H1B VISA

BIG DATA - HADOOP

JEMIMAH.C| Professional Diploma in Digital Transformation – Big Data with Hadoop | 02.05.2017

Tools : Apache Hadoop Framework – HDFS, MapReduce, Hive, Pig, Sqoop, MySql and MS Excel for Data Visualization

S170030100609

NIIT Limited PAGE 1

H1B VISA

**A PROJECT REPORT**

***Submitted by***

**JEMIMAH.C**

***in the partial fulfillment for the award of the course***

***of***

**PROFESSIONAL DIPLOMA**

***in***

**BIG DATA WITH HADOOP**

**NIIT, CHENNAI**

**MAY 2017**

**ACKNOWLEDGEMENT**

I find immense pleasure to convey my sincere and grateful thanks to **NIIT** and the management for providing necessary facilities in carrying out this project.

I greatly indebted to my Tech Mentor **Ms. Santhanalakshmi,** the batch instructor **Mr. Annu** and the SLT faculty **Mr. Sandeep** for constant support throughout the course and also for useful suggestions, constant encouragement and kind advice in bringing out this project as a success.

I express my regards and sincere thanks to the Academic Leader **Ms. Annie.J** for providing all necessary resource to complete the project.

I extend my thanks to all staff members of NIIT for their kind co-operation for the completion of the project successfully. I am grateful to thank my family and all my friends for their valuable feedback, encouragement and suggestions.

**CHAPTER 1**

**INTRODUCTION**

# 

# **GENERAL**

Big data is an all-encompassing terms for any collection of data sets so large and complex that it becomes difficult to process them using traditional data processing applications. Extremely large data sets that may be analyzed computationally to re veal patterns, trends, and associations, especially relating to human behavior and interactions.

# **DATA MINING**

Data Mining is an analytic process designed to explore data (usually large amounts of data - typically business or market related - also known as "big data") in search of consistent patterns and/or systematic relationships between variables, and then to validate the findings by applying the detected patterns to new subsets of data. The ultimate goal of data mining is prediction - and predictive data mining is the most common type of data mining and one that has the most direct business applications.

The process of data mining consists of three stages: (1) The initial exploration, (2) model building or pattern identification with validation/verification, and (3) deployment (i.e., the application of the model to new data in order to generate predictions).

# **Need for Data Mining**

* Enormous volume of data is being handled in the field of Information technology on a day-to-day basis. A robust mechanism is needed to turn this huge volume of data into useful information.
* The information extracted by data mining serves as an input for various applications such as,
* Market Analysis
* Fraud Detection
* Customer Retention
* Production Control
* Science Exploration

# **Scope of Data Mining**

Data mining technology generates various new business opportunities by providing the following capabilities.

**1.2.2.1 Automated prediction of trends and behaviors**

Data mining automates the process of finding predictive information in large databases. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.

**1.2.2.2 Automated discovery of previously unknown patterns**

Data mining tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors.

# **Data Mining Tasks**

1. **Common Data Mining Tasks**

* Classification(Predictive)
* Clustering(Descriptive)
* Association Rule Discovery(Descriptive)
* Sequential Pattern Discovery(Descriptive)
* Regression(Predictive)
* Deviation Detection(Predictive)

1. **Prediction Task**

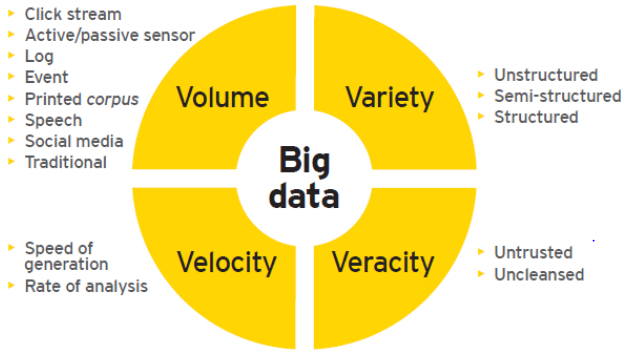
* It uses some variables to predict unknown or future values of other variables.

1. **Description Task**

* Finds human-interpretable patterns that describe the data.

# **BIG DATA**

Big Data and Big Compute provide unique capabilities for storing and processing large volumes of data. Big Data starts with large-volume, heterogeneous, autonomous sources with distributed and decentralized control, and seeks to explore complex and evolving relationships among data. One of the most important factor about Big Data is that it carries out computation on petabyte and even the exabyte. With the fast development of networking, data storage, and the data collection capacity, Big Data are now rapidly expanding in all science and engineering domains, including physical, biological and biomedical sciences. When dealing with big data there are many hurdles like data capture, storage, search, sharing, analytics, visualization etc.



**Figure 1.1** **Bigdata Characteristics**

Bigdata is characterized by the following 4V’s as shown in   
Figure 1.1

1. **Volume** - the vast amount of data generated every second that are larger than what the conventional relational database infrastructures can cope with.
2. **Velocity** - the frequency at which new data is generated, captured, and shared.
3. **Variety** - the increasingly different types of data (from financial data to social media feeds, from photos to sensor data, from video capture to voice recordings) that no longer fits into neat, easy to consume structures.
4. **Veracity** - the disarrayed data (Facebook posts with hash tags, abbreviations, typos, and colloquial speech)
   * 1. **Difference Between RDBMS and Bigdata**

The Table 1.1 summarizes the differences between Hadoop (Bigdata) and RDBMS in handling huge volume of data.

**Table 1.1 RDBMS Vs Big Data**

|  |  |  |
| --- | --- | --- |
| **Features** | **RDBMS** | **Hadoop** |
| Description | Traditional row-column databases used for transactional systems, reporting, and  archiving. | Distributed file system that stores large  amount of file data on a cloud of machines, handles data redundancy etc.  On top of that distributed file system, Hadoop provides an API for processing all that stored data - Map-Reduce. On top of this basic schema a Column Database, like HBase can be built. |
| Type of data supported | Works with structured data only | Works with structured, semi-structured, and unstructured data |
| Max data size | Terabytes | Hundreds of Petabytes |
| Limitations | Databases must slowly import data into a native representation before they can be queried, limiting their ability to handle streaming data. | Works well with streaming data |
| Read / write throughput limits | 1000s queries/second | Millions of queries per second |
| Data layout | Row-oriented | Column family oriented |

* + 1. **Applications of Bigdata**

Big data can be used in any industry or organization that handles massive volume of data. Listed below are some typical examples where Big data is applied.

* New queries
* Content ops
* Behavioral insights
* Predictive analysis
* Sense and respond
* New OLTP
* Mobile scale
* Risk

# **SUMMARY**

This chapter explains the concepts of Data Mining and Big Data, and the need of data analytics in healthcare for making smarter decisions.

**CHAPTER 3**

**SYSTEM MODULES**

# **BIG DATA**

Big data refers to large and complex datasets.

Its sizes range from few dozen terabytes to many petabytes of data in a single dataset.

Hard to capture, manage and process datasets using the on-hand database management tools. Difficulties processing Big Data include:

* Capturing
* Storing
* Searching
* Sharing
* Analytics on Big Data

Hadoop is a software framework for processing large amounts of data scattered across multiple commodity nodes.

The base hadoop environment contains HDFS and MapReduce piece.

# **HIVE**

Hive provides an SQL-like interface to hadoop.

Hive is a hadoop data warehouse system.

Using hive we can access data easily and create summarization.

We can carry out analysis of large data in hadoop using pig.

Hive is a top level apache project.

# **PIG**

Pig is a dataflow language written in java.

It allows users to specify explicit sequence of steps for data processing.

It internally chains multiple Map-Reduce jobs for executing data flows.

It implements Map-Reduce design patterns.

It supports relational model but not strongly typed.

It generates Map-Reduce program on execution for pig operators.

It optimizes Map-reduce execution for a given data flow.

Provides extensibility through User defined Functions.

It increases programmer productivity.

It minimizes the learning curve for new developers.

It reduces duplication of programming effort.

It opens the Map-Reduce programming power to more powers.

It eliminates the complexity of lengthy native java programs.

It insulates from hadoop version upgrades.

# **SQOOP**

Sqoop is an open source software product by apache hadoop foundation and part of ecosystem.

Collection of multiple command-line tools to move data between RDBMS and hadoop.

Imports are used to populate tables in hive or hbase, in addition to hdfs.

Exports can be used to put data from hadoop into an RDBMS.  
It uses MapReduce for data export and import providing parallel operation in addition to fault tolerance.

It uses JDBC drivers to connect to RDBMS.

# **SUMMARY**

This chapter deals with the proposed modules.

**CHAPTER 4**

**SYSTEM REQUIREMENTS**

# **GENERAL**

This chapter clearly depicts the software required to build the system and hardware required to host the system.

# **HARDWARE REQUIREMENTS**

Processor : Pentium IV

Clock speed : 550MHz

Hard Disk : 320GB

RAM : 2 GB

Cache Memory : 1 MB

Operating System : Windows XP/ Windows7

# **SOFTWARE REQUIREMENTS**

Operating System : Windows 7 / Ubuntu

Languages : Java SE, Hadoop 2.6.0

File System / DB : HDFS

Web Server : Tomcat 7.0

Tools : Eclipse Luna

# **JAVA SE**

# **Introduction**

Java is a computer programming language developed by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan and SUN Micro Systems Incorporation in 1991, now owned by Oracle. It enables programmers to write computer instructions using English based commands, instead of having to write in numeric codes. It’s known as a “high-level” language because it can be read and written easily by humans. Like English, Java has a set of rules that determine how the instructions are written. These rules are known as its “syntax”. Once a program has been written, the high-level instructions are translated into numeric codes that computers can understand and execute.

# **Advantages of Java**

* **Easy to Use:** The fundamentals of Java came from a programming language called C++. Although a powerful language, it was felt to be too complex in its syntax, and inadequate for all of Java’s requirements. Java built on, and improved the ideas of C++, to provide a programming language that was powerful and simple to use.
* **Reliability:** Java needed to reduce the likelihood of fatal errors from programmer mistakes. With this in mind, object-oriented programming was introduced. Once data and its manipulation were packaged together in one place, it increased Java’s robustness.
* **Platform Independent:** Programs needed to work regardless of the machine they were being executed on. Java was written to be a portable language that doesn’t care about the operating system or the hardware of the computer.
* **Support for Server Side Programming**: Java provides a high level of support to implement server side programming by JSP. It has a rich set of libraries when compared to other server side programming such as ASP, CGI.

# **TOOLS**

# **Eclipse Luna**

Eclipse is a multi-language Integrated development environment (IDE) comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java.

Eclipse Luna was released and it was the first simultaneous release that was built on Eclipse 4. Its previous release was Eclipse 3. Luna has come with major changes, particularly with Servlet 2.x support, the whole new look to the workbench, the ability to mix views and editors, universal search bars, detached editors. Luna is a combination of 62 projects with a combined total of 46 million lines of code. The workbench is represented as an EMF model and dynamically rendered. It completely detaches the presentation from the working logic which is a plug-able presentation engine. New projects in Luna include:

* Maven integration (M2E), which provides tight integration with the Maven development process
* Eclipse Scout, an enterprise framework and SDK for building distributed applications
* Jubula, an automated functional testing tool for Java and HTML-based applications.
* WindowBuilder, a drag-and-drop GUI editor for Java (SWT and Swing), which was donated by Google having purchased it off Instantiations
* Graphiti, a graphical tooling infrastructure for EMF based models
* EGit and JGit, which have been released as 1.0 products with Indigo's release

After Luna was released, Indigo, Kepler were released as a part of Eclipse development community. There are software about to be released with major changes. Comparing to Indigo, Luna supports well in handling XQUERY that is a core part of the logic implemented in our project.

# **HADOOP**

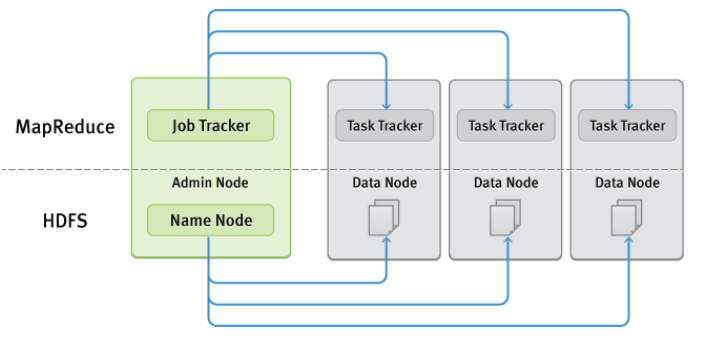
# **Introduction**

Hadoop is an open source implementation of MapReduce coded and released in Java by Apache. The Hadoop implementation of MapReduce uses the Hadoop Distributed File System (HDFS) as its underlying layer rather than GFS.

The Hadoop core is divided into two fundamental layers:

* Map Reduce Engine
* HDFS

The MapReduce engine is the computation engine running on top of HDFS as its data storage manager. The Figure 5.1 shows the Hadoop components.



**Figure 5.1 Hadoop Components**

# **Mapreduce Engine**

The topmost layer of Hadoop is the MapReduce engine that manages the data flow and control flow of MapReduce jobs over distributed computing systems. The MapReduce engine follows a master/slave architecture consisting of:

* Job Tracker (Master)
* Task Tracker (Slave)

**5.6.2.1 Job Tracker**

The Job Tracker is the service within Hadoop that farms out Map Reduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack. It manages the Map Reduce job over a cluster and is responsible for monitoring jobs and assigning tasks to Task Trackers.

* Client applications submit jobs to the Job tracker.
* The Job Tracker talks to the Name Node to determine the location of the data
* The Job Tracker locates Task Tracker nodes with available slots at or near the data
* The Job Tracker submits the work to the chosen Task Tracker nodes.
* The Task Tracker nodes are monitored. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different Task Tracker.
* A Task Tracker will notify the Job Tracker when a task fails. The Job Tracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may even blacklist the Task Tracker as unreliable.
* When the work is completed, the Job Tracker updates its status.
* Client applications can poll the Job Tracker for information.

The Job Tracker is a point of failure for the Hadoop Map Reduce service. If it goes down, all running jobs are halted.

**5.6.2.2 Task Tracker**

The Task Tracker manages the execution of the map and reduce tasks on a single computation node in the cluster. Each Task Tracker node has a number of simultaneous execution slots, each executing either a map or a reduce task. Slots are defined as the number of simultaneous threads supported by CPUs of the Task Tracker node.

# **HDFS**

HDFS is a distributed file system inspired by GFS that organizes files and stores their data on a distributed computing system. HDFS has a master/slave architecture including:

* Name Node (master)
* Data Node (slave)

**5.6.3.1 Name Node**

The Name Node is the centerpiece of an HDFS file system. It keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. Client applications talk to the Name Node whenever they wish to locate a file, or when they want to add/copy/move/delete a file. The Name Node responds the successful requests by returning a list of relevant Data Node servers where the data lives. Name Node builds metadata from the block report. Block report is sent by the Data Node. If the Name Node is down, then HDFS is down.

**5.6.3.2 Data Node**

Each slave machine in the cluster will host a Data Node daemon. NameNode does not store the actual data of the file system. Data Node is responsible for storing the data in the HDFS. Client applications can talk directly to a Data Node, once the Name Node has provided the location of the data. Data Node sends information to the Name Node about the files and blocks stored in that node and responds to the Name Node for all file system operations. Data Node instances can talk to each other.

# **SUMMARY**

This chapter explains the Hardware & Software specification to be used in this project. It also explains about the Java and Hadoop technologies that are used for the implementation of the system.

**CHAPTER 5**

**IMPLEMENTATION AND RESULTS**

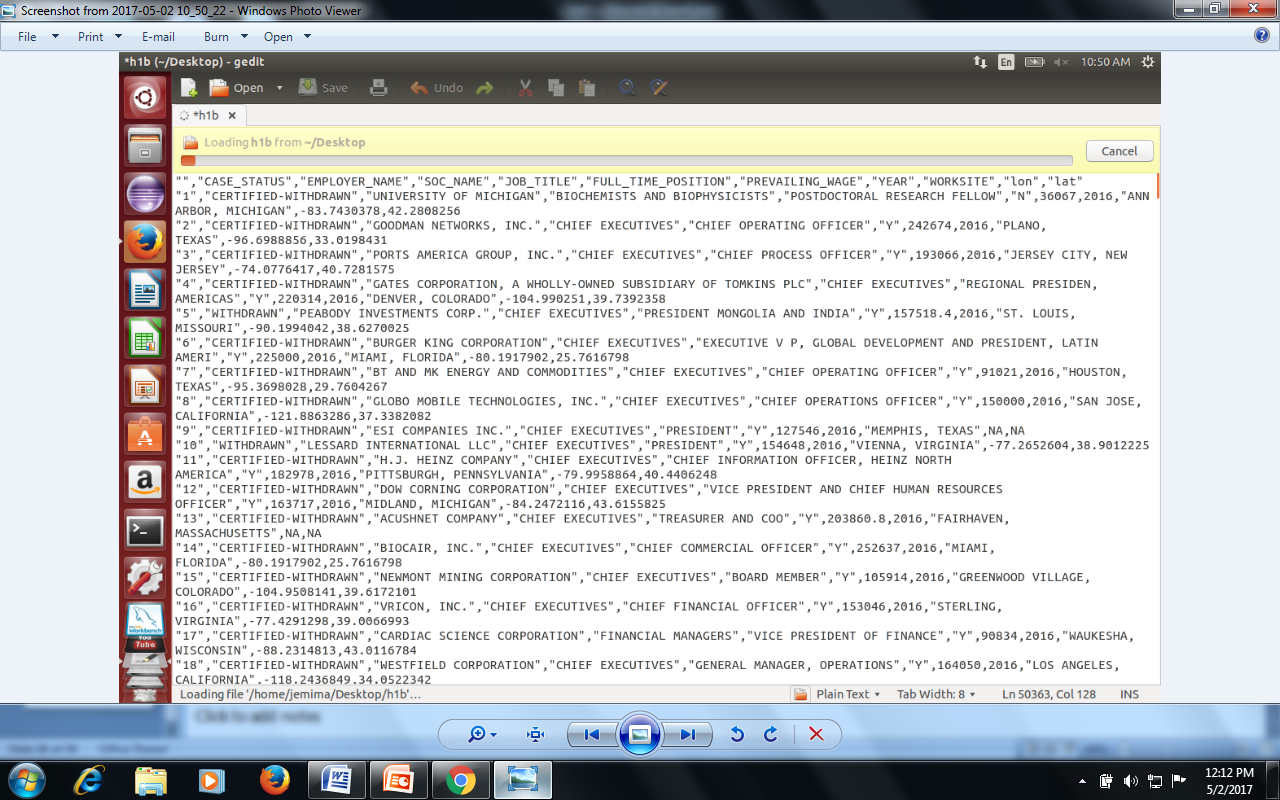
**6.1 GENERAL**

The execution results of Distributed Storage Configuration module is presented in this chapter.

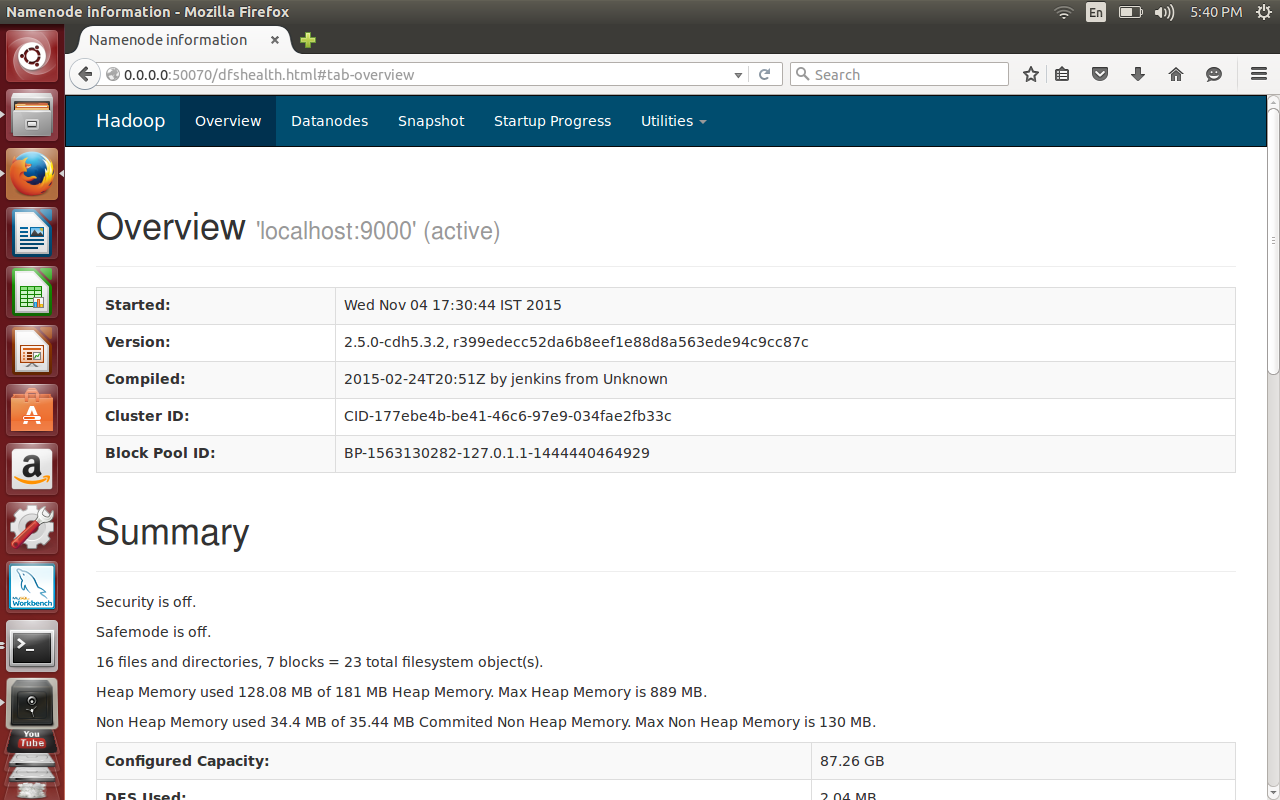
**6.2 RESULTS**

**6.2.1 Distributed Storage Configuration**

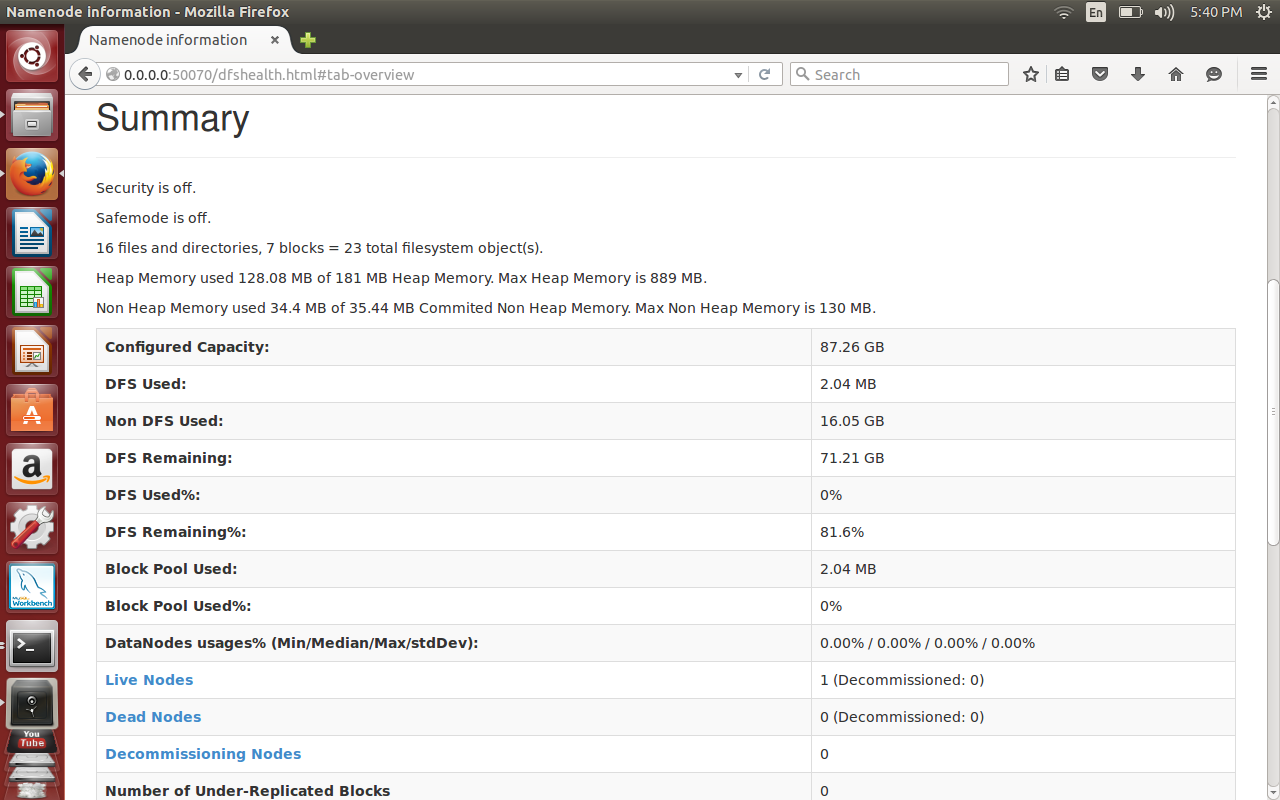
The data files gathered from hospital sources are stored in the Windows / Ubuntu file folders. The Distributed Storage Configuration module written in Java helps to copy the unstructured (video), semi-structured (XML) and structured (CSV) data files from Windows / Ubuntu folders into HDFS directories. This module throws a file selection window to choose the files that need to be copied into HDFS. This module allows the user to dynamically create a directory inside HDFS while copying the files. It will copy the files into an existing directory when the directory name entered by user already exists. The hadoop application must be up and running while executing this module otherwise connection exception will be invoked.

****

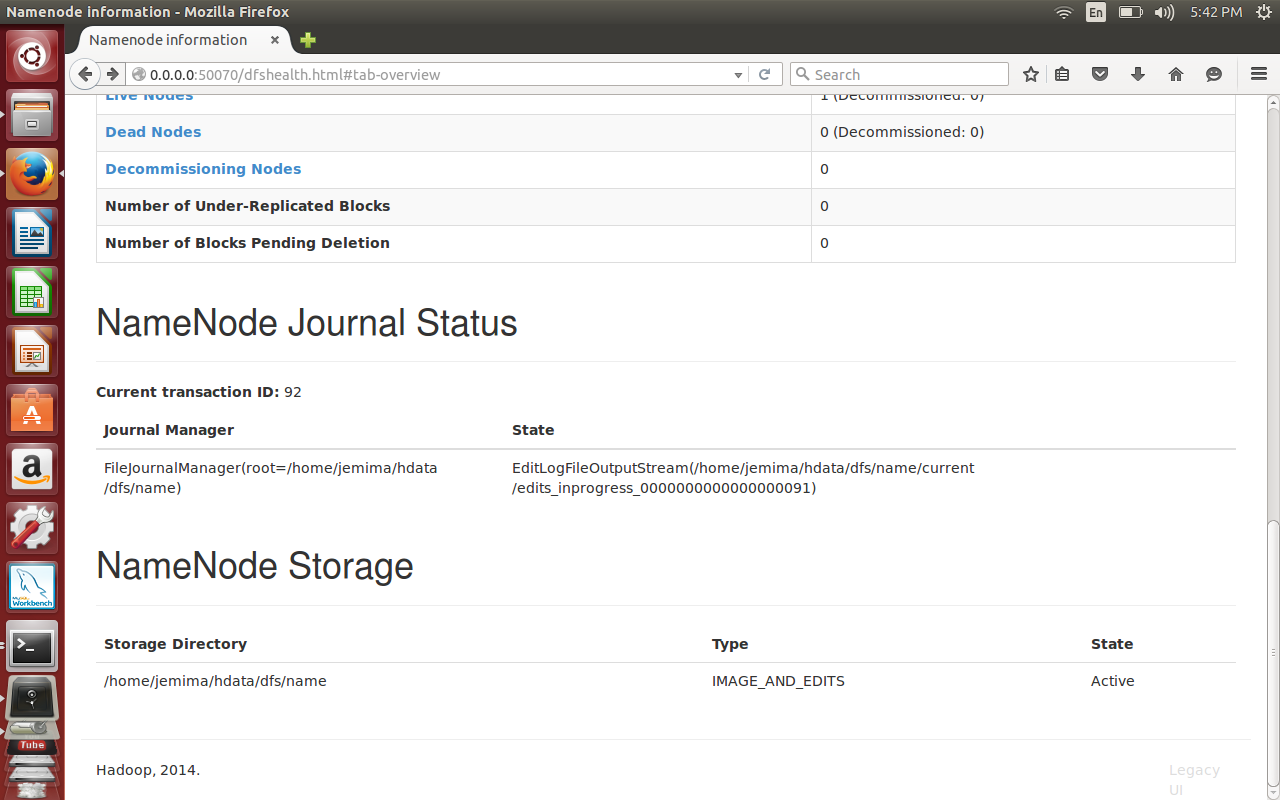
**Figure 6.1 Input dataset**

****

**Figure 6.8 Hadoop Home Page**

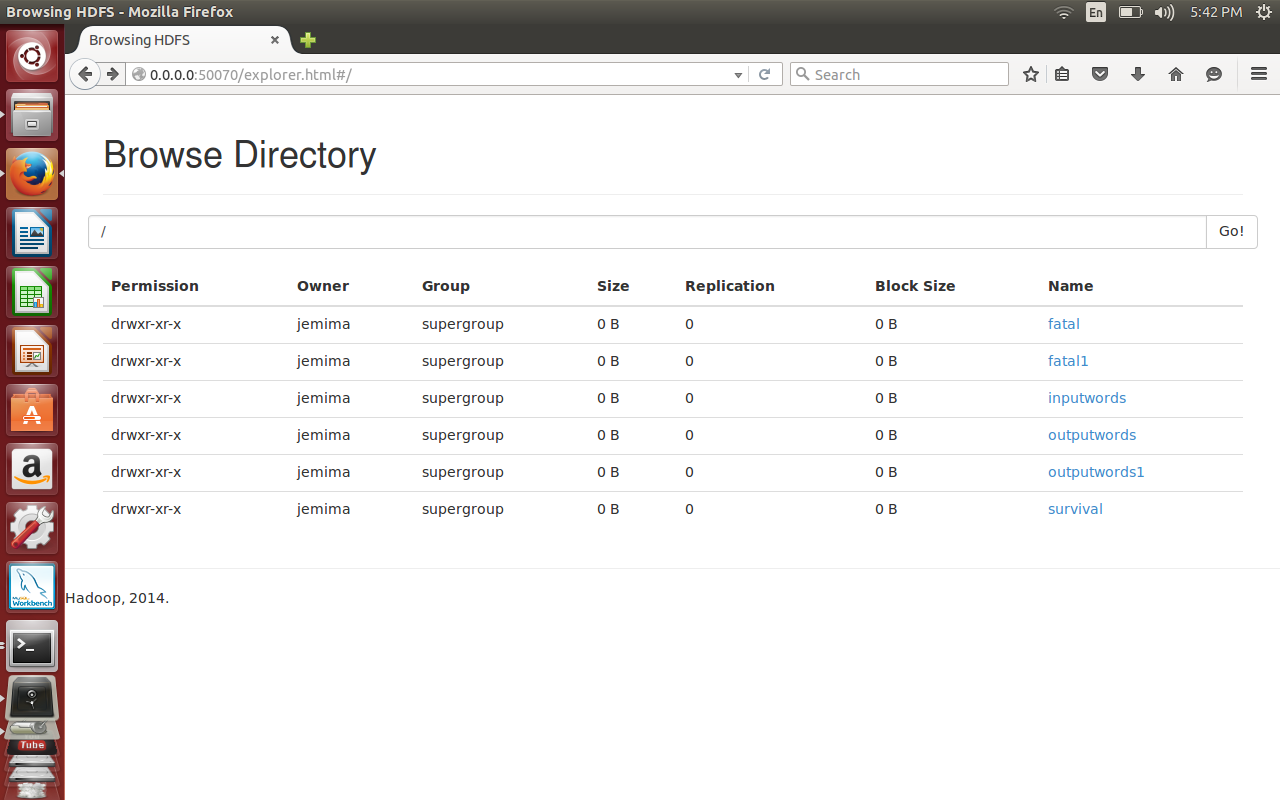
****

**Figure 6.9 Data Node Information**

****

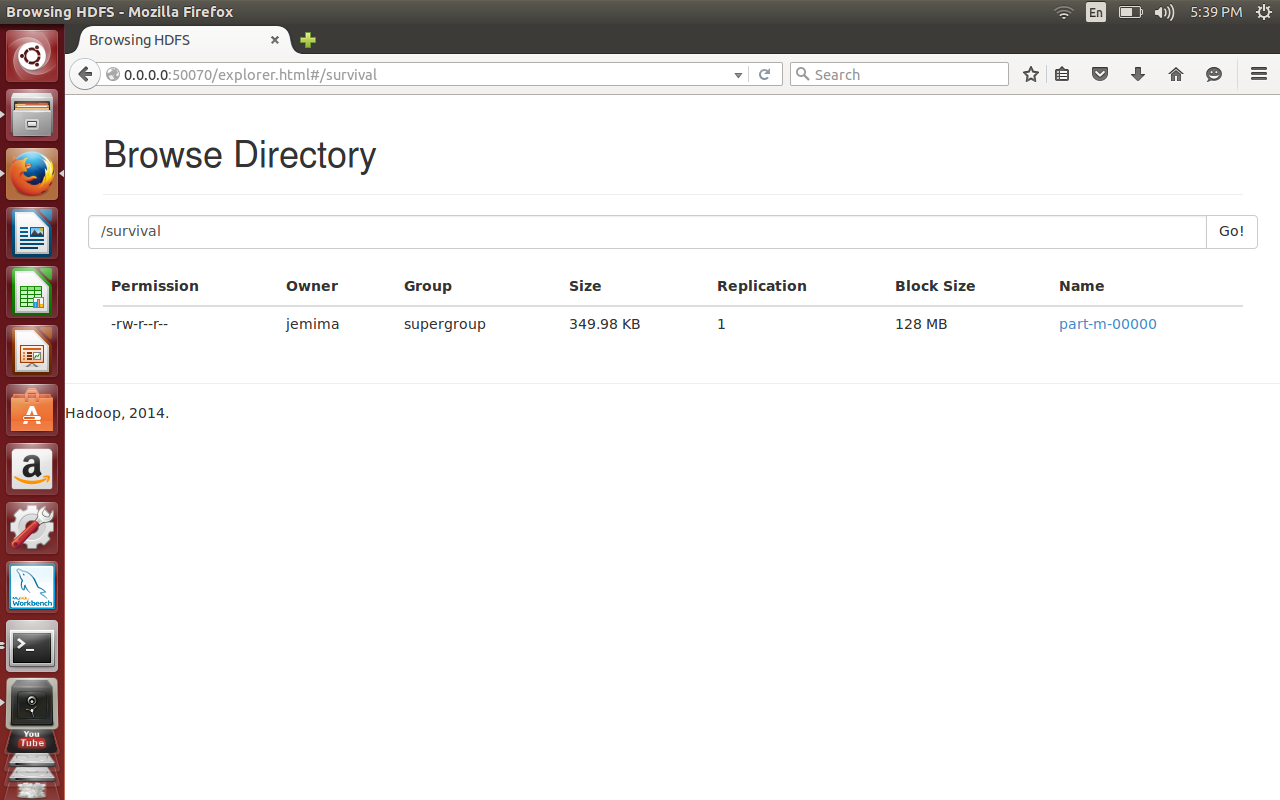
**Figure 6.10 Name node Information**

The following screenshot shows the web window of Name node running under hadoop file system. The user can validate the files copied into HDFS directory using the ‘Browse the Files system’ option.

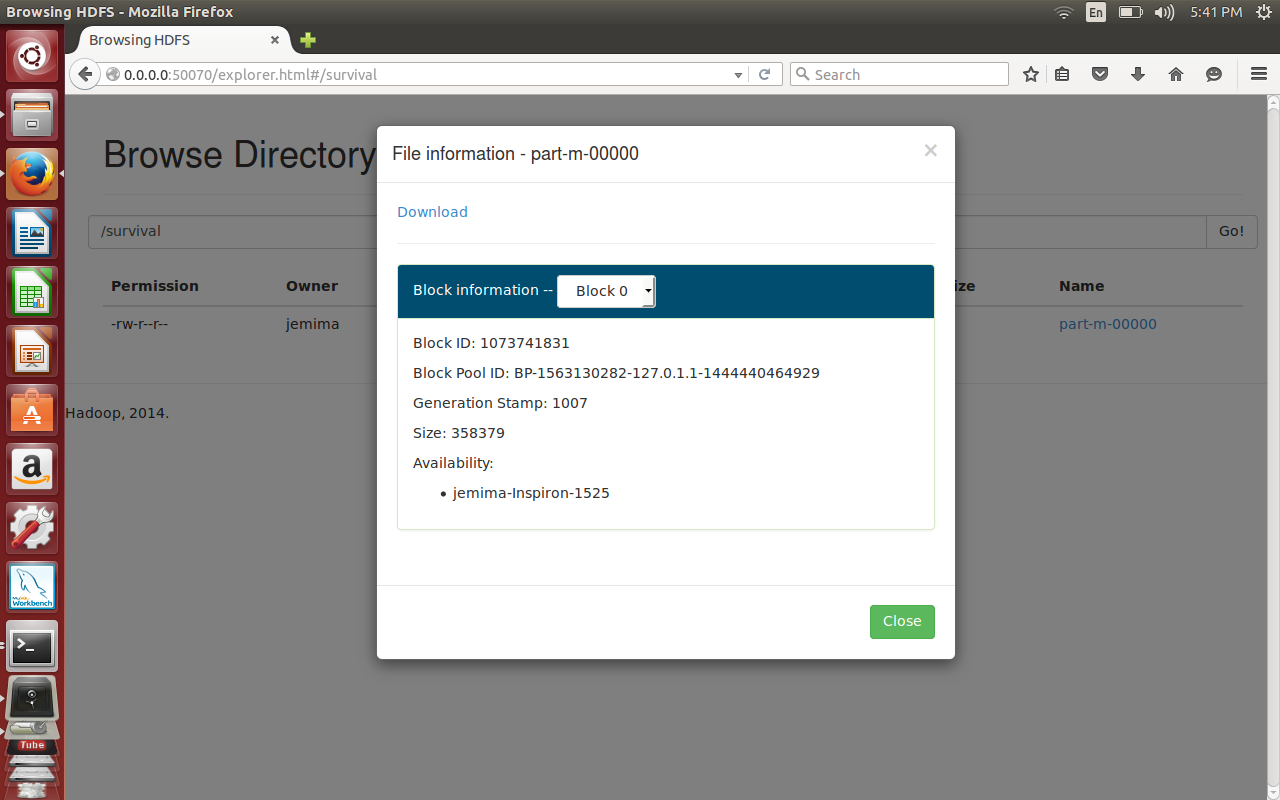
****

**Figure 6.11 Input Files Stored in Hadoop**

The files selected by user from Ubuntu file folder has been copied into HDFS as shown in the Figure 6.4. The web window also shows the file size, number of replication, HDFS block size, user and user permission information.

****

**Figure 6.12 Output File Executed by Hadoop**

****

**Figure 6.13 Output File Information**

**Figure 6.14 Output after the execution of the given Input in Hadoop**

**6.2.4 Performance Analyzer**

This module monitors the performance of the raw data with the new data. The performance is expected to be higher.

Performance = 1 / Execution Time

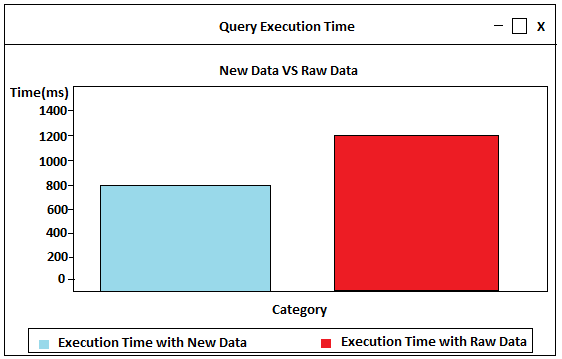
Performance ( Raw Data) = 1 / 1200 = 0.00083 per ms

Performance ( New Data) = 1 / 800 = 0.00125 per ms

Figure shows the time taken for execution between the raw data and the new data.

**Table 6.1 Execution Time**

|  |  |
| --- | --- |
| **Type of Data** | **Execution Time** |
| Raw Data | 1200 |
| New Data | 800 |

****

**Figure 6.15 Performance Graph**

**6.3 SUMMARY**

The sequence of tasks performed as part of Distributed Storage Configuration module has been explained in this chapter with the help of screenshots.

**CHAPTER 6**

**VALIDATION AND SYSTEM TESTING**

# **GENERAL**

# **System Testing**

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before the live operation commences. Testing is valid to the success of the system. An elaborate testing of the data is prepared and the system is tested using this test data. While testing, the errors are noted and corrections are made. The user is trained to operate the developed system. Both hardware and software securities are made to run the developed system successfully in the future.

# **LEVELS OF SOFTWARE TESTING**

* + Unit Testing
  + Integration Testing
  + System Testing
  + Acceptance Testing

# **Unit Testing**

Unit testing focuses on verification efforts on the smallest unit of the software design, the module. This is also known as “module testing”. The modules are tested separately. This is carried out during the programming stage. In this stage each module is found to be working satisfactorily as regarded to the expected output from the module.

# **Integration Testing**

Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

# **System Testing**

Once all the components are integrated, the application as a whole is tested rigorously to see that it meets quality standards. The application is tested thoroughly to verify that it meets the functional and technical specifications. System testing enables us to test, verify and validate both the business requirements as well as the applications architecture.

# **Acceptance Testing**

User acceptance testing of the system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with prospective system users at the time of developing. The testing of the software begins along with the coding.

# **TEST CASES**

A Test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. The mechanism for determining whether a software program or system has passed or failed such a test is known as test oracle. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. It may take many test cases to determine that a software program or system is functioning correctly. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites.

# **SUMMARY**

The software testing process and the test activities related to my project work are described in this chapter.

**CHAPTER 7**

**CONCLUSION AND FUTURE ENHANCEMENT**

# **CONCLUSION**

* Top job locations, job positions are found.
* Statistical analysis are depicted in graphs.
* The count of petitions can be found.
* The average wage for each job can be calculated.
* The top ten employers for who have the highest success rate can be detected.
* Similar steps can be for other data analysis also.

**FUTURE WORK**

Analysis could be made for the number of employers, the job positions the locations and the visa status**.**