A

Report on

<<Title of the project>>

Thesis submitted in partial fulfillment of the requirement for the award of the degree of

**Master of Technology**

**In**

**<<Branch Name>>**

Submitted By

<< Name of the Student >>

<< Hall Ticket No >>

*Under the Esteemed Guidance of*

<< Guide Name >>

<< Designation >>

<< College Logo >>

Department of << Department name >>,

<< College name (Upper case only) >>

(Affiliated to Jawaharlal Nehru Technological University)

<<Distinct Name >>-Pin code, << State name>> INDIA

2015

**<< College Name >>**

**(Affiliated to Jawaharlal Nehru Technological University)**

**<< Distinct Name >>- << Pin code >>. << State name >>., INDIA**

**CERTIFICATE**

**<< College Logo >>**

This is to certify that the Thesis entitled “**<<**title of the project**>>”** is being submitted by << student name>>bearing **<<**Roll number**>>** in partial fulfilment of the requirement for the award of the degree of M.Tech in << **Branch Name >>** to the <<College Name>>, <<University Name >> is a record of bonafide work carried out by him/her under my guidance. The results presented in this thesis have been verified and are found to be satisfactory. The results embodied in this thesis have not been submitted to any other University for the award of any other degree or diploma.

**Project Guide**  **Head of the Department**

**<<Guide Name>>, <<HOD Name>>**

**<<Designation>> <<Designation>>**

**EXTERNAL PRINCIPAL**

**Next page:** This page is required for the candidates who pursue the project work outside (other

organization), under the supervision of an **EXTERNAL GUIDE.** This page is another certificate

given on the organization letterhead where the project work is pursued, which should be certified

and duly signed by the External guide about work done by the candidate and clearly mentioning the duration of the project work. The format for this page will almost be the same as the previous page with double spacing between the lines.

**(or)**

Include the Declaration page in the below specified Format:

**DECLARATION**

I **<<Student Name>>** bearing **Roll Number <<Roll No>>**, a bonafide student of <<**College Name >>**, would like to declare that the project titled **“<<project title>>”** a partial fulfillment of **M.Tech** Degree course of << **University Name >>** is my original work in the year 2015 under the guidance of **<< Guide Name>>,** <<**Designation>>** of the **<< Department Name >> Department**.

Date: **<<Student Name>>**

**(Roll Number)**

**Acknowledgement**

I would like to express my sincere gratitude to my advisor, **<<Guide Name>>**, whose knowledge and guidance has motivated me to achieve goals I never thought possible. He has consistently been a source of motivation, encouragement, and inspiration. The time I have spent working under his supervision has truly been a pleasure.

I thank H.O.D **<<HOD Name>>** for his effort and guidance and all senior faculty members of CSE Department for their help during my course. Thanks to programmers and non-teaching staff of C.S.E Department of VITS.

I Thank my principal **<<PRINCIPAL NAME>>** and Management for providing excellent facilities to carry out my project work.

Finally Special thanks to my parents for their support and encouragement throughout my life and this course. Thanks to all my friends and well wishers for their constant support.

**Names of Student (roll number)**

**Seventh page:** The seventh page may contain an **ABSTRACT** of the Thesis. The candidates may emphasize here his/her contributions.

**Page 8th, 9th …..:** In these pages candidate must provide a **TABLE OF CONTENTS**, list of tables and list of figures, Screens and notation.

**NOTE:** All the above pages are to be numbered in Roman numerals of lower case.

**Arrangement of chapters:** The following is the suggested format for arranging the Thesis matter into various chapters:

**1) Introduction** ( Do not copy as it ease from Base Paper)

**2) Literature Survey** ( Quantity should be 10 – 15 Pages)

**3) SRS** ( Preferably IEEE Format- Do not include Technology Desciption))

**4) Analysis and Design** (Preferably ER-Diagrams & DFD)

**5) Implementation** ( Architecture, Algorithms used, Module detailed desc)

**6) Testing** (Min. 10 test caes)

**7) Results** (Navigation should be provided)

**8) Conclusion and Future Scope** (In two paragraphs)

**9) Reference / Bibliography** (Format is given below)

**10) Appendices** (if any)

a) Standards (if any)

b) Sample Code(Optional)

**The arrangement of paragraph in a chapter:** Each paragraph in a chapter should be properly numbered for example: 2.1, 2.2, etc., where the first digit represents the chapter number and the second digit the paragraph number. There is no need to indicate the number for the first paragraph in a chapter.

Sub-paragraphs, if any, may be indicated as 1.1.1, 1.1.2, etc., i.e., the first digit representing the chapter, the second representing the paragraph and the third representing the sub-paragraph.

**Photographs and Tables:** The Figures and tables occurring in a chapter may be serially numbered as **Fig: 1.1, 1.2,** etc., where the first digit represents the chapter, the second digit represents the figure number. ( It should be specified below the figure)

The tables may be represented as **Tab 1.1, 1.2,** etc., where the first digit represents the chapter,

the second digit represents the table number. ( It should be specified above the Table)

**Graphs:** The graph should clearly indicate the pinots which are used for drawing the curve or

curves. All the letters in the graph should be written with stencils.

**Bibliography or References:** The following format may be used for writing the Bibliography /References.

` At the end of Thesis where the listing of references is done, the list should be made strictly in alphabetic order of the names of the author. The references/websites have to be listed in the following format:

**[S.No] Author, Paper/Book, Publisher/Magazine/Conference, Volume/Edition, page numbers,Year**

**Examples:**

[1] Bruce, Cryptography, Tata McGraw Hill, 1978

[2] R. R. Duncan, "Remediation of Lead in Water Supplies," IEEE Trans. Microwave Theory Tech.,vol. 99, no. 18, pp. 257-278, Nov. 1986.

[3] <http://www.google.com>

**Guidelines for preparing the Documentation for M.Tech.**

**Size and copies:** The Documentation should be submitted in A4 size and 5 copies (4 Hard copies + 1Soft copy – CD) of the Thesis.

**Paper, Typing and Format:** Bond paper should be used for the preparation of the Thesis. Typing should be done on the 12 point size letters for the running text, 14 point size for the sub-headings and 16 point size for main headings/titles/chapter names/etc. The font type should be preferably TIMES NEW ROMAN.

The layout margin to be provided is 1.5’ on the left, 1’ on the top and bottom, 1’ on the right.

Fresh paragraph should commence after tab space. 1.5 line spacing shall be provided throughout the Thesis.

The page number shall be indicated at the bottom right of the each page. Front Pages Small Roman Numbers (Excluding title page, Certificate page, Acknowledgement page), Body pages 1,2,3 ………., Annexure 1,2,3……….. (Separate for each Annexure).

**Binding**: The Thesis shall be properly bound using resin of **Cement gray color** (Cement Color) for M.Tech. The bound front cover should indicate in golden embossed letter the following:

**Guidelines for the project work of all PG courses:**

1. **Nature of the content**: The project work must be an evidence of the utilization and practical application of the knowledge / techniques learnt in the theory courses pertaining to the faculty / speciality.

2. **Quality and quantity:** Both the extent and the level / standard of the work carried out by the student must be greater than that of an undergraduate project and be commensurate with the depth of coverage and sophistication reflected in the theory courses of the Post Graduate specialization.

3. **Originality:** A mere collection and review, how comprehensive it may be of the previous works or the state-of-art cannot qualify as the project work, though it can form a part of it.

4. The project work must be the result of the theoretical and / or practical investigation actually carried out by the candidate.

5. It must reflect the originality of the student and mere duplication of earlier works does not suffice. However, extensions and modifications of earlier work(s) are acceptable if there is original contribution by the candidate.

6. It should be possible to publish at least one paper in a reviewed journal from the work presented in the thesis.

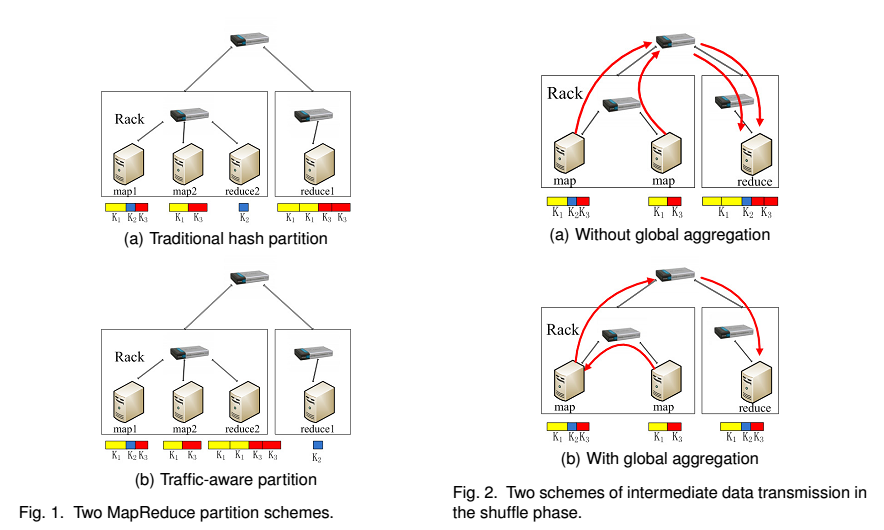
\* \* \*

**On Traffic-Aware Partition and Aggregation in MapReduce for Big Data Applications**

1. **Introduction**

MapReduce has emerged as the most popular computing framework for big data processing due to its simple programming model and automatic management  
of parallel execution. MapReduce and its open source implementation Hadoop have been adopted by leading companies, such as Yahoo!, Google and Facebook, for various big data applications, such as machine learning bioinformatics, and cybersecurity.  
MapReduce divides a computation into two main phases, namely map and reduce, which in turn are carried out by several map tasks and reduce tasks, respectively. In the map phase, map tasks are launched in parallel to convert the original input splits into intermediate data in a form of key/value pairs. These key/value pairs are stored on local machine and organized into multiple data partitions, one per reduce task. In the  
reduce phase, each reduce task fetches its own share of data partitions from all map tasks to generate the final result. There is a shuffle step between map and reduce phase. In this step, the data produced by the map phase are ordered, partitioned and transferred to  
the appropriate machines executing the reduce phase.

The resulting network traffic pattern from all map tasks to all reduce tasks can cause a great volume of network traffic, imposing a serious constraint on the efficiency  
of data analytic applications. For example, with tens of thousands of machines, data shuffling accounts for 58.6% of the cross-pod traffic and amounts to over 200 petabytes in total in the analysis of SCOPE jobs . For shuffle-heavy MapReduce tasks, the high traffic could incur considerable performance overhead up to 30-40 %  
as shown in. By default, intermediate data are shuffled according to a hash function in Hadoop, which would lead to large network traffic because it ignores network topology  
and data size associated with each key. As shown in Fig. 1, we consider a toy example with two map tasks and two reduce tasks, where intermediate data of three keys  
*K*1, *K*2, and *K*3 are denoted by rectangle bars under each machine. If the hash function assigns data of *K*1 and *K*3 to reducer 1, and *K*2 to reducer 2, a large amount  
of traffic will go through the top switch. To tackle this problem incurred by the traffic-oblivious partition scheme, we take into account of both task locations and data size associated with each key in this paper. By assigning keys with larger data size to reduce tasks closer to map tasks, network traffic can be significantly reduced. In the same example above, if we assign *K*1 and *K*3 to reducer 2, and *K*2 to reducer 1, as shown in  
Fig. 1(b), the data transferred through the top switch will be significantly reduced.



To further reduce network traffic within a MapReduce job, we consider to aggregate data with the same keys before sending them to remote reduce tasks. Although a similar function, called combiner, has been already adopted by Hadoop, it operates immediately after a map task solely for its generated data, failing to exploit the  
data aggregation opportunities among multiple tasks on different machines. As an example shown in Fig. 2(a), in the traditional scheme, two map tasks individually send  
data of key *K*1 to the reduce task. If we aggregate the data of the same keys before sending them over the top switch, as shown in Fig. 2(b), the network traffic will be reduced. In this paper, we jointly consider data partition and aggregation for a MapReduce job with an objective that is to minimize the total network traffic. In particular, we propose a distributed algorithm for big data applications by decomposing the original large-scale problem into several subproblems that can be solved in parallel.  
Moreover, an online algorithm is designed to deal with the data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost in both offline and online cases.

* 1. **Objective of the Project**

The MapReduce programming models simplifies large-scale data processing on commodity cluster by exploiting parallel map tasks and reduce tasks. Although many efforts have been made to improve the performance of MapReduce jobs, they ignore the network traffic generated in the shuffle phase, which plays a critical role in performance enhancement. Traditionally, a hash function is used to partition intermediate data among reduce tasks, which, however, is not traffic-efficient because network topology and data size associated with each key are not taken into consideration. In this paper, we study to reduce network traffic cost for a MapReduce job by designing a novel intermediate data partition scheme. Furthermore, we jointly consider the aggregator placement problem, where each aggregator can reduce merged traffic from multiple map tasks. A decomposition-based distributed algorithm is proposed to deal with the large-scale optimization problem for big data application and an online algorithm is also designed to adjust data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost under both offline and online cases.

**2. Literature survey:**

**MapReduce: Simplified Data Processing on Large Clusters**

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program’s execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google’s clusters every day.

The MapReduce programming model has been successfully used at Google for many different purposes. We attribute this success to several reasons. First, the model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing. Second, a large variety of problems are easily expressible as MapReduce computations. For example, MapReduce is used for the generation of data for Google’s production web search service, for sorting, for data mining, for machine learning, and many other systems. Third, we have developed an implementation of MapReduce that scales to large clusters of machines comprising thousands of machines. The implementation makes efficient use of these machine resources and therefore is suitable for use on many of the large computational problems encountered at Google.  
We have learned several things from this work. First, restricting the programming model makes it easy to parallelize and distribute computations and to make such computations fault-tolerant. Second, network bandwidth is a scarce resource. A number of optimizations in our system are therefore targeted at reducing the amount of data sent across the network: the locality optimization allows us to read data from local disks, and writing a single copy of the intermediate data to local disk saves network bandwidth. Third, redundant execution can be used to reduce the impact of slow machines, and to handle machine failures and data loss.

**Map Task Scheduling in MapReduce with Data Locality: Throughput and Heavy-Traffic Optimality**

Scheduling map tasks to improve data locality is crucial to the performance of MapReduce. Many works have been devoted to increasing data locality for better efficiency. However, to the best of our knowledge, fundamental limits of MapReduce computing clusters with data locality, including the capacity region and theoretical bounds on the delay performance, have not been studied. In this paper, we address these problems from a stochastic network perspective. Our focus is to strike the right balance between data-locality and load-balancing to simultaneously maximize throughput and minimize delay. We present a new queuing architecture and propose a map task scheduling algorithm constituted by the Join the Shortest Queue policy together with the MaxWeight policy. We identify an outer bound on the capacity region, and then prove that the proposed algorithm stabilizes any arrival rate vector strictly within this outer bound. It shows that the algorithm is throughput optimal and the outer bound coincides with the actual capacity Region. Further, we study the number of backlogged tasks under the proposed algorithm, which is directly related to the delay performance based on Little’s law. We prove that the proposed algorithm is heavy-traffic optimal, i.e., it asymptotically minimizes the number of backlogged tasks as the arrival rate vector approaches the boundary of the capacity region. Therefore, the proposed algorithm is also delay optimal in the heavy-traffic regime.

We considered map scheduling algorithms in MapReduce with data locality. We first presented the capacity region of a MapReduce computing cluster with data locality and then we proved the throughput optimality. Beyond throughput, we showed that the proposed algorithm asymptotically minimizes the number of backlogged tasks as the arrival rate vector approaches the boundary of the capacity region, i.e., it is heavy-traffic optimal.

**Joint Scheduling of MapReduce Jobs with Servers: Performance Bounds and Experiments**

MapReduce has achieved tremendous success for large-scale data processing in data centers. A key feature distinguishing MapReduce from previous parallel models is that it interleaves parallel and sequential computation. Past schemes, and especially their theoretical bounds, on general parallel models are therefore, unlikely to be applied to MapReduce directly. There are many recent studies on MapReduce job and task scheduling. These studies assume that the servers are assigned in advance. in current data centers, multiple MapReduce jobs of different Importance levels run together. In this paper, we investigate a schedule problem for MapReduce taking server assignment in to consideration as well. We formulate a MapReduce server-job organizer problem (MSJO) and show that it is NP-complete. We develop a 3-approximation algorithm and a fast heuristic. We evaluate our algorithms through both simulations and experiments on Amazon EC2 with an implementation in Hadoop. The results confirm the advantage of our algorithms

In this paper, we studied MapReduce job scheduling with consideration of server assignment. We showed that with- out such joint consideration, there can be great performance loss. We formulated a MapReduce server-job organizer problem. This problem is NP-complete and we developed a 3- approximation algorithm MarS. We evaluated our algorithm through extensive simulation. The results show that MarS can outperform state-of-the-art strategies by as much as 40 % in terms of total weighted job completion time. We also implement a prototype of MarS in Hadoop and test it with experiment on Amazon EC2. The experiment results confirm the advantage of our algorithm

**The Hadoop Distributed File System**

The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. We describe the architecture of HDFS and report on experience using HDFS to manage 25 petabytes of enterprise data at Yahoo.

This section presents some of the future work that the Hadoop team at Yahoo is considering; Hadoop being an open source project implies that new features and changes are d

e-ided by the Hadoop development community at large. The Hadoop cluster is effectively unavailable when its Name Node is down. Given that Hadoop is used primarily as a batch system, restarting the Name Node has been a satisfactory recovery means. However, we have taken steps towards auto-mated failover. Currently a Backup Node receives all transactions from the primary Name Node. This will allow a failover to a warm or even a hot BackupNode if we send block reports to both the primary NameNode and BackupNode. A few Hadoop users outside Yahoo! have experimented with manual failover. Our plan is to use Zookeeper, Yahoo’s distributed consensus technology to build an automated failover solution. Scalability of the NameNode has been akey struggle. Because the NameNode keeps all the namespace and block locations in memory, the size of the NameNode heap has limited the number of files and also the number of blocks address-able.

**Map-Reduce Meets Wider Varieties of Applications**

Recent studies and industry practices build data-center-scale computer systems to meet the high storage and processing demands of data-intensive and compute-intensive applications, such as web searches. The Map-Reduce programming model is one of the most popular programming paradigms on these systems. In this paper, we report our experiences and insights gained from implementing three data-intensive and compute-intensive tasks that have different Characteristics from previous studies: a large-scale machine learning computation, a physical simulation task, and a digital media processing task. We identify desirable features and places to improve in the Map-Reduce model. Our goal is to better understand such large-scale computation and data processing in order to design better supports for them.

In this paper, we studied three data-intensive and compute-intensive applications that have very different characteristics from previous reported Map-Reduce applications. We find that although we can easily implement a semantically correct Map-Reduce program, achieving good performance is tricky. For example, a computation that looks similar to word counting at the first sight may turn out to have very different characteristics, such as the number and variance of intermediate results, thus resulting in unexpected performance. Learning from the application studies, we explore the design space for supporting data-intensive and compute-intensive applications on data-center-scale computer systems. We find two directions are promising: (i) enhancing a job control system with a set of desirable features; (ii) supporting flexible compos able components and including more optimization supports in Map-Reduce

System. We plan to investigate these directions in future work.

**Declarative Systems for Large-Scale Machine Learning**

In this article, we make the case for a declarative foundation for data-intensive machine learning systems. Instead of creating a new system for each specific flavor of machine learning task, or hard-coding new optimizations, we argue for the use of recursive queries to program a variety of machine learning algorithms. By taking this approach, database query optimization techniques can be utilized to identify effective execution plans, and the resulting runtime plans can be executed on a single unified data-parallel query processing engine.

The growing demand for machine learning is pushing both industry and academia to design new types of highly scalable iterative computing systems. Examples include Mahout, Pregel, Spark, Twister, Hadoop, and PrItr. However, today’s specialized machine learning platforms all tend to mix logical representations and physical implementations. As a result, today’s platforms 1) require their developers to rebuild critical components and to hardcode optimization strategies and 2) limit themselves to specific runtime implementations that usually only (naturally) fit a limited subset of the potential machine learning workloads. This leads to the current state of

Practice: implementing new scalable machine learning algorithms is very labor-intensive and the overall data processing pipeline involves multiple disparate tools hooked together with file- and workflow-based glue. In contrast, we have advocated a declarative foundation on which specialized machine learning workflows can be easily constructed and readily tuned. We have verified our approach with Datalog implementations of two popular programming models from the machine learning domain: Iterative Map-Reduce-Update, for deriving linear models, and Pregel, for graphical algorithms (see [5]). The resulting Datalog programs are compact, tunable to a specific task (e.g., Batch Gradient Descent and PageRank), and translated to optimized physical plans. Our experimental results show that on a large real-world dataset and machine cluster, our optimized plans are very competitive with other systems that target the given class of ML tasks (see [5]). Furthermore, we demonstrated that our approach can offer a plan tailored to a given target task and data for a given machine resource allocation. In contrast, in our large experiments, Spark failed due to main-memory limitations and Hadoop succeeded but ran an order-of-magnitude less efficiently. The work reported here is just a first step. We are currently developing the ScalOps query processing components required to automate the remaining translation steps; these include the Planner/Optimizer as well as a more general algebraic foundation based on extending the Algebricks query algebra and rewrite rule framework of ASTERIX [4]. We also plan to investigate support for a wider range of machine learning tasks and for a more asynchronous, GraphLab-inspired programming model for encoding graphical algorithms.

**Distributed Machine Learning and Graph Processing with Sparse Matrices**

It is cumbersome to write machine learning and graph algorithms in data-parallel models such as MapReduce and Dryad. We observe that these algorithms are based on matrix computations and, hence, are inefficient to implement with the restrictive programming and communication interface of such frameworks. In this paper we show that array-based languages such as R [2] are suitable for implementing complex algorithms and can outperform current data parallel solutions. Since R is single threaded and does not scale to large datasets, we have built Pronto, a distributed system that extends R and addresses many of its limitations. Pronto efficiently shares sparse structured data can leverage multi-cores, and dynamically partitions data to mitigate load imbalance. Our results show the promise of this approach: many important machine learning and graph algorithms can be expressed in a single framework and are substantially faster than those in Hadoop and Spark. Pronto advocates the use of sparse matrix operations to simplify the implementation of machine learning and graph algorithms in a cluster. Pronto uses distributed arrays for structured processing, efficiently uses multi-cores, and dynamically partitions data to reduce load imbalance. Our experience shows that Pronto is a flexible computation model that can be used to implement a variety of complex algorithms

**Using Bioinformatics Applications on the Cloud**

Dealing with large genomic data on a limited computing resource has been an inevitable challenge in life science. Bioinformatics applications have required high performance computation capabilities for next-generation sequencing (NGS) data and the human genome sequencing data with single nucleotide polymorphisms (SNPs). From 2008, Cloud computing platforms have been widely adopted to deal with the large data sets with parallel processing tools. MapReduce parallel programming framework is dominantly used due to its fast and ancient performance for data processing on cloud clusters. This study introduces various research projects regarding to reducing a data analysis time and improving usability with their approaches. Hadoop implementations and work ow toolkits are focused on address parallel

data processing tools and easy-to-use environments

These days, individual research laboratory is able to generate terabytes of data (or even larger), which is no suprises to new sequencing technologies in genomic research. High performance computation environments keep improving on processing large-scale data at low cost. The combination of MapReduce and cloud computing facilitates fast and efficient parallel processing on the virtual environment for terabyte-scale data analysis in bioinformatics, if the analysis consists of embarrassingly parallel problems. MapReduce framework is suitable for the simple and dividable tasks such as read alignment, sequence search and image recognition. Easy-to-use methods and user-friendly cloud platforms have been provided to researchers so that they can easily have ac- cess to the cloud with their large data sets uploaded on the cloud in a secure manner. Scientic work ow may focus on improving data transfer and handling tasks regarding these usability problems. More challenges are expected to deal with data storage and analysis since it grows at unprecen-dented scales.

**Comparison of Distributed Data-Parallelization Patterns for Big Data Analysis: A Bioinformatics Case Study**

As a distributed data-parallelization (DDP) pattern, MapReduce has been adopted by many new big data analysis tools to achieve good scalability and performance in Cluster or Cloud environments. This paper explores how two binary DDP patterns, i.e., Co Group and Match, could also be used in these tools. We re-implemented an existing bioinformatics tool, called Cloudburst, with three different DDP pattern combinations. We identify two factors, namely, input data balancing and value sparseness, which could greatly affect the performances using different DDP patterns. Our experiments show: (i) a simple DDP pattern switch could speed up performance by almost two times; (ii) the identified factors can explain the differences wen the “big data” era, it is very popular and effective to use DDP patterns in order to achieve scalability and parallelization. These DDP patterns also bring challenges on which pattern or pattern combination is the best for a certain tool. This paper demonstrates different DDP patterns coul

d have a great impact on the performances of the same tool. We find that although MapReduce

can be used for wider range of applications with either one or two input datasets, it is not always the best choice in terms of application complexity and performance. To understand the differences, we identified two affecting factors, namely input data balancing and value sparseness, on their performance differences. The feasibility of these two factors is verified through experiments. We believe many tools in bioinformatics and other domains have a similar logic with CloudBurst as they need to match two input datasets, and therefore could also benefit from our findings. For future work, we plan to investigate more tools that are suitable for multiple DDP patterns and their performances on other DDP engines like Hadoop, which will generalize our findings. We will also study how to utilize the identified factors to automatically Select the best DDP pattern combination from multiple available ones.

**Compressed Nonnegative Matrix Factorization Is Fast and Accurate**

Nonnegative matrix factorization (NMF) has an established reputation as a useful data analysis technique in numerous applications. However, its usage in practical situations is undergoing challenges in recent years. The fundamental factor to this is the increasingly growing size of the datasets available and needed in the information sciences. To address this, in this work we propose to use structured random compression, that is, random projections that exploit the data structure, for two NMF variants: classical and separable. In separable NMF (SNMF) the Left factors are a subset of the columns of the input matrix. We present suitable formulations for each problem, dealing with different representative algorithms within each one. We show that the resulting compressed techniques are faster than their uncompressed variants, vastly reduce memory demands, and do not encompass any significant deterioration in performance. The proposed structured random projections for SNMF allow dealing with arbitrarily shaped large matrices, beyond the standard limit of tall-and-skinny matrices, granting access to very efficient computations in this general setting. We accompany the algorithmic presentation with theoretical foundations and numerous and diverse examples, showing the suitability of the proposed approaches.

In this work we proposed to use structured random projections for NMF and SNMF. For NMF, we presented formulations for three popular techniques, namely, multiplicative updates, active set method for nonnegative least squares and ADMM. For SNMF, we presented a general technique that can be used with any algorithm. In all cases, we showed that the resulting compressed techniques are faster than their uncompressed variants and, at the same time; do not introduce significant errors in the final result. There are in the literature very efficient SNMF algorithms for tall-and-skinny matrices. Interestingly, the use of structured random projections allows computing SNMF for arbitrarily large matrices, granting access to very efficient computations in the general setting. As a byproduct, we also propose an algorithmic solution for Computing structured random projections of extremely large matrices (i.e., matrices so large that even after compression they do not fit in main memory). This is useful as a general tool for computing many different matrix decompositions, such as the singular value decomposition, for example. We are currently investigating the problem of replacing the Frobenius norm with and Norm in our compressed variants of NMF and SNMF. In this setting, the fast Cauchy transform is a suitable alternative to structured random projections. Compression consists of sampling and rescaling rows of A, thus identifying the so-called corset of the problem. This formulation is of particular interest for network analysis, where we need to deal with sparse structures.

**3. Analysis**

**3.1. Introduction**

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process](http://en.wikipedia.org/wiki/Software_development_process).

**3.2. Existing System**

Intermediate data are shuffled according to a hash function in Hadoop, which would lead to large network traffic because it ignores network topology and data size associated with each key. To tackle this problem incurred by the traffic-oblivious partition scheme, we take into account of both task locations and data size associated with each key in this paper. By assigning keys with larger data size to reduce tasks closer to map tasks, network traffic can be significantly reduced.

To further reduce network traffic within a MapReduce job, we consider to aggregate data with the same keys before sending them to remote reduce tasks. Although a similar function, called combiner, has been already adopted by Hadoop, it operates immediately after a map task solely for its generated data, failing to exploit the data aggregation opportunities among multiple tasks on different machines.

**Disadvantages:**

* Traditionally, A hash function is used to partition intermediate data among reduce tasks, which, however, is not traffic-efficient because network topology and data size associated with each key are not taken into consideration.
* It leads to large network traffic because it ignores network topology and data size associated with each key.
* Network traffic can be significantly reduced.

**3.3. Proposed System**

In this paper, we jointly consider data partition and aggregation for a Map Reduce job with an objective that is to minimize the total network traffic. In particular, we propose a distributed algorithm for big data applications by decomposing the original large-scale problem into several sub problems that can be solved in parallel. Moreover, an online algorithm is designed to deal with the data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost in both offline and online cases.

**Advantages:**

* Each aggregator can reduce merged traffic from multiple map tasks. It is designed to adjust data partition and aggregation in a dynamic manner.
* It can significantly reduce network traffic cost in both offline and online cases.

**3.4. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and

textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term *requirements traceability*.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category.

For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.5. Software Requirement Specification**

**3.5.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.5.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly Java Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through Java capabilities.

**Software Interfaces**

The required software is JAVA1.6.

**Operating Environment**

Windows XP, Linux.

**HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Floppy Drive : 1.44 Mb.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* Ram : 512 Mb.

**SOFTWARE REQUIREMENTS:**

* Operating system : Windows XP/7.
* Coding Language : JAVA
* Frontend : AWT, Swings
* Backend : MySQL

**4. Design**

**UML diagrams**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + **User Model View**
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + **Structural Model view**
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* **Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation Model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

**4.1 Class diagram:-**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

A class with three sections.

In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake.

**Class diagram:**



**4.2 Use case diagram:-**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

**4.2.1 Use case diagram:**



**4.3. Sequence Diagram:**

A **sequence diagram** 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. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.

**4.3.1 Sequence diagram:**



**4.4 Collaborative diagram**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

**4.5.1 Collaborative diagram:**



**4.6 Component Diagram**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

**4.6.1 Component diagram:**

****

**4.7 Deployment Diagram**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

**4.7.1 Deployment diagram:**

****

**4.8 Activity diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

**4.8.1 Activity diagram:**

**Activity diagram:**

Produce time chart graph

Start the aggregation

Loading the input data

Run the reducer application

Define Reducer location details

Yes No

**4.9 Data Flow Diagram:**

[Data flow diagrams](http://www.edrawsoft.com/Data-Flow-Diagrams.php) illustrate how data is processed by a system in terms of inputs and outputs.

Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

**Dataflow diagram:**

To add reducer define reducer name Load the data

Upload

Latitude, longitude values data loaded successfully

Network traffic cost graph

Start the Map Reduce aggregation

Define reducer

Reducer details added after processing the aggregation, it displays the count result

Displays the graph between processing Database

Time and techniques

**TEST CASES:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case 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 location by providing latitude & longitude values | If we doesn’t provide latitude, longitude values | Location 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 we not run the application | 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 we can’t upload the data | We can’t reduce the network traffic | Input data loaded successfully | High | High |
| Start Mapreduce aggregation  04 | Aggregation using Mapreduce | It aggregates all the partitioned data | If we not start the aggregation | We can’t reduce the network traffic | After processing the aggregate data, it displays the count result. | High | High |
| Graph  05 | Network traffic cost graph | Displays the graph between processing time & Technique | If we can’t do any aggragation | Nothing will be displayed | Graph will be displayed using aggregated/no aggregated data | High | High |

**5. Implementation**

**5.1. Introduction of technologies used**

**About Java**:

Initially the language was called as “oak” but it was renamed as “java” in 1995.The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral)language that could be used to create software to be embedded in various consumer electronic devices.

* Java is a programmer’s language
* Java is cohesive and consistent
* Except for those constraint imposed by the Internet environment. Java gives the programmer, full control

Finally Java is to Internet Programming where c was to System Programming.

**Importance of Java to the Internet**

Java has had a profound effect on the Internet. This is because; java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs. in the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

**Applications and applets**

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++ .Java’s ability to create Applets makes it important. An Applet I san application, designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet I actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can be react to the user input and dynamically change.

**Java Architecture**

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

# Compilation of code

# When you compile the code, the Java compiler creates machine code (called byte code)for a hypothetical machine called Java Virtual Machine(JVM). The JVM is supposed t executed the byte code. The JVM is created for the overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines .This machine is called Java Virtual Machine.

**Compiling and interpreting java source code.**

**Source code**

**Pc compiler**

**Macintosh compiler**

**SPARC Compiler**

**Java Byte code**

**Platform independent**

**Java interpreter**

**Java interpretermacintosh**

**)))**

**Java interpreter(SPARC)**

During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be an Intel Pentium windows 95 or sun SPARCstation running Solaris or Apple Macintosh running system and all could receive code from any computer through internet and run the Applets.

**Simple**:

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ Programmer. Learning Java will oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

### Object oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank state. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

### Robust

The multi-platform environment of the web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs. Was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and runtime.

Java virtually eliminates the problems of memory management and deal location, which is completely automatic. In a well-written Java program, all run-time errors can and should be managed by your program.

**AWT and Swings:**

**AWT:**

**Graphical User Interface:**

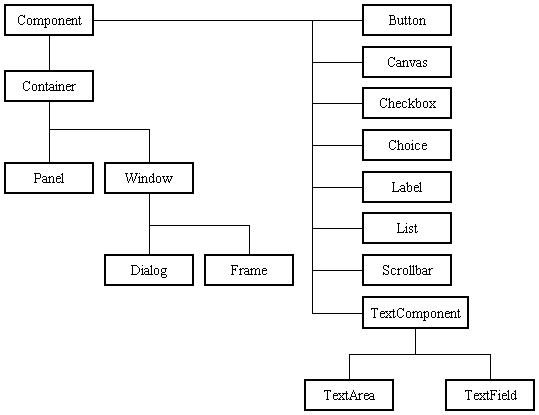
The user interface is that part of a program that interacts with the user of the program. GUI is a type of [user interface](http://en.wikipedia.org/wiki/User_interface) that allows [users](http://en.wikipedia.org/wiki/User_(computing)) to [interact](http://en.wikipedia.org/wiki/Human-computer_interaction) with electronic devices with images rather than text commands. A class library is provided by the Java programming language which is known as Abstract Window Toolkit (AWT) for writing graphical programs. The Abstract Window Toolkit (AWT) contains several graphical widgets which can be added and positioned to the display area with a layout manager.

As the Java programming language, the AWT is not platform-independent. AWT uses system peers object for constructing graphical widgets. A common set of tools is provided by the AWT for graphical user interface design. The implementation of the user interface elements provided by the AWT is done using every platform's native GUI toolkit. One of the AWT's significance is that the look and feel of each platform can be preserved.

**Components:**

A graphical user interface is built of graphical elements called components. A *component* is an object having a graphical representation that can be displayed on the screen and that can interact with the user. Components allow the user to interact with the program and provide the input to the program. In the AWT, all user interface components are instances of class Component or one of its subtypes. Typical components include such items as buttons, scrollbars, and text fields.

**Types of Components:**

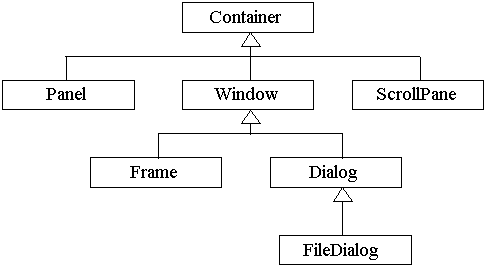
****

Before proceeding ahead, first we need to know what containers are. After learning containers we learn all components in detail.

**Containers:**

Components do not stand alone, but rather are found within containers. In order to make components visible, we need to add all components to the container. Containers contain and control the layout of components. In the AWT, all containers are instances of class Container or one of its subtypes. Components must fit completely within the container that contains them. For adding components to the container we will use add() method.

**Types of containers:**

****

**Basic GUI Logic:**

The GUI application or applet is created in three steps. These are:

* Add components to Container objects to make your GUI.
* Then you need to setup event handlers for the user interaction with GUI.
* Explicitly display the GUI for application.

A new thread is started by the interpreter for user interaction when an AWT GUI is displayed. When any event is received by this new thread such as click of a mouse, pressing of key etc then one of the event handlers is called by the new thread set up for GUI. One important point to note here is that the event handler code is executed within the thread.

**Creating a Frame:**

**Method1:**

In the first method we will be creating frame by extending Frame class which is defined in java.awt package. Following program demonstrate the creation of a frame.

import java.awt.\*;

public class FrameDemo1 extends Frame

{

FrameDemo1()

{

setTitle("Label Frame");

setVisible(true);

setSize(500,500);

}

public static void main(String[] args)

{

new FrameDemo1 ();

}

}

In the above program we are using three methods:

setTitle: For setting the title of the frame we will use this method. It takes String as an argument which will be the title name.

SetVisible: For making our frame visible we will use this method. This method takes Boolean value as an argument. If we are passing true then window will be visible otherwise window will not be visible.

SetSize: For setting the size of the window we will use this method. The first argument is width of the frame and second argument is height of the frame.

**Method 2:**

In this method we will be creating the Frame class instance for creating frame window. Following program demonstrate Method2.

import java.awt.\*;

public class FrameDemo2

{

public static void main(String[] args)

{

Frame f = new Frame();

f.setTitle("My first frame");

f.setVisible(true);

f.setSize(500,500);

}

}

**Types of Components:**

1. **Labels :**

This is the simplest component of Java Abstract Window Toolkit. This component is generally used to show the text or string in your application and label never perform any type of action.

Label l1 = new Label("One");

Label l2 = new Label("Two");

Label l3 = new Label("Three",Label.CENTER);

In the above three lines we have created three labels with the name “one, two, three”. In the third label we are passing two arguments. Second argument is the justification of the label. Now after creating components we will be adding it to the container.

add(l1);

add(l2);

add(l3);

We can set or change the text in a label by using the **setText( )** method. You can obtain the current label by calling **getText( )**. These methods are shown here:

void setText(String *str*)

String getText( )

1. **Buttons :**

This is the component of Java Abstract Window Toolkit and is used to trigger actions and other events required for your application. The syntax of defining the button is as follows :

Button l1 = new Button("One");

Button l2 = new Button("Two");

Button l3 = new Button("Three");

We can change the Button's label or get the label's text by using the Button.setLabel(String) and Button.getLabel() method.

1. **CheckBox:**

A *check box is a control that is used to turn an option on or off. It consists of a small box* that can either contain a check mark or not. There is a label associated with each check box that describes what option the box represents. You change the state of a check box by clicking on it. The syntax of the definition of Checkbox is as follows :

Checkbox Win98 = new Checkbox("Windows 98/XP", null, true);

Checkbox winNT = new Checkbox("Windows NT/2000");

Checkbox solaris = new Checkbox("Solaris");

Checkbox mac = new Checkbox("MacOS");

The first form creates a check box whose label is specified in first argument and whose group is specified in second argument*.* If this check box is not part of a group, then *cbGroup* must be **null**. (Check box groups are described in the next section.) The value truedetermines the initial state of the check box is checked. The second form creates a check box with only one parameter.

To retrieve the current state of a check box, call **getState( )**. To set its state, call **setState( )**. You can obtain the current label associated with a check box by calling **getLabel( )**. To set the label, call **setLabel( )**. These methods are as follows:

boolean getState( )

void setState(boolean *on*)

String getLabel( )

void setLabel(String *str*)

Here, if *on* is **true**, the box is checked. If it is **false**, the box is cleared. The string passed in *str* becomes the new label associated with the invoking check box.

1. **Radio Button:**

This is the special case of the Checkbox component of Java AWT package. This is used as a group of checkboxes which group name is same. Only one Checkbox from a Checkbox Group can be selected at a time. Syntax for creating radio buttons is as follows:

CheckboxGroup cbg = new CheckboxGroup();

Checkbox Win98 = new Checkbox("Windows 98/XP", cbg , true);

Checkbox winNT = new Checkbox("Windows NT/2000",cbg, false);

Checkbox solaris = new Checkbox("Solaris",cbg, false);

Checkbox mac = new Checkbox("MacOS",cbg, false);

For Radio Button we will be using CheckBox class. The only difference in Checkboxes and radio button is in Check boxes we will specify null for checkboxgroup but whereas in radio button we will be specifiying the checkboxgroup object in the second parameter.

1. **Choice:**

The Choice class is used to create a pop-up list of items from which the user may choose. Thus, a Choice control is a form of menu. Syntax for creating choice is as follows:

Choice os = new Choice();

/\* adding items to choice \*/

os.add("Windows 98/XP");

os.add("Windows NT/2000");

os.add("Solaris");

os.add("MacOS");

We will be creating choice with the help of Choice class. Pop up list will be creating with the creation of object, but it will not have any items. For adding items we will be using add() method defined in Choice class.

To determine which item is currently selected, you may call either **getSelectedItem( )** or **getSelectedIndex( )**. These methods are shown here:

String getSelectedItem( )

int getSelectedIndex( )

The **getSelectedItem( )** method returns a string containing the name of the item. **getSelectedIndex( )** returns the index of the item. The first item is at index 0. By default, the first item added to the list is selected.

1. **List:**

List class is also same as choice but the only difference in list and choice is, in choice user can select only one item whereas in List user can select more than one item. Syntax for creating list is as follows:

List os = new List(4, true);

First argument in the List constructor specifies the number of items allowed in the list. Second argument specifies whether multiple selections are allowed or not.

/\* Adding items to the list \*/

os.add("Windows 98/XP");

os.add("Windows NT/2000");

os.add("Solaris");

os.add("MacOS");

In list we can retrieve the items which are selected by the users. In multiple selection user will be selecting multiple values for retrieving all the values we have a method called getSelectedValues() whose return type is string array. For retrieving single value again we can use the method defined in Choice i.e. getSelectedItem().

1. **TextField:**

Text fields allow the user to enter strings and to edit the text using the arrow keys, cut and paste keys. TextField is a subclass of TextComponent. Syntax for creating list is as follows:

TextField tf1 = new TextField(25);

TextField tf2 = new TextField();

In the first text field we are specifying the size of the text field and the second text field is created with the default value. **TextField** (and its superclass **TextComponent**) provides several methods that allow you to utilize a text field. To obtain the string currently contained in the text field, call **getText( )**. To set the text, call **setText( )**. These methods are as follows:

String getText( )

void setText(String *str*)

We can control whether the contents of a text field may be modified by the user by calling **setEditable( )**. You can determine editability by calling **isEditable( )**. These methods are shown here:

boolean isEditable( )

void setEditable(boolean *canEdit*)

**isEditable( )** returns **true** if the text may be changed and **false** if not. In **setEditable( )**, if *canEdit* is **true**, the text may be changed. If it is **false**, the text cannot be altered.

There may be times when we will want the user to enter text that is not displayed, such as a password. We can disable the echoing of the characters as they are typed by calling **setEchoChar( )**.

1. **TextArea:**

TextArea is a multiple line editor. Syntax for creating list is as follows:

TextArea area = new TextArea(20,30);

Above code will create one text area with 20 rows and 30 columns. **TextArea** is a subclass of **TextComponent**. Therefore, it supports the **getText( )**, **setText( )**, **getSelectedText( )**, **select( )**, **isEditable( )**, and **setEditable( )** methods described in the preceding section.

**TextArea** adds the following methods:

void append(String *str*)

void insert(String *str*, int *index*)

void replaceRange(String *str*, int *startIndex*, int *endIndex*)

The **append( )** method appends the string specified by *str* to the end of the current text. **insert( )** inserts the string passed in *str* at the specified index. To replace text, call **replaceRange( )**. It replaces the characters from *startIndex* to *endIndex*–1, with the replacement text passed in *str.*

**Layout Managers:**

A layout manager automatically arranges controls within a window by using some type of algorithm. Each **Container** object has a layout manager associated with it. A layout manager is an instance of any class that implements the **LayoutManager** interface. The layout manager is set by the **setLayout( )** method. If no call to **setLayout( )** is made, then the default layout manager is used. Whenever a container is resized (or sized for the first time), the layout manager is used to position each of the components within it. The **setLayout( )** method has the following general form:

void setLayout(LayoutManager *layoutObj*)

Here, *layoutObj* is a reference to the desired layout manager. If you wish to disable the layout manager and position components manually, pass **null** for *layoutObj.* If we do this, you will need to determine the shape and position of each component manually, using the setBounds( ) method defined by Component.

Void setBounds(int x , int y , int width, int length)

In which first two arguments are the x and y axis. Third argument is width and fourth argument is height of the component.

Java has several predefined **LayoutManager** classes, several of which are described next. You can use the layout manager that best fits your application.

**FlowLayout:**

**FlowLayout** is the default layout manager. This is the layout manager that the preceding examples have used. **FlowLayout** implements a simple layout style, which is similar to how words flow in a text editor. Components are laid out from the upper-left corner, left to right and top to bottom. When no more components fit on a line, the next one appears on the next line. A small space is left between each component, above and below, as well as left and right. Here are the constructors for **FlowLayout**:

FlowLayout( )

FlowLayout(int *how*)

FlowLayout(int *how*, int *horz*, int *vert*)

The first form creates the default layout, which centers components and leaves five pixels of space between each component. The second form lets you specify how each line is aligned. Valid values for *how* are as follows:

FlowLayout.LEFT

FlowLayout.CENTER

FlowLayout.RIGHT

These values specify left, center, and right alignment, respectively. The third form allows you to specify the horizontal and vertical space left between components in *horz* and *vert,* respectively.

**BorderLayout:**

THE JAVA LIBRARYThe **BorderLayout** class implements a common layout style for top-level windows. It has four narrow, fixed-width components at the edges and one large area in the center. The four sides are referred to as north, south, east, and west. The middle area is called the center. Here are the constructors defined by **BorderLayout**:

BorderLayout( )

BorderLayout(int *horz*, int *vert*)

The first form creates a default border layout. The second allows you to specify the horizontal and vertical space left between components in *horz* and *vert,* respectively. **BorderLayout** defines the following constants that specify the regions:

BorderLayout.CENTER BorderLayout.SOUTH

BorderLayout.EAST BorderLayout.WEST

BorderLayout.NORTH

When adding components, you will use these constants with the following form of **add( )**, which is defined by **Container**:

void add(Component *compObj,* Object *region*);

Here, *compObj* is the component to be added, and *region* specifies where the component will be added.

**GridLayout:**

**GridLayout** lays out components in a two-dimensional grid. When you instantiate a **GridLayout**, you define the number of rows and columns. The constructors supported by **GridLayout** are shown here:

GridLayout( )

GridLayout(int *numRows*, int *numColumns* )

GridLayout(int *numRows*, int *numColumns*, int *horz*, int *vert*)

The first form creates a single-column grid layout. The second form creates a grid layout with the specified number of rows and columns. The third form allows you to specify the horizontal and vertical space left between components in *horz* and *vert*, respectively. Either *numRows* or *numColumns* can be zero. Specifying *numRows* as zero allows for unlimited-length columns. Specifying *numColumns* as zero allows for unlimited-length rows.

**Swings:**

**About Swings:**

Swing is important to develop Java programs with a graphical user interface (GUI). There are many components which are used for the building of GUI in Swing. The Swing Toolkit consists of many components for the building of GUI. These components are also helpful in providing interactivity to Java applications. Following are components which are included in Swing toolkit:

* list controls
* buttons
* labels
* tree controls
* table controls

All AWT flexible components can be handled by the Java Swing. Swing toolkit contains far more components than the simple component toolkit. It is unique to any other toolkit in the way that it supports integrated internationalization, a highly customizable text package, rich undo support etc. Not only this you can also create your own look and feel using Swing other than the ones that are supported by it. The customized look and feel can be created using Synth which is specially designed. Not to forget that Swing also contains the basic user interface such as customizable painting, event handling, drag and drop etc.

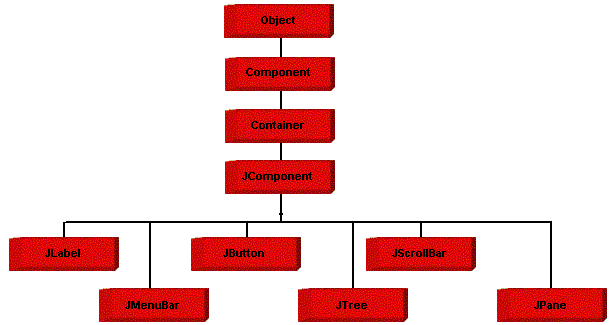
The Java Foundation Classes (JFC) which supports many more features important to a GUI program comprises of Swing as well. The features which are supported by Java Foundation Classes (JFC) are the ability to create a program that can work in different languages, the ability to add rich graphics functionality etc.

There are several components contained in Swing toolkit such as check boxes, buttons, tables, text etc. Some very simple components also provide sophisticated functionality. For instance, text fields provide formatted text input or password field behavior. Furthermore, the file browsers and dialogs can be used according to one's need and can even be customized.

**Difference between Swings and AWT:**

|  |  |
| --- | --- |
| **Swings** | **AWT** |
| Swings are the light weight components. | AWTs are the heavy weight components. |
| Swings are developed by using pure java language. | AWTs are developed by using C and C++. |
| We can have different look and feel in swings. | This feature is not available in awt. |
| Swing has many advanced features like JTabel, JTabbedPane and JTree | This are not available in awt. |

**Java Swing Class Hierarchy:**



**Swing Components:**

All the components which are supported in AWT same components are also supported in Swings with a slight change in their class name.

|  |  |
| --- | --- |
| **AWT Components** | **Swing Components** |
| Label | JLabel |
| TextField | JTextField |
| TextArea | JTextArea |
| Choice | JComboBox |
| Checkbox | JCheckBox |
| List | JList |
| Button | JButton |
| - | JRadioButton |
| - | JPasswordField |
| - | JTable |
| - | JTree |
| - | JTabbedPane |
| MenuBar | JMenuBar |
| Menu | JMenu |
| MenuItem | JMenuItem |
| - | JFileChooser |
| - | JOptionPane |

We will discuss only those components which are not discussed in AWT chapter.

**JTabbedPane class:**

 The JTabbedPane container allows many panels to occupy the same area of the interface, and the user may select which to show by clicking on a tab.

**Constructor**

JTabbedPane tp = new JTabbedPane();

## Adding tabs to the JTabbedPane

Add tabs to a tabbed pane by calling addTab and passing it a String title and an instance of a class which should be called when we pressed a tab. That class should be a subclass of JPanel.

addTab(“String”,instance);

**Example program:**

import javax.swing.\*;

import java.awt.\*;

public class TabbedPaneDemo extends JFrame

{

TabbedPaneDemo()

{

setLayout(new FlowLayout(FlowLayout.LEFT));

setTitle("Tabbed Demo");

setVisible(true);

setSize(500,500);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JTabbedPane pane = new JTabbedPane();

pane.addTab("Countries",new Count());

pane.addTab("Cities",new Cit());

add(pane);

}

public static void main(String a[])

{

new TabbedPaneDemo();

}

}

class Count extends JPanel

{

Count()

{

JButton b1 = new JButton("India");

JButton b2 = new JButton("SriLanka");

JButton b3 = new JButton("Australia");

add(b1);

add(b2);

add(b3);

}

}

class Cit extends JPanel

{

Cit()

{

JCheckBox cb1 = new JCheckBox("Hyderabad");

JCheckBox cb2 = new JCheckBox("Banglore");

JCheckBox cb3 = new JCheckBox("Pune");

add(cb1);

add(cb2);

add(cb3);

}

}

**JMenuBar, JMenu, JMenuItem**

A top-level window can have a menu bar associated with it. A menu bar displays a list of top-level menu choices. Each choice is associated with a drop-down menu. This concept is implemented in Java by the following classes: JMenuBar, JMenu, and JMenuItem. In general, a menu bar contains one or more JMenu objects. Each JMenu object contains a list of JMenuItem objects. Each JMenuItem object represents something that can be selected by the user. To create a menu bar, first create an instance of JMenuBar. This class only defines the default constructor. Next, create instances of JMenu that will define the selections displayed on the bar. Following are the constructors for Menu:

JMenu( )

JMenu(String *optionName*)

Here, *optionName* specifies the name of the menu selection. The first form creates an empty menu. Individual menu items are of type MenuItem. It defines these constructors:

JMenuItem( )

JMenuItem(String *itemName*)

Here, *itemName* is the name shown in the menu.

**5.2. Sample code**

**Main.java**

package mapreduce;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JButton;

import javax.swing.JPanel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.UIManager;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.Color;

import java.awt.Font;

import javax.swing.JOptionPane;

import javax.swing.JFileChooser;

import java.io.File;

import java.io.BufferedReader;

import java.io.FileReader;

import java.util.ArrayList;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.sql.Statement;

import java.sql.ResultSet;

import org.jfree.ui.RefineryUtilities;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

public class Main extends JFrame

{

GradientPanel p1;

JPanel p2;

JLabel title;

JButton b1,b2,b3,b4,b5;

Font f1;

JFileChooser chooser;

File file;

static StringBuilder sb = new StringBuilder();

ArrayList<Location> location = new ArrayList<Location>();

public void remove(){

File remove = new File("output");

File list[] = remove.listFiles();

if(list != null){

for(int i=0;i<list.length;i++){

if(list[i] != null)

System.out.println(list[i].delete()+" delete =======");

}

}

if(remove != null)

remove.delete();

}

public Main(){

super("Map Reduce System");

p1 = new GradientPanel(600,200);

p1.setLayout(null);

f1 = new Font("Courier New",Font.BOLD,14);

p2 = new JPanel();

p2.setBackground(new Color(204, 110, 155));

title = new JLabel("<HTML><BODY><CENTER>ON TRAFFIC-AWARE PARTITION AND AGGREGATION IN MAPREDUCE<BR/>FOR BIG DATA APPLICATIONS</CENTER></BODY></HTML>");

title.setForeground(Color.white);

title.setFont(new Font("Times New ROMAN",Font.PLAIN,17));

p2.add(title);

chooser = new JFileChooser(new File("."));

chooser.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

JPanel pan3 = new JPanel();

b1 = new JButton("Define Reducer Location");

b1.setFont(f1);

b1.setBounds(220,50,250,30);

p1.add(b1);

b1.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent ae){

DefineReducer dr = new DefineReducer();

dr.setVisible(true);

dr.setSize(600,360);

dr.setLocationRelativeTo(null);

dr.setResizable(false);

}

});

b2 = new JButton("Upload Documents");

b2.setFont(f1);

b2.setBounds(220,100,250,30);

p1.add(b2);

b2.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent ae){

int option = chooser.showOpenDialog(Main.this);

if(option == chooser.APPROVE\_OPTION){

sb.delete(0,sb.length());

file = chooser.getSelectedFile();

JOptionPane.showMessageDialog(Main.this,"Input documents loaded");

}

}

});

b3 = new JButton("Start MapReduce Aggregation");

b3.setFont(f1);

b3.setBounds(220,150,250,30);

p1.add(b3);

b3.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent ae){

try{

remove();

long start = System.currentTimeMillis();

String a[] = {"test"};

WordFrequency.setInputPath(file.getPath());

WordFrequency.start(a);

Location loc = location.get(0);

Socket soc = null;

if(loc.getReducer().equals("Reducer1"))

soc = new Socket("localhost",2222);

if(loc.getReducer().equals("Reducer2"))

soc = new Socket("localhost",3333);

ObjectOutputStream out = new ObjectOutputStream(soc.getOutputStream());

Object req[] = {"input",sb.toString()};

out.writeObject(req);

out.flush();

ObjectInputStream in = new ObjectInputStream(soc.getInputStream());

Object res[] = (Object[])in.readObject();

String msg = (String)res[0];

long end = System.currentTimeMillis();

ViewResult vr = new ViewResult();

vr.setVisible(true);

vr.setSize(600,400);

if(msg.equals("output")){

String output = (String)res[1];

String arr[] = output.split("\n");

for(int i=0;i<arr.length;i++){

Object ar[] = arr[i].split("\t");

vr.dtm.addRow(ar);

}

}

FileWriter fw = new FileWriter("D:/agg.txt");

fw.write(""+(end-start));

fw.close();

JOptionPane.showMessageDialog(Main.this,"Processing Time "+(end-start));

}catch(Exception e){

e.printStackTrace();

}

}

});

b4 = new JButton("Network Traffic Cost Graph");

b4.setFont(f1);

b4.setBounds(220,200,250,30);

p1.add(b4);

b4.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent ae){

try{

BufferedReader br = new BufferedReader(new FileReader("D:/no\_agg.txt"));

int noagg = Integer.parseInt(br.readLine().trim());

br.close();

br = new BufferedReader(new FileReader("D:/agg.txt"));

int agg = Integer.parseInt(br.readLine().trim());

br.close();

Chart chart1 = new Chart("Processing Time Chart",noagg,agg);

chart1.pack();

RefineryUtilities.centerFrameOnScreen(chart1);

chart1.setVisible(true);

}catch(Exception e){

e.printStackTrace();

}

}

});

b5 = new JButton("Exit");

b5.setFont(f1);

b5.setBounds(220,250,250,30);

p1.add(b5);

b5.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent ae){

System.exit(0);

}

});

getContentPane().add(p1,BorderLayout.CENTER);

getContentPane().add(p2,BorderLayout.NORTH);

}

public static void main(String a[])throws Exception{

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

Main main = new Main();

main.setVisible(true);

main.setExtendedState(JFrame.MAXIMIZED\_BOTH);

main.readReducerLoc();

}

public double distance(double lat1, double lon1, double lat2, double lon2, char unit){

double theta = lon1 - lon2;

double dist = Math.sin(deg2rad(lat1)) \* Math.sin(deg2rad(lat2)) + Math.cos(deg2rad(lat1)) \* Math.cos(deg2rad(lat2)) \* Math.cos(deg2rad(theta));

dist = Math.acos(dist);

dist = rad2deg(dist);

dist = dist \* 60 \* 1.1515;

if (unit == 'K') {

dist = dist \* 1.609344;

} else if (unit == 'N') {

dist = dist \* 0.8684;

}

return (dist);

}

public double deg2rad(double deg) {

return (deg \* Math.PI / 180.0);

}

public double rad2deg(double rad) {

return (rad \* 180.0 / Math.PI);

}

public void readReducerLoc(){

try{

location.clear();

Statement stmt = DBCon.getCon().createStatement();

ResultSet rs = stmt.executeQuery("select \* from reducer");

while(rs.next()){

String reducer = rs.getString(1);

double lat = rs.getDouble(2);

double lon = rs.getDouble(3);

double dis = distance(17.4359786,78.4481956,lat,lon,'M');

Location loc = new Location();

loc.setReducer(reducer);

loc.setDistance(dis);

location.add(loc);

}

java.util.Collections.sort(location,new Location());

for(int i=0;i<location.size();i++){

Location loc = location.get(i);

System.out.println(loc.getReducer());

}

}catch(Exception e){

e.printStackTrace();

}

}

}

**DefineReducer.java**

package mapreduce;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.JPanel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.UIManager;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.Color;

import java.awt.Font;

import javax.swing.JComboBox;

import javax.swing.JOptionPane;

public class DefineReducer extends JFrame{

GradientPanel p1;

JPanel p2;

JLabel l1,l2,l3,l4;

JTextField tf1,tf2;

JButton b1;

JComboBox c1;

Font f1;

public DefineReducer(){

super("Define Reducer");

p1 = new GradientPanel(600,200);

p1.setLayout(null);

l1 = new JLabel("Define Reducer Location Screen");

l1.setFont(new Font("Courier New",Font.BOLD,18));

l1.setBounds(250,20,400,30);

p1.add(l1);

l2 = new JLabel("Reducer Name");

l2.setFont(f1);

l2.setBounds(200,60,100,30);

p1.add(l2);

c1 = new JComboBox();

c1.addItem("Reducer1");

c1.addItem("Reducer2");

c1.setFont(f1);

c1.setBounds(300,60,130,30);

p1.add(c1);

l3 = new JLabel("Latitude");

l3.setFont(f1);

l3.setBounds(200,110,100,30);

p1.add(l3);

tf1 = new JTextField(15);

tf1.setFont(f1);

tf1.setBounds(300,110,130,30);

p1.add(tf1);

l4 = new JLabel("Longitude");

l4.setFont(f1);

l4.setBounds(200,160,100,30);

p1.add(l4);

tf2 = new JTextField(15);

tf2.setFont(f1);

tf2.setBounds(300,160,130,30);

p1.add(tf2);

b1 = new JButton("Save Reducer");

b1.setFont(f1);

b1.setBounds(220,210,140,30);

p1.add(b1);

b1.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent ae){

login();

}

});

getContentPane().add(p1,BorderLayout.CENTER);

}

public void clear(){

tf1.setText("");

tf2.setText("");

}

public void login(){

String reducer = c1.getSelectedItem().toString().trim();

String lat = tf1.getText();

String lon = tf2.getText();

if(lat == null || lat.trim().length() <= 0){

JOptionPane.showMessageDialog(this,"Latitude must be enter");

tf1.requestFocus();

return;

}

if(lon == null || lon.trim().length() <= 0){

JOptionPane.showMessageDialog(this,"Longitude must be enter");

tf2.requestFocus();

return;

}

double lat1 = 0;

double lon1 = 0;

try{

lat1 = Double.parseDouble(lat.trim());

}catch(NumberFormatException nfe){

JOptionPane.showMessageDialog(this,"Latitude must be decimal value only");

tf1.requestFocus();

return;

}

try{

lon1 = Double.parseDouble(lon.trim());

}catch(NumberFormatException nfe){

JOptionPane.showMessageDialog(this,"Longitude must be decimal value only");

tf2.requestFocus();

return;

}

try{

String msg = DBCon.addReducer(reducer,lat.trim(),lon.trim());

if(msg.equals("success")){

JOptionPane.showMessageDialog(this,"Reducer details added");

setVisible(false);

}else{

JOptionPane.showMessageDialog(this,"Error in adding reducer details");

}

}catch(Exception e){

e.printStackTrace();

}

}

}

**Chart.java**

package mapreduce;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.GradientPaint;

import javax.swing.JPanel;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.axis.CategoryAxis;

import org.jfree.chart.axis.CategoryLabelPositions;

import org.jfree.chart.axis.NumberAxis;

import org.jfree.chart.labels.StandardCategorySeriesLabelGenerator;

import org.jfree.chart.plot.CategoryPlot;

import org.jfree.chart.plot.PlotOrientation;

import org.jfree.chart.renderer.category.BarRenderer;

import org.jfree.data.category.CategoryDataset;

import org.jfree.data.category.DefaultCategoryDataset;

import org.jfree.ui.ApplicationFrame;

import org.jfree.ui.RefineryUtilities;

import java.util.ArrayList;

import java.awt.event.WindowEvent;

import javax.swing.JScrollPane;

import org.jfree.chart.ChartUtilities;

public class Chart extends ApplicationFrame{

static int noagg;

static int agg;

static String title;

public Chart(String paramString,int a1,int a2){

super(paramString);

noagg = a1;

agg = a2;

JPanel localJPanel = createDemoPanel();

localJPanel.setPreferredSize(new Dimension(800, 370));

JScrollPane jsp = new JScrollPane(localJPanel);

setContentPane(localJPanel);

}

private static CategoryDataset createDataset(){

DefaultCategoryDataset localDefaultCategoryDataset = new DefaultCategoryDataset();

localDefaultCategoryDataset.addValue(noagg,"No Aggregation","No Aggregation");

localDefaultCategoryDataset.addValue(agg,"Aggregation","Aggregation");

return localDefaultCategoryDataset;

}

public void windowClosing(WindowEvent we) {

this.setVisible(false);

}

private static JFreeChart createChart(CategoryDataset paramCategoryDataset){

JFreeChart localJFreeChart = ChartFactory.createBarChart(title,"Technique Name", "Processing Time", paramCategoryDataset, PlotOrientation.VERTICAL, true, true, false);

CategoryPlot localCategoryPlot = (CategoryPlot)localJFreeChart.getPlot();

localCategoryPlot.setDomainGridlinesVisible(true);

localCategoryPlot.setRangeCrosshairVisible(true);

localCategoryPlot.setRangeCrosshairPaint(Color.blue);

NumberAxis localNumberAxis = (NumberAxis)localCategoryPlot.getRangeAxis();

localNumberAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());

BarRenderer localBarRenderer = (BarRenderer)localCategoryPlot.getRenderer();

localBarRenderer.setDrawBarOutline(false);

GradientPaint localGradientPaint1 = new GradientPaint(0.0F, 0.0F, Color.blue, 0.0F, 0.0F, new Color(0, 0, 64));

GradientPaint localGradientPaint2 = new GradientPaint(0.0F, 0.0F, Color.green, 0.0F, 0.0F, new Color(0, 64, 0));

GradientPaint localGradientPaint3 = new GradientPaint(0.0F, 0.0F, Color.red, 0.0F, 0.0F, new Color(64, 0, 0));

localBarRenderer.setSeriesPaint(0, localGradientPaint1);

localBarRenderer.setSeriesPaint(1, localGradientPaint2);

localBarRenderer.setSeriesPaint(2, localGradientPaint3);

localBarRenderer.setLegendItemToolTipGenerator(new StandardCategorySeriesLabelGenerator("Tooltip: {0}"));

CategoryAxis localCategoryAxis = localCategoryPlot.getDomainAxis();

localCategoryAxis.setCategoryLabelPositions(CategoryLabelPositions.createUpRotationLabelPositions(0.5235987755982988D));

return localJFreeChart;

}

public static JPanel createDemoPanel(){

JFreeChart localJFreeChart = createChart(createDataset());

return new ChartPanel(localJFreeChart);

}

}

**DBCon.java**

package mapreduce;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

import java.util.ArrayList;

public class DBCon{

private static Connection con;

public static Connection getCon()throws Exception {

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost/mapreduce","root","root");

return con;

}

public static String addReducer(String reducer,String lat,String lon)throws Exception{

String msg="fail";

con = getCon();

PreparedStatement stmt=con.prepareStatement("delete from reducer where reducer\_name=?");

stmt.setString(1,reducer);

stmt.executeUpdate();

PreparedStatement stat=con.prepareStatement("insert into reducer values(?,?,?)");

stat.setString(1,reducer);

stat.setString(2,lat);

stat.setString(3,lon);

int i=stat.executeUpdate();

if(i > 0)

msg = "success";

return msg;

}

}

**Reducer1.java**

package mapreduce;

import java.io.ObjectOutputStream;

import java.io.ObjectInputStream;

import java.net.Socket;

import java.net.ServerSocket;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Container;

import java.awt.Font;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JButton;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.SwingUtilities;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.UIManager;

public class Reducer1 extends JFrame{

ProcessThread thread;

JPanel p1,p2;

JLabel l1;

JScrollPane jsp;

JTextArea area;

Font f1,f2;

ServerSocket server;

Socket socket;

static StringBuilder sb = new StringBuilder();

public void start(){

try{

area.append("Node1 startup\n");

server = new ServerSocket(2222);

while(true){

socket = server.accept();

socket.setKeepAlive(true);

thread = new ProcessThread(socket,area);

thread.start();

}

}catch(Exception e){

e.printStackTrace();

}

}

public Reducer1(){

setTitle("Reducer1 Node");

p1 = new JPanel();

l1 = new JLabel("<html><body><center><font size=6 color=#f5ea01>Reducer1 Node</font></center></body></html>");

l1.setForeground(Color.white);

p1.add(l1);

p1.setBackground(Color.black);

f2 = new Font("Courier New", 1, 13);

p2 = new JPanel();

p2.setLayout(new BorderLayout());

area = new JTextArea();

area.setFont(f2);

jsp = new JScrollPane(area);

area.setEditable(false);

p2.add(jsp);

getContentPane().add(p1, "North");

getContentPane().add(p2, "Center");

addWindowListener(new WindowAdapter(){

@Override

public void windowClosing(WindowEvent we){

try{

if(socket != null){

socket.close();

}

server.close();

}catch(Exception e){

//e.printStackTrace();

}

}

});

}

public static void main(String a[])throws Exception{

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

Reducer1 r1 = new Reducer1();

r1.setVisible(true);

r1.setSize(600,400);

new ServerThread(r1);

}

}

**Reducer2.java**

package mapreduce;

import java.io.ObjectOutputStream;

import java.io.ObjectInputStream;

import java.net.Socket;

import java.net.ServerSocket;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Container;

import java.awt.Font;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JButton;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.SwingUtilities;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.UIManager;

public class Reducer2 extends JFrame{

ProcessThread thread;

JPanel p1,p2;

JLabel l1;

JScrollPane jsp;

JTextArea area;

Font f1,f2;

ServerSocket server;

Socket socket;

static StringBuilder sb = new StringBuilder();

public void start(){

try{

area.append("Node2 startup\n");

server = new ServerSocket(3333);

while(true){

socket = server.accept();

socket.setKeepAlive(true);

thread = new ProcessThread(socket,area);

thread.start();

}

}catch(Exception e){

e.printStackTrace();

}

}

public Reducer2(){

setTitle("Reducer2 Node");

p1 = new JPanel();

l1 = new JLabel("<html><body><center><font size=6 color=#f5ea01>Reducer2 Node</font></center></body></html>");

l1.setForeground(Color.white);

p1.add(l1);

p1.setBackground(Color.black);

f2 = new Font("Courier New", 1, 13);

p2 = new JPanel();

p2.setLayout(new BorderLayout());

area = new JTextArea();

area.setFont(f2);

jsp = new JScrollPane(area);

area.setEditable(false);

p2.add(jsp);

getContentPane().add(p1, "North");

getContentPane().add(p2, "Center");

addWindowListener(new WindowAdapter(){

@Override

public void windowClosing(WindowEvent we){

try{

if(socket != null){

socket.close();

}

server.close();

}catch(Exception e){

//e.printStackTrace();

}

}

});

}

public static void main(String a[])throws Exception{

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

Reducer2 r2 = new Reducer2();

r2.setVisible(true);

r2.setSize(600,400);

new ServerThread(r2);

}

}

**6. Testing**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

### Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

### Integration Testing

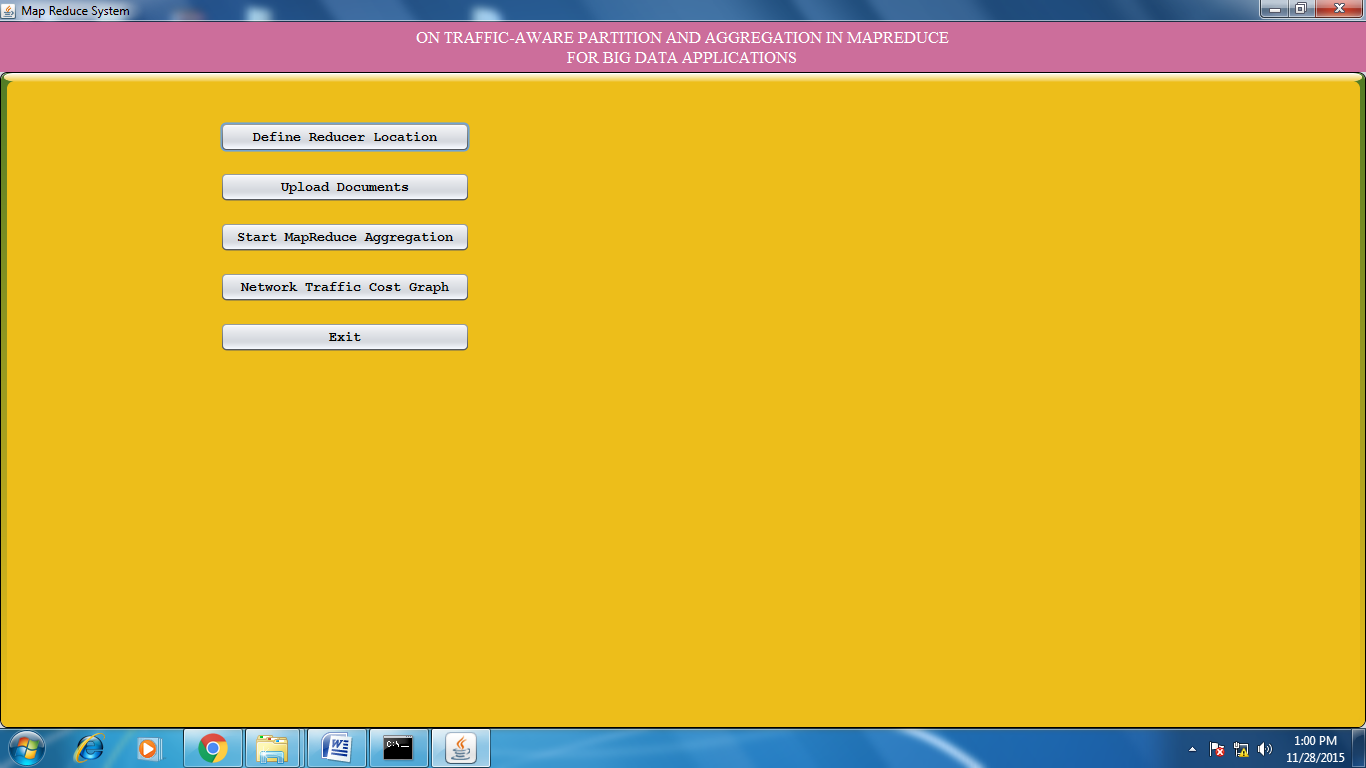
After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

### Acceptance Testing

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

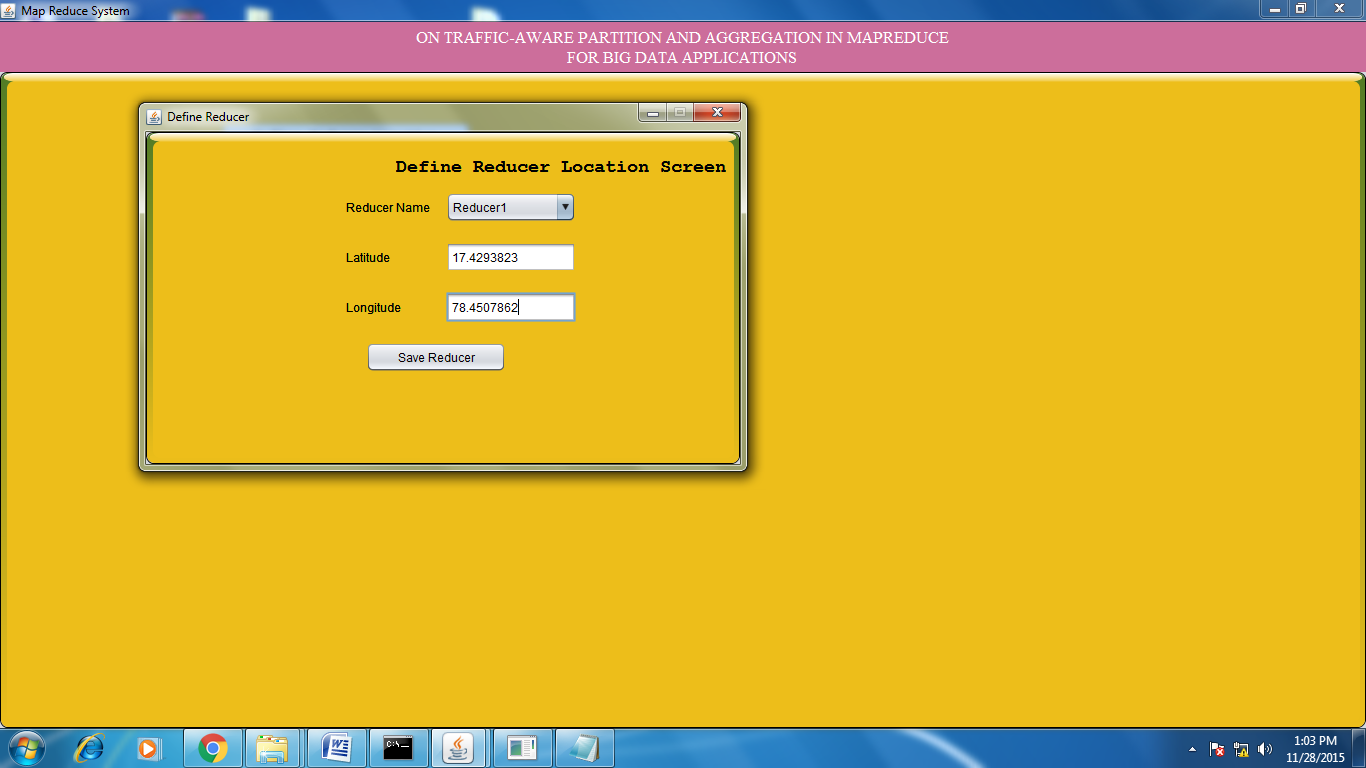
**7. Screen Shots**

Mapper application home screen:

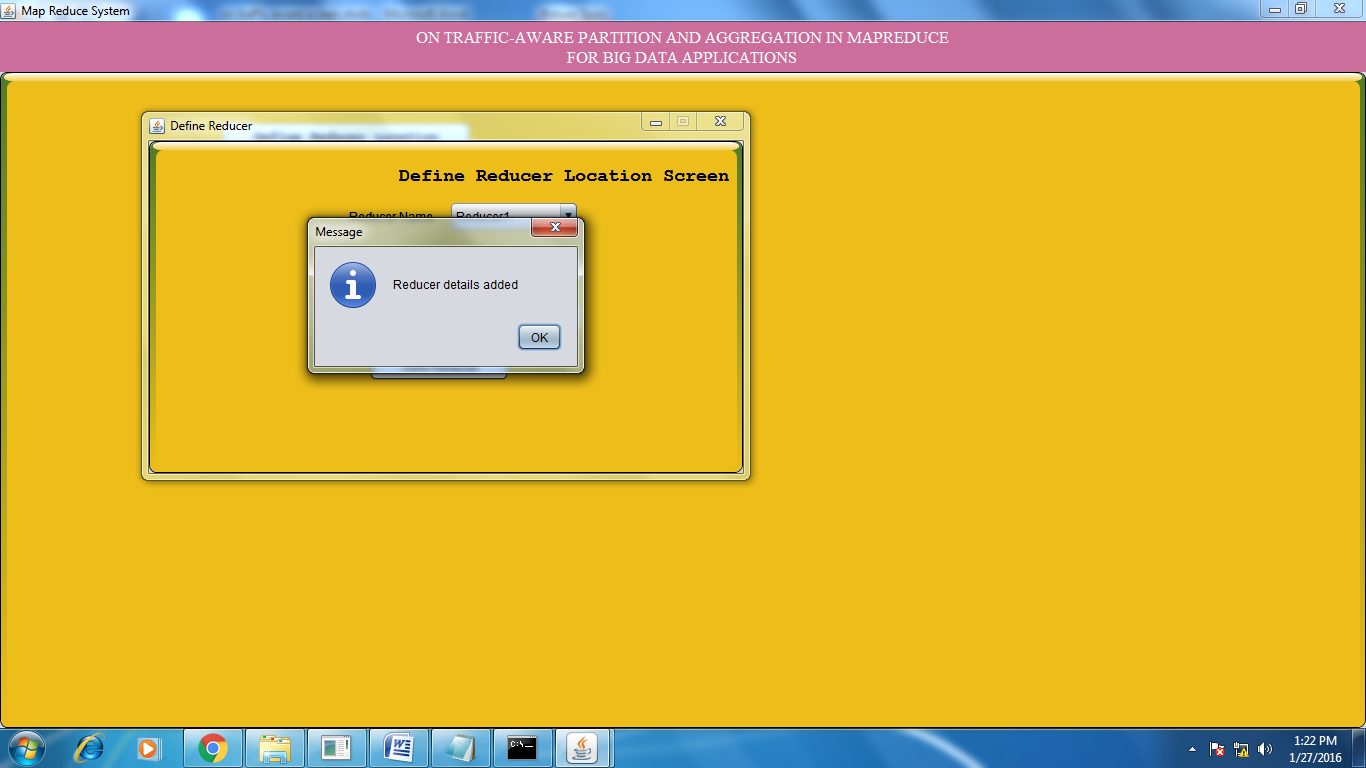


Click on Define Reducer location and add 2 reducer locations.

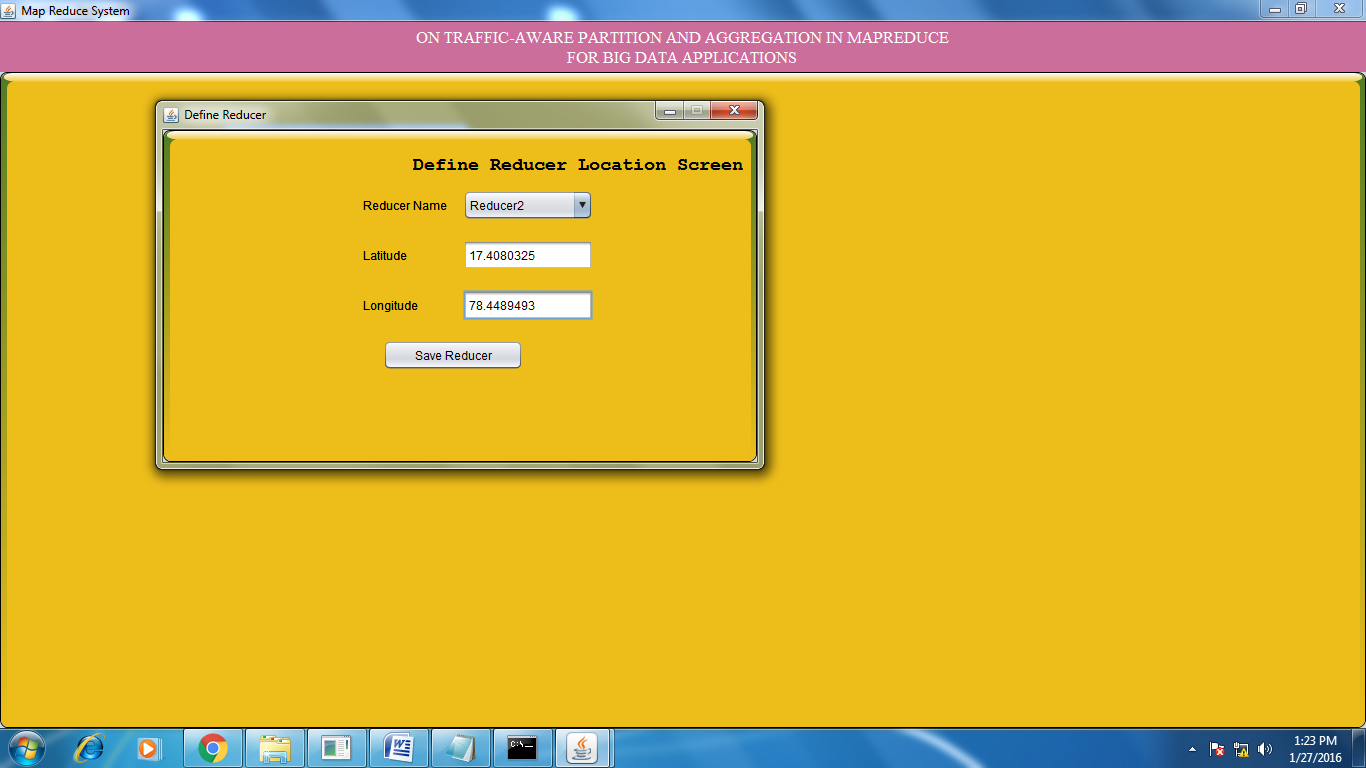
Defining reducer 1 at certain location. (For location we are taking the latitude and longitude values)



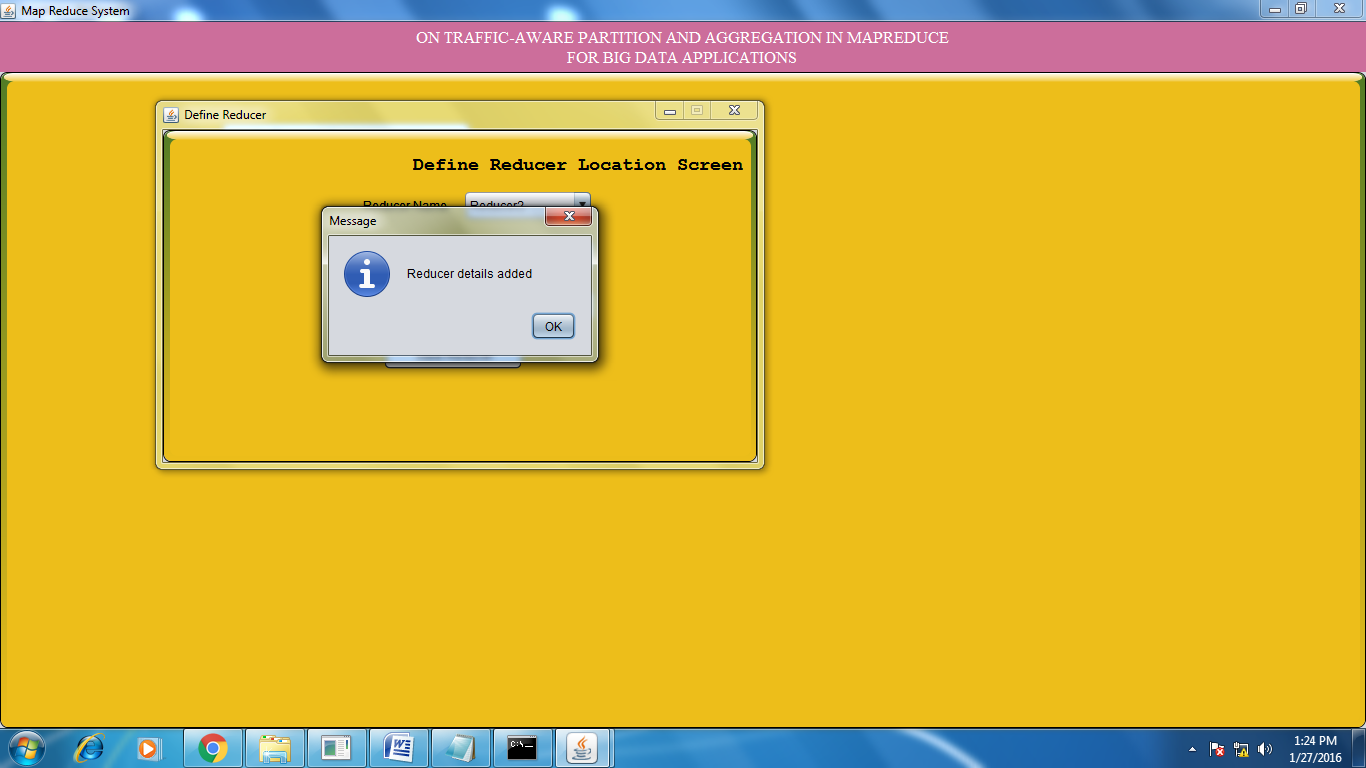
After successfully adding the reducer 1 details:

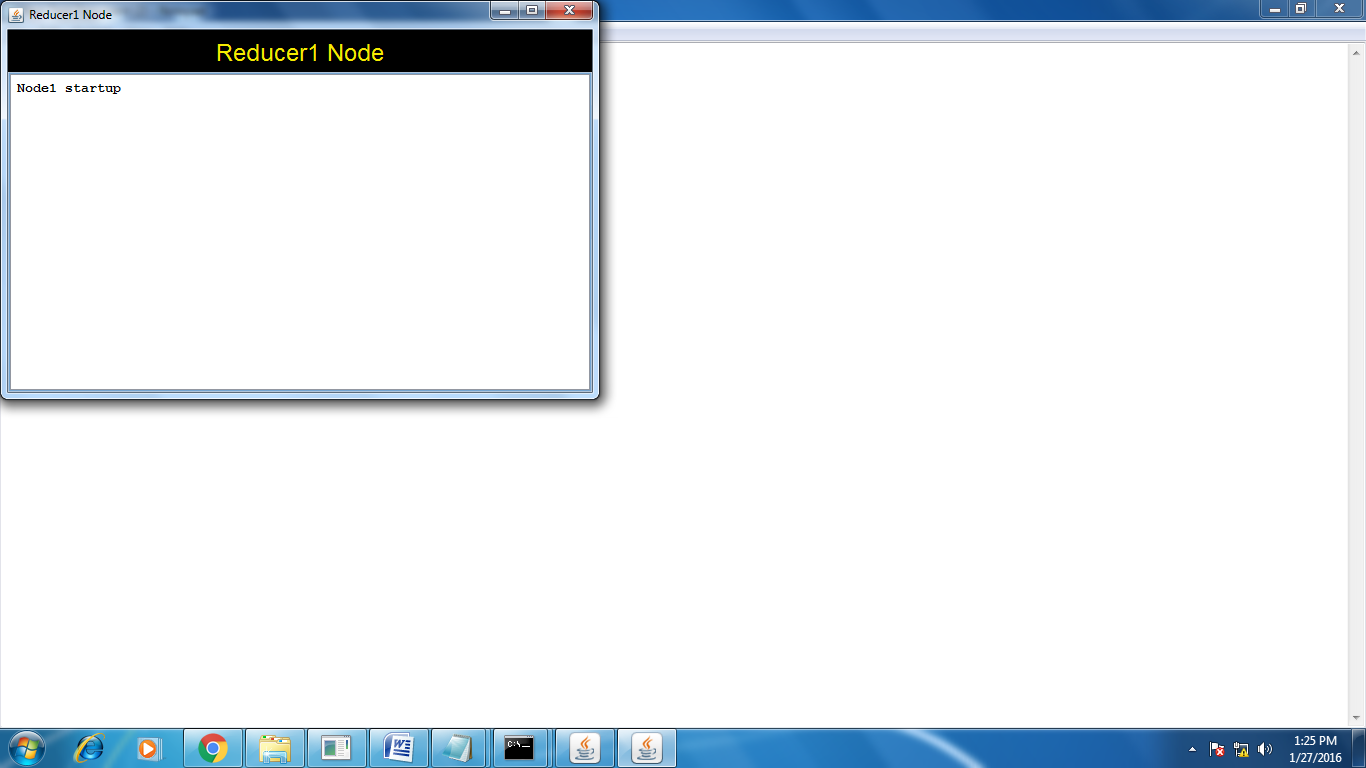
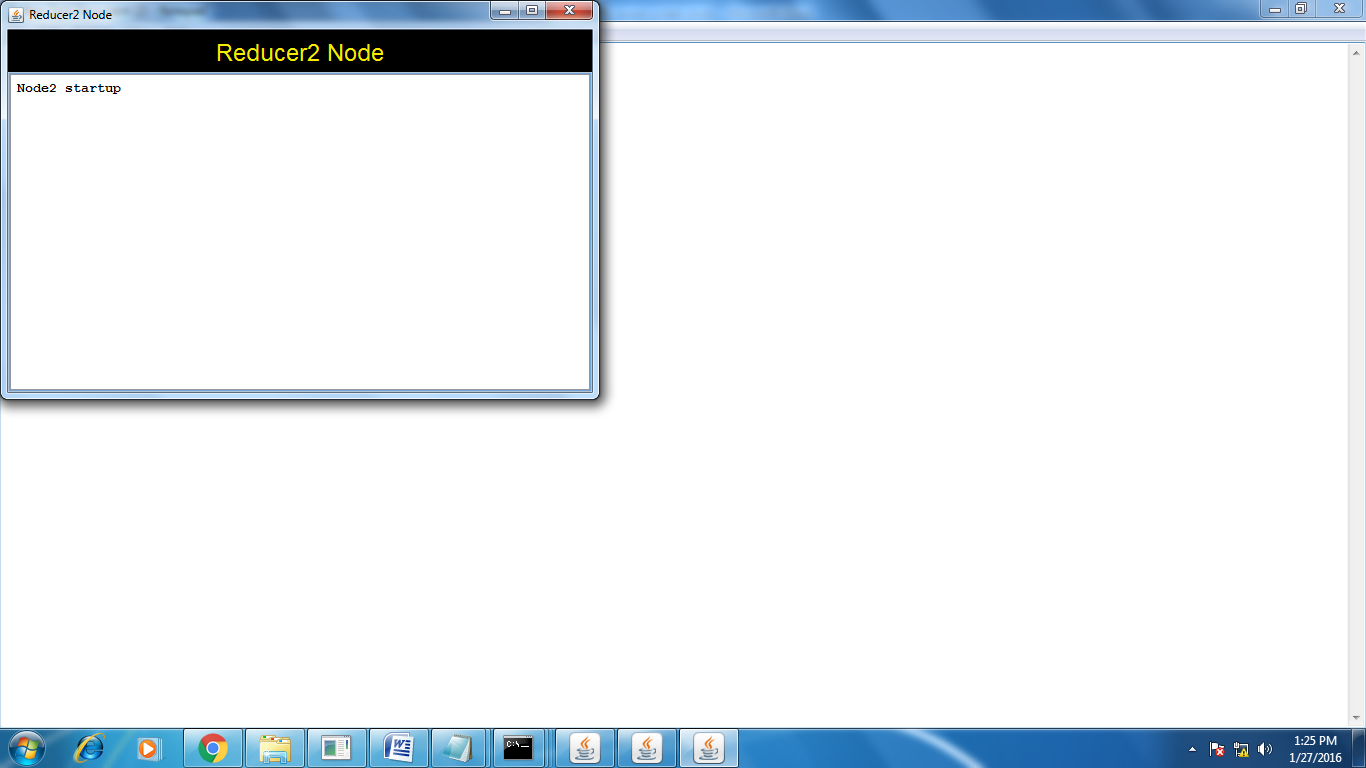


Defining reducer 2 at certain location

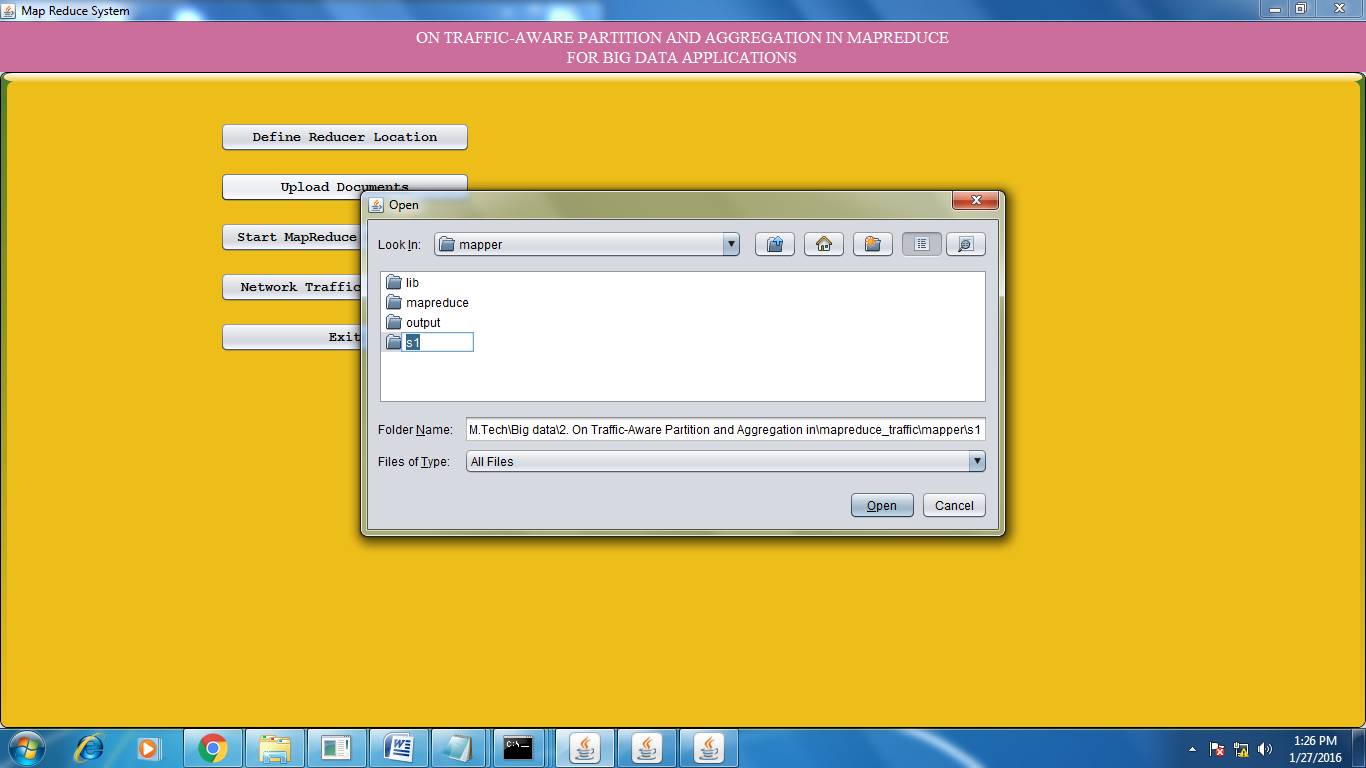


After successfully adding the reducer 2 details:

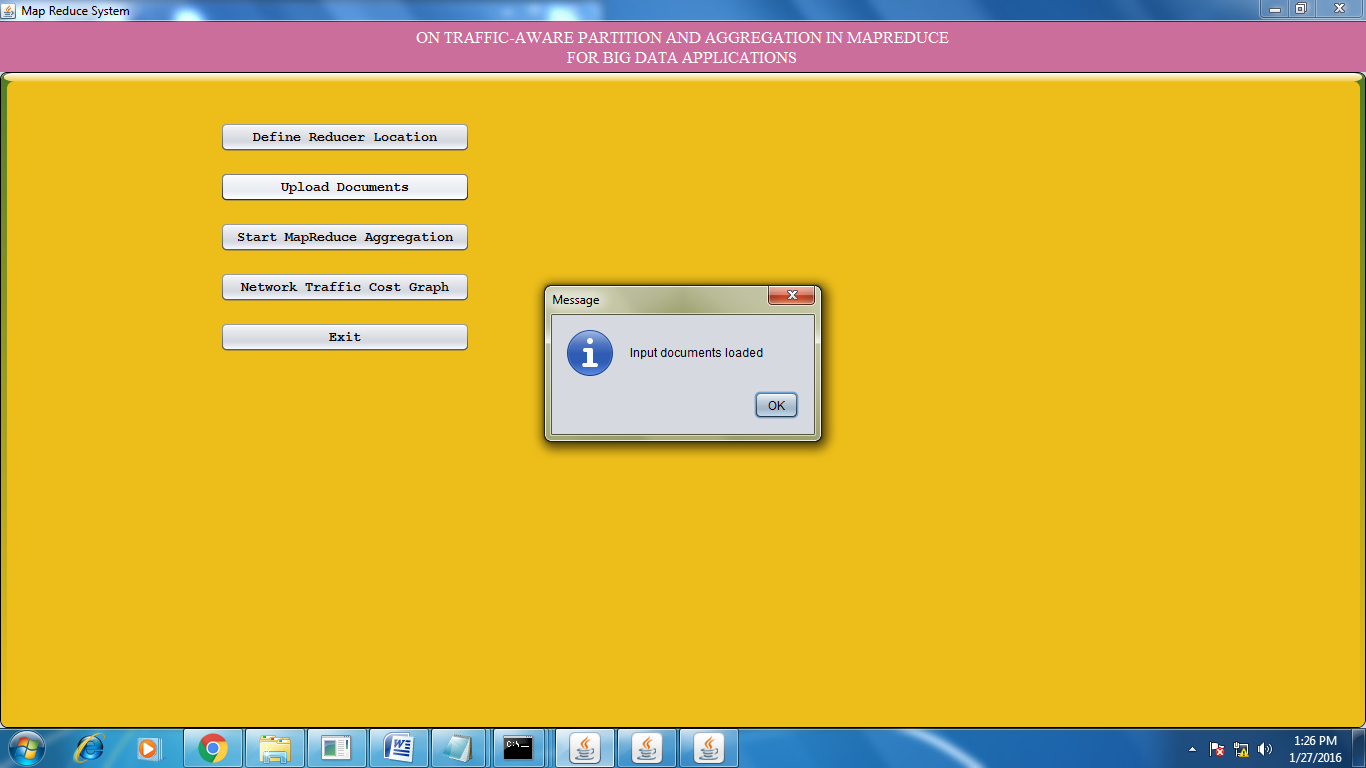


Run the reducer applications (reducer 1 & reducer 2): 

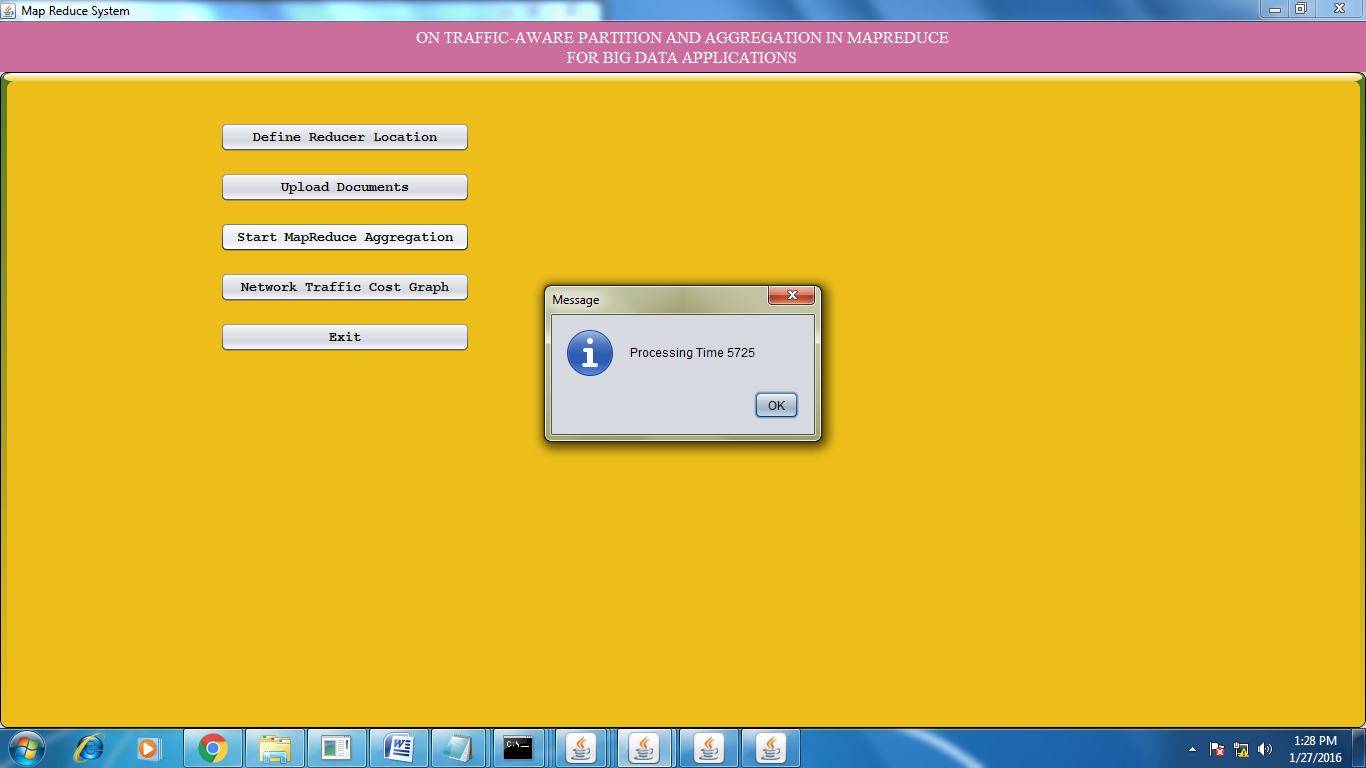
Click on upload documents and upload the input data to our application:



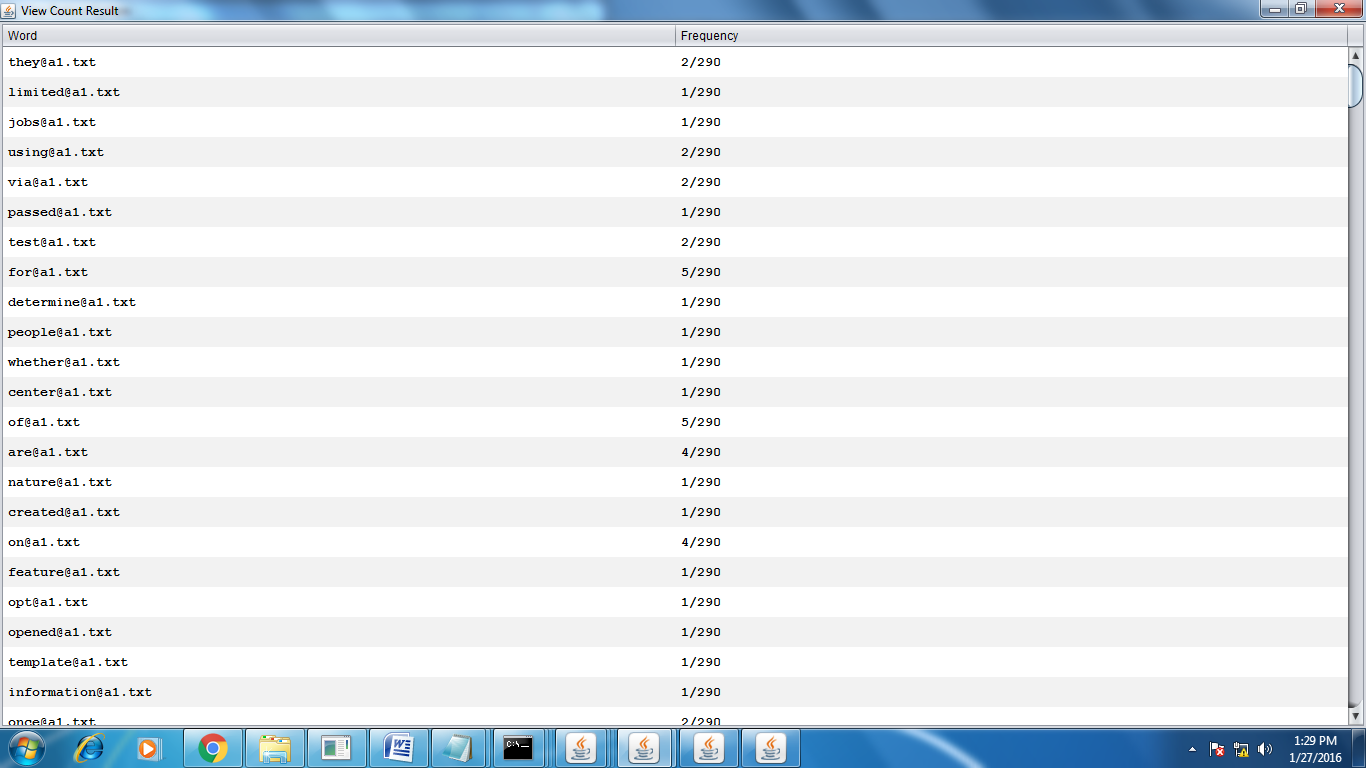
After successfully loading the input data:



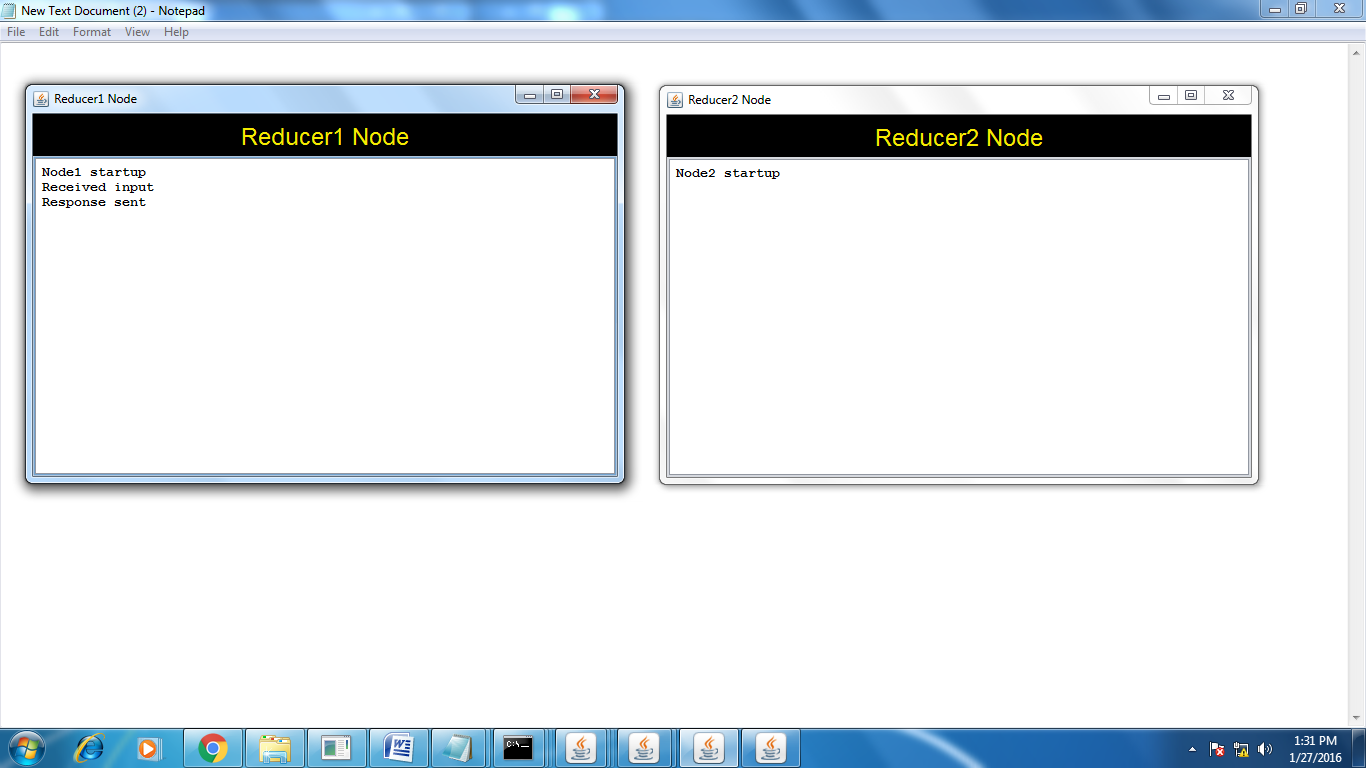
Click on start MapReduce Aggregation:



Aggregated data after successfully processing:

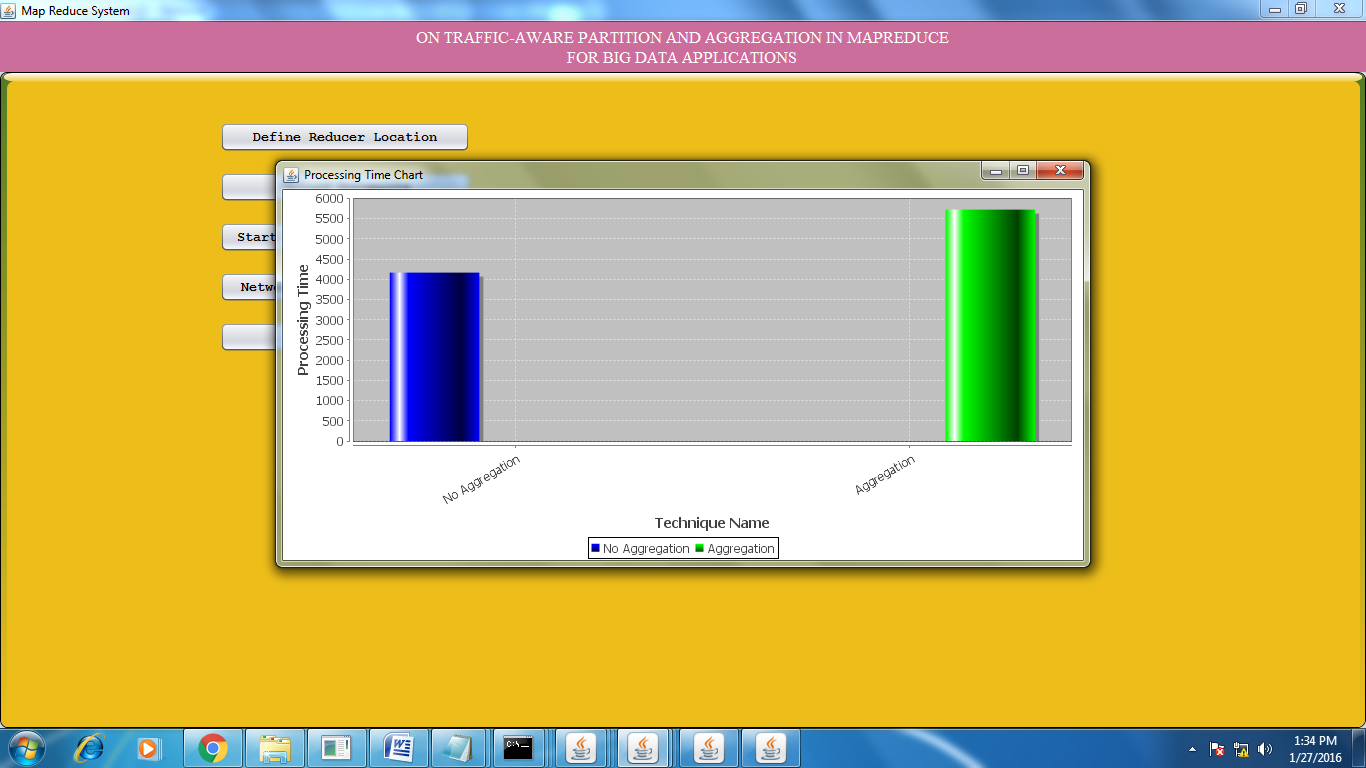


Reducer applications:



In the above, the request has been processed by Reducer 1, because the reducer 1 is nearer to the mapper location . (for mapper application location details refer Main.java line number 214)

Click on network traffic cost graph:



**8. Conclusion**

In this paper, we study the joint optimization of intermediate data partition and aggregation in MapReduce to minimize network traffic cost for big data applications.  
We propose a three-layer model for this problem and formulate it as a mixed-integer nonlinear problem, which is then transferred into a linear form that can be solved by  
mathematical tools. To deal with the large-scale formulation due to big data, we design a distributed algorithm to solve the problem on multiple machines. Furthermore,  
we extend our algorithm to handle the MapReduce job in an online manner when some system parameters are not given. Finally, we conduct extensive simulations to evaluate our proposed algorithm under both offline cases and online cases. The simulation results demonstrate that our proposals can effectively reduce network traffic cost under various network settings.

**9. Bibliography**

[1] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, “Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality,” in *INFOCOM, 2013 Proceedings IEEE*.IEEE, 2013, pp. 1609–1617.

[3] F. Chen, M. Kodialam, and T. Lakshman, “Joint scheduling of processing and shuffle phases in mapreduce systems,” in *INFOCOM,2012 Proceedings IEEE*. IEEE, 2012, pp. 1143–1151.

[4] Y. Wang, W. Wang, C. Ma, and D. Meng, “Zput: A speedy data uploading approach for the hadoop distributed file system,” in *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–5.

[5] T. White, *Hadoop: the definitive guide: the definitive guide*. ” O’Reilly  
Media, Inc.”, 2009.

[6] S. Chen and S. W. Schlosser, “Map-reduce meets wider varieties of applications,” *Intel Research Pittsburgh, Tech. Rep. IRP-TR-08-05*, 2008.

[7] J. Rosen, N. Polyzotis, V. Borkar, Y. Bu, M. J. Carey, M. Weimer, T. Condie, and R. Ramakrishnan, “Iterative mapreduce for large scale machine learning,” *arXiv preprint arXiv:1303.3517*, 2013.

[8] S. Venkataraman, E. Bodzsar, I. Roy, A. AuYoung, and R. S. Schreiber, “Presto: distributed machine learning and graph processing with sparse matrices,” in *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013, pp. 197–  
210.

[9] A. Matsunaga, M. Tsugawa, and J. Fortes, “Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications,” in *eScience, 2008. eScience’08. IEEE Fourth International Conference on*. IEEE, 2008, pp. 222–229.

[10] J. Wang, D. Crawl, I. Altintas, K. Tzoumas, and V. Markl, “Comparison of distributed data-parallelization patterns for big data analysis: A bioinformatics case study,” in *Proceedings of the Fourth International Workshop on Data Intensive Computing in the Clouds (DataCloud)*, 2013.

[11] R. Liao, Y. Zhang, J. Guan, and S. Zhou, “Cloudnmf: A mapreduce  
implementation of nonnegative matrix factorization for largescale biological datasets,” *Genomics, proteomics & bioinformatics*, vol. 12, no. 1, pp. 48–51, 2014.