LOVELY PROFESSIONAL UNIVERSITY

Transforming Education Transforming India

Enhancement in the resource utilization and sharing in distributed systems

A Dissertation Submitted

By

Awadhesh Kumar Shukla

To

Department of Computer Science and Engineering

In partial fulfilment of the Requirement for the

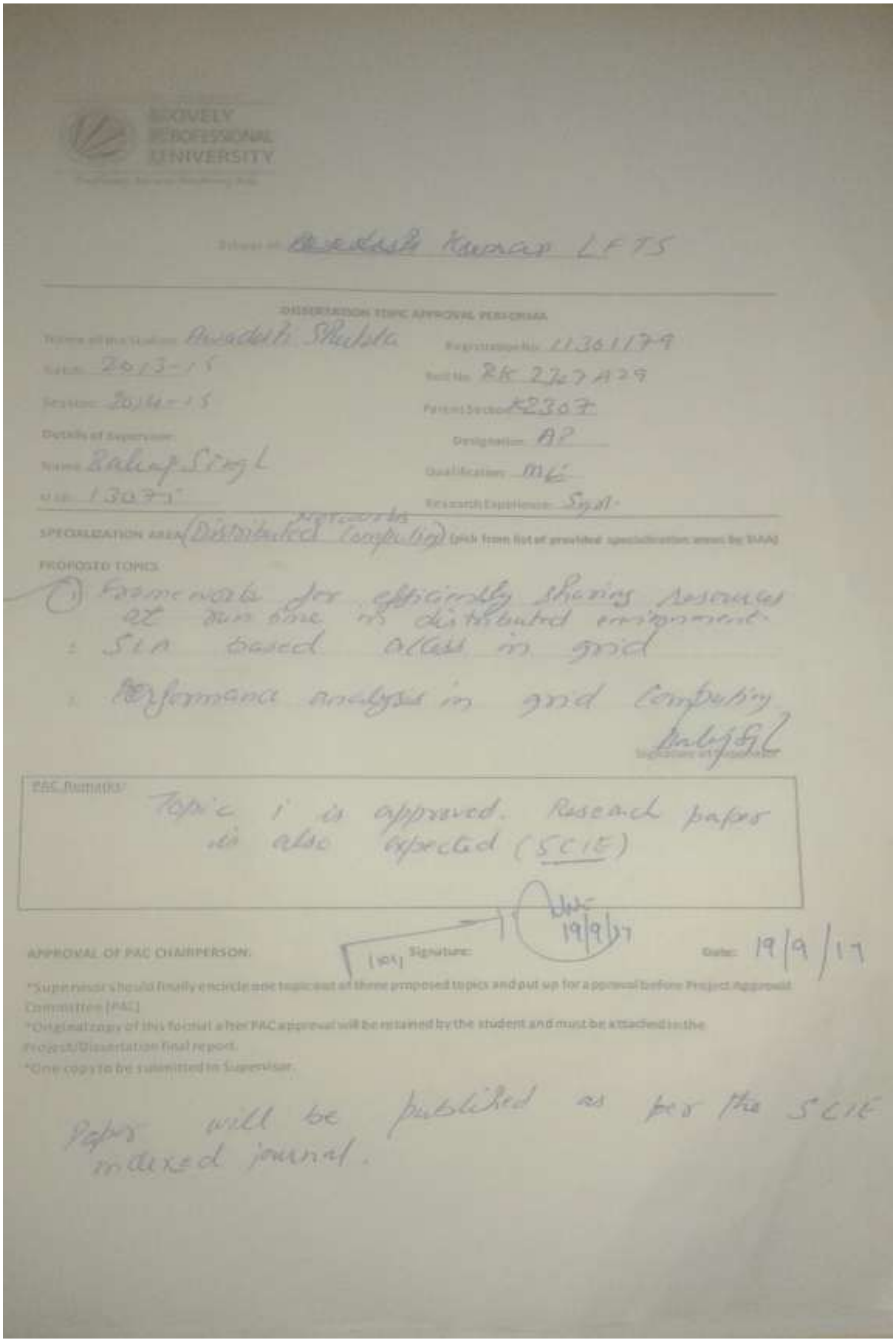
Award of the Degree of

Master of Technology in Computer Science and Engineering

Under the guidance of

(Balraj Singh ID :13075)

(May, 2015)



LOVELY PROFESSIONAL UNIVERSITY

School of: Awadesh Kumar LFTS

DISSERTATION TOPIC APPROVAL PERFORMA

Name of the Student: Awadesh Shukla	Registration No: 11361179
Batch: 2013-15	Roll No: RK 2307 A29
Session: 2014-15	Parent Section: K2307
Details of Supervisor:	Designation: AP
Name: Baljit Singh	Qualification: ME
U.ID: 13075	Research Experience: 5yr

SPECIALIZATION AREA: (Distributed ~~Computing~~ Networks) (pick from list of provided specialization areas by DAA)

PROPOSED TOPICS

1. Framework for efficiently sharing resources at run time in distributed environment.
2. SLA based access in grid
3. Performance analysis in grid Computing

[illegible]
Signature of Supervisor

PAC Remarks: Topic 1 is approved. Research paper is also expected (SCIE)

APPROVAL OF PAC CHAIRPERSON: [illegible] Signature: [illegible] 19/9/17 Date: 19/9/17

*Supervisor should finally encircle one topic out of three proposed topics and put up for approval before Project Approval Committee (PAC)

*Original copy of this format after PAC approval will be retained by the student and must be attached in the Project/Dissertation final report.

*One copy to be submitted to Supervisor.

Paper will be published as per the SCIE indexed journal.

ABSTRACT

The distributed systems had provided an enhanced approach towards the processing of the data. The information is processed at multiple locations to gain high performance and speedy results. This has also provided utilization of different resources. But the resources and processing powers seems to be limited in front of the size of the data. The need for the hour is to bring effectiveness in the exiting methodologies and techniques for bringing more utilization out of them and improve the overall performance of the work in demand. Hadoop has shown huge lead in processing the large amounts of data through distributed environment. The process of scheduling the jobs and effective utilization of resources will always be a challenge looking at the way data is growing. Our proposed work has given performance improvements in the existing scheduling processes and implemented better utilization of the resources. The limitations of the existing algorithms have been explored in our work. The existing system is unable to process the jobs efficient with multiple resource requirements. Our strategy has overcome this limitation by packing together the memory and processing requirement together by allocating it to a single job and addressed the issue of multiple resource requirements and effective resource utilization. Along with it overcomes the problem of fragmentation.

CERTIFICATE

This is to certify that Awadhesh Kumar Shukla has completed M.Tech dissertation titled **“Enhancement in the resource utilization and sharing in distributed systems”** under my guidance and supervision. To the best of my knowledge the present work is the result of his original investigation and study. No part of the dissertation has ever been submitted for any other degree or diploma.

The dissertation is fit for submission and the partial fulfillment of the conditions award of M.Tech Computer Science and Engineering.

Date: **Signature of Advisor**

Name: ______________________

UID: _______________________

ACKNOWLEDGEMENT

The dissertation proposal “Enhancement in the resource utilization and sharing in distributed systems” is proposed by me could not possible without the moral and academic support of my research guide Mr Balraj Singh. Special thanks to him for guiding my efforts to perfection.

Awadhesh Kumar Shukla

DECLARATION

I hereby declare that the dissertation entitled, **Enhancement in the resource utilization and sharing in distributed systems** submitted for the M.Tech Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date: Thursday, May 07, 2015 **Awadhesh Kumar Shukla**

Regn. No. 11301179

Table of Contents

List of Figures

CHAPTER 1
INTRODUCTION

1.1 Distributed Computing

A distributed system is a collection of independent computers that appears to its user a single coherent system. This definition can explain several important aspects. The first fact is that a distributed system is the collection of different types of component's for example computer, networking devices, storage, printers etc. The second aspect is that user which is using this distributed system think that they are dealing with single system. This means distributed system is the collection of different heterogeneous system from different geographical location and how these systems get collaborated is the most important concern when developing a distributed system. Distributed systems have some characteristics which are very important. The first characteristic is that the various independent components of distributed system are the loosely coupled and the network topology by which they are communicating with each other is hidden from the user. The second characteristic is user can access distributed system from any location such that the geographical distance and the location is not any constraint for accessing the distributed system. The third characteristic is scalability, it is easy to enhance and extend the distributed system. The most important characteristic is that the system is available even though some components stop working such that users are unaware of the component's failure, replacements and substituted by new components. All details are hidden from the user. Distributed system is lying as middleware layer between higher layer which consists of different type of application and lower layer which consists of heterogeneous type of operating systems to support independent computers and networks and providing coherent single view to the users as shown in the figure [1] .

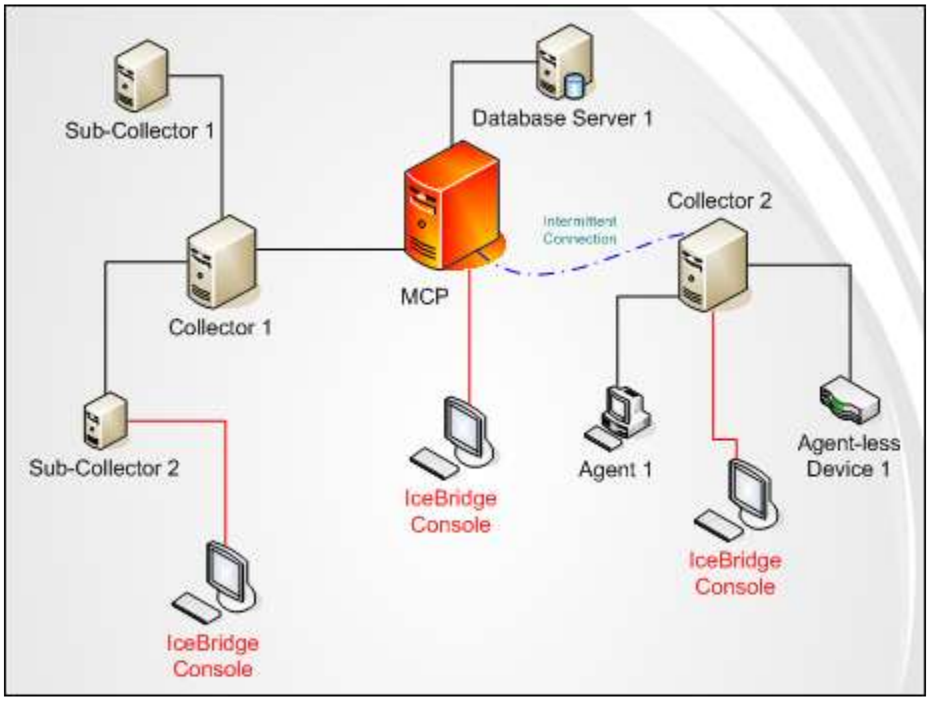


Fig 1: Distributed System

1.2 Distributed System Objectives

This section describes why we use distributed system and what we achieve from the distributed system. It is any worth for us using distributed system or not. It is very important to know that what we need. This section describes some important objectives of distributed systems.

1.2.1 Making Resource Accessible

The main objective of distributed system is to provide easy and efficient access of resources to all the users. Resources can be any things such as computes, processors, printer, files, storage, network etc. Resource sharing has many advantages such as economics, which share expensive resources such as supercomputers, high performance storage and other expensive devices and peripherals more effectively and efficiently.

1.2.2 Distribution Transparency

This is an important objective of the distributed system which conceals the users processes and resources are distributed to geographically dispersed independent computers for the processing. A transparent distributed system provides a single coherent view to its users and application. The distributed system having different types of transparencies, some of them are discussed below.

Access: This transparency conceals the difference in data representation and the way by which resources are accessed by the users. Distributed system is the collection of heterogeneous types of the system which having different types of architecture and different schemes for data representation which must be hidden from these users.

Location: This transparency conceals the location of the resources from the user such that user does not know where the resources are physically located. Logical names play very important role in providing this transparency.

Migration: This transparency conceals that the resources are physically moved to different location without affecting the usage of these resources.

Relocation: This transparency conceals that the resources are moved from the location while they are in accessed without affecting the user.

Replication: In this resources are physically replicated and have multiple copies while users are unaware of this and work as if they are working on a single copy only.

Concurrency: This transparency conceals that the resources are shared by multiple users and application simultaneously.

Failure: This transparency conceal all kinds of failure from the end users and application user doesn’t know when some of the components of the system stops working and when this component get replaced or removed from the error.

1.2.3 Openness

Openness is one of the important objectives of the distributed system. The open distributed system provides service according to the specific rules and standard rules which define syntax and semantics of the services offered by the distributed system. We can explain it by taking computer network example in which standard rules define the format of the message sent and received. The rules are formed by the interface definition languages (IDL) in distributed system which captures the syntax of those services.

1.2.4 Scalability

Scalability is also the key objective of the distributed systems. It has three dimensions which are enlisted below:

- The distributed system is scalable with respect to its size.
- The distributed system is geographically scalable.
- The distributed system is administratively scalable.

Size scalability deals with easily adding of more system in the distributed system. Location scalability deals with the distance of user and resources. Administrative scalability deals with the systems that are scalable from more than one administrative domain [1].

1.3 Types of Distributed Systems

Distributed computing system

- Cluster computing system
- Grid computing system

Distributed Information Systems

- Transaction Processing Systems
- Enterprise Application Integration

Distributed Pervasive Systems

- Home Systems
- Electronic Health Care Systems
- Sensor Networks

1.3.1 Distributed Computing System

This is the class of distributed computing which is used for massive computation analysis or we can say high performance computation having two subgroups. Cluster computing having homogenous type of systems which are locally connected by the high speed network. Grid computing consists of heterogeneous types of system which are geographically dispersed in location and collaborative perform tasks which require massive computation.

1.3.1.1 Cluster Computing System

A cluster is a collection of the homogenous systems which are connected with each other using high speed local area network, such as gigabit Ethernet, SCI, Myrinet and Infiniband. The system in cluster works collaboratively to perform a task which is not performed by single system or we can say a task which needs high computation using commodity hardware instead of the extra expenditure in high performance system. Clusters are used primarily for increasing availability, load-balancing of the computation. In cluster if some components are fail the service is not get stooped the task is transferred to other working system of the cluster. Cluster contains multiple computers so the works is shared by all the computer system in the cluster which provide better load balancing. Cluster also follow the transparency user and application can view cluster as a single system they cannot get aware about that any components are fail and how cluster preform their work all these hidden from user and application. The challenges faced in the cluster computing are given below:

a) Middleware: The middleware is used to provide single coherent system view to users and applications means it is used to hide that the resource are distributed to various independent system. Middleware is also taken care how collect the result from multiple cluster computer and collaborate them before delivering to the user and application.
b) Program: The application and user programs that run on the cluster must be coded in such a way that easily incorporates the distribution of the task to various systems and the communication between the distributed modules is also possible easily for better load balancing and better resource sharing.
c) Elasticity: The difference between the real time and actual response time when the workload or we can say number of requests get increases.
d) Scalability: when any user and application request for the additional or enhancement in the infrastructure or the system in general.

1.3.1.2 Grid Computing System

Grid computing is the collection of heterogeneous computer resources from geographically dispersed domains and executes the computational operation which demand huge amount of computational power to solve a task. Grid computing is similar to electric grid like electric grid computational grids also collects the resources from multiple domains and provides the computational resource according to the demand of application user. The grid computing is a category of distributed computing which uses unused, idle instruction cycle to solve a computational task which is not possible to solve by a single resource. Grid computing enables maximize use of computational power which is unused during the ideal and time. Grid computing is comes under the distributed computing the main aim of grid computing is provide on demand service to the application user in transparent manner when user submits the job grid middleware distributed the job to various resources and the resource perform computational operation and result get back to the grid middleware and this middleware gives it back to the application user transparently means user cannot able to understand the complexity of resource provision to the given task. This figure shows a grid which contains heterogeneous type of resource from multiple domains which are geographical far from each other and communicating through high speed network connection. The grid computing also faced lot of challenges when deployed in the real world we must take care of these challenges for better grid deployment and maximize resource utilization of the grid resources. The given figure 2 shows heterogeneous grid. The various challenges faced by the grid computing are explained below the figure 2.

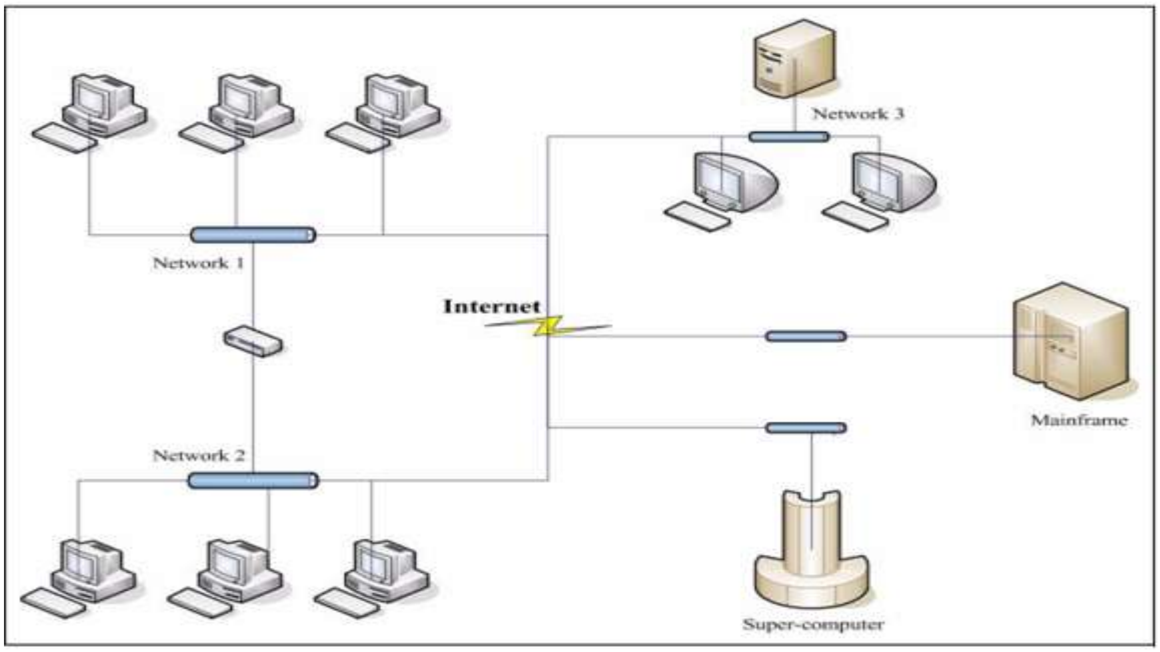


Fig 2: A Heterogeneous Grid

Dynamicity: In Grid, the resources from various geographically dispersed administrative domains enters and leaves the grid at random, any time that leads to extra burden on the grid for the management of the resources and also keeps track which resources are in and which left the grid.

Administration: Grid having the resources from multiple administrative domains and forms a pool of the resources and supplies the resources on demand to the users and applications. This increases heavy burden of system administration and also map the local polices of various administrative domains to global polices for grid.

Development: Grid resources are distributed across multiple geographically dispersed administrative domain so the main problem are coding the application which run on grid with the care of distribution of the module to various processing elements for processing and when processing finished the reassembly of the result from all the processing element and give back to the requested user.

Heterogeneity: The resources in the grid computing are heterogeneous in the nature so we need to create an efficient framework for the data intensive programming and the scheduling which take care of heterogeneity of the resources for better utilization of the resources of the grid.

Programming: The processing elements are distributed in different geographical location which leads to complex programming for various grid applications.

Types of Grid

Grid computing is used in the various fields of the computing and exploits the use of unused instruction cycle on the basis of use in the various field grids can be divided into different types. Grid computing is used in the field of computation, storage, networks it is used to provide massive computational power in geographically dispersed environment. Grid computing enables us to solve a computational task which is not solved by single node. The following types of grid are used:-

Computational grid: This type of grid mainly focuses on computing power or we can say computational power and share this computational power among the multiple users according to their demand. We can say computational grid provide resource on demand just like electric grid. Computational grid is used to provide massive computational power for preforming high computational task. Computational grid uses the unused instruction cycle to perform high computational task and provide better processing speed and solve the task that is not executed by single system.

Data grid: This is also similar to computational grid but we know that the aim of computational grid is to provide computational power that share between multiple users, like the computational grid data grid is used to provide storage for multiple grid user. Data grid is used to share data storage to multiple grid users for the requirement of massive amount of data storage

Network grid: We all know grid is the collection of resource from different geographical location due to geographically disperse nature it require better communication. Network grid is used to provide better communication services in this type of grid each node works as a router between the two communicating points and provide other services which is needed to increase the communication speed on the demand of the user [1, 2].

1.4 Hadoop

Hadoop is a distributed computing architecture based upon the open source implementation of Google's MapReduce which supports processing of huge amount of data sets across multiple distributed systems. The present technological era does not depend only on a standalone computation, rather demands huge data computation through distributed computing along with performance. Almost all the technological giants like Yahoo, Google and Facebook use data intensive computation for their business. Handling high amounts of work load for computation is somehow a challenging task since it is bounded with the performance constraints and availability of the resources. Hadoop has proved to be an effective platform for this purpose. Hadoop is designed such that it can accommodate scaling up from single standalone systems to excessively large number of systems where each machine provides both computation and storage together [3, 4].

1.4.1 Hadoop Ecosystem

- Avro: Avro is serialization system for efficient, cross language RPC and persistent data storage.
- Pig
- Hive
- HBase
- ZooKeeper
- Sqoop
- Oozie

1.4.1 Major Components of Hadoop

- MapReduce
- HDFS

1.4.2 MapReduce

Hadoop MapReduce is a programming model for data processing. Hadoop MapReduce is inspired by the paper published by the google on the MapReduce technology. MapReduce is a programming model and associated implementation for processing and generating large data sets. Users specify a map function that process a key/value pair to generate a set of intermediate key/value pair and a reduce function that merges all intermediate values associated with the same intermediate key.

1.4.2.1 Data Flow in MapReduce

Input files: The input files are the files which are given to MapReduce for processing. These files are resided in HDFS (Hadoop Distributed File System). These files are very large in the sizes because Hadoop is used for massive computation analysis.

InputFormat: The format of reading input files are defined by the InputFormat. InputFormat provides us the following given functionality:-

a) Selects the files or other objects that should be used for input

b) It breaks the files into tasks by defining the InputSplits.

c) It offers a unit for RecordReader objects that read the file

Several Input formats are provided with Hadoop in which FileInputFormat and InputFormats operates on files. When Hadoop start running a job, the FileInput Format is provided to the files which needs to be read. Then FileInput Format reads the entire file from given location and divides the files into one or more than one InputSplits. The user can define which InputFormat wish to apply to on input files by calling setInputFormat() function JobConf object that defines the job.

InputSplits: InputSplit defines a unit of work that encompasses a single map task in a MapReduce program. In the Hadoop the default input split size are 64MB and 128MB

RecordReader: The RecordReader is used to load the data from the source and converts it into key/values pair appropriate for reading by the Mapper.

Mapper: The Map function takes key value pair as input and generates intermediate key value pair. The output generated by map function become input for the reduce function.

Partition & Shuffle: Shuffling and Partitioning are very important part of MapReduce. Shuffling is used to moves the output generated by the map function to the reduce function and partitioning take care of allocating output generated by map function to specified reducer for the purpose of database shard.

Sort: The sorting take care of sort the intermediate keys generated by the map function before passing these to the reducer for the processing.

Reduce: A Reducer is used to perform summary operation of the result generated by the map function. The reducer function performs the summary operation on the basis of same key generated by the map function.

Output file: The output files are the result produced by the reducer function and stored on HDFS (Hadoop Distributed File System) for future use [5].

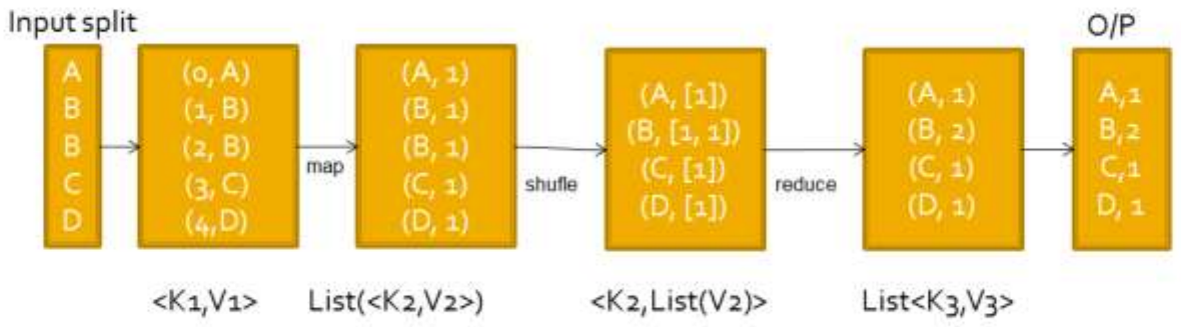


Fig 3: Key/Value pair generation

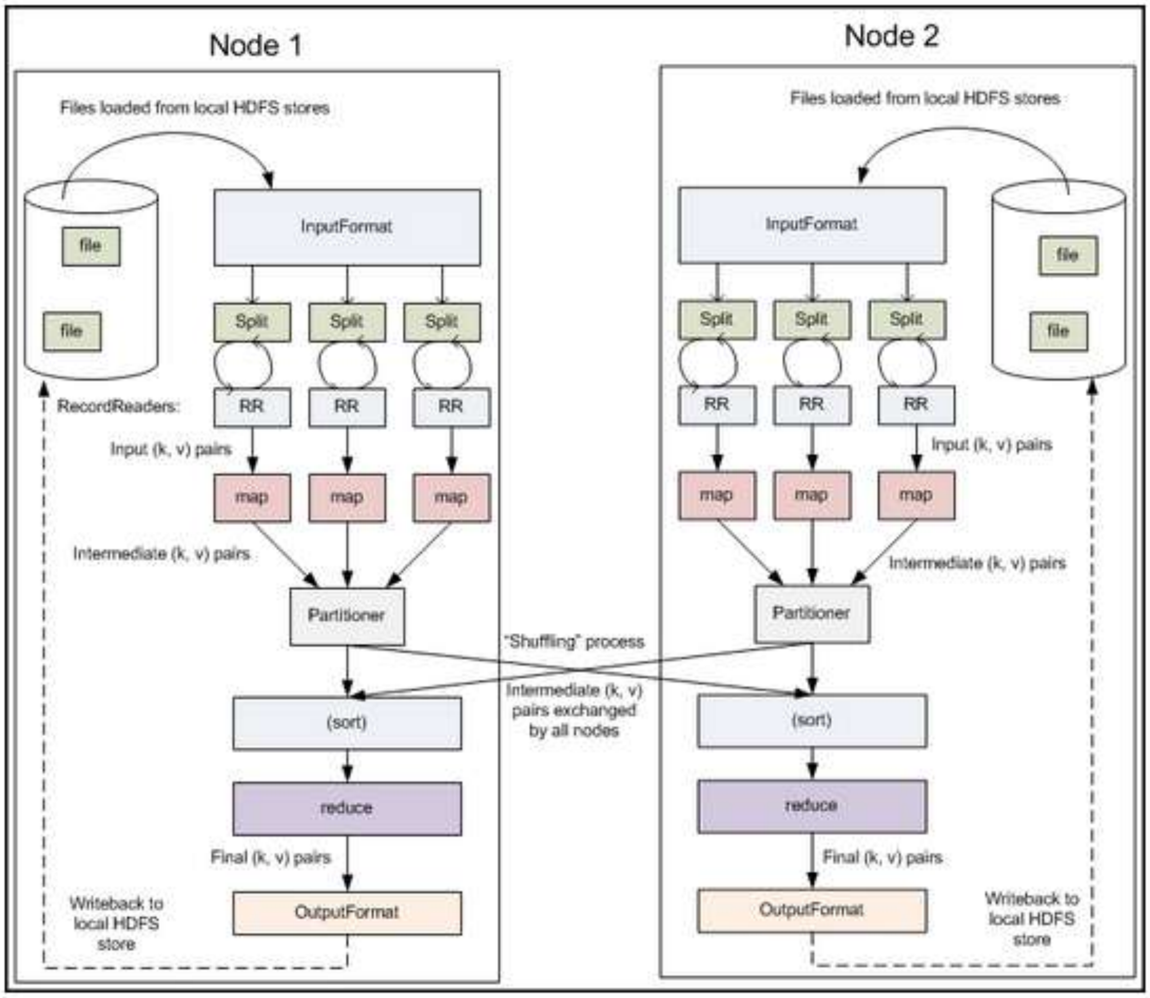


Fig 4: MapReduce Data Flow

1.4.2.2 MapReduce Nodes

- JobTracker
- TaskTracker

JobTracker: A JobTracker node manages MapReduce jobs. There is only one of these on the cluster. It receives jobs submitted by clients. It schedules the map task and reduce tasks on the appropriate.

TaskTracker: TaskTracker in a rack-aware manner and monitors for any falling tasks that need to be rescheduled on a different TaskTracker. To achieve the parallelism for your map and reduce tasks, there are many TaskTracker in a Hadoop cluster. Each TaskTracker spawns java virtual machines to run your map or reduce task.

1.4.3 HDFS (Hadoop Distributed File System)

HDFS runs on top of the existing file system on the each node in Hadoop cluster. It is designed for a very specific data access pattern. Hadoop works best with very large files. The larger the file the less time Hadoop spends seeking for the next data location on the disk and the more time Hadoop run at the limit of the bandwidth of your disks. Seeks are generally expensive operation that are useful when you only need to analyze a small subset dataset. Since Hadoop is designed to run over your entire dataset, it is best to minimize seeks by using large files. Hadoop is designed for streaming or sequential data access rather than the random data access. Sequential data access means fewer seeks, since Hadoop only seeks to the beginning of each block and begins reading sequentially from there.

1.4.3.1 HDFS-Blocks and HDFS-Replication

HDFS-Blocks: Hadoop uses blocks to store a file or parts of the file. Hadoop having default to 64 megabytes each and most system runs with block sizes of 128 megabytes and larger. A Hadoop block is a file on the underlying file system. The file system stores files as blocks, one Hadoop block may consist of many blocks in the underlying file system as shown in the figure:

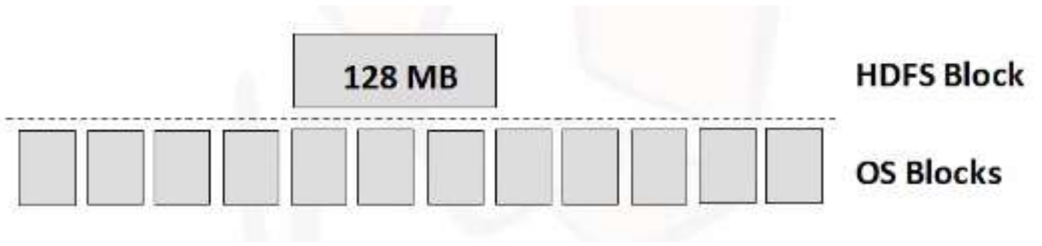


Fig 5: File system storing files as blocks

Blocks have several advantages first they are fixed in size. This makes it easy to calculate how many blocks are fit on a disk. Second, being made up of blocks that can be spread over multiple nodes, a file can be larger than any single disk in the cluster. HDFS blocks also do not waste space. If a file is not even multiple of the block size, the block containing the remainder does not occupy the space of an entire block. Finally blocks fit well with replication, which allows HDFS to be fault tolerant and available on commodity hardware.

HDFS-Replication: Hadoop blocks are replicated to multiple nodes this allows for node failure without data loss. We can set the number of replication by changing Hadoop configuration file.

1.4.3.2 HDFS-Nodes

- NameNode
- DataNode

NameNode: There is only one NameNode in the cluster. While the data that makes up a file is stored in the blocks at the data nodes, the metadata for a file is stored at NameNode. The NameNode is also responsible for the file system namespace to compensate for the fact that there is only NameNode one should configure the NameNode to write a copy of its state information to multiple locations, such as a local disk and NFS mount. If there is one node in the cluster to spend money on the best enterprise hardware for maximum reliability it is the NameNode. The NameNode should also have as much RAM as possible because it keeps the entire file system metadata in memory.

DataNode: HDFS cluster has many DataNode. They store the blocks of data and when a client requests a file, it finds out from the NameNode which DataNodes store the blocks that make up that file and the client directly reads the blocks from the individual DataNodes. Each DataNodes also reports to the NameNode periodically with the list of blocks it stores. DataNodes do not require expensive enterprise hardware or replication at the hardware layer. The DataNodes are designed to run on commodity hardware and replication is provided at the software layers.

1.4.3.3 Writing HDFS File

The user makes a ‘create file’ request to the NameNode. The NameNode checks that file does not already exist and also check the user having permission to perform this action if succeed. NameNode find the DataNode for writing the file. If client is running on the DataNode it will try to place their otherwise it chooses a random location by default data is replicated to two other places in the cluster. A pipeline is built between the three DataNodes that make up the pipeline. The second DataNode is randomly chosen node on a rack other than that of the first replica of the blocks. The final replica is placed on a random node within the same rack as the second replica. The data is piped from the second DataNode to the third to ensure the write was successful before continuing acknowledgement packets are sent back from the third DataNode to the second, from second DataNode to the first and from the first DataNode to the client. This process occurs for each of the blocks that make up the file. When the client is done writing to the DataNode Pipeline and has received acknowledgements, it tells the NameNode that it is complete. The NameNode will check that the blocks are at least minimally replicated before responding. This figure shows all the step involved in the writing a file to HDFS:-

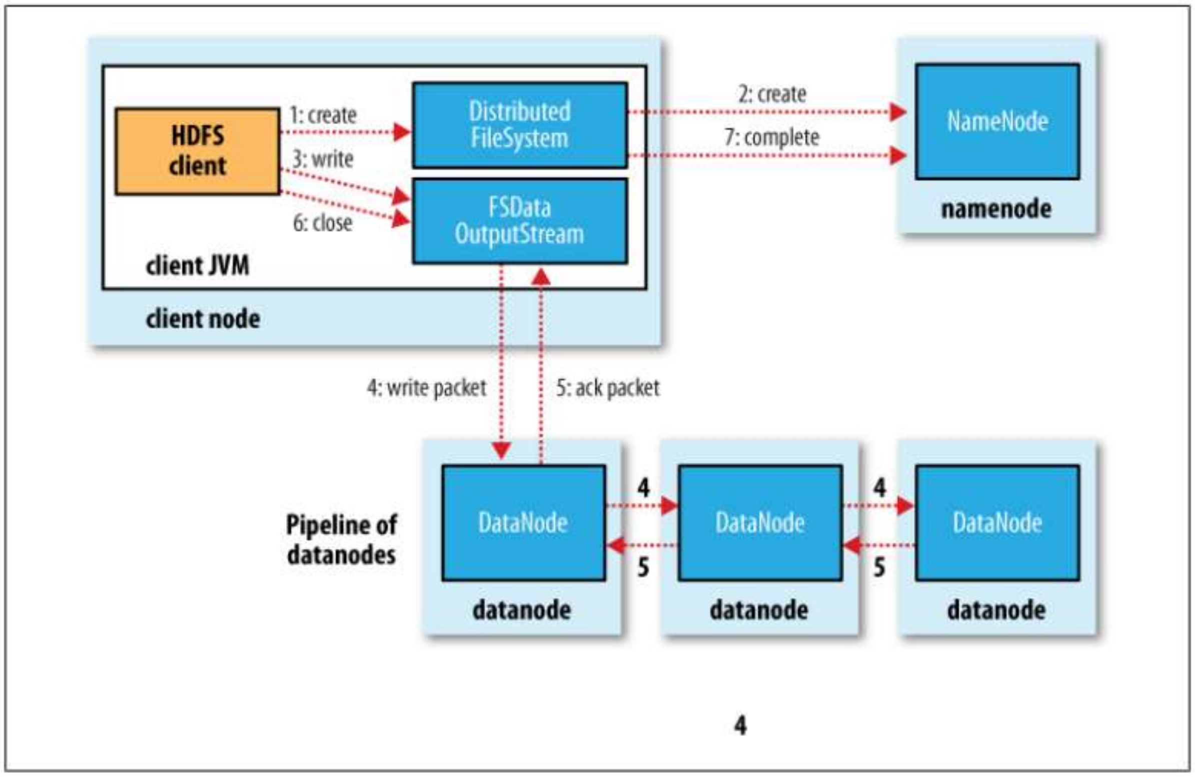


Fig 6: Writing a file to HDFS

1.4.4 Broad classification of the scheduling algorithms in Hadoop

1.4.4.1 FIFO scheduling

The Hadoop computing system uses the first in first out algorithm. It works on the mechanism that the job which entered the system first will be executed first and usually supports same category of job requests [7]. The major challenge faced by this algorithm is that it results in lower utilizations and poor performance since a large job may be occupying the resources and giving a huge waiting time to a job which is waiting in the queue for execution. The smaller job may require very lesser amount of resources but have to wait the finishing of the task which eventually affects the overall performance of the system. FIFO is the basic and default scheduler of the Hadoop distributed system [8].

1.4.4.2 Capacity scheduler

Capacity scheduler is an enhancement on the FIFO which overcomes the issue of resource blocking by large job and extended waiting time. It sets up the Upper and lower limits for resource sharing such that none of the job dominates the resource share and everyone gets share as per the given limits only. Once the jobs are shared among the jobs as per this algorithm, than further in its subsequent processing, they work on the basic mechanism of FIFO only. On the downside, this algorithm also suffers with performance issues since the algorithm needs to learn the information about the system and create a queue which results in huge number of bottlenecks and lower down the performance.

1.4.4.3 Fair scheduling

This scheduling algorithm gives an equal share of available resource to all the jobs. Available resources are shared among all the jobs on equal share. The benefits are that the smaller jobs get enough shares to get executed well in time [6, 8]. The downside is that it only works on the sharing of memory only. Further possibilities are unexplored yet.

CHAPTER 2
REVIEW of LITREATURE

S. Agarwal et al 2012 “discussed about Re-optimizing data parallel computing”. In this paper author proposed a RoPE (Re- optimizing Data-Parallel Computing) which is used for the re-optimization of the data-parallel jobs. RoPE uses piggybacking for the collection of the certain data properties and code on job execution and then it adapts the plan of the execution by serving these properties to a query optimizer. Accurate estimation of the property of code and data is not an easy task in distributed environment and forecasting attributes by the collection of statistics of the stored raw data is not appropriate because the commonness of user operation. The knowledge of these attributes provides us a huge space of enhancements. The absolute numbers of jobs indicates attributes are estimated dynamically. The proposed solution RoPE gathers statistics from multiple location and uses innovative way to combine the entire attribute. The elasticity permitted for the user to state random code leads to ample snugger connection between data and computation in data parallel clusters. Future scope of proposed work develop more advance technique that selects plans We defer to future work some advanced techniques that choose plans having defined level of the validity range which is specified over given statistics and perform substituting to these plans during runtime depending on the detected statistics [9].

Mohammad Al-Fares et al 2008 “discussed about the A Scalable, Commodity Data Center Network Architecture”. In the proposed research work author discussed about data centers which consist of thousands of the nodes which having substantial cumulative bandwidth requirement. Network architecture usually consist of the a tree structure of routing and switching components which having more expensive components on moving up in the hierarchy and deployment of highest end IP based routers and switches, which causes to topologies may support only half of the aggregate bandwidth available at the edge of the network and still sustaining tremendous cost. Due to the non-uniform bandwidth in between the data centers nodes application development become a complicated and also reduce the performance of the system. Author proposed a solution of this problem by scalable commodity data center network architecture which shows how a data center having large number of nodes uses influences largely commodity Ethernet switches to provision of full collective bandwidth of cluster. Authors says that suitably architected and organized commodity switches provides better performance at low cost than the current existing high end solutions and this approach not require any modification to the end user network interface, running application and operating system platform. In this proposed paper author uses flat tree topology and provides a technique to perform scalable routing with the backward compatible with the Ethernet, IP and TCP and the proposed idea of author reduce the cost of delivering the bandwidth as compare to the current techniques [10].

Mosharaf Chowdhury et al 2011 "discussed about the Managing Data Transfers in Computer Clusters with Orchestra". Author discussed about the cluster application such as MapReduce and Dryad which transfer huge amount of data during their computation. The huge data transfer between cluster applications can have substantial effect on the performance of executing completion times. Regardless of this effect having little work for the management of the data flow to get improved performance and the management of network for per flow management become area of interest for researcher. In proposed work author addressed the limitations by proposing set of algorithms and a global architecture. First it reduces the time taken in transfer of data during communication, broadcast and shuffle phases. Second author uses scheduling policies at the level of the transfer both results in the form of reduction in the broadcast time. An author takes two common pattern of transfer are shuffle and broadcast for this he proposed an scheduling algorithm name WSS (Weighted Shuffle Scheduling) for the shuffling which improves the overall performance and high priority transfer are get improved by the inter transfer scheduling. The proposed algorithm is implemented at application layer and it does not need any hardware changes for running in the data center and clouds due to this no extra expenditure of hardware is not required [11].

Mosharaf Chowdhury et al 2013 "discussed about the Leveraging Endpoint Flexibility in Data-Intensive Clusters". In this work, the authors discussed that endpoint is not restrain for the network transmission but while such kind of extensive distance data transfer gives a vast quantity of the bytes in the network. By defining the end points we avoid the congestion in the links and time of completion of these transfers without the improvement in the flexibility improved. In the proposed work author emphasized on leveraging the flexibility in replica placement during writes to cluster file system which reduces the cross rack traffic to fifty percent in massive data intensive clusters. The cluster file system replicas are placed at cluster on the realistic machines the long distance in different fault field and guarantee the balanced used of the cluster storage. The author planned and estimated that the proposed system find out imbalance in the network by the measuring after a particular time period and achieved the flexibility in the system to traverse the links which causes congestion in the network. The proposed research reduces the average time taken in the writing blocks and also reduces the time average time of completion of the data exhaustive jobs [12].

Ali Ghodsi et al 2011 "discussed about the Dominant Resource Fairness: Fair Allocation of Multiple Resource Types". Author says that we have suffered from the fair resource allocation problem when we have systems which have different types of the resources and the users' demands are different for each of the system resources. Addressing this problem of resource distribution author proposed DRF (Dominant Resource Fairness) technique which is the simplification of the max-min fairness to different resource types. He shows how DRF fulfills the very much wanted properties. DRF provide inducements to the users to share the resources are equally distributed between the users then no user is better off. DRF uses strategy to proof the user requirement so the user cannot increase requirement by providing false requirements. DRF is greed free as no user need to trade her provision with that of alternative uses. DRF offers the Pareto efficient allocation of the resource to the user's means that it is not possible to allocation more resources to one user without decreasing resources to another user. Author says that he implemented his DRF allocation of resources at Mesons cluster resource manager and it provides us better throughput and the fairness then the current slot based schedulers. DRF allows the scheduler to take care of heterogeneous demand of multiple cluster applications and provide both more utilization of the resources and provide fair allocation of the resources. There is lot of future scope for research. First problem is without compromising with fair allocation of the resource reducing the resource fragmentation. Second problem is when tasks have placements restriction for the resources then providing fairness. Third area is the exploring the use of DRF in the operating system [13].

Albert Greenberg et al 2009 “discussed about the VL2: A Scalable and Flexible Data Center Network”. Authors discussed about the data centers provide dynamic resource allocation across large pools is provide cost effective services. The data centers must permit any server to allot any service. Achieve these goals author proposed a solution a scalable and flexible data center network VL2, this provide a network architecture that scales to support enormous data centers with unchanging great capacity between servers, performance separation services, and Ethernet layer semantics. The proposed network architecture uses flat addressing for the placement of service instances placed anywhere in the network. The proposed solution provides load balancing to spread the traffic uniformly across the network and also provide address resolution which scale the pools without increasing the management complexity of the network. The VL2 is plan is derived from the measurement of the traffic and the faulty data from the cloud service providers. VL2’s enactment influences confirmed network technologies, previously offered at low cost in high speed hardware implementations, to develop easily scalable and reliable architecture of the network. VL2 provides huge benefit to the programmer of the cloud services and also provide ease to the operator of the data centers. VL2 provides a facility that any server is allocated any service at the networks sustains isolation between unchanging high bandwidth and performance. Due to the simple design of the VL2 it is easily adapted by the current networking technologies it does not require any kind of changes in the controlling of switch and data capabilities it is efficient and achieved extraordinary tcp fairness [14].

Sumit Gulwani et al 2009 “discussed about the Precise and Efficient Static Estimation of Program Computational Complexity”. In the proposed study author discussed about an technique that is based on inter procedural for calculating the symbolic bounds on the number of statement executed by the procedure in the expressions of the scalar response and by the user demarcated measurable function of input data. The calculations of computational bonds are typically disjunctive, non-linear, and include heaps numerical attributes. The author addresses the challenge of calculating these bonds. Author provides numerous counter arrangements based on the proof methodology in which counter value is incremented and decremented at multiple location of the program which permits the calculation of the bound on counter variable independently. The boundaries on these counters are then calm together to create total boundaries that are non-linear and disjunctive. Author also provides an algorithm for the dynamic proofing of the approach. The proposed approach produce the boundaries of the difficulty bonds that are not accurate for computational difficulty it also include the persistent factors. Author also defined notation for the user defined methods. The proposed study says collectively these techniques permits the definition of exact boundaries of the computation. In the proposed research work author with the help of the multiple counters author removed the problem of the creating disjunctive invariants and also remove the generation of non-linear invariants defining the dependencies in counters. Author also removed the precision by providing minimum number of dependency of minimal number of counters [15].

Chuanxiong Guo et al 2009 "A High Performance, Server-centric Network Architecture for Modular Data Centers". In the proposed research work, authors discussed about a network architecture which is precisely planned for the modular data centers and for shipping containers name BCube. The proposed BCube is centered to server in network configuration where servers consist of many network ports for the connection of commodity of the shelf small switches. The serves act as both hosts and nodes for one and another. The proposed BCube architecture various applications which are bandwidth intensive by hurtling up one to one traffic pattern, one to many traffic pattern and one to all traffic patterns and it also deliver very high capacity to the network. BCube demonstrations agile performance dilapidation as switch and servers failure frequency upsurges. Author say the implementation of the BCube is very competently deployed in the software and hardware. The agile performance dilapidation and meet the unusual necessities are with the help of the BSR routing protocol and the future scope of the study how this approach is scale multiple containers [16].

Mark E. Crovella et al 2009 "discussed about the Connection Scheduling in Web Servers". In the proposed study authors discussed about the behavior of the web services under high workload. How web servers serves huge number of the connection simultaneously. The order in which these simultaneous connections served is the responsibility of the operating system. In this proposed work author analysis the behavior of the service by using nontraditional service ordering and in the situation of serving static files then the benefits and costs for giving favored service to short connection. Author analyzed the behavior of the commonly used server in the respect of the size of connection and illustrations that it not gives the favored service to short connections and then examine the improvements in the probable concert of strategy which does not favor short connection. Author shows that mean reply time can be upgraded by the aspects of four or five under shortest connection first as compared to a size liberated strategy and they also finds the cost of shortest connection based scheduling in the terms of unfairness. An author shows how long connection pay very small penalty in the condition of the shortest connection first. Future work in the proposed study controlling the scheduling of the kernel mode operations and due to these reason exact improvements required by the scheduling polices are not computed [17].

Michael Isard et al 2007 "discussed about the Distributed Data-Parallel Programs from Sequential Building Blocks". In the proposed research work author proposed Dryad which is used in distributed environment for the accomplishment of the abrasive grain data analogous applications. It combines the vertices of the computation with the network communication medium for a data flow graph. The proposed Dryad uses this graph for running application by verifying the vertices produced by the graph on the available set of the computers which are communicating each other through TCP, shared memory and files. The vertices detection is very simple for the programmer because these are written in the sequential languages of the computer programming without any intervention of the thread mechanism of locking and creation. The issue of concurrency gets up from the Dryad scheduling vertices to run at the same time on multiple processing units on same computer or multiple processing units on different computers. Size and the placement of data is discovered by the application dynamically at run time to modify the values of the graph for better utilization of the resources. The proposed study is applicable to all the single system which having multicore processing unit, cluster which contain few systems and the distributed data centers having thousands of the systems and it take care of problem faced in the creation of concurrent application for distributed computing, data transportation and better scheduling of the resources for better resource utilization [18].

Michael Isard et al 2009 "discussed about the Fair Scheduling for Distributed Computing Clusters". The proposed research study addresses the problem of the scheduling parallel jobs on clusters where the data required by application is stored on the different computational system. The strategy of putting computation close to data is followed of many of current technologies such as MapReduce, Hadoop and numbers of distributed grid computing environments. The author says that the scheduling with vicinity and equality restrictions has not previously been broadly considered underneath resource sharing. Author provide a framework for the scheduling the parallel jobs for distributed computing with the fine grain resource distribution strategy. The problem faced is charted on a graph, on which graph the demand of the locality, fairness and starvation are calculated with the help of the weight on the edges [19].

Lei Lu et al 2013 "discussed about the "Predictive VM Consolidation on Multiple Resources: Beyond Load Balancing". In the proposed research work author addresses the issue of the fair distribution of the load on numerous servers in the virtualized data centers and focused on the applications which are multi-tiered and different demands of the resource in each tier and the key issue taken is consideration is the best match of the resource to the application that leads to reduced performance interloping. To take consideration of this problem author addresses the following two steps. First steps take care of fair allocation of the resources for the better load balancing by allocating dissimilar cybernetic machines through the numerous servers. This approach is framed as multi-dimensional vector scheduling which uses new PATS (polynomial-time approximation schemes). Second approach is used for selecting the optimal solution using analytical model of queuing over the proposed minimum and maximum solution. Experimental result of the proposed research study shows that the given mechanism is tough for predicting the ideal amalgamation strategy. The future work of the proposed research work is enhancing the forecast system of queuing approach by taking care of burstiness [20].

Alexander Rasmussen et al 2012 "discussed about the Themis: An I/O-Efficient MapReduce". In the proposed paper author discussed analysis of the big data utilizes programming mode of MapReduce for the analysis of huge amount of data collectively. Many jobs of the MapReduce are input/output bounded so reducing the frequency of input/output is serious to enhancing performance. In the proposed work author present Themis, it is the implementation of the MapReduce which provide read and write of the data twice on the disk, which is the smallest quantity probable for data sets that cannot appropriate in memory. Themis relinquishes task-level mistake acceptance, trusting as an alternative of job-level mistake acceptance. To provide reduce number of the input/output operation the proposed work take different design decision from the preceding MapReduce execution. The proposed work has implementation of an extensive range of MapReduce jobs at approximately the consecutive rapidity of the essential storage layer, and is on equivalence with TritonSort's record organization performance [21].

Alan Shieh et al 2011 "discussed about the Sharing the Data Center Network". Authors say data centers suffer from the effective sharing of the network resources even they are multiplexed for the non-cooperating applications and Depend on TCP's bottleneck control. Author addresses this problem and provides a mechanism name Seawall which is technique of the allocating bandwidth of the network. This technique divides the network bandwidth according to administrator specification and polices. The proposed technique computes the requirement and allocation is based on traffic tunneling by the overcrowding control in the edge to edge and point to multipoint tunnels. The result of allocation is persist stable irrespective of the flows, set of protocols and application destination traffic and it also support the run time changes or we can say dynamic changes with scrambling the count of application in the data centers. An author says that proposed achieved high performance seclusion at the cost of small overhead [22].

Todd Tannenbaum et al 2001 "discussed about the Condor – A Distributed Job Scheduler". In this proposed study, authors discussed about the Condor which is used for scheduling the jobs in the distributed environment it is specialized for the workload supervision for the massive computation tasks, it provides mechanism for job queueing, policy for scheduling, management of the resources, monitoring of the resources. The end user gives their task to the condor and it pushes into the queue and then takes care of when the job gets executed and all the resource requirement of the job along with policies. And it also monitors the progress of the jobs provided by the user and provides acknowledgement to the user on completion of the job. The key features of the condor are Distributed submission, job priorities, user priorities, job dependence, multiple job model support, checkpoints, migration, parodic checkpoint, job suspend, job resume, remote system calls, machine pools can work together, authentication, authorization, heterogeneous platform, grid computing [23].

Ashish Thusoo et al 2009 "discussed about the A Warehousing Solution Over a Map-Reduce Framework". In the proposed research work author discussed about the rapidly increasing size of the data collected from the various sources and perform analysis for better decision making, the rapid growth of the data causes expensive storage solution. In this paper author proposed Hive which is also an open source library and provide data warehousing service and it is built on top of Hadoop and this is also support the query languages examples are (SQL-like declarative language-HiveQL). Hive query language provide the MapReduce are plugged into queries. The author say Hive also contains a system catalog, statistics, Hive-Metastore, and containing schemas, which are important for the exploring data and optimization of the queries. Hive need to improvement towards working on all SQL syntax, building Hive on the basis of the optimization and adaptive techniques of optimization for better efficient result [24].

Rajni Aron and Inderveer Chana et al 2012 "discussed about formal QoS policy based resource provisioning framework in". In this paper he discussed a framework for resource provisioning with QoS(quality of service) parameter. This proposed frameworks offer on demand resource delivery and efficient scheduling of the resources. This framework negotiate with the user by SLA (service level agreement) to analyze customer requirement and define a procedure how to accomplishment of user requested service that is satisfactory to their resource user. The QoS parameter that should be defined in the SLA (cost, time, money, reliability). Cost is calculated on the basis of resources consumed per unit time by the user. Time is calculated on the basis of historical data that was collected from the analysis of the job executed previously and also perform an analysis before executing the task. Security is considered by identifying reliance on the nodes and the reliance of node is recognized by the foregoing transaction and present environmental individualities. Reliability of node checked before provision of resource, fault tolerance, storage is some of the parameters used to measure the reliability. In this paper describes two matrices specifically submission time and cost metric to estimating the performance of quality of service based resource provisioning. The main proposed idea if this paper how to reduces the complexity from resource provisioning on demand for the task execution and efficient scheduling of resources as per availability of the scare resources. Through the usage of a persistent law, the number of migrations and trouble of laws is reduced. In the proposed work researcher demonstrates the effectiveness and worth of rules to provide resource to the user on demand and also maintain all QoS (Quality of service) parameter desired by the user [25].

Naidila Sadashiv and S. M Dilip Kumar et al 2011 "discussed a detailed comparison of Cluster, Grid, Cloud computing". In this paper authors discussed the all possible difference between cluster, cloud and grid. Cluster is the collection of homogenous type (means same type) of computational nodes which are connected using high speed network. These clusters work together on the task which require massive computation that not possible on single node. Clusters are mainly used for load balancing, high availability and for compute purpose. Grid computing combines various heterogeneous resources from geographically multiple managerial domains to solve a massive computational task and after solving the task return all the resources to the provider. Grid computing uses a middleware to divide the job to various computers which are heterogeneous and geographically dispersed and combine the result from these systems and provide to user in transparent fashion. Cloud computing is one of the emerging field of computer science. Cloud computing provide services on (application delivery and hardware resources) on the demand of the user by using high speed network connection. Cloud is a kind of parallel and virtualized computers that are enthusiastically provisioned on demand services to the user if user wants measured service [26].

"**Thomas Lehman et al 2006 DRAGON: A Framework for Service Provisioning in Heterogeneous Grid Networks**". In the proposed framework, a type of network organization is set up that permit dynamically provisioning of network resources to derive and set up a path to respond the request from the user application. This framework provides progressive e-science applications to dynamically get enthusiastic for determined resources of the networks to provide several computational facilities to get desired service to the user the given computational services are desired (CPU clusters, storage, picturing facilities, remote sensors and many other instruments in geographically dispersed environment and the way to specify topology which is best suited to the application which is demanding) [27].

Rajkumar Buyya et al 2009 “Performance analysis of allocation policies for interGrid resource provisioning”. In this paper authors discussed about the performance study of distribution rules for inter-Grid resources provisioning. This paper discussed the performance evaluation of polices which are used in resource provisioning across the Grids and also discussed how grid redirect request to the other grid during the peak workload using cost-aware load sharing mechanism. This mechanism relies on the information available by different scheduling policies at provider sites. The provider policies enable the information about the resources management and this information is used for load sharing among the grids [28].

“Michael A. et al 2009 Dynamic Provisioning of Virtual Organization Clusters”. In this paper , the authors discussed about the virtual cluster scheduler for efficient scheduling in the virtual organization which is the collection of various virtual clusters. This proposed work shows how overall throughput is conserved with the dynamic resource provisioning without increasing overhead of scheduling delay. The proposed work shows the size of virtual cluster of virtual organization increases on the demand basis and we can also define set of rules that defines the maximum or minimum simultaneous request running at the same time for a VO (virtual organization). This proposed framework provide that power of restricting the resource usage by physical grid mean grid is able to define how much computational power he wand to give individual virtual organization. Starting and stopping of virtual machine in the response of grid workloads is need attention of researchers [29].

A Filali et al 2008 “proposed adaptive resources provisioning for grid applications and services”. In the proposed work, authors presented a resource providing scheme which increases the efficiency of the resources usage and also delivers the demanded level of QoS (Quality of service), it reduces the chance of request deny and increase the profits of service sponsor. The proposed work also exploits the consumption of network and reduces the waste of network resources. In the proposed work, it uses an optimization model BIP (Binary integer programming) and after this a heuristic based technique is user to steadfastness by taking measure of response time and the request based movement. It gives us better result with low cost [30, 31].

"Amos Brocco et al 2009 "Service Provisioning Framework for a Self-Organized Grid". In the proposed work authors discussed about the framework which provide on demand service and having two layers which preform various operation to deliver the requested service. They define this frame works as two layer architecture. Working of layers are a lower layer using fully distributed bio-inspired algorithm for providing member management and basic communication an upper layer uses the services provided but its lover layer and facilities grid services for example (Discovery of the resources, resource monitoring). In this paper researcher combined advantages and disadvantages of both organized and unorganized system and overwhelming of their boundaries and proposed a framework that is self-organizing in the nature according to different grid procedure. This framework provides membership management and basic communication among the groups which effectively lead to deliver high level grid service. In the given framework low-level communication is achieved by recognizing local connection between the nodes of grid in order to reduce the length of path and exploited the use of network resources which are decentralized in nature by using Bio-inspired algorithm to cooperate them. Optimization of network is based on entirely dispersed centered on collective bio motivated algorithm. This framework is composed of two layer a one is management layer used for self-organization and other layer is used to provide grid interface layer. The proposed framework reduces the distance between the nodes which are cooperating in the grid [32].

Rasjid et al 2004 "discussed Analysis and Provision of QoS for Distributed GridApplications". In this paper authors proposed a framework that livelihood provide QoS (Quality of service) management in heterogeneous geographically dispersed distributed grid computing in the framework (OGSA- Open grid service architecture). This framework has three functioning phase's instituting, action, expiry. In the institution phase a user application request for service with desired QoS (Quality of service) parameter then proposed framework analyze the user requirement and run a facility discovery process based on their user application requirements which is defined by user. In action phase preform many different type of operation such as monitoring of resources, accounting, adaption and if not able to provide desired service that time then communicate the user and tell again submit his requirements. (Resource bound, Time bound). In resource domain user can state a confident measurement of resource he want for expiation we discuss one case that user required to access to 20% processing unit time, 15Mbps network bandwidth is accessible out of a total of 100Mbps available. In time domain apportionment approach a user may appeal for whole resource to be reserved for secluded use and no other user and application are permitted to segment the resource for e.g. preserving full capability of CPU [33].

CHAPTER 3
PRESENT WORK

3.1 Problem Formulation

3.1.1 Short comings of the current existing algorithms:

The effective utilization of resources and process of execution of jobs lies at the heart of the system. Huge number of mechanisms are devised to bring more effectiveness into the utilization of resources and improve the overall execution of jobs with the help of various given factors and benchmarks.

The existing schedulers assign the resource to the system on the fairness factor only which limits the ability of the process where a job requires diverse set of resources for the execution. Secondly, the scheduler has its limitations in defining the optimum size for the block size to carry the data where a bargain is required between the efficiency in job completion and resource utilization. As a consequences, this in-efficiency and in-ability leads to problem of assigning the resources more than required by the job (excessive allocation) and also results in fragmentation .This has a high impact on the effective utilization of the resources since it leads to the problem of under utilization which eventually effects the performance by enhancing the waiting periods, turnaround time, make span etc.

3.2 Objectives

Following objects are given in regard to the proposed work:

a) To improve the resource allocation process.

b) To achieve better performance by enhancing the job scheduling process.

c) Compare the existing algorithms with the proposed method.

d) Validate the results towards the claim of enhancement in the resource utilization and scheduling.

3.3 METHODOLOGY

1. Selection of appropriate data sets and platform for the distributed systems environment

2. Review of the existing methodologies.

3. Implementing the existing algorithms and techniques with single type of resource requirement and generating the results.

4. Implementing the proposed work:

    4.1. Pack the memory and processor together as a single unit of resource to accommodate multiple resource requirements.

    4.2. Perform the mapping of the available resources with the job requirement for the best fit (best combination of the resources for optimum utilization)

    4.3. Allocation of the job to the best fit as per the step 4.2.

    4.4. Perform the execution the proposed technique.

5. Analysis of the result and comparison with the existing techniques.

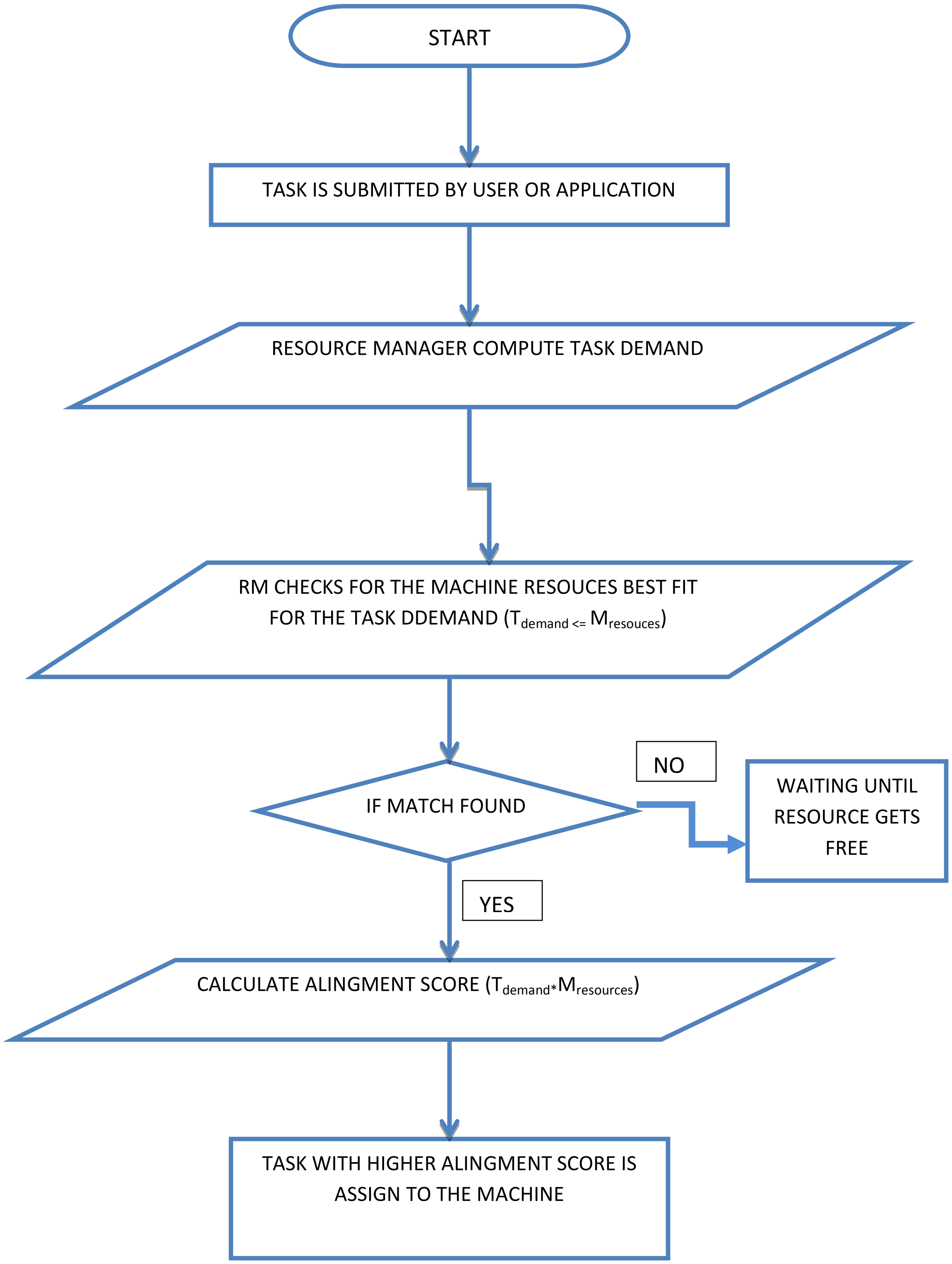

START
TASK IS SUBMITTED BY USER OR APPLICATION
RESOURCE MANAGER COMPUTE TASK DEMAND
RM CHECKS FOR THE MACHINE RESOUCES BEST FIT FOR THE TASK DDEMAND ($T_{demand} <= M_{resouces}$)
IF MATCH FOUND
NO
WAITING UNTIL RESOURCE GETS FREE
YES
CALCULATE ALINGMENT SCORE ($T_{demand} * M_{resources}$)
TASK WITH HIGHER ALINGMENT SCORE IS ASSIGN TO THE MACHINE

Fig 7: Flow chart of proposed algorithm

CHAPTER 4
Result and Discussion

The evaluation setup of the both the scheduler is:

Parameters	Values
Number of application	10
Number of queues	3
Number of nodes	4
Number of racks	1
Number of tasks	2472
Node memory (MB)	10240
Average tasks per application	248
Average application per queue	3
Node vcores	10

For the experimental and simulations, the implementation of the existing strategies have been done using Hadoop 2-6.0 which is the latest stable release of the Hadoop. The Yarn scheduler load simulator libraries have been used for performing the analysis of the scheduler. The trace file .json provided by Apache has been used as dataset for testing the schedulers.

Initially the existing strategies are being implemented .The analysis as per be below given graphs represent that the schedulers are working with the single resource requirement only .In figure 8 the containers are not fully utilized throughout. In figure 11, it represents the underutilization of the resource memory in queue. Figure 13 represent the application cost when it is added to by the scheduler which is very high initially. Further from figure 14 to figure 19, the results are represented for the proposed resource allocation technique. Our proposed work represents the better utilization of containers in figure 15. When the multiple resources are packed together the proposed work shows the increased utilization of both memory and core as per the figures 16. The cost for adding the new applications is comparatively lesser in our proposed strategy.

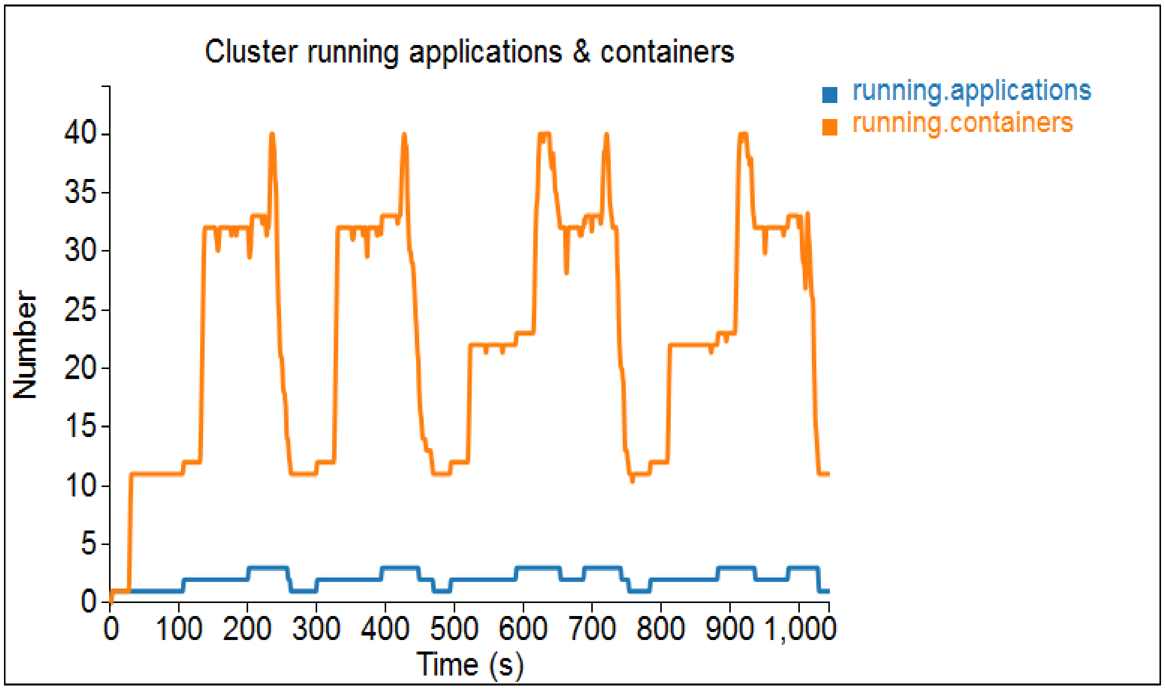


Fig 8: Labels the number of containers and running applications in existing scheduler.

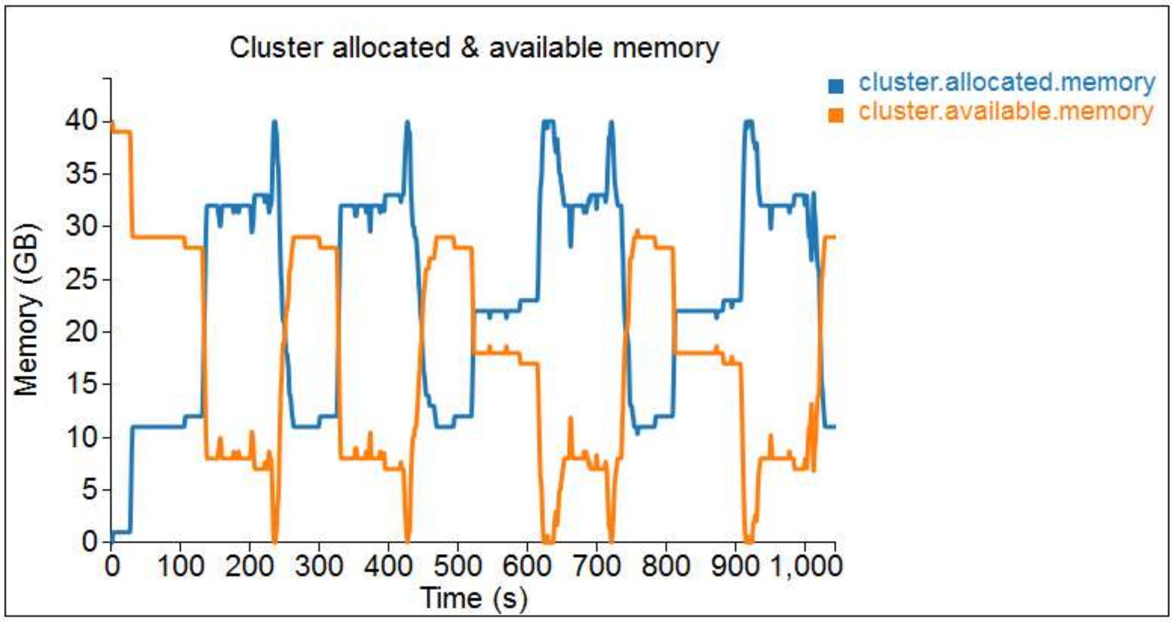


Fig 9: Labels the available memory and allocated in the cluster existing scheduler.

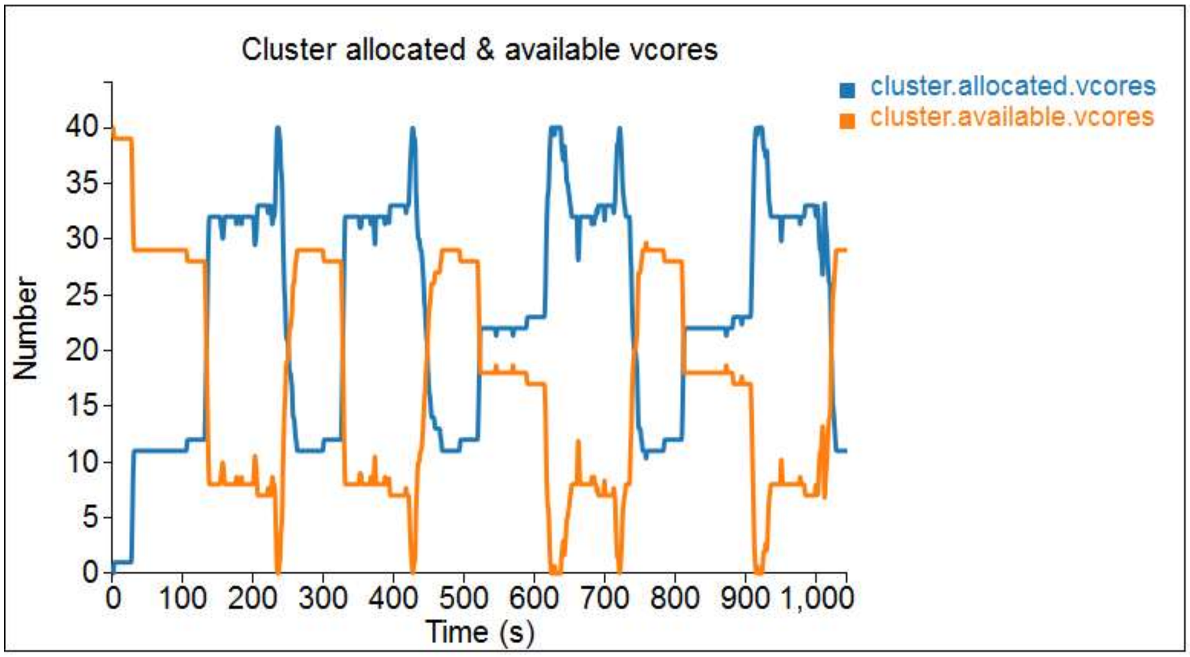


Fig 10: Available Vcores and allocated Vcores in the existing scheduler.

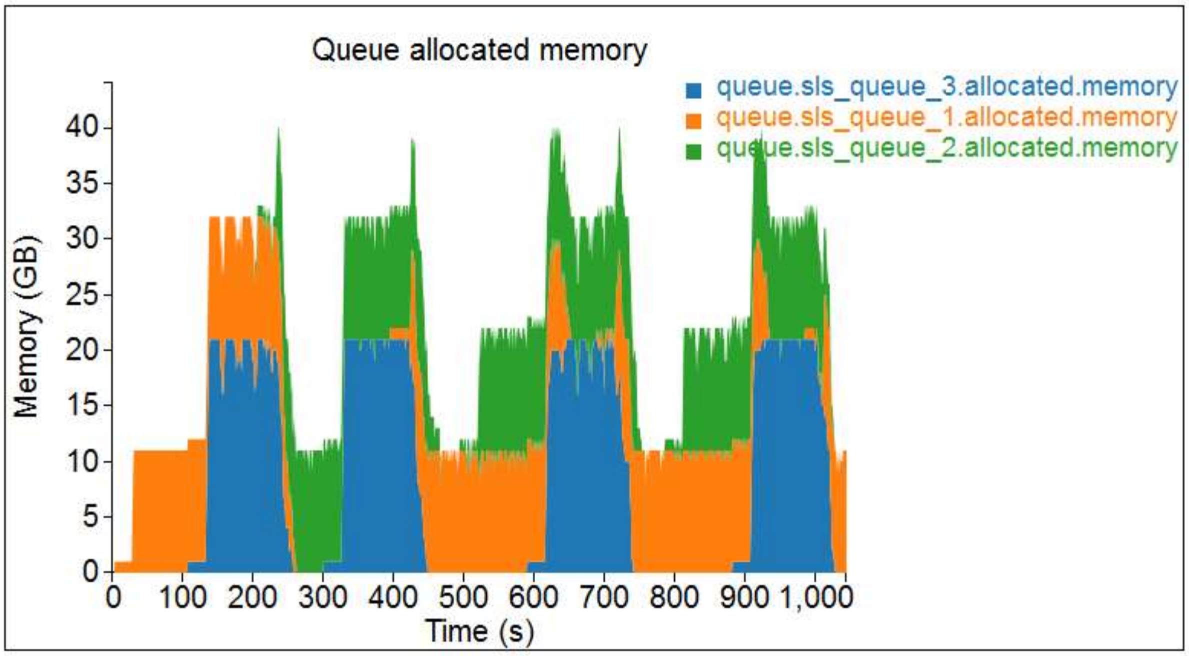


Fig 11: Memory allocation for each queue in the existing scheduler.

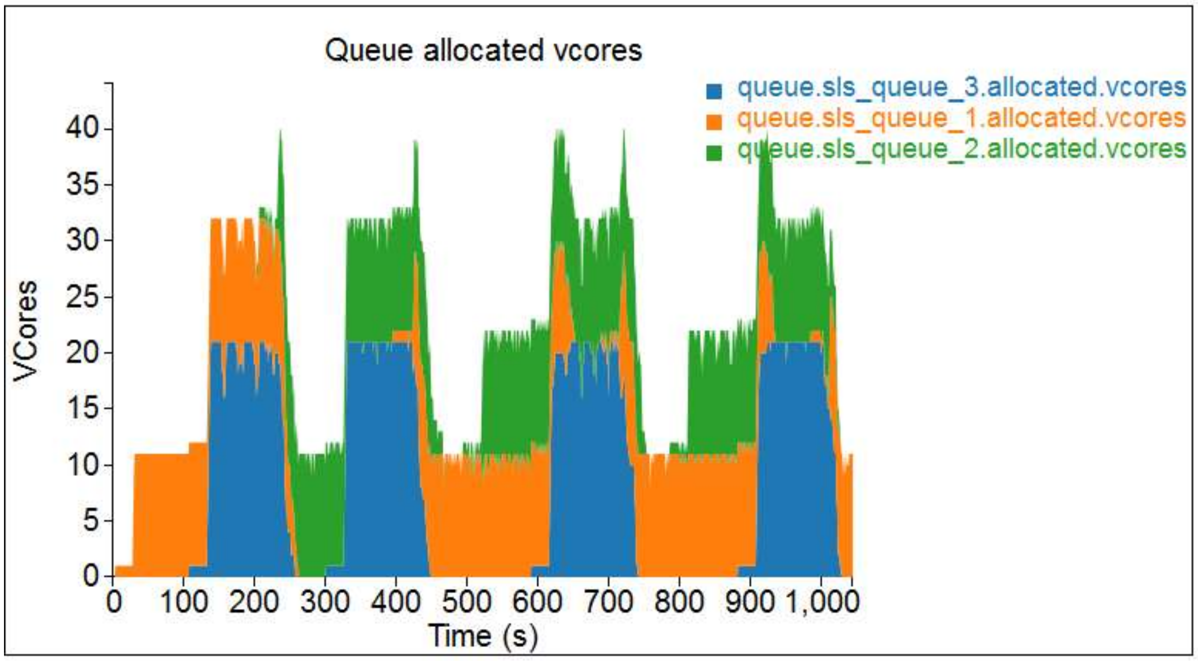


Fig 12: Vcores allocation for each queue in the existing scheduler.

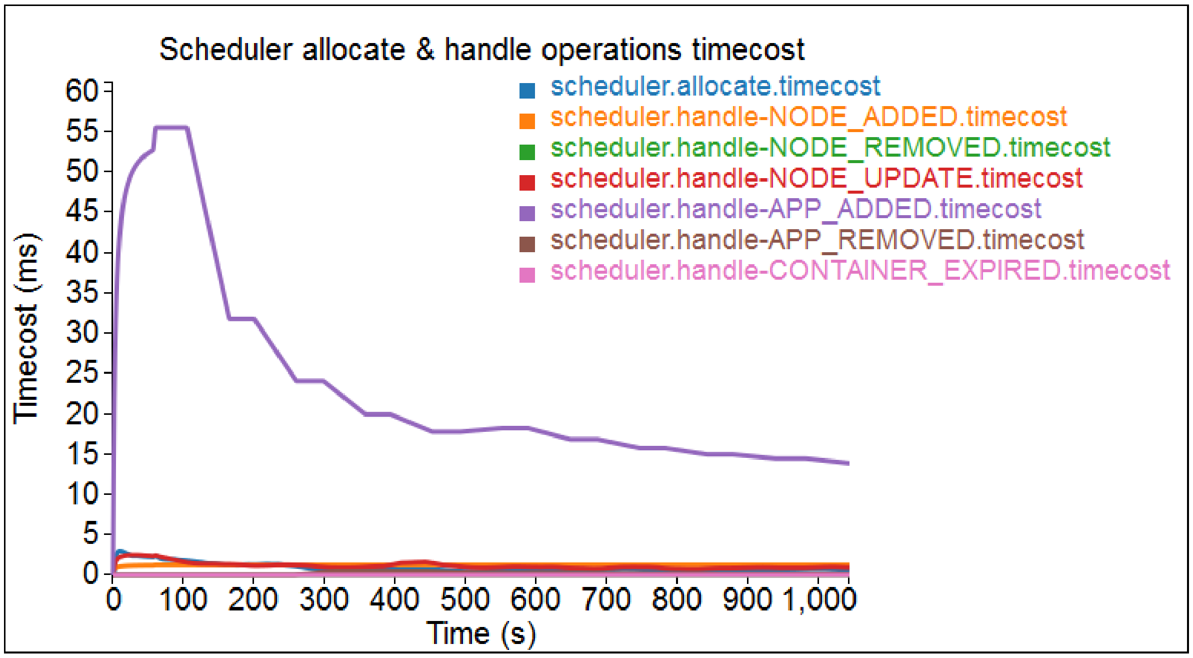
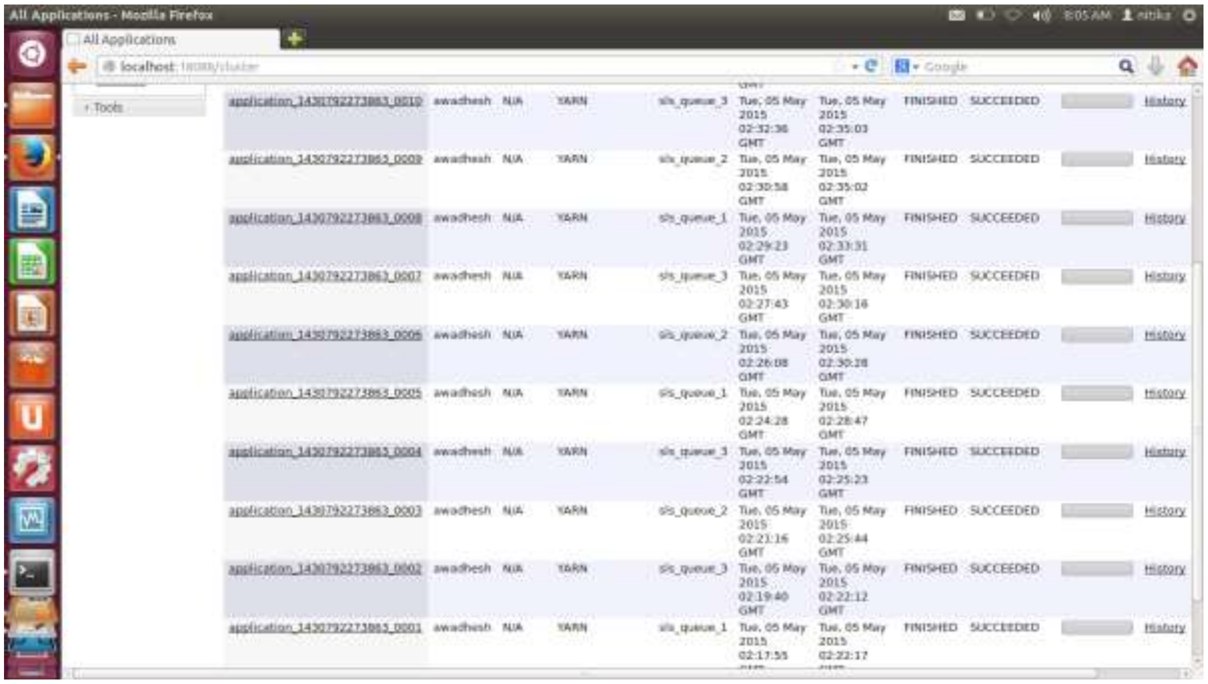


Fig 13: Labels the timecost for each scheduler operation in the existing scheduler.



application_1430792273863_0010	awadhesh	N/A	YARN	s/s_queue_3	Tue, 05 May 2015 02:32:36 GMT	Tue, 05 May 2015 02:35:03 GMT	FINISHED	SUCCEEDED	History
application_1430792273863_0009	awadhesh	N/A	YARN	s/s_queue_2	Tue, 05 May 2015 02:30:58 GMT	Tue, 05 May 2015 02:35:02 GMT	FINISHED	SUCCEEDED	History
application_1430792273863_0008	awadhesh	N/A	YARN	s/s_queue_1	Tue, 05 May 2015 02:29:23 GMT	Tue, 05 May 2015 02:33:31 GMT	FINISHED	SUCCEEDED	History
application_1430792273863_0007	awadhesh	N/A	YARN	s/s_queue_3	Tue, 05 May 2015 02:27:43 GMT	Tue, 05 May 2015 02:30:16 GMT	FINISHED	SUCCEEDED	History
application_1430792273863_0006	awadhesh	N/A	YARN	s/s_queue_2	Tue, 05 May 2015 02:26:08 GMT	Tue, 05 May 2015 02:30:28 GMT	FINISHED	SUCCEEDED	History
application_1430792273863_0005	awadhesh	N/A	YARN	s/s_queue_1	Tue, 05 May 2015 02:24:28 GMT	Tue, 05 May 2015 02:28:47 GMT	FINISHED	SUCCEEDED	History
application_1430792273863_0004	awadhesh	N/A	YARN	s/s_queue_3	Tue, 05 May 2015 02:22:54 GMT	Tue, 05 May 2015 02:25:23 GMT	FINISHED	SUCCEEDED	History
application_1430792273863_0003	awadhesh	N/A	YARN	s/s_queue_2	Tue, 05 May 2015 02:21:16 GMT	Tue, 05 May 2015 02:25:44 GMT	FINISHED	SUCCEEDED	History
application_1430792273863_0002	awadhesh	N/A	YARN	s/s_queue_3	Tue, 05 May 2015 02:19:40 GMT	Tue, 05 May 2015 02:22:12 GMT	FINISHED	SUCCEEDED	History
application_1430792273863_0001	awadhesh	N/A	YARN	s/s_queue_1	Tue, 05 May 2015 02:17:55	Tue, 05 May 2015 02:22:17	FINISHED	SUCCEEDED	History

Fig 14: Snapshot of the jobs execution using existing technique of resource allocation

The result of proposed resource allocation technique are given below

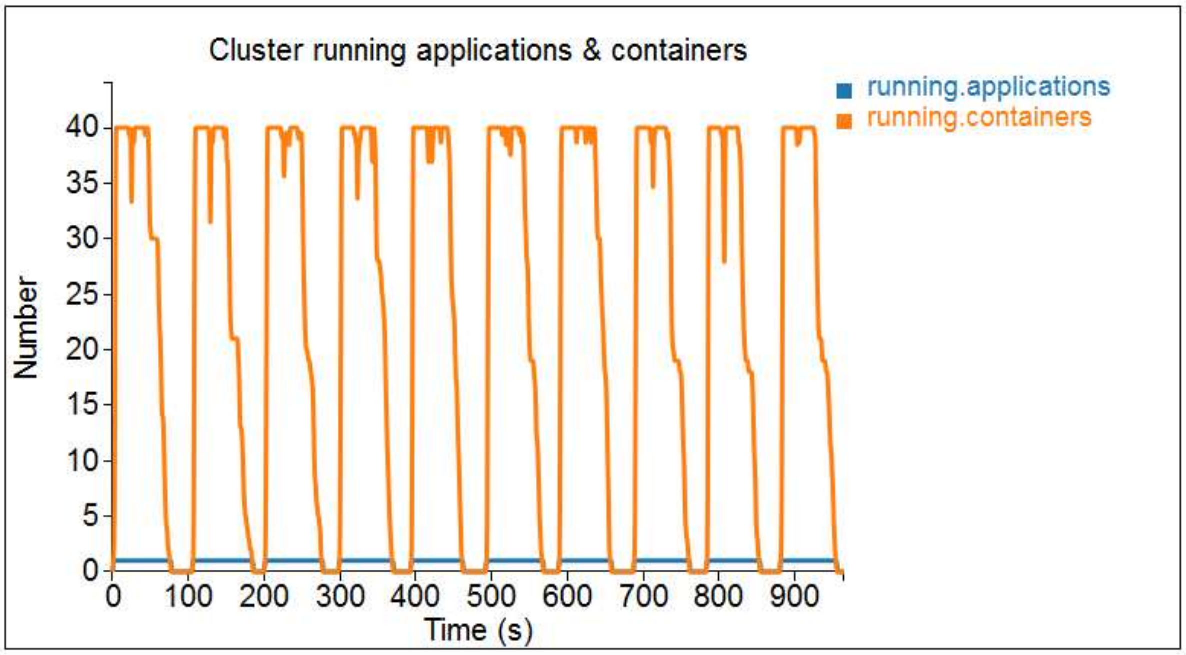


Fig 15: Labels the number of containers and running applications in proposed scheduler.

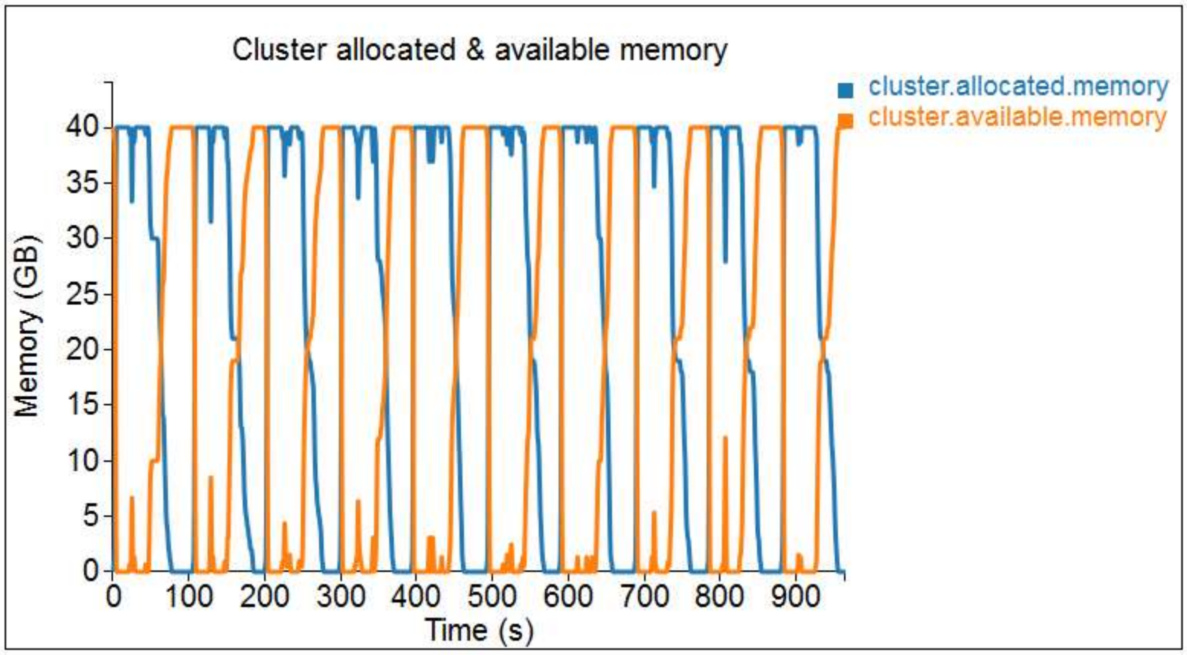


Fig 16: Labels the available memory and allocated in the proposed scheduler.

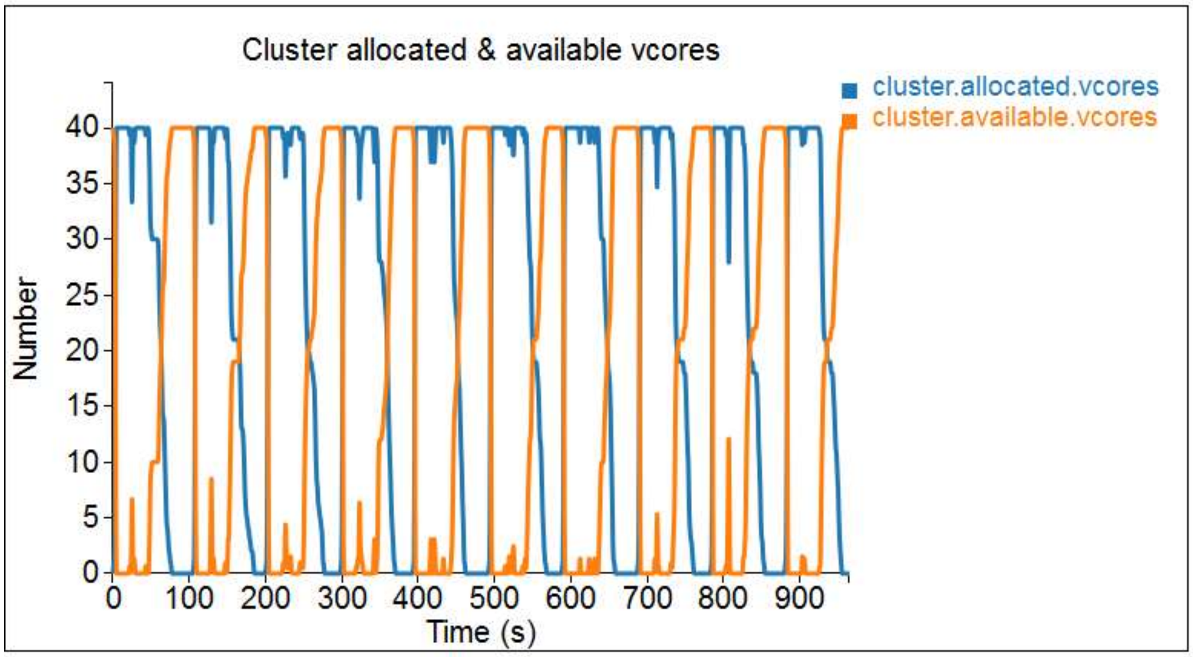


Fig 17: Labels the available Vcores and allocated Vcores in the proposed scheduler

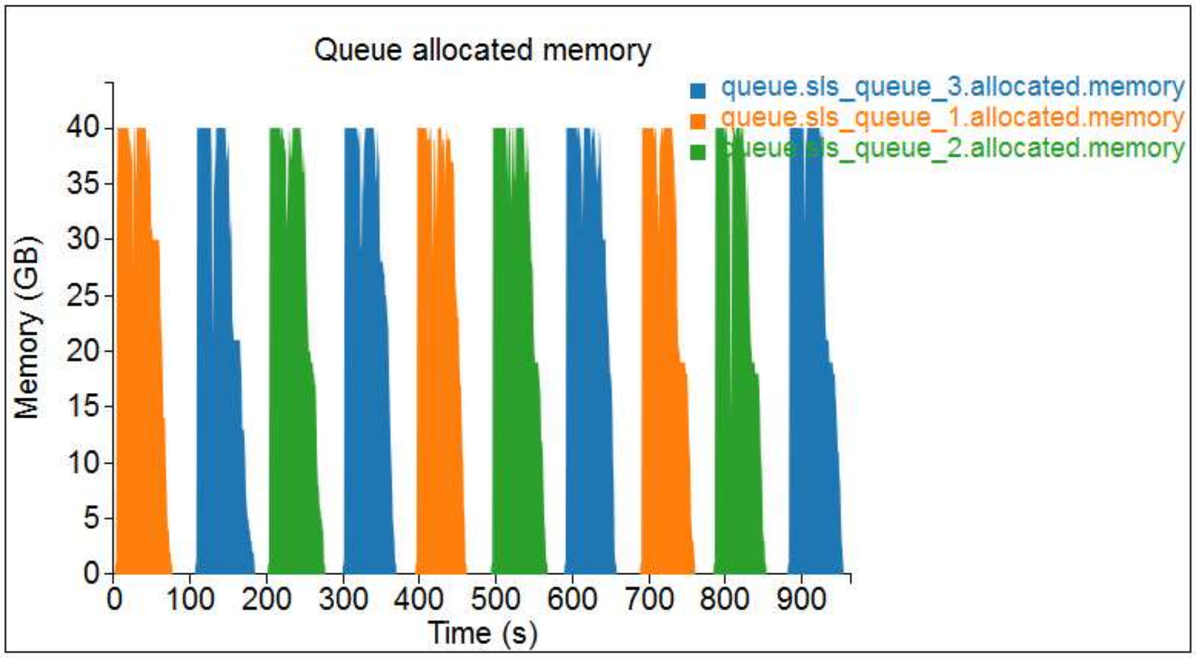


Fig 18: Labels memory allocation for each queue in proposed scheduler.

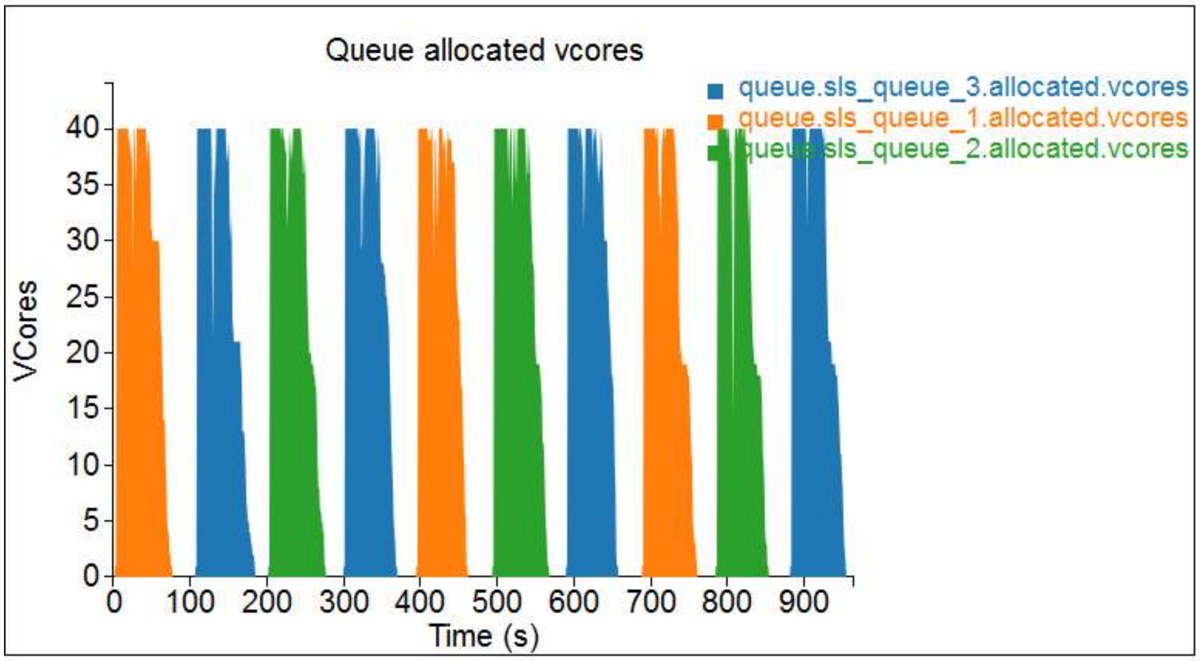


Fig 19: Labels the Vcores allocation for each queue proposed scheduler.

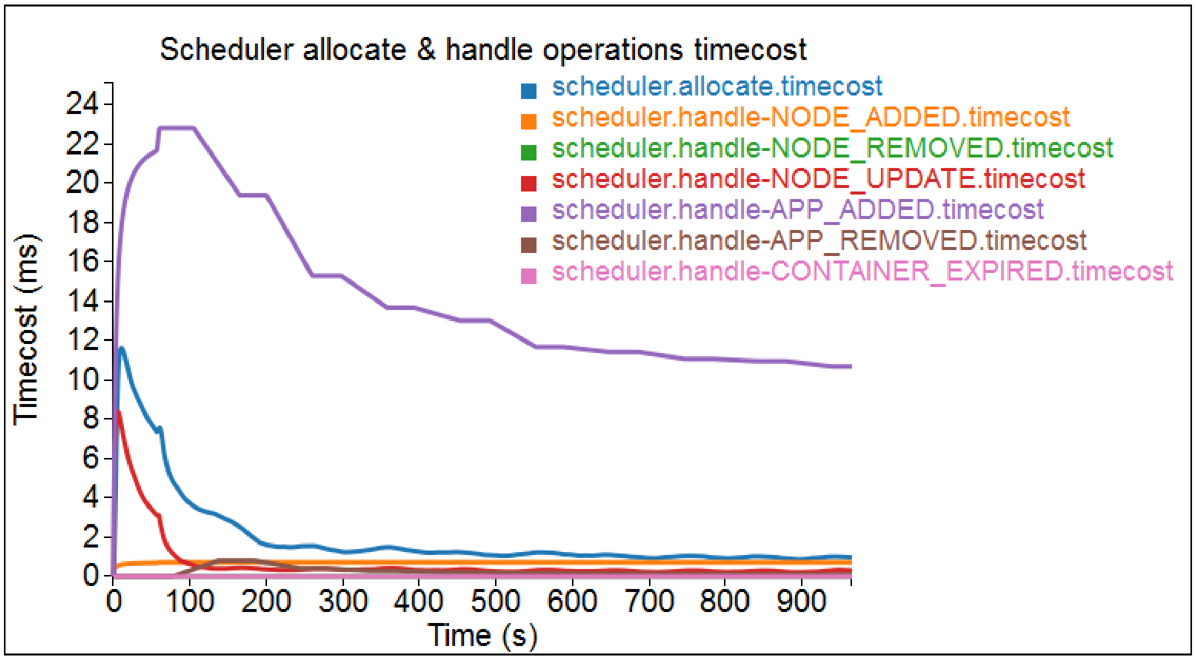
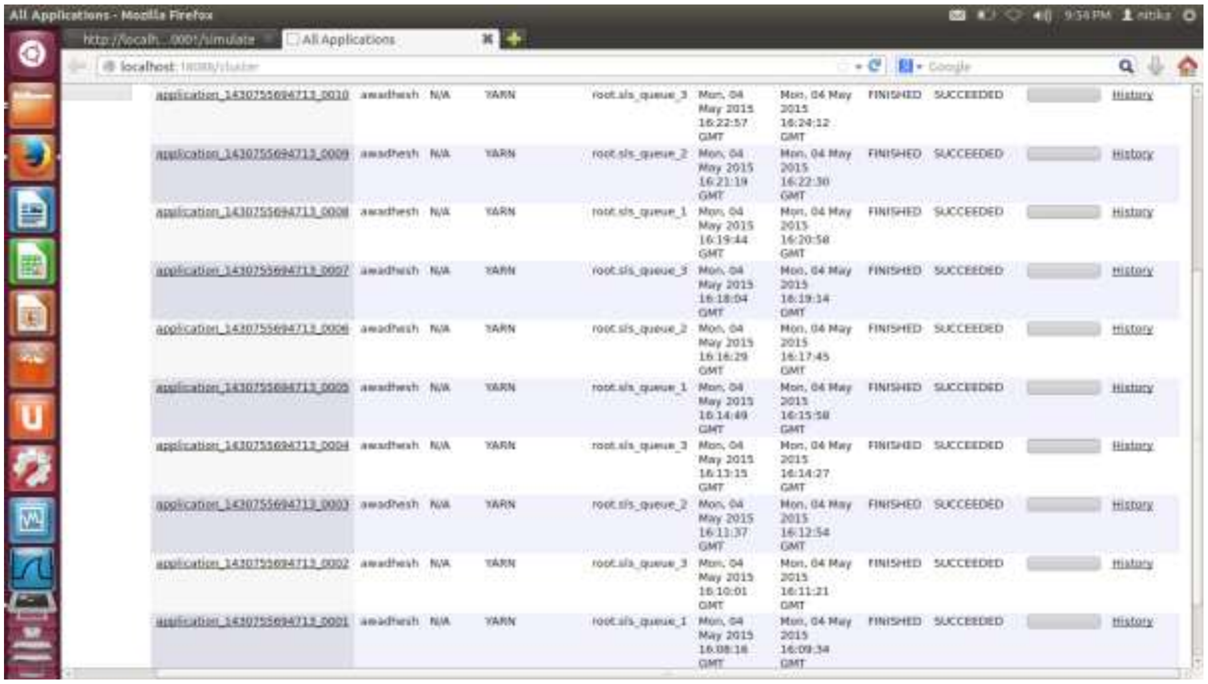


Fig 20: Labels the timecost for each scheduler operation in proposed scheduler



All Applications - Mozilla Firefox

application_1430755694713_0010	awadhesh	N/A	YARN	root.sls_queue_3	Mon, 04 May 2015 16:22:57 GMT	Mon, 04 May 2015 16:24:12 GMT	FINISHED	SUCCEEDED		History
application_1430755694713_0009	awadhesh	N/A	YARN	root.sls_queue_2	Mon, 04 May 2015 16:21:19 GMT	Mon, 04 May 2015 16:22:30 GMT	FINISHED	SUCCEEDED		History
application_1430755694713_0008	awadhesh	N/A	YARN	root.sls_queue_1	Mon, 04 May 2015 16:19:44 GMT	Mon, 04 May 2015 16:20:58 GMT	FINISHED	SUCCEEDED		History
application_1430755694713_0007	awadhesh	N/A	YARN	root.sls_queue_3	Mon, 04 May 2015 16:18:04 GMT	Mon, 04 May 2015 16:19:14 GMT	FINISHED	SUCCEEDED		History
application_1430755694713_0006	awadhesh	N/A	YARN	root.sls_queue_2	Mon, 04 May 2015 16:16:29 GMT	Mon, 04 May 2015 16:17:45 GMT	FINISHED	SUCCEEDED		History
application_1430755694713_0005	awadhesh	N/A	YARN	root.sls_queue_1	Mon, 04 May 2015 16:14:49 GMT	Mon, 04 May 2015 16:15:58 GMT	FINISHED	SUCCEEDED		History
application_1430755694713_0004	awadhesh	N/A	YARN	root.sls_queue_3	Mon, 04 May 2015 16:13:15 GMT	Mon, 04 May 2015 16:14:27 GMT	FINISHED	SUCCEEDED		History
application_1430755694713_0003	awadhesh	N/A	YARN	root.sls_queue_2	Mon, 04 May 2015 16:11:37 GMT	Mon, 04 May 2015 16:12:54 GMT	FINISHED	SUCCEEDED		History
application_1430755694713_0002	awadhesh	N/A	YARN	root.sls_queue_3	Mon, 04 May 2015 16:10:01 GMT	Mon, 04 May 2015 16:11:21 GMT	FINISHED	SUCCEEDED		History
application_1430755694713_0001	awadhesh	N/A	YARN	root.sls_queue_1	Mon, 04 May 2015 16:08:18 GMT	Mon, 04 May 2015 16:09:34 GMT	FINISHED	SUCCEEDED		History

Fig 21: Snapshot of the job execution using proposed technique of resource allocation

The comparative analysis of the graph

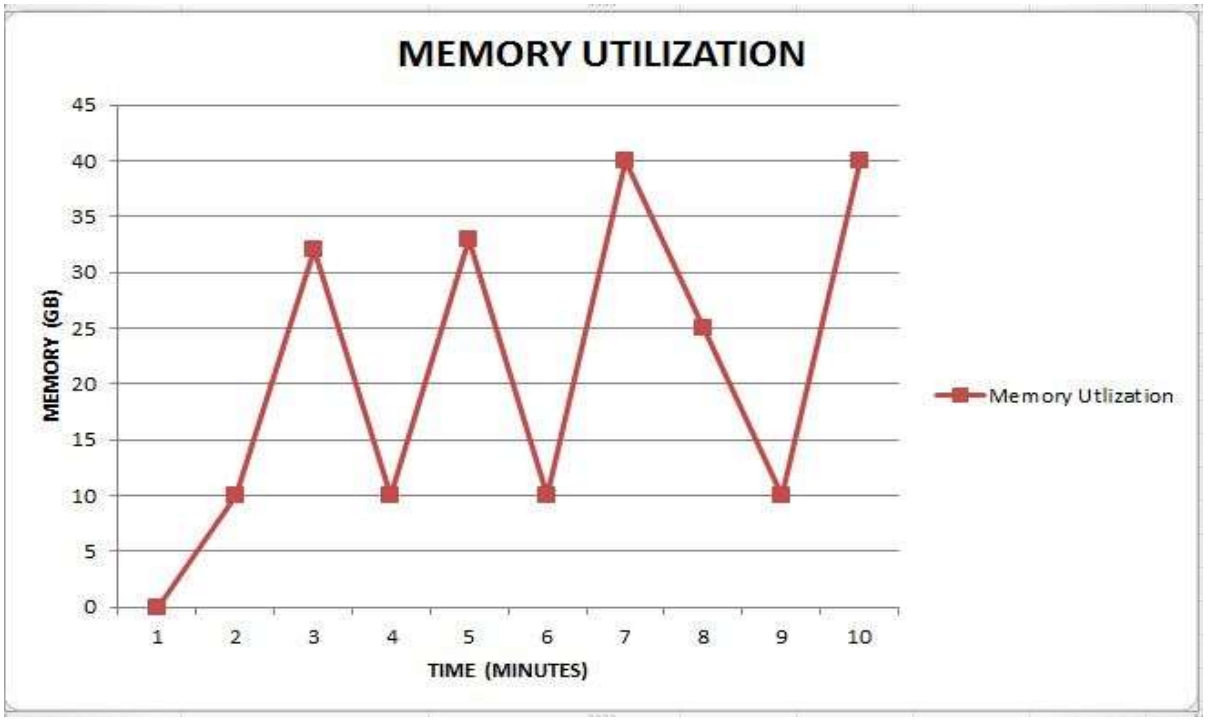


Fig 22: Labels the memory utilization of single resource scheduling technique

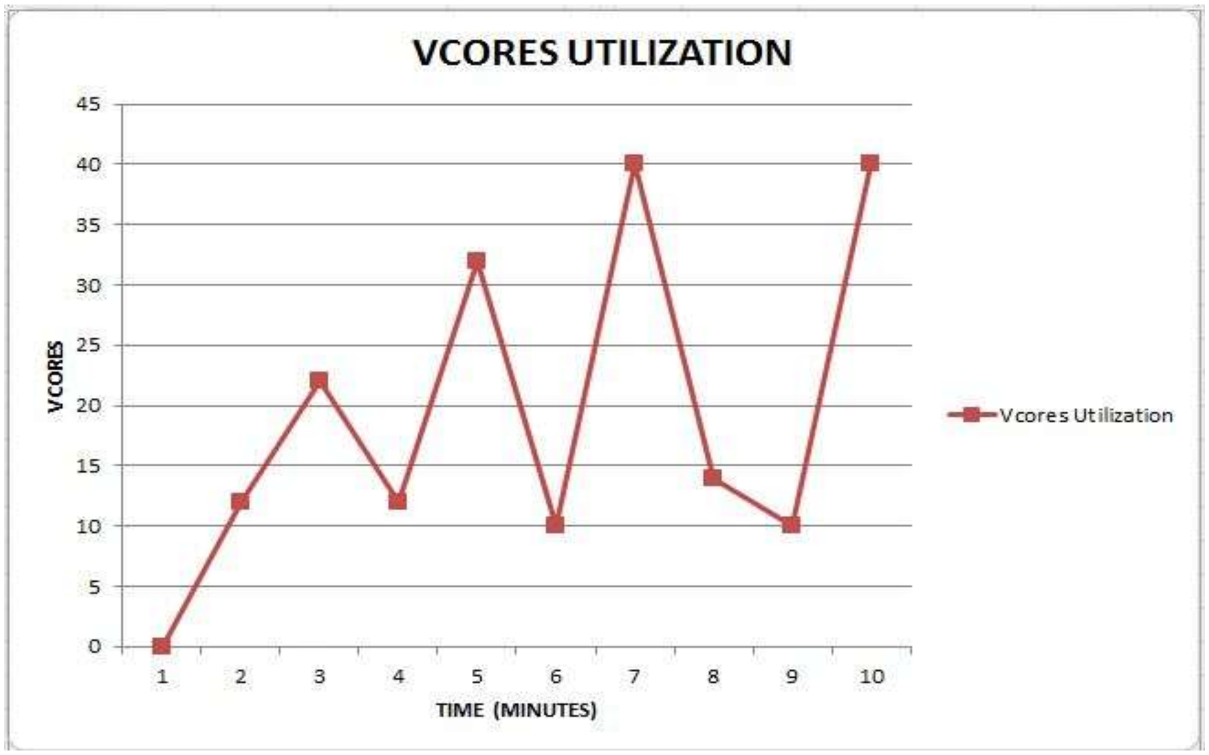


Fig 23: Labels the vcores utilization of single resource scheduling technique

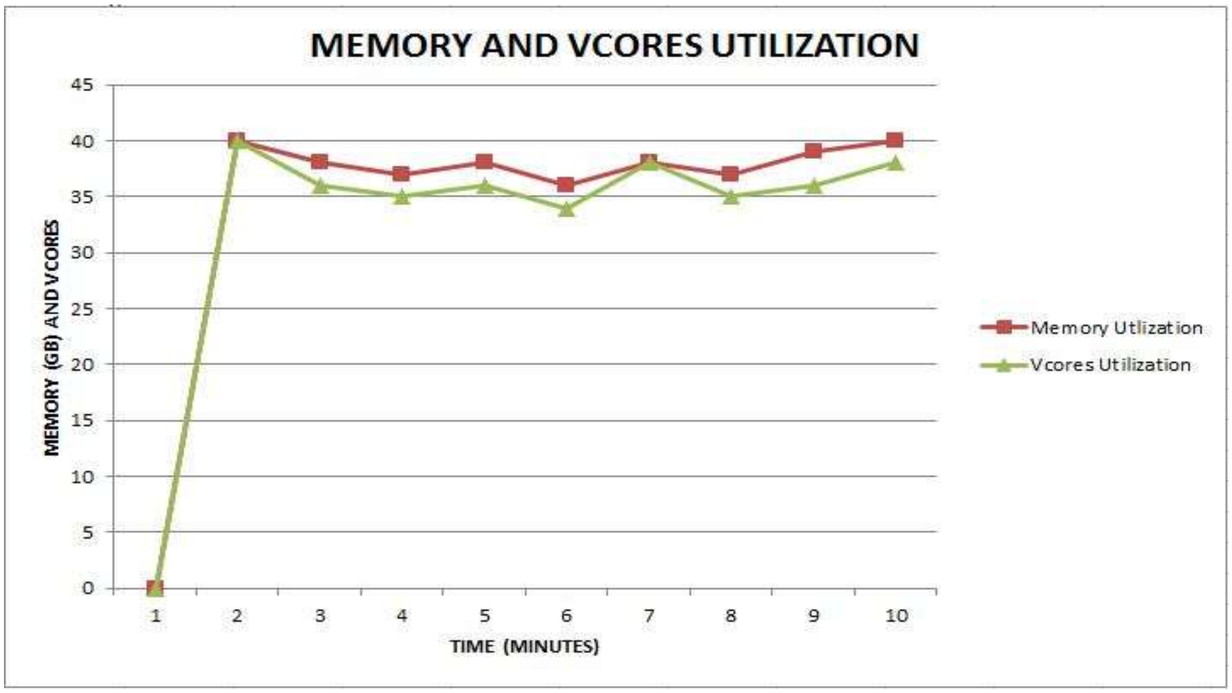


Fig 24: Labels the vcores and memory utilization of single multi resource scheduling technique

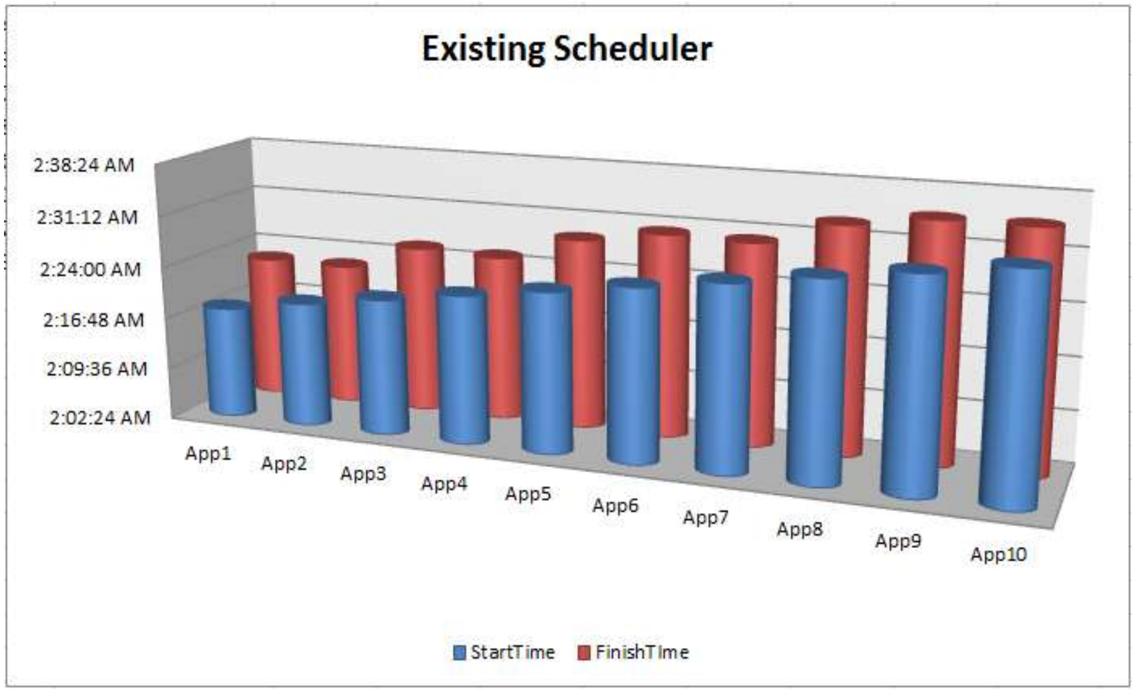


Fig 25: Time taken by the each application execution using existing technique

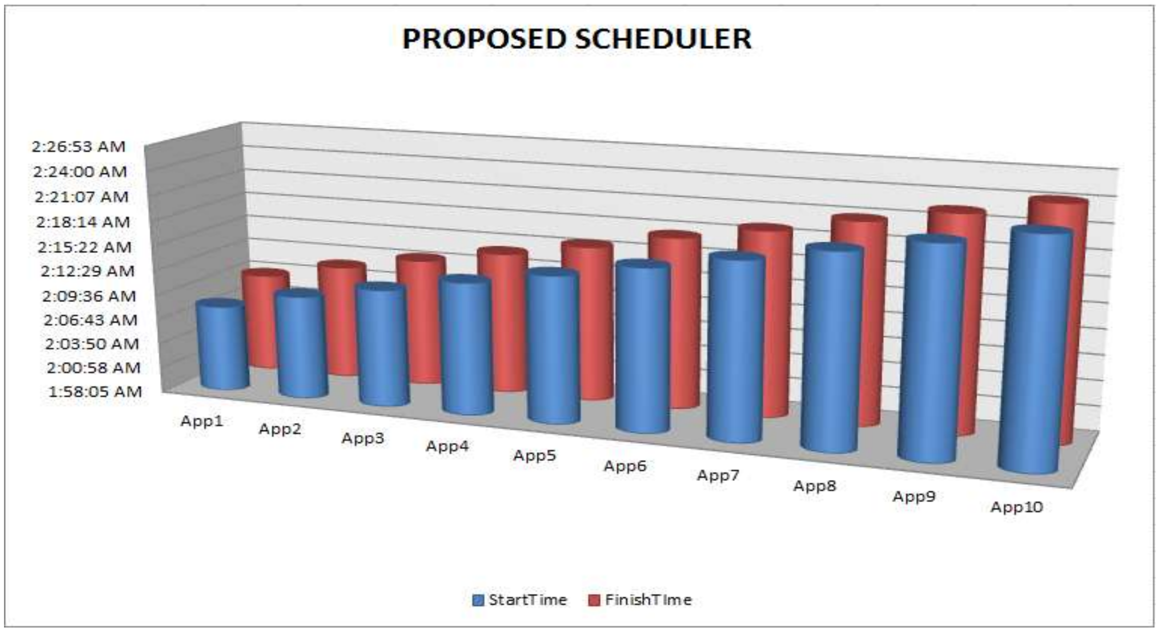


Fig 26: Time taken by the each application execution using proposed technique

CHAPTER 5
Conclusion and Future Scope

With speedy growth in the massive amounts of data every day, the challenges of processing it are also getting higher and higher. The resources and processing powers seems to be limited in front of the size of the data. The need for the hour is to bring effectiveness in the exiting methodologies and techniques for bringing utilization out of them and improve the overall performance of the work in demand. Hadoop has shown huge lead in processing the large amounts of data through distributed environment. The existing state of art systems and infrastructure are undoubtedly performing well but the still demands and brings in the need for further enhancements and improvement in them. The process of scheduling the jobs and effective utilization of resources is and will always be a challenge looking at the way data is growing. The proposed work has given different dimensions for processing the data by bringing improvements in the existing scheduling processes and resource. The limitations of the existing algorithms have been exploited and removed. The existing system is unable to process the job with multiple resource requirements. Our system has overcome this limitation by packing together the memory and processing requirement together for allocating to a single job and addressed the issue of multiple resource requirements which overcomes problem of fragmentation and also demonstrated the increased utilization of the resources. In this work we have used two measures for job packing which is memory and processing, in future there is a further scope for exploring the other multiple requirements of jobs such as graphics, networks, local vs distant processing, cache etc.

CHAPTER 6
References

[1] Andrew S. Tanenbaum .(2011) Distributed System Princple and Paradigms, PHI Learning Ltd., New Delhi.

[2] Luis Ferreira, Fabiano Lucchese Grid Computing in Research and Education By IBM/Redbooks.

[3] Yang Wang, Wei Shi, "Budget-Driven Scheduling Algorithms for Batches of MapReduce Jobs in Heterogeneous Clouds," IEEE transaction on cloud computing, vol. 2, issue. 1, p. 306-319, 2013.

[4] M. Hammoud and M.F. Sakr, "Locality-aware reduce task scheduling for MapReduce," In Cloud Computing Technology and Science (CloudCom) IEEE Third International Conference, p. 570-576, 2011.

[5] Yahoo, " MapReduce," [online]

Available: https://developer.yahoo.com/hadoop/tutorial/module4.html

[6] M. Zaharia et al. Delay Scheduling, "Delay Scheduling :A Technique For Achieving Locality And Fairness In Cluster Scheduling," In EuroSys 5th European conference on Computer systems, p. 265-278, 2010.

[7] M. Isard, M. Budiu, Y. Yu, "Distributed Data-Parallel Programs from Sequential Building Blocks," In Proc. of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, p.59-72, 2009.

[8] Jilan Chen,Dan Wang and Wenbing Zhao, "A task scheduling algorithm for hadoop platform," Journal of computers," vol. 8, no.4, 2013.

[9] S. Agarwal et al. "Re-optimizing data parallel computing" In NSDI, 2012.[8] M. Al-Fares et al. A Scalable, Commodity Data Center Network Architecture. In SIGCOMM, 2008.

[10] M. Al-Fares et al. “A Scalable, Commodity Data Center Network Architecture,” In SIGCOMM, 2008.

[11] M. Chowdhury et al. “Managing Data Transfers in Computer Clusters with Orchestra,” In SIGCOMM, 2011.

[12] M. Chowdhury et al. “Leveraging Endpoint Flexibility in Data-Intensive Clusters,” In SIGCOMM, 2013.

[13] A. Ghodsi et al. “Dominant Resource Fairness: Fair Allocation Of Multiple Resource Types,” In NSDI, 2011.

[14] A. Greenberg et al. “A Scalable and Flexible Datacenter Network,” In SIGCOMM, 2009.

[15] S. Gulwani et al. “SPEED: Precise And E›cient Static Estimation Of Program Computational Complexity,” In POPL, 2009.

[16] C. Guo et al. “BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers,” In SIGCOMM, 2009.

[17] M. Harchol-Balter et al. Connection Scheduling in Web Servers. In USITS, 1999.

[18] M. Isard et al. “Dryad: Distributed Data-Parallel Programs From Sequential Building Blocks,” In EuroSys, 2007.

[19] M. Isard et al. “Quincy: Fair Scheduling For Distributed Computing Clusters,” In SOSP, 2009.

[20] L. Lu et al. “Predictive VM Consolidation on Multiple Resources: Beyond Load Balancing,” In IWQoS, 2013.

[21] A. Rasmussen et al. “Themis: An I/O-E›cient MapReduce,” In SoCC, 2012.

[22] A. Shieh et al. “Sharing the Data Center Network,” In NSDI, 2011.

[23] T. Tannenbaum et al. “Condor – A Distributed Job Scheduler. In Beowulf Cluster Computing with Linux,” MIT Press, 2001.

[24] A. Tusoo et al. “Hive: A Warehousing Solution Over A Map-Reduce Framework,” Proc. VLDB Endow, 2009.

[25] Rajni Aron, Inderveer Chana et al “Formal QoS Policy Based Grid Resource Provisioning Framework, ” J. Grid Computing (2012) 10:249–264 DOI 10.1007/s10723-012-9202-y.

[26] Naidila Sadashiv, S. M Dilip Kumar et al “Cluster, Grid and Cloud Computing: A Detailed Comparison,” In 6th International Conference on Computer Science & Education (ICCSE 2011) August 3-5, 2011. Superstars Virgo, Singapore.

[27] Lehman, T., Sobieski, J., Jabbari, B.: “DRAGON: a technique for service provisioning in heterogeneous Grid networks,” Commun. Mag.44(3), 84–90 (2006)

[28] Assuncao, M.D., Buyya, R.: “Performance analysis of allocation policies for intergrid resource provisioning,” Information and Software Technology Journal, vol. 51(1), pp. 42–55. ISSN: 0950-5849. Elsevier Science, Amsterdam, The Netherlands (2009)

[29] Murphy, M.A., Kagey, B., Fenn, M., Goasguen, S.: “Dynamic provisioning of virtual organization clusters,” In:Proceedings of 9th IEEE International Symposium on Cluster Computing and the Grid (CCGrid’09), pp. 364–371. Shanghai, China (2009).

[30] Filali, A., Hafid, A., Gendreau, M.: “Adaptive Resources Provisioning for Grid applications and services,” In: Proceedings of IEEE International Conference on Communications, ICC’08, pp. 186–191. China (2008)

[31] Filali, A., Hafid, A., Gendreau, M.: “Bandwidth and Computing Resources Provisioning for Grid Applications and Services,” In: Proceedings of IEEE International Conference on Communications, ICC’09,pp. 1–6 (2009)

[32] Brocco, A., Hirsbrunner, B.: “Service Provisioning Framework for a Self-Organized Grid,” In: Proceedings of 18th International Conference Computer Communications and Networks (ICCCN 2009). ISSN: 1095-2055, Print ISBN: 978-1-4244-4581-3, pp. 1–6 (2009).doi:10.1109/ICCCN.2009.5235315

[33] Al-Ali, R., Amin, K., Laszewski, V.G., Rana, O., Walker, D., Hategan, M., Zaluzec, N.: "Analysis and provision of QoS for distributed Grid applications," J. Grid Computing2(2), 163–182(2004)doi:10.1007/s10723-004-6743-8.

CHAPTER 7
Appendix

HDFS- Hadoop Distributed File System

DFS- Distributed File System

SLS- Scheduler Load Simulator

YARN- Yet Another Resource Negotiator

DS- Distributed System