Eliminating Homogeneous Cluster Setup for Efficient Parallel Data Processing

Piyush Saxena
M.Tech (CS&E)
Amity University,
Noida, India

Satyajit Padhy
M.Tech (CS&E)
Amity University,
Noida, India

Praveen Kumar
Assistant Professor,
Amity University,
Noida, India

ABSTRACT

This project proposes to eliminate homogeneous cluster setup in a parallel data processing environment. A homogeneous cluster setup supports static nature of processing which is a huge disadvantage for optimising the response time towards clients. Parallel data processing is performed more often in today's internet and it is very important for the server to deliver the services to its client in optimal time. In order to avail utmost client satisfaction, the server needs to eliminate homogeneous cluster setup that is encountered usually in parallel data processing. The homogeneous cluster setup is static in nature and dynamic allocation of resources is not possible in this kind of environment. The project will also make sure that the user gets its entire requirement fulfilled in optimal time. This will improve the overall resource utilization and, consequently, reduce the processing cost.

General Terms

Map reduce algorithm, Homogeneous cluster setup

Keywords

Data mining, Data warehousing, Parallel data processing.

1. INTRODUCTION

In today's digital generation, a huge amount of data is been processed parallel in the internet. Providing optimal data processing with good response time improvises the output of parallel data processing. There are many users that try to access the same data over the web and it is a challenging task for the server to deliver optimal result. The vast amount of data they have to deal with every day has made traditional database solutions prohibitively expensive. Instead, these companies have popularized an architectural paradigm based on a large number of commodity servers. There are problems like processing large documents split into several independent subtasks, distributed among the available nodes, and computed in parallel.

Parallel data processing is a key feature in accessing and operating on huge set of data's. [2] There are several ways available to process data parallel which improvises time and response. Today's framework has a huge disadvantage that can be termed by a homogeneous cluster setup. A homogeneous cluster setup is a cluster of nodes which consists of a cluster head node and many sensor nodes connected to it. The cluster head node is responsible to direct a task to the sensor nodes for executing it. In a homogeneous cluster setup, all the sensor nodes avail uniform battery energy and all of them terminate at the same instant of time. The main disadvantage with a homogeneous cluster setup is that it is static in nature, i.e. once all the sensor nodes are created and started to execute then no more extra sensor nodes can be added further in that cluster.

It is quite evident that the static nature of a homogeneous cluster setup is a huge disadvantage in parallel data processing. Once the nodes are created and have started executing there cannot be addition of any further nodes if required to be added dynamically. The dynamic factors are absent basically if there exists a homogeneous cluster setup. In this proposed project, the job manager acts like a cluster head node and all the task managers act like sensor nodes. The objective of this project is to motivate dynamic allocation of resources which can be achieved more efficiently if we eliminate homogeneous cluster setup.

[4]To be more precise, there is a job manager (main server) is allocated with a job it then divides that job into many sub jobs and it allocates to each task manager. Now once this cluster is setup and the parallel data processing begins, there can be no possible ways by which we can add more task managers or eliminate any executed task managers until all have executed. This is an ambiguous situation when there can be no resource allocation during the middle of data processing. This creates a problem for the server (Job Manager) to offer complete results to its users. Parallel data processing is more efficient if it can be executed dynamically and this dynamic environment improves the optimum response time. If we eliminate the homogeneous cluster setup, any number of sensor nodes can be created at any instant of time.

2. BACKGROUND

There are several amount data being processed in today's web. There are several challenges during parallel processing of this huge amount of data. During our research [5], there were many ambiguities we came across regarding parallel data processing.

The ambiguities were like:-

1) Delayed response time due to homogeneous cluster setup
2) Resources cannot be allocated dynamically once the number of task managers are created (static cluster)
3) High traffic if data with many peers in action
4) Data access used to be slower if the required data is unavailable with the server.

All these challenges being very generic, the most important problem was if the data to be accessed by the user was of huge size than the processing becomes slower. Facing these challenges regarding parallel data processing, there was a straightforward approach that was deployed in our proposed project.

The ***first scenario*** that [1] was applied in our research is the use of map reduce algorithm. This algorithm is the most effective algorithm that can be used for parallel data processing of large data. This map reduce algorithm stated a divide & conquer structure of working with data. This algorithm made it easier for the host server (job manager) to handle the job efficiently. It breakdowns the job into many sub jobs and execute them individually with the help of task managers.

The ***second scenario*** was all about eliminating the homogeneous cluster setup [19] of network. This will allow the allocation of resources dynamically to the host server at any instant of time. In order to eliminate homogeneous cluster setup, we need to avoid static nature of the cluster of nodes and allow addition of task managers to the cluster at any instant of time. The transformation of homogeneous cluster to heterogeneous cluster network will avoid static nature of the cluster and even reduce the overall hardware cost. Dynamic allocation of resources allows optimizing the parallel data processing in a new manner. This methodology offers a new scope of viewing these given challenges and moulding it to operate in an efficient manner.

The experiment is analyzed by taking into account various time slots that allows imagining the whole operation between job manager, task manager & user.

Once a user has fit his program into the required map and reduce pattern, the execution framework takes care of splitting the job into subtasks, distributing and executing them. A single Map Reduce job always consists of a distinct map and reduce program. Server - Client computing or networking is a distributed application architecture that partitions tasks or workloads between service providers (servers) and service requesters, called clients.

Often clients and servers operate over a computer network on separate hardware. A server machine is a high-performance host that is running one or more server programs which share its resources with clients. A client also shares any of its resources; Clients therefore initiate communication sessions with servers which await incoming requests.

Processing is based on implementation of the theorem uses (network-based) search operations as off the shelf building blocks. Thus, the NAP query evaluation methodology is readily deployable on existing systems, and can be easily adapted to different network storage schemes. In this case, the queries are evaluated in a batch. We propose the network-based anonymization and processing (NAP) framework, the first system for K- anonymous query processing in road networks. NAP relies on a global user ordering that satisfies reciprocity and guarantees K-anonymity. [11] We identify the ordering characteristics that affect subsequent processing, and qualitatively compare alternatives. Then, we propose query evaluation techniques that exploit these characteristics. In addition to user privacy, NAP achieves low computational and communication costs, and quick responses overall. It is readily deployable, requiring only basic network operations.

3. PROJECT SYSTEM OVERVIEW

In recent years a variety of systems has been proposed to facilitate web warehousing has been developed. Although these systems typically share common goals (e.g. to hide issues of parallelism or fault tolerance), they aim at different fields of application. [7] Map Reduce algorithm is designed to run data analysis jobs on a large amount of data, i.e. in order to improvise the parallel data processing between large number of nodes (users) and servers.

The proposed system also demonstrates the discrete allocation of resources that are not available on the host server but are available on remote servers, parallel while processing the present data. The proposed framework allows a platform for the server and users in efficient and optimized parallel data processing. It allows the job manager to allocate resources [4] at any instant of time and this improvises the response time. Hence the problem of a homogeneous cluster network is eliminated and thus it is more optimized approach.

Once a user has fit his data for processing into the required map and reduce pattern [13], the execution framework takes care of splitting the job into subtasks, distributing and executing them. A single Map Reduce job always consists of a distinct map and reduce pattern. The mapping is done by the job manager to its entire task manager with individual sub tasks and finally all the task managers execute each of their tasks and reduce it to one single solution and return it back to the user. The map-reduce algorithm works as a divide & conquer approach and it is very efficient in parallel data processing. The proposed system also offers dynamic allocation of resources to any of the task managers during execution. The allocated resources are then available on the host server always and it can be operated later.

4. FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.

The **client/server model** is a computing model that acts as a distributed application which partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients [8] and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server machine is a high performance host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Whereas the Servers take the request from the client and try to fulfill these requests by providing the resources the clients need.

A job manager is a computer application for controlling, managing and splitting the request of resources/files from the client. The job manager accepts the job to be executed as a request from the client and accordingly splits it as large number of packets. This large number of packets is allocated to the task managers for executing it in optimal time.

Generally large in number Task Managers are the part of the software that manage the responses given by the Job Manager and try to execute these responses n return the result of these set of responses to the client. The task manager is responsible for executing the individual packets allocated to them by the job manager.

5. OBJECTIVE OF THE PROJECT

The objectives and purpose for this project are to improvise and optimize the scenario of parallel data processing by eliminating homogeneous cluster setup [12]. Millions of data

are accessed in the web by the user and it is the utmost responsibility of the server to provide satisfaction to the user. It is always viable to be on the other side but dealing with such huge amount of data everyday makes the situation more complicated. Hence there are very few loopholes in the current framework but these are enough to degrade the performance in parallel data processing.

Therefore the main objective and purpose of this project is to optimize [2] parallel data processing by avoiding homogeneous cluster setup. In order to avail utmost client satisfaction, the host server needs to be upgraded with the latest technology to fulfil all requirements. The homogeneous cluster setup is static in nature the proposed map reduce algorithm is used in this generic framework that can be deployed in this scenario. Another important goal of this project is to allocate resources or data dynamically to the host server (job manager) so that every requirement of resources can be fulfilled at any instant of time. The current problem of formation of a homogeneous cluster setup is eliminated so that any number of sensor nodes or task managers can be created at any instant of time. The static nature of existing framework of parallel data processing is terminated. This allows higher and sharper response time and avoiding delay in transfer. The below figure states that the job manager upon receiving a job divides it into many packets (files) and the task mangers execute those packets at an instant of time. The time is stated in the figure and the actual response time is highly optimised with the proposed framework.

The job manager should be aligned [17] with all its task managers to avail maximum optimization. The task managers are mapped by the job manager with many jobs and they all solve it individually which is later reduced to return it back to the client. This allows the load for the execution to be shared and the overall execution of huge sized data is more feasible in less time. Typically data of huge size are the toughest challenge to be dealt in the web for parallel data processing. This project makes sure that the user gets its entire requirement fulfilled in optimal time.

We discussed pros and cons of Map Reduce and classified its improvements. Map Reduce is simple [1] but provides good scalability and fault-tolerance for massive data processing. The performance evaluation gives a first impression on how the ability to assign specific jobs to specific task manager of a processing job, as well as the possibility to automatically allocate/de-allocate virtual machines in the course of a job execution, can help to improve the overall resource utilization and, consequently, reduce the processing cost.

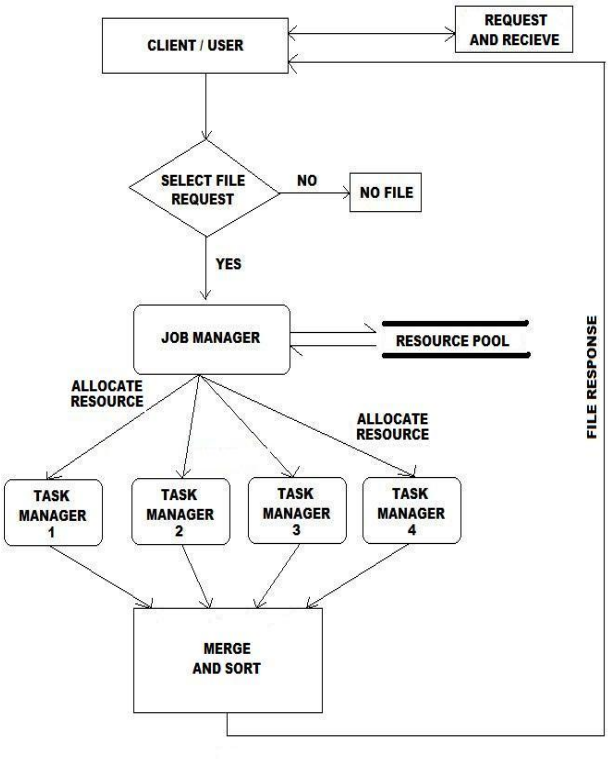

Fig 1: Architecture of proposed system

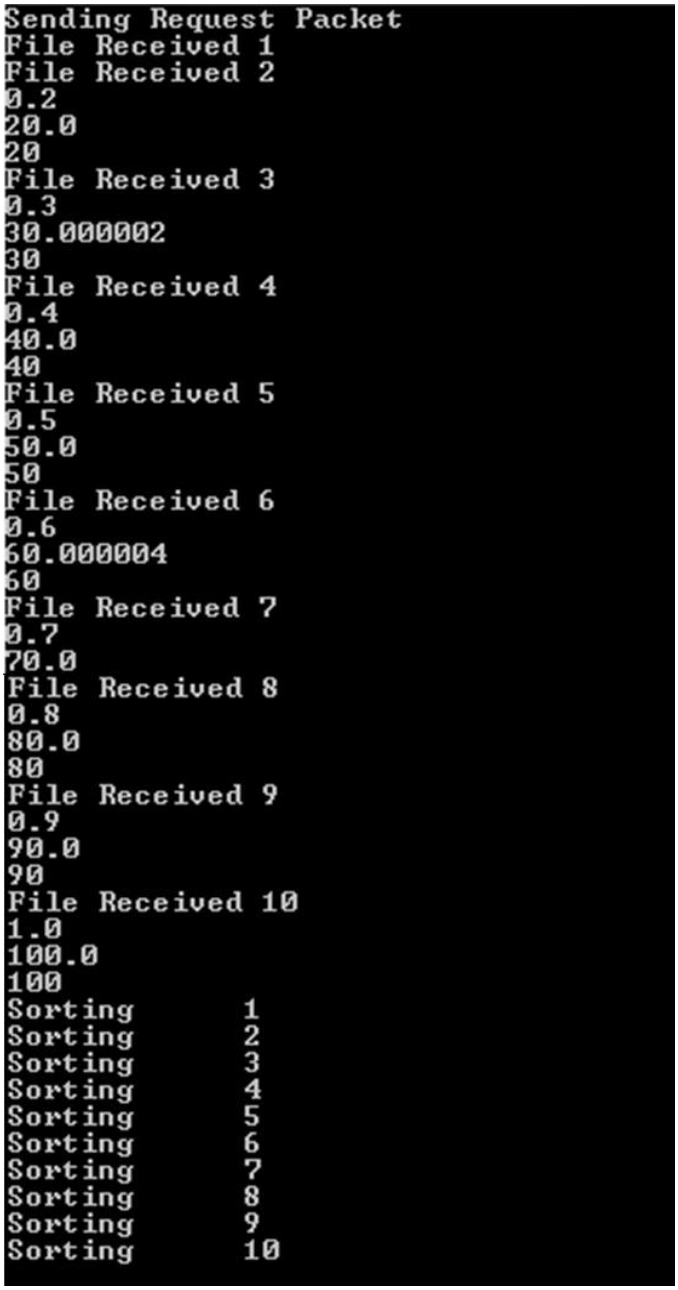

Fig 2: Command line output showing time instances of packet retrievals

6. MODULES OF THE PROJECTED SYSTEM

Client

This module deals with the Client or the Customer whose needs are to be fulfilled. The client always requests [10] to the server for executing a particular operation and send a response back to it accordingly. Nevertheless a client is always volatile about its operation. In our proposed project, the client selects the file that it wants to download. After the file is selected the client clicks on the download button. It is obvious though that the client always tries to request to download a file in this scenario. After clicking the download button, it waits for the server to send a response back. The status of the downloading interface is shown to the client so as to it can check the status of the downloading. The client is demonstrated by building a simple interface which consists of simple components. The client interface is event driven and the concept of swings in java is largely implemented. The client interface consists of a text area which displays all the files that are available presently in the database [6]. The text area is updated dynamically as per the uploading of resources dynamically. The name in the title bar of the interface is named as select file which basically states to select a file to download. There are two swing buttons included named as download and cancel. Both these buttons are event driven and upon clicking on the button a specified event takes place. The download button allows downloading a file and the cancel button closes the client interface.

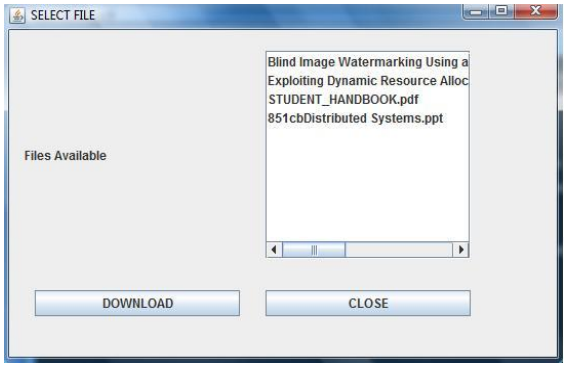

Fig 3: Client Interface

Server

A server is a computer system that is responsible for servicing the request made by the client. The server is normally located remotely [3] and is used to service requests from multiple clients. The server is always responsible for maintaining resources and allocating them as required by the host clients. The server even manages and controls data processing between server & client. In today's modern multiprocessor architecture, parallel data processing plays an important role. In order to have efficient parallel data processing where many clients participate to execute certain tasks, the server needs to execute data processing faster and efficiently. In our proposed project the server is an entity that services the request made by the client. The client request for downloading a file and the server makes it sure that the file is downloaded and opened at the end of downloading for the clients. The server interface consists of two important criterions that are job manager and task manager. The server interface even consists of a menu bar that has one menu element names as file. This menu consists of two menu elements named as resource allocation and exit. The resource allocation option should be selected only if the server needs to allocate resources dynamically at the same time when the file is getting downloaded. The server interface even consists of a drop down select menu which has the default value of parallel. This states that the [9] data distribution type is parallel and the data processing will be parallel in nature. The server upon getting the request from the client displays certain parameters in its command prompt output. It is the name of the file that is downloaded, the port numbers that will be involved during downloading, the total number of packets sent and received etc. The server even showed pictorially how the resources are allocated dynamically from the job manager to task manager.

Fig 4: Client Download Interface

Job Manager

The job manager is an essential component of the server. It's like the master component of the entire client server layout. The job manager [2] accepts the request that comes from the client and is responsible for processing it. The job manager follows divide and conquer approach for executing the job that. The job manager has to schedule and control the execution of the jobs and returning back a valid response to the client as per its request.

In the first scenario upon achieving a request from the client the job manager divides the job into many sub jobs or packets. It distributes evenly and randomly all the sub jobs and allocates it to the task managers. After the task managers finish executing their individual sub jobs, they return the resultant data to the job manager. All the sub jobs or packets returned by the task manager to the job manager would not be in sorted order and hence the job manager sorts all the packets as they were allocated initially. After sorting all the packets the job manager tries to merger all the executed packets into one data so that it can return a single solution to the client.

The command prompt terminal output shows the time instances at which the job manager sends a packet to the task managers. It even shows the sorting and merging of packets accordingly so as to return a single solution to the client. During dynamic resource allocation to the task managers, the job manager itself uploads the file to the task manager. The job manager is responsible for uploading the files to the task manager whenever there is a need of allocation of resources.

Fig 5: Job Manager

Task Manager

The task manager is an essential part of the server. It is like a basic block of execution that helps the job manager to execute the sub tasks and return it back to the job manager. The task manager responsibility is to execute the individual packets allocated to them and return it back to the job manager. When a client requests for downloading a file, the task manager is the one which is responsible for executing the operation and performs efficient parallel data processing. Upon the use of task managers with the job manager, the time for parallel data processing is much more efficient and it even supports dynamic resource allocation.

The task manager interface is a simple representation of the operation it performs. It shows that whenever a client makes a request to download the file, the server with the help of task manager tries to execute the request and return an optimal solution back to the client. It even represents the distributed type data processing which is parallel in nature and states that the file uploading is done with the task manager. After uploading the necessary file in the back end database, it is ready to return the request back to the client. In our proposed

project we have chose four task managers and each task manager is represented with a unique port number. This port number can be initialized by us but moreover it shows the participation of each task manager in the parallel data processing. It plays a major part even in allocating resources dynamically.

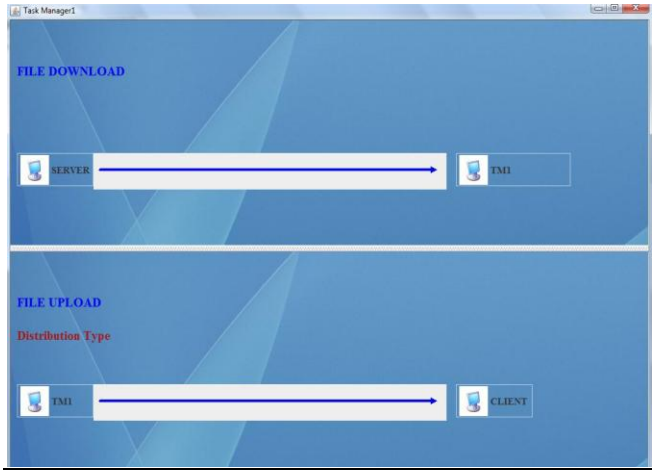

Fig 6: Task Manager

7. CONCLUSION AND FUTURE SCOPE

Today's digital generation executes its key ingredient at a regular basis and that is data [1]. Everyday there are many probabilities of parallel data processing. There are many search engines like Google or Yahoo which has to process a lot of data simultaneously for returning a response [3] to its users and even at a faster rate. The reliability and feasibility should not be hampered during this parallel data execution. Presently the mechanisms used for parallel data execution creates a homogeneous cluster setup within the network. The homogeneous cluster setup states that when there is a parallel downloading environment [8] under processing between the client and server, if the client at the same time requests for downloading a particular file and the server does not have it currently in it back end database then it causes a huge problem. The file cannot be uploaded until all the downloading under progress stops its execution.

In order to avoid this kind of scenario and to decrease the delay in response time from the server, we propose a framework that represents efficient parallel data processing with [10] no homogeneous cluster setup. It makes sure that when a client request for a file that is not present in the server, it can dynamically allocate that resource or file to the client even at the same time all the parallel downloading scenario is under progress. This improves the reliability and response time since the client has to no more wait for its response. This framework defines a new level in parallel data processing that is not encountered in today's world.

The experimental results demonstrating the comparison of response time between the existing framework and the proposed framework is shown in fig 7. It clearly states that the existing framework takes more time to respond to a client request comparatively to proposed framework. Due to the obvious problem of homogeneous cluster setup, the existing framework posses delay due to static nature. The dynamic nature of the proposed framework enhances the response time. As the size of the data increases, the response time is demonstrated for both the framework.

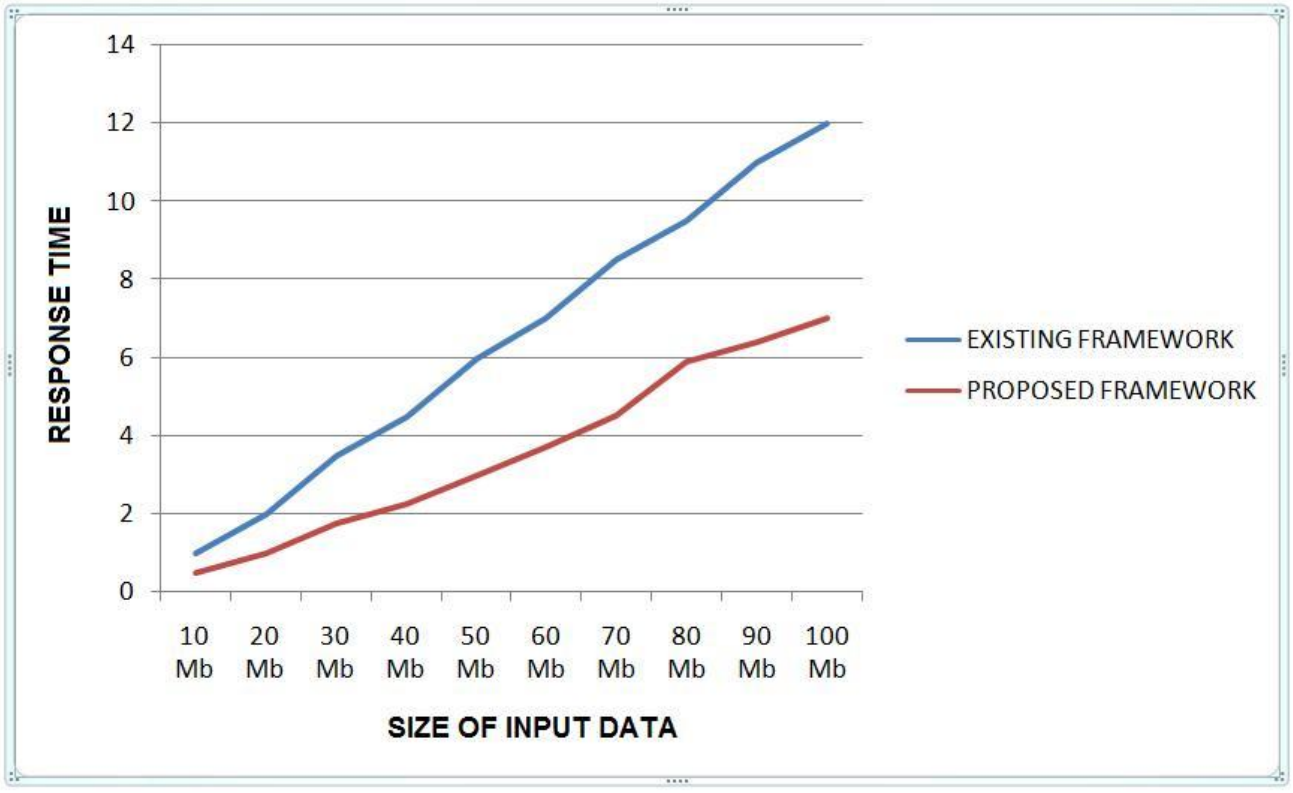

Fig 8: Experimental analysis of existing v/s proposed framework

8. REFERENCES

[1] "Parallel Data Processing with Map Reduce: A Survey" by Kyong-Ha Lee and Yoon-Joon Lee, Department of Computer Science KAIST, December 2011.

[2] Query Optimization for Massively Parallel Data Processing by Sai Wu , Feng Li, Sharad Mehrotra, Beng Chin Ooi School of Computing, National University of Singapore, March 2012

[3] S. Babu. Towards automatic optimization of map reduce programs. In Proceedings of the 1st ACM symposium on Cloud computing, pages 137–142, 2010.

[4] Parallel Data Processing: http://server-demo-ec2.cloveretl.com/clover/docs/clustering-parallel-processing.html

[5] H. chih Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-Reduce-Merge: Simplified Relational Data Processing on Large clusters. In SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 1029–1040, New York, NY, USA, 2007. ACM.

[6] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[7] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. Sci. Program 13(3):219–237, 2005.

[8] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the Data: Parallel Analysis with Sawzall. Sci. Program., 13(4):277–298, 2005.

[9] B. Li et al . A Platform for Scalable One-Pass Analytics using MapReduce. In Proceedings of the 2011 ACM SIGMOD, 2011.

[10] D. Jiang et al. Map-join-reduce: Towards scalable and efficient data analysis on large clusters. IEEE Transactions on Knowledge and Data Engineering, 2010.

[11] Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, David E. Culler, Joseph M. Hellerstein, and David A. Patterson. High-performance sorting on networks of workstations. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, May 1997.

[12] William Gropp, Ewing Lusk, and Anthony Skjellum. Using MPI: Portable Parallel Programming with the Message-Passing Interface. MIT Press, Cambridge, MA, 1999.

[13] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: The Condor experience. Concurrency and Computation: Practice and Experience, 2004.

[14] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wychoff, and R. Murthy, "Hive - a warehousing solution over a map-reduce framework," in VLDB, 2009.

[15] D. DeWitt and J. Gray, "Parallel database systems: the future of high performance database systems," Commun. ACM, 1992.

[16] S. Fushimi, M. Kitsuregawa, and H. Tanaka, "An overview of the system software of a parallel relational database machine grace," in VLDB '86: Proceedings of the 12th International Conference on Very Large Data Bases. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1986, pp. 209–219.

[17] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan, "Interpreting the data: Parallel analysis with sawzall," Sci. Program., vol. 13, no. 4, pp. 277–298, 2005.

[18] M. Ziane, M. Za¨ıt, and P. Borla-Salamet, "Parallel query processing with zigzag trees," The VLDB Journal, vol. 2, no. 3, pp. 277–302, 1993

[19] Homogeneous vs Heterogeneous Clustered Sensor Networks: A Comparative Study by Vivek Mhatre, Catherine Rosenberg School of Electrical and Computer Eng., Purdue University, West Lafayette, IN 47907-1285.

AUTHOR'S PROFILE

Praveen Kumar Assistant Professor, Amity University, India. Qualified as M.Sc., M.Tech., Ph.D. (Pursuing) with areas of interest in e-goverence / Java programming, Data mining

Satyajit Padhy B.Tech, Mtech (CSE) pursuing. Published two research papers on data processing in data warehousing and services of cloud computing respectively. Areas of interest lies in cloud computing, data warehousing and data mining.

Piyush Saxena B.Tech, Mtech (CSE) pursuing. Published two research papers on data processing in data warehousing and services of cloud computing respectively. Areas of interest lies in cloud computing, data warehousing and data mining.