1. INTRODUCTION

1.1 BIG DATA

“Big data” is a term used to describe a collection of data sets with the following three characteristics:

- Volume- Large amounts of data generated.
- Velocity-Frequency and speed of which data are generated, captured and shared
- Variety-Diversity of data types and formats from various sources.

The size and complexity of big data makes it difficult to use traditional database management and data processing tools. Data is being created in much shorter cycles from hours to milliseconds. There is also a trend underway to create larger databases by combining smaller data sets so that data correlations can be discovered.

BIG DATA ANALYTICS

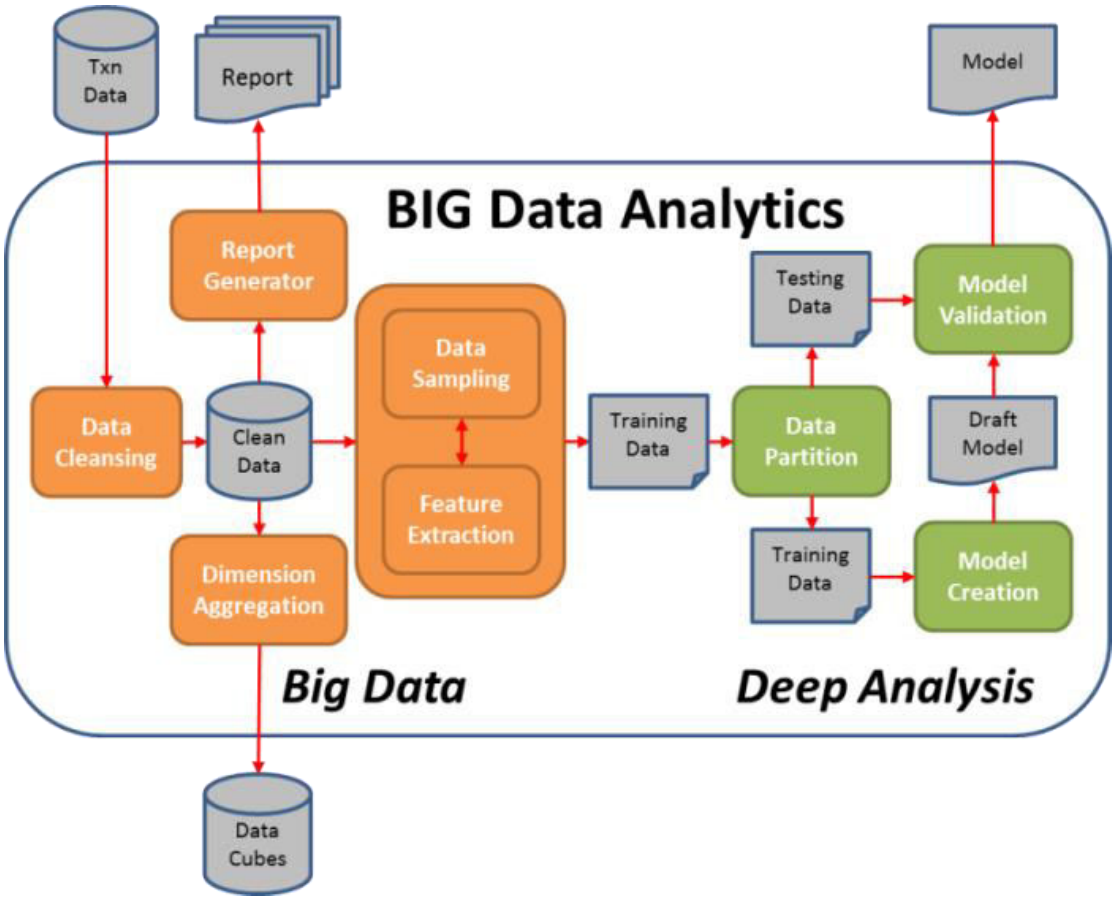
Big data has become the new frontier of information management given the amount of data today’s systems are generating and consuming. It has driven the need for technological infrastructure and tools that can capture, store, analyze and visualize vast amounts of disparate structured and unstructured data. These data are being generated at increasing volumes from data intensive technologies including, but not limited to, the use of the Internet for activities such as accesses to information, social networking, mobile computing and commerce. Corporations and governments have begun to recognize that there are unexploited opportunities to improve their enterprises that can be discovered from these data.

Analytics when applied in the context of big data is the process of examining large amounts of data, from a variety of data sources and in different formats, to deliver insights that can enable decisions in real or near real time. Various analytical concepts such as data mining, natural language processing, artificial intelligence and predictive analytics can be employed to analyze, contextualize and visualize the data. Big data analytical approaches can be employed to recognize inherent patterns, correlations and anomalies which can be discovered as a result of integrating vast amounts of data from different data sets.

Big data analytics requires the use of new frameworks, technologies and processes to manage it. Yet its arrival in the enterprise software space has created some confusion as business leaders try to understand the differences between it and traditional data warehousing (DW) and business intelligence (BI) tools.

There are important distinctions and sufficient differentiating value between BDA and DW/BI systems which make BDA unique.

Gartner defines a data warehouse as "a storage architecture designed to hold data extracted from transaction systems, operational data stores and external sources. The warehouse then combines that data in an aggregate, summary form suitable for enterprise-wide data analysis and reporting for predefined business needs."



Forrester Research has defined business intelligence as “a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical, and operational insights and decision-making."

BDA solutions will not replace DW/BI, rather they will co-exist side-by-side to unlock hidden value in the massive amount of data that exists within and outside the enterprise.

BDA functions are unique because they:

- Handle open ended "how and why" type questions whereas BI tools are designed to query specific "what and where".

- Process unstructured data to find patterns, whereas DW systems process structured and mostly aggregated data.

BIG DATA ANALYTICS

The term “Analytics” refers to the logic and algorithms, both deduction and inference, performed on BD to derive value, insights and knowledge from it. Analytical methods such as data mining, natural language processing, artificial intelligence and predictive analytics are employed to analyze, contextualize and visualize the data. These computerized analytical methods recognize inherent patterns, correlations and anomalies which are discovered as a result of integrating vast amounts of data from different datasets. Together, the term “Big Data Analytics” represents, across all industries, new data-driven insights which are being used for competitive advantage over peer organizations to more effectively market products and services to targeted consumers. Examples include real-time purchasing patterns and recommendations back to consumers, and gaining better understandings and insights into consumer preferences and perspectives through affinity to certain social groups. Gartner defines a data warehouse as “a storage architecture designed to hold data extracted from transaction systems, operational data stores and external sources. The warehouse then combines that data in an aggregate, summary form suitable for enterprise-wide data analysis and reporting for predefined business needs.”

The origin of BDA comes from web-based search engines such as Google and Yahoo, the popularity of social media and social networking services such as Facebook and Twitter, and data-generating sensors, telehealth and mobile devices. All have increased and generated new data and opportunities for new insights on customer behaviours and trends. While BDA frameworks have been in operation since 2005, they have just recently moved into other industries and sectors including financial services firms and banks, online retailers and health care.

BIG DATA COMPUTING

The rising importance of big-data computing stems from advances in many different technologies. Sensors: Digital data are being generated by many different sources, including digital imagers (telescopes, video cameras, MRI machines), chemical and biological sensors), and even the millions of individuals and organizations generating web pages. Computer networks: Data from the many different sources can be collected into massive data sets via localized sensor networks, as well as the Internet.

Data storage: Advances in magnetic disk technology have dramatically decreased the cost of storing data. For example, a one-terabyte disk drive, holding one trillion bytes of data, costs around $100. As a reference, it is estimated that if all of the text in all of the books in the Library of Congress could be converted to digital form, it would add up to only around 20 terabytes.
Cluster computer systems: A new form of computer systems, consisting of thousands of "nodes," each having several processors and disks, connected by high-speed local-area networks, has become the chosen hardware configuration for data-intensive computing systems. These clusters provide both the storage capacity for large data sets, and the computing power to organize the data, to analyze it, and to respond to queries about the data from remote users. Compared with traditional high-performance computing (e.g., supercomputers), where the focus is on maximizing the raw computing power of a system, cluster computers are designed to maximize the reliability and efficiency with which they can manage and analyze very large data sets. The "trick" is in the software algorithms – cluster computer systems are composed of huge numbers of cheap

commodity hardware parts, with scalability, reliability, and programmability achieved by new software paradigms.

Cloud computing facilities: The rise of large data centers and cluster computers has created a new business model, where businesses and individuals can rent storage and computing capacity, rather than making the large capital investments needed to construct and provision large-scale computer installations. For example, Amazon Web Services (AWS) provides both network-accessible storage priced by the gigabyte-month and computing cycles priced by the CPU-hour. Just as few organizations operate their own power plants, we can foresee an era where data storage and computing become utilities that are ubiquitously available.

BIG DATA ANALYTICS IN CLOUD ENVIRONMENT

Most corporate enterprises face significant challenges in fully leveraging their data. Frequently, data is locked away in multiple databases and processing systems throughout the enterprise, and the questions customers and analysts ask require an aggregate view of all data, sometimes totaling hundreds of terabytes.

Cerri et al proposed 'Knowledge in the cloud' in place of 'data in the cloud' to support collaborative tasks which are computationally intensive and facilitate distributed, heterogeneous knowledge. This is termed as "Utility Computing" derived from required data in and out of Cloud the utilities like electricity, gas for which we only pay for what we use from a shared resource. With the growing interest in cloud, analytics is a challenging task.In general, Business Intelligence applications such as image processing, web searches, understanding customers and their buying habits, supply chains and ranking and Bio-informatics (e.g. gene structure prediction) are data intensive applications. Cloud can be a perfect match for handling such analytical services. For example, Google's Map Reduce can be leveraged for analytics as it intelligently chunks the data into smaller storage units and distributes the computation among low-cost processing units. Several research teams have started working on creating Analytic frameworks and engines which help them provide Analytics as a Service. For example, Zementis launched the ADAPA predictive analytics decision engine on Amazon EC2, allowing its users to deploy, integrate, and execute statistical scoring models like neural networks, support vector machine (SVM), decision tree, and various regression models.

Booz Allen’s IT professionals, equipped with extensive expertise in the application of cloud computing technology has described a way for setting a course for mastering your big data. Cloud technology combines the best practices of virtualization, grid computing, utility computing, and web technologies. The result is a technology that inherits the agility of virtualization, the scalability of grid computing, and simplicity of Web 2.0. Cloud computing is an evolutionary step in computing that unifies the resources of many computers to function as one entity, allowing the construction of massively scalable systems that can take in and store, process and analyze all of your enterprise’s data. The definitive application of cloud technology is as a large-scale data storage, development and processing system, allowing your enterprise to master big data. But the agility of cloud computing has applications beyond effective use of data. Because all data is now maintained in a centralized system, we can help develop and implement a centralized security policy that can be easily enforced, allowing precise and well-documented control of sensitive data. In addition, the cloud provides an environment in which to prototype, test, and deploy new applications in a fraction of the time and cost of traditional systems.

The benefits continue to accrue as your “cloud” grows. As more datasets are aggregated, the cloud gains a critical mass of data across an enterprise, becoming “the place” to put data. As each dataset is added, and potentially analyzed with the other datasets, there is an exponential increase in benefit to the enterprise. We can enable your enterprise with simplified programming and data models, which, combined with easy access to a wide range of data, results in an explosion of innovation from across your enterprise in the form of data mashups, data-mining applications. Few decades back, the problem was the shortage in information or data. In recent past, this problem has been overcome with the advent of Internet and reduced Storage Memory cost. But a new challenge is how to analyse the data. Data is getting generated at a much faster pace than the speed at which it can be processed with the current infrastructure. Huge and dedicated servers were developed to solve this problem. But the problem is with the cost of such an infrastructure which is not affordable to all the companies for Availability of data and accessing each and every specific purpose. So today, these companies are looking it is the key success factor for Cloud computing which makes feasible for all these companies to hire on a temporary basis, the computational power and storage space value-based analytics. Migrating for a specific purpose.

MOVING BIG DATA INTO CLOUD

Big Data is a data analysis methodology enabled by recent advances in technologies and architecture. However, big data entails a huge commitment of hardware and processing resources, making adoption costs of big data technology prohibitive to small and medium sized businesses. Cloud computing offers the promise of big data implementation to small and medium sized businesses.

Big Data processing is performed through a programming paradigm known as MapReduce. Typically, implementation of the Map Reduce paradigm requires networked attached storage and parallel processing. The computing needs of Map Reduce programming are often beyond what small and medium sized business are able to commit.

The defacto approach of hard drive shipping is not flexible or secure. This work studies timely, cost-minimizing upload of massive, dynamically-generated, geo-dispersed data into the cloud, for processing using a Map Reduce-like framework. Targeting at a cloud encompassing disparate data centers, we model a cost-minimizing data migration problem, and propose two online algorithms: an online lazy migration (OLM) algorithm and a randomized fixed horizon control (RFHC) algorithm, for optimizing at any given time the choice of the data center for data aggregation and processing, as well as the routes for transmitting data there. Careful comparisons among these online and offline algorithms in realistic settings are conducted through extensive experiments, which demonstrate close-to-offline-optimum performance of the online algorithms.

KEY TECHNOLOGIES FOR EXTRACTING BUSINESS VALUE FROM BIG DATA

Big data technologies describe a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data by enabling high-velocity capture, discovery and/or analysis. Storage and processing technologies have been designed specifically for large data volumes. Computing models such as parallel processing, clustering, virtualization, grid environments and cloud computing, coupled with high-speed connectivity, have redefined what is possible. Here are three key technologies that can help you get a handle on big data – and even more importantly, extract meaningful business value from it. Information management for big data: Manage data as a strategic, core asset, with ongoing process control for big data analytics. High-performance analytics for big data: Gain rapid insights from big data and the ability to solve increasingly complex problems using more data. Big data will also intensify the need for data quality and governance, for embedding analytics into operational systems, and for issues of security, privacy and regulatory compliance. Everything that was problematic before will just grow larger. Unified data management capabilities, including data governance, data integration, data quality and metadata management. Complete analytics management, including model management, model deployment, monitoring and governance of the analytics information asset. Effective decision management capabilities to easily embed information and analytical results directly into business processes while managing the necessary business rules, workflow and event logic.

1.2 EXISTING SYSTEM

- A naive approach to find the optimum Hadoop configuration is to try every combination of configuration parameter values and choose the best one. Unfortunately, this is unrealistic because of the huge number of Hadoop configuration parameter combinations.
- To make things even worse, every run of a Hadoop application with a large input data set takes a considerable amount of time, leading to impractically long times if one were to explore the huge Hadoop configuration parameter optimization space exhaustively.

- Performance models can predict the performance of an application with a given configuration much faster than an approach that requires executing the application.
- Herodotou et al. propose fine-grained analytical models to predict the execution time of each phase; putting together per-phase predictions then yields an estimate for the total execution time.
- Lama and Zhou build a support vector machines (SVM) based model to predict the performance of Hadoop jobs.

DISADVANTAGES OF EXISTING SYSTEM:

- These analytical performance models are based on oversimplified assumptions, which limits overall performance.
- Previously proposed analytical models typically assume the execution time of per-byte or per-record processing to be constant, even as Hadoop configurations change, statistical models such as linear regression algorithms typically assume that the relationship between configuration parameters is linear.

1.3 PROPOSED SYSTEM:

- In this project, we propose a novel approach based on random-forest learning, which we call RFHOC, to predict the performance of an application on a given cluster with a given Hadoop configuration.
- Strictly speaking, random-forest is not a machine learning algorithm; instead, it is a robust ensemble model that combines the advantages of statistical reasoning and machine learning approaches.
- We consider a number of observations from a real Hadoop systemto train an ensemble model for each phase via random forest learning. The model takes Hadoop configurations as input and outputs a performance prediction. In a subsequent step, we then use the performance prediction models for each phase as part of a genetic algorithm (GA) to search for the optimum Hadoop configuration for the application of interest.

ADVANTAGES OF PROPOSED SYSTEM:

- RFHOC does not make any assumptions on the cost (execution time) of per-byte or per-record processing and the relationship between configuration parameters.
- RFHOC on the other hand does not make these simplifying assumptions and recognizes that Hadoop configuration parameters interact with each other in complex non-linear ways.
- Second, random-forest learning makes predictions based on a set of regression or classification trees, rather than a single tree, which makes the performance prediction not only accurate, but also stable and robust when deployed on previously unseen data sets.

2. LITERATURE SURVEY

1)Heuristic artificial intelligent algorithm for genetic algorithm

AUTHORS: L. Lie

A genetic algorithm is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

2)Jaql: A scripting language for large scale semi structured data analysis

AUTHORS: K. S. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Eltabakh,

C.-C. Kanne, F. Ozcan, and E. J. Shekita

This paper describes Jaql, a declarative scripting language for analyzing large semistructured datasets in parallel using Hadoop's MapReduce framework. Jaql is currently used in IBM's InfoSphere BigInsights [5] and Cognos Consumer Insight [9] products. Jaql's design features are: (1) a flexible data model, (2) reusability, (3) varying levels of abstraction, and (4) scalability. Jaql's data model is inspired by JSON and can be used to represent datasets that vary from flat, relational tables to collections of semistructured documents. A Jaql script can start without any schema and evolve over time from a partial to a rigid schema. Reusability is provided through the use of higher-order functions and by packaging related functions into modules. Most Jaql scripts work at a high level of abstraction for concise specification of logical operations (e.g., join), but Jaql's notion of physical transparency also provides a lower level of abstraction if necessary. This allows users to pin down the evaluation plan of a script for greater control or even add new Koperators. The Jaql compiler automatically rewrites Jaql scripts so they can run in parallel on Hadoop. In addition to describing Jaql's design, we present the results of scale-up experiments on Hadoop running Jaql scripts for intranet data analysis and log processing.

3) An integrated home financial investment learning environment applying cloud computing in social network analysis

AUTHORS: M.-P. Wen, H.-Y. Lin, A.-P. Chen, and C. Yang

This paper tried to apply cloud computing technology in social network analysis for a comprehensive home financial learning environment that individual investors may use as a reference in establishing web-based learning and investment platforms. The major contributions were described in three parts. First, this paper advanced the social network analysis technology to be able to handle millions of nodes and links. Second, we demonstrate how cloud computing can be applied to advanced computing in social network. Third, we performed several intelligent analyses on a very popular social network, IHFILE, to identify some interesting and important features of it. In addition to analyzing a homogeneous social network such as IHFILE, we also propose direction of how cloud computing can be performed on a social network analysis as our future work.

4) Starfish: A self-tuning system for big data analytics

AUTHORS: H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin,S. Babu

Timely and cost-effective analytics over “Big Data ” is now a key ingredient for success in many businesses, scientific and engineering disciplines, and government endeavors. The Hadoop software stack—which consists of an extensible MapReduce execution engine, pluggable distributed storage engines, and a range of procedural to declarative interfaces is a popular choice for big data analytics. Most practitioners of big data analytics like computational scientists, systems researchers, and business analysts lack the expertise to tune the system to get good performance. Unfortunately, Hadoop’s performance out of the box leaves much to be desired, leading to suboptimal use of resources, time, and money (in payas-you-go clouds). We introduce Starfish, a self-tuning system for big data analytics. Starfish builds on Hadoop while adapting to user needs and system workloads to provide good performance automatically, without any need for users to understand and manipulate the many tuning knobs in Hadoop. While Starfish's system architecture is guided by work on self-tuning database systems, we discuss how new analysis practices over big data pose new challenges; leading us to different design choices Starfish.

5) Profiling, What-if analysis, and cost-based optimization of MapReduce programs

AUTHORS: H. Herodotou and S. Babu

MapReduce has emerged as a viable competitor to database systems in big data analytics. MapReduce programs are being written for a wide variety of application domains including business data processing, text analysis, natural language processing, Web graph and social network analysis, and computational science. However, MapReduce systems lack a feature that has been key to the historical success of database systems, namely, cost-based optimization. A major challenge here is that, to the MapReduce system, a program consists of black-box map and reduce functions written in some programming language like C++, Java, Python, or Ruby. We introduce, to our knowledge, the first Cost-based Optimizer for simple to arbitrarily complex MapReduce programs. We also introduce a Profiler to collect detailed statistical information from unmodified MapReduce programs, and a What-if Engine for fine-grained cost estimation.The effectiveness of each component is demonstrated through a comprehensive evaluation using representative MapReduce programs from various application domains.

3. SYSTEM ANALYSIS

3.1 FESABILITY 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

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.

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.

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.

3.2 SOFTWARE REQUIREMENTS SPECIFICATIONS:

- The purpose of Software Requirements Specifications is to reduce the communication gap between the clients and the developers.
- A good SRS should satisfy all the parties involved in the system.
- Having a clear distinction between the needs of the clients, any software can be developed which will satisfy the client's requirement meeting his specifications.

Functional requirements:

Functional requirements describe what the system should do, i.e. the sevices provided for the users and for the other systems.

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 follow

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.

Non-Functional Requirements:

Scalability:

Scalability, is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner or to be readily enlarged.

Reliability:

Reliability (systemic def.) is the ability of a person or system to perform and maintain its functions in routine circumstances, as well as hostile or unexpected circumstances

Integrity:

Integrity as a concept has to do with perceived consistency of actions, values, methods, measures, principles, expectations and outcome.

Extensibility:

The System should be extendible with other application techniques

Reusability:

The System should be 60% reusable so as to be applied for other datasets.

3.3 SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

• System	:	i3 Processor
• Hard Disk	:	500 GB.
• Monitor	:	15'' LED
• Input Devices	:	Keyboard, Mouse
• Ram	:	4GB.

SOFTWARE REQUIREMENTS:

• Operating system	:	Windows 7/UBUNTU.
• Coding Language	:	Java 1.7 ,Hadoop 0.8.1 (for Maper and Reducer)
• Back End	:	Hadoop Cluster
• Tool	:	Virtual Box Oracle tool
• Evaluation	:	PHP, Javascript (Intelligent Graph)

4. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

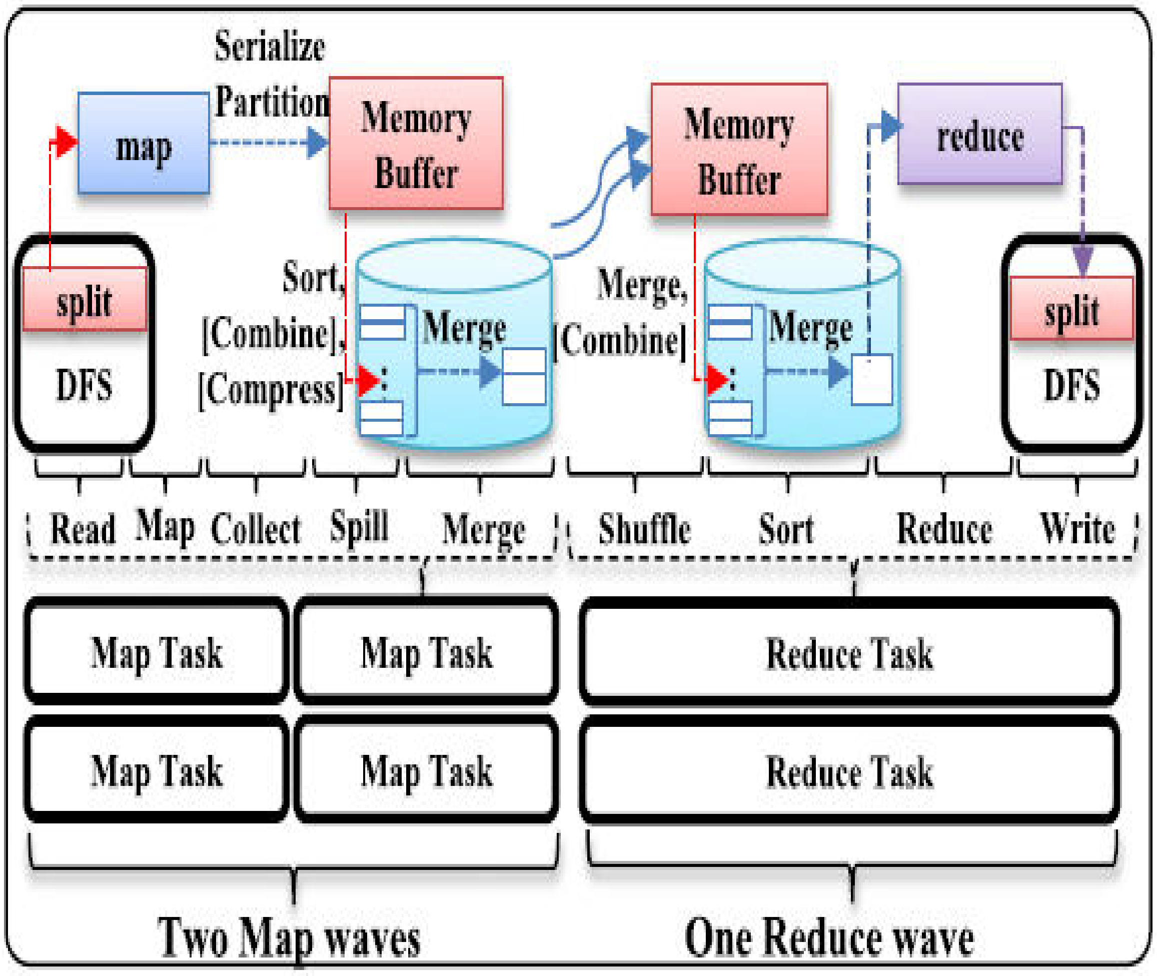
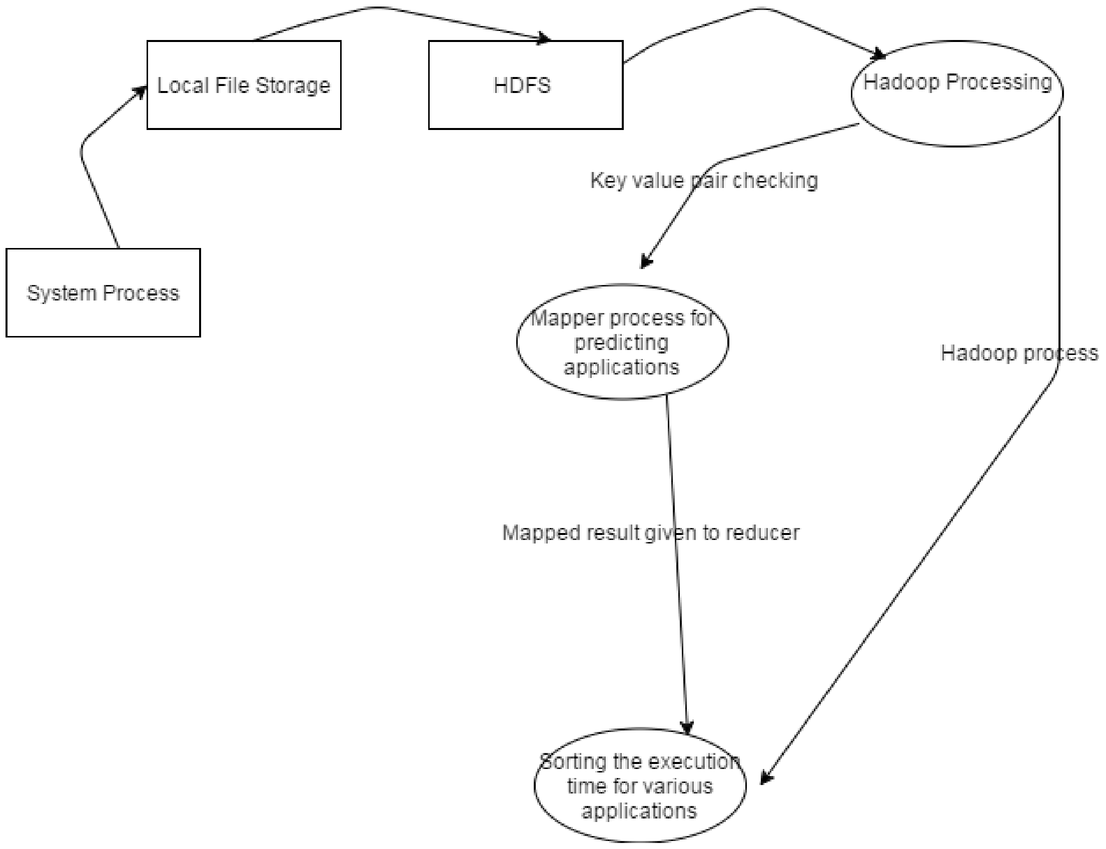


Fig: System Architecture

4.2 DATA FLOW DIAGRAM

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modify ed by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.


Local File Storage
HDFS
Hadoop Processing
Key value pair checking
System Process
Mapper process for predicting applications
Hadoop process
Mapped result given to reducer
Sorting the execution time for various applications

4.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

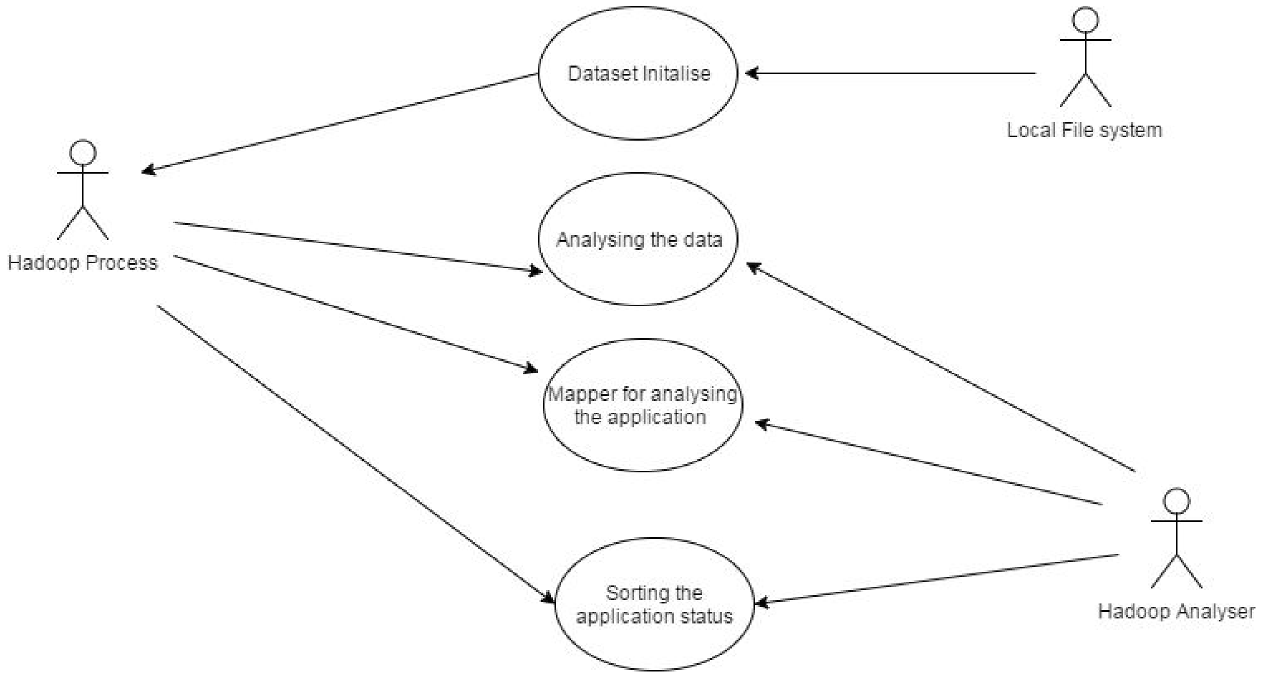
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks.
7. Integrate best practices.

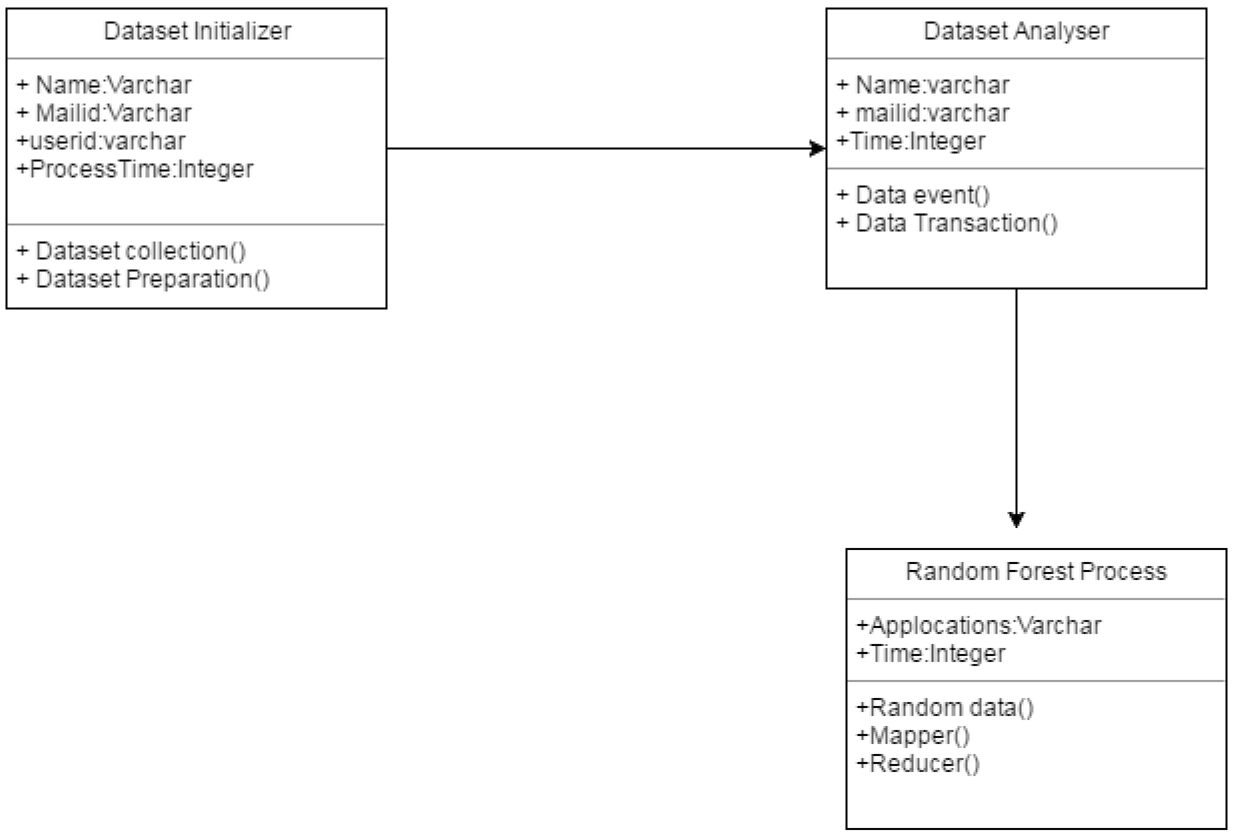
4.3.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



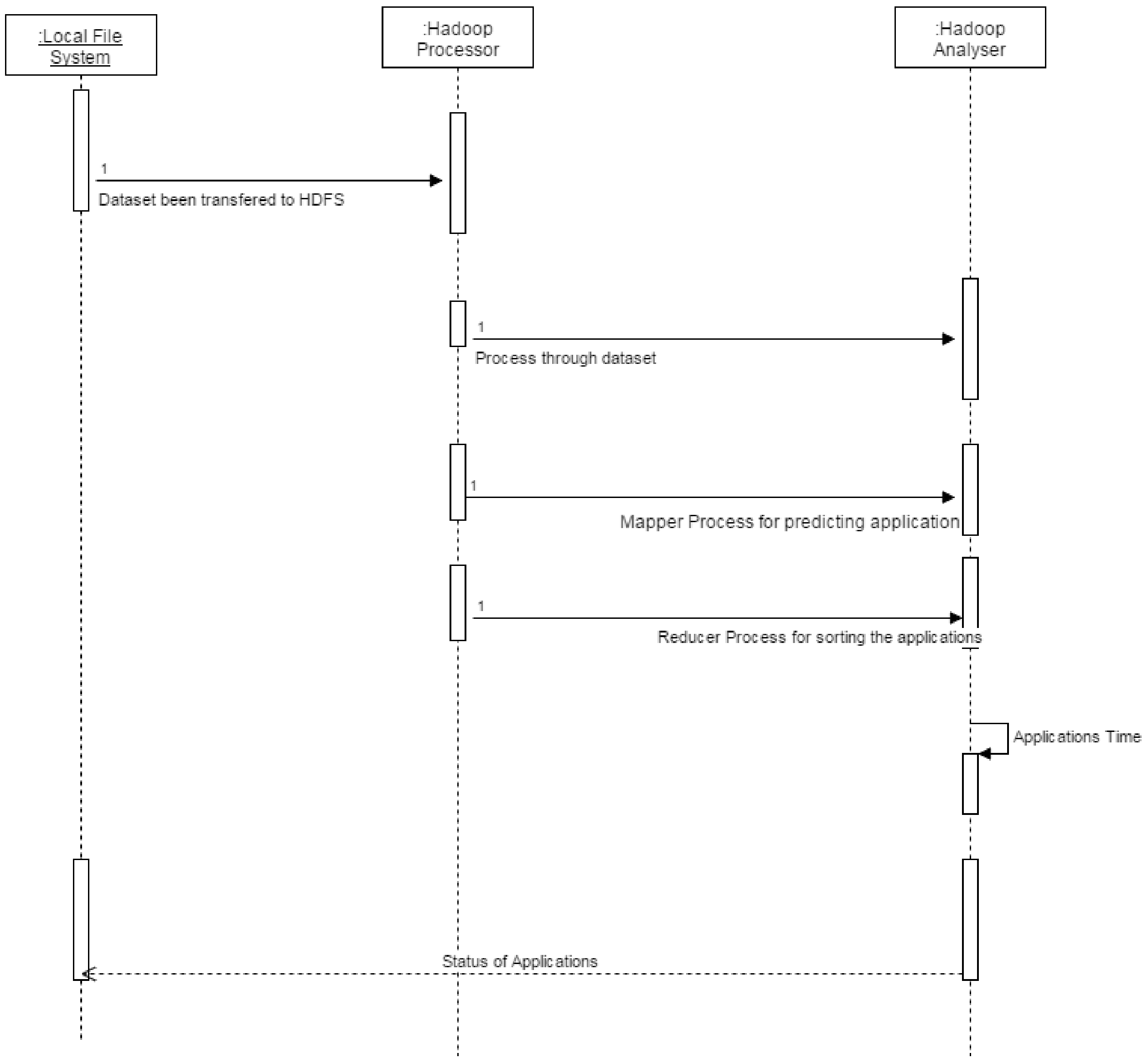
4.3.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



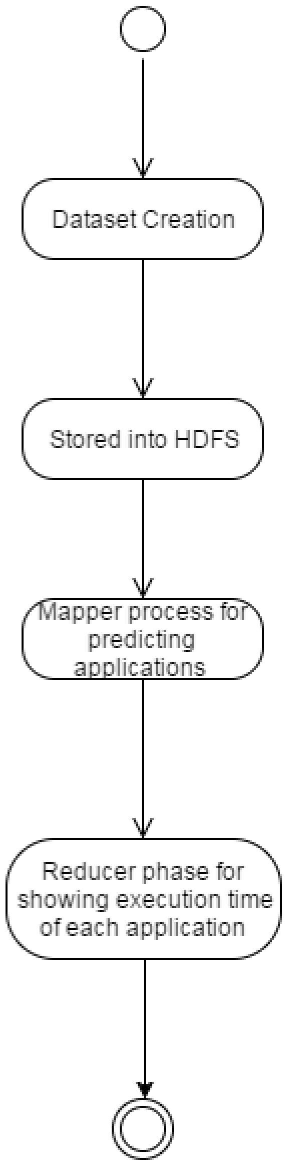
4.3.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



4.3.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.


Dataset Creation
Stored into HDFS
Mapper process for predicting applications
Reducer phase for showing execution time of each application

5. LANGUAGE SPECIFICATION:

HADOOP

Introduction

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. The project URL is http://hadoop.apache.org/.

Assumptions and Goals

Hardware Failure

Hardware failure is the norm rather than the exception. An HDFS instance may consist of hundreds or thousands of server machines, each storing part of the file system's data. The fact that there are a huge number of components and that each component has a non-trivial probability of failure means that some component of HDFS is always non-functional. Therefore, detection of faults and quick, automatic recovery from them is a core architectural goal of HDFS.

Streaming Data Access

Applications that run on HDFS need streaming access to their data sets. They are not general purpose applications that typically run on general purpose file systems. HDFS is designed more for batch processing rather than interactive use by users. The emphasis is on high throughput of data access rather than low latency of data access. POSIX imposes many hard requirements that are not needed for applications that are targeted for HDFS. POSIX semantics in a few key areas has been traded to increase data throughput rates.

Large Data Sets

Applications that run on HDFS have large data sets. A typical file in HDFS is gigabytes to terabytes in size. Thus, HDFS is tuned to support large files. It should provide high aggregate data bandwidth and scale to hundreds of nodes in a single cluster. It should support tens of millions of files in a single instance.

Simple Coherency Model

HDFS applications need a write-once-read-many access model for files. A file once created, written, and closed need not be changed except for appends and truncates. Appending the content to the end of the files is supported but cannot be updated at arbitrary point. This assumption simplifies data coherency issues and enables high throughput data access. A Map Reduce application or a web crawler application fits perfectly with this model.

"Moving Computation is Cheaper than Moving Data"

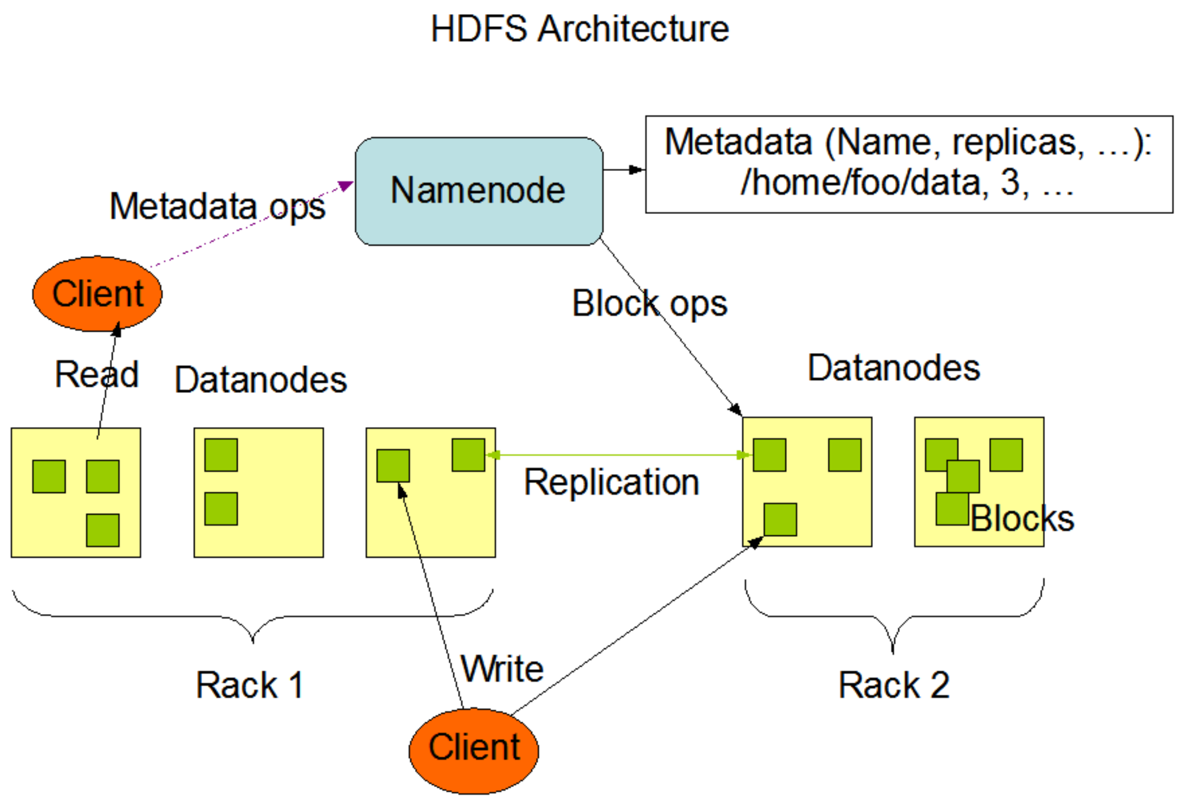
A computation requested by an application is much more efficient if it is executed near the data it operates on. This is especially true when the size of the data set is huge. This minimizes network congestion and increases the overall throughput of the system. The assumption is that it is often better to migrate the computation closer to where the data is located rather than moving the data to where the application is running. HDFS provides interfaces for applications to move themselves closer to where the data is located.

Portability Across Heterogeneous Hardware and Software Platforms

HDFS has been designed to be easily portable from one platform to another. This facilitates widespread adoption of HDFS as a platform of choice for a large set of applications.

NameNode and DataNodes

HDFS has a master/slave architecture. 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 a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.



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 Name Node or the Data Node software. Usage of the highly portable 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 Name Node software. Each of the other machines in the cluster runs one instance of the Data Node software. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case.

The existence of a single Name Node in a cluster greatly simplifies the architecture of the system. The NameZNode is the arbitrator and repository for all HDFS metadata. The system is designed in such a way that user data never flows through the Name Node.

The File System Namespace

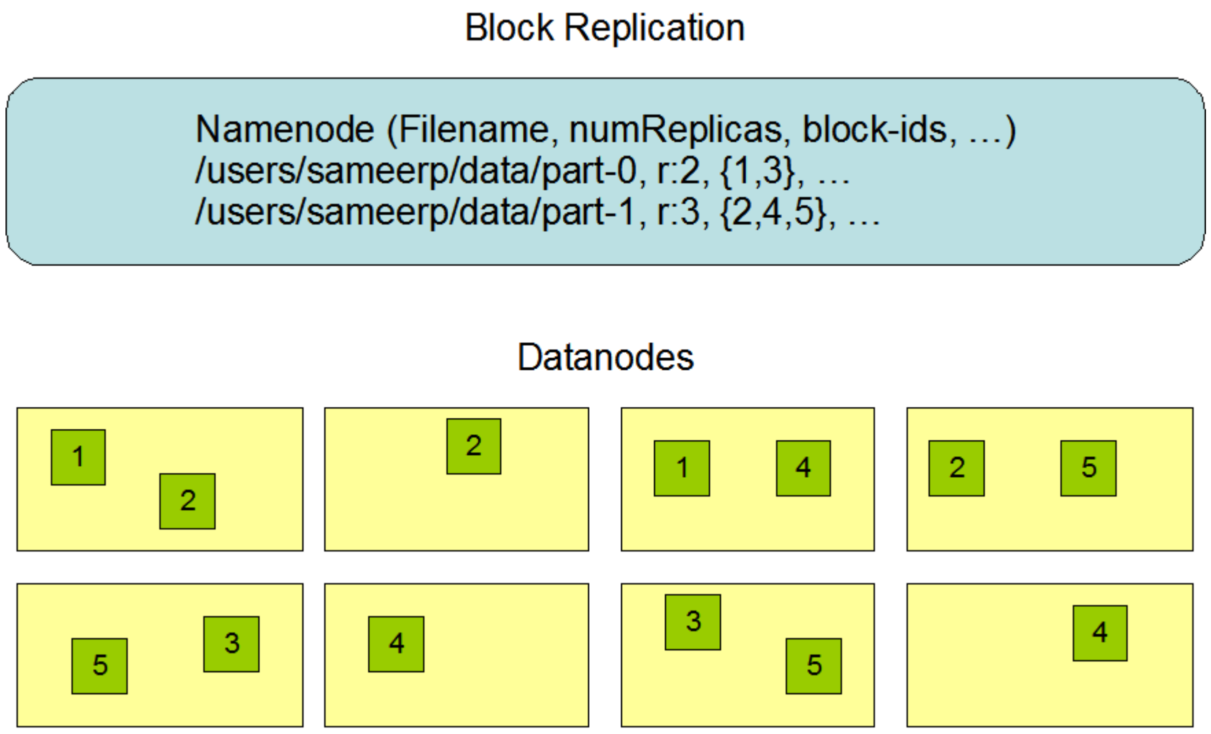
HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories. The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file. HDFS does not yet implement user quotas or access permissions. HDFS does not support hard links or soft links. However, the HDFS architecture does not preclude implementing these features.

The Name Node maintains the file system namespace. Any change to the file system namespace or its properties is recorded by the NameNode. An application can specify the number of replicas of a file that should be maintained by HDFS. The number of copies of a file is called the replication factor of that file. This information is stored by the Name Node.

Data Replication

HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and have strictly one writer at any time.

The Name Node makes all decisions regarding replication of blocks. It periodically receives a Heartbeat and a Block report from each of the Data Nodes in the cluster. Receipt of a Heartbeat implies that the Data Node is functioning properly. A Block report contains a list of all blocks on a Data Node.


Block Replication
Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …
Datanodes
1
2
2
1
4
2
5
5
3
4
3
5
4

Replica Placement: The First Baby Steps

The placement of replicas is critical to HDFS reliability and performance. Optimizing replica placement distinguishes HDFS from most other distributed file systems. This is a feature that needs lots of tuning and experience. The purpose of a rack-aware replica placement policy is to improve data reliability, availability, and network bandwidth utilization. The current implementation for the replica placement policy is a first effort in this direction. The short-term goals of implementing this policy are to validate it on production systems, learn more about its behavior, and build a foundation to test and research more sophisticated policies.

Large HDFS instances run on a cluster of computers that commonly spread across many racks. 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.

The NameNode determines the rack id each DataNode belongs to via the process outlined in Hadoop Rack Awareness. A simple but non-optimal policy is to place replicas on unique racks. This prevents losing data when an entire rack fails and allows use of bandwidth from multiple racks when reading data. This policy evenly distributes replicas in the cluster which makes it easy to balance load on component failure. However, this policy increases the cost of writes because a write needs to transfer blocks to multiple 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 different node in the local rack, and the last on a different node in a different 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 node failure; this policy does not 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, two thirds of replicas are on one rack, and the other third are evenly distributed across the remaining racks. This policy improves write performance without compromising data reliability or read performance.

The current, default replica placement policy described here is a work in progress.

Replica Selection

To minimize global bandwidth consumption and read latency, HDFS tries to satisfy a read request from a replica that is closest to the reader. If there exists a replica on the same rack as the reader node, then that replica is preferred to satisfy the read request. If angg/ HDFS cluster spans multiple data centers, then a replica that is resident in the local data center is preferred over any remote replica.

Safe mode

On startup, the NameNode enters a special state called Safemode. Replication of data blocks does not occur when the NameNode is in the Safemode state. The NameNode receives Heartbeat and Blockreport messages from the DataNodes. A Blockreport contains the list of data blocks that a DataNode is hosting. Each block has a specified minimum number of replicas. A block is considered safely replicated when the minimum number of replicas of that data block has checked in with the NameNode. After a configurable percentage of safely replicated data blocks checks in with the NameNode (plus an additional 30 seconds), the NameNode exits the Safemode state. It then determines the list of data blocks (if any) that still have fewer than the specified number of replicas. The NameNode then replicates these blocks to other DataNodes.

The Persistence of File System Metadata

The HDFS namespace is stored by the NameNode. The NameNode uses a transaction log called the EditLog to persistently record every change that occurs to file system metadata. For example, creating a new file in HDFS causes the NameNode to insert a record into the EditLog indicating this. Similarly, changing the replication factor of a file causes a new record to be inserted into the EditLog. The NameNode uses a file in its local host OS file system to store the EditLog. The entire file system namespace, including the mapping of blocks to files and file system properties, is stored in a file called the FsImage. The FsImage is stored as a file in the NameNode's local file system too.

The NameNode keeps an image of the entire file system namespace and file Blockmap in memory. This key metadata item is designed to be compact, such that a NameNode with 4 GB of RAM is plenty to support a huge number of files and directories. When the NameNode starts up, it reads the FsImage and EditLog from disk, applies all the transactions from the EditLog to the in-memory representation of the FsImage, and flushes out this new version into a new FsImage on disk. It can then truncate the old EditLog because its transactions have been applied to the persistent FsImage. This process is called a checkpoint. In the current implementation, a checkpoint only occurs when the NameNode starts up. Work is in progress to support periodic checkpointing in the near future.

The DataNode stores HDFS data in files in its local file system. The DataNode has no knowledge about HDFS files. It stores each block of HDFS data in a separate file in its local file system. The DataNode does not create all files in the same directory. Instead, it uses a heuristic to determine the optimal number of files per directory and creates subdirectories appropriately. It is not optimal to create all local files in the same directory because the local file system might not be able to efficiently support a huge number of files in a single directory. When a DataNode starts up, it scans through its local file system, generates a list of all HDFS data blocks that correspond to each of these local files and sends this report to the NameNode: this is the Blockreport.

The Communication Protocols

All HDFS communication protocols are layered on top of the TCP/IP protocol. A client establishes a connection to a configurable TCP port on the NameNode machine. It talks the Client Protocol with the NameNode. The Data Nodes talk to the Name Node using the DataNode Protocol. A Remote Procedure Call (RPC) abstraction wraps both the Client Protocol and the DataNode Protocol. By design, the NameNode never initiates any RPCs. Instead, it only responds to RPC requests issued by DataNodes or clients.

Robustness

The primary objective of HDFS is to store data reliably even in the presence of failures. The three common types of failures are NameNode failures, DataNode failures and network partitions.

Data Disk Failure, Heartbeats and Re-Replication

Each DataNode sends a Heartbeat message to the NameNode periodically. A network partition can cause a subset of DataNodes to lose connectivity with the NameNode. The NameNode detects this condition by the absence of a Heartbeat message. The NameNode marks DataNodes without recent Heartbeats as dead and does not forward any new IO requests to them. Any data that was registered to a dead DataNode is not available to HDFS any more. DataNode death may cause the replication factor of some blocks to fall below their specified value. The NameNode constantly tracks which blocks need to be replicated and initiates replication whenever necessary. The necessity for re-replication may arise due to many reasons: a DataNode may become unavailable, a replica may become corrupted, a hard disk on a DataNode may fail, or the replication factor of a file may be increased.

Cluster Rebalancing

The HDFS architecture is compatible with data rebalancing schemes. A scheme might automatically move data from one DataNode to another if the free space on a DataNode falls below a certain threshold. In the event of a sudden high demand for a particular file, a scheme might dynamically create additional replicas and rebalance other data in the cluster. These types of data rebalancing schemes are not yet implemented.

Data Integrity

It is possible that a block of data fetched from a DataNode arrives corrupted. This corruption can occur because of faults in a storage device, network faults, or buggy software. The HDFS client software implements checksum checking on the contents of HDFS files. When a client creates an HDFS file, it computes a checksum of each block of the file and stores these checksums in a separate hidden file in the same HDFS namespace. When a client retrieves file contents it verifies that the data it received from each DataNode matches the checksum stored in the associated checksum file. If not, then the client can opt to retrieve that block from another DataNode that has a replica of that block.

Metadata Disk Failure

The FsImage and the EditLog are central data structures of HDFS. A corruption of these files can cause the HDFS instance to be non-functional. For this reason, the NameNode can be configured to support maintaining multiple copies of the FsImage and EditLog. Any update to either the FsImage or EditLog causes each of the FsImages and EditLogs to get updated synchronously. This synchronous updating of multiple copies of the FsImage and EditLog may degrade the rate of namespace transactions per second that a NameNode can support. However, this degradation is acceptable because even though HDFS applications are very data intensive in nature, they are not metadata intensive. When a NameNode restarts, it selects the latest consistent FsImage and EditLog to use.

The NameNode machine is a single point of failure for an HDFS cluster. If the NameNode machine fails, manual intervention is necessary. Currently, automatic restart and failover of the NameNode software to another machine is not supported.

Snapshots

Snapshots support storing a copy of data at a particular instant of time. One usage of the snapshot feature may be to roll back a corrupted HDFS instance to a previously known good point in time. HDFS does not currently support snapshots but will in a future release.

Data Organization

Data Blocks

HDFS is designed to support very large files. Applications that are compatible with HDFS are those that deal with large data sets. These applications write their data only once but they read it one or more times and require these reads to be satisfied at streaming speeds. HDFS supports write-once-read-many semantics on files. A typical block size used by HDFS is 64 MB. Thus, an HDFS file is chopped up into 64 MB chunks, and if possible, each chunk will reside on a different DataNode.

Staging

A client request to create a file does not reach the NameNode immediately. In fact, initially the HDFS client caches the file data into a temporary local file. Application writes are transparently redirected to this temporary local file. When the local file accumulates data worth over one HDFS block size, the client contacts the NameNode.

The NameNode inserts the file name into the file system hierarchy and allocates a data block for it. The NameNode responds to the client request with the identity of the DataNode and the destination data block. Then the client flushes the block of data from the local temporary file to the specified DataNode. When a file is closed, the remaining un-flushed data in the temporary local file is transferred to the DataNode. The client then tells the NameNode that the file is closed. At this point, the NameNode commits the file creation operation into a persistent store. If the NameNode dies before the file is closed, the file is lost.

The above approach has been adopted after careful consideration of target applications that run on HDFS. These applications need streaming writes to files. If a client writes to a remote file directly without any client side buffering, the network speed and the congestion in the network impacts throughput considerably. This approach is not without precedent. Earlier distributed file systems, e.g. AFS, have used client side caching to improve performance. A POSIX requirement has been relaxed to achieve higher performance of data uploads.

Replication Pipelining

When a client is writing data to an HDFS file, its data is first written to a local file as explained in the previous section. Suppose the HDFS file has a replication factor of three. When the local file accumulates a full block of user data, the client retrieves a list of DataNodes from the NameNode. This list contains the DataNodes that will host a replica of that block. The client then flushes the data block to the first DataNode. The first DataNode starts receiving the data in small portions, writes each portion to its local repository and transfers that portion to the second DataNode in the list. The second DataNode, in turn starts receiving each portion of the data block, writes that portion to its repository and then flushes that portion to the third DataNode. Finally, the third DataNode writes the data to its local repository. Thus, a DataNode can be receiving data from the previous one in the pipeline and at the same time forwarding data to the next one in the pipeline. Thus, the data is pipelined from one DataNode to the next.

Accessibility

HDFS can be accessed from applications in many different ways. Natively, HDFS provides a FileSystem Java API for applications to use. A C language wrapper for this Java API is also available. In addition, an HTTP browser can also be used to browse the files of an HDFS instance. Work is in progress to expose HDFS through the WebDAV protocol.

FS Shell

HDFS allows user data to be organized in the form of files and directories. It provides a commandline interface called FS shell that lets a user interact with the data in HDFS. The syntax of this command set is similar to other shells (e.g. bash, csh) that users are already familiar with. Here are some sample action/command pairs:

Action	**Command**
Create a directory named /foodir	bin/hadoop dfs -mkdir /foodir
Remove a directory named /foodir	bin/hadoop fs -rm -R /foodir
View the contents of a file named /foodir/myfile.txt	bin/hadoop dfs -cat /foodir/myfile.txt

FS shell is targeted for applications that need a scripting language to interact with the stored data.

DFS Admin

The DFS Admin command set is used for administering an HDFS cluster. These are commands that are used only by an HDFS administrator. Here are some sample action/command pairs:

Action	**Command**

Action	Command
Put the cluster in Safemode	bin/hdfs dfsadmin -safemode enter
Generate a list of DataNodes	bin/hdfs dfsadmin -report
Recommission or decommission DataNode(s)	bin/hdfs dfsadmin -refreshNodes

Browser Interface

A typical HDFS install configures a web server to expose the HDFS namespace through a configurable TCP port. This allows a user to navigate the HDFS namespace and view the contents of its files using a web browser.

Space Reclamation

File Deletes and Undeletes

When a file is deleted by a user or an application, it is not immediately removed from HDFS. Instead, HDFS moves it to a trash directory (each user has its own trash directory under /user/<username>/.Trash). The file can be restored quickly as long as it remains in trash. Most recent deleted files are moved to the current trash directory (/user/<username>/.Trash/Current), and in a configurable interval, HDFS creates checkpoints.

(under /user/<username>/.Trash/<date>) for files in current trash directory and deletes old checkpoints when they are expired. After the expiry of its life in trash, the NameNode deletes the file from the HDFS namespace. The deletion of a file causes the blocks associated with the file to be freed. Note that there could be an appreciable time delay between the time a file is deleted by a user and the time of the corresponding increase in free space in HDFS.

Currently, the trash feature is disabled by default (deleting files without storing in trash). User can enable this feature by setting a value greater than zero for parameter fs.trash.interval (in core-site.xml). This value tells the NameNode how long a checkpoint will be expired and removed from HDFS. In addition, user can configure an appropriate time to tell NameNode how often to create checkpoints in trash (the parameter stored as fs.trash.checkpoint.interval in core-site.xml), this value should be smaller or equal to fs.trash.interval.

Decrease Replication Factor

When the replication factor of a file is reduced, the NameNode selects excess replicas that can be deleted. The next Heartbeat transfers this information to the DataNode. The DataNode then removes the corresponding blocks and the corresponding free space appears in the cluster. Once again, there might be a time delay between the completion of the setReplication API call and the appearance of free space in the cluster.

6. METHODOLOGY

6.1 MODULES

Random forest is an ensemble model that can be used

- Random Forest
- Analytical Models
- Resource consumption

MODULES DESCRIPTION:

Random Forest

Both classification and regression. It operates by constructing a multitude of decision trees at training time; prediction then combines the outputs by the individual trees to arrive at the final output (e.g., majority voting for classification, and average for regression. A key feature of random forests is that they correct for decision trees' tendency to overfit to their training data.

Analytical Models

Although the analytical performance models are fine-grained at the level of Map Reduce phases, they assume the execution time per operation to depend on the cluster resources only. In addition, they further assume it to be constant across different Hadoop configurations. To verify whether the above assumption holds true, we take the same 10 Hadoop configuration parameters and we assign random values to them, forming six different Hadoop configurations,

Resource Consumption:

Support Vector Machines (SVM) to build a model called AROMA to predict the performance of Hadoop applications with different configurations. AROMA is based on the observation that jobs with similar resource consumption characteristics exhibit similar performance behavior across Hadoop configurations. This observation might hold true if the performance models were constructed for very fine-grained objects. However, the amount of CPU, I/O, and network consumed at the Hadoop job level to identify the resource consumption characteristics, which is a coarse-grained approach.

AROMA groups the sort and word count benchmarks in the same group because they consume similar cluster resources. To verify this assumption, we conduct an experiment to observe the resource consumption for both benchmarks at the phase level. The phases of sort process almost the same amount of data while the phases of word count are significantly different, which indicates that the resources consumed by each phase is similar for sort but is dramatically different for word count.

6.2 LANGUAGE IMPLEMENTATION

Decision Tree:

```
package com.rf.fast;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Map.Entry;

public class DecisionTree {

        private static final int INDEX_SKIP=3;
        private static final int MIN_SIZE_TO_CHECK_EACH=10;
        private static final int MIN_NODE_SIZE=5;
        private int N;
        private int testN;

        private int correct;



        private int[] importances;
```

```
	public ArrayList<String> predictions;
	private TreeNode root;

	private RandomForest forest;

	public DecisionTree(ArrayList<ArrayList<String>> data,RandomForest forest,
int treenum) {
		// TODO Auto-generated constructor stub
		this.forest=forest;
		N=data.size();
		importances = new int[RandomForest.M];

		ArrayList<ArrayList<String>> train = new
ArrayList<ArrayList<String>>(N);
		ArrayList<ArrayList<String>> test = new
ArrayList<ArrayList<String>>();

		BootStrapSample(data,train,test,treenum);
		testN=test.size();
		correct=0;

		root=CreateTree(train,treenum);
		FlushData(root, treenum);
	}
	@SuppressWarnings("unchecked")
	private void BootStrapSample(ArrayList<ArrayList<String>>
data,ArrayList<ArrayList<String>> train,ArrayList<ArrayList<String>> test,int
numb){
		ArrayList<Integer> indices=new ArrayList<Integer>();
		for (int n=0;n<N;n++)
```

```
                indices.add((int)Math.floor(Math.random()*N));
            ArrayList<Boolean> in=new ArrayList<Boolean>();
            for (int n=0;n<N;n++)
                in.add(false); //have to initialize it first
            for (int num:indices){
                ArrayList<String> k = data.get(num);
                train.add((ArrayList<String>) k.clone());
                in.set(num,true);
            }//System.out.println("created training-data for tree : "+numb);
            for (int i=0;i<N;i++)
                if (!in.get(i))
                    test.add(data.get(i));//System.out.println("created     testing-
data for tree : "+numb);//everywhere its set to false we get those to test data

//          System.out.println("bootstrap          N:"+N+"          size          of
bootstrap:"+bootstrap.size());
        }


        private TreeNode CreateTree(ArrayList<ArrayList<String>> train, int ntree){
            TreeNode root=new TreeNode();
            root.label = "|ROOT|";
            root.data=train;

            RecursiveSplit(root,ntree);
            return root;
        }

        public class TreeNode implements Cloneable{
            public boolean isLeaf;
            public ArrayList<TreeNode> ChildNode ;
```

```
                    public HashMap<String, String> Missingdata;
//                  public TreeNode left;
//                  public TreeNode right;
                    public int splitAttributeM;//which attribute its split on...
                    public boolean spiltonCateg ;
                    public String Class;
                    public ArrayList<ArrayList<String>> data;
                    public String splitValue;//check this if it return false on splitonCateg
                    public String label;//Label of each node
                    public int generation;

                    public TreeNode(){
                        splitAttributeM=-99;
                        splitValue="-99";
                        generation=1;
                    }
                    public TreeNode clone(){ //"data" element always null in clone
                        TreeNode copy=new TreeNode();
                        copy.isLeaf=isLeaf;
                        for(TreeNode TN : ChildNode){
                            if(TN != null){
                                copy.ChildNode.add(TN.clone());
                            }
                        }

                        copy.splitAttributeM=splitAttributeM;
                        copy.Class=Class;
                        copy.splitValue=splitValue;
                        copy.spiltonCateg = spiltonCateg;
                        copy.label=label;
                        return copy;
```

```
		}
	}
	/**
	 * hold the entropy
	 * @author Mohammad
	 *
	 */
	private class DoubleWrap{
		public double d;
		public DoubleWrap(double d){
			this.d=d;
		}
	}

	public ArrayList<String> CalculateClasses(ArrayList<ArrayList<String>>
traindata,ArrayList<ArrayList<String>> testdata, int treenumber){
		ArrayList<String> predicts = new ArrayList<String>();
		for(ArrayList<String> record:testdata){
			String Clas = Evaluate(record, traindata);
			if(Clas==null){
				predicts.add("n/a");
			}
			else
			predicts.add(Clas);
		}
		predictions = predicts;
		return predicts;
	}

	public String Evaluate(ArrayList<String> record, ArrayList<ArrayList<String>>
tester){
```

```
                    TreeNode evalNode=root;
                        while (true) {
                            if(evalNode.isLeaf)
                                return evalNode.Class;
                            else{
                                if(evalNode.spiltonCateg){
                                // if its categorical
                                    String              recordCategory              =
record.get(evalNode.splitAttributeM);
                                    boolean     found=false;String     Res     =
evalNode.Missingdata.get(GetClass(record));

                                    for(TreeNode child:evalNode.ChildNode){

                                        // Check for child with label same
the data point

    if(recordCategory.equalsIgnoreCase(child.label)){//what if the category is not
present at all
                                                evalNode = child;
                                                found = true;
                                                break;
                                        }
                                    }if(!found){
                                        for(TreeNode
child:evalNode.ChildNode){
                                                if(Res!=null){

    if(Res.trim().equalsIgnoreCase(child.label)){
                                                        evalNode      =
child;
```

```
                                                            break;
                                                        }
                                                    }else{
                                                        return "n/a";
                                                    }
                                                }
                                            }
                                        }else{
                                            //if its real-valued
                                            double          Compare          =
Double.parseDouble(evalNode.splitValue);
                                            double          Actual          =
Double.parseDouble(record.get(evalNode.splitAttributeM));
                                            if(Actual <= Compare){

    if(evalNode.ChildNode.get(0).label.equalsIgnoreCase("Left"))

    evalNode=evalNode.ChildNode.get(0);
                                                else

    evalNode=evalNode.ChildNode.get(1);
//                                              evalNode=evalNode.left;
//                                              System.out.println("going   in   child
:"+evalNode.label);
                                            }else{

    if(evalNode.ChildNode.get(0).label.equalsIgnoreCase("Right"))

    evalNode=evalNode.ChildNode.get(0);
                                                else
                                        }
                                    }
                                }
                            }
                        }

        private         TreeNode     getChildtoTraverse(ArrayList<TreeNode>      Chil,int
splitAttributeM, String classofRecord) {
                // TODO Auto-generated method stub
                int max=0;TreeNode res=new TreeNode();
                for(int i=0;i<Chil.size();i++){
                        if(Chil.get(i)!=null && Chil.get(i).data.size()>0){
                                int k=0;
                                for(ArrayList<String> SSS:Chil.get(i).data){
                                        if(GetClass(SSS).equalsIgnoreCase(classofRecord))
                                                k++;
                                }if(k>max){
                                        max=k;
                                        res = Chil.get(i);
                                }
                        }
                }return res;
        }

        private void RecursiveSplit(TreeNode parent, int Ntreenum){

                if (!parent.isLeaf){


                        //------------------------------Step A
                        String Class=CheckIfLeaf(parent.data);
                        if (Class != null){
                                parent.isLeaf=true;
                                parent.Class=Class;
                                return;
                        }


                        //-------------------------------Step B
                        int Nsub=parent.data.size();

                        parent.ChildNode = new ArrayList<DecisionTree.TreeNode>();
                        for(TreeNode TN: parent.ChildNode){
                                TN = new TreeNode();
                                TN.generation = parent.generation+1;
                        }

                        ArrayList<Integer>   vars=GetVarsToInclude();//randomly   selects
Ms.Nos of attributes from M
                        DoubleWrap lowestE=new DoubleWrap(Double.MAX_VALUE);


                        //-------------------------------Step C
                        for (int m:vars){//m is from 0-M
                                ArrayList<Integer>                DataPointToCheck=new
ArrayList<Integer>();// which data points to be scrutinized

                                SortAtAttribute(parent.data,m);//sorts    on    a    particular
column in the row
                                for (int n=1;n<Nsub;n++){
                                        String classA=GetClass(parent.data.get(n-1));
                                        String classB=GetClass(parent.data.get(n));
```

```
                                        if(!classA.equalsIgnoreCase(classB))
                                                DataPointToCheck.add(n);
                                }

                                if (DataPointToCheck.size() == 0){//if all the Y-values are
same, then get the class directly
                                        parent.isLeaf=true;
                                        parent.Class=GetClass(parent.data.get(0));
                                        continue;
                                }

                                if          (DataPointToCheck.size()          >
MIN_SIZE_TO_CHECK_EACH){
                                        for                                (int
i=0;i<DataPointToCheck.size();i+=INDEX_SKIP){
                                                CheckPosition(m, DataPointToCheck.get(i),
Nsub, lowestE, parent, Ntreenum);
                                                if (lowestE.d == 0)//lowestE now has the
minimum conditional entropy so IG is max there
                                                        break;
                                        }
                                }else{
                                        for (int k:DataPointToCheck){
                                                CheckPosition(m,k, Nsub, lowestE, parent,
Ntreenum);
                                                if (lowestE.d == 0)//lowestE now has the
minimum conditional entropy so IG is max there
                                                        break;
                                        }
                                }
                                if (lowestE.d == 0)
```

```
                    break;
                }
                //System.out.println("Adding        "+parent.ChildNode.size()+"
children at level: "+parent.generation);
                //------------------------------Step D
                for(TreeNode Child:parent.ChildNode){
                    if(Child.data.size()==1){
                        Child.isLeaf=true;
                        Child.Class=GetClass(Child.data.get(0));
                    }else if(Child.data.size()<MIN_NODE_SIZE){
                        Child.isLeaf=true;
                        Child.Class=GetMajorityClass(Child.data);
                    }else{
                        Class=CheckIfLeaf(Child.data);
                        if(Class==null){
                            Child.isLeaf=false;
                            Child.Class=null;
                        }else{
                            Child.isLeaf=true;
                            Child.Class=Class;
                        }
                    }
                    if(!Child.isLeaf){
                        RecursiveSplit(Child, Ntreenum);
                    }
                }
            }
        }


        private void SortAtAttribute(ArrayList<ArrayList<String>> data, int m) {
                if(forest.DataAttributes.get(m) == 'C')
                        System.out.print("");//Collections.sort(data,new
AttributeComparatorCateg(m));
                else
                        Collections.sort(data,new AttributeComparatorReal(m));

        }


        private         class           AttributeComparatorCateg             implements
Comparator<ArrayList<String>>{
                /** the specified attribute */
                private int m;

                public AttributeComparatorCateg(int m){
                        this.m=m;
                }

                @Override
                public  int  compare(ArrayList<String>  arg1,  ArrayList<String>  arg2)
{//compare strings
                        // TODO Auto-generated method stub
                        String a = arg1.get(m);
                        String b = arg2.get(m);
                        return a.compareToIgnoreCase(b);
                }
        }


        private         class           AttributeComparatorReal              implements
Comparator<ArrayList<String>>{
```

```
                /** the specified attribute */
                private int m;

                public AttributeComparatorReal(int m){
                        this.m=m;
                }

                @Override
                public   int   compare(ArrayList<String>   arg1,   ArrayList<String>   arg2)
{//compare value of strings
                        // TODO Auto-generated method stub
                        double a2 = Double.parseDouble(arg1.get(m));
                        double b2 = Double.parseDouble(arg2.get(m));
                        if(a2<b2)
                                return -1;
                        else if(a2>b2)
                                return 1;
                        else
                                return 0;
                }
        }


        private String GetMajorityClass(ArrayList<ArrayList<String>> data){
                // find the max class for this data.
                ArrayList<String> ToFind = new ArrayList<String>();
                for(ArrayList<String> s:data){
                        ToFind.add(s.get(s.size()-1));
                }
                String MaxValue = null; int MaxCount = 0;
                for(String s1:ToFind){
                        int count =0;
                        for(String s2:ToFind){
                                if(s2.equalsIgnoreCase(s1))
                                        count++;
                        }
                        if(count > MaxCount){
                                MaxValue = s1;
                                MaxCount = count;
                        }
                }return MaxValue;
        }

        private double CheckPosition(int m,int n,int Nsub,DoubleWrap
lowestE,TreeNode parent, int nTre){

                double entropy =0;

                if (n < 1) //exit conditions
                        return 0;
                if (n > Nsub)
                        return 0;

                if(forest.DataAttributes.get(m)=='C'){

                        //this is a categorical thing
                        // find out the distinct values in that attribute...from parent.data
                        ArrayList<String> uni_categ = new ArrayList<String>(); //unique
categories
                        ArrayList<String> uni_classes = new ArrayList<String>();
//unique classes
```

```
			HashMap<String, String> ChildMissingMap = new
HashMap<String, String>();// Class Vs Node-label
			HashMap<String, Integer> ChilFreq = new HashMap<String,
Integer>();//Node-Label Vs frequency

			for(ArrayList<String> s:parent.data){
				if(!uni_categ.contains(s.get(m).trim())){
					uni_categ.add(s.get(m).trim());
					ChilFreq.put(s.get(m), 0);
				}

				if(!uni_classes.contains(GetClass(s)))
					uni_classes.add(GetClass(s));
			}

			//data pertaining to each of the value
			HashMap<String, ArrayList<ArrayList<String>>> ChildDataMap
= new HashMap<String, ArrayList<ArrayList<String>>>();
			for(String s:uni_categ){
				ArrayList<ArrayList<String>> child_data = new
ArrayList<ArrayList<String>>();
				for(ArrayList<String> S:parent.data){
					if(s.trim().equalsIgnoreCase(S.get(m).trim()))
						child_data.add(S);
				}
				ChildDataMap.put(s, child_data);
			}

			//can merge the above two
			//Adding missing-data-suits
			for(String S1:uni_classes){
```

```
                                        int max=0;String Resul = null;
                                        for(ArrayList<String> S2:parent.data){
                                                if(GetClass(S2).equalsIgnoreCase(S1)){
                                                        if(ChilFreq.containsKey(S2.get(m)))
                                                                ChilFreq.put(S2.get(m),
ChilFreq.get(S2.get(m))+1);
                                                }
                                                if(ChilFreq.get(S2.get(m))>max){
                                                        max=ChilFreq.get(S2.get(m));
                                                        Resul = S2.get(m);
                                                }
                                        }
                                        ChildMissingMap.put(S1,
Resul);//System.out.println("Mapping Class: "+S1+" to attribute: "+Resul);
                                }
                                //calculating entropy
                                for(Entry<String,ArrayList<ArrayList<String>>>
entry:ChildDataMap.entrySet()){

        entropy+=CalEntropy(getClassProbs(entry.getValue()))*entry.getValue().size();
                                }
                                entropy = entropy/((double)Nsub);
                                //if its the least...
                                if (entropy < lowestE.d){
                                        lowestE.d=entropy;
                                        parent.splitAttributeM=m;
                                        parent.spiltonCateg = true;
                                        parent.splitValue=parent.data.get(n).get(m);
                                        parent.Missingdata=ChildMissingMap;
                                        /**
                                         * Adding Data to Child
                                        *
                                        */
                                        ArrayList<TreeNode>        Children        =        new
ArrayList<TreeNode>();
                                        for(Entry<String,ArrayList<ArrayList<String>>>
entry:ChildDataMap.entrySet()){
                                                TreeNode Child = new TreeNode();
                                                Child.data=entry.getValue();
                                                Child.label=entry.getKey();
                                                Children.add(Child);
                                        }
                                        parent.ChildNode=Children;
                                }
                }else{

                                //this is a real valued thing

                                HashMap<String,    ArrayList<ArrayList<String>>>    UpLo    =
GetUpperLower(parent.data, n, m);

                                ArrayList<ArrayList<String>> lower = UpLo.get("lower");
                                ArrayList<ArrayList<String>> upper = UpLo.get("upper");

                                ArrayList<Double> pl=getClassProbs(lower);
                                ArrayList<Double> pu=getClassProbs(upper);
                                double eL=CalEntropy(pl);
                                double eU=CalEntropy(pu);

                                entropy =(eL*lower.size()+eU*upper.size())/((double)Nsub);

                                if (entropy < lowestE.d){
```

```
                                lowestE.d=entropy;
                                parent.splitAttributeM=m;
                                parent.spiltonCateg=false;
                                parent.splitValue = parent.data.get(n).get(m).trim();
                                /**
                                 * Adding Data to Left/Right Child
                                 *
                                 */
                                ArrayList<TreeNode>        Children2        =        new
ArrayList<TreeNode>();
                                TreeNode Child_left = new TreeNode();
                                Child_left.data=lower;
                                Child_left.label="Left";
                                Children2.add(Child_left);
                                TreeNode Child_Right = new TreeNode();
                                Child_Right.data=upper;
                                Child_Right.label="Right";
                                Children2.add(Child_Right);
                                parent.ChildNode=Children2;//clone karo....
                        }
                }
                return entropy;
        }


        private        HashMap<String,        ArrayList<ArrayList<String>>>
GetUpperLower(ArrayList<ArrayList<String>> data, int n2,int m) {

                HashMap<String, ArrayList<ArrayList<String>>> UpperLower = new
HashMap<String, ArrayList<ArrayList<String>>>();
```

```
		ArrayList<ArrayList<String>>		lowerr		=		new
ArrayList<ArrayList<String>>();
		ArrayList<ArrayList<String>>		upperr		=		new
ArrayList<ArrayList<String>>();
		for(int n=0;n<n2;n++)
			lowerr.add(data.get(n));
		for(int n=n2;n<data.size();n++)
			upperr.add(data.get(n));
		UpperLower.put("lower", lowerr);
		UpperLower.put("upper", upperr);

		return UpperLower;
	}

	private	ArrayList<Double>	getClassProbs(ArrayList<ArrayList<String>>
record){
		double N=record.size();
		HashMap<String, Integer > counts = new HashMap<String, Integer>();
		for(ArrayList<String> s : record){
			String clas = GetClass(s);
			if(counts.containsKey(clas))
				counts.put(clas, counts.get(clas)+1);
			else
				counts.put(clas, 1);
		}
		ArrayList<Double> probs = new ArrayList<Double>();
		for(Entry<String, Integer> entry : counts.entrySet()){
			double prob = entry.getValue()/N;
			probs.add(prob);
		}return probs;
	}
	private static final double logoftwo=Math.log(2);

	private double CalEntropy(ArrayList<Double> ps){
		double e=0;
		for (double p:ps){
			if (p != 0) //otherwise it will divide by zero - see TSK p159
				e+=p*Math.log(p)/logoftwo;
		}
		return -e; //according to TSK p158
	}

	private ArrayList<Integer> GetVarsToInclude() {
		boolean[] whichVarsToInclude=new boolean[RandomForest.M];

		for (int i=0;i<RandomForest.M;i++)
			whichVarsToInclude[i]=false;

		while (true){
			int a=(int)Math.floor(Math.random()*RandomForest.M);
			whichVarsToInclude[a]=true;
			int N=0;
			for (int i=0;i<RandomForest.M;i++)
				if (whichVarsToInclude[i])
					N++;
			if (N == RandomForest.Ms)
				break;
		}

		ArrayList<Integer>                                    shortRecord=new
ArrayList<Integer>(RandomForest.Ms);
```

```
		for (int i=0;i<RandomForest.M;i++)
			if (whichVarsToInclude[i])
				shortRecord.add(i);
		return shortRecord;//values from 0-M
	}

	public static String GetClass(ArrayList<String> record){
		return record.get(RandomForest.M).trim();
	}

	private String CheckIfLeaf(ArrayList<ArrayList<String>> data){
//		System.out.println("checkIfLeaf");
		boolean isCLeaf=true;
		String ClassA=GetClass(data.get(0));
		for(ArrayList<String> record : data){
			if(!ClassA.equalsIgnoreCase(GetClass(record))){
				isCLeaf = false;
				return null;
			}
		}
		if (isCLeaf)
			return GetClass(data.get(0));
		else
			return null;
	}

	private void FlushData(TreeNode node, int treenum){
		node.data=null;
		if(node.ChildNode!=null){
			for(TreeNode TN : node.ChildNode){
```

```
                        if(TN != null)
                            FlushData(TN,treenum);
                    }
                }
            }

}
```

Random forest:

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class RandomForest {
  private static int NUM_THREADS;//=Runtime.getRuntime().availableProcessors();
        public static int C;

        public static int M;

        public      static      int      Ms;//recommended      by      Breiman:
=(int)Math.round(Math.log(M)/Math.log(2)+1);
```

```
	private long time_o;

	private int numTrees;

	private double update;
	private double progress;
	private int[] importances;
	private HashMap<int[],int[]> estimateOOB;
		private double error;
	private ExecutorService treePool;
	private ArrayList<ArrayList<String>> data;
	private ArrayList<ArrayList<String>> testdata;
	private ArrayList<ArrayList<String>> Prediction;
public ArrayList<Character> DataAttributes;
	public HashMap<ArrayList<String>, String> FinalPredictions;
public int corretsPredictions;


	@SuppressWarnings("static-access")
	public RandomForest(ArrayList<Character> dataLayout,int numTrees,int numThreads,int M,int Ms,int C, ArrayList<ArrayList<String>> train,ArrayList<ArrayList<String>> test) {
		// TODO Auto-generated constructor stub
		StartTimer();
		this.numTrees=numTrees;
		this.NUM_THREADS=numThreads;
		this.data=train;
		this.testdata=test;
		this.M=M;
		this.Ms=Ms;
		this.C=C;
		this.DataAttributes = dataLayout;

		trees2 = new ArrayList<DecisionTree>(numTrees);
		update=100/((double)numTrees);
		progress=0;
		corretsPredictions =0;
		System.out.println("creating "+numTrees+" trees in a random Forest. . . ");
		System.out.println("total data size is "+train.size());
		System.out.println("number of attributes "+M);
		System.out.println("number of selected attributes "+Ms);

		estimateOOB=new HashMap<int[],int[]>(data.size());
		Prediction = new ArrayList<ArrayList<String>>();
		FinalPredictions = new HashMap<ArrayList<String>, String>();

	}

	@SuppressWarnings("unchecked")
	public void Start(boolean forAccuracy,boolean withThreads) {
		// TODO Auto-generated method stub
		System.out.println("Number of threads started : "+NUM_THREADS);
		System.out.print("Starting trees");
		treePool=Executors.newFixedThreadPool(NUM_THREADS);
		for (int t=0;t<numTrees;t++){
			treePool.execute(new CreateTree(data,this,t+1));
		}treePool.shutdown();
		try {
			treePool.awaitTermination(10,TimeUnit.SECONDS);  //effectively
infinity
	    } catch (InterruptedException ignored){
		System.out.println("interrupted exception in Random Forests");
```

```
        }
            System.out.println("Trees        Production        completed        in
"+TimeElapsed(time_o));

            if(forAccuracy){
                if(withThreads){
                    System.out.println("Testing  Forest  for  Accuracy  with
threads");
                    ArrayList<DecisionTree>           Tree1           =
(ArrayList<DecisionTree>) trees2.clone();
                    TestforAccuracy(Tree1,testdata,data);
                }else{
                    System.out.println("Testing  Forest  for  Accuracy  without
threads");
                    ArrayList<DecisionTree>           Tree2           =
(ArrayList<DecisionTree>) trees2.clone();
                    TestForestForAccuracy(Tree2, data, testdata);
                }

            }else{
                if(withThreads){
                    System.out.println("Testing   Forest   for   Labels   with
threads");
                    ArrayList<DecisionTree>           Tree3           =
(ArrayList<DecisionTree>) trees2.clone();
                    TestForestForLabelWT(Tree3, data, testdata);
                }else{
                    System.out.println("Testing  Forest  for  Labels  without
threads");
                    ArrayList<DecisionTree>           Tree4           =
(ArrayList<DecisionTree>) trees2.clone();
```

```
                        TestForestForLabel(Tree4, data, testdata);
                    }
                }
            }


        private                    void                    TestforAccuracy(ArrayList<DecisionTree>
trees,ArrayList<ArrayList<String>> Testdata,ArrayList<ArrayList<String>> TrainData)
{
                long time2 = System.currentTimeMillis();
                ExecutorService                          TestthreadPool                              =
Executors.newFixedThreadPool(NUM_THREADS);

                for(ArrayList<String> TP:Testdata){
                        TestthreadPool.execute(new TestTree(TP,trees,TrainData));
                }TestthreadPool.shutdown();
                try{
                        TestthreadPool.awaitTermination(10, TimeUnit.SECONDS);
                }catch(InterruptedException ignored){
                        System.out.print("Interuption in testing");
                }System.out.println("Testing Complete");

                System.out.println("Results are ...");
                System.out.println("Forest                    Accuracy                    is
"+((corretsPredictions*100)/Testdata.size())+"%");
                System.out.println("this test was done in "+TimeElapsed(time2));
                System.out.println("");System.out.println("");

        }
```

```
	private void TestForestForLabel(ArrayList<DecisionTree>
trees,ArrayList<ArrayList<String>> traindata,ArrayList<ArrayList<String>> testdata) {
		// TODO Auto-generated method stub
		long time = System.currentTimeMillis();
		int treee=1;
		System.out.println("Predicting Labels now");
		for(DecisionTree DTC : trees){
			DTC.CalculateClasses(traindata, testdata, treee);treee++;
			if(DTC.predictions!=null)
			Prediction.add(DTC.predictions);
		}
		for(int i = 0;i<testdata.size();i++){
			ArrayList<String> Val = new ArrayList<String>();
			for(int j=0;j<trees.size();j++){
				Val.add(Prediction.get(j).get(i));
			}
			String pred = ModeofList(Val);
			System.out.println("["+pred+"]: Class predicted for data point:
"+i+1);
		}
		System.out.println("this test was done in "+TimeElapsed(time));
	}

	private void TestForestForLabelWT(ArrayList<DecisionTree>
tree,ArrayList<ArrayList<String>> traindata,ArrayList<ArrayList<String>> testdata) {
		long time = System.currentTimeMillis();
		ExecutorService TestthreadPool =
Executors.newFixedThreadPool(NUM_THREADS);int i=1;
		for(ArrayList<String> TP:testdata){
			TestthreadPool.execute(new
TestTreeforLabel(TP,tree,traindata,i));i++;
```

```
        }TestthreadPool.shutdown();
        try{
            TestthreadPool.awaitTermination(10, TimeUnit.SECONDS);
        }catch(InterruptedException ignored){
            System.out.print("Interuption in testing");
        }
        System.out.println("Testing Complete");
        System.out.println("this test was done in "+TimeElapsed(time));
    }

    public void TestForestForAccuracy(ArrayList<DecisionTree> trees,ArrayList<ArrayList<String>> train,ArrayList<ArrayList<String>> test){
        long time = System.currentTimeMillis();
        int correctness=0;ArrayList<String> ActualValues = new ArrayList<String>();

        for(ArrayList<String> s:test){
            ActualValues.add(s.get(s.size()-1));
        }int treee=1;
        System.out.println("Testing forest now ");

        for(DecisionTree DTC : trees){
            DTC.CalculateClasses(train, test, treee);treee++;
            if(DTC.predictions!=null)
            Prediction.add(DTC.predictions);
        }
        for(int i = 0;i<test.size();i++){
            ArrayList<String> Val = new ArrayList<String>();
            for(int j=0;j<trees.size();j++){
                Val.add(Prediction.get(j).get(i));
            }
```

```
                        String pred = ModeofList(Val);
                        if(pred.equalsIgnoreCase(ActualValues.get(i))){
                                correctness = correctness +1;
                        }
                }
                System.out.println("The Result of Predictions :-");
                System.out.println("Total Cases : "+test.size());
                System.out.println("Total CorrectPredicitions  : "+correctness);
                System.out.println("Forest                                        Accuracy
:"+(correctness*100/test.size())+"%");
                System.out.println("this test was done in "+TimeElapsed(time));
        }


        public String ModeofList(ArrayList<String> predictions) {
                // TODO Auto-generated method stub
                String MaxValue = null; int MaxCount = 0;
                for(int i=0;i<predictions.size();i++){
                        int count=0;
                        for(int j=0;j<predictions.size();j++){

        if(predictions.get(j).trim().equalsIgnoreCase(predictions.get(i).trim()))
                                        count++;
                                if(count>MaxCount){
                                        MaxValue=predictions.get(i);
                                        MaxCount=count;
                                }
                        }
                }return MaxValue;
        }
```

```
	private class CreateTree implements Runnable{
		private ArrayList<ArrayList<String>> data;
		private RandomForest forest;
		private int treenum;

		public    CreateTree(ArrayList<ArrayList<String>>    data,RandomForest
forest,int num){
			this.data=data;
			this.forest=forest;
			this.treenum=num;
		}

		public void run() {
			//trees.add(new DTreeCateg(data,forest,treenum));
			trees2.add(new DecisionTree(data, forest, treenum));
			progress+=update;
		}
	}

	public class TestTree implements Runnable{

		public ArrayList<String> testrecord;
		public ArrayList<DecisionTree> Trees;
		public ArrayList<ArrayList<String>> trainData;

		public   TestTree(ArrayList<String>   testpoint,   ArrayList<DecisionTree>
Dtrees, ArrayList<ArrayList<String>> train){

			this.testrecord = testpoint;
			this.Trees = Dtrees;
			this.trainData = train;
	}

	@Override
	public void run() {
		//System.out.print("Testing...");
		ArrayList<String> predictions = new ArrayList<String>();

		for(DecisionTree DT:Trees){
			String Class = DT.Evaluate(testrecord, trainData);
			if(Class == null)
				predictions.add("n/a");
			else
				predictions.add(Class);
		}

		String finalClass = ModeofList(predictions);
		if(finalClass.equalsIgnoreCase(testrecord.get(M)))
			corretsPredictions++;
		//System.out.println(finalClass);
		FinalPredictions.put(testrecord,finalClass);
	}
}


public class TestTreeforLabel implements Runnable{

	public ArrayList<String> testrecord;
	public ArrayList<DecisionTree> Trees;
	public ArrayList<ArrayList<String>> trainData;
	public int point;
```

```
		public TestTreeforLabel(ArrayList<String> dp, ArrayList<DecisionTree>
dtree, ArrayList<ArrayList<String>> data,int i){
			this.testrecord = dp;
			this.Trees = dtree;
			this.trainData = data;
			this.point =i;
		}

		@Override
		public void run() {
			ArrayList<String> predictions = new ArrayList<String>();

			for(DecisionTree DT:Trees){
				String Class = DT.Evaluate(testrecord, trainData);
				if(Class == null)
					predictions.add("n/a");
				else
					predictions.add(Class);
			}

			String finalClass = ModeofList(predictions);
			System.out.println("["+finalClass+"]: Class predicted for data
point: "+point);
		}
	}
	private void StartTimer(){
		time_o=System.currentTimeMillis();
	}


	private static String TimeElapsed(long timeinms){
		double s=(double)(System.currentTimeMillis()-timeinms)/1000;
		int h=(int)Math.floor(s/((double)3600));
		s-=(h*3600);
		int m=(int)Math.floor(s/((double)60));
		s-=(m*60);
		return ""+h+"hr "+m+"m "+s+"sec";
	}

}
```

Random forest categ:

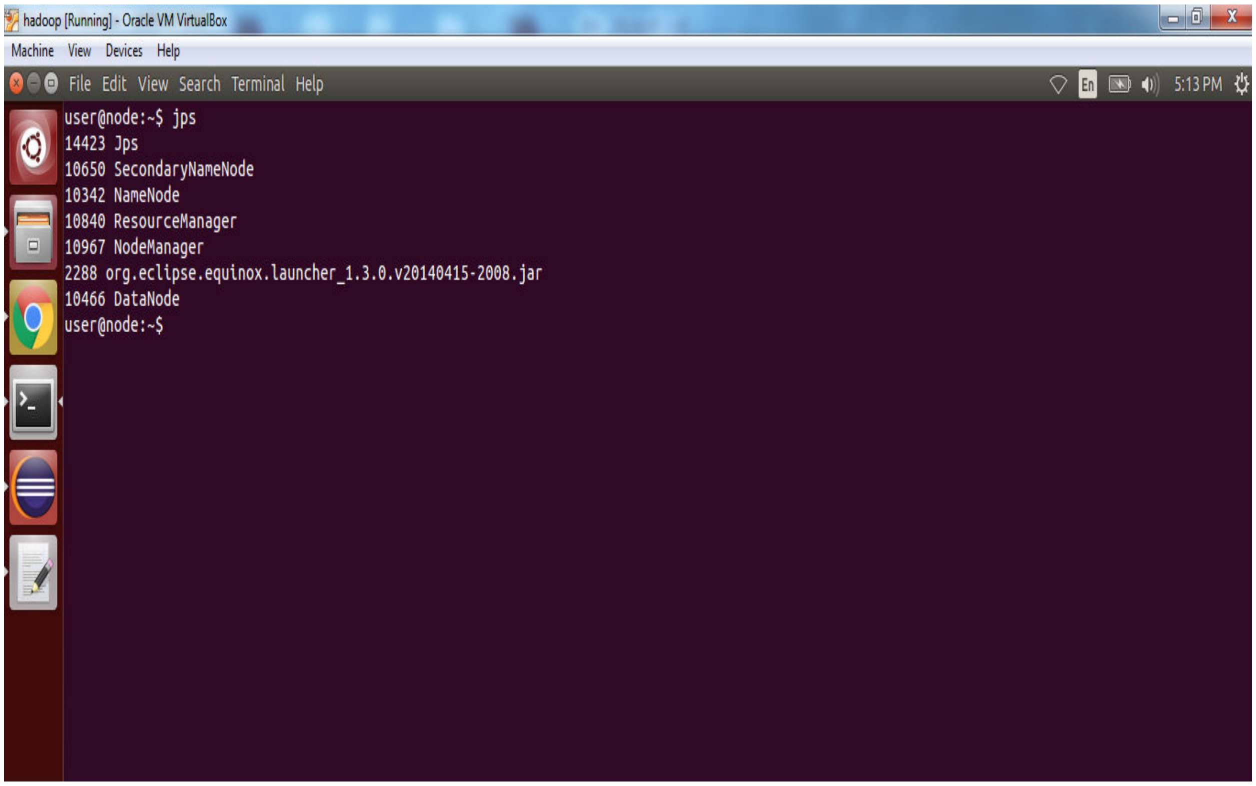
```
package com.rf.real.categ;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;


public class RandomForestCateg {

	private static final int NUM_THREADS=Runtime.getRuntime().availableProcessors();
	public static int C;
	public static int M;
	public	static	int	Ms;//recommended	by	Breiman:
=(int)Math.round(Math.log(M)/Math.log(2)+1);
	private ArrayList<DTreeCateg> trees;
	private ArrayList<DTreeCateg2> trees2;
	private long time_o;
	private int numTrees;
	private double update;
	private double progress;
private int[] importances;
	private HashMap<int[],int[]> estimateOOB;
	private double error;
	private ExecutorService treePool;
	private ArrayList<ArrayList<String>> data;
private ArrayList<ArrayList<String>> testdata;
	private ArrayList<ArrayList<String>> Prediction;

	public ArrayList<Integer> TrainAttributes;
	public ArrayList<Integer> TestAttributes;



	public	RandomForestCateg(int	numTrees,int	M,int	Ms,int	C,
ArrayList<ArrayList<String>> train,ArrayList<ArrayList<String>> test) {
		// TODO Auto-generated constructor stub
		StartTimer();
		this.numTrees=numTrees;
		this.data=train;
		this.testdata=test;
		this.M=M;
		this.Ms=Ms;
		this.C=C;
		this.TrainAttributes=GetAttributes(train);
		this.TestAttributes=GetAttributes(test);
		trees = new ArrayList<DTreeCateg>(numTrees);
		trees2 = new ArrayList<DTreeCateg2>(numTrees);
		update=100/((double)numTrees);
		progress=0;
		System.out.println("creating "+numTrees+" trees in a random Forest. . . ");
		System.out.println("total data size is "+train.size());
		System.out.println("number of attributes "+M);
		System.out.println("number of selected attributes "+Ms);

		estimateOOB=new HashMap<int[],int[]>(data.size());
		Prediction = new ArrayList<ArrayList<String>>();
	}
	/**
	 * Begins the random forest creation
	 */
	public void Start() {
		// TODO Auto-generated method stub
		System.out.println("Number of threads started : "+NUM_THREADS);
		System.out.println("Starting trees");
		treePool=Executors.newFixedThreadPool(NUM_THREADS);
		for (int t=0;t<numTrees;t++){
			treePool.execute(new CreateTree(data,this,t+1));
		}treePool.shutdown();
		try {

	treePool.awaitTermination(Long.MAX_VALUE,TimeUnit.SECONDS);
//effectively infinity
	      } catch (InterruptedException ignored){
		System.out.println("interrupted exception in Random Forests");
```

```
        }
            if(data.get(0).size()>testdata.get(0).size()){
                //TestForestNoLabel2(trees2, data, testdata);
                TestForestNoLabel(trees,data,testdata);
            }
            else if(data.get(0).size()==testdata.get(0).size()){
                TestForest2(trees2, data, testdata);
                //TestForest(trees,data,testdata);
            }

            else
                System.out.println("Cannot test this data");

            System.out.print("Done in "+TimeElapsed(time_o));
    }

    private void TestForestNoLabel(ArrayList<DTreeCateg>
trees2,ArrayList<ArrayList<String>> data2,ArrayList<ArrayList<String>> testdata2) {
            // TODO Auto-generated method stub
            ArrayList<String> TestResult = new ArrayList<String>();
            System.out.println("Predicting Labels now");
            for(ArrayList<String> DP:testdata2){
                ArrayList<String> Predict = new ArrayList<String>();
                for(DTreeCateg DT:trees2){
                    Predict.add(DT.Evaluate(DP, testdata2));
                }
                TestResult.add(ModeofList(Predict));
            }
    }
```

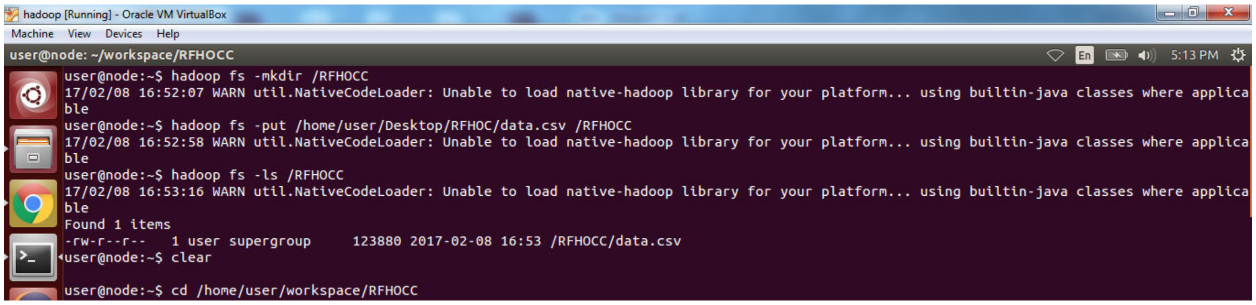
```
	public void TestForest(ArrayList<DTreeCateg> trees,ArrayList<ArrayList<String>> train,ArrayList<ArrayList<String>> test){
		int correctness=0;ArrayList<String> ActualValues = new ArrayList<String>();

		for(ArrayList<String> s:test){
			ActualValues.add(s.get(s.size()-1));
		}int treee=1;
		System.out.println("Testing forest now ");

		for(DTreeCateg DTC : trees){
			DTC.CalculateClasses(train, test, treee);treee++;
			if(DTC.predictions!=null)
			Prediction.add(DTC.predictions);
		}
		for(int i = 0;i<test.size();i++){
			ArrayList<String> Val = new ArrayList<String>();
			for(int j=0;j<trees.size();j++){
				Val.add(Prediction.get(j).get(i));
			}
			String pred = ModeofList(Val);
			if(pred.equalsIgnoreCase(ActualValues.get(i))){
				correctness = correctness +1;
			}
		}
		System.out.println("The Result of Predictions :-");
		System.out.println("Total Cases : "+test.size());
		System.out.println("Total CorrectPredicitions  : "+correctness);
		System.out.println("Forest Accuracy :"+(correctness*100/test.size())+"%");
	}
	private	void	TestForestNoLabel2(ArrayList<DTreeCateg2>
trees22,ArrayList<ArrayList<String>> data2,ArrayList<ArrayList<String>> testdata2) {
		// TODO Auto-generated method stub
		ArrayList<String> TestResult = new ArrayList<String>();
		System.out.println("Predicting Labels now");
		for(ArrayList<String> DP:testdata2){
			ArrayList<String> Predict = new ArrayList<String>();
			for(DTreeCateg2 DT:trees22){
				Predict.add(DT.Evaluate(DP, testdata2));
			}
			TestResult.add(ModeofList(Predict));
		}
	}

	public	void	TestForest2(ArrayList<DTreeCateg2>
trees,ArrayList<ArrayList<String>> train,ArrayList<ArrayList<String>> test){
		int	correctness=0;ArrayList<String>	ActualValues	=	new
ArrayList<String>();

		for(ArrayList<String> s:test){
			ActualValues.add(s.get(s.size()-1));
		}int treee=1;
		System.out.println("Testing forest now ");

		for(DTreeCateg2 DTC : trees){
			DTC.CalculateClasses(train, test, treee);treee++;
			if(DTC.predictions!=null)
			Prediction.add(DTC.predictions);
		}
		for(int i = 0;i<test.size();i++){
				ArrayList<String> Val = new ArrayList<String>();
				for(int j=0;j<trees.size();j++){
					Val.add(Prediction.get(j).get(i));
				}
				String pred = ModeofList(Val);
				if(pred.equalsIgnoreCase(ActualValues.get(i))){
					correctness = correctness +1;
				}
			}
			System.out.println("The Result of Predictions :-");
			System.out.println("Total Cases : "+test.size());
			System.out.println("Total CorrectPredicitions  : "+correctness);
			System.out.println("Forest                                        Accuracy
:"+(correctness*100/test.size())+"%");
		}

		public String ModeofList(ArrayList<String> predictions) {
			// TODO Auto-generated method stub
			String MaxValue = null; int MaxCount = 0;
			for(int i=0;i<predictions.size();i++){
				int count=0;
				for(int j=0;j<predictions.size();j++){

		if(predictions.get(j).trim().equalsIgnoreCase(predictions.get(i).trim()))
						count++;
					if(count>MaxCount){
						MaxValue=predictions.get(i);
						MaxCount=count;
					}
				}
			}return MaxValue;
```

```
	}

	private class CreateTree implements Runnable{
		private ArrayList<ArrayList<String>> data;
		private RandomForestCateg forest;
private int treenum;

		public				CreateTree(ArrayList<ArrayList<String>>
data,RandomForestCateg forest,int num){
			this.data=data;
			this.forest=forest;
			this.treenum=num;
		}

		public void run() {
			//trees.add(new DTreeCateg(data,forest,treenum));
			trees2.add(new DTreeCateg2(data, forest, treenum));
			progress+=update;
		}
	}

	private void StartTimer(){
		time_o=System.currentTimeMillis();
	}

	private static String TimeElapsed(long timeinms){
		double s=(double)(System.currentTimeMillis()-timeinms)/1000;
		int h=(int)Math.floor(s/((double)3600));
		s-=(h*3600);
		int m=(int)Math.floor(s/((double)60));
		s-=(m*60);
		return ""+h+"hr "+m+"m "+s+"sec";
	}

	private boolean isAlphaNumeric(String s){
		char c[]=s.toCharArray();boolean hasalpha=false;
		for(int j=0;j<c.length;j++){
			hasalpha = Character.isLetter(c[j]);
			if(hasalpha)break;
		}return hasalpha;
	}

	private ArrayList<Integer> GetAttributes(List<ArrayList<String>> data){
		ArrayList<Integer> Attributes = new ArrayList<Integer>();int iter = 0;
		ArrayList<String> DataPoint = data.get(iter);
		if(DataPoint.contains("n/a") || DataPoint.contains("N/A")){
			iter = iter +1;
			DataPoint = data.get(iter);
		}
		for(int i =0;i<DataPoint.size();i++){
			if(isAlphaNumeric(DataPoint.get(i)))
				Attributes.add(1);
			else
				Attributes.add(0);
		}
		return Attributes;
	}
}
```

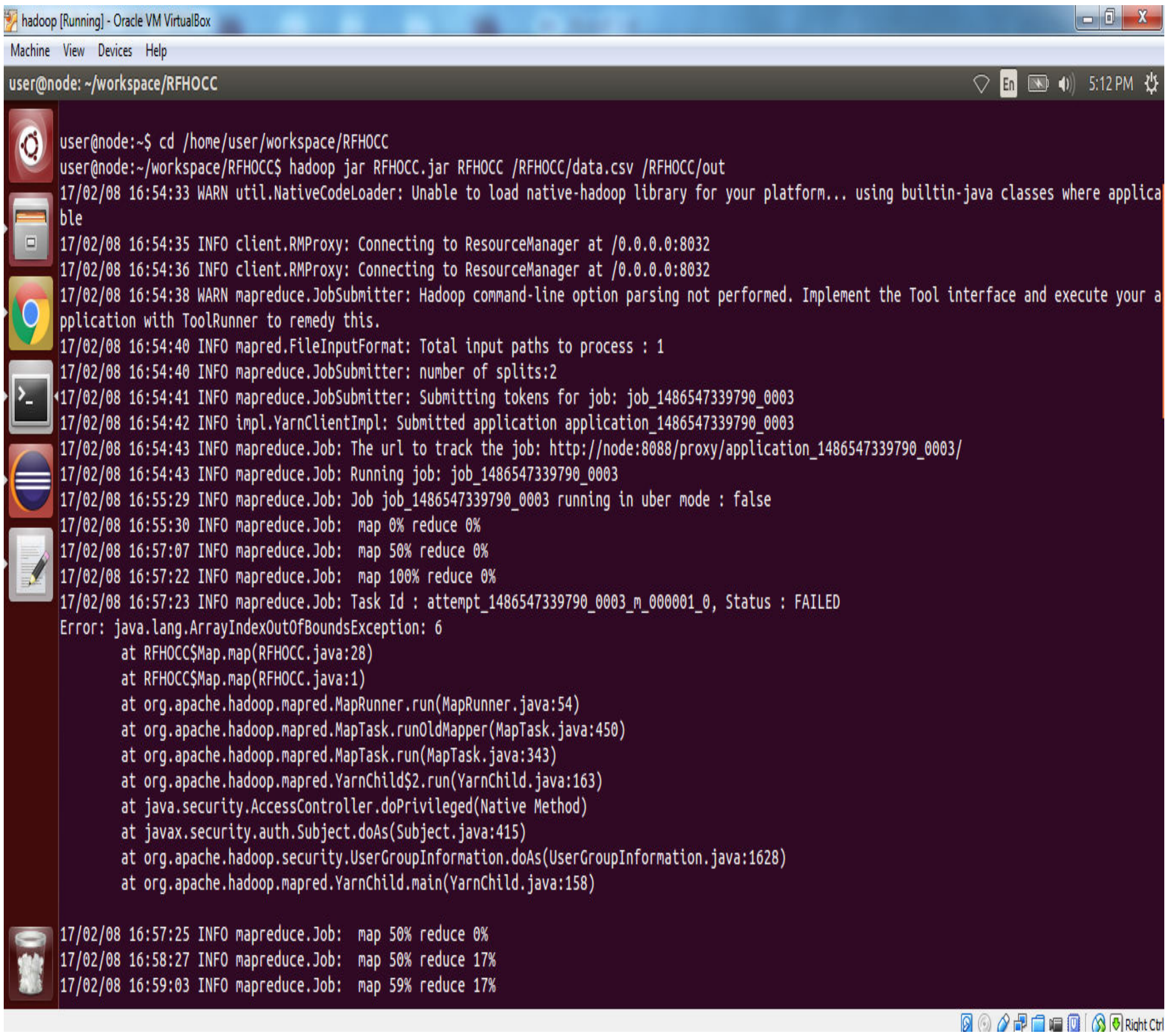
7. RESULTS


hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
File Edit View Search Terminal Help
5:13 PM

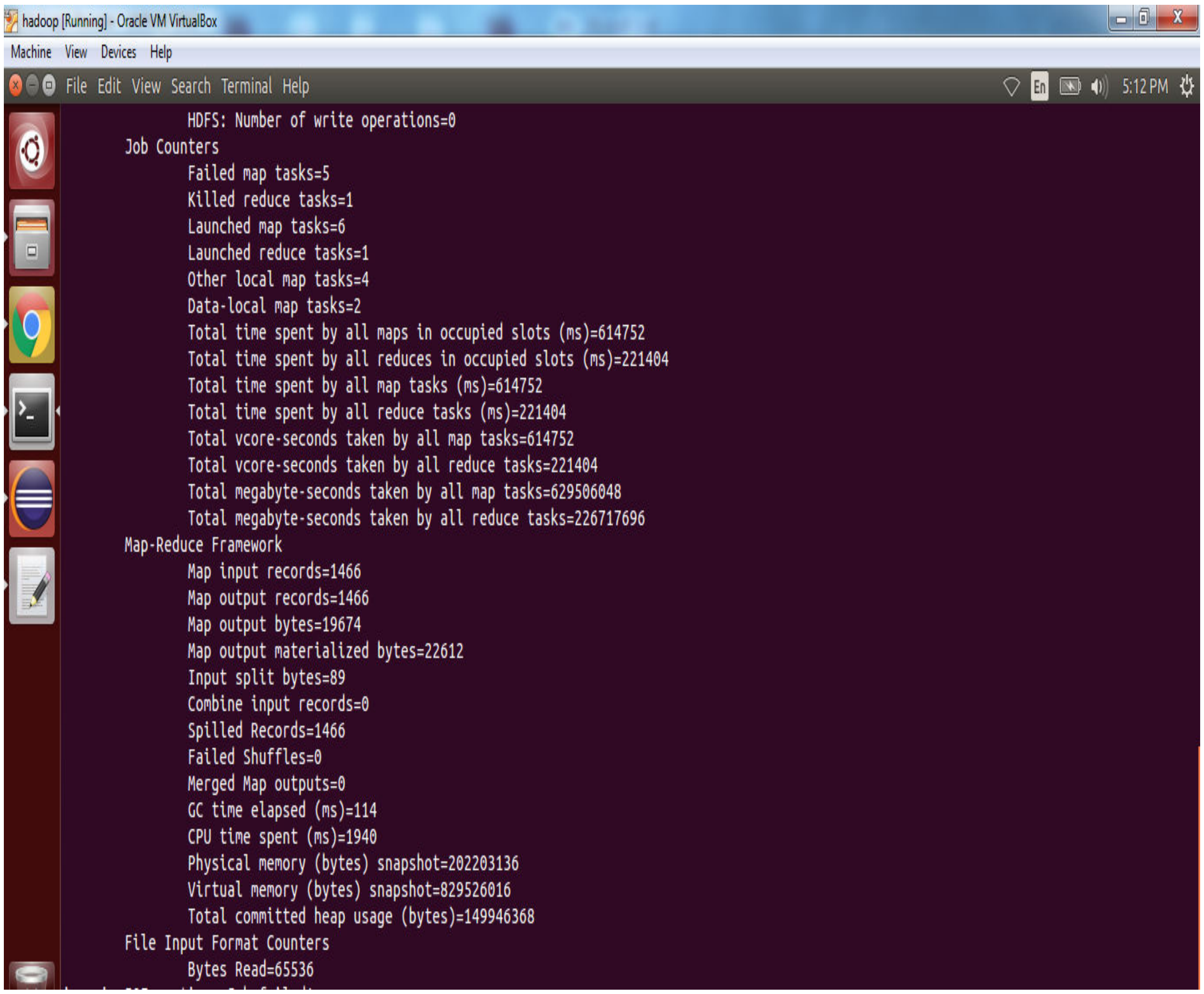
```
user@node:~$ jps
14423 Jps
10650 SecondaryNameNode
10342 NameNode
10840 ResourceManager
10967 NodeManager
2288 org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar
10466 DataNode
user@node:~$
```


hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
user@node: ~/workspace/RFHOCC
5:13 PM

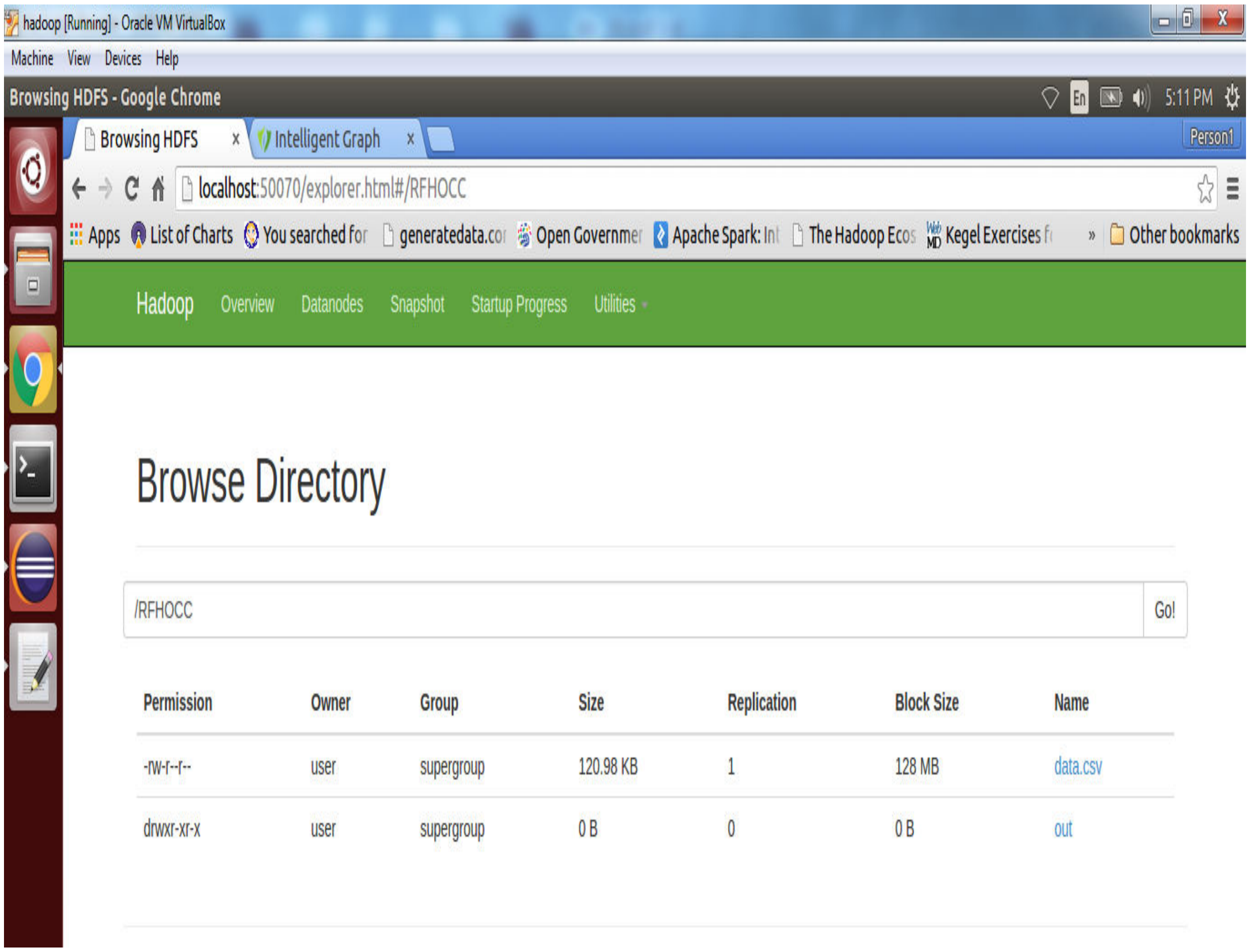
```
user@node:~$ hadoop fs -mkdir /RFHOCC
17/02/08 16:52:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
user@node:~$ hadoop fs -put /home/user/Desktop/RFHOC/data.csv /RFHOCC
17/02/08 16:52:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
user@node:~$ hadoop fs -ls /RFHOCC
17/02/08 16:53:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
Found 1 items
-rw-r--r--   1 user supergroup     123880 2017-02-08 16:53 /RFHOCC/data.csv
user@node:~$ clear

user@node:~$ cd /home/user/workspace/RFHOCC
```


hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
user@node: ~/workspace/RFHOCC
5:12 PM
Right Ctrl

```
user@node:~$ cd /home/user/workspace/RFHOCC
user@node:~/workspace/RFHOCC$ hadoop jar RFHOCC.jar RFHOCC /RFHOCC/data.csv /RFHOCC/out
17/02/08 16:54:33 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
17/02/08 16:54:35 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/02/08 16:54:36 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/02/08 16:54:38 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your a
pplication with ToolRunner to remedy this.
17/02/08 16:54:40 INFO mapred.FileInputFormat: Total input paths to process : 1
17/02/08 16:54:40 INFO mapreduce.JobSubmitter: number of splits:2
17/02/08 16:54:41 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1486547339790_0003
17/02/08 16:54:42 INFO impl.YarnClientImpl: Submitted application application_1486547339790_0003
17/02/08 16:54:43 INFO mapreduce.Job: The url to track the job: http://node:8088/proxy/application_1486547339790_0003/
17/02/08 16:54:43 INFO mapreduce.Job: Running job: job_1486547339790_0003
17/02/08 16:55:29 INFO mapreduce.Job: Job job_1486547339790_0003 running in uber mode : false
17/02/08 16:55:30 INFO mapreduce.Job:  map 0% reduce 0%
17/02/08 16:57:07 INFO mapreduce.Job:  map 50% reduce 0%
17/02/08 16:57:22 INFO mapreduce.Job:  map 100% reduce 0%
17/02/08 16:57:23 INFO mapreduce.Job: Task Id : attempt_1486547339790_0003_m_000001_0, Status : FAILED
Error: java.lang.ArrayIndexOutOfBoundsException: 6
        at RFHOCC$Map.map(RFHOCC.java:28)
        at RFHOCC$Map.map(RFHOCC.java:1)
        at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:54)
        at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:450)
        at org.apache.hadoop.mapred.MapTask.run(MapTask.java:343)
        at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:163)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:415)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1628)
        at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:158)

17/02/08 16:57:25 INFO mapreduce.Job:  map 50% reduce 0%
17/02/08 16:58:27 INFO mapreduce.Job:  map 50% reduce 17%
17/02/08 16:59:03 INFO mapreduce.Job:  map 59% reduce 17%
```


hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
File Edit View Search Terminal Help
5:12 PM

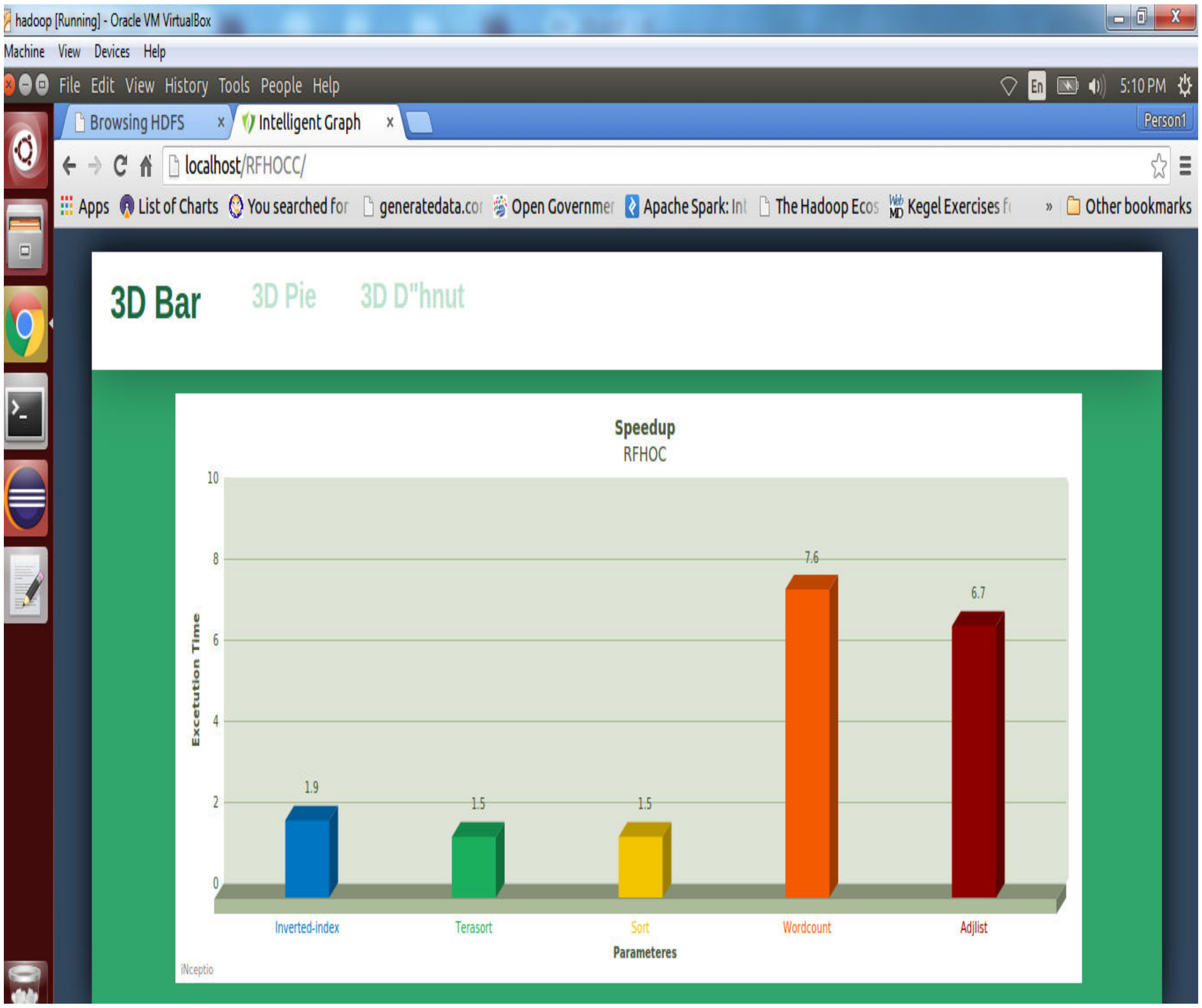
```
		HDFS: Number of write operations=0
	Job Counters
		Failed map tasks=5
		Killed reduce tasks=1
		Launched map tasks=6
		Launched reduce tasks=1
		Other local map tasks=4
		Data-local map tasks=2
		Total time spent by all maps in occupied slots (ms)=614752
		Total time spent by all reduces in occupied slots (ms)=221404
		Total time spent by all map tasks (ms)=614752
		Total time spent by all reduce tasks (ms)=221404
		Total vcore-seconds taken by all map tasks=614752
		Total vcore-seconds taken by all reduce tasks=221404
		Total megabyte-seconds taken by all map tasks=629506048
		Total megabyte-seconds taken by all reduce tasks=226717696
	Map-Reduce Framework
		Map input records=1466
		Map output records=1466
		Map output bytes=19674
		Map output materialized bytes=22612
		Input split bytes=89
		Combine input records=0
		Spilled Records=1466
		Failed Shuffles=0
		Merged Map outputs=0
		GC time elapsed (ms)=114
		CPU time spent (ms)=1940
		Physical memory (bytes) snapshot=202203136
		Virtual memory (bytes) snapshot=829526016
		Total committed heap usage (bytes)=149946368
	File Input Format Counters
		Bytes Read=65536
```

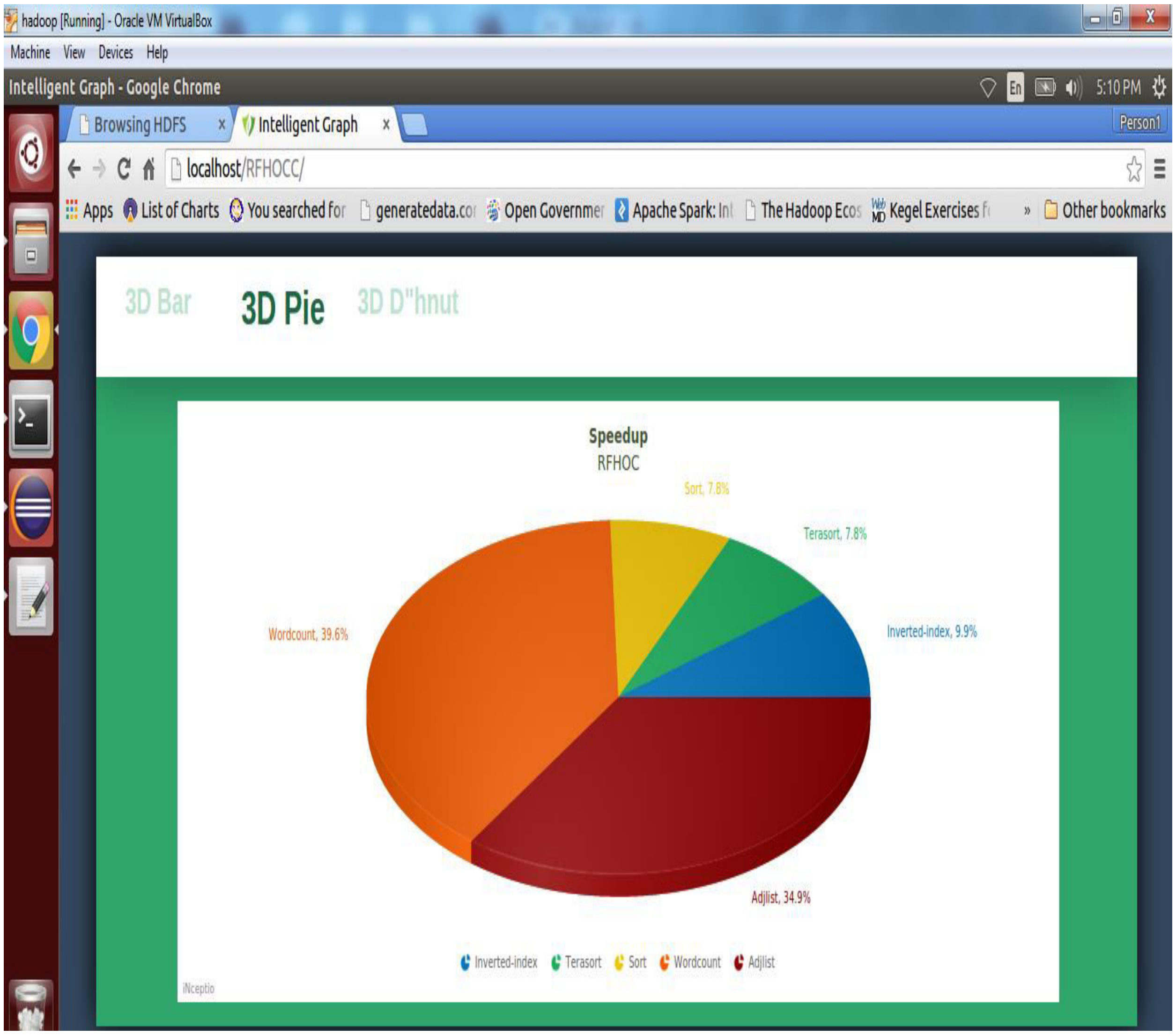

hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
Browsing HDFS - Google Chrome
Browsing HDFS
Intelligent Graph
localhost:50070/explorer.html#/RFHOCC
Hadoop Overview Datanodes Snapshot Startup Progress Utilities
Go!

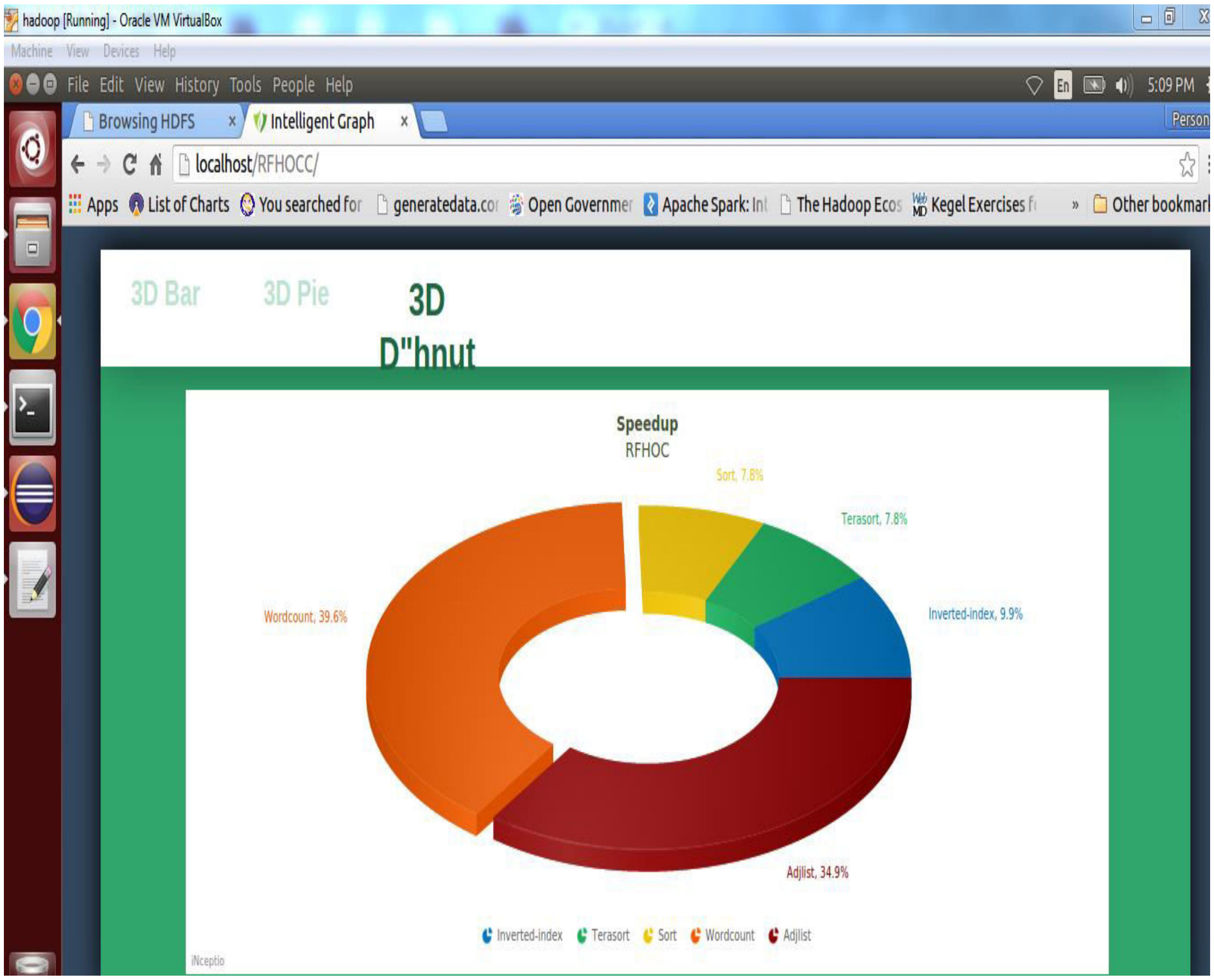
Browse Directory

/RFHOCC

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	user	supergroup	120.98 KB	1	128 MB	data.csv
drwxr-xr-x	user	supergroup	0 B	0	0 B	out


hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
File Edit View History Tools People Help
Browsing HDFS
Intelligent Graph
localhost/RFHOCC/
3D Bar
3D Pie
3D D"hnut
Speedup
RFHOC
Excetution Time
Parameteres
Inverted-index
Terasort
Sort
Wordcount
Adjlist
1.9
1.5
1.5
7.6
6.7


hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
Intelligent Graph - Google Chrome
5:10 PM
Browsing HDFS
Intelligent Graph
localhost/RFHOCC/
3D Bar
3D Pie
3D D"hnut
Speedup
RFHOC
Sort, 7.8%
Terasort, 7.8%
Inverted-index, 9.9%
Wordcount, 39.6%
Adjlist, 34.9%
Inverted-index
Terasort
Sort
Wordcount
Adjlist


hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
File Edit View History Tools People Help
Browsing HDFS
Intelligent Graph
localhost/RFHOCC/
3D Bar
3D Pie
3D D"hnut
Speedup
RFHOC
Sort, 7.8%
Terasort, 7.8%
Inverted-index, 9.9%
Wordcount, 39.6%
Adjlist, 34.9%
Inverted-index
Terasort
Sort
Wordcount
Adjlist

8. 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.

8.1 TYPES OF TESTING

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should bSe 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.

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.

Functional test

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 and the effective value of current tests is determined.

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.

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.

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.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

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

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

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.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encounte

TEST CASES:

TestCase Id	Test Case Name	Test Case Desc	Test Steps			Test Case Status	Test Priority
			Step	**Expected**	**Actual**		
Define Reducers 01	Reducer location details	It defines the reducers particular website details by providing relevance coefficient similarities.	If I doesn’t provide relevance coefficient similarities of each page.	website details will not be saved	Reducers details will be saved successfully	High	High

Reducer 1 &2 02	Run reducers	Start the reducer nodes ,and all details will be updated at reducer node	If I not run the applicatio n	Reducer don’t know the updated details	Reducer node will be started	High	High
Upload 03	Upload the input data	Data will be uploaded from shuffle phase	If I can’t upload the data	I can’t reduce the relevance coefficient similarities	Input data loaded successfully	High	High
Start Mapreduce For Auto-Tuning 04	Auto-Tuning using Mapredu ce	It Auto tunes all the partitioned data	If I not start the Auto-tuning	I can’t reduce the relevance coefficient similarities	After processing the Auto-Tuning data, it displays the count result.	High	High
Graph 05	Random Forest Auto Tuning graph	Displays the graph better processing time & Technique	If I can’t do any Auto-Tuning	Nothing will be displayed	Graph will be displayed using Auto-tuning Processed data	High	High

9. CONCLUSION

Performance tuning is a challenging problem for Hadoop/Map Reduce workloads because of the large number of Hadoop configuration parameters. Yet, there is a significant opportunity for optimizing performance beyond Hadoop's default runtime configuration parameters. Previously proposed techniques to automatically tune the Hadoop configuration parameters build analytical models based on oversimplified assumptions, affecting the overall model's accuracy and ultimately the achievable performance improvements.

In this project, we propose RFHOC, a novel methodology to optimize Hadoop performance by leveraging the notion of a random forest to build accurate and robust performance prediction models for the phases of the map and reduce stage of a Hadoop program of interest. Taking the output of these models as the input to a genetic algorithm to automatically search the Hadoop configuration space yields a Hadoop configuration setting that leads to optimized application performance. We evaluate RFHOC using five Hadoop benchmarks, each with five input data sets ranging from 50 GB to 1 TB.

The results show that RFHOC speeds up Hadoop programs significantly over the previously proposed cost-based optimization approach. The speedups achieved are significant: by 2.11_ on average and up to 7.4_. Furthermore, we find RFHOC's performance benefits to increase with increasing input data set sizes.

10. FUTURE ENHANCEMENT

Abstract Random Forest is an ensemble supervised machine learning technique, based on bagging and random feature selection, number of decision trees (base classifier) is generated and majority voting is taken for classification. In this paper we are presenting heuristic based on improvements towards effective learning of random forest classifier. These efforts include disjoint partitions of datasets for learning of base trees, reducing depth of base trees by avoiding repetitive selection of attributes, and selecting smaller subsets of attributes for split at each node. The results of our work and encouraging there is future research scope in this direction.

11. BIBILIOGRAPHY

[1] L. Lie, “Heuristic artificial intelligent algorithm for genetic algorithm,” Key Eng. Materials, vol. 439, pp. 516–521, 2010.

[2] K. S. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Eltabakh, C.-C. Kanne, F. Ozcan, and E. J. Shekita, “Jaql: A scripting language for large scale semistructured data analysis,” in Proc. VLDB Conf., Sep. 2011, pp. 1272–1283.

[3] M.-P. Wen, H.-Y. Lin, A.-P. Chen, and C. Yang, “An integrated home financial investment learning environment applying cloud computing in social network analysis,” in Proc. Int. Conf. Adv. Social Netw. Anal. Mining, Jul. 2011, pp. 751–754.

[4] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, “Starfish: A self-tuning system for big data analytics,” in Proc. Biennial Int. Conf. Innovative Data Syst. Res., Jan. 2011, pp. 261–272.

[5] H. Herodotou and S. Babu, “Profiling, What-if analysis, and cost based optimization of MapReduce programs,” Proc. VLDB Endowment, vol. 4, no. 11, pp. 1111–1122, 2011.

[6] P. Lama and X. Zhou, “Aroma: Automated resource allocation and configuration of MapReduce environment in the cloud,” in Proc. 9th ACM Int. Conf. Autonomic Comput., Sep. 2012, pp. 63–72.

[7] L. Breiman, “Random forests,” Mach. Learning, vol. 45, no. 1, pp. 5–32, 2001.

[8] B. Efron and R. J. Tibshirani, An Introduction to the Bootstrap, vol. 57. Boca Raton, FL, USA: CRC Press, 1994.

.