

Spring 2020

## Business insights of user usage records of data cards

Rajsanjyot Kantipudi

Follow this and additional works at: <https://lib.dr.iastate.edu/creativecomponents>



Part of the [Data Storage Systems Commons](#)

### Recommended Citation

Kantipudi, Rajsanjyot, "Business insights of user usage records of data cards" (2020). *Creative Components*. 516.

<https://lib.dr.iastate.edu/creativecomponents/516>

This Creative Component is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Creative Components by an authorized administrator of Iowa State University Digital Repository. For more information, please contact [digirep@iastate.edu](mailto:digirep@iastate.edu).

Business insights of user usage records of data cards

By

Rajsanjyot Kantipudi

Department of Management Information Systems

Iowa State University

A Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Iowa state university

Spring 2020

Committee Member:

1) Dr. Anthony M. Townsend

Associate professor of Information systems

## Business insights of user usage records of data cards

### Abstract

The aim of this project is finding the business insights of current user records data and get the benefits for business growth. The parameters to be considered for analysis are

1. Daily user count and bytes transmitted for a time slot.
2. Area wise business(usage) share in the total business
3. Every network owner will be depending on partners to get the service where they do not have the service tower.

From point1:

We can find the exact time when a greater number of users are using the network and at what time maximum number of downloads and uploads are happening. Based on that, they can concentrate tower capacity enhancements. If the tower is underutilized, they can reduce the tower capacity.

From point2:

They can concentrate on the areas where they can invest more to get maximum number of users.

From point3:

Find out the areas of partner leading and try to improve the installations of the owner tower.

All the above activities are currently happening by using data warehousing technologies. But they are more expensive and time consuming. To help better in this area, we will be using the Hadoop and Hadoop Echo systems.

## TABLE OF CONTENTS

1.	INTRODUCTION .....	6
	Data processing with Hive .....	7
	Apache Sqoop.....	8
1.1.	PURPOSE.....	9
2.	LITERATURE SURVEY .....	10
2.1	EXISTING SYSTEM.....	11
2.2	FEASIBILITY STUDY .....	12
2.3.1.	ECONOMICAL FEASIBILITY .....	12
2.3.2.	TECHNICAL FEASIBILITY .....	12
2.3.3.	SOCIAL FEASIBILITY .....	12
2.3	SOFTWARE AND HARDWARE .....	13
2.4.1.	HARDWARE REQUIREMENT FOR EACH NODE/MACHINE IN A CLUSTER .....	13
3.	SYSTEM ANALYSIS AND DESIGN.....	14
3.1.	FUNCTIONAL REQUIREMENTS: .....	14
	Output Design .....	14
	Output Definition .....	14
	Output Media: .....	15
	Input Design: .....	15
	Objectives.....	16
	Input Stages:.....	16
	Input Types:.....	16
	Input Media:.....	17
	Error Avoidance .....	17
	Error Detection.....	17
	Data Validation.....	17
	User Interface Design .....	18
	User interface systems can be broadly classified as: .....	18
	User Initiated Interfaces .....	18
	Computer-Initiated Interfaces.....	18
	Error message design: .....	19
3.2.	PERFORMANCE REQUIREMENTS .....	19
3.3.	MODULES.....	19
	Module Description.....	20
	Expecting Results from the reporting tool .....	21
3.4.	PROPOSED ARCHITECTURE:.....	22

4. SYSTEM IMPLEMENTATION .....	24
4.1    TECHNOLOGIES USED: .....	24
What Is Big Data? .....	24
Capturing and managing lots of information:.....	25
Working with many new types of data: .....	25
Exploiting these masses of information and new data types with new styles of applications:.....	25
Differentiating between Big Data and traditional enterprise relational data.....	26
What is Hadoop .....	27
Hdfs assumptions and goals:.....	27
HDFS Architecture Design .....	28
HDFS Files .....	29
Block Allocation .....	30
MapReduce .....	31
Inputs and Outputs.....	32
Apache Hive.....	33
Why Hive .....	33
Hive Internal Working: .....	33
Tableau.....	35
Key differentiators.....	35
4.2.    IMPLENTATION .....	36
4.2.1.    SINGLE NODE SETUP .....	36
Environment Setup.....	36
Pre-requisites .....	36
Install Java 1.6+ .....	36
Add a dedicated Hadoop user account .....	39
Configuring SSH Access .....	39
Disable IPv6 .....	41
Hadoop Installation .....	42
Hadoop Configuration .....	43
Update \$Home/.bashrc file .....	43
Configure hadoop-env.sh.....	44
4.2.2.    MULTI NODE CLUSTER SETUP .....	51
ii. Enabling SSH: .....	52
Configurations: .....	53
a. masters: .....	53

b. slaves: .....	54
Configuring all *-site.xml files: .....	54
c. core-site.xml: .....	55
d. hdfs-site.xml: .....	55
e. mapred-site.xml:.....	56
Formatting and Starting/Stopping the HDFS filesystem via the NameNode: .....	56
Starting the multi-node cluster: .....	57
a. To start HDFS daemons: .....	57
b. To start Map Red daemons: .....	58
4.2.3.    HIVE CONFIGURATION.....	60
4.2.4.    SQOOP CONFIGURATION.....	63
\$sqoop version .....	64
4.2.5.1.    Import table from MySql to HDFS .....	64
4.2.5.2.    Analysing the data with Hive .....	66
Scripts to Run .....	67
4.5.2.3.Export the results Back to MySql.....	70
4.2.5.4.    Generating reports .....	71
Generating the reports and View.....	73
Steps to be followed for generating each report.....	73
Output .....	76
5. SYSTEM TESTING .....	77
5.1.    TYPES OF TESTING .....	77
5.1.1.    UNIT TESTING.....	77
5.1.2.    BLACK BOX TESTING .....	78
5.1.3.    WHITE BOX TESTING.....	78
5.1.4.    TEST STRATEGY AND APPROACH.....	78
5.2.    TEST OBJECTIVES.....	78
5.3.    FEATURES TO BE TESTED .....	79
5.3.1.    INTEGRATION TESTING .....	79
5.3.2.    FUNCTIONAL TESTING .....	79
5.3.3.    SYSTEM TESTING.....	80
5.4.    TESTS CONDUCTED .....	80
Test Results .....	80
6. CONCLUSION .....	81
7. REFERENCES/BIBLIOGRAPHY.....	82

## 1. INTRODUCTION

There is a lot of buzz around “big data” and rightly so. Organizations that are capturing and analyzing large amounts of data in real time or near-real time are creating significant competitive advantages for themselves, their customers and business partners. Communications service providers (CSPs) are no exception. CSPs that can ingest and analyze network, location and customer data in real time or near-real time have much to gain. They will be able to quickly introduce new capabilities such as location-based services, intelligent marketing campaigns, next best actions for sales and service, social media insights, network intelligence and fraud detection to significantly increase revenues and reduce costs.

We are living in an information age and there is enormous amount of data that is flowing between systems, internet, telephones, and other media. The data is being collected and stored at unprecedented rates. There is a great challenge not only to store and manage the large volume of data, but also to analyze and extract meaningful information from it. There are several approaches to collecting, storing, processing, and analyzing big data. The focus of the paper is to draw an analogy for data management between the traditional relational database systems and the Big Data technologies.

Data creation is occurring at an unprecedented rate. In 2014, the world generated over 7ZB of data; and by 2018, we have generated 11ZB of data. IBM estimates that every day, 2.5 quintillion bytes of data is created – so much that 90% of the data in the world today has been created in the last two years. Increasingly large numbers of embedded sensors, smartphones, PCs, and tablet computers connected to network are generating enormous amounts of data. This data creates new opportunities to "extract more value" for the areas that it is needed. We have entered the age of "Big Data." Just as this data is generated by people in real time, it can be analyzed in real time by high performance computing networks, thus creating a potential for improved decision-making. The International Data Corporation (IDC) believes organizations that are best able to make real-time business decisions using Big Data solutions will thrive, while those that are unable to embrace and make use of this shift will increasingly find themselves at a competitive disadvantage in the market and face

potential failure.

**Big Data** is an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process using on-hand data management tools or traditional data processing applications. The challenges include capture, curation, storage, search, sharing, transfer, analysis and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, prevent diseases, combat crime and so on.

**Hadoop** is a free, open source Java-based framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation. Hadoop makes it possible to run applications on systems with thousands of nodes involving thousands of terabytes. Its distributed file system facilitates rapid data transfer rates among nodes and allows the system to continue operating uninterrupted in case of a node failure. This approach lowers the risk of catastrophic system failure, even if a significant number of nodes become inoperative. The Hadoop framework is used by major players including Google, Yahoo and IBM, largely for applications involving search engines and advertising. The preferred operating systems are anyflavor of Linux, windows (here we need to use Cygwin) but Hadoop can also work with BSD and OS X.

### **Data processing with Hive**

Hive is a Data Warehouse software that facilitates querying and managing huge data residing in distributed storage .Instead of writing huge raw map reduce programs in some programming language, Hive provides a SQL-like interface to data stored in Hadoop File System. And there is another popular Hadoop eco-system i.e Pig which is a scripting language with a focus on data flows. Hive provides a database query interface to Apache Hadoop. People often ask why Pig and Hive exist when they seem to do much of the same thing. Hive because of its SQL like query language is often used as the interface to an Apache Hadoop based data warehouse. Hive is considered friendlier and more familiar to users who are used to using SQL for querying data. Pig fits in through its data flow strengths where it takes on the tasks of bringing data into Apache Hadoop and working with it to get it into the form for querying. A good overview of how this works is in Alan Gates posting on the Yahoo Developer



blog titled Pig and Hive at Yahoo! From a technical point of view both Pig and Hive are feature complete so you can do tasks in either tool. However, you will find one tool or the other will be preferred by the different groups that have to use Apache Hadoop. The good part is they have a choice and both tools work together.

### **Apache Sqoop**

Sqoop (SQL-to-Hadoop) is a big data tool that offers the capability to extract data from non-Hadoop data stores, transform the data into a form usable by Hadoop, and then load the data into HDFS. This process is called ETL, for Extract, Transform, and Load. While getting data into Hadoop is critical for processing using MapReduce, it is also critical to get data out of Hadoop and into an external data source for use in other kinds of application. While it is sometimes necessary to move the data in real time, it is most often necessary to load or unload data in bulk. Like Pig, Sqoop is a command-line interpreter. You type Sqoop commands into the interpreter and they are executed one at a time.

Four key features are found in Sqoop:

**Bulk import:** Sqoop can import individual tables or entire databases into HDFS. The data is stored in the native directories and files in the HDFS file system.

**Direct input:** Sqoop can import and map SQL (relational) databases directly into Hive and HBase.

**Data interaction:** Sqoop can generate Java classes so that you can interact with the data programmatically.

**Data export:** Sqoop can export data directly from HDFS into a relational database using a target table definition based on the specifics of the target database.

Sqoop works by looking at the database you want to import and selecting an appropriate import function for the source data. After it recognizes the input, it then reads the metadata for the table (or database) and creates a class definition of your input requirements.

Sqoop can be to very selective so that you get just the columns you are looking for before input rather than doing an entire input and then looking for your data. This can save considerable time. The actual import from the external database to HDFS is performed by a Map Reduce job created behind the scenes by Sqoop.

## 1.1. PURPOSE

In This project we can find the exact what time more users using the network and what time more downloads and uploads happening. Based on that, they can concentrate tower capacity enhancements. If the tower is underutilized then they can reduce the tower capacity. They can concentrate the area where they can invest more to get the more users. Find out the areas of partner leading and try to improve the owner tower installations.

## 2. LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the below considerations and opinions are considered for developing the proposed system.

The process of the research into complex data basically concerned with the revealing of hidden patterns. **Sagiroglu, S.; Sinanc, D. (20-24 May 2013), "Big Data: A Review"** describe the big data content, its scope, methods, samples, advantages, and challenges of Data. The critical issue about the Big data is the privacy and security. Big data samples describe the review about the atmosphere, biological science, and research. Life sciences etc. From this paper, we can conclude that any organization in any industry having big data can take the benefit from its careful analysis for the problem solving purpose. Using Knowledge Discovery from the Big data easy to get the information from the complicated data sets. The overall Evaluation describe that the data is increasing and becoming complex. The challenge is not only to collect and manage the data also how to extract the useful information from that collected data. According to the Intel IT Center, there are many challenges related to Big Data which are data growth, data infrastructure, data variety, data visualization, data velocity.

**Garlasu, D.; Sandulescu, V. ; Halcu, I. ; Neculoiu, G. ;( 17-19 Jan. 2013), "A Big Data implementation based on Grid Computing", Grid Computing** offered the advantage about the storage capabilities and the processing power and the Hadoop technology is used for the implementation purpose. Grid Computing provides the concept of distributed computing. The benefit of Grid computing center is the high storage capability and the high processing power. Grid Computing makes the big contributions among the scientific research, help the scientists to analyze and store the large and complex data.

**Mukherjee, A.; Datta, J.; Jorapur, R.; Singhvi, R.; Haloi, S.; Akram, W. (18- 22 Dec. 2012) “Shared disk big data analytics with Apache Hadoop”** Big data analytics define the analysis of large amount of data to get the useful information and uncover the hidden patterns. Big data analytics refers to the Mapreduce Framework which is developed by the Google. Apache Hadoop is the open source platform which is used for the purpose of implementation of Google’s Mapreduce Model [2]. In this the performance of SF-CFS is compared with the HDFS using the SWIM by the facebook job traces .SWIM contains the workloads of thousands of jobs with complex data arrival and computation patterns.

**Aditya B. Patel, Manashvi Birla, Ushma Nair (6-8 Dec. 2012) “Addressing Big Data Problem Using Hadoop and Map Reduce”** reports the experimental work on the Big data problems. It describe the optimal solutions using Hadoop cluster, Hadoop Distributed File System (HDFS) for storage and Map Reduce programming framework for parallel processing to process large data sets.

**Real Time Literature Review about the Big Data According** to 2013, Facebook has 1.11 billion people active accounts from which 751 million using Facebook from a mobile. Another example is flicker having feature of Unlimited photo uploads (50MB per photo), Unlimited video uploads (90 seconds max, 500MB per video), the ability to show HD Video, Unlimited storage, Unlimited bandwidth. Flickr had a total of 87 million registered members and more than 3.5 million new images uploaded daily.

## **2.1 EXISTING SYSTEM**

Telecom service providers are dealing with huge amounts of data cards usage records every day. There is a great challenge not only to store and manage such a large amount of data, but also to analyze and extract meaningful information from it and getting the benefit out of that analysis. There are several approaches to collecting, storing, processing, and analyzing big data .Currently, these analysis activities are happening using data warehousing technologies. But it is more expensive and time consuming. To help better in this area, we are using the Hadoop and Hadoop Eco-systems.

## **2.2 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

### **2.3.1. ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the Developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **2.3.2. TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### **2.3.3. SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## **2.3 SOFTWARE AND HARDWARE**

### **2.4.1. HARDWARE REQUIREMENT FOR EACH NODE/MACHINE IN A CLUSTER**

Processor	: Intel
Speed	: 2.5 GHz
RAM	: 8 GB or More
Hard Disk	: 500 GB or More

### **2.4.2 .SOFTWARE REQUIREMENT Operating**

System	: Linux (CENT-
OS) Technology	: Hadoop
Tools	: Hive,Sqoop
Reporting Tool	: Tableau
Database	: My SQL
Java Version	: JDK1.6 or Higher version

### 3. SYSTEM ANALYSIS AND DESIGN

#### 3.1. FUNCTIONAL REQUIREMENTS:

##### Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system, results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.
  - The output form of an information system should accomplish one or more of the following objectives.
  - Convey information about past activities, current status or projections of the Future.
  - Signal important events, opportunities, problems, or warnings.
  - Trigger an action.
  - Confirm an action.

##### Output Definition

The outputs should be defined in terms of the following points:

- Type of the output.
- Content of the output.
- Format of the output.
- Location of the output.
- Frequency of the output.
- Volume of the output.
- Sequence of the output.

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

For Example

- Will decimal points need to be inserted
- Should leading zeros be suppressed.

#### **Output Media:**

In the next stage it is to be decided that which medium is the most appropriate for the output. The main considerations when deciding about the output media are:

- The suitability for the device to the application.
- The need for a hard copy.
- The response time required.
- The location of the users
- The software and hardware available.

Keeping in view the above description the project is to have outputs mainly coming under the category of internal outputs. The main outputs desired according to the requirement specification are:

The outputs were needed to be generated as a hot copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

#### **Input Design:**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?



- The dialog to guide the operating personnel in providing input. Methods for preparing input validations and steps to follow when error occur.

### **Objectives**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follows

### **Input Stages:**

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

### **Input Types:**

- It is necessary to determine the various types of inputs. Inputs can be categorized as follows:
- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?

- Interactive, which are inputs entered during a dialogue.

### **Input Media:**

At this stage, choice must be made about the input media. To conclude about the input media consideration must be given to.

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive.

Input data is to be the directly keyed in by the user, the keyboard can be the most suitable input device.

### **Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded up to the stage in which the data is accepted by the system.

This can be achieved only by means of careful control each time the data is handled.

### **Error Detection**

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

### **Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with pop up menus.

### **User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

#### **User interface systems can be broadly classified as:**

1. User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
2. Computer initiated interfaces.

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer acts or displays further information.

### **User Initiated Interfaces**

#### **User initiated interfaces fall into two approximate classes:**

1. Command driven interfaces: In this type of interface the user inputs commands or queries, which are interpreted by the computer.
2. Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

### **Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

1. The menu system for the user is presented with a list of alternatives and the user chooses one, of alternatives.

2. Questions – answer type dialog system where the computer asks question and takes based on the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

#### **Error message design:**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

### **3.2. PERFORMANCE REQUIREMENTS**

Performance is measured in terms of the output provided by the application.

Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

### **3.3. MODULES**

In our project, we have the following modules

- Importing the data from RDBMS to HDFS
- Analyzing the data

- Exporting the analysis results back to RDBMS
- Creating the reports

### Module Description

**Importing the data from RDBMS to HDFS :** If you have a system where we are producing huge volume of data, and we want to extract some part of the data and do some analysis and wish to view the results of such analysis quite often. We cannot use Hadoop system to have quick views of our data. So, a possible workflow for such a problem can be

Capture data into HDFS -> Analyze data with Map Reduce- >HDFS to RDBMS.

The benefit of using this kind of workflow is based on simple fact that Hadoop systems are not for quick reads whereas an RDBMS system can effectively resolve the problem of quick reads. We are using the MapReduce paradigm to turn our raw data to substantial information (may be usually at the end of day using some batch process) and finally pushing it into an RDBMS system for display purpose.

Here we are using a big data tool that offers the capability to extract data from non-Hadoop data stores, transform the data into a form usable by Hadoop, and then load the data into HDFS. This process is called ETL, for Extract, Transform, and Load. While getting data into Hadoop is critical for processing using MapReduce, it is also critical to get data out of Hadoop and into an external data source for use in other kinds of application. Sqoop can do this as well. While it is sometimes necessary to move the data in real time, it is most often necessary to load or unload data in bulk.

Analyzing the data: As the size of data sets being collected and analyzed in the telecom industry for business intelligence is growing rapidly, making traditional warehousing solutions prohibitively expensive. Hadoop is a popular open-source map-reduce implementation, which is being used as an alternative to store and process extremely large data sets on commodity hardware. However, the map-reduce programming model is very low level and requires developers to write custom programs which are hard to maintain and reuse. In this project, we are using Hive, an open-source data warehousing solution built on top of Hadoop as shown in Fig.1. Hive supports queries expressed in a SQL-like declarative language - HiveQL, which are compiled into map-reduce jobs executed on Hadoop. In addition, HiveQL supports custom map-reduce scripts to be plugged into queries.

### **Exporting the Analysis results back to RDBS:**

After analyzing the data set we will get the following things as the result.

4. Daily user count and bytes transmitted on a particular time slot.
5. Area wise business (usage) share in the total business
6. Since every network owner will be depending on partners to get the service where they does not have the service tower.

to be the most common export needs are to Excel, to a flat file, and to PDF, and a good report tool must able to export to all three formats here We will be exporting the results back to our rdbms to get connected with reporting tools and generate the reports on that data.

### **Generating the Reports:**

Data is useless if all it does is sitting in the data warehouse. As a result, the presentation layer is of very high importance. Most of the OLAP vendors already have a front-end presentation layer that allows users to call up pre-defined reports or create ad hoc reports. Reporting tools are widely used to create such reports to support decision making and measure performance. Companies use them for financial consolidation, for evaluation of strategies and policies and often just for plain reporting. In our project we are going to use tableau as the reporting tool which will produce the reports in the required format (i.e., graphs, charts etc...).

### **Expecting Results from the reporting tool**

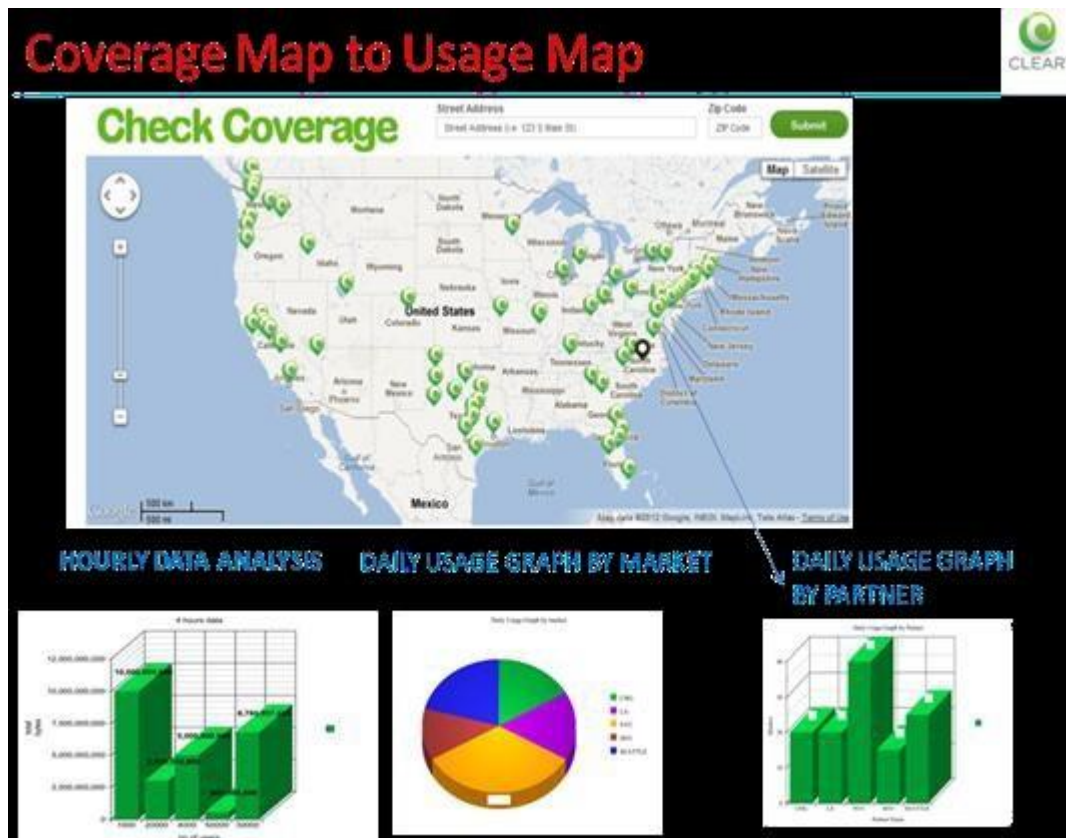


Fig.3.1: Expecting Output Reports

### 3.4. PROPOSED ARCHITECTURE:

After analyzing all the requirements, I have designed and going to implement the following architecture. As we see in the following figure, first we are going to load the user's data card usage records data (.csv files) from MySQL RDBMS into Hadoop HDFS and then that data we are going to process with some other bigdata technology called Hive and after that we will be exporting the results back to our MySQL RDBMS and generating the reports on that.

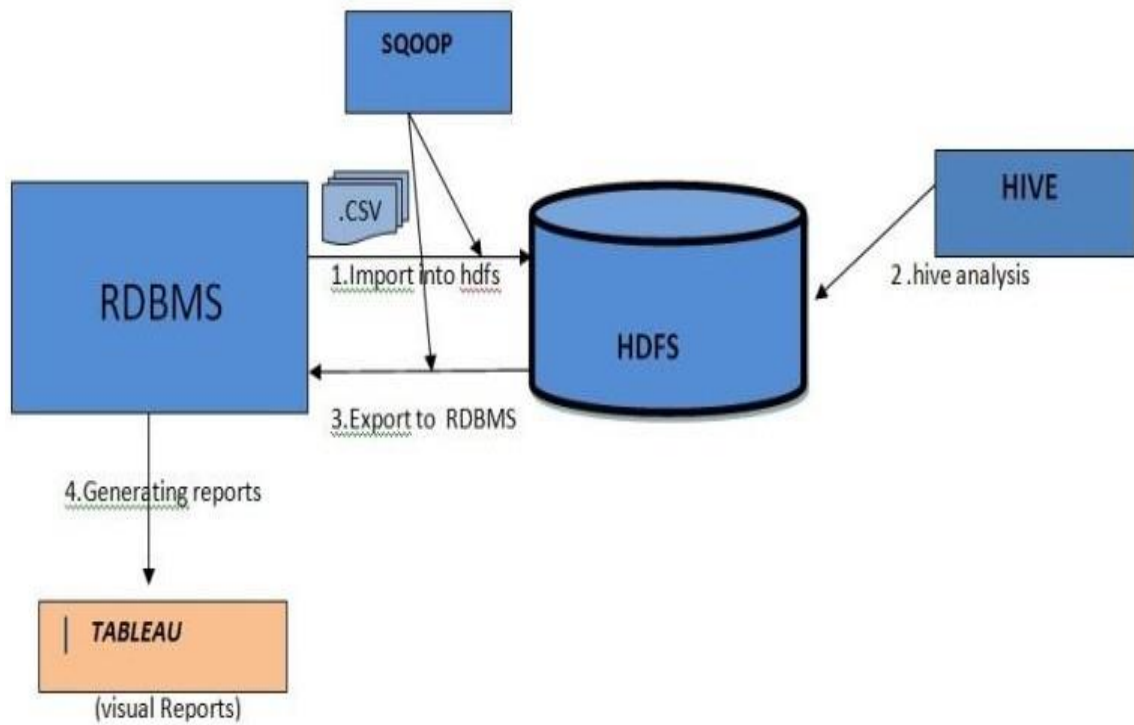


Fig.3.2: Proposed Architecture



## 4. SYSTEM IMPLEMENTATION

### 4.1 TECHNOLOGIES USED:

The following are the technologies, which we are going to use in our actual implementation of our project

1. MySql
2. Apache Hadoop
3. Apache Hive
4. Apache Sqoop
5. Tableau

#### Technologies Description

##### MySQL:

MySQL is a relational database system. MySQL is faster, more reliable, and cheaper -- or, simply put, better -- than any other database system (including commercial systems such as Oracle and DB2). Many MySQL opponents continue to challenge this viewpoint, going even so far as to assert that MySQL is not even a relational database system.

We can safely say that there is a large bandwidth of opinion.

- The fact is that there are an ever-increasing number of MySQL users, and most of them are quite satisfied with MySQL. Thus, for these users we may say that MySQL is good enough.
- It is also the fact, however, that MySQL still lacks a number of features that are taken for granted with other database systems. If you require such features, then MySQL is (at least for the present) not the database system for you. MySQL is not a panacea.

##### What Is Big Data?

The first thing to recognize is that Big Data does not have one single definition. In fact, it's a term that describes at least three separates, but interrelated, trends:

**Capturing and managing lots of information:**

Numerous independent market and research studies have found that data volumes are doubling every year. Moreover, extra new information, a significant percentage of organizations are also storing three or more years of historic data.

**Working with many new types of data:**

Studies also indicate that 80 percent of data is unstructured (such as images, audio, tweets, text messages, and so on). And until recently, most enterprises have been unable to take full advantage of all this unstructured information. These materials are the copyright of John Wiley & Sons, Inc. and any dissemination, distribution, or unauthorized use is strictly prohibited.

**Exploiting these masses of information and new data types with new styles of applications:**

Many of the tools and technologies that were designed to work with relatively large information volumes have not changed much in the past 15 years. They simply cannot keep up with Big Data, so new classes of analytic applications are reaching the market, all based on a next generation Big Data platform.

These new solutions have the potential to transform the way you run your business. Driving the growth of Big Data Just as no single definition of Big Data exists; no specific cause exists for what is behind its rapid rate of adoption. Instead, several distinct trends have contributed to Big Data's momentum. New data sources today, we have more generators of information than ever before. These data creators include devices such as mobile phones, tablet computers, sensors, medical equipment, and other platforms that gather vast quantities of information.

### **Differentiating between Big Data and traditional enterprise relational data**

Thinking of Big Data as “just lots more enterprise data” is tempting, but it is a serious mistake. First, Big Data is notably larger — often by several orders of magnitude. Secondly, Big Data is commonly generated outside of traditional enterprise applications. And finally, Big Data is often composed of unstructured or semi-structured information types that continually arrive in enormous amounts. To get maximum value from Big Data, it needs to be associated with traditional enterprise data, automatically or via purpose-built applications, reports, queries, and other approaches. For example, a retailer might want to link its Web site visitor behavior logs (a classic Big Data application) with purchase information (commonly found in relational databases). In another case, a mobile phone provider might want to offer a wider range of smartphones to customers (inventory maintained in a relational database) based on text and image message volume trends (unstructured Big Data). Knowing what you can do with Big Data Big Data has the potential to revolutionize the way you do business. It can provide new insights into everything about your enterprise, including the following:

- The way your customers locate and interact with you
- The way you deliver products and services to the marketplace
- The position of organization vs. your competitors
- Strategies you can implement to increase profitability
- And many more

## **What is Hadoop**

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. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers, each of which may be prone to failures. The core components of Hadoop are

- HDFS(Hadoop Distributed File System)
- Map Reduce Programming Model

### **HDFS( Hadoop Distributed File System)**

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is part of the Apache Hadoop Core project

#### **Hdfs assumptions and goals:**

HDFS is a distributed file system designed to handle large data sets and run on commodity hardware. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project.

## HDFS Architecture Design

The figure below gives a run-time view of the architecture showing three types of address spaces: the application, the NameNode and the DataNode. An essential portion of HDFS is that there are multiple instances of DataNode.

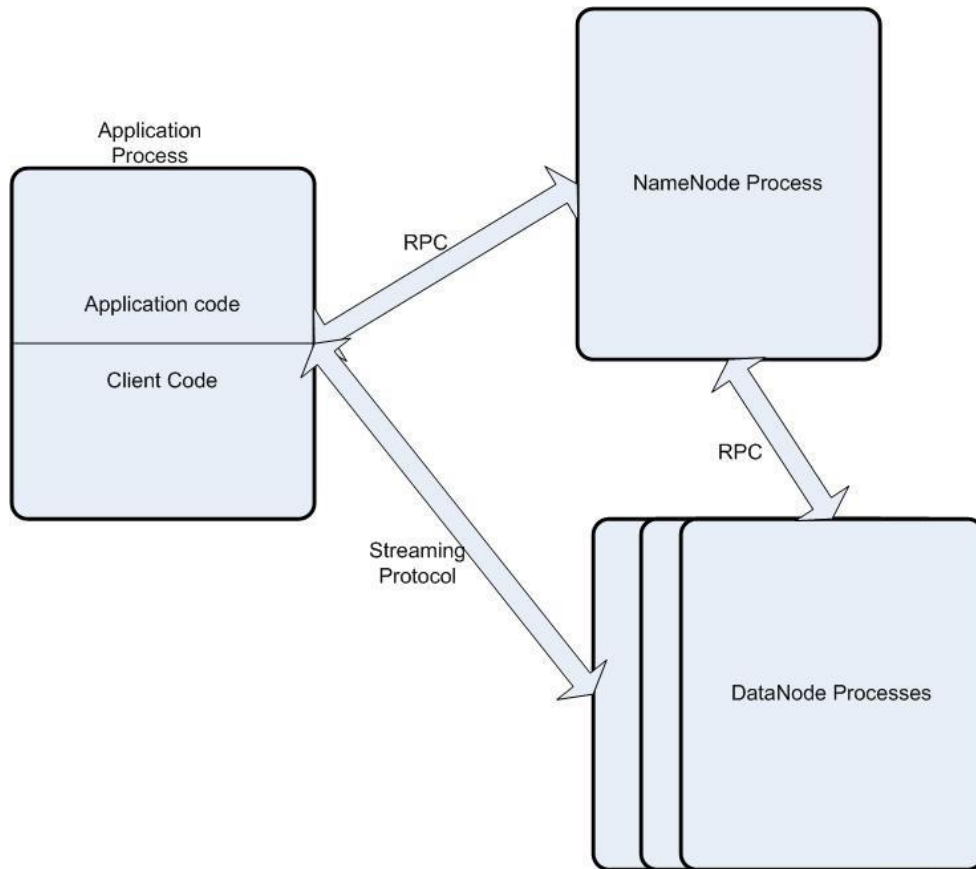


Fig.4.1.HDFS Design Architecture

The application incorporates the HDFS client library into its address space. The client library manages all communication from the application to the Name Node and the DataNode. An HDFS cluster consists of a single NameNode—a master server that manages the file system namespace and regulates access to files by clients. In addition, there are several DataNodes, usually one per computer node in the cluster, which manage storage attached to the nodes that they run on.

The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case.

### **HDFS Files**

There is a distinction between an HDFS file and a native (Linux) file on the host computer. A computer in an HDFS installation is (typically) allocated to one NameNode or one DataNode. Each computer has its own file system and the NameNode manages information about an HDFS file—the metadata— and persistent information is stored in the NameNode's host file system. The information contained in an HDFS file is managed by a DataNode and stored on the DataNode's host computer file system.

HDFS exposes a file system namespace and allows user data to be stored in HDFS files. An HDFS file consists of a number of blocks. Each block is typically 64MBytes. Each block is replicated some specified number of times. The replicas of the blocks are stored on different DataNodes chosen to reflect loading on a DataNode as well as to provide both speed in transfer and resiliency in case of failure of a rack. See Block Allocation for a description of the allocation algorithm.

A standard directory structure is used in HDFS. That is, HDFS files exist in directories

that may in turn be sub-directories of other directories, and so on. There is no concept of a current directory within HDFS. HDFS files are referred to by their fully qualified name which is a parameter of many of the elements of the interaction between the Client and the other elements of the HDFS architecture.

The NameNode executes HDFS file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The list of HDFS files belonging to each block, the current location of the block replicas on the DataNodes, the state of the file, and the access control information is the metadata for the cluster and is managed by the NameNode.

### **Block Allocation**

Each block is replicated some number of times—the default replication factor for HDFS is three. When `addBlock()` is invoked, space is allocated for each replica. Each replica is allocated on a different DataNode. The algorithm for performing this allocation attempts to balance performance and reliability. Considering the following factors does this:

- ❑ The dynamic load on the set of DataNodes. Preference is given to more lightly loaded DataNodes.
- ❑ The location of the DataNodes. Communication between two nodes in different racks has to go through switches. In most cases, network bandwidth between machines in the same rack is greater than network bandwidth between machines in different racks.
- ❑ For the common case, when the replication factor is three, HDFS's placement policy is to put one replica on one node in the local rack, another on a node in a different (remote) rack, and the last on a different node in the same remote rack. This policy cuts the inter-rack write traffic which generally improves write performance. The chance of rack failure is far less than that of a node failure; therefore this co-location policy does not adversely impact data reliability and availability guarantees. However, it does reduce the aggregate network bandwidth used when reading data since a block is placed in only two unique racks rather than three. With this policy, the replicas of a file do not evenly distribute across the racks. One third of replicas are on one node on some rack; the other two thirds of replicas are on distinct nodes





the output of the job are stored in a filesystem. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

Minimally, applications specify the input/output locations and supply *map* and *reduce* functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the *job configuration*. The Hadoop *job client* then submits the job (jar/executable etc.) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

### Inputs and Outputs

The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and value classes have to be serializable by the framework and hence need to implement the Writable interface. Additionally, the key classes have to implement the WritableComparable interface to facilitate sorting by the framework.

Input and Output types of a MapReduce job:

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

## **Apache Hive**

Hive is a data warehouse that uses MapReduce to analyze data stored on HDFS. It provides a query language called HiveQL that closely resembles the common Structured Query Language (SQL) standard.

### **Why Hive**

Actually, for Developing MapReduce Programs, we have Hadoop Streaming and explained that one large benefit of Streaming is how it allows faster turn-around in the development of MapReduce jobs. Hive takes this a step further. Instead of providing away of more quickly developing map and reduce tasks, it offers a query language based on the industry standard SQL. Hive takes these HiveQL statements and immediately and automatically translates the queries into one or more MapReduce jobs. It then executes the overall MapReduce program and returns the results to the user. Whereas Hadoop Streaming reduces the required code/compile/submit cycle, Hive removes it entirely and instead only requires the composition of HiveQL statements.

### **Hive Internal Working:**

Hive internal design includes the following

UI - The user interface for users to submit queries and other operations to the system. Currently the system has a command line interface and a web based GUI is being developed.

Driver - The component which receives the queries. This component implements the notion of session handles and provides execute and fetch APIs modeled on JDBC/ODBC interfaces.

Compiler - The component that parses the query, does semantic analysis on the different query blocks and query expressions and eventually generates an execution plan with the help of the table and partition metadata looked up from the metastore.

Metastore - The component that stores all the structure information of the various tables and partitions in the warehouse including column and column type information, the

serializers and deserializers necessary to read and write data and the corresponding hdfs files where the data is stored.

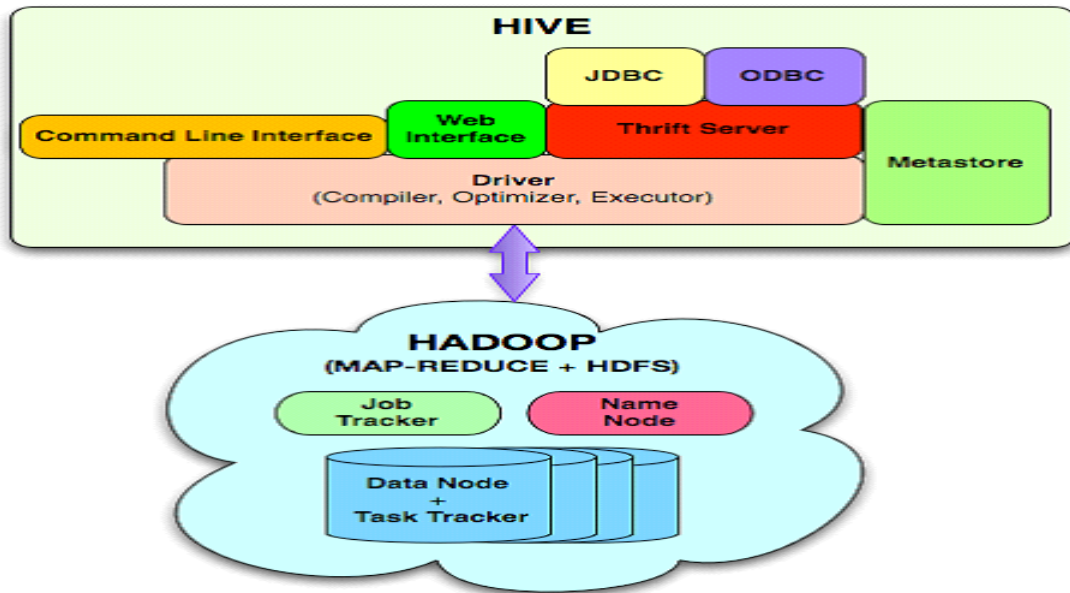


Fig.4.3.Hive Architecture

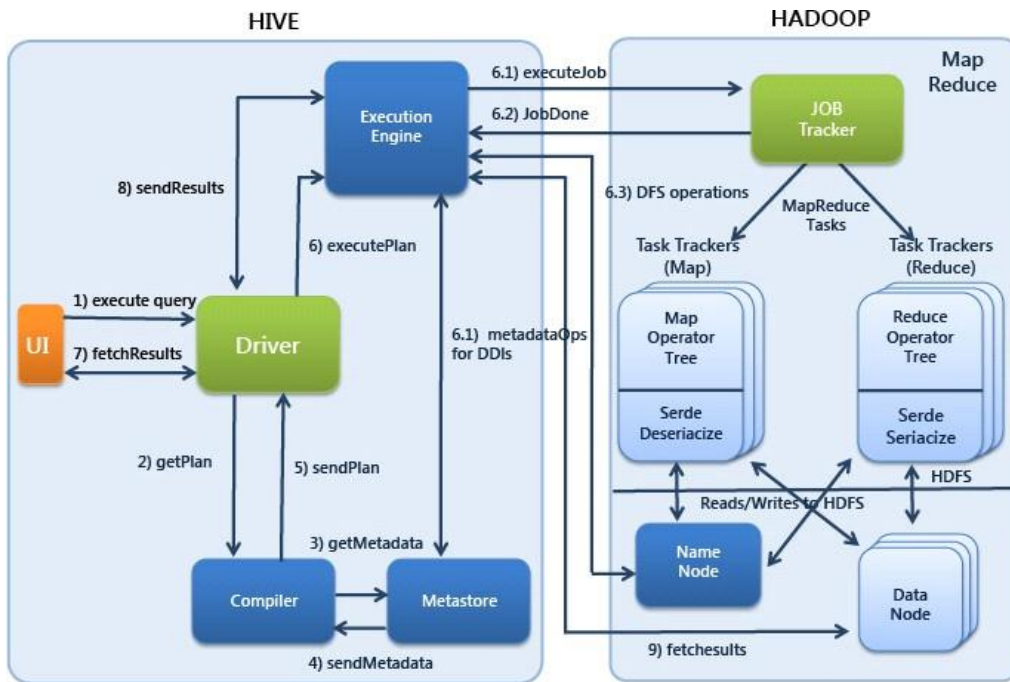


Fig:4.4.Hive Job execution flow

## Tableau

In today's competitive world organizations are expecting a solution that can satisfy their information needs in just one click; solve business queries in seconds; with option to customize as per organizational needs. What Information Technology offers is a range of Business Intelligence Tools that assist organizations in taking business decisions more effectively and efficiently.

Out of 100s of Business Intelligence Tools available in the market, I experienced tableau and believe me it's just awesome. This BI Tool comes with many rich features. You simply have to drag and drop things and you will get what you actually want, that too in just fraction of seconds.

Tableau offers four business intelligence products viz.

- ❑ Tableau Desktop
- ❑ Tableau Server
- ❑ Tableau Digital
- ❑ Tableau Public

## Key differentiators



Fig.4.5. Key Features of Tableau

## 4.2. IMPLEMENTATION

The first phase of our project is installing Hadoop cluster . The proper approach would be a 2-step approach. The first step is to install single-node Hadoop machines, configure and test them as local Hadoop systems. Second step would be to merge that single-node systems into a multi-node cluster. So first let us see how to setup the single node machine.

### 4.2.1. SINGLE NODE SETUP

#### Environment Setup

- Download Ubuntu Linux 12.04.3 LTS.
- Download Hadoop 1.2.1, released August 2013.
- Once the Linux server is set-up, install openssh, by executing the following command.

```
sudo apt-get install openssh-server
```

#### Pre-requisites

There are certain required steps to be done before starting the Hadoop Installation. They are listed as below. Each step is further described below with screenshots and commands for additional understanding. Before proceeding, make sure the Linux box is updated with the latest set of packages from all repositories and PPA's. It can be updated by executing the following command.

```
sudo apt-get update
```



```
hadoopuser@hadoopuser:~$ sudo apt-get update  
[sudo] password for hadoopuser. █
```

#### Install Java 1.6+

Hadoop requires an installation of Java 1.6 or higher. It is always better to go with the latest Java version. In our case we have installed Java 7u-25.

- Download the latest Oracle Java for Linux from the Oracle website by executing the following command.

```
wget https://edelivery.oracle.com/otn-pub/java/jdk/7u25-b18/jdk-7u25-linux-x64.tar.gz
```

- In the event, the download fails, try executing the below command which bypasses the username and password.

```
wget --no-cookies --no-check-certificate --header "Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com" "https://edelivery.oracle.com/otn-pub/java/jdk/7u45-b18/jdk-7u25-linux-x64.tar.gz"
```

```
HTTP request sent, awaiting response... 200 OK
Length: 96316511 (92M) [application/x-gzip]
Saving to: `jdk-7u25-linux-x64.tar.gz'

100%[----->] 96,316,511 311K/s in 5m 3s

2013-10-24 14:34:22 (311 KB/s) - `jdk-7u25-linux-x64.tar.gz' saved [96316511/96316511]
```

- Copy the downloaded Java binaries to the folder /usr/local/java, by executing the following command.

```
sudo cp -r jdk-7u25-linux-x64.tar.gz /usr/local/java
```

- Go to the directory /usr/local/java by executing the following command

```
cd /usr/local/java
```

- Unzip the downloaded Java file by executing the following commands. This unzips the java binaries in the folder /usr/local/java.

```
sudo tar xvzf jdk-7u25-linux-x64.tar.gz
```

```
hadoopuser@hadoopuser:~$ cd /usr/local/java
hadoopuser@hadoopuser:/usr/local/java$ ls
jdk-7u25-linux-x64.tar.gz
hadoopuser@hadoopuser:/usr/local/java$ sudo tar xvzf jdk-7u25-linux-x64.tar.g
```

- Add the following system variables to the path by editing the system Path file located in /etc/profile.

```
sudo nano /etc/profile OR sudo gedit /etc/profile
```

```
hadoopuser@hadoopuser:~$ sudo nano /etc/profile
```

- Add the following lines below to the end of your /etc/profile file:

- JAVA\_HOME=/usr/local/java/jdk1.7.0\_25
- PATH=\$PATH:\$HOME/bin:\$JAVA\_HOME/bin
- export JAVA\_HOME
- export PATH

```
GNU nano 2.2.6 File: /etc/profile Modif
    if [ -r $i ]; then
        . $i
    fi
done
unset i
fi
JAVA_HOME=/usr/local/java/jdk1.7.0_25
PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
export JAVA_HOME
export PATH
```

- Execute the following command to indicate where Oracle Java JDK/JRE is located. This will tell the system that the new Oracle Java version is available for use.

```
sudo update-alternatives --install "/usr/bin/javac" "javac"
"/usr/local/java/jdk1.7.0_40/bin/javac"
```

```
hadoopuser@hadoopuser:~$ sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/local/java/jdk1.7.0_25/bin/javac"
```

## Add a dedicated Hadoop user account

While it is not required, it is generally recommended to create a separate user account to run the Hadoop installation.

- Start by adding a group by executing the following command.

```
sudo addgroup hadoop
```

- Create a user (hduser) and add it to the group (hadoop) created above.

```
sudo adduser --ingroup hadoop hduser
```

You will be asked to provide the password and other information.

```
hadoopuser@hadoopuser:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1001) ...
Done.
hadoopuser@hadoopuser:~$ sudo adduser --ingroup hadoop hduser
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
  Full Name []: hduser
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] Y
hadoopuser@hadoopuser:~$
```

## Configuring SSH Access

Hadoop requires SSH access to manage its nodes, i.e. remote machines and your local machine if you want to use Hadoop on it. This access allows master node to login to its slave nodes and to start and stop the services on them. For single node setup of Hadoop (as in this tutorial), we need to configure SSH access to localhost for the hduser user we created in the step above.

Make sure that SSH is up and running on the Linux box, and configured to allow SSH public key authentication.



- Generate SSH key for user hduser, by executing the following command.

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

- You will be asked the location where you would like to save the key file. Just click <Enter> to go with the default. The RSA key will be generated at '/home/hduser/.ssh'. This key generated with an empty password. Generally, it is not a good practice to have empty passwords, but in this case we need that there should not be any manual intervention when Hadoop is interacting with its nodes.

```
hadoopuser@hadoopuser:~$ su - hduser
Password:
hduser@hadoopuser:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
80:00:79:1a:c2:95:28:1a:17:31:d7:aa: hduser@hadoopuser
The key's randomart image is:
+--[ RSA 2048 ]-----+
|.+.
|..+o.
|-...o
|E...o
|o.
|..
+-----+
hduser@hadoopuser:~$
```

- Enable SSH Access to your local machine with this newly created key, by executing the following command.

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

- Last step is to test the ssh setup by connecting your local machine with the hduser user. This will add your machine's host key fingerprint to the hduser user's known\_hosts file.

```
ssh hduser@localhost
```

```
hduser@hadoopuser:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
hduser@hadoopuser:~$ ssh hduser@localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is [redacted]:[redacted]:[redacted]:[redacted]:41-[redacted]-[redacted]:45:dc:[redacted]:42:[redacted]:[redacted].
Are you sure you want to continue connecting (yes/no)? yes
Please type 'yes' or 'no': yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 12.04.3 LTS (GNU/Linux 3.8.0-29-generic x86_64)

* Documentation:  https://help.ubuntu.com/

System information as of Thu Oct 24 14:54:41 PDT 2013

System load:  0.0                Processes:            87
Usage of /:   6.1% of 20.31GB     Users logged in:    1
Memory usage: 8%                 IP address for eth0: 172.16.17.61
Swap usage:  0%

Graph this data and manage this system at https://landscape.canonical.com/

66 packages can be updated.
29 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

## Disable IPv6

One problem with IPv6 on Ubuntu is that using 0.0.0.0 for the various networking-related Hadoop configuration options will result in Hadoop binding to the IPv6 addresses of the Ubuntu box.

- Log in as the *root*. Open the file `/etc/sysctl.conf` by executing the following command.

```
sudo gedit /etc/sysctl.conf
```

- Add the following lines to the end of the file and reboot the machine, to update the configurations correctly.

- `#disable ipv6`
- `net.ipv6.conf.all.disable_ipv6 = 1`
- `net.ipv6.conf.default.disable_ipv6 = 1`

```
net.ipv6.conf.lo.disable_ipv6 = 1
```

```
# Do not accept IP source route packets (we are not a router)
net.ipv4.conf.all.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
#
# Log Martians Packets
net.ipv4.conf.all.log_martians = 1
#
# Disable IPv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
-- INSERT --
```

```
hadoopuser@hadoopuser:~$ sudo vim /etc/sysctl.conf
[sudo] password for hadoopuser:
hadoopuser@hadoopuser:~$ sudo reboot
```

## Hadoop Installation

Download the latest Hadoop version. In our case it is Hadoop 1.2.1.

- Execute following command to download Hadoop version 1.2.1

```
wget http://mirrors.gigenet.com/apache/hadoop/common/hadoop-1.2.1/hadoop-1.2.1.tar.gz
```

```
hadoopuser@hadoopuser:~$ wget http://mirrors.gigenet.com/apache/hadoop/common/hadoop-1.2.1/hadoop-1.2.1.tar.gz
--2013-10-24 15:35:06-- http://mirrors.gigenet.com/apache/hadoop/common/hadoop-1.2.1/hadoop-1.2.1.tar.gz
Resolving mirrors.gigenet.com (mirrors.gigenet.com)... 69.65.15.34
Connecting to mirrors.gigenet.com (mirrors.gigenet.com)|69.65.15.34|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 63851630 (61M) [application/x-gzip]
Saving to: `hadoop-1.2.1.tar.gz'

1% [          ] 1,147,387   311K/s   eta 3m 22s
```

- Unzip the compressed hadoop file by executing the following command:

```
tar -xvzf hadoop-1.2.1.tar.gz
```

```
hadoopuser@hadoopuser:~$ tar -xvzf hadoop-1.2.1.tar.gz
```

- Move hadoop directory and its contents to the location of your choice. In our case it is /usr/local.

```
sudo mv hadoop /usr/local/
```

```
hadoopuser@hadoopuser:~$ ls
hadoop-1.2.1  hadoop-1.2.1.tar.gz
hadoopuser@hadoopuser:~$ mv hadoop-1.2.1 hadoop
hadoopuser@hadoopuser:~$ ls
hadoop  hadoop-1.2.1.tar.gz
hadoopuser@hadoopuser:~$ sudo mv hadoop /usr/local/
[sudo] password for hadoopuser:
hadoopuser@hadoopuser:~$ ls
hadoop-1.2.1.tar.gz
hadoopuser@hadoopuser:~$ cd /usr/local
hadoopuser@hadoopuser:/usr/local$ ls
bin  etc  games  hadoop  include  java  jdk  man  sbin  share  src
hadoopuser@hadoopuser:/usr/local$
```

- Change the ownership of the files to user hduser and group hadoop.

```
sudo chown -R hduser:hadoop Hadoop
```

```
hadoopuser@hadoopuser:~$ cd /usr/local
hadoopuser@hadoopuser:/usr/local$ sudo chown -R hduser:hadoop hadoop
hadoopuser@hadoopuser:/usr/local$ ls -l
total 40
drwxr-xr-x  2 root  root   4096 Oct 24 12:38 bin
drwxr-xr-x  2 root  root   4096 Oct 24 12:38 etc
drwxr-xr-x  2 root  root   4096 Oct 24 12:38 games
drwxr-xr-x 15 hduser hadoop 4096 Jul 22 15:26 hadoop
drwxr-xr-x  2 root  root   4096 Oct 24 12:38 include
drwxr-xr-x  3 root  root   4096 Oct 24 14:39 java
drwxr-xr-x  3 root  root   4096 Oct 24 12:39 jdk
lrwxrwxrwx  1 root  root     9 Oct 24 12:38 man -> share/man
drwxr-xr-x  2 root  root   4096 Oct 24 12:38 sbin
drwxr-xr-x  6 root  root   4096 Oct 24 12:44 share
drwxr-xr-x  2 root  root   4096 Oct 24 12:38 src
hadoopuser@hadoopuser:/usr/local$
```

## Hadoop Configuration

There are certain files that need to be updated and certain configuration steps to be completed, before Hadoop is up and running. The configurations to be done are listed below. Each step is further described below with screenshots and commands for additional understanding.

### Update \$Home/.bashrc file

```
hduser@hadoopuser:~$  
hduser@hadoopuser:~$ vi .bashrc
```

- Add the following lines to the end of the file.

- export JAVA\_HOME='/usr/local/java/jdk1.7.0\_25'
- export HADOOP\_HOME='/usr/local/hadoop'

```
export PATH=$HADOOP_HOME/bin:$JAVA_HOME/bin:$PATH
```

```
# sources /etc/bash.bashrc).  
if [ -f /etc/bash_completion ] && ! shopt -oq posix; then  
  . /etc/bash_completion  
fi  
export JAVA_HOME='/usr/local/java/jdk1.7.0_25'  
export HADOOP_HOME='/usr/local/hadoop'  
export PATH=$HADOOP_HOME/bin:$PATH  
-- INSERT --
```

- Reload the .bashrc file by executing the following command.

```
..bashrc
```

#### Configure hadoop-env.sh.

The only required environment variable we have to configure for Hadoop in this guide is JAVA\_HOME. Open conf/hadoop-env.sh in the editor of your choice (if you used the installation path in this tutorial, the full path is /usr/local/hadoop/conf/hadoop-env.sh) and set the JAVA\_HOME environment variable to the Sun JDK/JRE 7 directory.

```
export JAVA_HOME=/usr/local/java/jdk1.7.0_25
```



```

hadoopuser@hadoopuser: /usr/local/hadoop$ cd conf/
hadoopuser@hadoopuser: /usr/local/hadoop/conf$ ls
capacity-scheduler.xml      hadoop-policy.xml          slaves
configuration.xml          hdfs-site.xml             ssl-client.xml.example
core-site.xml              log4j.properties         ssl-server.xml.example
fair-scheduler.xml        mapred-queue-acls.xml    taskcontroller.cfg
hadoop-env.sh             mapred-site.xml          task-log4j.properties
hadoop-metrics2.properties masters
hadoopuser@hadoopuser: /usr/local/hadoop/conf$ vi hadoop-env.sh
hadoopuser@hadoopuser: /usr/local/hadoop/conf$ vi hadoop-env.sh
hadoopuser@hadoopuser: /usr/local/hadoop/conf$ sudo vi hadoop-env.sh

```

```

# The Java implementation to use. Required.
export JAVA_HOME=/usr/local/java/jdk1.7.0_25

# Extra Java CLASSPATH elements. Optional.
# export HADOOP_CLASSPATH=

```

Configure /conf/\*-site.xml files

In this section, we will configure the directory where Hadoop stores its data files, the network ports it listens to etc.

- Login as a non hadoop user. Create a directory called data, by executing the following command.

```
sudo mkdir /data
```

```

hadoopuser@hadoopuser:~$ sudo mkdir /data
hadoopuser@hadoopuser:~$

```

- Change the ownership of this folder to user *hduser*.

```
sudo chown hduser:hadoop /data
```

```

hadoopuser@hadoopuser:~$ sudo chown hduser:hadoop /data
hadoopuser@hadoopuser:~$ ls -l /
total 81
drwxr-xr-x  2 root  root   4096 Oct 24 12:43 /bin
drwxr-xr-x  4 root  root   1024 Oct 24 12:45 /boot
drwxr-xr-x  2 hduser hadoop 4096 Oct 24 16:25 /data

```

- Create a directory **tmp** under the data directory.

- Login as user *hduser*. Edit the *core-site.xml* in */usr/local/hadoop/conf* directory

```
$su -hduser
$Cd /usr/local/hadoop/conf
$vi core-site.xml
```

```
hadoopuser@hadoopuser:~$ su - hduser
Password:
hduser@hadoopuser:~$ cd /usr/local/hadoop/conf/
hduser@hadoopuser:/usr/local/hadoop/conf$ ls
capacity-scheduler.xml      hadoop-policy.xml          slaves
configuration.xsl          hdfs-site.xml             ssl-client.xml.example
core-site.xml               log4j.properties          ssl-server.xml.example
fair-scheduler.xml         mapred-queue-acls.xml     taskcontroller.cfg
hadoop-env.sh              mapred-site.xml           task-log4j.properties
hadoop-metrics2.properties masters
hduser@hadoopuser /usr/local/hadoop/conf$ vi core-site.xml
```

- Add the following entry to the file. Save and quit the file.

```
<property>
<name>hadoop.tmp.dir</name>
<value>/data/tmp</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:54310</value>
<description>The name of the default file system.
A URI whose scheme and authority determine the FileSystem implementation. The
uri's scheme determines the config property (fs.SCHEME.impl) naming the File
System
implementation class. The uri's authority is used to determine the host, port, etc.
for a file system.</description>
</property>
```

```

<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/data/tmp</value>
  <description>A base for other temporary directories.</description>
</property>

<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system. A URI whose
  scheme and authority determine the FileSystem implementation. The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class. The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
</property>
</configuration>

```

- Edit the mapred-site.xml in /usr/local/hadoop/conf directory

```
vi mapred-site.xml
```

```
hduser@hadoopuser: /usr/local/hadoop/conf$ vi mapred-site.xml
```

- Add the following entry to the file and save and quit the file.

```

<property>
<name>mapred.job.tracker</name>
<value>localhost:54311</value>
<description>The host and port that the MapReduce job tracker runs at. If "local",
then jobs are run in-process as a single map and reduce task.
</description>
</property>

```

```
hduser@hadoopuser: /usr/local/hadoop/conf$ vi mapred-site.xml
```

- Edit the hdfs-site.xml in /usr/local/hadoop/conf directory.



- Add the following entry to the file and save and quit the file.

```
<Property>
<Name>dfs.replication</name>
<Value>1</value>
<Description>Default block replication. The actual number of replications can be
specified
when the file is created. The default is used if replication is not specified in create
time.
</description>
</property>
```

Format the HDFS File system and Starting Hadoop server

The first step to starting your Hadoop installation is the formatting of the Hadoop file system (HDFS) implemented on top of your local file system of your cluster. This step is required the first time you set up a Hadoop cluster. Do not format a running Hadoop file system as you will lose all the data currently in the cluster (in HDFS)!

- To format the file system (which simply initializes the directory specified by the `dfs.name.dir` variable), execute the following command in the `$HADOOP_HOME/bin` directory

```
hadoop namenode -format
```

## start-all.sh

```
hduser@hadoopuser:/usr/local/hadoop/bin$ start-all.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/local/hadoop/libexec/../../logs/hadoop-hduser-namenode-hadoopuser.out
localhost: starting datanode, logging to /usr/local/hadoop/libexec/../../logs/hadoop-hduser-datanode-hadoopuser.out
localhost: starting secondarynamenode, logging to /usr/local/hadoop/libexec/../../logs/hadoop-hduser-secondarynamenode-hadoopuser.out
starting jobtracker, logging to /usr/local/hadoop/libexec/../../logs/hadoop-hduser-jobtracker-hadoopuser.out
localhost: starting tasktracker, logging to /usr/local/hadoop/libexec/../../logs/hadoop-hduser-tasktracker-hadoopuser.out
hduser@hadoopuser:/usr/local/hadoop/bin$
```

```
hadoopuser@hadoopuser:~$ cd /usr/local/hadoop/bin
hadoopuser@hadoopuser:/usr/local/hadoop/bin$ ls
hadoop          start-all.sh          stop-balancer.sh
hadoop-config.sh start-balancer.sh      stop-dfs.sh
hadoop-daemon.sh start-dfs.sh           stop-jobhistoryserver.sh
hadoop-daemons.sh start-jobhistoryserver.sh stop-mapred.sh
rcc              start-mapred.sh        task-controller
slaves.sh        stop-all.sh
hadoopuser@hadoopuser:/usr/local/hadoop/bin$ hadoop namenode -format
```

```
hduser@hadoopuser:/usr/local/hadoop/bin$ hadoop namenode -format
Warning: $HADOOP_HOME is deprecated.

13/10/24 16:49:40 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:  host = hadoopuser/127.0.1.1
STARTUP_MSG:  args = [-format]
STARTUP_MSG:  version = 1.2.1
STARTUP_MSG:  build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r 1503152; compiled by 'mattf' on Mon Jul 22 15:23:09 PDT 2013
STARTUP_MSG:  java = 1.7.0_25
*****/
13/10/24 16:49:40 INFO util.GSet: Computing capacity for map BlocksMap
13/10/24 16:49:40 INFO util.GSet: VM type = 64-bit
13/10/24 16:49:40 INFO util.GSet: 2.0% max memory = 932118528
13/10/24 16:49:40 INFO util.GSet: capacity = 2^21 = 2097152 entries
13/10/24 16:49:40 INFO util.GSet: recommended=2097152, actual=2097152
13/10/24 16:49:40 INFO namenode.FSNamesystem: fsOwner=hduser
13/10/24 16:49:40 INFO namenode.FSNamesystem: supergroup=supergroup
13/10/24 16:49:40 INFO namenode.FSNamesystem: isPermissionEnabled=true
13/10/24 16:49:40 INFO namenode.FSNamesystem: dfs.block.invalidate.limit=100
13/10/24 16:49:40 INFO namenode.FSNamesystem: isAccessTokenEnabled=false accessKeyUpdateInterval=0 min(s), accessTokenLifetime=0 min(s)
13/10/24 16:49:40 INFO namenode.FSEditLog: dfs.namenode.edits.toleration.length = 0
```

- Start the Hadoop server by executing the following command

- Run `jps` command to see your all the services up and running

```
hduser@hadoopuser:/usr/local/hadoop$ jps
2443 Jps
1748 DataNode
2011 SecondaryNameNode
1516 NameNode
2328 TaskTracker
2092 JobTracker
hduser@hadoopuser:/usr/local/hadoop$
```

- Run `netstat -plten | grep java` to see list of ports running.

```
hduser@hadoopuser:~$ netstat -plten | grep java
(Most all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:50075        0.0.0.0:*            LISTEN    1001      10704      1748/java
tcp        0      0 0.0.0.0:42877       0.0.0.0:*            LISTEN    1001      9976       2011/java
tcp        0      0 0.0.0.0:50020       0.0.0.0:*            LISTEN    1001      9998       1748/java
tcp        0      0 127.0.0.1:54310     0.0.0.0:*            LISTEN    1001     10387      1516/java
tcp        0      0 127.0.0.1:54311     0.0.0.0:*            LISTEN    1001     10029      2092/java
tcp        0      0 0.0.0.0:50090       0.0.0.0:*            LISTEN    1001     10809      2011/java
tcp        0      0 0.0.0.0:52171       0.0.0.0:*            LISTEN    1001     10006      2092/java
tcp        0      0 0.0.0.0:50060       0.0.0.0:*            LISTEN    1001     11284      2328/java
tcp        0      0 127.0.0.1:48621     0.0.0.0:*            LISTEN    1001     11027      2328/java
tcp        0      0 0.0.0.0:50030       0.0.0.0:*            LISTEN    1001     10035      2092/java
tcp        0      0 0.0.0.0:51503       0.0.0.0:*            LISTEN    1001     9841       1748/java
tcp        0      0 0.0.0.0:50070       0.0.0.0:*            LISTEN    1001     9842       1516/java
```

- Stop Hadoop by running the following command

```
stop-all.sh
```

```
hduser@hadoopuser:~$ stop-all.sh
Warning: $HADOOP_HOME is deprecated.

stopping jobtracker
localhost: stopping tasktracker
stopping namenode
localhost: stopping datanode
localhost: stopping secondarynamenode
hduser@hadoopuser:~$
```

So now we have seen how to setup a single node machine .Now we will see how to setup a multimode cluster from these single node machines.

## 4.2.2. MULTI NODE CLUSTER SETUP

Networking plays an important role here, before merging both single node servers into a multi node cluster we need to make sure that all the nodes ping each other( they need to be connected on the same network / hub or all the machines can speak to each other). Once we are done with this process, we will be moving to the next step in selecting the master node and slave node, here we are selecting 172.16.17.68 as the master machine(Hadoopmaster) and remaining are as slaves (hadoopnode) . Then we need to add them in '/etc/hosts' file on each machine as follows.

```
sudo vi /etc/hosts
```

```
hadoopmaster@Hadoopmaster:~$ sudo vi /etc/hosts
```

```
172.16.17.68 Haadoopmaster
```

```
172.16.17.61 hadoopnode
```

**Note:** The addition of more slaves should be updated here in each machine using unique names for slaves (e.g.: 172.16.17.xx hadoonode01, 172.16.17.xy slave02 so on..).

```
172.16.17.68 Hadoopmaster
172.16.17.61 hadoopnode
127.0.0.1 localhost localhost.localdomain

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouter
```

## ii. Enabling SSH:

hduser on master(Hadoopmaster) machine need to able to connect to its own master (Hadoopmaster) account user and also need to connect hduser to the slaves (hadoopnode) machines via password-less SSH login.

```
hduser@Hadoopmaster:~$ ssh-copy-id -i ~/.ssh/id_rsa.pub hduser@hadoopnode
```

```
hduser@Hadoopmaster:~$ ssh-copy-id -i ~/.ssh/id_rsa.pub hduser@hadoopnode
The authenticity of host 'hadoopnode (172.16.17.61)' can't be established.
ECDSA key fingerprint is 45:9f:1e:67:6d:74:41:74:15:45:d5:dc:52:42:de:56.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hadoopnode,172.16.17.61' (ECDSA) to the list of known hosts.
hduser@hadoopnode's password: █
```

If you can see the below output when you run the given command on both master and slaves, then we configured it correctly.

```
ssh Hadoopmaster
```

or

```
ssh master
```

```
ssh hadoopnode
```

```
[hduser@master ~]$ ssh master
Last login: Sun Feb 15 15:45:12 2015 from master
[hduser@master ~]$ █
```

```
hduser@Hadoopmaster:~$ ssh hadoopnode
Welcome to Ubuntu 12.04.3 LTS (GNU/Linux 3.8.0-29-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Sat Oct 26 16:34:15 PDT 2013

System load:  0.0                Processes:            79
Usage of /:   8.7% of 20.31GB    Users logged in:    0
Memory usage: 1%                IP address for eth0: 172.16.17.61
Swap usage:  0%

Graph this data and manage this system at https://landscape.canonical.com/

Last login: Sat Oct 26 16:31:56 2013 from hadoopmaster
hduser@hadoopnode:~$ exit
logout
Connection to hadoopnode closed.
```

### Configurations:

The following are the required files we will use for the perfect configuration of the multi node Hadoop cluster.

- a. masters
- b. slaves
- c. core-site.xml
- d. mapred-site.xml
- e. hdfs-site.xml

Lets configure each and every config file accordingly:

#### a. masters:

In master (Hadoopmaster) machine we need to configure masters file accordingly as shown in the image and add the master (Hadoopmaster) node name.

```
vi masters

Hadoopmaster
```



```
hduser@Hadoopmaster:~$ cd /usr/local/hadoop/conf
hduser@Hadoopmaster:/usr/local/hadoop/conf$ ls
capacity-scheduler.xml      hadoop-policy.xml          slaves
configuration.xml          hdfs-site.xml             ssl-client.xml.example
core-site.xml              log4j.properties         ssl-server.xml.example
fair-scheduler.xml        mapred-queue-acls.xml    taskcontroller.cfg
hadoop-env.sh              mapred-site.xml           task-log4j.properties
hadoop-metrics2.properties masters
hduser@Hadoopmaster:/usr/local/hadoop/conf$ vi masters
```

```
Hadoopmaster
```

**b. slaves:**

Lists the hosts, one per line, where the Hadoop slave daemons (DataNodes and TaskTrackers) will be running as shown:

```
Hadoopmaster
hadoopnode
```

```
hduser@Hadoopmaster:~$ cd /usr/local/hadoop/conf
hduser@Hadoopmaster:/usr/local/hadoop/conf$ ls
capacity-scheduler.xml      hadoop-policy.xml          slaves
configuration.xml          hdfs-site.xml             ssl-client.xml.example
core-site.xml              log4j.properties         ssl-server.xml.example
fair-scheduler.xml        mapred-queue-acls.xml    taskcontroller.cfg
hadoop-env.sh              mapred-site.xml           task-log4j.properties
hadoop-metrics2.properties masters
hduser@Hadoopmaster:/usr/local/hadoop/conf$ vi masters
hduser@Hadoopmaster:/usr/local/hadoop/conf$ vi slaves
```

```
Hadoopmaster
hadoopnode
```

If you have additional slave nodes, just add them to the conf/slaves file, one hostname per line.

**Configuring all \*-site.xml files:**

We need to use the same configurations on all the nodes of hadoop cluster, i.e. we need to edit all \*-site.xml files on each and every server accordingly.

### c. core-site.xml:

We are changing the host name from 'localhost' to Hadoopmaster, which specifies the NameNode (the HDFS master) host and port.

```
vi core-site.xml
```

```
hduser@Hadoopmaster:/usr/local/hadoop/conf$ ls
capacity-scheduler.xml      hadoop-policy.xml          slaves
configuration.xml          hdfs-site.xml             ssl-client.xml.example
core-site.xml               log4j.properties          ssl-server.xml.example
fair-scheduler.xml         mapred-queue-acls.xml     taskcontroller.cfg
hadoop-env.sh              mapred-site.xml           task-log4j.properties
hadoop-metrics2.properties masters
hduser@Hadoopmaster:/usr/local/hadoop/conf$ vi core-site.xml
```

```
<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/data/tmp</value>
  <description>A base for other temporary directories.</description>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://Hadoopmaster:54310</value>
  <description>The name of the default file system. A URI whose
scheme and authority determine the FileSystem implementation. The
uri's scheme determines the config property (fs.SCHEME.impl) naming
the FileSystem implementation class. The uri's authority is used to
determine the host, port, etc. for a filesystem.</description>
</property>
```

### d. hdfs-site.xml:

We are changing the replication factor to "2", The default value of dfs.replication is 3. However, we have only two nodes available, so we set dfs.replication to 2.

```
vi hdfs-site.xml
```



```

<configuration>
<property>
  <name>dfs.replication</name>
  <value>2</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
</property>
</configuration>

```

#### e. mapred-site.xml:

We are changing the host name from 'localhost' to Hadoopmaster, which specifies the JobTracker (MapReduce master) host and port

```
vi mapred-site.xml
```

```

<configuration>
<property>
  <name>mapred.job.tracker</name>
  <value>Hadoopmaster:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at. If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
</property>
</configuration>

```

#### Formatting and Starting/Stopping the HDFS filesystem via the NameNode:

The first step to starting up your multi-node Hadoop cluster is formatting the Hadoop filesystem which is implemented on top of the local filesystem of your cluster. To format the filesystem (which simply initializes the directory specified by the dfs.name.dir variable), run the given command.

```
hadoop namenode -format
```

```
[hduser@master ~]$ ssh master
Last login: Sun Feb 15 15:45:12 2015 from master
[hduser@master ~]$ hadoop namenode -format
15/02/15 15:49:00 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = master/172.16.40.173
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 1.0.4
STARTUP_MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.0 -r 139329
; compiled by 'hortonfo' on Wed Oct 3 05:13:58 UTC 2012
*****/
Re-format filesystem in /app/hadoop/tmp/dfs/name ? (Y or N) █
```

### Starting the multi-node cluster:

Starting the cluster is performed in two steps.

We begin by starting the HDFS daemons first, the NameNode daemon is started on Hadoopmaster and DataNode daemons are started on all nodes(slaves).

Then we will start the MapReduce daemons, the JobTracker is started on Hadoopmaster and TaskTracker daemons are started on all nodes (slaves).

#### a. To start HDFS daemons:

```
start-dfs.sh
```

This will get NameNode up and DataNodes up listed in conf/slaves.

```
hduser@Hadoopmaster:~$ start-dfs.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/local/hadoop/libexec/./logs/hadoop-hduser-na
menode-Hadoopmaster.out
hadoopnode: starting datanode, logging to /usr/local/hadoop/libexec/./logs/hado
op-hduser-datanode-hadoopnode.out
Hadoopmaster: starting datanode, logging to /usr/local/hadoop/libexec/./logs/ha
dooop-hduser-datanode-Hadoopmaster.out
Hadoopmaster: starting secondarynamenode, logging to /usr/local/hadoop/libexec/.
./logs/hadoop-hduser-secondarynamenode-Hadoopmaster.out
```

By running jps command, we will see list of java processes running on master and slaves:

```
hduser@Hadoopmaster:~$ jps
10446 SecondaryNameNode
9927 NameNode
10508 Jps
10171 DataNode
hduser@Hadoopmaster:~$
```

**b. To start Map Red daemons:**

```
start-mapred.sh
```

This will bring up the MapReduce cluster with the JobTracker running on the machine you ran the previous command on, and TaskTrackers on the machines listed in the conf/slaves file.

```
hduser@Hadoopmaster:~$ start-mapred.sh
Warning: $HADOOP_HOME is deprecated.

starting jobtracker, logging to /usr/local/hadoop/libexec/./logs/hadoop-hduser-
jobtracker-Hadoopmaster.out
Hadoopmaster: starting tasktracker, logging to /usr/local/hadoop/libexec/./logs
/hadoop-hduser-tasktracker-Hadoopmaster.out
hadoopnode: starting tasktracker, logging to /usr/local/hadoop/libexec/./logs/h
adoop-hduser-tasktracker-hadoopnode.out
hduser@Hadoopmaster:~$
```

By running jps command, we will see list of java processes including JobTracker and TaskTracker running on master and slaves:.

```
hduser@Hadoopmaster:~$ jps
10446 SecondaryNameNode
10833 JobTracker
11196 Jps
9927 NameNode
11080 TaskTracker
10171 DataNode
hduser@Hadoopmaster:~$
```

**c. To stop Map Red daemons:**

```
stop-mapred.sh
```

```
hduser@Hadoopmaster:~$ stop-mapred.sh
Warning: $HADOOP_HOME is deprecated.

stopping jobtracker
hadoopnode: stopping tasktracker
Hadoopmaster: stopping tasktracker
hduser@Hadoopmaster:~$
```

d. To stop HDFS daemons:

```
stop-dfs.sh
```

```
hduser@Hadoopmaster:~$ stop-dfs.sh
Warning: $HADOOP_HOME is deprecated.

stopping namenode
Hadoopmaster: stopping datanode
hadoopnode: stopping datanode
Hadoopmaster: stopping secondarynamenode
hduser@Hadoopmaster:~$
```

Now our cluster setup is done .And the next step is to configure the hive .So now let us see how to install hive.

### 4.2.3. HIVE CONFIGURATION

#### Introduction

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. Apache Hive supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem. It provides an SQL-like language called HiveQL(Hive Query Language) while maintaining full support for map/reduce.

#### Installing HIVE:

- Browse to the link: <http://apache.claz.org/hive/stable/> to download the hive
- Click the apache-hive-0.13.0-bin.tar.gz
- Save and Extract it

#### Commands

```
user@ubuntu:~$ cd /usr/lib/  
user@ubuntu:~$ sudo mkdir hive  
user@ubuntu:~$ cd Downloads  
user@ubuntu:~$ sudo mv apache-hive-0.13.0-bin /usr/lib/hive
```

Setting Hive environment variable:

#### Commands

```
user@ubuntu:~$ cd  
user@ubuntu:~$ sudo gedit ~/.bashrc
```

Copy and paste the following lines at end of the file

```
# Set HIVE_HOME  
export HIVE_HOME="/usr/lib/hive/apache-hive-0.13.0-bin"  
PATH=$PATH:$HIVE_HOME/bi
```

```
user@ubuntu:~$ cd /usr/lib/hive/apache-hive-0.13.0-bin/bin
```

```
user@ubuntu:~$ sudo gedit hive-config.sh
```

Go to the line where the following statements are written

```
# Allow alternate conf dir location.
```

```
HIVE_CONF_DIR="${HIVE_CONF_DIR:-$HIVE_HOME/conf}"
```

```
export HIVE_CONF_DIR=$HIVE_CONF_DIR
```

```
export HIVE_AUX_JARS_PATH=$HIVE_AUX_JARS_PATH
```

Below this write the following

```
export HADOOP_HOME=/usr/local/hadoop      (write the path where hadoop file is there)
```

Create Hive directories within HDFS

Command

```
user@ubuntu:~$ hadoop fs -mkdir /usr/hive/warehouse
```

Setting READ/WRITE permission for table

Command

```
user@ubuntu:~$ hadoop fs -chmod g+w /usr/hive/warehouse
```

HIVE launch

Command

```
user@ubuntu:~$ hive
```

Hive shell will prompt:

OUTPUT

Shell will look like

```
<property>
<name>hive.metastore.local</name>
<value>TRUE</value>
<description>controls whether to connect to remove metastore server or open a
new metastore server in Hive Client JVM</description>
</property>
```

```
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://usr/lib/hive/apache-hive-0.13.0-
bin/metastore_db? createDatabaseIfNotExist=true</value>
<description>JDBC connect string for a JDBC metastore</description>
</property>
```

```
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
```

```
<property>
<name>hive.metastore.warehouse.dir</name>
<value>/usr/hive/warehouse</value>
<description>location of default database for the warehouse</description>
</property>
```

Creating a database

Command

```
hive> create database mydb;
```

#### 4.2.4. SQOOP CONFIGURATION

##### Prerequisites

Before we can use Sqoop, a release of Hadoop must be installed and configured. Sqoop is currently supporting 4 major Hadoop releases - 0.20, 0.23, 1.X.X and 2.X.X. We have installed Hadoop 1.2.1 and it is compatible with sqoop 1.4.4. We are using a Linux environment Ubuntu 12.04 to install and run sqoop. The basic familiarity with the purpose and operation of Hadoop is required to use this product.

##### Installation

To install the sqoop 1.4.4 we followed the given sequence of steps:

1. Download the sqoop-1.4.4.bin\_hadoop-1.0.0.tar.gz file from [www.apache.org/dyn/closer.cgi/sqoop/1.4.4](http://www.apache.org/dyn/closer.cgi/sqoop/1.4.4)
2. Unzip the tar file: `sudo tar -zxvf sqoop-1.4.4.bin_hadoop1.0.0.tar.gz`
3. Move sqoop-1.4.4.bin\_hadoop1.0.0 to sqoop using command

```
user@ubuntu:~$ sudo mv sqoop-1.4.4.bin_hadoop1.0.0 /usr/local/sqoop
```

4. Create a directory sqoop in usr/lib using command

```
user@ubuntu:~$ sudo mkdir /usr/lib/sqoop
```

5. Go to the zipped folder sqoop-1.4.4.bin\_hadoop-1.0.0 and run the command

```
user@ubuntu:~$ sudo mv /* /usr/lib/sqoop
```

6. Go to root directory using cd command

```
user@ubuntu:~$ cd
```

7. Open bashrc file using

```
user@ubuntu:~$ sudo gedit ~/.bashrc
```

8. Add the following lines



9. export SQOOP\_HOME=jusr/lib/sqoop
10. export PATH=\$PATH:\$SQOOP\_HOME/bin
11. To check if the sqoop has been installed successfully type the command

**\$sqoop version**

#### **4.2.5. DATA PROCESSING TASK**

Up to now we have configured the hadoop and its eco systems/tools required to get that data into hadoop and process that data. So now the first thing to do here is getting the data from Mysql into HDFS .Before to that we need to grant the privileges on that database. The command to be executed for granting the privileges are given below.

```
GRANT ALL PRIVILEGES ON mydb.* TO 'user'@'%';  
GRANT ALL PRIVILEGES ON mydb.* TO '%'@'%';  
GRANT ALL PRIVILEGES ON mydb.* TO "@'%';
```

##### **4.2.5.1. Import table from MySql to HDFS**

Import table from Mysql to HDFS using the following command

```
$Sqoop import --connect jdbc:mysql://127.0.0.1/mydb --table udr_table -m1
```

```
Applications Places System Sun Feb 15, 3:26 PM lohith
hduser@master:~
File Edit View Search Terminal Help
[sudo] password for hduser:
[hduser@master ~]$ sqoop import --connect jdbc:mysql://master/test_db --table udr --direct
Warning: /home/hduser/sqoop/./hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/hduser/sqoop/./accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/hduser/sqoop/./zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
15/02/15 15:24:42 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5
15/02/15 15:24:42 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
15/02/15 15:24:42 INFO tool.CodeGenTool: Beginning code generation
15/02/15 15:24:42 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `udr` AS t LIMIT 1
15/02/15 15:24:42 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `udr` AS t LIMIT 1
15/02/15 15:24:42 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/local/hadoop
Note: /tmp/sqoop-hduser/compile/5a4ba40060d6a75f4edb72d8f9d54cdd/udr.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
15/02/15 15:24:45 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-hdus
```

Now we can check our imported data in hdfs using the following command

```
$hadoop fs -lsr
```

Or we can view the data from the browser also as below by using the name node URI  
127.0.0.1:50070

<a href="#">hourly_poc</a>	dir			2015-02-04 17:39	rwxr-xr-x	hduser	superg
<a href="#">market_poc</a>	dir			2015-02-04 17:42	rwxr-xr-x	hduser	superg
<a href="#">partner_poc</a>	dir			2015-02-04 17:41	rwxr-xr-x	hduser	superg
<a href="#">partner_share_poc</a>	dir			2015-02-04 17:44	rwxr-xr-x	hduser	superg
<a href="#">udr_poc</a>	dir			2015-02-04 17:34	rwxr-xr-x	hduser	superg

[Go back to DFS home](#)

## Local logs

#### 4.2.5.2. Analysing the data with Hive

Now we have the data in hadoop distributed file system(HDFS) ,so we have to load and process this data in hive now.For this we have to execute the following script in Hive Shell

First enter into Hive shell using the following command

```
$Hive
```

Shell will look like

```
[hduser@master ~]$ hive
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated. Please use org.apache.hadoop.log.metrics.EventCounter in all the log4j.properties files.
Logging initialized using configuration in jar:file:/home/hduser/hive-0.9.0/lib/hive-common-0.9.0.jar!/hive-log4j.properties
Hive history file=/tmp/hduser/hive_job_log_hduser_201502151531_849549347.txt
hive>
```

In this Hive shell we have to run our following scripts

### Scripts to Run

```
create table if not exists
```

```
udr_poc( Sub_Name string,
```

```
Start_time
```

```
string,
```

```
End_time
```

```
string,
```

```
Trans_Bytes
```

```
bigint,
```

```
Rec_Bytes
```

```
bigint,
```

```
Tot_Bytes
```

```
bigint, partner
```

```
String, market
```

```
String )
```

```
partitioned by ( date1 string )
```

```
row format delimited fields terminated
```

```
by ',' stored as textfile ;
```

```
load data inpath '/home/hduser/hiveload/udr_poc/part-m-00000' into
```

```
table udr_poc partition ( date1 = '2013-09-30' );
```

```
create table if not exists
```

```
hourly_poc( date1 string,
```

```
day_slot
```

```
int,
```

```
sub_count
```

```
bigint,
```

```
tot_bytes
```

```
bigint )
```

```
row format delimited fields terminated
```

```

Partner_name String,
Market_name string,
Tot_bytes
                bigi
nt, Date1
                strin
g
)
row format delimited fields terminated
by ',' stored as textfile;

insert overwrite table partner_poc
select Partner,market, sum(tot_Bytes) as sum1, date1 from udr_poc group
by Partner,market,date1;

```

```

create table if not exists
market_poc( Market_name
String,
Tot_bytes
bigint, Date1
                st
ring
)
row format delimited fields terminated
by ',' stored as textfile;

insert overwrite table market_poc
select market , sum(tot_Bytes) as sum1, date1 from udr_poc group by market,date1;

```

```

create table if not exists
partner_share_poc( Market_name

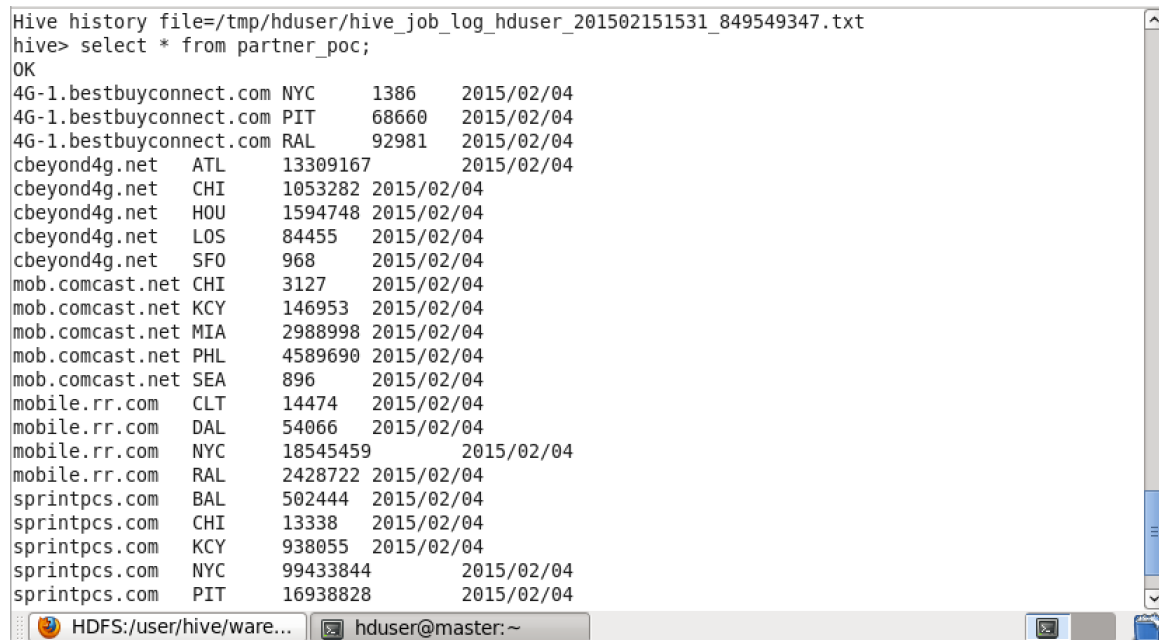
```

```

)
row format delimited fields terminated
by ',' stored as textfile;
insert overwrite table partner_share_poc
select
market_poc.market_name,market_poc.tot_bytes,p.partner_name,
p.tot_bytes, p.date1, (p.tot_bytes /market_poc.tot_bytes) * 100 as
rate1 from market_poc
join partner_poc p on
market_poc.market_name = p.market_name
and market_poc.date1 = p.date1 ;

```

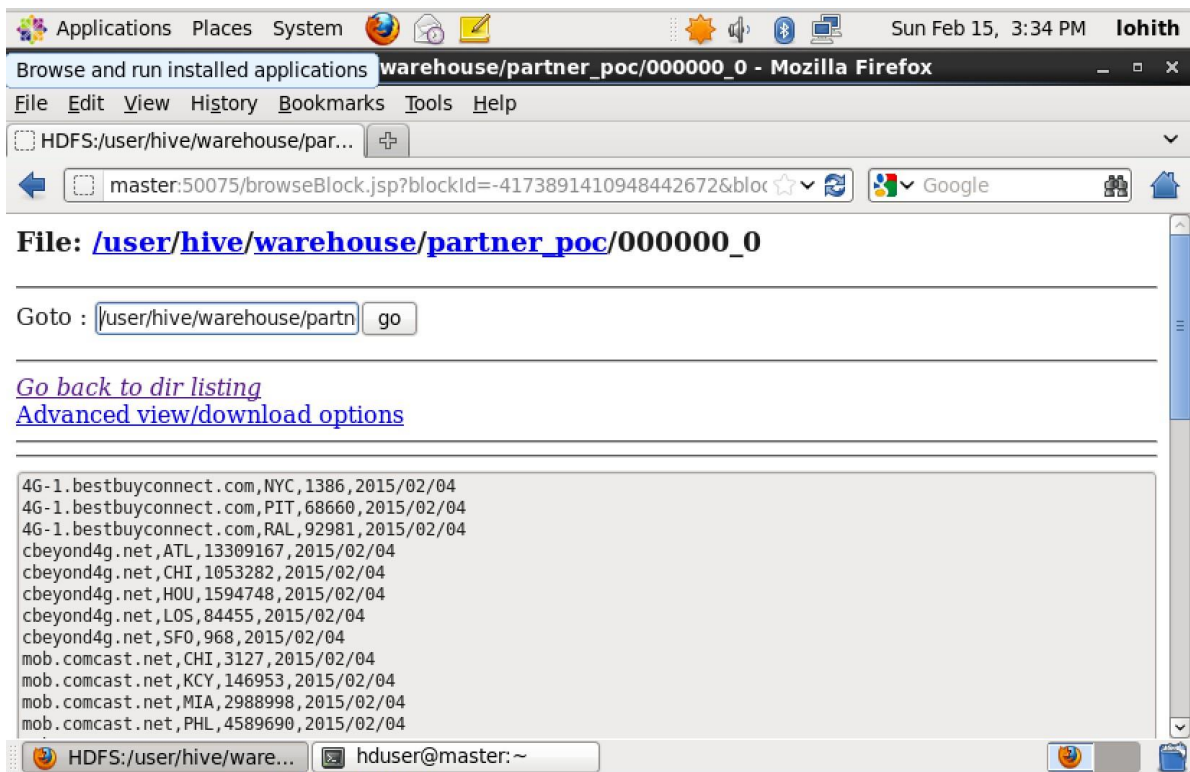
Now the data has been processed and stored in hive. We can view that data from the browser or console.



```

Hive history file=/tmp/hduser/hive_job_log_hduser_201502151531_849549347.txt
hive> select * from partner_poc;
OK
4G-1.bestbuyconnect.com NYC      1386    2015/02/04
4G-1.bestbuyconnect.com PIT     68660  2015/02/04
4G-1.bestbuyconnect.com RAL     92981  2015/02/04
cbeyond4g.net      ATL    13309167 2015/02/04
cbeyond4g.net      CHI    1053282 2015/02/04
cbeyond4g.net      HOU   1594748 2015/02/04
cbeyond4g.net      LOS    84455   2015/02/04
cbeyond4g.net      SFO    968     2015/02/04
mob.comcast.net   CHI    3127    2015/02/04
mob.comcast.net   KCY   146953  2015/02/04
mob.comcast.net   MIA   2988998 2015/02/04
mob.comcast.net   PHL   4589690 2015/02/04
mob.comcast.net   SEA    896     2015/02/04
mobile.rr.com     CLT   14474   2015/02/04
mobile.rr.com     DAL   54066   2015/02/04
mobile.rr.com     NYC   18545459 2015/02/04
mobile.rr.com     RAL   2428722 2015/02/04
sprintpcs.com     BAL   502444  2015/02/04
sprintpcs.com     CHI   13338   2015/02/04
sprintpcs.com     KCY   938055  2015/02/04
sprintpcs.com     NYC   99433844 2015/02/04
sprintpcs.com     PIT   16938828 2015/02/04

```



#### 4.5.2.3. Export the results Back to Mysql

After processing the data with hive we have to move the results back to MySQL for generating the reports .

First Execute the following scripts/queries in Mysql to create the tables for storing the coming result from sqoop export.

```
create table hourly_poc(date1 varchar(50),day_slot int,sub_count bigint,tot_bytes bigint );
```

```
create table partner_poc(Partner_name varchar(50),Market_name varchar(50),Tot_bytes bigint,Date1 varchar(50));
```

```
create table market_poc(Market_name varchar(50),Tot_bytes bigint,Date1 varchar(50));
```

```
create table partner_share_poc(Market_name
varchar(50),Market_bytes Bigint,Partner_name
varchar(50),Partner_bytes Bigint,Date1 varchar(50), partner_share_poc
```

To export the resultant tables we have to execute the following script in sqoop.

```
sqoop export --connect jdbc:mysql://127.0.0.1/mydb --table hourly_poc --export-dir /user/hive/warehouse/udr_db.db/hourly_poc
```

```
sqoop export --connect jdbc:mysql://127.0.0.1/mydb --table market_poc --export-dir /user/hive/warehouse/udr_db.db/market_poc
```

```
sqoop export --connect jdbc:mysql://127.0.0.1/mydb --table partner_poc --export-dir /user/hive/warehouse/udr_db.db/partner_poc
```

```
sqoop export --connect jdbc:mysql://127.0.0.1/mydb --table partner_share_poc --export-dir /user/hive/warehouse/udr_db.db/partner_share_poc
```

Now we have the processed results in our Mysql database .So next step is to generate the reports on that.

#### 4.2.5.4. Generating reports

Here first we need to connect from tableau to our database in mysql as follows.

1. Select Data > Connect to Data to open the Connect to Data page.
2. Select MySQL.
3. Follow the steps in the MySQL Connection dialog box to complete the connection.
  1. Step 1 – Type the name of the server that hosts the database.
  2. Step 2 – Enter your username and password to log on to the server.
  3. Step 3 – Establish the connection.

If the connection is unsuccessful, verify that your user name and password are correct. If the connection continues to fail, your computer



trouble locating the server. Contact your network administrator or database administrator.

4. Step 4 – Select the database that you want to connect to.
5. Step 5 – Select a table or view from the database.

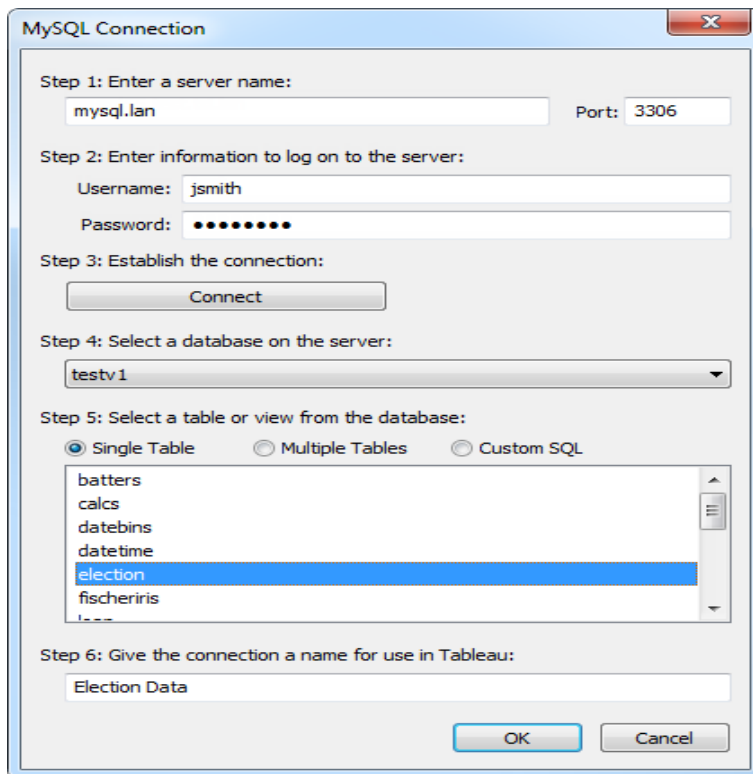
MySQL databases can contain multiple tables and views. Specify which table or view within the database you want to connect to.

Alternatively, you can connect to a set of tables that are related by join conditions. Select either Multiple Tables or Custom SQL when you are connecting to multiple tables. You can also add joins later.

6. Step 6 – Give the connection a name for use in Tableau.

Specify a unique name for the connection. A default name is automatically generated.

A completed Connection dialog box for MySQL is shown below.



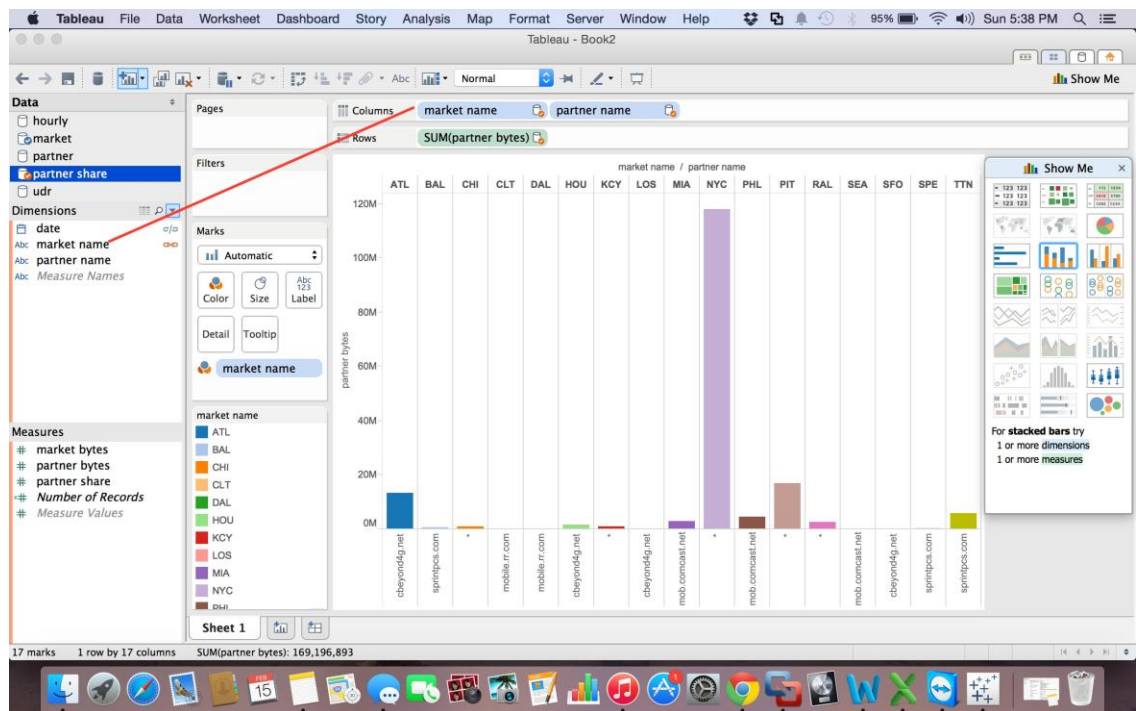
## Generating the reports and View

After you connect to data, fields are displayed on the left side of the workbook as Dimensions and Measures.

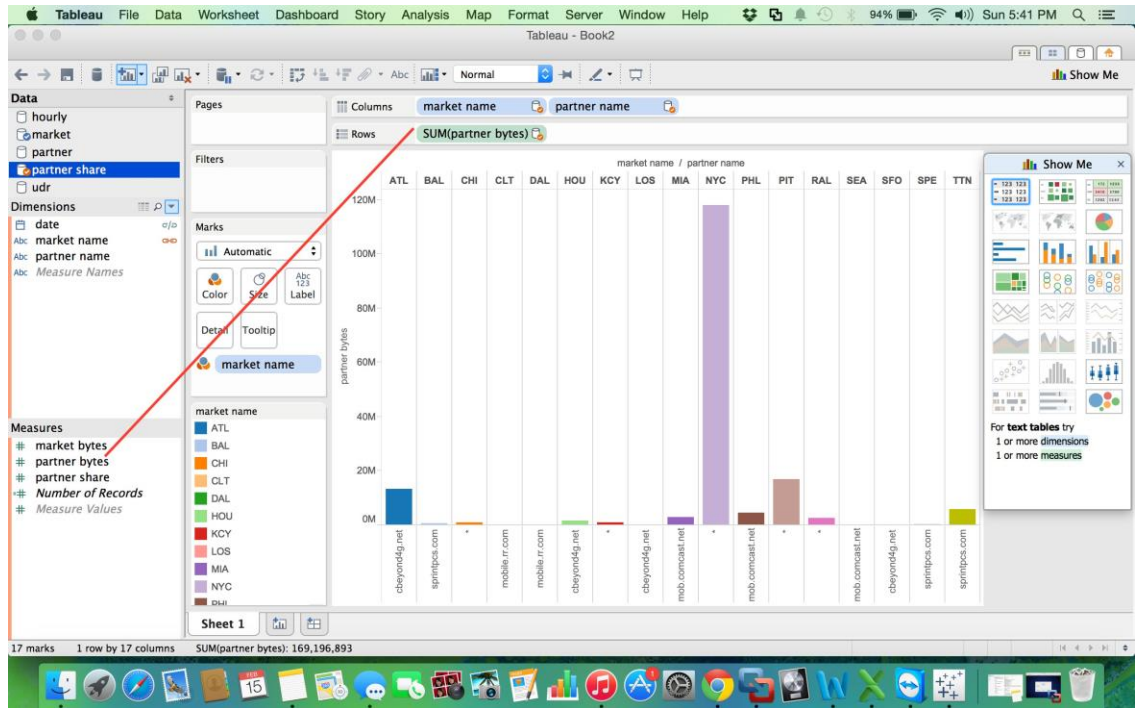
Create views by dragging and dropping fields onto shelves.

### Steps to be followed for generating each report

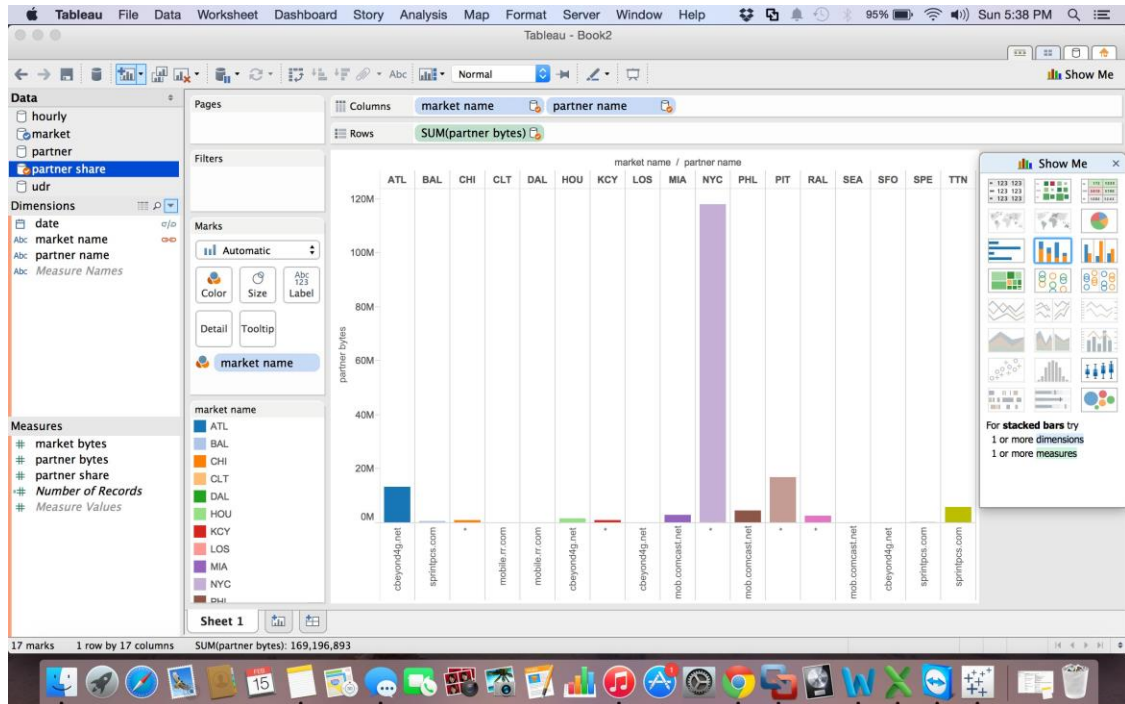
#### 1. Drag a dimension to the Columns shelf



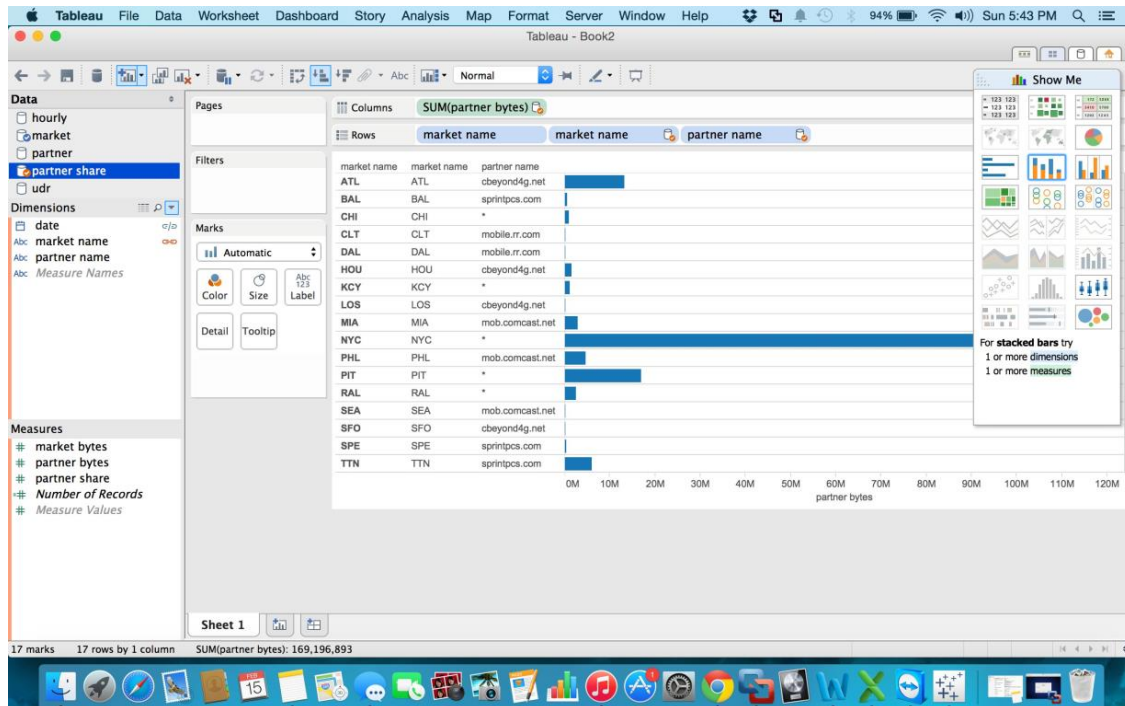
## 2. Drag a dimension to the Rows shelf



### 3. Drag a measure to Text



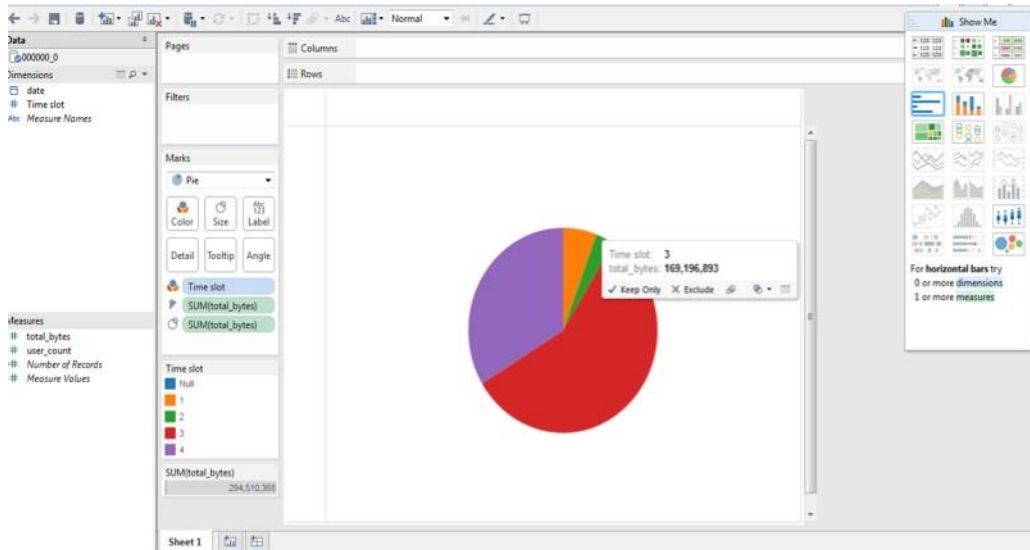
### 4. Visualize your data



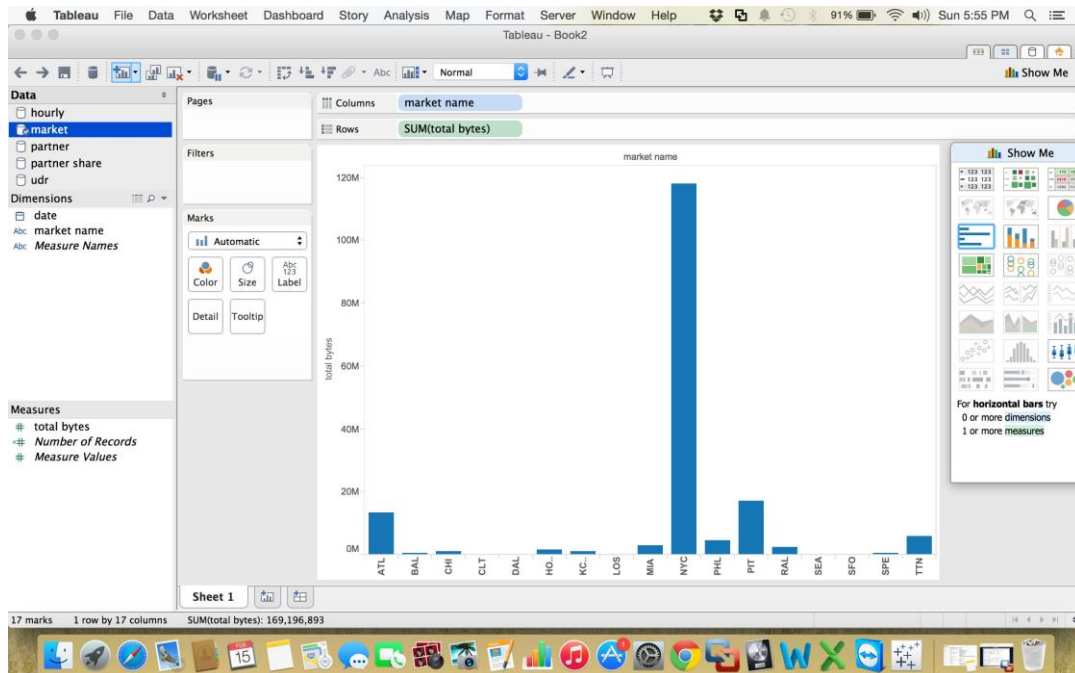
4. We can use color to show more information Drag the Region dimension from the Data window and drop it on Color.

## Output

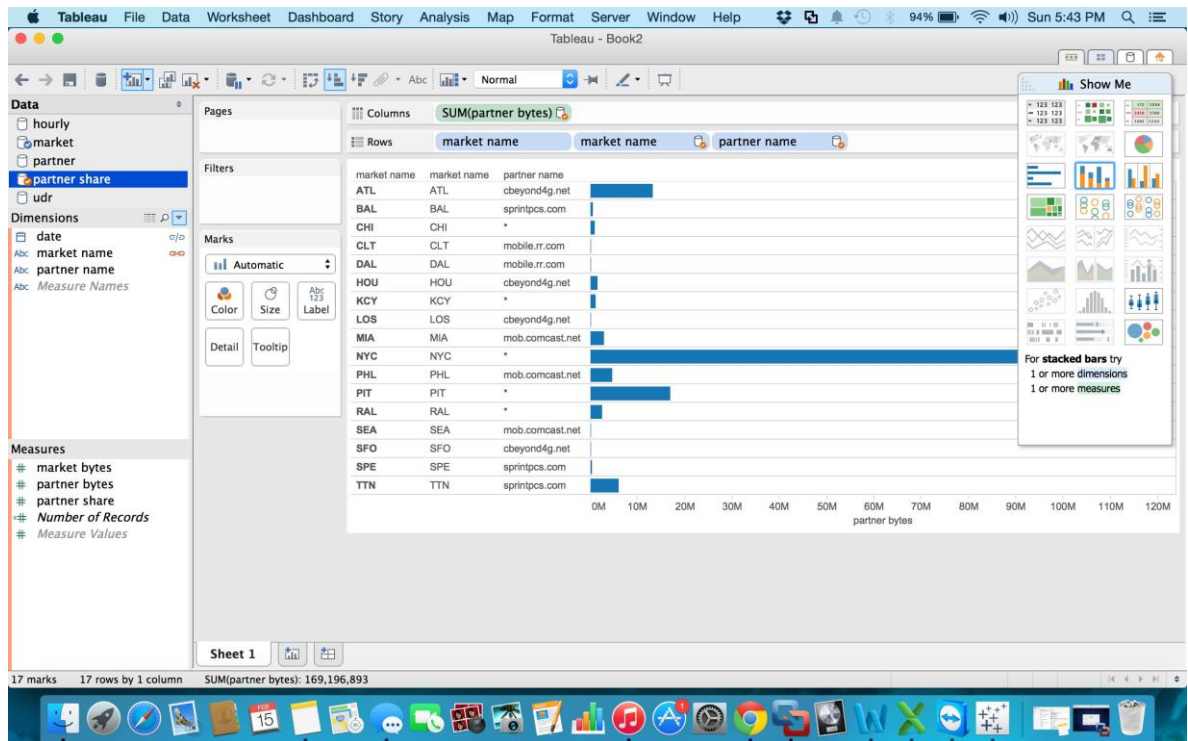
1. Daily user count and bytes transmitted on a particular time slot.



2. Area wise business (usage) share in the total business



3. Report to find out the areas of partner leading and try to improve the owner tower installations.



## 5. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### 5.1. TYPES OF TESTING

#### 5.1.1. UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program

branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### **5.1.2. BLACK BOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### **5.1.3. WHITE BOX TESTING**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

### **5.1.4. TEST STRATEGY AND APPROACH**

Field testing will be performed manually and functional tests will be written in detail.

## **5.2. TEST OBJECTIVES**

- All field entries must work properly.
- Pages must be activated from the identified link.



### 5.3. FEATURES TO BE TESTED

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

#### 5.3.1. INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components

#### 5.3.2. FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be

rejected. Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified



### **5.3.3. SYSTEM TESTING**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### **5.4. TESTS CONDUCTED**

Tests are conducted to check that every node in the cluster is actively working or not and the manually some test are conducted to check the output values

#### **Test Results**

All the test cases mentioned above passed successfully. No defects encountered.

## 6.

### CONCLUSION

We found the business insights of current user records data (i.e data cards usage records). And get the benefits for business growth. The parameters to be considered for analysis and gave them the results like Daily user count and bytes transmitted on a particular time slot, Area wise business(usage) share in the total business and Since every network owner will be depending on partners to get the service where they does not have the service tower. We solved the Problem Statement present in existing system in this project.

## 7. REFERENCES/BIBLIOGRAPHY

1. Agrawal, D., Das, S., & Abbadi, A. E. (2011). Big data and cloud computing. Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT 11.
2. Agrawal, D., Das, S., & Abbadi, A. E. (2010). Big data and cloud computing. Proceedings of the VLDB Endowment, 3(1-2), 1647–1648.
3. Aulbach, S., Jacobs, D., Kemper, A., & Seibold, M. (2009). A comparison of flexible schemas for software as a service. Proceedings of the 35th SIGMOD International Conference on Management of Data - SIGMOD 09.
4. “Understanding Hadoop Clusters & Networks.” Art of IT Infrastructures, November 17, 2013. <https://datacenternetworks.wordpress.com/2013/09/25/understanding-hadoop-clusters-networks/comment-page-1/>.
5. Data Modeling Hadoop. (n.d.). Retrieved from [http://docshare.tips/data-modeling-hadoop\\_590e5d46ee3435b2349941e7.html](http://docshare.tips/data-modeling-hadoop_590e5d46ee3435b2349941e7.html)
6. Data Modeling Hadoop. (n.d.). Retrieved April 15, 2020, from <https://es.scribd.com/document/260227682/Data-Modeling-Hadoop>
7. Bradley, C., Hollinshead, R., Kraus, S. D., Lefler, J., & Taheri, R. (1970, January 1). [PDF] Data Modeling Considerations in Hadoop and Hive: Semantic Scholar. Retrieved from <https://www.semanticscholar.org/paper/Data-Modeling-Considerations-in-Hadoop-and-Hive-Bradley-Hollinshead/79d4282345d8c4b770b406ed3767a54c39f1797a>