. Reduce work of Reducer
167. b. Reduce work of Combiner
168. c. Chose one Reducer when there are more than **1** Mapers
169. d. Chose one Reducer when there are more than **1** Reducers
170.
171.
172.
173. **36.** All of the following accurately describe Hadoop, EXCEPT:
174.
175. a. Open Source
176. b. Real Time
177. c. Java Based
178. d. Distributed Processing
179.
180. **37.** Hadoop is a framework that works with a variety of related tools. Common coh
rts include:
181. a. Pig, Hive,Oozie
182. b. Pig, Sheep, Goat
183. c. Linux, Windows, Hadoop
184. d. Pig, Hive, JDK
185.
186. **38.** Consider **case** scenario: In M/R system, - HDFS block size is **64** MB. We have **3** file
s of size 64K, 65Mb and 127Mb. How many input splits will be made by Hadoop framework?

187. a. 1
188. b. 2
189. c. 15
190. d. 5
191.
192. 39. Compulsory Phases of Hadoop are
a. Map, Reduce
b. Map, Shuffle and Sort, Reduce
c. Map, Combiner, Partitioner, Shuffle and Sort, Reduce
d. Map, Combiner, Reduce
193.
194.
195.
196.
197.
198. 40. If the number of Mappers is 3, then the number of Combiners is
a. 1
b. 2
c. 3
d. 4
199.
200.
201.
202.
203.
204. 41. If number of Reducers is 6, then number of Partitioner is
a. 1
b. 3
c. 5
d. 6
205.
206.
207.
208.
209. 42. The port number of namenode is
a. 8020
b. 8021
c. 8022
d. 8024
210.
211.
212.
213.
214.
215. 43. The port number of Job Tracker is
a. 8020
b. 8021
c. 8022
d. 8023
216.
217.
218.
219.
220.
221. 44. Url of Namenode is
a. localhost:55070
b. localhost:55030
c. localhost:50090
d. localhost:60090
222.
223.
224.
225.
226.
227. 45. Sqoop can be used to

- 228.
229. a. Only Import data from SQL Databases
230. b. Import Export data from to SQL Databases
231. c. Only Export data to SQL Databases
232. d. Delete data from SQL Databases
- 233.
234. 46. Combiner can be used only when the operation is
235. a. Commutative and Associative
236. b. Commutative and Descriptive
237. c. Associative and Descriptive
238. d. Only descriptive
- 239.
240. 47. Which of the following statements is correct? In Hadoop there exists
241. a. one JobTracker per Hadoop job
242. b. one JobTracker per Mapper
243. c. one JobTracker per node
244. d. one JobTracker per cluster
- 245.
- 246.
- 247.
248. 48. Which of the following is true for Namenode?
249. a. The master node that manages the file system namespace.
250. b. It maintains the file system tree and metadata for all the files/directories within.
251. c. Keeps track of the location of all the datanodes for a given file.
252. d. All of the above

22. Hadoop Interview Questions

1. Explain "Big Data" and what are five V's of Big Data?

"Big data" is the term for a collection of large and complex data sets, that makes it difficult to process using relational database management tools or traditional data processing applications. It is difficult to capture, curate, store, search, share, transfer, analyze, and visualize Big data. Big Data has emerged as an opportunity for companies. Now they can successfully derive value from their data and will have a distinct advantage over their competitors with enhanced business decisions making capabilities.

Volume: The volume represents the amount of data which is growing at an exponential rate i.e. in Petabytes and Exabytes.

Velocity: Velocity refers to the rate at which data is growing, which is very fast. Today, yesterday's data are considered as old data. Nowadays, social media is a major contributor in the velocity of growing data.

Variety: Variety refers to the heterogeneity of data types. In another word, the data which are gathered has a variety of formats like videos, audios, csv, etc. So, these various formats represent the variety of data.

Veracity: Veracity refers to the data in doubt or uncertainty of data available due to data inconsistency and incompleteness. Data available can sometimes get messy and maybe difficult to trust. With many forms of big data, quality and accuracy are difficult to control. The volume is often the reason behind for the lack of quality and accuracy in the data.

Value: It is all well and good to have access to big data but unless we can turn it into a value it is useless. By turning it into value I mean, Is it adding to the benefits of the organizations? Is the organization working on Big Data achieving high ROI (Return On Investment)? Unless, it adds to their profits by working on Big Data, it is useless.

2. What is Hadoop and its components.

When “Big Data” emerged as a problem, Apache Hadoop evolved as a solution to it. Apache Hadoop is a framework which provides us various services or tools to store and process Big Data. It helps in analyzing Big Data and making business decisions out of it, which can’t be done efficiently and effectively using traditional systems.

Storage unit – HDFS (NameNode, DataNode)

Processing framework – YARN (ResourceManager, NodeManager)

3. What are HDFS and YARN?

HDFS (Hadoop Distributed File System) is the storage unit of Hadoop. It is responsible for storing different kinds of data as blocks in a distributed environment. It follows master and slave topology.

NameNode: NameNode is the master node in the distributed environment and it maintains the metadata information for the blocks of data stored in HDFS like block location, replication factors etc.

DataNode: DataNodes are the slave nodes, which are responsible for storing data in the HDFS. NameNode manages all the DataNodes.

YARN (Yet Another Resource Negotiator) is the processing framework in Hadoop, which manages resources and provides an execution environment to the processes.

ResourceManager: It receives the processing requests, and then passes the parts of requests to corresponding NodeManagers accordingly, where the actual processing takes place. It allocates resources to applications based on the needs.

NodeManager: NodeManager is installed on every DataNode and it is responsible for execution of the task on every single DataNode.

4. Tell me about the various Hadoop daemons and their roles in a Hadoop cluster.

Generally approach this question by first explaining the HDFS daemons i.e. NameNode, DataNode and Secondary NameNode, and then moving on to the YARN daemons i.e. ResourceManager and NodeManager, and lastly explaining the JobHistoryServer.

NameNode: It is the master node which is responsible for storing the metadata of all the files and directories. It has information about blocks, that make a file, and where those blocks are located in the cluster.

Datanode: It is the slave node that contains the actual data.

Secondary NameNode: It periodically merges the changes (edit log) with the FslImage (Filesystem Image), present in the NameNode. It stores the modified FslImage into persistent storage, which can be used in case of failure of NameNode.

ResourceManager: It is the central authority that manages resources and schedule applications running on top of YARN.

NodeManager: It runs on slave machines, and is responsible for launching the application's containers (where applications execute their part), monitoring their resource usage (CPU, memory, disk, network) and reporting these to the ResourceManager.

JobHistoryServer: It maintains information about MapReduce jobs after the Application Master terminates.

5. Compare HDFS with Network Attached Storage (NAS).

In this question, first explain NAS and HDFS, and then compare their features as follows:

Network-attached storage (NAS) is a file-level computer data storage server connected to a computer network providing data access to a heterogeneous group of clients. NAS can either be a hardware or software which provides services for storing and accessing files. Whereas Hadoop Distributed File System (HDFS) is a distributed filesystem to store data using commodity hardware.

In HDFS Data Blocks are distributed across all the machines in a cluster. Whereas in NAS data is stored on a dedicated hardware.

HDFS is designed to work with MapReduce paradigm, where computation is moved to the data. NAS is not suitable for MapReduce since data is stored separately from the computations.

HDFS uses commodity hardware which is cost effective, whereas a NAS is a high-end storage devices which includes high cost.

6. What are the basic differences between relational database and HDFS?

Here are the key differences between HDFS and relational database:

RDBMS vs. Hadoop

	RDBMS	Hadoop
Data Types	RDBMS relies on the structured data and the schema of the data is always known.	Any kind of data can be stored into Hadoop i.e. Be it structured, unstructured or semi-structured.
Processing	RDBMS provides limited or no processing capabilities.	Hadoop allows us to process the data which is distributed across the cluster in a parallel fashion.
Schema on Read Vs. Write	RDBMS is based on ‘schema on write’ where schema validation is done before loading the data.	On the contrary, Hadoop follows the schema on read policy.
Read/Write Speed	In RDBMS, reads are fast because the schema of the data is already known.	The writes are fast in HDFS because no schema validation happens during HDFS write.
Cost	Licensed software, therefore, I have to pay for the software.	Hadoop is an open source framework. So, I don’t need to pay for the software.
Best Fit Use Case	RDBMS is used for OLTP (Online Transactional Processing) system.	Hadoop is used for Data discovery, data analytics or OLAP system.

7. List the difference between Hadoop 1 and Hadoop 2.

This is an important question and while answering this question, we have to mainly focus on two points i.e. Passive NameNode and YARN architecture.

In Hadoop 1.x, “NameNode” is the single point of failure. In Hadoop 2.x, we have Active and Passive “NameNodes”. If the active “NameNode” fails, the passive “NameNode” takes charge. Because of this, high availability can be achieved in Hadoop 2.x.

Also, in Hadoop 2.x, YARN provides a central resource manager. With YARN, you can now run multiple applications in Hadoop, all sharing a common resource. MRV2 is a particular type of distributed application that runs the MapReduce framework on top of YARN. Other tools can also perform data processing via YARN, which was a problem in Hadoop 1.x.

Hadoop 1.x vs. Hadoop 2.x

	Hadoop 1.x	Hadoop 2.x
Passive NameNode	NameNode is a Single Point of Failure	Active & Passive NameNode
Processing	MRV1 (Job Tracker & Task Tracker)	MRV2/YARN (ResourceManager & NodeManager)

8. What are active and passive “NameNodes” ?

In HA (High Availability) architecture, we have two NameNodes – Active “NameNode” and Passive “NameNode” .

Active “NameNode” is the “NameNode” which works and runs in the cluster.

Passive “NameNode” is a standby “NameNode” , which has similar data as active “NameNode” .

When the active “NameNode” fails, the passive “NameNode” replaces the active “NameNode” in the cluster. Hence, the cluster is never without a “NameNode” and so it never fails.

9. Why does one remove or add nodes in a Hadoop cluster frequently?

One of the most attractive features of the Hadoop framework is its *utilization of commodity hardware*. However, this leads to frequent “DataNode” crashes in a Hadoop cluster. Another striking feature of Hadoop Framework is the *ease of scale* in accordance with the rapid growth in data volume. Because of these two reasons, one of the most common task of a Hadoop administrator is to commission (Add) and decommission (Remove) “Data Nodes” in a Hadoop Cluster.

10. What happens when two clients try to access the same file in the HDFS?

HDFS supports exclusive writes only.

When the first client contacts the “NameNode” to open the file for writing, the “NameNode” grants a lease to the client to create this file. When the second client tries to open the same file for writing, the “NameNode” will notice that the lease for the file is already granted to another client, and will reject the open request for the second client.

11. How does NameNode tackle DataNode failures?

NameNode periodically receives a Heartbeat (signal) from each of the DataNode in the cluster, which implies DataNode is functioning properly.

A block report contains a list of all the blocks on a DataNode. If a DataNode fails to send a heartbeat message, after a specific period of time it is marked dead.

The NameNode replicates the blocks of dead node to another DataNode using the replicas created earlier.

12. What will you do when NameNode is down?

The NameNode recovery process involves the following steps to make the Hadoop cluster up and running:

Use the file system metadata replica (FsImage) to start a new NameNode.

Then, configure the DataNodes and clients so that they can acknowledge this new NameNode, that is started.

Now the new NameNode will start serving the client after it has completed loading the last checkpoint FsImage (for metadata information) and received enough block reports from the DataNodes.

13. What is a checkpoint?

In brief, “Checkpointing” is a process that takes an FsImage, edit log and compacts them into a new FsImage. Thus, instead of replaying an edit log, the NameNode can load the final in-memory state directly from the FsImage. This is a far more efficient operation and reduces NameNode startup time. Checkpointing is performed by Secondary NameNode.

14. How is HDFS fault tolerant?

When data is stored over HDFS, NameNode replicates the data to several DataNode. The default replication factor is 3. You can change the configuration factor as per your need. If a DataNode goes down, the NameNode will automatically copy the data to another node from the replicas and make the data available. This provides fault tolerance in HDFS.

15. Can NameNode and DataNode be a commodity hardware?

The smart answer to this question would be, DataNodes are commodity hardware like personal computers and laptops as it stores data and are required in a large number. But from your experience you can tell that, NameNode is the master node and it stores metadata about all the blocks stored in HDFS. It requires high memory (RAM) space, so NameNode needs to be a high-end machine with good memory space.

16. Why do we use HDFS for applications having large data sets and not when there are a lot of small files?

HDFS is more suitable for large amounts of data sets in a single file as compared to small amount of data spread across multiple files. As you know, the NameNode stores the metadata information regarding file system in the RAM. Therefore, the amount of memory produces a limit to the number of files in my HDFS file system. In other words, too much of files will lead to generation of too much meta data. And, storing these meta data in the RAM will become a challenge. As a thumb rule, metadata for a file, block or directory takes 150 bytes.

17. How do you define “block” in HDFS? What is the default block size in Hadoop 1 and in Hadoop 2? Can it be changed?

Blocks are the nothing but the smallest continuous location on your hard drive where data is stored. HDFS stores each as blocks, and distribute it across the Hadoop cluster. Files in HDFS are broken down into block-sized chunks, which are stored as independent units.

Hadoop 1 default block size: 64 MB

Hadoop 2 default block size: 128 MB

Yes, blocks can be configured. The `dfs.block.size` parameter can be used in the `hdfs-site.xml` file to set the size of a block in a Hadoop environment.

18. What does ‘jps’ command do?

The ‘jps’ command helps us to check if the Hadoop daemons are running or not. It shows all the Hadoop daemons i.e namenode, datanode, resourcemanager, nodemanager etc. that are running on the machine.

19. How do you define “Rack Awareness” in Hadoop?

Rack Awareness is the algorithm in which the “NameNode” decides how blocks and their replicas are placed, based on rack definitions to minimize network traffic between “DataNodes” within the same rack. Let’s say we consider replication factor 3 (default), the policy is that “for every block of data, two copies will exist in one rack, third copy in a different rack”. This rule is known as the “Replica Placement Policy” .

20. What is “speculative execution” in Hadoop?

If a node appears to be executing a task slower, the master node can redundantly execute another instance of the same task on another node. Then, the task which finishes first will be accepted and the other one is killed. This process is called “speculative execution” .

21. How can I restart “NameNode” or all the daemons in Hadoop?

This question can have two answers, we will discuss both the answers. We can restart NameNode by following methods:

You can stop the NameNode individually using. **`/sbin/hadoop-daemon.sh stop namenode`** command and then start the NameNode using. **`/sbin/hadoop-daemon.sh start namenode`** command.

To stop and start all the daemons, use. **`/sbin/stop-all.sh`** and then use **`/sbin/start-all.sh`** command which will stop all the daemons first and then start all the daemons.

These script files reside in the sbin directory inside the Hadoop directory.

22. What is the difference between an “HDFS Block” and an “Input Split” ?

The “HDFS Block” is the physical division of the data while “Input Split” is the logical division of the data. HDFS divides data in blocks for storing the blocks together, whereas for processing, MapReduce divides the data into the input split and assign it to mapper function.

23. Name the three modes in which Hadoop can run.

The three modes in which Hadoop can run are as follows:

Standalone (local) mode: This is the default mode if we don’t configure anything. In this mode, all the components of Hadoop, such NameNode, DataNode, ResourceManager, and NodeManager, run as a single Java process. This uses local filesystem.

Pseudo distributed mode: A single-node Hadoop deployment is considered as running Hadoop system in pseudo-distributed mode. In this mode, all the Hadoop services, including both the master and the slave services, were executed on a single compute node.

Fully distributed mode: A Hadoop deployments in which the Hadoop master and slave services run on separate nodes, are stated as fully distributed mode.

24. What is “MapReduce” ? What is the syntax to run a “MapReduce” program?

It is a framework/a programming model that is used for processing large data sets over a cluster of computers using parallel programming. The syntax to run a MapReduce program is `hadoop_jar_file.jar /input_path /output_path`.

25. What are the main configuration parameters in a “MapReduce” program?

The main configuration parameters which users need to specify in “MapReduce” framework are:

Job’ s input locations in the distributed file system

Job’ s output location in the distributed file system

Input format of data

Output format of data

Class containing the map function

Class containing the reduce function

JAR file containing the mapper, reducer and driver classes

26. State the reason why we can’ t perform “aggregation” (addition) in mapper? Why do we need the “reducer” for this?

This answer includes many points, so we will go through them sequentially.

We cannot perform “aggregation” (addition) in mapper because sorting does not occur in the “mapper” function. Sorting occurs only on the reducer side and without sorting aggregation cannot be done.

During “aggregation” , we need output of all the mapper functions which may not be possible to collect in the map phase as mappers may be running on different machine where the data blocks are stored.

And lastly, if we try to aggregate data at mapper, it requires communication between all mapper functions which may be running on different machines. So, it will consume high network bandwidth and can cause network bottlenecking.

27. What is the purpose of “RecordReader” in Hadoop?

The “InputSplit” defines a slice of work, but does not describe how to access it. The “RecordReader” class loads the data from its source and converts it into (key, value) pairs suitable for reading by the “Mapper” task. The “RecordReader” instance is defined by the “Input Format” .

28. Explain “Distributed Cache” in a “MapReduce Framework” .

Distributed Cache can be explained as, a facility provided by the MapReduce framework to cache files needed by applications. Once you have cached a file for your job, Hadoop framework will make it available on each and every data nodes where you map/reduce tasks are running. Then you can access the cache file as a local file in your Mapper or Reducer job.

29. How do “reducers” communicate with each other?

This is a tricky question. The “MapReduce” programming model does not allow “reducers” to communicate with each other. “Reducers” run in isolation.

30. What does a “MapReduce Partitioner” do?

A “MapReduce Partitioner” makes sure that all the values of a single key go to the same “reducer”, thus allowing even distribution of the map output over the “reducers”. It redirects the “mapper” output to the “reducer” by determining which “reducer” is responsible for the particular key.

31. How will you write a custom partitioner?

Custom partitioner for a Hadoop job can be written easily by following the below steps:

Create a new class that extends Partitioner Class

Override method – getPartition, in the wrapper that runs in the MapReduce.

Add the custom partitioner to the job by using method set Partitioner or add the custom partitioner to the job as a config file.

32. What is a “Combiner” ?

A “Combiner” is a mini “reducer” that performs the local “reduce” task. It receives the input from the “mapper” on a particular “node” and sends the output to the “reducer” .

“Combiners” help in enhancing the efficiency of “MapReduce” by reducing the quantum of data that is required to be sent to the “reducers” .

33. What do you know about “SequenceFileInputFormat” ?

“SequenceFileInputFormat” is an input format for reading within sequence files. It is a specific compressed binary file format which is optimized for passing the data between the outputs of one “MapReduce” job to the input of some other “MapReduce” job.

Sequence files can be generated as the output of other MapReduce tasks and are an efficient intermediate representation for data that is passing from one MapReduce job to another.

34. What are the benefits of Apache Pig over MapReduce?

Apache Pig is a platform, used to analyze large data sets representing them as data flows developed by Yahoo. It is designed to provide an abstraction over MapReduce, reducing the complexities of writing a MapReduce program.

Pig Latin is a high-level data flow language, whereas MapReduce is a low-level data processing paradigm.

Without writing complex Java implementations in MapReduce, programmers can achieve the same implementations very easily using Pig Latin.

Apache Pig reduces the length of the code by approx 20 times (according to Yahoo). Hence, this reduces the development period by almost 16 times.

Pig provides many built-in operators to support data operations like joins, filters, ordering, sorting etc. Whereas to perform the same function in MapReduce is a humongous task.

Performing a Join operation in Apache Pig is simple. Whereas it is difficult in MapReduce to perform a Join operation between the data sets, as it requires multiple MapReduce tasks to be executed sequentially to fulfill the job.

In addition, pig also provides nested data types like tuples, bags, and maps that are missing from MapReduce.

35. What are different data types in Pig Latin?

Pig Latin can handle both atomic data types like int, float, long, double etc. and complex data types like tuple, bag and map.

Atomic data types: Atomic or scalar data types are the basic data types which are used in all the languages like string, int, float, long, double, char[], byte[]].

Complex Data Types: Complex data types are Tuple, Map and Bag.

36. What are the different relational operations in “Pig Latin” you worked with?

Different relational operators are:

- for each
- order by
- filters
- group
- distinct
- join
- limit

37. What is a UDF?

If some functions are unavailable in built-in operators, we can programmatically create User Defined Functions (UDF) to bring those functionalities using other languages like Java, Python, Ruby, etc. and embed it in Script file.

38. What is “SerDe” in “Hive” ?

Apache Hive is a data warehouse system built on top of Hadoop and is used for analyzing structured and semi-structured data developed by Facebook. Hive abstracts the complexity of Hadoop MapReduce.

The “SerDe” interface allows you to instruct “Hive” about how a record should be processed. A “SerDe” is a combination of a “Serializer” and a “Deserializer”. “Hive” uses “SerDe” (and “FileFormat”) to read and write the table’s row.

39. Can the default “Hive Metastore” be used by multiple users (processes) at the same time?

“Derby database” is the default “Hive Metastore”. Multiple users (processes) cannot access it at the same time. It is mainly used to perform unit tests.

40. What is the default location where “Hive” stores table data?

The default location where Hive stores table data is inside HDFS in /user/hive/warehouse.

41. What is Apache HBase?

HBase is an open source, multidimensional, distributed, scalable and a NoSQL database written in Java. HBase runs on top of HDFS (Hadoop Distributed File System) and provides BigTable (Google) like capabilities to Hadoop. It is designed to provide a fault tolerant way of storing large collection of sparse data sets. HBase achieves high throughput and low latency by providing faster Read/Write Access on huge data sets.

42. What are the components of Apache HBase?

HBase has three major components, i.e. HMaster Server, HBase RegionServer and Zookeeper.

Region Server: A table can be divided into several regions. A group of regions is served to the clients by a Region Server.

HMaster: It coordinates and manages the Region Server (similar as NameNode manages DataNode in HDFS).

ZooKeeper: Zookeeper acts like as a coordinator inside HBase distributed environment. It helps in maintaining server state inside the cluster by communicating through sessions.

43. What are the components of Region Server?

The components of a Region Server are:

WAL: Write Ahead Log (WAL) is a file attached to every Region Server inside the distributed environment. The WAL stores the new data that hasn't been persisted or committed to the permanent storage.

Block Cache: Block Cache resides in the top of Region Server. It stores the frequently read data in the memory.

MemStore: It is the write cache. It stores all the incoming data before committing it to the disk or permanent memory. There is one MemStore for each column family in a region.

HFile: HFile is stored in HDFS. It stores the actual cells on the disk.

44. Explain "WAL" in HBase?

Write Ahead Log (WAL) is a file attached to every Region Server inside the distributed environment. The WAL stores the new data that hasn't been persisted or committed to the permanent storage. It is used in case of failure to recover the data sets.

45. Mention the differences between “HBase” and “Relational Databases” ?

HBase is an open source, multidimensional, distributed, scalable and a *NoSQL database* written in Java. HBase runs on top of HDFS and provides BigTable like capabilities to Hadoop. Let us see the differences between HBase and relational database.

HBase vs. Relational Database

HBase	Relational Database
It is schema-less	It is schema based database
It is column-oriented data store	It is row-oriented data store
It is used to store de-normalized data	It is used to store normalized data
It contains sparsely populated tables	It contains thin tables
Automated partitioning is done in HBase	There is no such provision or built-in support for partitioning

46. What is Apache Spark?

The answer to this question is, Apache Spark is a framework for real time data analytics in a distributed computing environment. It executes in-memory computations to increase the speed of data processing.

It is 100x faster than MapReduce for large scale data processing by exploiting in-memory computations and other optimizations.

47. Can you build “Spark” with any particular Hadoop version?

Yes, one can build “Spark” for a specific Hadoop version.

48. Define RDD.

RDD is the acronym for Resilient Distribution Datasets – a fault-tolerant collection of operational elements that run parallel. The partitioned data in RDD are immutable and distributed, which is a key component of Apache Spark.

49. What is Apache ZooKeeper and Apache Oozie?

Apache ZooKeeper coordinates with various services in a distributed environment. It saves a lot of time by performing synchronization, configuration maintenance, grouping and naming.

Apache Oozie is a scheduler which schedules Hadoop jobs and binds them together as one logical work. There are two kinds of Oozie jobs:

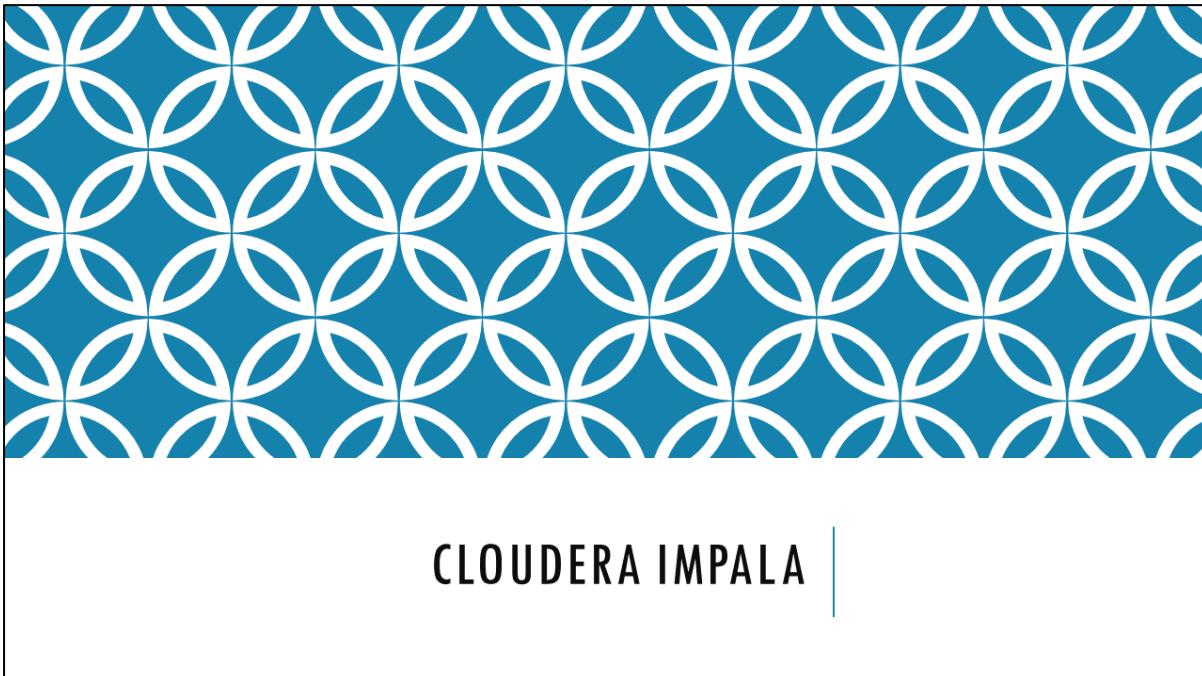
Oozie Workflow: These are sequential set of actions to be executed. You can assume it as a relay race. Where each athlete waits for the last one to complete his part.

Oozie Coordinator: These are the Oozie jobs which are triggered when the data is made available to it. Think of this as the response-stimuli system in our body. In the same manner as we respond to an external stimulus, an Oozie coordinator responds to the availability of data and it rests otherwise.

50. How do you configure an “Oozie” job in Hadoop?

“Oozie” is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs such as “Java MapReduce”, “Streaming MapReduce”, “Pig”, “Hive” and “Sqoop” .

Additional Reference



Impala is written in C++.

Impala uses Java to communicate with various Hadoop components.

Depends on Hive for Metastore.

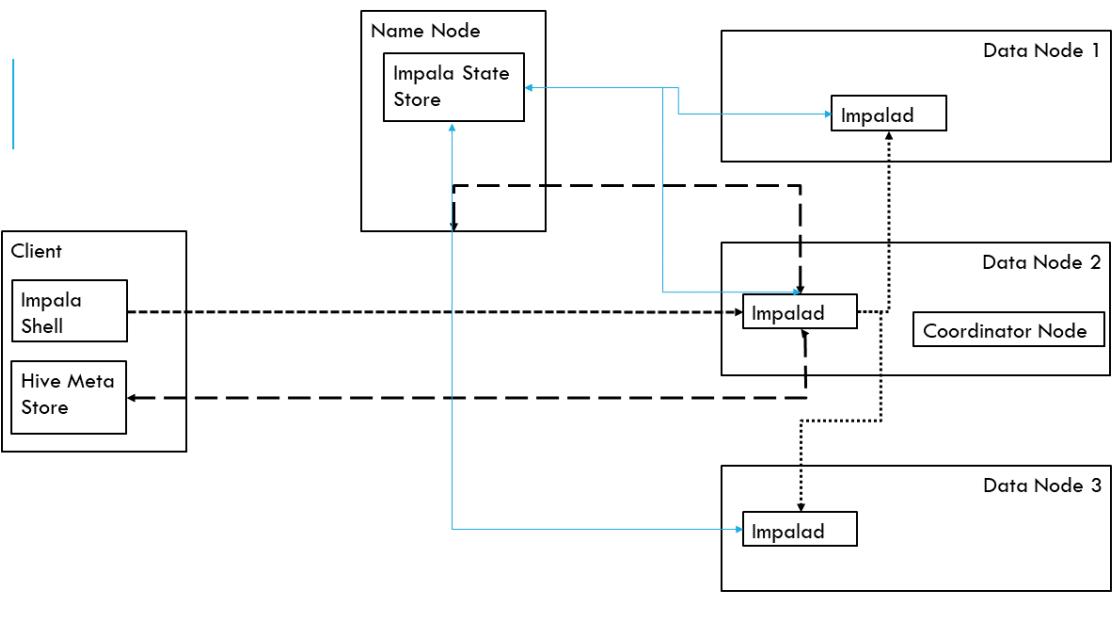
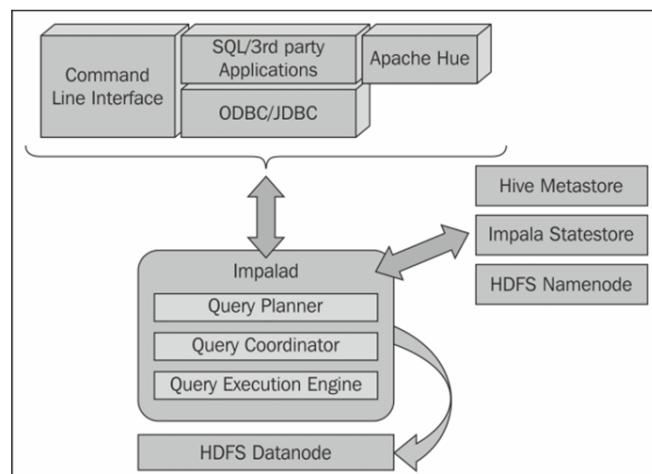
Runs directly on DataNodes where the data is present.

Provides a Massively Parallel Processing Execution Engine.

Provides HA

CORE COMPONENTS

Impala daemon
Impala statestore
Impala metadata and metastore



WHY FASTER

Impala does not write intermediate results on disk; instead full SQL processing is done in memory

The query starts its execution instantly compared to MapReduce

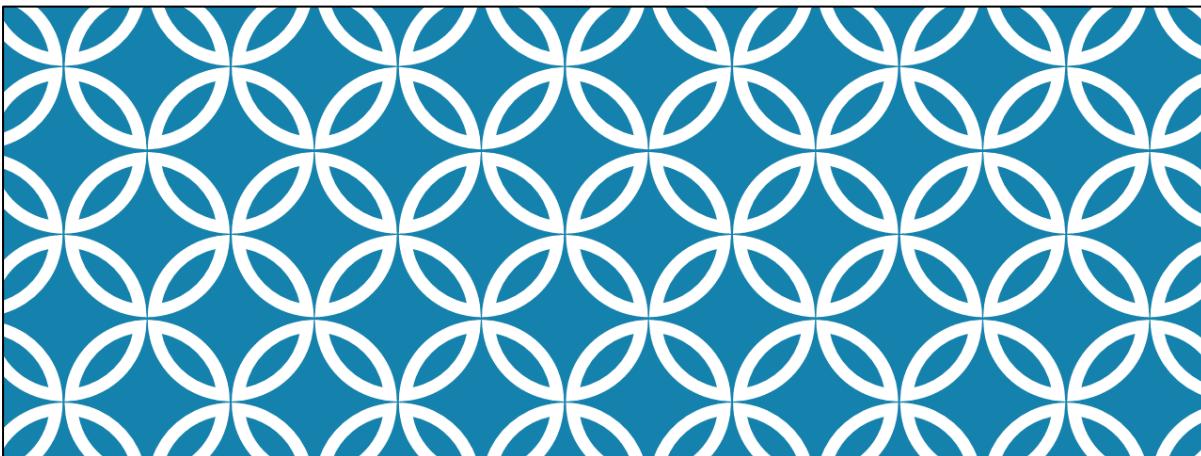
Impala Query Planner

Assembly-level code

Meta Data

IMPALA VS. HIVE

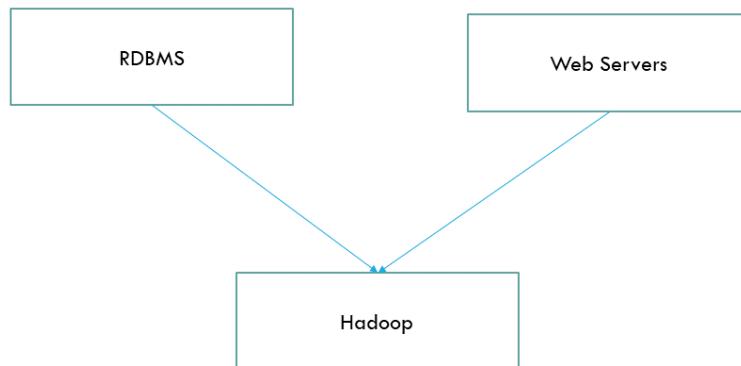
Impala	Hive
Performs in-memory Query Processing	Performs disk Query Processing
Uses own Query Engine	Uses MapReduce
NA	UDF, SerDe
All queries can run on hive	NA

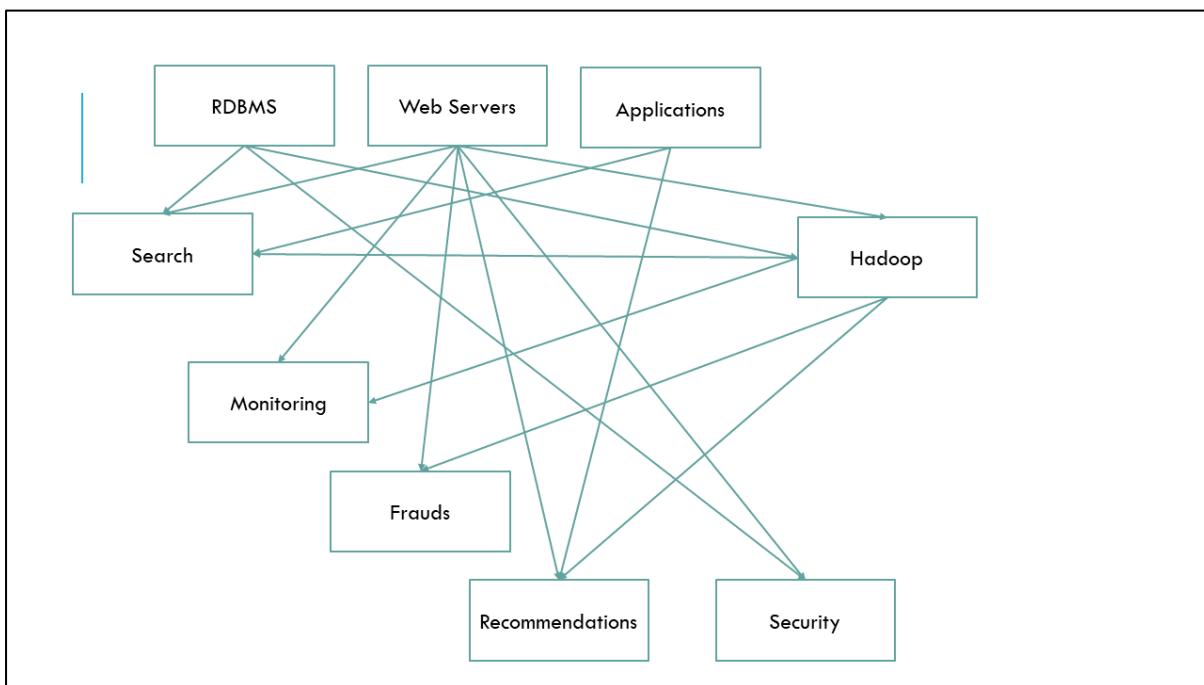


APACHE KAFKA

INTRODUCTION

It's a messaging Service





WORKING

Messages are organised into Topics

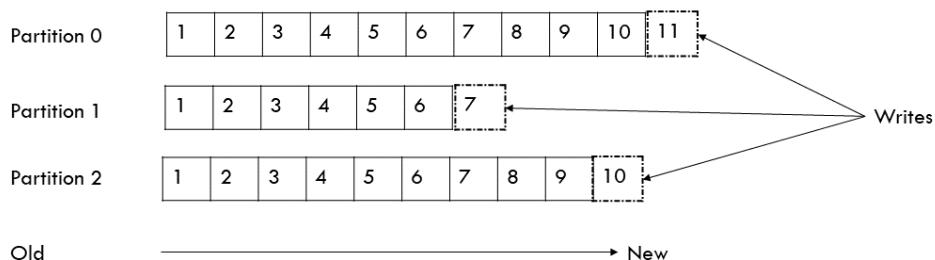
Producers push messages

Consumers pull messages

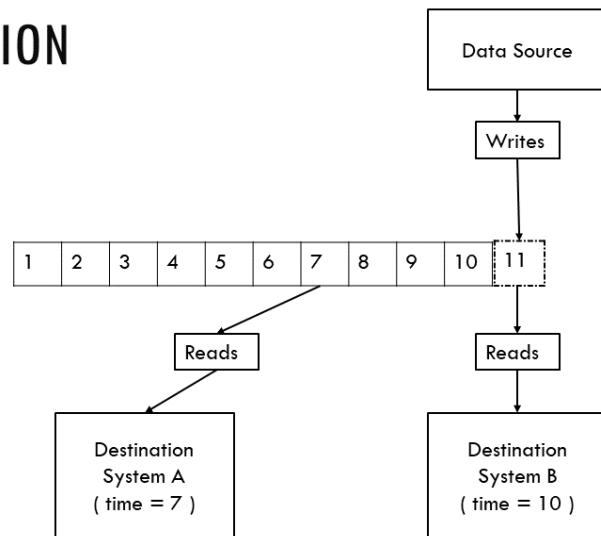
Kafka runs in Clusters

Nodes are called as Brokers

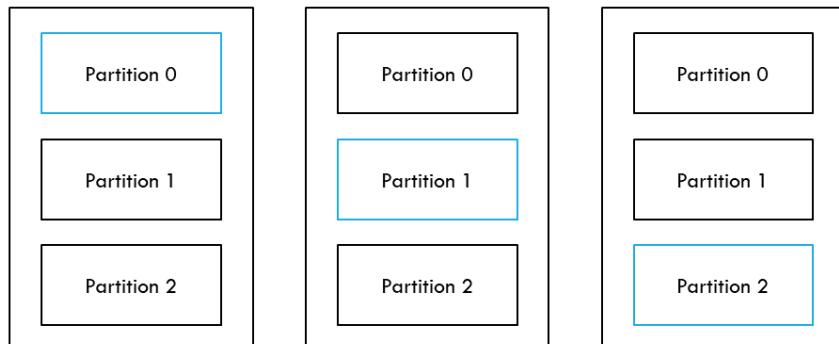
TOPIC



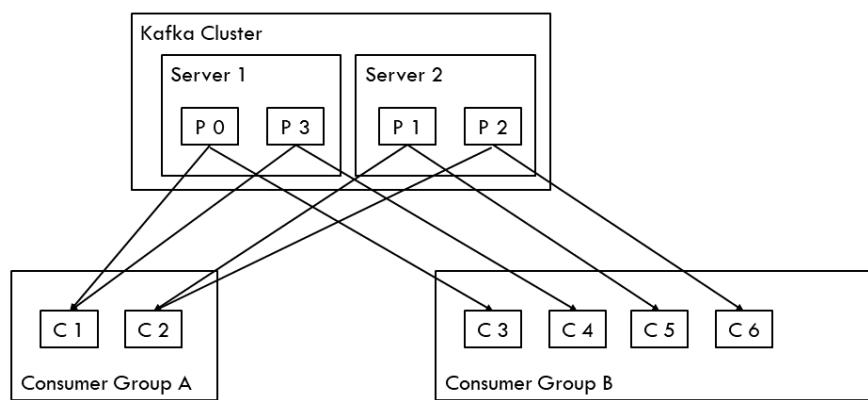
PARTITION



BROKER



CONSUMERS



APPLICATIONS

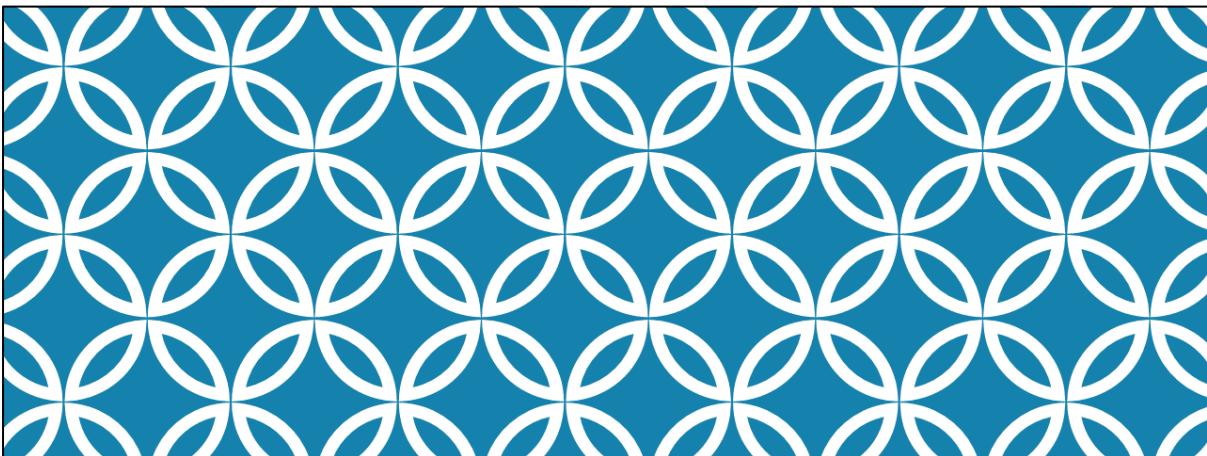
- Communications between Apps
- Website Activity Tracking
- Metrics Collections
- Audit

ADVANTAGES

- Large number of Consumers
- Ad-hoc Consumers
- Batch Consumers
- Automatic recover from Broker failures
- Data is maintained for a specific period of time

DISADVANTAGES

- Not an end to end User Solution
- Not many ready-made Consumers and Producers
- No Data Transformations
- No Encryption, Authorization or Authentication



APACHE MAHOUT

An Introduction

INTRODUCTION

Apache Mahout is an open source project that is primarily used in producing scalable machine learning algorithms.

A *mahout* is one who drives an elephant as its master.

1. Recommendation
2. Classification
3. Clustering

APPLICATIONS

Companies such as Adobe, Facebook, LinkedIn, Foursquare, Twitter, and Yahoo use Mahout internally.

Foursquare helps you in finding out places, food, and entertainment available in a particular area. It uses the recommender engine of Mahout.

Twitter uses Mahout for user interest modelling.

Yahoo! uses Mahout for pattern mining.

MACHINE LEARNING

Machine learning is a branch of science that deals with programming the systems in such a way that they automatically learn and improve with experience.

There are several ways to implement machine learning techniques, however the most commonly used ones are **supervised** and **unsupervised learning**.

SUPERVISED LEARNING

Supervised learning deals with learning a function from available training data.

1. Classifying e-mails as spam,
2. Labelling webpages based on their content

There are many supervised learning algorithms such as neural networks, Support Vector Machines (SVMs), and Naive Bayes classifiers.

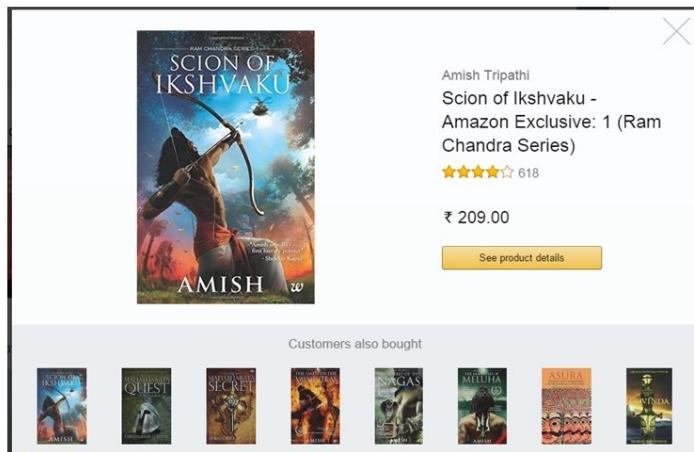
Mahout implements Naive Bayes classifier.

UNSUPERVISED LEARNING

Unsupervised learning makes sense of unlabelled data without having any predefined dataset for its training.

1. k-means
2. self-organizing maps, and
3. hierarchical clustering

RECOMMENDATION



CLASSIFICATION



CLUSTERING

Welcome to Apache™ Hadoop®!

<https://hadoop.apache.org/> ▾
 The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.
[Releases](#) · [Apache Hadoop 2.7.1](#) · [PoweredBy](#) · [Release 2.5.2](#)

Apache Hadoop - Wikipedia, the free encyclopedia

https://en.wikipedia.org/wiki/Apache_Hadoop ▾
 Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware.
[MapReduce](#) · [Big Data](#) · [Apache Hive](#) · [Pig \(programming tool\)](#)

What is Hadoop? - Hortonworks

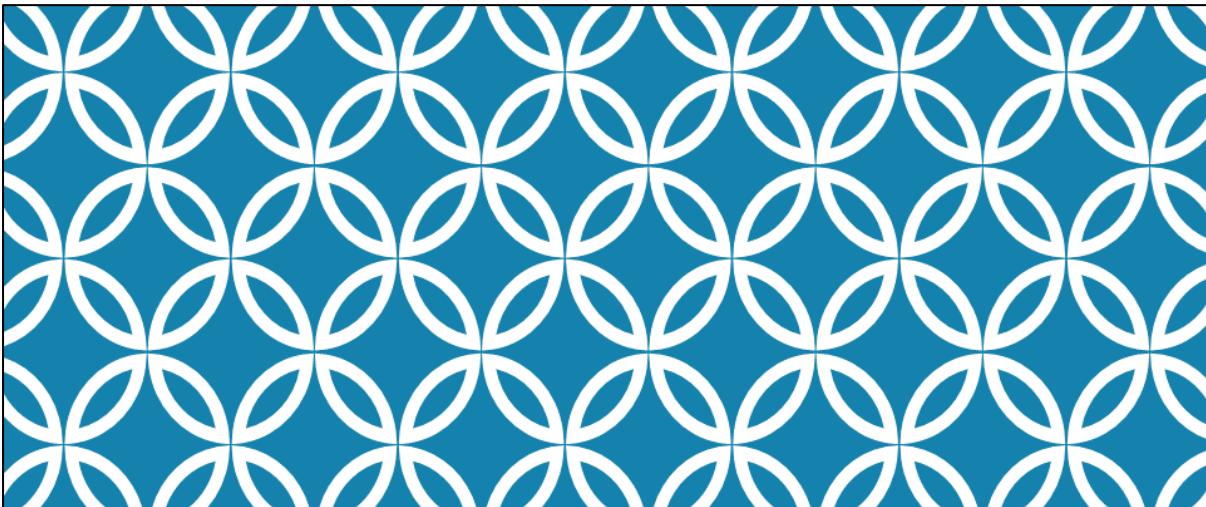
hortonworks.com/hadoop/ ▾
 Apache Hadoop is an open source project enabling you to store, process and gain insight from big data at low cost and huge scale.

Movies		
ID	Name	Category
001	Main Hun Na	Drama
002	Ragini MMS	Horror
003	Hungama	Comedy
004	Happy New Year	Drama
005	San Andreas	Thriller

Customers	
ID	Name
101	Ramesh
102	Suresh
103	Anjali
104	Ravi

Transactions		
Customer Id	Movie Id	Ratings
101	001	5
101	004	4.5
101	005	5
102	002	5
102	005	5
103	001	3
103	004	4
103	002	4
103	003	5
104	001	4
104	002	4
104	003	5
104	004	3
104	005	4

Item	Ramesh	Suresh	Anjali	Ravi
001	5		3	4
002		5	4	4
003			5	5
004	4.5			3
005	5	5		4



MANAGING AND SCHEDULING JOBS

DISPLAYING RUNNING JOBS

```
[training@localhost ~]$ hadoop job -list
1 jobs currently running

JobId  State  StartTime  UserName  Priority  SchedulingInfo
job_20110311158_0008  1  1320210148487  training  NORMALNA
```

DISPLAYING ALL JOBS

To do:

```
[training@localhost ~]$ hadoop job -list all
7 jobs submitted
States are:
Running : 1    Succeeded : 2    Failed : 3  Prep : 4
JobId      State          StartTime      UserName      Priority
SchedulingInfo
job_201110311158_0004 2        1320177624627 training NORMAL NA
job_201110311158_0005 2        1320177864702 training NORMAL NA
job_201110311158_0006 2        1320209627260 training NORMAL NA
job_201110311158_0007 2        1320210018614 training NORMAL NA
job_201110311158_0008 2        1320210148487 training NORMAL NA
job_201110311158_0001 2        1320097902546 training NORMAL NA
job_201110311158_0003 2        1320099376966 training NORMAL NA
```

DISPLAYING ALL JOBS (CONT'D)

Note that states are displayed as numeric values

- 1: Running
- 2: Succeeded
- 3: Failed
- 4: In preparation
- 5: (undocumented) Killed

Easy to write a cron job that periodically lists (for example) all failed jobs, running a command such as

- hadoop job -list all | grep '<tab>3<tab>'

DISPLAYING THE STATUS OF AN INDIVIDUAL JOB

`hadoop job -status<job_id>` provides status about an individual job

- Completion percentage
- Values of counters
- System counters and user-defined counters

Note: job name is not displayed!

- The Web user interface is the most convenient way to view more details about an individual job

KILLING A JOB

It is important to note that once a user has submitted a job, they cannot stop it just by hitting CTRL-C on their terminal

- This stops job output appearing on the user's console
- The job is still running on the cluster!

KILLING A JOB (CONT'D)

```
To [training@localhost~]$ hadoop job -list
      1 jobs currently running

      JobId  State  StartTime  UserName  Priority  SchedulingInfo
      Job_201110311158_0009  1 1320210791739  training  NORMALNA

[training@localhost~]$ hadoop job -kill job_201110311158_0009
Killed job job_201110311158_0009

[training@localhost~]$ hadoop job -list
      0 jobs currently running
      JobId  State  StartTime  UserName  Priority  SchedulingInfo
```



THE FIFO SCHEDULER

JOB SCHEDULING BASICS

A Hadoop job is composed of

- An unordered set of Map tasks which have locality preferences
- An unordered set of Reduce tasks

Tasks are scheduled by the JobTracker

- They are then by TaskTrackers
- One TaskTracker per node
- Each TaskTracker has a fixed number of slots for Map and Reduce tasks

JOB SCHEDULING BASICS (CONT'D)

- This may differ per node - a node with a powerful processor may have more slots than one with a slower CPU
- TaskTrackers report the availability of free task slots to the JobTracker on the Master node

Scheduling a job requires assigning Map and Reduce tasks to available Map and Reduce task slots

THE FIFO SCHEDULER

Default Hadoop job scheduler is FIFO

- First In, First Out

Given two jobs A and B, submitted in that order, all Map tasks from job A are scheduled before any Map tasks from job B are considered

- Similarly for Reduce tasks

Order of task execution within a job may be shuffled around

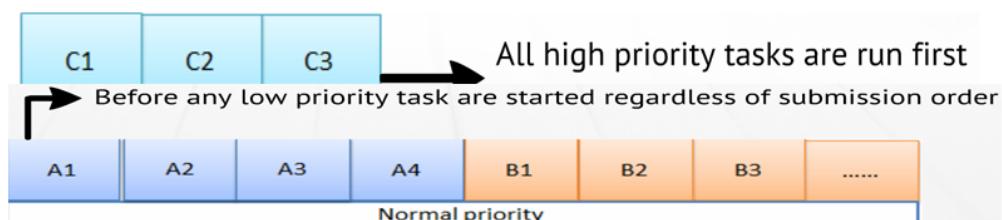


PRIORITIES IN THE FIFO SCHEDULER

The FIFO Scheduler supports assigning priorities to jobs

- Priorities are VERY _HIGH, HIGH, NORMAL, LOW, VERY _LOW
- Set with the mapred.job.priority property
- May be changed from the command-line as the job is running
- `hadoop job -set-priority <job_id> <priority>`
- All work in each queue is processed before moving on to the next

PRIORITIES IN THE FIFO SCHEDULER (CONT'D)



PRIORITIES IN THE FIFO SCHEDULER: PROBLEMS

Problem: Job A may have 2,000 tasks; Job B may have 20

- Job B will not make any progress until Job A has nearly finished
- Completion time should be proportional to job size

Users with poor understanding of the system may flag all their jobs as HIGH_PRIORITY

- Thus starving other jobs of processing time

'All or nothing' nature of the scheduler makes sharing a cluster between production jobs with SLAs and interactive users challenging



THE FAIR SCHEDULER

FAIR SCHEDULING

Base Scenario

- 3 guys x, y, z
- Hadoop cluster has 60 slots

FAIR SCHEDULING CONTD..

Scenario 1:

- All 3 get 20 slots.
- If somebody needs lesser number of slots then the remaining slots are equally distributed between the other 2 guys.
- If somebody needs more than 20 then he needs to wait for other 2 to release their slots.

FAIR SCHEDULING CONTD..

Scenario 2:

- X has minShare = 40 slots
- As long as he needs 40 slots he will get it and remaining slots will be distributed equally among other 2.
- Even if he needs more, he will still get 40 if the other 2 guys need the remaining slots.

FAIR SCHEDULING CONTD..

Scenario 3:

- X has minShare = 40 slots but needs only 10.
- He will get only 10 and the other 30 will become open for Fair Scheduling

FAIR SCHEDULING CONTD..

Scenario 4 :

- X has minShare = 80 slots and Y has minShare = 40 slots.
- The slots will be divided between X and Y on a ratio of 2:1.
- Z will not get any slot unless it is freed by the other 2 guys.

FAIR SCHEDULING CONTD..

Scenario 5:

- X has minShare = 10 Slots.
- If minShare < fairShare then we need to ignore minShare and the normal Fair Scheduling takes place.



CLUSTER MONITORING WITH GANGLIA

MONITORING CHALLENGES

System monitoring becomes a challenge when dealing with large numbers of systems

Multiple solutions exist, such as

- Nagios
- Hyperic
- Zabbix

Many of these are very 'General purpose'

- Fine for monitoring the machines themselves
- Not so useful for integrating with Hadoop

MONITORING CLUSTER METRICS WITH GANGLIA

Ganglia is an open-source, scalable, distributed monitoring product for high-performance computing systems

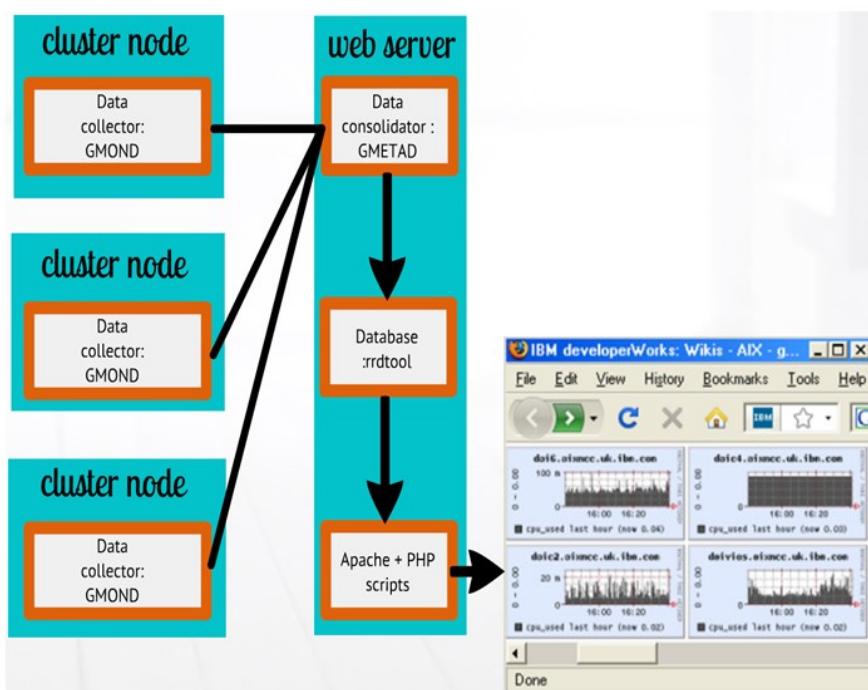
- Specifically designed for clusters of machines

Collects, aggregates, and provides times-series views of metrics

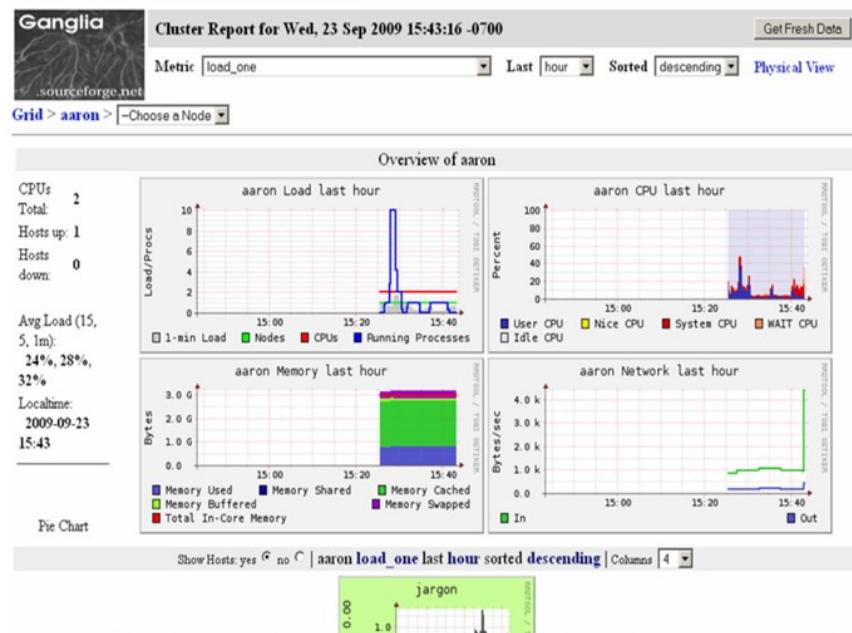
Integrates with Hadoop's metrics-collection system

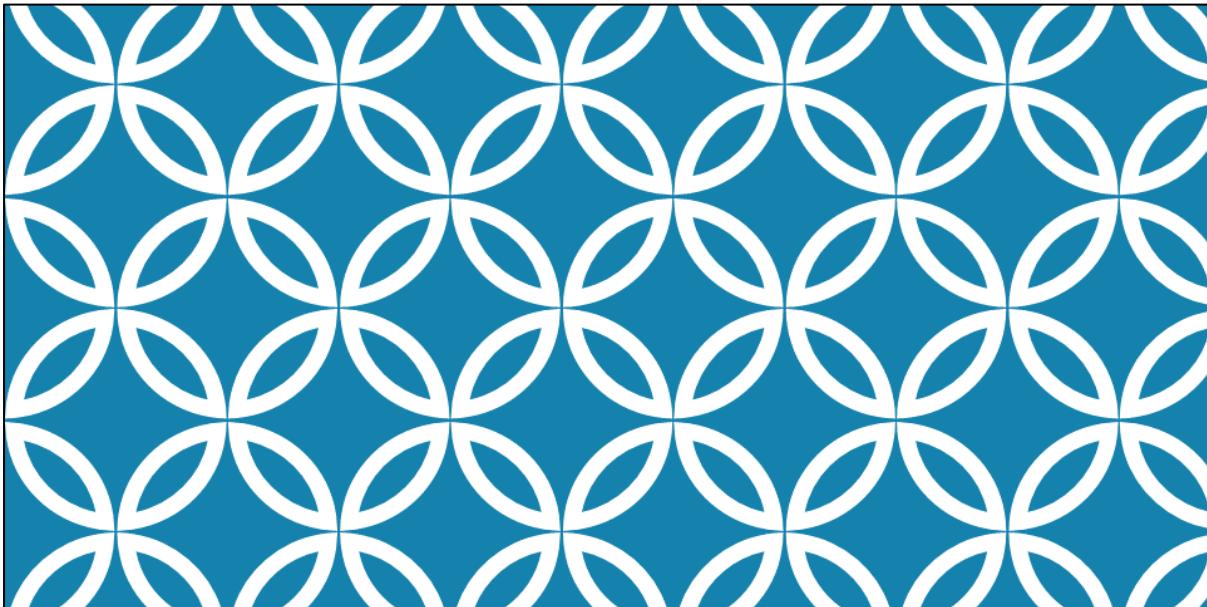
Note: Ganglia doesn't provide alerts

GANGLIA NETWORK ARCHITECTURE

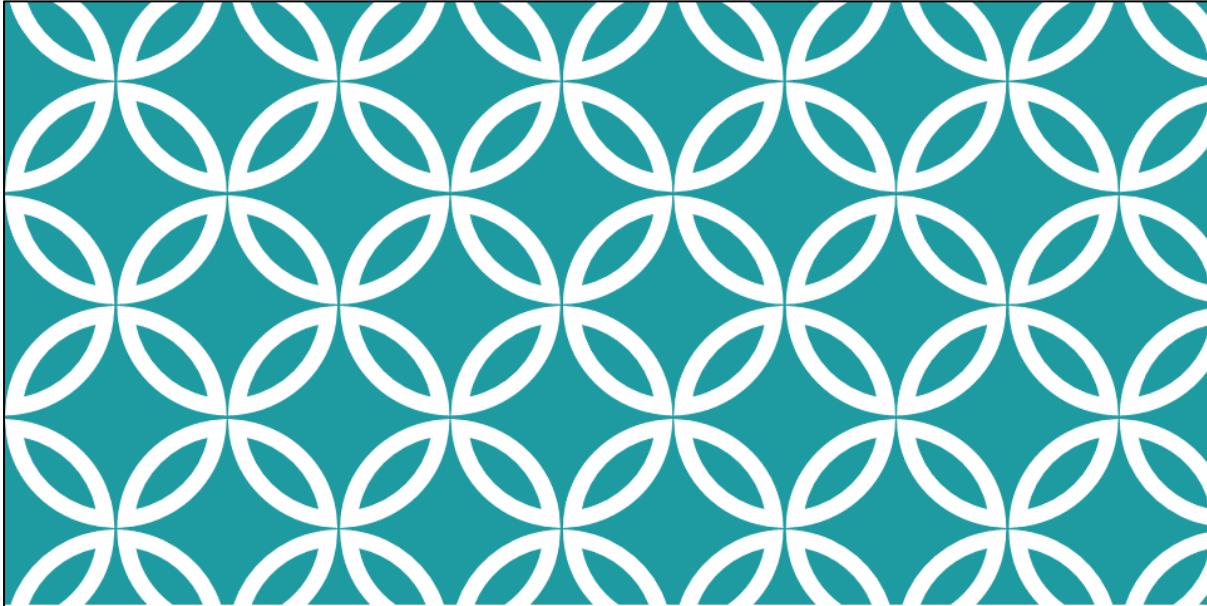


EXAMPLE GANGLIA WEB APP OUTPUT





CLUSTER MAINTENANCE



CHECKING HDFS STATUS

CHECKING FOR CORRUPTION IN HDFS

fsck checks for missing or corrupt data blocks

Unlike system fsck, does not attempt to repair errors

Can be configured to list all files

Also all blocks for each file, all block locations, all racks

Examples:

- `hadoop fsck /`
- `hadoop fsck / -files`
- `hadoop fsck / -files -blocks`
- `hadoop fsck / -files -blocks -locations`
- `hadoop fsck / -files -blocks -locations -racks`

CHECKING FOR CORRUPTION IN HDFS (CONT'D)

Good idea to run fsck as a regular cron job that e-mails the results to administrators

Choose a low-usage time to run the check

`move` option moves corrupted files to `/lost+found`

A corrupted file is one where all replicas of a block are missing

`delete` option deletes corrupted files

USING DFSADMIN

dfsadmin provides a number of administrative features including:

List info: `$ hadoop dfsadmin -report`

Re-report: `$ hadoop dfsadmin -refreshNodes`

USING DFSADMIN (CONT'D)

Manually set the filesystem to 'safe mode'

NameNode starts up in safe mode

Read-only - no changes can be made to the metadata

Does not replicate or delete blocks

Leaves safe mode when the (configured) minimum percentage of blocks are replicated

`$ hadoop dfsadmin - safemode enter`

`$ hadoop dfsadmin - safemode leave`

USING DFSADMIN (CONT'D)

Can also block until safemode is exited

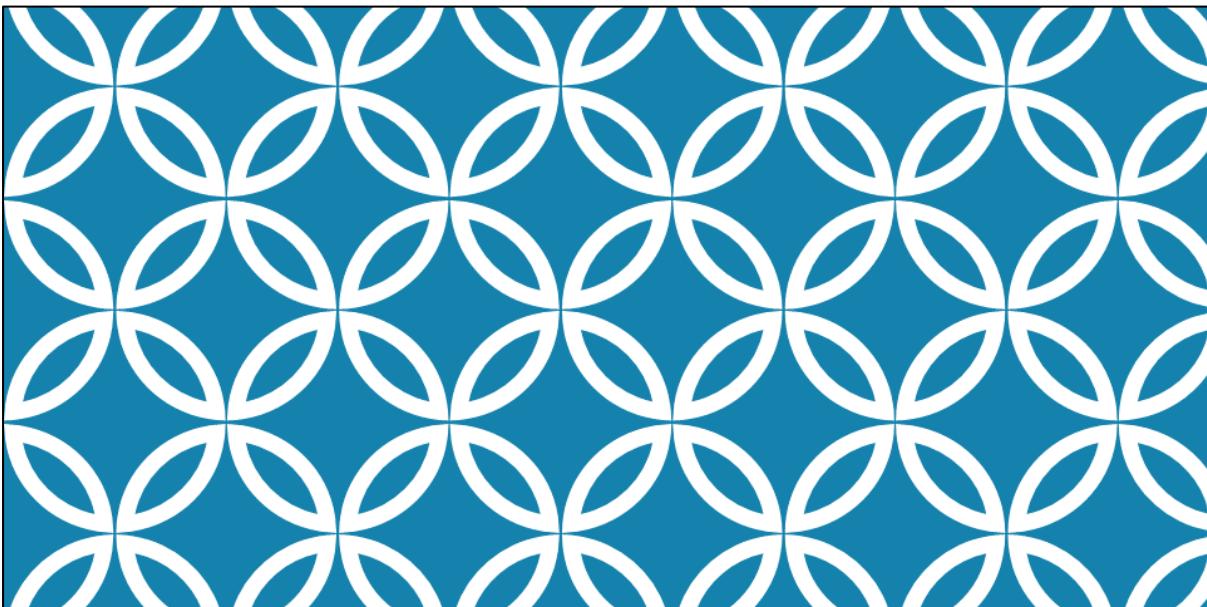
Useful for shell scripts

`hadoop dfsadmin -safemode wait`

Saves the NameNode metadata to disk and resets the edit log

N

```
$ hadoop dfsadmin - saveNamespace
```



TOOL RUNNER

WHY USE TOOLRUNNER ?

ToolRunner uses the GenericOptionsParser class internally

- Allows you to specify configuration options on the command line
- Also allows you to specify items for the Distributed Cache on the command line (see later)

