Energy Efficient Load Balancing in Computational Grid

By

ZEENAT TARIQ

RESEARCH CENTRE FOR MODELING & SIMULATION

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

2015

Energy Efficient Load Balancing in Computational Grid

Zeenat Tariq

Research Centre for Modeling & Simulation

A thesis submitted to the National University of Sciences & Technology in partial fulfillment of the requirement for the degree of Masters of Science

2015

STATEMENT OF ORIGINALITY

I Zeenat Tariq Registration No.NUST201260252MRCMS64012F here by certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

Date Student Signature

DEDICATION

This effort of mine is dedicated to my PARENTS who supported me throughout, to my HUSBAND and to my BROTHERS without their support the completion of this report wouldn’t have been possible.

ACKNOWLEDGEMENTS

"Recite in the name of your Lord who created - Created man from a clinging substance. Recite and your Lord is the most Generous - Who taught by the pen - Taught man that which he knew not."

The QURAN, Sura Al-Alaq 96/1-5

First and foremost I must thank Allah Almighty, Who has given me the strength to complete my degree and thesis. I am indebted to the Holy Prophet Muhammad (PBUH) for the teachings of life which is throughout guidance for me.

I am extremely thankful to my parents, brothers and my family members for their prayers, love, patience and continuous support throughout my MS degree.

I am highly grateful to my supervisor Dr. Salman Shahid and my Co Supervisor Dr. Babar Nazir from COMSATS Institute of Information and Technology Abbottabad for their valuable guidance, keen interest and for providing all the support in completion of this research. Their help and guidelines have always been inspiring and worth mentioning.

I am also thankful to my G.E.C committee members Dr Adnan Maqsood and Engr Taufeeq Ur Rehman for their valuable support and suggestions which helped me to improve the thesis. I would like to thank Engr Sikander Hayat Mirza for getting me started with my research. Without his support and motivation I was unable to complete my research successfully.

Last but not the least, I would say thanks to a number of people around me who helped and supported me at different stages of my MS studies and without their support it might become much difficult for me to accomplish this degree.

Abstract

Computational grid is a pool of computational resources which handles large computations and consumes enormous energy. Load balancing and energy consumption are two main performance parameters of the computational grid. Since performance of a computational grid is highly susceptible to variations in load and energy, therefore it is imperative to pay attention to these aspects in grid environment.

In this thesis we proposed an energy aware dynamic load balancing strategy for tasks scheduling in grid referred to as Energy Efficient Load Balancing (EELB) in computational grid. EELB combined energy efficiency and load balancing paradigm to improve the utilization and to reduce the energy consumption of the resources. EELB considered a scenario where users require services which are either time or budget constraint and based on user's requirements the tasks are allocated to the grid resources for execution.

The simulations were carried out using GridSim environment and the results were compared with the Improved Load Balancing on Enhanced GridSim with Deadline Control (IEGDC) method. The performance indices such as: finished gridlets, unfinished gridlets, resubmitted times and response time, along with proposed parameters such as: cost, energy consumption and energy saved were used for the comparison of both methods. The simulation results showed that EELB outperformed the published IEGDC method by 28% in terms of finished gridlets, 35% by cost and 38% in energy consumption.

Keywords: Computational Grid, Load Balancing, Energy efficiency, GridSim.

Table of Contents

LIST OF FIGURES

LIST OF ABBREVIATIONS

QoS	Quality of Service
CO	Carbon Dioxide
CPU	Central Processing Unit
SPM	Static Power Management
DPM	Dynamic Power Management
PE	Processing Entity
MI	Multiple Instruction
DCD	Dynamic Component Deactivation
DPS	Dynamic Performance Scaling
EGDC	Enhanced GridSim with deadline control
IEGDC	Improved Enhanced GridSim with Deadline Control
EELB	Energy Efficient Load Balancing
UT	Upper Threshold
LT	Lower Threshold

Chapter 1

1 Introduction

1.1 Outline

In this chapter the area of research and basic concepts about the research are discussed. These include grid computing, balancing load, power consumption and workload consolidation. Further the aim and objectives of the research. In the last, the organization of thesis is described.

1.2 Area of Research

Grid is a pool of computer resources from different locations to achieve a common goal. Computational Grids are mainly for computational intensive applications that involves large number of tasks. Grid computing is an effective computing environment with advance development of wide-area high-speed networks and powerful computational resources. The storage space and computational limitations of traditional distributed systems can be overcome by completely pointing the resources of computing sites/servers that are under-utilized. Load and resource management are very important grid services at the service level of grid computing infrastructure where issue of efficient load balancing shows a common problem for most grid computing infrastructure developers [1].

The old dispersed methods are not precise in grid computing because of the dispersion of enormous resources and large size tasks to be moved [2]. The load balancing is an open research issue as there are many problems in selecting an appropriate resource for data that is when and which resource should be selected for computing data, separation of computation data from one resource to another. Thus, it is a demanding issue to design an algorithm for load balancing which can cover all the features. Load Balancing techniques are used to reduce makespan, storage consumption, SQA violation, access latency and network bandwidth. Many approaches and algorithms have been suggested, executed, and shown in previous studies. In those studies, the load balancing algorithms can increase the response time of a user submitted tasks by confirming high utilization of existing resources [3-5].

In Grid environment minimizing job's response time and turnaround time (queue size in which jobs are waiting, execution time of the job and data transfer time of the job) mostly rely on selection of location it should be placed for execution of job and where to transfer the load for job execution. Hence scheduler/broker should assign the jobs to proper grid resources while ensuring minimum energy consumption.

1.2.1 *Power Consumption*

The power consumption of data server also increases with overutilization of resources. Power utilization in data servers will remain to rise quickly unless an innovative approach to power management techniques are applied [6]. The excessive amount of energy used by computational grid provides reason to study into energy saving strategies both from environmental and commercial point of view. Many of the research groups are concentrating on improving energy efficiency. Corporations such as Google [7] are devoted to increasing energy efficiency in data centers/servers.

The usage of electrical energy in data servers causes a great rise in functional rates and CO_2 production. Worldwide data server's energy depletion has increased by 56% in the five years i.e. from 2005 to 2010. In 2010 it was expected to be in range of 1.5% -2% of the entire electrical energy consumed[8].

Energy efficiency is considered in systems ranging from small systems to large computational centers. So it is significant to decrease waste of power consumed by the large data server due to high utilization of resources. This can be done in the way these resources are utilized to handle tasks load by proper management algorithms and resource selection.

1.2.2 *Workload Consolidation*

There are many ways to reduce power consumption of the resources. The power of underutilized resources can be handled by implementing power ratio to the server, where power consumption of the server is according to the workload competence. Dynamic voltage frequency scaling (DVFS) method can be applied for this purpose in which voltage and frequency are adjusted dynamically according to the resource demand. Using this technique a physical server can still

consumes 70% of its power if it is in idle state since 70% of the dynamic power is utilized when a server is in idle state and 30% of the power is consumed by other components [9].

Task Migration and switching off resource is another technique for balancing load and achieving energy efficiency in computational grid. Transferring of tasks from one resource to another resource is done to improve the performance, hardware use and power consumption. In order to efficient complete the tasks in deadline also reduce the network burden, the migration of tasks is needed. So if a resource is over utilized, transfer load from overloaded resource to lightly loaded resources, and if a resource is under-utilize, transfer all load from underutilized resource to another resources and switch off resource thus it is possible to attain energy proficiency. The different entities can be modeled for managing power and load of the resources.

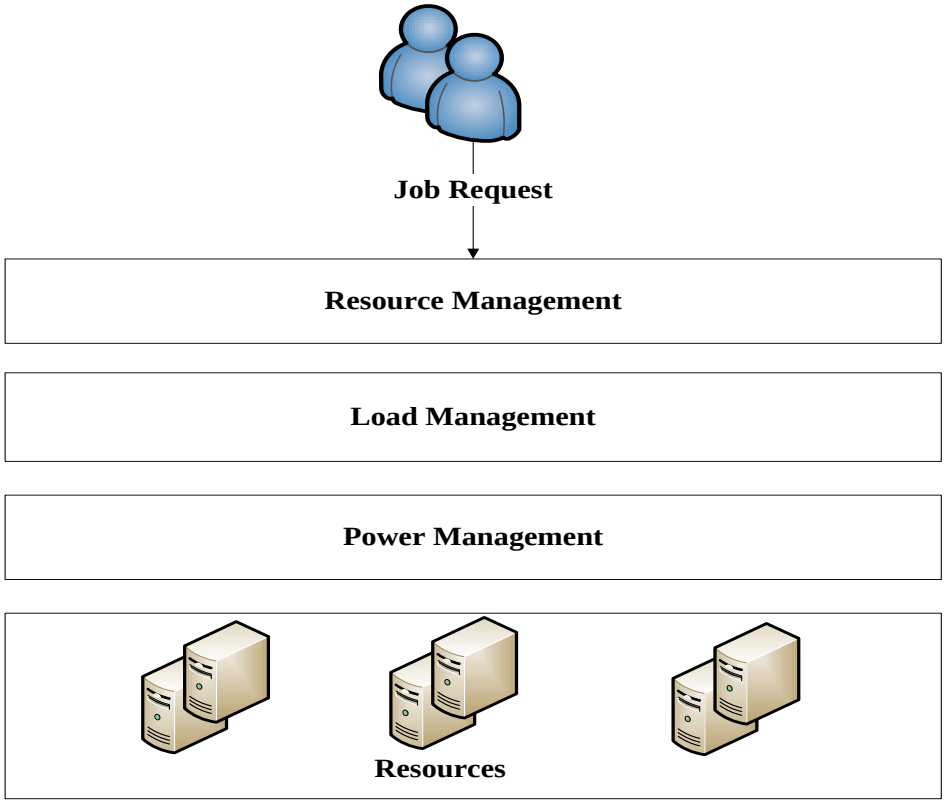

Figure 1 the main view of the systems

1.3 Problem Statement

Even though previous work addressed many methods for balancing load and smart scheduling of tasks among resources there are no such methods that distribute tasks based on user requirement.

The problem: Schedule tasks in such a way that minimizes the response time of the tasks, and reduces the energy consumption of the resources. We have to develop an efficient scheduling technique, which will be used for load balancing while ensuring minimization of energy consumption and maintaining QoS requirement. Managing tasks without QoS requirement may result in the poor response of task execution as well as energy inefficiency. Tasks on the resource are mainly concerned with two things: energy efficiency and load balancing. Our designed method is an effective scheduling method for tasks that will be used for balancing load and energy efficiency. For this purpose, we created a scenario to address user requirements. There are two different types of users: time constraint user and budget constraint users. Time type users are associated with the timely completion of tasks. Budget type users are associated with the savings that is cost and budget. So for that case we keep the underutilized resources turned off to save energy plus get services within budget. While in case of time constraint users, we will apply the balancing algorithm and keep the resources on for the tasks to get completed within time deadline. So for this purpose we have simulated the proposed mechanism and existing schemes and compared them. Simulation of a proposed work is carried out in GridSim Toolkit 5.2 that defines data grid architecture to support job scheduling and load balancing. Extensive simulations were carried out to compare the results.

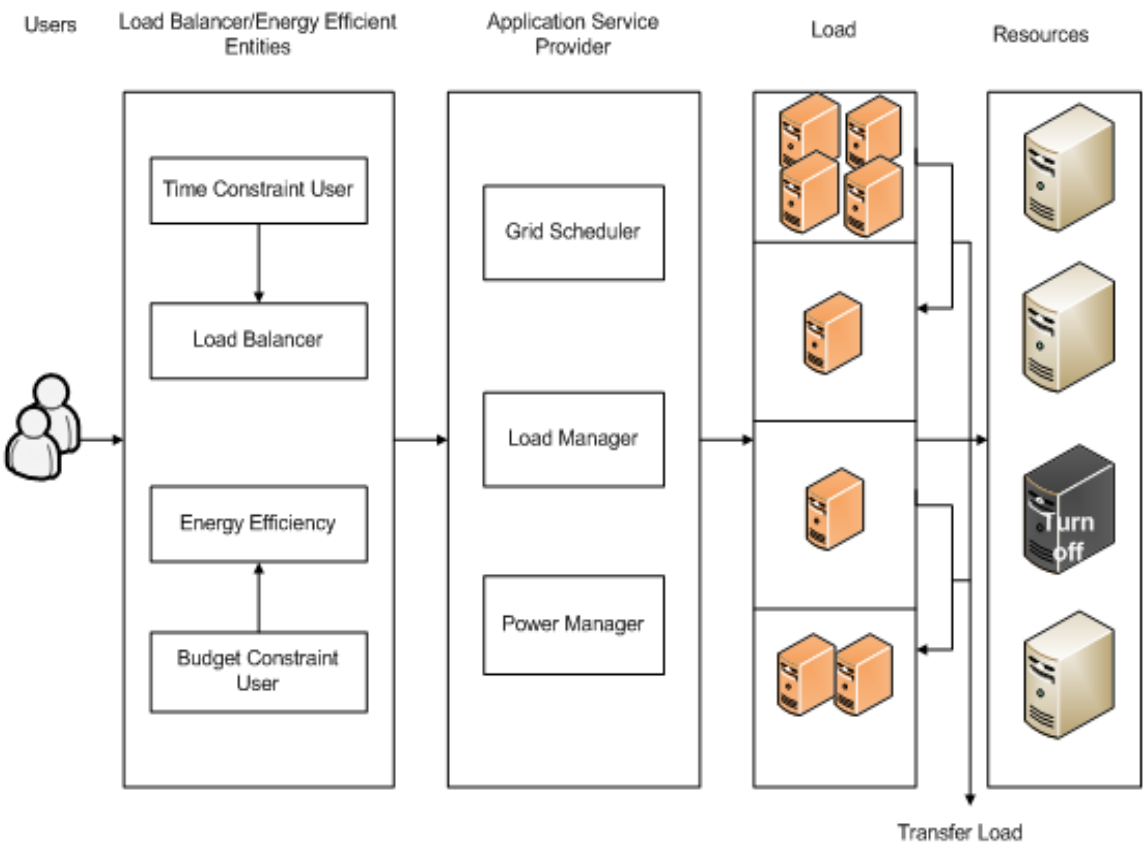

Figure 2 the proposed system architecture

1.4 Research Objectives

The primary objective of the research was to balance the load among resources such that SQA should not be violated and energy efficiency is achieved. The main objectives are:

- To get knowledge of existing scheduling mechanisms
- Study the working of GridSim environment
- Construction of Energy Efficient Load Balancing Model
- Designing an algorithm for the proposed model
- Implementation of the algorithm in GridSim tool

- Comparison of algorithm with the previous implemented algorithm
- Propose suggestions to future research in the same area

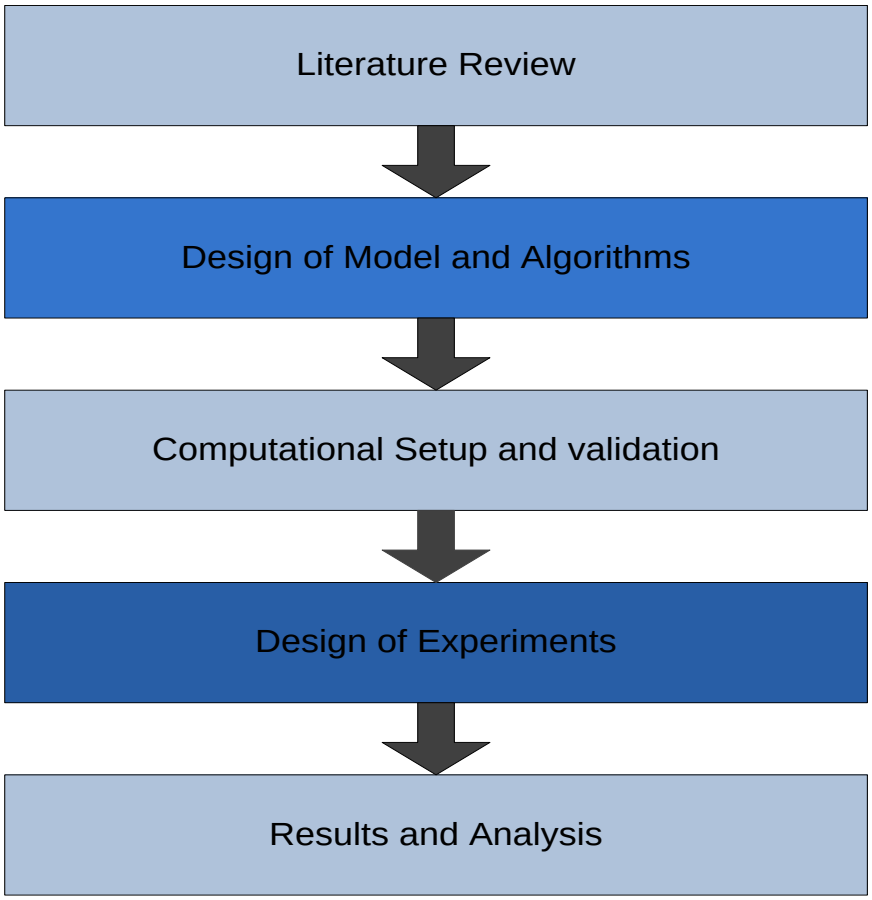

Figure 3 Research Objectives Flow Chart

1.5 Organization of Thesis

Chapter 1: Introduction

A comprehensive background and area of research are documented in this chapter. The motivation, problem statement and research objectives are also documented in the chapter.

Chapter 2: Literature Survey

This chapter comprises basic concept of grid computing, information about load balancing, challenges/issues in scheduling tasks, and a review on power management.

Chapter 3: System Model

This chapter focuses on the solution of the problem statement. An optimized architecture of system model is proposed. Furthermore this chapter defines the algorithms designed for the proposed work.

Chapter 4: Implementation

In this chapter, the implementation details and experimental description of the GridSim is mentioned.

Chapter 5: Simulation and Results

In this chapter the simulation results are discussed and a comparison of the proposed work is done with the previous work.

Chapter 6: Conclusion, Recommendation and Future Work

In this chapter conclusion of the thesis is derived and some recommendations for future line of action are described.

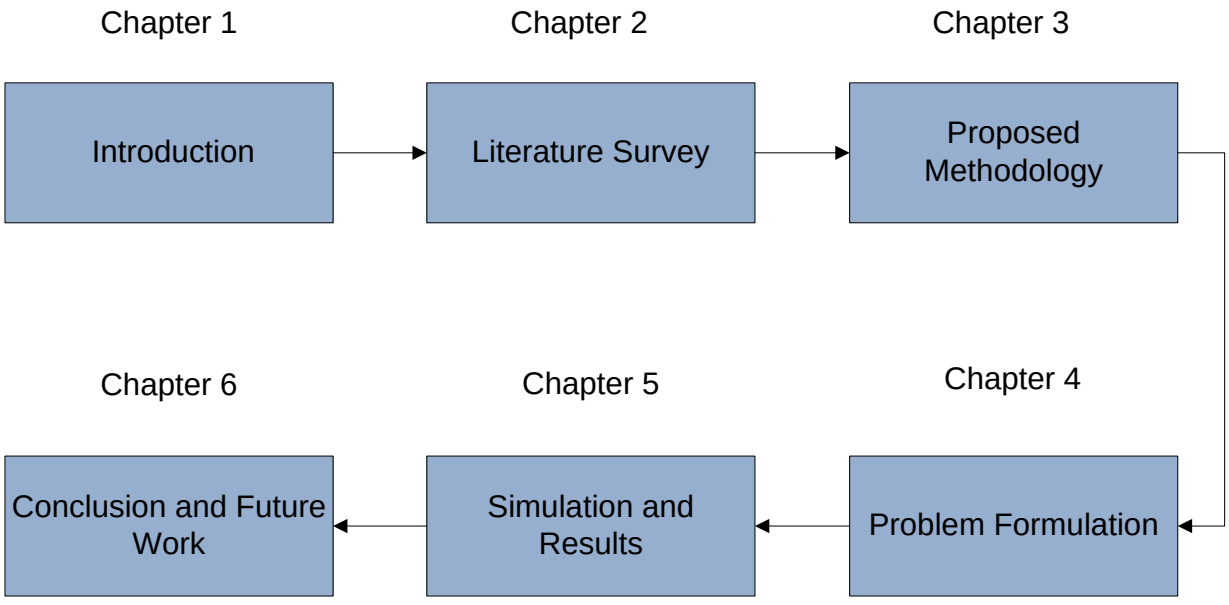

Figure 4 The classification of Thesis

Chapter 2

2 Literature Survey

2.1 Outline

In this chapter, the definition and background of grid computing is described in details. Further the comprehensive survey is carried out on the scheduling tasks and its associated issues/challenges, power consumption and management of the servers. In the end, identification of existing work and some gaps in the literature from the survey is also presented that gives the direction for future research.

2.2 Grid Computing

Grid computing is an approach of using computing as utility. It is a group of computer resources from several sites to reach a collective objective. Grid Computing is an extremely attractive field of research since its inception. It can be thought of as a dispersed structure having heavy data load involving large numbers of tasks. Basically grid computing is of two types: data grid and computational grid. A computational grid is a pool of computational resources that consumes enormous energy due to the large tasks given for execution where as a data grid is for storage of large amount of data.

Grids are created with common grid middleware software libraries[1]. These practical prospects have led to the possibility of using multiple located resources to solve huge issues in science, engineering and commerce. The computing resources from several managerial domains combine to make one large grid. This can enable deals as in utility computing or make it convenient to accumulate free computing networks.

To yield complete benefit of such grid scheme, resource management and job placement are important methods that are delivered at the service level of the grid software framework. The problem of task placement and load balancing signify a collective issue for most of the grid environment.

For dealing with such main problems of load balancing in grid computing, a ‘Service Oriented Load Balancing Framework’ is suggested by Gooswamey et al. [5]. A layered design is followed

in this framework where the top most layers consist of grid users that submit tasks for computation. Resource broker is the middle layer and is responsible for distributing the tasks among resources and the details about the available resources. The bottom layer consists of group of resources which are responsible for executing jobs. These resources can be PCs, clusters, supercomputer etc.

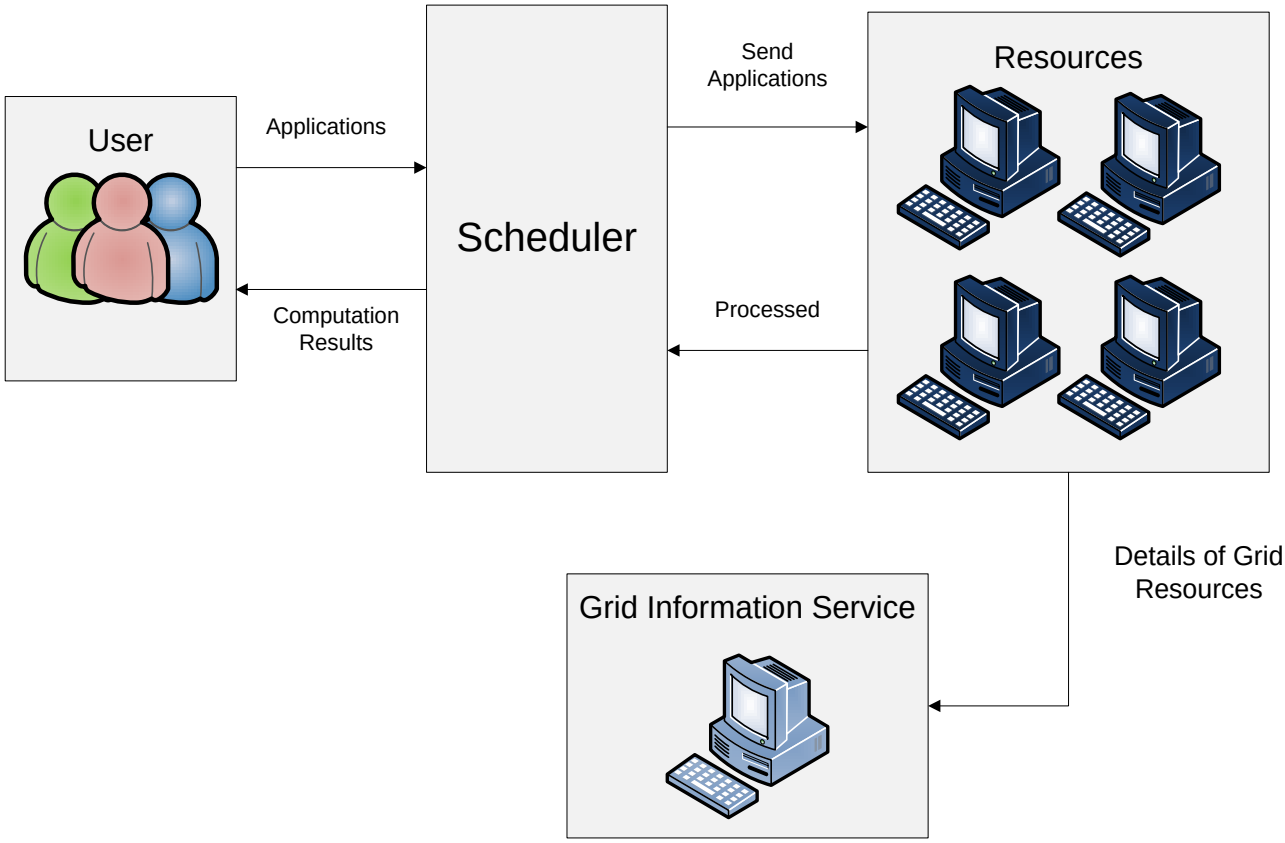

Figure 5 Basic Working of the Grid system

2.3 Issues in Scheduling Tasks

With the growing number of computer intensive applications there is a need to ensure proper resource management for large number of computational tasks. The centralized data servers have been transformed into performance bottleneck by the pervasive development of Internet. In a very short span thousands of requests may be received by popular servers. Such frequent queries may lead to overhead on servers and the network resulting in increased delay of services being

provided to the end users. Such bottleneck may prove to be more effective when we consider media servers or grids that may process very large datasets to process remote applications [10].

For this reason, the tasks are distributed on multiple resources for execution in Computational Grid. High utilization of resources may decrease the response time of the tasks but also results in high consumption of energy. If the tasks are not equally distributed and the load is not balanced among the resources then it may violate user's requirement. It is therefore not appropriate to avoid task scheduling with such increased complexity, scalability and functionality in a Computational Grid. A millisecond delay in data execution can cause a great loss for commercial business oriented applications in terms of customer's satisfaction. It may not only violate the service quality agreement (SQA) but also affect customer's base and revenue respectively. Efficient scheduling is thus of critical importance in paradigm of distributed systems. Scheduling decision includes when and where to place tasks? Which tasks to transfer? The utilization of the resource is checked before sending any tasks.

2.4 Load Balancing Techniques

For managing resource selection and tasks placement, dynamically managing resources and decision making of load transfer, when and where to transfer data, basically there are two categories of load balancing techniques named as static balancing and dynamic balancing. In a static algorithm, the information regarding load decision are decided statically during the grid setup time and remains constant during runtime. Even though the static load balancing techniques are simple to implement and has minimal runtime overhead they are not used on large scale because it does not support the balancing of load during runtime. Dynamic load balancing technique can adapt changes based on user requests profile, bandwidth, and storage capacity. It can use the runtime state information for making load balancing decisions. Here the decision for scheduling and balancing tasks is done by using centralized technique or distributive technique. In the centralized approach, one resource in the system acts as a scheduler and makes all the load balancing decision[11] where as in distributive approach all the resources are involved in decision making of load balancing which is very costly. This approach may suffer from the overhead due to multitude of information exchange between resources [12].

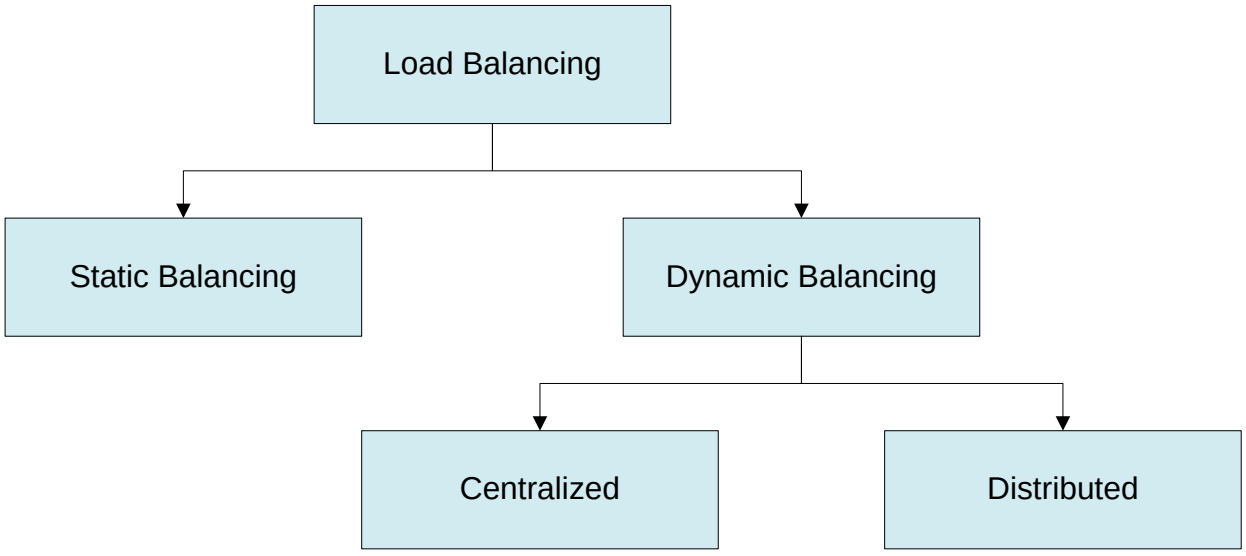

Figure 6 Load Balancing Techniques

In the previous research load balancing techniques used performance metrics like load balancing and task completion time whereas the problem of high consumption of energy still existed. J. Cao, et. al[13] used the method of intelligent agents for balancing the load in grid environment. The execution time is reduced by these agents by exchanging information with each other. Research work is carried out on power efficiency at cluster and data server level to decrease the consumption of energy. Ludwig and Maollem [14] also proposed an ant colony optimization technique for balancing load. They also presented another algorithm based on particle swarm optimization.

2.5 Power Consumption

Power is the rate at which system performs work. The SI unit of power is Watts. While energy is the sum of total work performed over a period of time. SI unit of Energy is joule which is equal to watt-hour (Wh) or kilowatt-hour (kWh). Both power and energy are represented in Equation (1) and (2).

$$P = \frac{W}{T} \tag{1}$$

Where P represents power, W is work and T is the time.

$$E = \frac{P}{T} \tag{2}$$

Where E is the energy, P is power and T is the time.

The massive amount of power is provided by the grid for execution of computational tasks whereas the problem of high consumption energy makes the grid environment unreasonable for computing heavy tasks [15, 16]. The different states of the resources can be 'on' state, 'off' state, 'idle' state, 'booting' state and 'shutting' state. For managing power, two methods are applied that is switching off unutilized parts of clusters [17-19]. The second method used for power saving is Voltage and Frequency Scaling (VFS) which adjust CPU processing speed [18]. VFS strategy can be used at CPU level by reducing the frequency of CPU either manually or dynamically named as Dynamic Voltage Frequency Scaling (DVFS) [20]. This frequency is scaled down by processor technologies such as Intel's SpeedStep [21]. According to Meisner, et. al. [22] ideal and simple energy saving method is switching off the unutilized resources but it becomes difficult in case of overloaded resources. Orgerie et al. [19] proposed a prediction algorithm which sum up the workload and switch off the resources to save energy consumption in Computational Grids. Hence to save energy the scheduler decides to allocate the tasks on the resources having minimum CPU frequency. Energy efficient techniques are applied using virtual machines and changes in physical systems. For improving energy consumption of the resources, research exists on the processor level of clusters [23] and virtualization based task scheduling mechanism [24]. In grid environment, many scheduling policies are used for efficient resource allocation. Tasks can be scheduled on priority based or on FCFS based. Gridway [25] is an adaptive Grid system that uses First come First serve (FCFS) scheduling mechanism policy and provides services to the clients. It changes according to the environment for heterogeneous resources to fulfill the requirements of Grid tasks. Even though research work on energy aware policies is done at cluster level or data center level, the problem of energy consumption in grid computing remains open.

2.6 Main Source of Power Consumption

Power Consumption in resources can be classified into two types; Static power consumption and dynamic power consumption. Static power consumption is the power consumption of devices at the low level systems. Leakage power exists in any active circuits and such leakages can be reduced by improvement of the low level design at the initial level [9]. This power depends on

transistors and processor technology and is independent of system clock rates. On the other hand dynamic power consumption mostly depends on system clock rates, input/output activities, circuit activities (fluctuation of register values, transistor's fluctuation etc.).

According to [32] the power consumption of the resource is 108 W when it is on and occupied by tasks and is 50 W when a resource is not occupied by any tasks but still on. A large proportion of power in data servers is used for cooling systems and CPUs. CPU is the main source of power consumption shown by Intel Labs [26]. This power is saved by applying power saving methods and improvement of CPU power efficiency. Hence before giving any solution to the problem, it is significant to check and understand the power consumption of the devices and resources in the data server [27]. By applying the DVFS method the power consumption of the CPU can be reduced while sustaining the ability to run programs.

2.7 Power Management

The two types of power management are: Static Power Management (SPM) includes the different methods for managing power and can be applied at the initial design and architecture of the circuit level [28]. On the other hand Dynamic Power Management (DPM) techniques can be applied at run time of the system based upon the current constraint of the resources. Our focus in this research is on DPM, consisting of different techniques for power optimization.

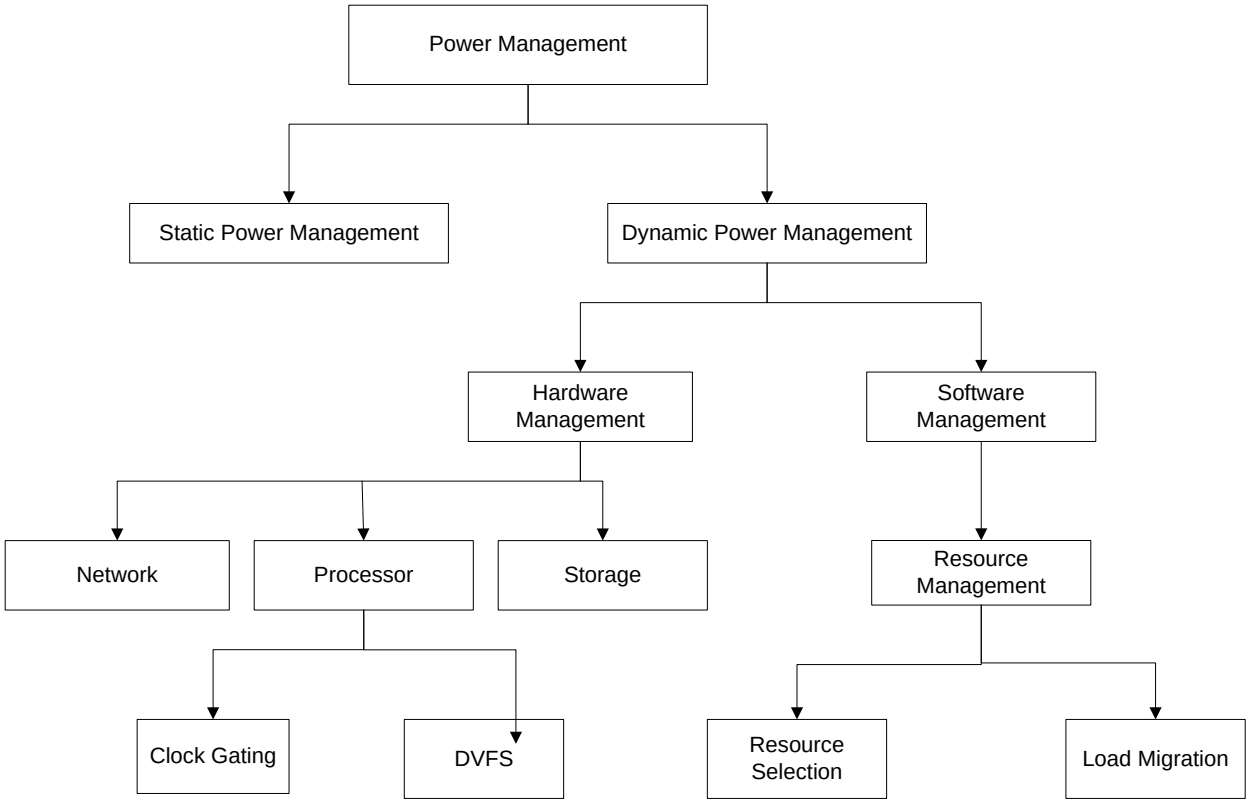

Figure 7 Power Management Classification

2.8 Dynamic Power Management

Dynamic power management can be classified into two categories that are hardware level and software level. At hardware level, DPM techniques can be divided into two kinds which are Dynamic component Deactivation (DCD) and Dynamic Performance Scaling (DPS). Both DCD and DPS can be applied at the hardware level to save energy efficiently. DCD works on the basis of clock gating method that switch off the systems that are in idle state while DPS dynamically alters the performance of the components of the system on the basis of current resource demands.

At the server level DVFS technique can be applied dynamically for reducing power. Although this technique is efficient but it still consumes 70% of power when the server is in idle position. Hence to reduce this power consumption, the idle resources must be turned off or kept into the sleep mode by using DVFS method [28]. The tasks should be adjusted to the underutilized resources and turn off the idle resources to save energy and enhance utilization.

2.9 Existing Recent Work

Load balancing is very important technique for improving response time in computational grid. Improved enhanced GridSim with deadline control (IEGDC) is a technique for balancing load based on deadline control. It is an improved version of Enhanced GridSim with deadline control (EGDC). In EGDC a dynamic distributed load balancing approach is proposed where the resources and grid scheduler is involved in balancing load. The deadline is given to the gridlets and performance of finished and unfinished gridlets is examined. The comparison is of EGDC is performed with 'without load balancing'(WLB) algorithm [29].

In IEGDC mechanism, the finished rate of the Gridlets is improved and also the resource is prevented from the overloading. For IEGDC, a new selection method is presented that considers the resource bandwidth and capacity of every resource. It reduces the response time of the Gridlets, resubmitted times, the number of finished Gridlets and unfinished Gridlets.

2.10 Gaps in the Literature

A lot of work is done for balancing load among resources in a computational grid. A computational grid consumes a lot of power and energy with the high utilization of resources. For reducing energy consumption different techniques has been showed in the literature. Even though previous work addressed many methods for balancing load and smart scheduling of tasks among resources there are no such methods that distribute tasks based on user requirement which also not violate the QoS requirement. By considering the proper QoS requirement, the existing techniques of task scheduling can be improved.

Chapter 3

3 Methodology

3.1 Outline

In this chapter, first we discussed the working architecture of our proposed work. After that we explained the sequence diagram of our proposed system and the steps for balancing load that are followed for our proposed work.

3.2 Proposed Model Architecture

Different entities/modules of the proposed system are designed to schedule the tasks efficiently and provide services to the users/clients. We have designed the system model by using Microsoft Visio. The working of the different modules is explained below:

3.2.1 *Grid Scheduler*

Grid scheduler is very important part of the grid. A grid scheduler offers computing services to the users. Each task is submitted to the Grid Scheduler which is then scheduled by its different entities according to the user's requirement. Scheduler decides the future of the job on the resources and resources have to process the request and send back the results to the scheduler.

3.2.2 *Grid user*

User submits task to Grid Scheduler by identifying features of jobs (i.e. job's length communicated in Millions of Instructions per Second (MIPS)) and excellence of service requirements (i.e. processing time and budget).

3.2.3 *Grid Information Provider*

Grid Information Provider sends the information to the scheduler about the computing capacity of each resource of the server so that it can easily compare the job and resource computing

capacity for scheduling purpose. These resources are monitored by grid during joining and leaving the grid.

3.2.4 *Load Manager*

Load Manager collects the information about the load on the each resource. After collecting power consumption of resource from the power manager and load of all the resource then scheduler will decide that where to execute the jobs.

3.2.5 *Power Manager*

Power Manager collects the current power consumption of each resource and sends information to the load manager where the load on the resources is managed accordingly.

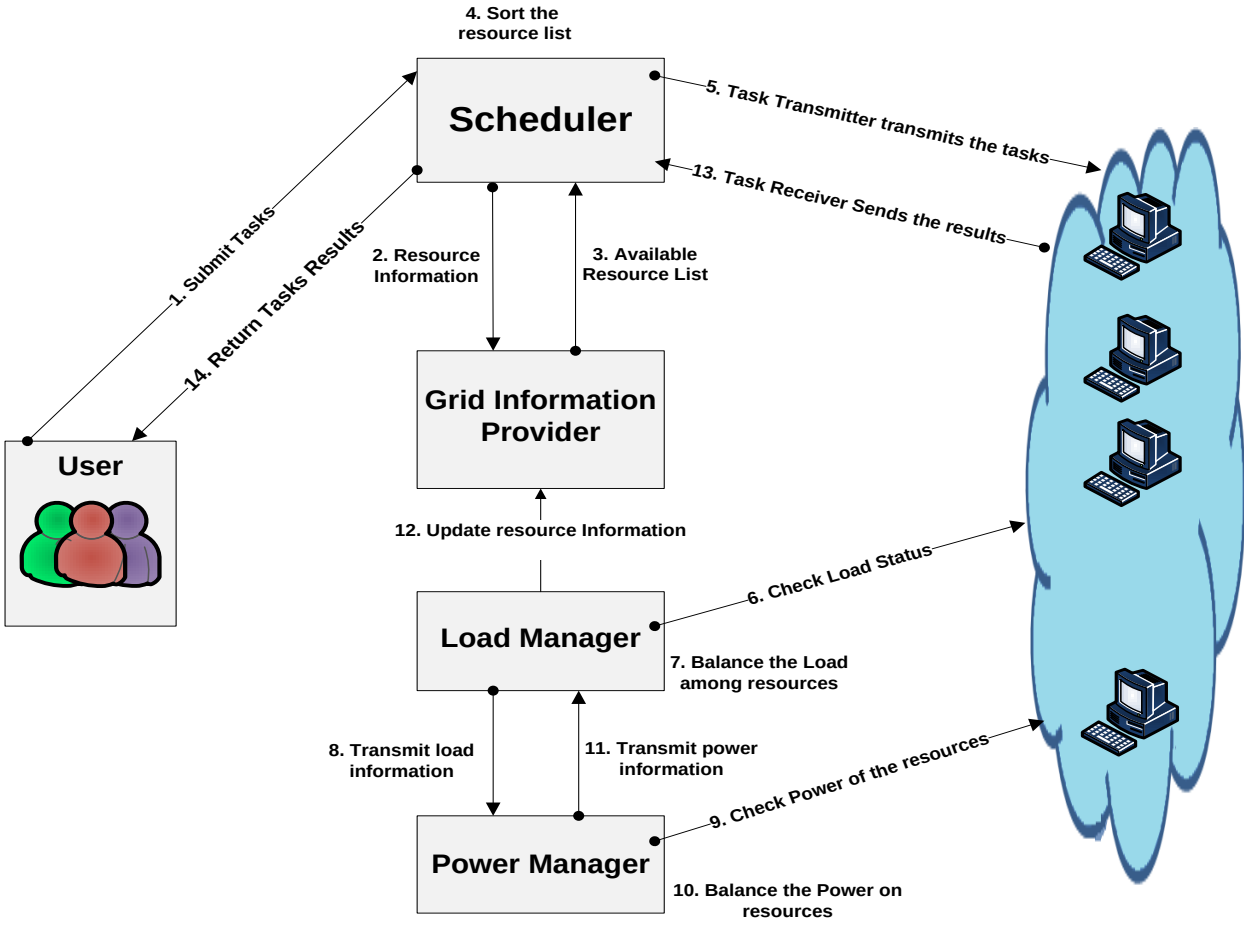

Figure 8 The Proposed working of the System Model

The sequence of interactions between components of the proposed scheduling system is shown in Figure 9 as follows:

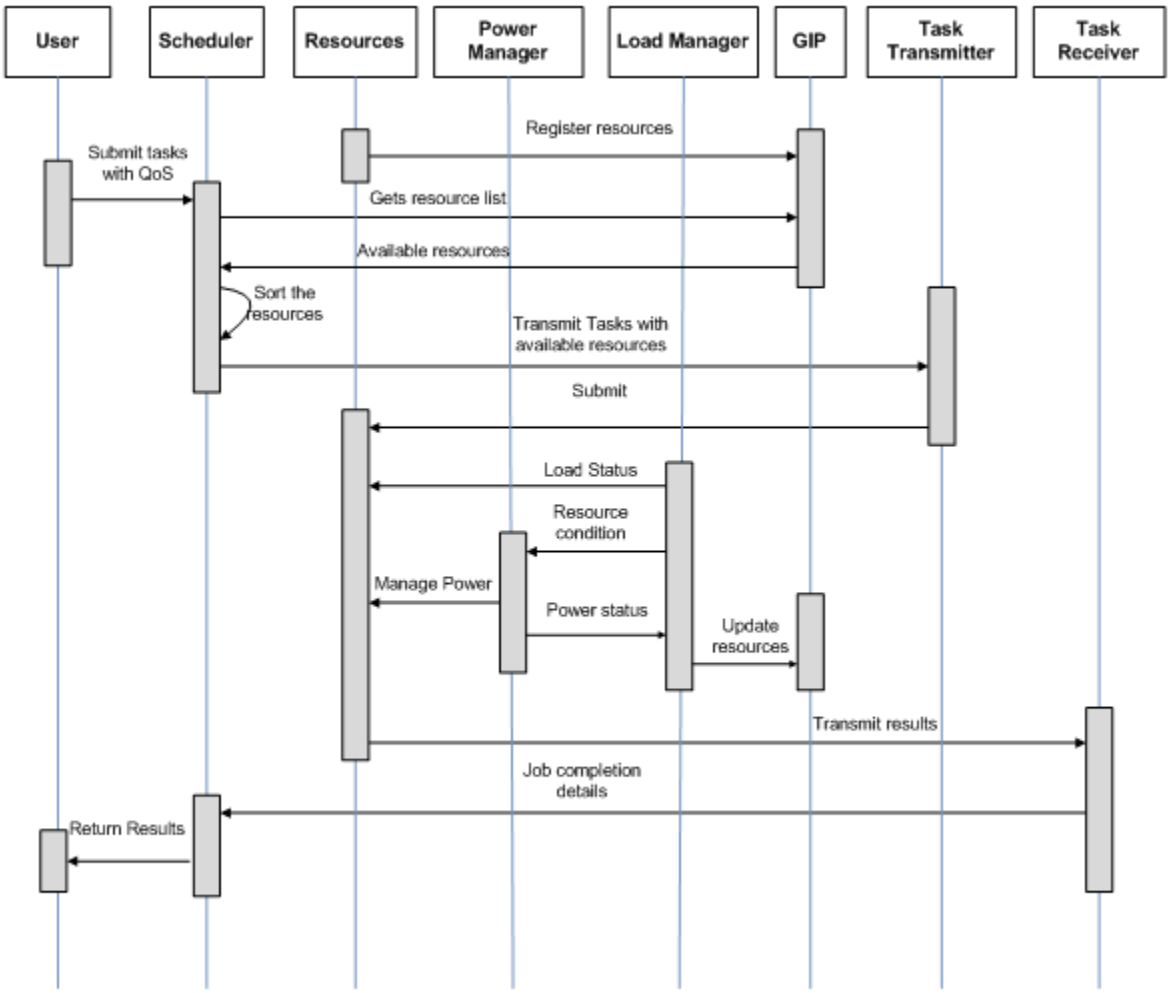

Figure 9 Sequence Diagram of the Model

1. Users submit tasks N of size M to the user to the scheduler. The tasks includes information about cost, deadline etc.
2. The resources get registered to the Grid information provider (GIP). All the resource provides their computing capacity and costs at which they offer services to the users.
3. The scheduler gets all information regarding all the available resource from GIP whenever it has tasks for execution. It sorts the available resources in descending order according to their utilization.

4. The scheduler transmits the tasks along with the available resource information to the transmitter.
5. The Task transmitter transmits the tasks to the resources.
6. The status of the resources during tasks execution is examined by the Load manager which transfers this status to the Power manager.
7. While based on load status of the resources, the power manager manages the power on/off status of the resources.
8. After finishing the executions, the resources transmit the results to the task receiver.
9. The task receiver transmits the job completion details to the Scheduler.
10. The scheduler returns the result to the scheduler.

Scheduling the tasks on resources aims to reduce the energy consumption of the resources. The tasks are migrated from the resources which have more tasks for execution and are overloaded with the tasks or the resources that have less number of tasks and hence are under loaded. For decision making which and when the tasks should be migrated, we need to follow steps as in Figure:

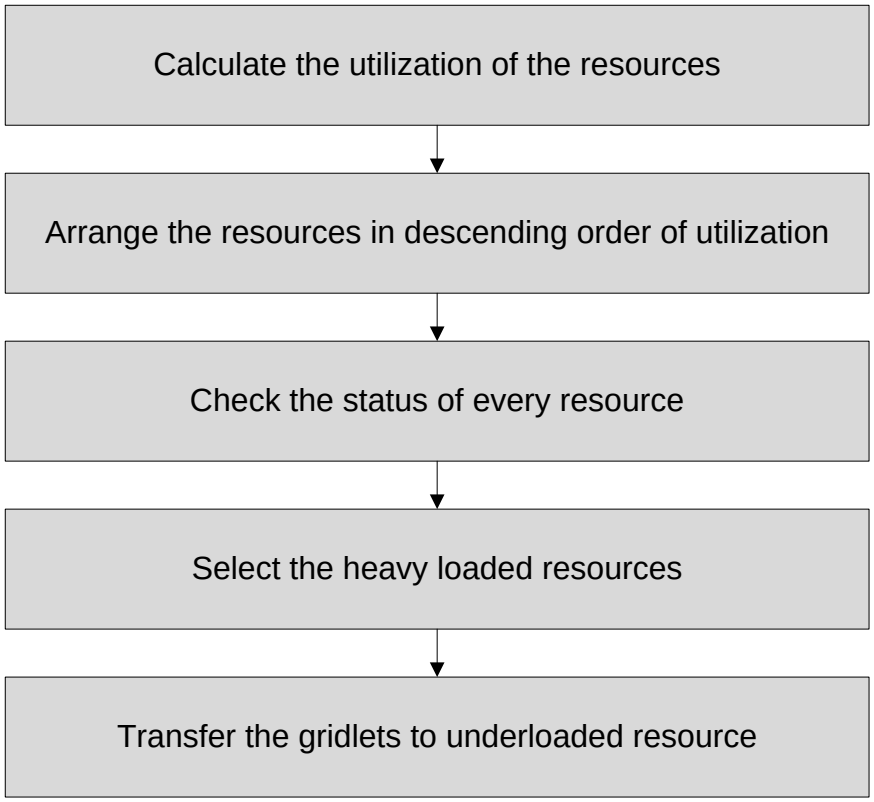

Figure 10 Steps for Balancing Policy

Initially tasks on resources is placed which has minimum load. If the resources are fully loaded, the new resources may need to be powered on. For making it convenient, we have statically chosen resources with no load and results are not affected.

Chapter 4

4 Algorithm Design

4.1 Outline

In this chapter first we discussed the existing and our proposed algorithms in detail. After that we discussed the GridSim tool used for our simulations.

4.2 Improved enhanced GridSim with deadline control (IEGDC)

IEGDC is an extension of EGDC [30], while IEGDC improves the utilization of the resources and stops the resources from overloading. For thresholds three state of resource is set that is under loaded, normal loaded and over loaded resources. The load on resource is kept normal loaded according to the specified threshold to meet the balancing requirement. The comparison of this algorithm is performed on makespan, resubmitted time, finished and unfinished Gridlets.

In IEGDC the energy consumption and cost increases due to high utilization of resources and the resources are always in state ‘on’. Valid scheduling mechanism can reduce the energy consumption meaningfully and that can also meet the QoS requirement.

4.3 Proposed Algorithms

EELBS (Energy Efficient Load Balanced Scheduling) is the proposed algorithm of our work. This algorithm extends the work done in EGDC and improves the work done in IEGDC[31]. Our work is based on QoS requirement which is able to reduce the energy cost and meet the deadline of the tasks.

For energy consumption, we have also implemented the scenario when the resources are on, off based on the work done by A. Fernández-Montes,et al. [32].

4.3.1 *The Logic of Scheduler*

To resolve the proposal, this algorithm first sorts the resources in descending order of their current utilization by the entity scheduler. The scheduler then assigns the tasks to the appropriate resources.

```
1: Input: No of Tasks
2: Out Put: Resource utilization
3: For each number of resource
4: ResourceUtil ⟵ Getutil ( )
5: Sort the resources in descending order of their utilization such that R1>R2>R3……>RN
6: Minimizeutil = Interger.max
7: Assignresource = null
8:        if ResourceUtil < Minimizeutil
9:                Assign task=resource
10:               Minimizeutil=Resourceutil
11:       Endif
12: Endloop
```

Figure 11 The Logic of Scheduler

The scheduler assigns the tasks according to the utilization of the resources.

4.3.2 The Logic of Load Manager

For load to be transferred, a double threshold policy is presented. For load transfer algorithm is used shown in figure. The double threshold that is upper and lower thresholds is assigned for the load to the resources. The total utilization of the CPU is kept in consideration among these thresholds. If utilization of a resource is less than the lower threshold, all the load is transferred from this resource to the other resource. If the workload on a CPU goes above the upper threshold, the load has to be transferred from this resource to another in order to keep the workload between the given thresholds.

```
1: Input: Tasks Assigned,
2: Output: Load Balancing
3: While (Tasks in execution)
4:        if (ResourceUtil >= upper_threshold && ResourceUtil <= lower_threshold)
5:                if (Task.UserType=TimeConstraint)
6:                        Check the state of tasks;
7:                Else
8:                        if (Task.UserType=BudgetConstraint)
9:                                Resource.Migrate Tasks;
10:                               Resource.Remove Tasks;
11:                       End if
12:               End if
13:       End if
14: End Loop
```

Figure 12 The Logic of Load Manager

We further checked QoS parameter while transferring loads from one resource to another resource that is check the task state. The task state shows that if the tasks are of time constraint type then those are not migrated and leave for execution.

4.3.3 *Task Status Logic*

For the tasks to be transferred from one resource to another, the task status must be checked. If the tasks are of the type time constraint and is in Queue or ready for execution or is in execution those tasks should not be shifted from one resource to another resource.

```
1: Input: Task_id, Resource_id,
2: Output: Tasks State
3: While all Tasks in resource
4:        if (Task_state==INQUEUE && Task_state == READY && Task_state==INEXEC)
5:                Continue ( )
6:        Else
7:                Resource.Migrate Tasks
8:                Resource.Remove Tasks
9:        End if
10: End Loop
```

Figure 13 Check Task State

4.3.4 *The Logic of Power Manager*

The power of the resources is managed according to the utilization of the resources. The power utilization of resources depends on the number and execution of tasks. The power of the resources is switched on and off when required. The utilization of the resources is kept in between thresholds and all other resources are switched off accordingly.

```
1: Input: Task_id, Resource_id
2: Output: Power on/off resources
3: For every resource in the list
4: ResourcePower ← Get Power ( )
5:          if (ResourcePower==null)
6:                    Resource.powerOff;
7:          Else
8:                    if (Tasks == INQUEUE)
9:                              Resource.powerOn;
10:                   Else
11:                             Continue;
12:                   End if
13:         End if
14: End Loop
```

Figure 14 The Logic of Power Manager

Scheduling the tasks on resources aims to reduce the energy consumption of the resources. The tasks are migrated from the resources which have more tasks for execution and are overloaded with the tasks or the resources that have less number of tasks and hence are under loaded. The idea of scheduling tasks with the energy efficiency is explained in the below example:

Example: Users submit the tasks to the scheduler by specifying its features. These features are the number of tasks, task length, deadline, cost budget and the number of MIPS required. Scheduler gets all information regarding the resource from Grid Information Provider (GIP) whenever it has tasks for execution. The Scheduler has all related information about the resource utilization. It schedules the tasks on the resources and sends the updated list of resources to the load manager. The load manager checks the runtime load of the resources and balances the resource load. The information of the balanced load is then transmitted to the Power Manager where power on the resources is managed. Hence the power and load on the resource are balanced efficiently meeting the QoS requirements.

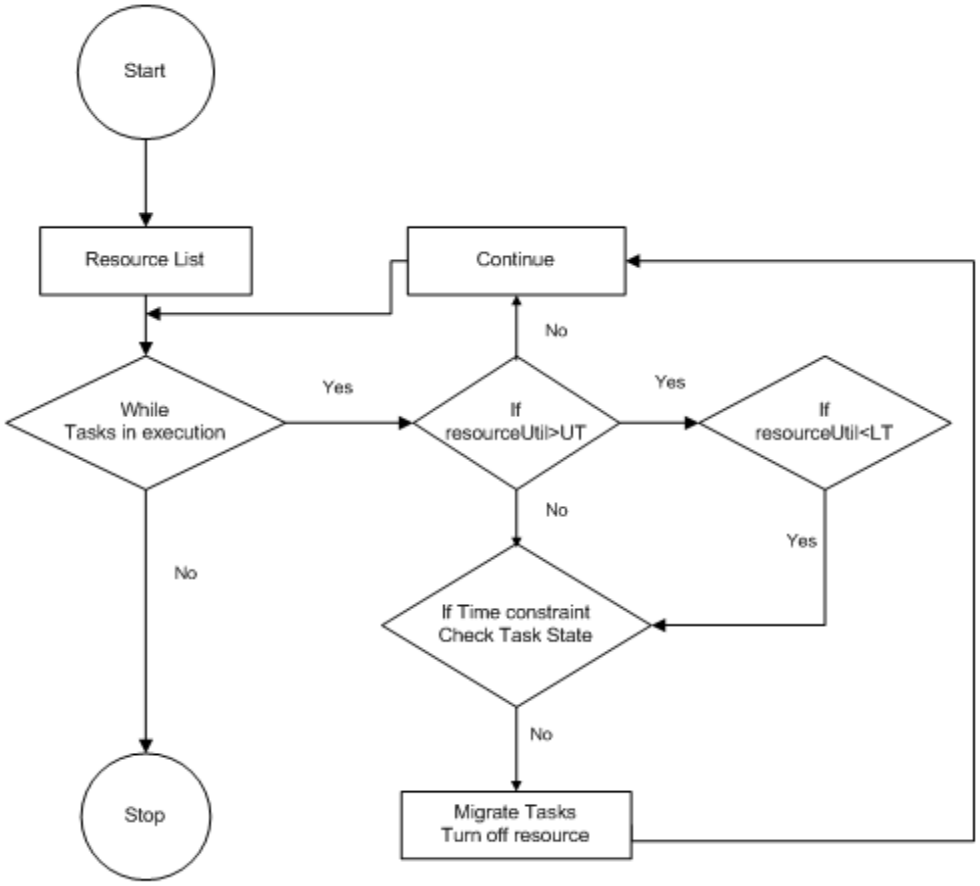

Figure 15 The Flow Chart

For load to be transferred, a double threshold policy is presented. The flow chart for the load transfer is shown in figure 7. The double threshold that is upper and lower thresholds is assigned for the load to the resources. The total utilization of the CPU is kept in consideration among these thresholds. If utilization of a resource is less than the lower threshold and the task is not time constraint the load is transferred from this resource to the other resource. If the workload on a CPU goes above the upper threshold, the load has to be transferred from this resource to another in order to keep the workload between the given thresholds. The task state shows that if the tasks are of time constraint type then those are not migrated and leave for execution. The power of the resources is managed according to the utilization of the resources. The power utilization of resources depends on the number and execution of tasks. The power of

the resources is switched on and off when required. The utilization of the resources is kept in between thresholds and all other resources are switched off accordingly.

4.4 GridSim

GridSim is a framework that provides features for simulation and modeling of a wide range of heterogeneous resources. It is used for modeling of users, applications, resource brokers, schedulers and management of multiple heterogeneous resources [29]. It also provides edges for allocating tasks/Gridlets to the resources and running their execution. These features provide a platform for evaluating the performance of scheduling algorithms and heuristics.

4.5 Design of GridSim

The layered design of GridSim is shown in figure 16. The three main components of GridSim are as follows: First the users have detailed information about the resource management policies and features of applications. Second GridSim components provide the modeling and simulation of core grid entities such as resources allocation, information services and so on. Third is the Event driven simulation engine which is responsible for conducting events and system update.

The GridSim layer Java interface and the runtime machinery, called JVM (Java Virtual Machine) is the initial layer of the tool following the next layers and ends the last layer of services. Each layer is responsible for its own functionalities. The implementation of JVM is available for single and multiprocessor systems including clusters [25]. The GridSim toolkit supplies modelling high level and important Grid modules that are resource management, and application model.

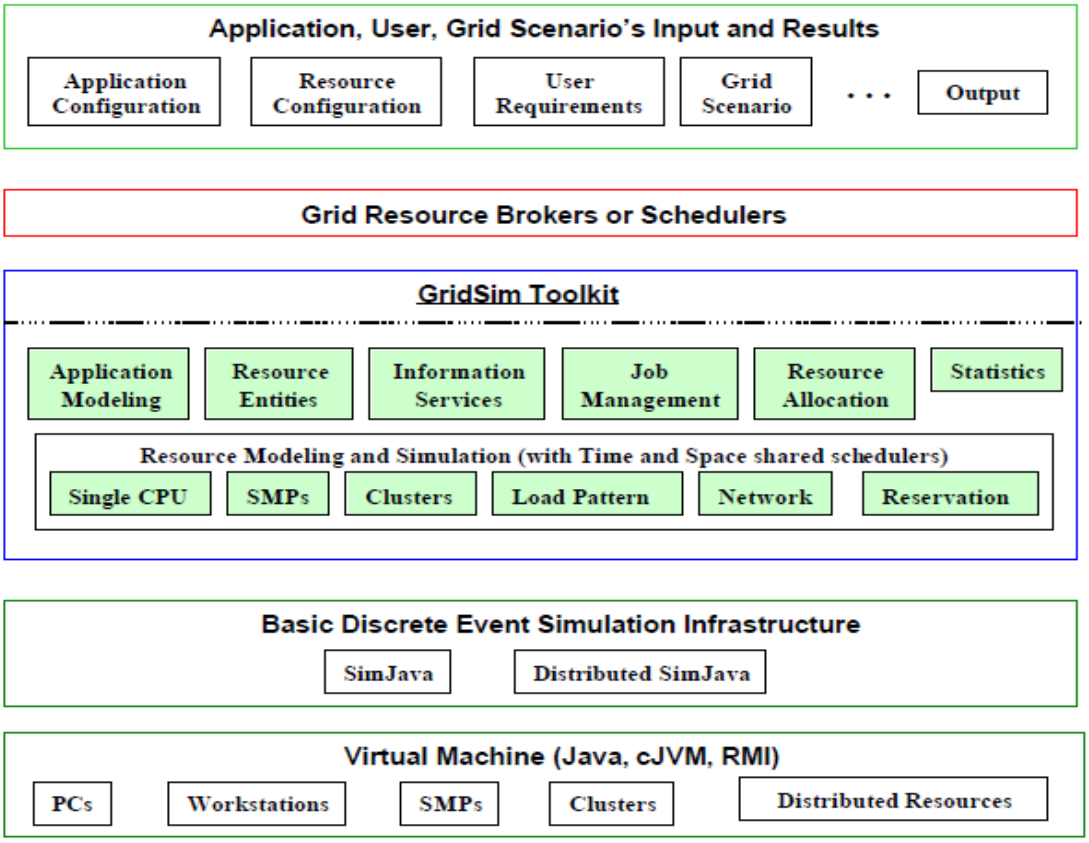

Figure 16 A modular architecture for GridSim's platform and component

4.6 GridSim Class Design

4.6.1 *GridSim*

This class represents the methods for simulation initialization, management and flow control. The GridSim environment must be primed to arrange simulation environment before making any GridSim entities at the user level.

4.6.2 *Grid Resource*

This class is used for the formation of resources. The Grid resources are made from PE objects with suitable MIPS/SPECS rating. Then these PE are combined together to form a machine. Finally, from different machines a resource can be formed.

4.6.3 *Gridlet*

In GridSim applications/tasks are composed of Gridlets. This class act as a job package that has a predefined length specified in MI. Gridlets are created by individual users for processing them on grid resources which are assigned by Scheduling entities (schedulers/resource brokers).

4.6.4 *Gridlet List*

This class can be used to keep a list of Gridlets and supports functions/methods for forming them. The Gridlet can be added, paused, moved and cancelled from the list.

4.6.5 *Grid Information Service*

This entity is used to provide information about the resources. The grid resources get registered to Grid Information Service when they enter the GridSim. All the GridSim entities such as scheduler/resource broker contact Grid Information Service for the registered resources.

4.6.6 *GridSim Shutdown*

This entity waits for the termination of all user entities and then sends signals of conclusion to all entities of GridSim.

Chapter 5

5 Simulation Results and Analysis

5.1 Outline

In this chapter the simulation results are shown for the proposed algorithms. In Section 5.2, simulation setup is discussed. Section 5.3 explains the parameters taken to conduct simulation. Section 5.4 describes the cases of simulations, and for each case we explained its scenario and discussed the results of the simulations.

5.2 Simulation Setup

Simulation of the proposed algorithm is performed by using the GridSim simulator. The results of the simulations are compared with the existing method IEGDC [31] . We have created a scenario based on the work carried out by Montes et al. [32] for checking the ‘on’ and ‘off’ power status of the resources. In our work, we conducted simulations and compared them with our proposed work by keeping the resources ‘on’ and ‘off’. The comparison of the proposed algorithm is based on finished Gridlets, unfinished Gridlets, resubmitted time, response time, cost, energy consumed and energy saved. The response time is the total time of the simulation that is measured from the time first Gridlet is sent to the Grid till the last Gridlet. The finished Gridlets is the number of Gridlets successfully completed within deadline. We also have selected the number of unfinished Gridlets as standard performance criteria. These Gridlets may not be completed in their deadline or overloading of the resources or due to the offline features of resources. For this reason resubmitted time is another standard for our simulation. Two different cases are considered for our simulation:

Case: 1 Simulation with constant number of PEs

Case: 2 Simulation with constant number of Gridlets

Simulations are performed by taking 50 resources in the server. Every resource has 2 machines which has 5 PEs. So the grid system has total 500 PEs. The rating of every PEs is between 1-5

MIPS. The Gridlet length is expressed in PEs and is between 1-5 MIs. The deadline for every Gridlet is in the range of 1-4s. In our simulation, we have taken heterogeneous resources that are connected by communication channels. For two resources to be connected, the network bandwidth changes from 0.5-10Mbps. With the first router, user entity is connected having a connection of 1 Mbps. All the other entities are connected with the second router with different connections that are 0.5-10Mbps. We have selected the resource threshold on the basis of [31]. We selected the lower threshold as 0.6 an upper threshold as 0.8.

No. of Resources (PE)	**500**
PE rating (MIPS)	1-5
No. of Gridlets	2000
Gridlet length (MI)	1-5
Gridlet deadline (seconds)	1-4
Network Bandwidth (mbps)	0.5-10
User Type	2

Figure 17 Classifications of Parameters

Following figures [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, and 31] shows the results of our proposed algorithm and its comparison with the existing algorithm.

5.3 Parameters

The following parameters are used to conduct simulations and compared our proposed scenario with the present policy associated to the scheduling mechanism.

5.3.1 Gridlets Performance

The first parameter is the gridlets performance observed on finished gridlets, unfinished gridlets, resubmitted time, and response time. The finished gridlets is the number of gridlets successfully completed within a given deadline. The Unfinished gridlets are the gridlets that were not successfully executed in a given deadline. The resubmitted time is for unfinished gridlets which may not be completed in their deadline or overloading of the resources or due to the offline features of resources. These tasks are resubmitted again for execution. The response time is the total time of the simulation that is measured from the time first gridlet sent for execution till the last gridlet.

Resource capacity r_c is the speed of the resource; $NumPE$ is the number of processing entities. And Rating is expressed in spec or MIPS.

$$r_c = \sum_{k=1}^{n} NumPE_k \times RatingPE_k \quad (3)$$

5.3.2 *Cost*

The third parameter cost per unit is calculated for the resources used by the users for their tasks execution. This cost is measured in Grid dollars (G$). It is calculated by using following formula.

$$Cost = \sum_{k=1}^{n} Ci \quad (4)$$

5.3.3 *Energy Consumption*

The third parameter is energy consumption of the resources caused due to the consumption of resources for executing tasks. The below model is applied for finding the energy consumption of resources [35].

$$E_i = (P_{max} - P_{min}) \times U_i + P_{min} \quad (5)$$

where P_{max} is the power consumption at the peak load or when a resource is occupied by tasks, and P_{min} is the minimum power consumption of the resource in the active mode. In our case we have taken 100W as a maximum value when a resource is fully utilized and 1W as a minimum value when it is not occupied by any tasks.

5.4 Case 1

5.4.1 *Scenario*

In this case, the simulations are shown by assuming PEs constant and Gridlets to change. For constant PEs, we have chosen 500 PEs for our simulations. We have performed a comparison of finished Gridlets, unfinished Gridlets, resubmitted time of the Gridlets, and response time of the Gridlets. Further we have taken Cost of PEs, total energy consumed by the resources, total energy saved by the resources as our proposed performance metric to conduct simulations. We have checked the energy consumption of resources when they are switched ‘on’ and ‘off’. This shows that the proposed algorithm performs better than the existing techniques.

Results and Discussion

Figure [18] shows the simulation results of finished Gridlets vs total number of Gridlets for the existing method IEGDC [31] and proposed EELB.

In fig [18, 19] the simulations were performed by changing the number of Gridlets starting from 500 and ending at 2,000 with a step of 500 while the number of PEs is kept constant. Fig [18] presents finished Gridlets vs total number of Gridlets for the existing method compared with the proposed method (EELB). The number of finished Gridlets increases by increasing the total

number of Gridlets because for time constraint users, the Gridlets are not migrated and remained for execution.

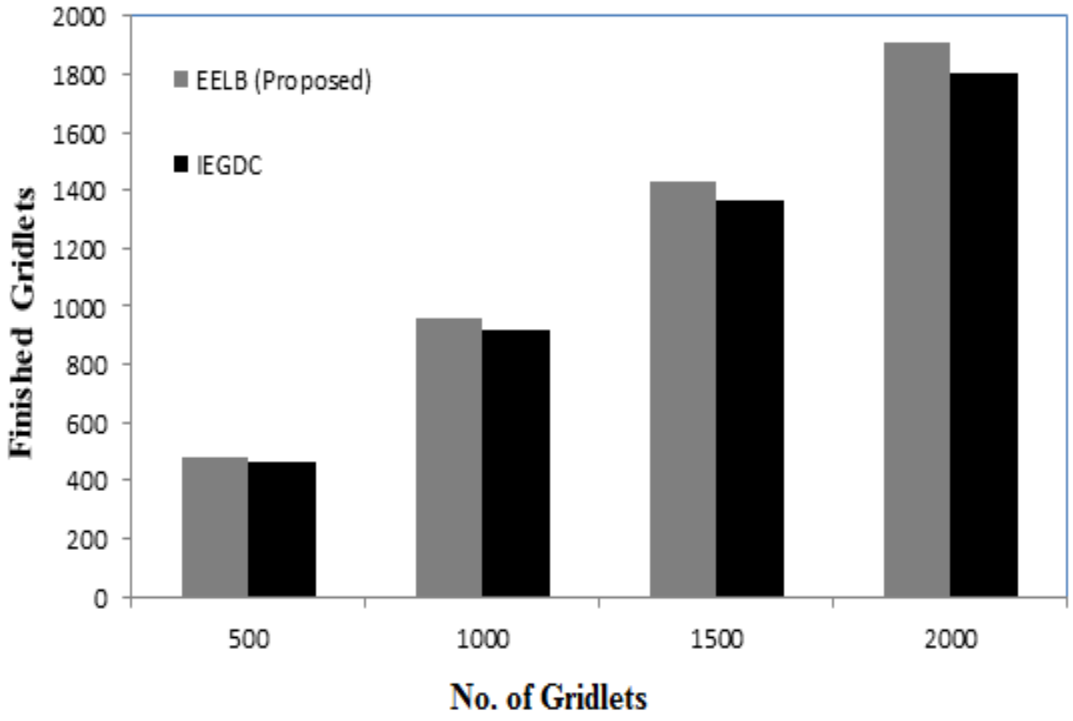

Figure 18 the Finished Gridlets

From fig [18] it is clear that EELB performs better than the existing methods.

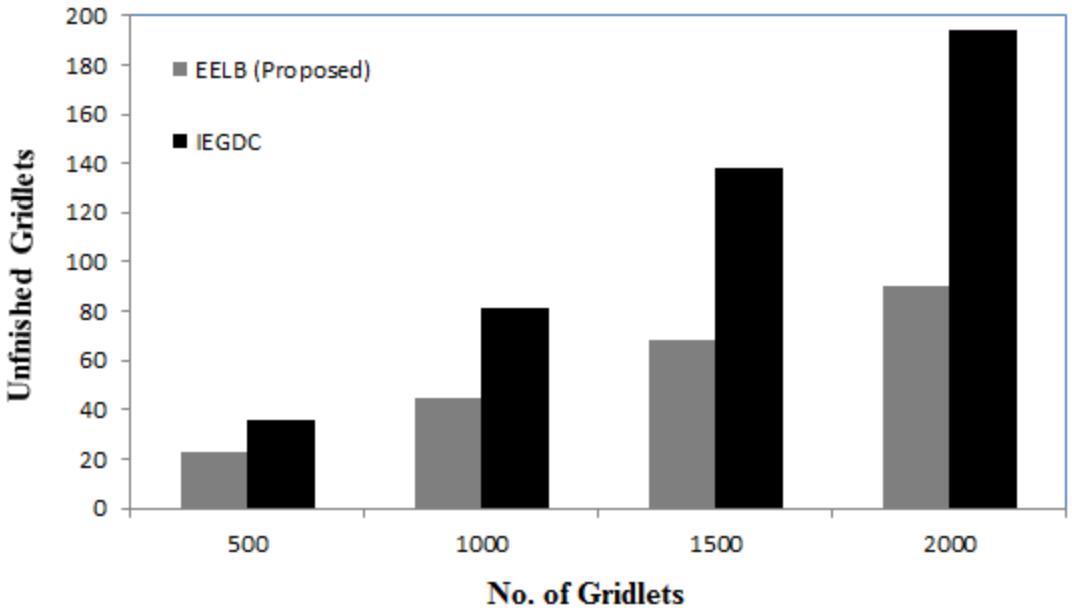

Figure 19 The Finished Gridlets

Fig [19] presents the number of unfinished Gridlets vs the total number of Gridlets for the proposed method compared with the existing method. This shows that by increasing the number of Gridlets, the unfinished Gridlets in case of EELB are always found to be less than the IEGDC method.

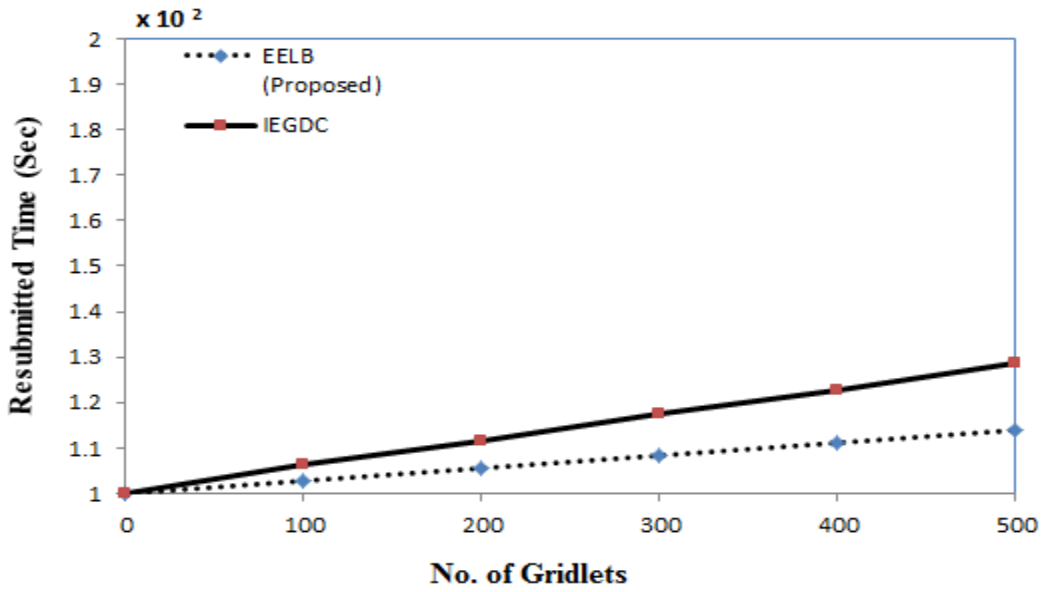

Figure 20 The Resubmitted Time

Fig [20] presents the resubmitted time of the Gridlets that were rescheduled for execution. The simulations were performed for 500 Gridlets that were left unfinished when we provided Gridlets starting from 500 and ending at 2,000 with a step of 500 while the number of PEs were kept constant. The rescheduled Gridlets started from 100 and ended at 500 with a step of 100 whereas number of PEs was kept constant. This shows that the resubmitted time of EELB is less than IEGDC by increasing the number of Gridlets.

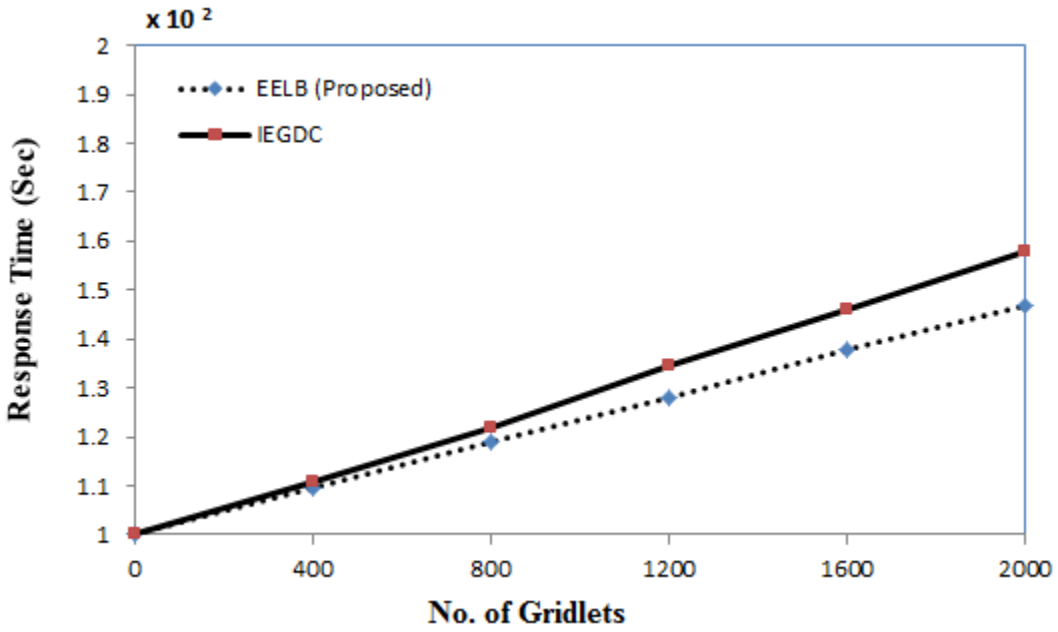

Figure 21 The Response Time

Figure [21] represents the response time of the Gridlet vs total number of Gridlets and compared with IEGDC. The Gridlets started from 400 and ending at 2,000 with a step of 400 while the number of PEs was kept constant.

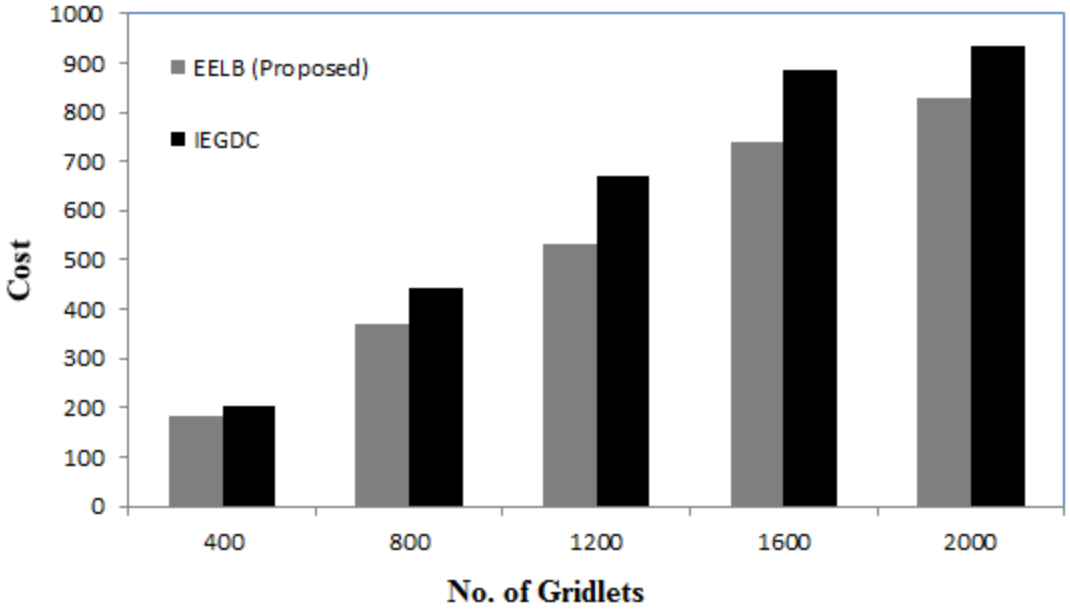

Figure 22 The Total Cost

We have added cost parameter to perform the simulations. Fig [22] represents the cost vs number of gridlets. This shows that the cost per gridlets increases as we increase the number of gridlets. The figure shows that the cost of EELB is less than IEGDC as they consume much time in execution of tasks.

For the energy consumption, we have performed the simulations by switching off the resources and switching them on according to the status of gridlets. Fig [23, 24] shows the energy consumed and saved during the execution of gridlets. Initially the consumption of energy with 500 gridlets in IEGDC is 7.86 kWh higher than EELB. The method IEGDC without energy consumes 40.31kWh by providing 500 gridlets keeping the number of PEs constant at 500. The proposed policy leads to 7.86 less energy consumption than IEGDC policy. This is because the proposed scheme uses the energy aware policy and maintaining QoS requirement while keeping the same threshold level as in IEGDC. Lastly providing 2000 gridlets with the same number of PEs, the energy consumption in IEGDC is 42.78kWh and is reduced by 10.48kWh in case of proposed EELB method. As per our results shown, it can be concluded that the proposed method performs better than the existing method in terms of the performance of gridlets, cost and consumption of energy.

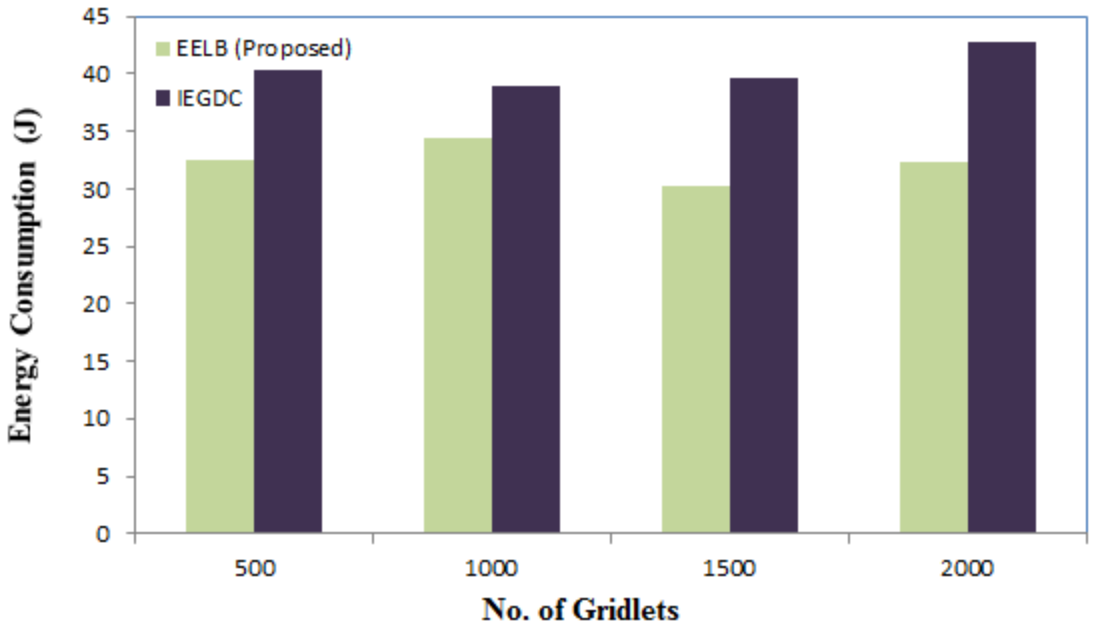

Figure 23 The Energy Consumed

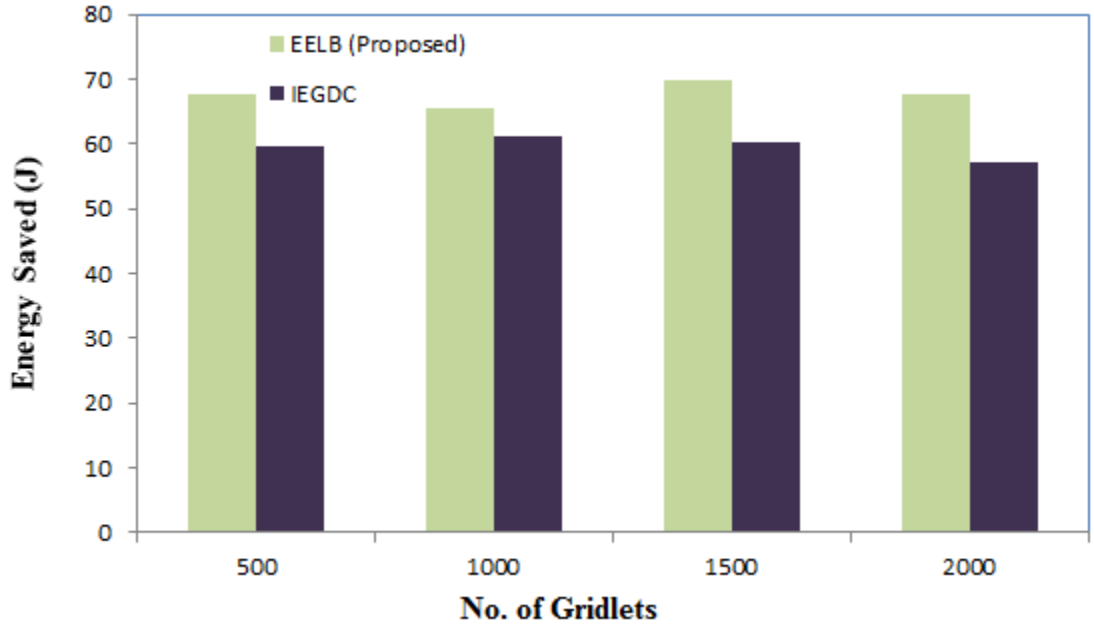

Figure 24 The Energy Saved

5.5 Case 2

5.5.1 *Scenario*

In this case, the simulations are shown by assuming Gridlets constant and PEs to change. For constant Gridlets, we have chosen 1000 Gridlets for our simulations. For constant Gridlets again we have performed a comparison of finished Gridlets, unfinished Gridlets, resubmitted time of the Gridlets, and response time of the Gridlets. Further we have taken Cost of PEs, total energy consumed by the resources, total energy saved by the resources as our proposed performance metric to conduct simulations. Further we have added the same scenario as in case of constant PEs. This scenario is based on [32] for the resources that is switching ‘on’ and ‘off’ the resources. This shows that the proposed algorithm with constant Gridlets performs better than the existing techniques.

Results and Discussion

Fig [25, 26] shows the results of finished Gridlets vs No. of PEs and the results of unfinished Gridlets vs No. of PEs.

The simulations were performed by changing the number of PEs from starting from 50 and ending till 500 with a gap of 50 while the number of Gridlets was kept constant. This shows that the number of finished Gridlets increases when we increase the number of PEs while the number of unfinished gets decreased by increasing the number of PEs. The results were compared with the exiting method. This shows the finished rate of the Gridlets is higher in EELB than the existing method IEGDC as for time constraint Gridlets, we created a scenario that is the time gridlet in execution or ready for execution was not be migrated.

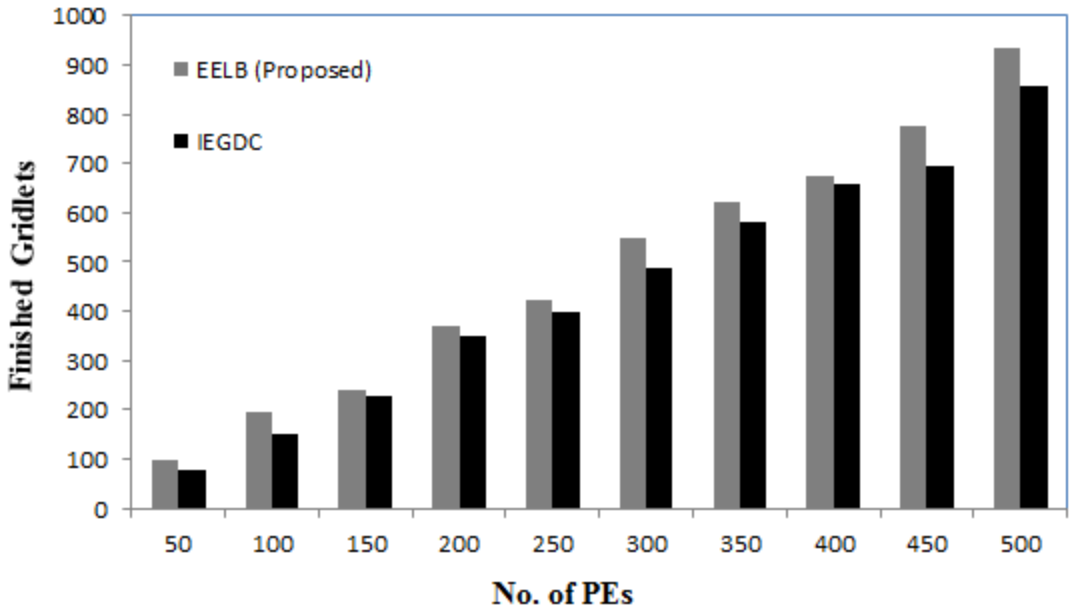

Figure 25 The Finished Gridlets

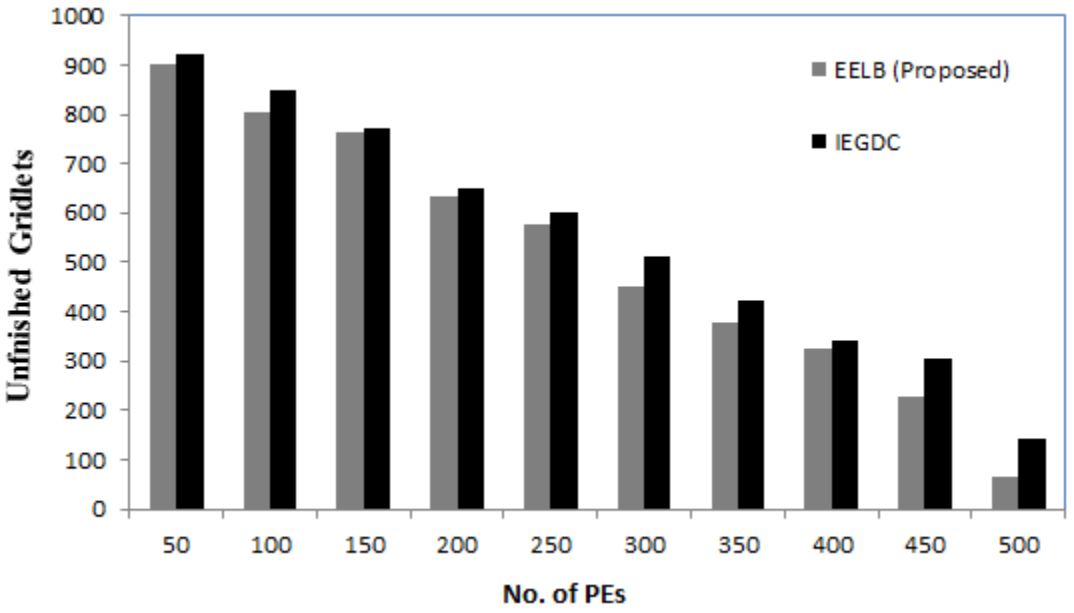

Figure 26 The Unfinished Gridlets

Fig [27] shows the resubmitted time for the unfinished Gridlets that were rescheduled for execution. In this case the simulations were performed for more number of the Gridlets with the PEs starting from 50 and ending till 500 with a step of 50. The simulations were performed for all Gridlets as with 50 number of PEs the Gridlets were remain Unfinished. The resubmitted time

of the EELB is compared with the resubmitted time of IEGDC. This shows that as Unfinished Gridlets rate is higher in IEGDC so the resubmitted time is higher than EELB.

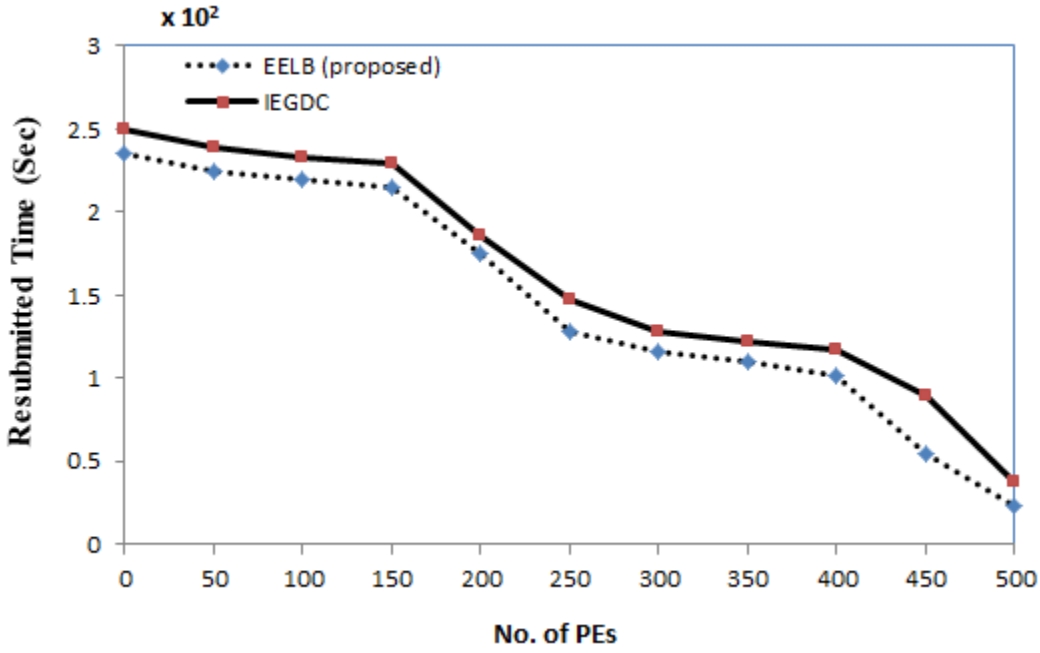

Figure 27 The Resubmitted Times

Fig [28] represents the response time of the Gridlets that were kept constant. The results are compared with the existing method IEGDC. Initially the response time of the Gridlets was low because of less PEs and it increased by increasing the number of PEs.

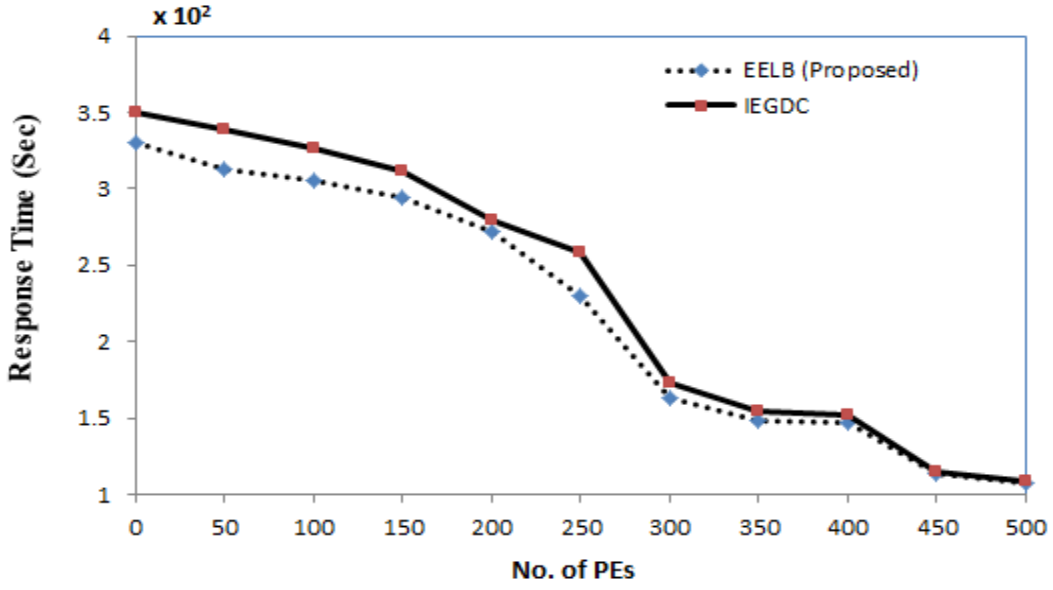

Figure 28 The Response Time

Fig [29] represents the cost versus number of PEs by varying number of PEs starting from 50 and ending toll 500 with a step of 50. The figure shows that the cost of IEGDC is more than the proposed EELB because no energy technique is applied in IEGDC to the resources which results in consumption of much time and energy in execution of Gridlets.

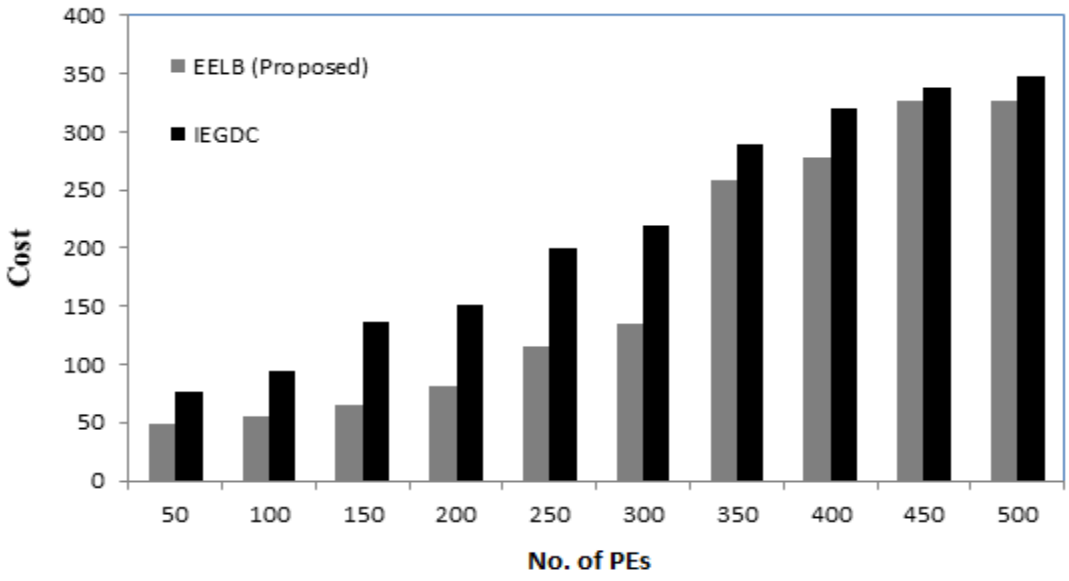

Figure 29 The Total Cost

Fig [30, 31] shows the amount of energy consumed and saved by varying number of PEs and keeping gridlets constant. It shows that that IEGDC consumes 17.71 kWh with the 50 PEs while EELB reduce this energy consumption to 14.6 kWh with the same number of PEs. The energy consumption of the PEs increases as we increased the number of PEs with the constant number of gridlets. By increasing the PEs till 500, IEGDC has consumed 23.66 kWh than EELB. Similarly with maximum number of PEs that is 500PEs, we have saved 23.95 kWh of the energy. This shows that applying the proposed method much amount of energy is saved by EELB than the IEGDC.

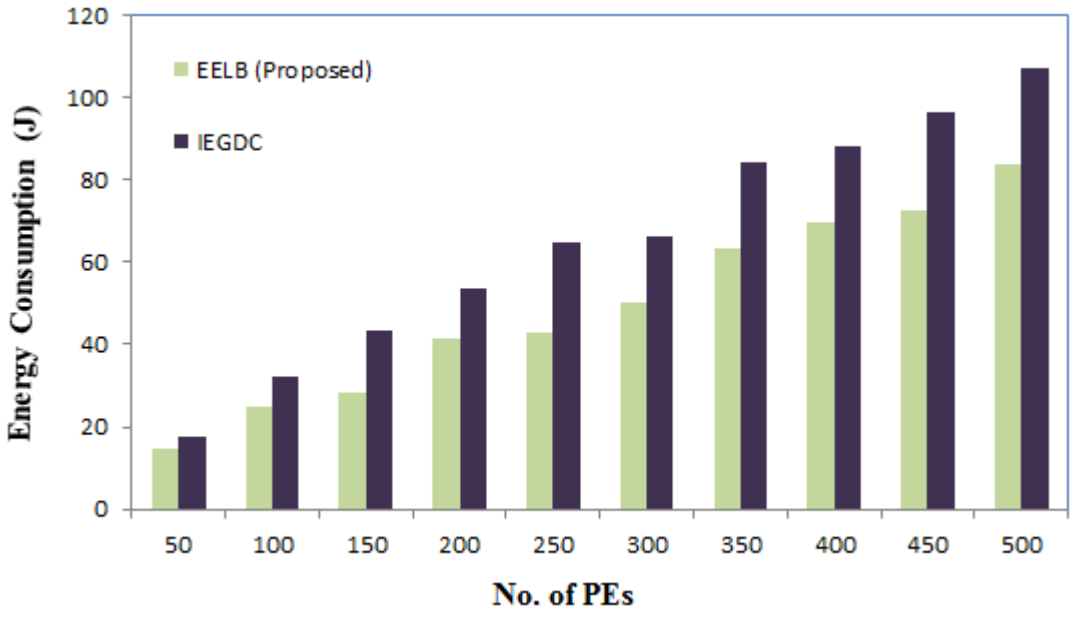

Figure 30 The Energy Consumed

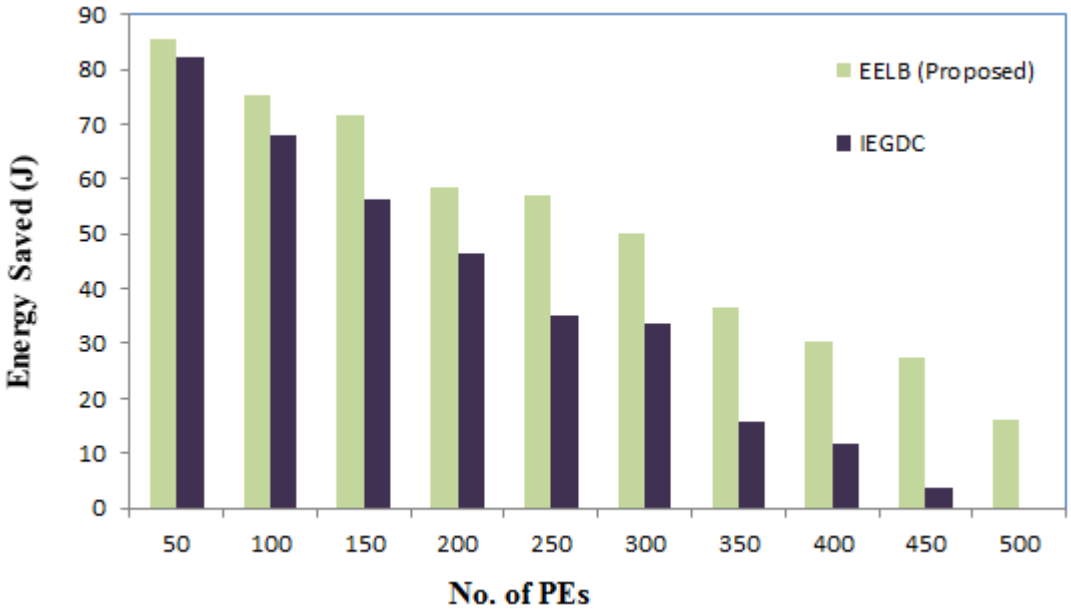

Figure 31 The Energy Saved

As per our results discussed above, we have surely improved response time, number of finished Gridlets by varying size of Gridlets and PEs for carrying out different simulations and by giving deadlines to the Gridlets. Although the finished rate of the Gridlets is near to the existing method but in contrast we have reduced the cost, energy consumption of the resources. From the results it is shown that the finished rate of gridlets is improved by 28% than IEGDC which result in high response time of the gridlets and energy consumption is reduced 38% than the existing IEGDC. The costs the resources are reduce by 35% of without energy aware algorithm. Load balancing is important factor to consider while reducing the energy consumption. If we consider energy efficiency only than we may violate the QoS and the response time of the tasks. Similarly if we consider the load balancing only, the cost as well as energy consumption increases. Hence our algorithm shows the better results for energy efficient scheduling.

Chapter 5

6 Conclusion and Future Line

6.1 Outline

This chapter concludes the discussion of all chapters and draws the future line of work for researchers.

6.2 Conclusion

In the past chapters, we showed models and algorithm for efficient scheduling of tasks which allocate resources by considering users QoS requirement and migration of tasks based on the given thresholds. We compared proposed algorithm with the existing method and we observed the proposed algorithms which consider energy efficiency and user QoS requirement for scheduling decisions perform better than the existing one. Our proposed algorithm not only maintains the load balancing but also effectively saves energy consumption of the resources while maintaining the QoS requirement. EELB produces better results for implementing job scheduling on the servers, but also reduces cost by minimizing the resources power.

6.3 Future Line

This section gives a brief direction of work for the new researchers based on the current implementations.

For future line of direction, firstly the proposed algorithm can be explained with fault tolerance strategy. The implementation cost and response time of the tasks may get increased if any failure of load migration occurs in any computing server.

Secondly we propose to improve the network energy consumption. By using correct model of energy for the network will help the proposed algorithm to reduce the cost of energy for network. More energy models for model devices should be implemented to reduce the energy

consumption in Grid. Further a parameter throughput can be added to the scenario to check the better performance of the resources.

Lastly, we designed the results in simulations environment. It's better to have results compilation using our algorithm in real computational grid environment so that we can intellect the real time effects of computational grid environment using our proposed algorithm.

7 References

[1] **M. Baker, R. Buyya, and D. Laforenza, "The Grid: International efforts in global computing," in *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, 2000.**

[2] **Y. H. Lee, S. Leu, and R. S. Chang, "Improving job scheduling algorithms in a grid environment," *Future Generation Computer Systems-the International Journal of Grid Computing and Escience*, vol. 27, pp. 991-998, Oct 2011.**

[3] **T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," *Software Engineering, IEEE Transactions on*, vol. 14, pp. 141-154, 1988.**

[4] **M. J. Zaki, W. Li, and S. Parthasarathy, "Customized dynamic load balancing for a network of workstations," in *High Performance Distributed Computing, 1996., Proceedings of 5th IEEE International Symposium on*, 1996, pp. 282-291.**

[5] **S. Goswami and A. De Sarkar, "A Comparative Study of Load Balancing Algorithms in Computational Grid Environment," in *Computational Intelligence, Modelling and Simulation (CIMSim), 2013 Fifth International Conference on*, 2013, pp. 99-104.**

[6] **J. G. Koomey, "Worldwide electricity used in data centers," *Environmental Research Letters*, vol. 3, p. 034008, 2008.**

[7] **G. Ren, E. Tune, T. Moseley, Y. X. Shi, S. Rus, and R. Hundt, "Google-Wide Profiling: A Continuous Profiling Infrastructure for Data Centers," *Ieee Micro*, vol. 30, pp. 65-78, Jul-Aug 2010.**

[8] **J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, 2011.**

[9] **V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Computing Surveys (CSUR)*, vol. 37, pp. 195-237, 2005.**

[10] **D. Andresen, T. Yang, V. Holmedahl, and O. H. Ibarra, "Sweb: Towards a scalable world wide web server on multicomputers," in *Parallel Processing Symposium, 1996., Proceedings of IPPS'96, The 10th International*, 1996, pp. 850-856.**

[11] **K. Soni and K. Mishra, "An analysis of various job scheduling strategies in grid computing," in *2010 2nd International Conference on Signal Processing Systems*, 2010.**

[12] **R. Rajavel, "De-Centralized Load Balancing for the Computational Grid environment," in *Communication and Computational Intelligence (INCOCCI), 2010 International Conference on*, 2010, pp. 419-424.**

[13] **J. Cao, D. P. Spooner, S. A. Jarvis, and G. R. Nudd, "Grid load balancing using intelligent agents," *Future generation computer systems*, vol. 21, pp. 135-149, 2005.**

[14] **S. A. Ludwig and A. Moallem, "Swarm intelligence approaches for grid load balancing," *Journal of Grid Computing*, vol. 9, pp. 279-301, 2011.**

[15] **T. M. Lynar and R. D. Herbert, "Allocating grid resources for speed and energy conservation," in *Proceedings of 6th International Conference on Information Technology and Applications*, 2009, pp. 55-60.**

[16] **T. M. Lynar, R. D. Herbert, and W. J. Chivers, "A grid resource allocation mechanism for heterogeneous e-waste computers," in *Proceedings of the Seventh***

Australasian Symposium on Grid Computing and e-Research-Volume 99, 2009, pp. 69-76.

[17] E. Laure, H. Stockinger, and K. Stockinger, "Performance engineering in data Grids," *Concurrency and Computation: Practice and Experience*, vol. 17, pp. 171-191, 2005.

[18] T. M. Lynar, R. D. Herbert, S. Chivers, and W. J. Chivers, "Resource allocation to conserve energy in distributed computing," *International Journal of Grid and Utility Computing*, vol. 2, pp. 1-10, 2011.

[19] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas, "Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems," in *Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE International Conference on*, 2008, pp. 171-178.

[20] J. Kolodziej, S. U. Khan, and F. Xhafa, "Genetic algorithms for energy-aware scheduling in computational grids," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on*, 2011, pp. 17-24.

[21] B. Khargharia, S. Hariri, and M. S. Yousif, "Autonomic power and performance management for computing systems," *Cluster computing*, vol. 11, pp. 167-181, 2008.

[22] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: eliminating server idle power," *ACM SIGARCH Computer Architecture News*, vol. 37, pp. 205-216, 2009.

[23] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *ACM SIGMETRICS Performance Evaluation Review*, 2005, pp. 303-314.

[24] A. Verma, P. Ahuja, and A. Neogi, "Power-aware dynamic placement of hpc applications," in *Proceedings of the 22nd annual international conference on Supercomputing*, 2008, pp. 175-184.

[25] E. Huedo, R. S. Montero, and I. M. Llorente, "A framework for adaptive execution in grids," *Software: Practice and Experience*, vol. 34, pp. 631-651, 2004.

[26] L. Minas and B. Ellison, *Energy efficiency for information technology: How to reduce power consumption in servers and data centers*: Intel Press, 2009.

[27] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, 2007, pp. 13-23.

[28] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," in *Proceedings of the 32nd annual ACM/IEEE Design Automation Conference*, 1995, pp. 242-247.

[29] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience*, vol. 14, pp. 1175-1220, 2002.

[30] Y. Hao, G. Liu, and N. Wen, "An enhanced load balancing mechanism based on deadline control on GridSim," *Future Generation Computer Systems*, vol. 28, pp. 657-665, 2012.

[31] D. K. Patel and C. Tripathy, "An improved approach for load balancing among heterogeneous resources in computational grids," *Engineering with Computers*, pp. 1-15, 2014.

[32] A. Fernández-Montes, L. Gonzalez-Abril, J. A. Ortega, and L. Lefèvre, "Smart scheduling for saving energy in grid computing," *Expert Systems with Applications*, vol. 39, pp. 9443-9450, 2012.

[33] F. Howell and R. McNab, "SimJava: A discrete event simulation library for java," ***Simulation Series,*** **vol. 30, pp. 51-56, 1998.**

[34] F. Dong and S. G. Akl, "Scheduling algorithms for grid computing: State of the art and open problems," Technical report2006.

[35] G. L. Valentini, S. U. Khan, and P. Bouvry, "Energy-efficient resource utilization in cloud computing," ***Large Scale Network-centric Computing Systems, John Wiley & Sons, Hoboken, NJ, USA,*** **2013.**