An Energy Efficient Scheduling Approach for Grid

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

Master of Engineering
in
Software Engineering

Submitted By
Nidhi Jain
(801231018)

Under the supervision of:
Dr. Inderveer Chana
Associate Professor
CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA 147004

June 2014

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*An Energy Efficient Scheduling Approach for Grid*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Inderveer Chana* and refers other researchers work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Nidhi Jain)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

14/07/14

(Dr. Inderveer Chana)
Associate Professor
CSED, Thapar University
Patiala

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University,Patiala

14/7

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

i

Acknowledgement

First of all, I am thankful to God for his blessings and showing me the right direction. With His mercy, it has been made possible for me to reach so far. I wish to express my deep gratitude to Dr. Inderveer Chana, Associate Professor, Computer Science and Engineering Department for providing her immense help, guidance, simulating suggestions and encouragement all the time. She always provided a motivating and enthusiastic atmosphere to work with; it was a great pleasure to do this thesis under her supervision.

I am also thankful to Dr. Deepak Garg, Head, Computer Science and Engineering Department and Ms. Damandeep Kaur, P.G Coordinator for their kind help and cooperation. I express my gratitude to all the staff members of Computer Science and Engineering Department for providing seminars and encouraging towards research work.

I extend my thanks to Mrs. Nidhi Jain Kansal, Ms. Tarandeep and Mr. Sukhpal Singh for sharing their expertise and time to help me accomplish this work.
I want to express my appreciation to every person who contributed with either inspirational or actual work to this thesis. Last but not the least I am highly grateful to all my family members for their inspiration and ever encouraging moral support, which enables me to pursue my studies.

Nidhi Jain

Abstract

Grid Computing computes the large-scale and complex problems in science, engineering and commerce by uniting the power of disseminated resources dynamically based on their availability, potentiality, makespan, cost and users Quality of Service (QoS) requirements. In Grid computing, complex and large problems are divided into small problems and distributed over the resources. Thus, grid computing increases the computational power through efficient resource utilization.

Grid Computing faces many challenges like resource management, security, energy-efficiency, Load balancing etc. In Grid Computing, it is very difficult to map the resources with tasks efficiently due to fluctuation in users requirements and to achieve better performance within budget and time. The energy consumption of this large scale distributed system is also of important concern. Reducing energy consumption increases the execution time of a task on a processing element; however, the overall energy consumption may decrease but it decreases the performance as well. Thus simultaneous optimization of energy and performance in Grid Computing is a big challenge. This challenge can be addressed by efficient management and scheduling of tasks and resources to reduce energy consumption without degrading the performance.

In this thesis, energy and performance aware layered Grid architecture has been proposed for the implementation of Energy-efficient and High Performance(EEHP) algorithm. This architecture contains three layers in which, the first layer, a Grid Portal has been designed and presented for the submission of tasks, the second layer, a Provisioning Manager takes care of performance and energy factors by analyzing the submitted tasks and resources and the third layer, Scheduler maps the tasks to resources according to the proposed EEHP algorithm. This algorithm attempts to map more complex and dependent tasks to more energy and performance efficient resources and thus minimizes the energy. Gridsim Toolkit has been used to validate the experimental results. These experimental results demonstrate that the proposed approach reduces energy more efficiently as compared to the existing algorithms without degrading the performance.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviations	**Descriptions**
QoS	Quality of Service
PC	Personal Computer
DVS	Dynamic Voltage Scaling
kbp	K percent best machines
EGRS	Energy Constraint Grid Resource Scheduling Algorithms
SLA	Service Level Agreement
LOC	Line of Code
CCCC	C and C++ Code Counter
SSD	Solid State Disks
EEHP	Energy-efficient and High Performance
PE	Processing Element
MIPS	Millions of Instruction Per Second
ODBC	Open Database Connectivity

Chapter 1

Introduction

Grid Computing is an epitome that furnishes seamless access to widely distributed resources for solving the big Computing applications. It is the main solution to comprise geographically disseminated and heterogeneous resources on a large scale [1]. This chapter introduces Grid Computing, its evolution, characteristics, applications, challenges and status of energy consumption in Grid Computing. It briefly presents the research motivation. At the end, it includes a discussion on the organization of the rest of the thesis.

1.1 Grid Computing

Grid Computing refers to the dynamic conglomeration of distributed resources on the basis of their capacity like capability, availability, performance, cost and users QoS requirements for computing big problems of different fields like science, engineering and commerce [2].

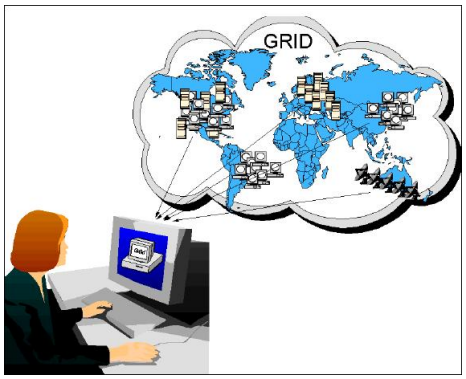

Figure 1.1: Grid Computing Resources [3]

It joins various computing resources together such as workstations, PCs, servers, storage elements like disks, and provides the method required to access them as

shown in Figure 1.1. The infrastructure or the environment that makes this thing possible is called the Grid.

1.2 Evolution of Grid Computing

Grid Computing came from previous ideas and efforts as described below as shown in Figure 1.2:

(i) In 1992, Larry smarr proposed the term Metacomputing. It was used to connect US supercomputing centers so that available computing and information resources will increase [4].

(ii) 1n 1995, Information Wide area year (I-Way) directed to link the supercomputers to create one super meta-computer [4].

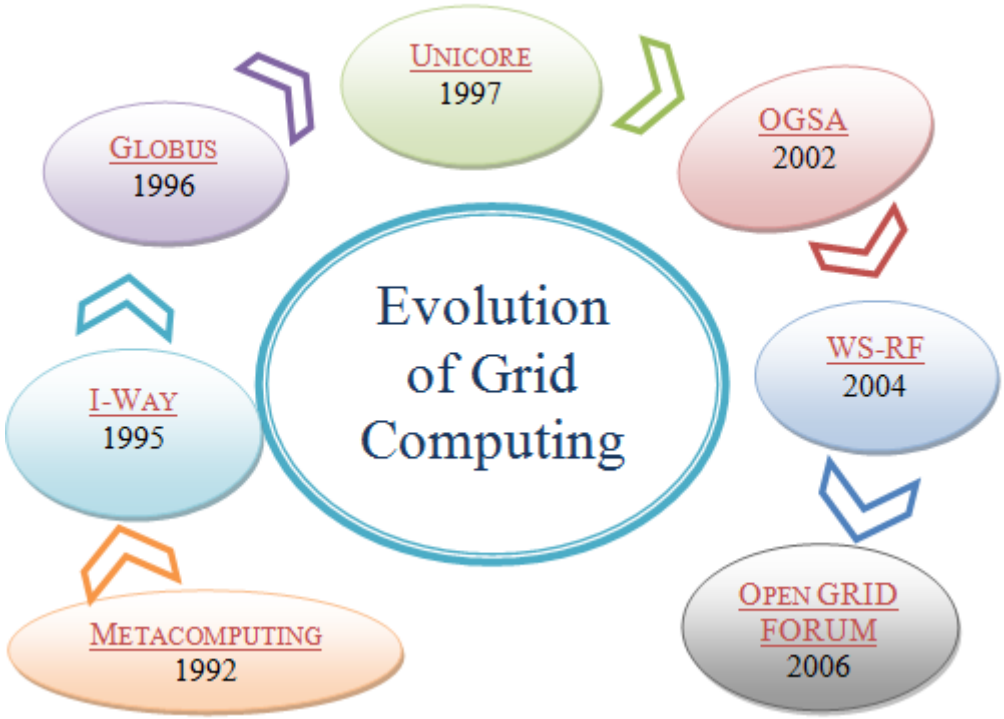

Figure 1.2: Evolution of Grid Computing

(iii) In 1996, GLOBUS project was proposed to build a middleware i.e. global nexus to provide help for discovery of resources, access of data, authorization and authentication [4].

(iv) In 1997, Uniform Interface to Computing Resources (UNICORE) project was started to provide seamless access to remote resources [4].

(v) In 2002, Open Grid Services Architecture (OGSA) model followed Web Services Description Language (WSDL) to consolidate the Grid resources across heterogeneous, disseminated and dynamic environment [4].

(vi) In 2004, Web Services Resource Framework (WS-RF) was conceived to connect the web services and Grid services [4].

(vii) In 2006, OPEN GRID FORUM aimed to develop the standards for Grids [4].

1.3 Characteristics of Grid

Main characteristics of Grid are described as follows and shown in Figure 1.3:

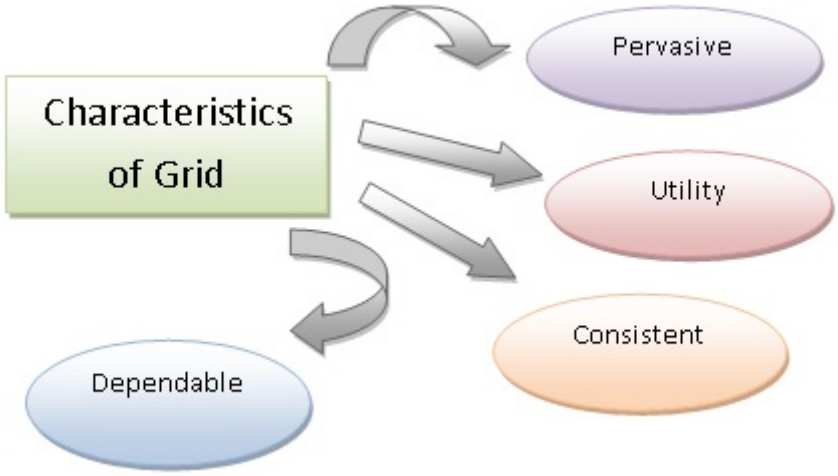

Figure 1.3: Characteristics of Grid

(i) Pervasive: The Grid grants the access of remote computing resources to the users from different platforms like PCs, laptops and mobile phones. It does not mean that resources are available everywhere but that the Grid manages resources to extract the maximum performance [5].

(ii) Utility: Users can get resources like computer power or storage capacity and also pay for what that he gets [5].

(iii) Consistent: Grid furnishes stable and regular services even in the presence of heterogeneous resources because Grid is built with standard interfaces, protocols and services and thus conceals the heterogeneity of the resources. Without such standards and criteria, development of application and pervasive use would not be possible [6].

(iv) Dependable access: Grid must deliver quality requirements. Dependable service is required because users require surety that they will get high levels of performance [5].

1.4 Grid Applications

The Grid is not only a solution for providing computational resources to solve large scale applications. Instead, it is an infrastructure that connects and unifies widely distributed resources in order to offer computing support for solving the applications or tasks. Various types of computing support provided by Grids are:

(i) Distributed supercomputing support allows the job to use the Grid to link the various computational resources to complete the job in minimum time. For example, Distributive Interactive Simulation (DIS) provides a base for connecting simulations of different types from different locations to create composite, real and virtual environment for the simulation of extremely interactive activities. It is used in military for planning and training [7].

(ii) High-output computing support permits the job to use the Grid to utilize the unused processor cycles for loosely coupled or independent tasks. For example, The Condor system manages large number of workstations worldwide. These resources have been used for the study of ground-penetrating radar and for the design of diesel engines [7] [8].

(iii) On-demand computing support allows the user to use the Grid to extract the resources that cannot be available locally at reasonable cost. For example, Aerospace Corporation Systems uses dynamic supercomputer resources for the processing of data from meteorological satellites to send the outcomes to remote meteorologists of a cloud detection algorithm in real time [7] [9].

(iv) Data-focused computing support permits applications to utilize the Grid to combine and form new information from distributed existing data repositories and databases. For example, The Digital Sky Survey allows the large amount of astronomical graphical data to be available to large number of network-accessible databases. This facility helps in astronomical research to get the new ideas which are based on distributed analysis [7].

(v) Cooperative computing support allows applications to utilize the Grid to provide and raise interactions among people in either synchronous or asynchronous way [11]. For example, University of Illinois at Chicago develops

the NICE system that allows the student to indulge in the creation of realistic virtual environment for studies and entertainment [10].

1.5 Challenges in Grid Computing

(i) Management: Many organizations and institutes are using Grid Computing. It disseminates the resources on wide geographically distributed environments and accesses the diverse resources, devices and machines. So it is a major challenge to manage and handle the administration of Grid Computing [12].

(ii) Load balancing: Scope of Grid Computing is too much wide. It is very much essential to balance the load among the different nodes of the network otherwise it degrades the performance. So an attempt is required to distribute the load to the resources evenly to improve the performance.

(iii) Resource monitoring: Resource management is essential to manage a large scale of Grid resources for dynamic access. Grid resources are inherently disseminated over the wide-area network. To solve the big applications, these resources require to be incorporated with corresponding Grid services. A resource monitoring mechanism is required to find and keep track of Grid disseminated resources [12] [13].

(iv) Energy consumption: Sustainability and cost reduction are the two main benefits of saving energy. Distributed system infrastructures like Grid, Cloud are the hungriest consumers of energy. Large amount of energy can be saved if an adequate and efficient energy saving policy is applied.

(v) Quality of Service: Quality means fulfillment of users different requirements and requirements of users always vary according to different environment. Fluctuation in the resources availability and decentralized control in Grid environment creates issue for achieving the quality.

1.6 Energy Consumption: A Grid Concern

Energy consumption is a major concern in distributed system because it not only increases cost but also effects the environment. The infrastructure that is required for maintaining these distributed systems like Grid, Cloud has resulted to exuberant power consumption. It is figured that servers use 0.5 percent of the worlds total electric power usage [14] [15], which if current need continues, are

projected to four times by 2020. Increase in energy consumption by distributed system and high performance computing system becomes an important issue for financial, technical and environmental reasons [16].
In a computational Grid [7], there is a need to cut down the power consumption and time, as energy is the amount of power used over a specific time interval. Reducing the power consumption will increase the execution time of a task on a processing element; however, the overall energy consumption will decrease because less power is consumed but performance will also decrease. The primary goal of these large-scale distributed systems is to provide the high performance. But now cost spent in energy consumption is increasing so rapidly that there is a great need to control it. Distributed system management is in dilemma and facing the question whether to control the energy consumption at the cost of performance is worth or not. So to come out from this energy-performance trade off, an energy performance efficient algorithm is required.

1.7 Research Motivation

The infrastructure of distributed systems like Grid consumes large amount of energy. So, increase in energy consumption is a major issue for technical, financial and environmental reasons.
Most of the existing techniques to reduce the energy consumption are hardware based techniques like using Solid State Disks (SSD) which is more energy-efficient storage media instead of Hard Disks and Tapes, using processors with high performance/watt rate, by setting the laptops in power down state like in sleep mode or in hibernate mode. These techniques attempt to save the energy consumption but up to some extent and all of these techniques are hardware based only. There is a need to build software techniques to save the energy consumption. Software techniques require managing the tasks and resources in such a way so that minimal consumption of energy will take place. This research work is based on the motivation to reduce power consumption by prioritizing and scheduling the tasks to the resources in such a way that the energy consumption will reduce without degrading the performance.

1.8 Organization of Thesis

The rest of the thesis is organized as follows:
Chapter 2 This chapter describes the research methodology and literature survey in detail to study the various existing energy and performance aware Grid scheduling algorithms.

Chapter 3 This chapter describes the research gaps analysis in detail along with the problem statement. It also describes the objectives of thesis.

Chapter 4 This chapter describes the proposed architecture and algorithm to solve the stated problem with the help of UML diagrams.

Chapter 5 This chapter focuses on the implementation details and experimental results description of Gridsim, Netbeans and snapshots of the designed Grid Portal, implemented algorithm and experimental results.

Chapter 6 This chapter concludes the thesis work and explains the contribution to the research work done. It also describes the possible future research work.

Chapter 2

Literature Review

The resources in the Grid have different availability, diversity of usage and cost plans and users have different priorities and goals that vary with time. Similarly, Grid workflows have different size and complexities. There are different resource and workflow scheduling approaches that aim to reduce the energy consumption or to improve the performance. This chapter discusses the state of art and analysis of energy and performance efficient Grid workflow and resource scheduling algorithms.

2.1 Grid Scheduling

Scheduling is a procedure to map and handle the execution of jobs over the distributed resources. It assigns the suitable resources to the jobs so that processing and execution of jobs can be completed to satisfy all the requirements of the users. Scheduling can have important affect on the performance of the system if it is done properly. As energy consumption is an important concern in Grid computing system. Proper scheduling can also have important impact on the energy consumption of the system. Various Grid scheduling algorithms have been studied to find how they can improve or modify to reduce the energy consumption in Grid environment.

2.2 Research Questions

The Literature review presented in this chapter summarizes the present state of the art in energy efficient and performance aware algorithms in Grid Computing by proposing answers to the following questions:

1. What is Grid Computing?

2. How to reduce the energy consumption in Grid Computing and how energy reduction degrades the performance?

3. How to overcome the main challenge of simultaneous optimization of energy consumption and performance maintenance in Grid Computing?

4. What are the different workflows based and resource management based scheduling approaches that are used to reduce the energy consumption and to improve the performance?

5. What are the different parameters to measure and compare the different algorithmic approaches in terms of energy and performance?

Table 2.1: Keywords

Keywords	**Synonyms**
Grid Computing	Computational Grid
Energy-efficient scheduling algorithms in Grid Computing	Energy aware scheduling algorithms in Grid
Resource scheduling algorithms in Grid Computing	Resource management in Grid

The systematic review started with the research questions and then proceeded by defining the searching keywords. The keywords are listed in Table 2.1. An in-depth research was carried out using the keywords and questions to find the factors that are useful in Grid workflow and Grid resource scheduling. Research questions helped to filter the research papers related to energy and performance efficient Grid scheduling algorithms. On the basis of research papers that have been studied, these scheduling algorithms are categorized into three categories: Energy & performance efficient heuristics, energy-efficient generic algorithms and energy-efficient policies as shown in Figure 2.1.

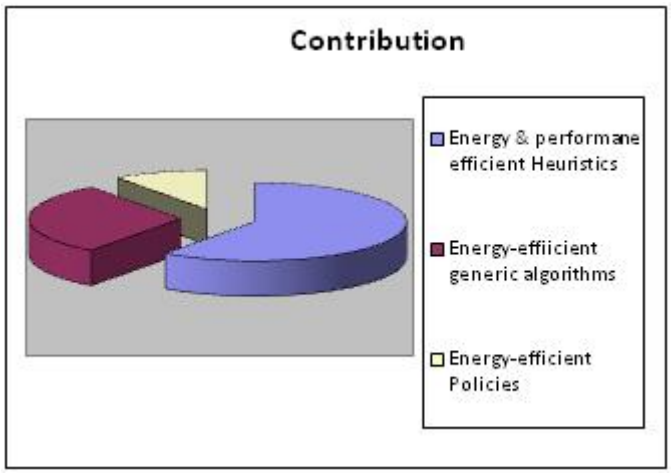

Figure 2.1: Breakdown of Algorithms

2.3 Energy and Performance-Efficient Heuristics

Energy-efficient heuristics are shown in Figure 2.2 and have been discussed in the consecutive section:

2.3.1 Grid Workflow Scheduling Algorithms

Grid Workflow Scheduling Algorithms have been proposed to schedule the workflow applications in the heterogeneous environment of resources [17]. They are divided into two types:

1. Best-effort based scheduling
2. QoS constraint based scheduling.

1. Best-effort based scheduling aimed to minimize the execution time of the workflow applications. It includes several Heuristics and Metaheuristics approaches as shown in Figure 2.2.

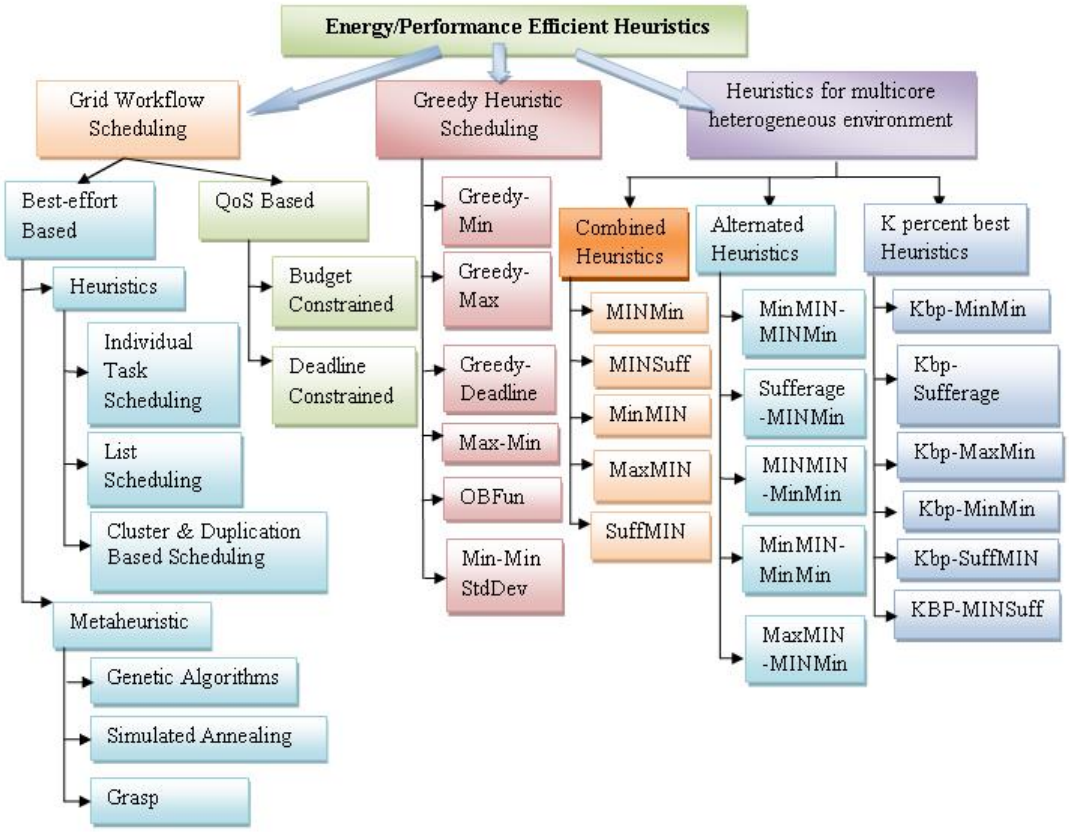

Figure 2.2: Energy/Performance efficient Heuristics

In heuristics based techniques:

(i) Individual task scheduling is appropriate for simple workflow applications in which tasks occur in sequence for execution.

(ii) List scheduling [2] is used when many tasks contend for confined number of resources. It focuses only to find efficient execution order of tasks and to determine efficient resource for these tasks to reduce the overall execution time. But it does not consider the data transmission time between the tasks that also contributes in overall execution time.

(iii) Cluster based & duplication based scheduling techniques minimize the data transmission time between interdependent tasks by grouping them or duplicating them to reduce the overall execution time of tasks [17].

In Metaheuristic [18] [19] based techniques:

(i) Genetic algorithms: It provides solution of high quality which is obtained from large population by applying the process of evolution.

(ii) Simulated annealing: It generates a new solution by applying number of iterations at different temperature.

(iii) Greedy randomized adaptive search procedures: It finds the global solution that minimizes overall execution time by doing number of iterations.

2. QoS constraint based scheduling algorithm examines the users QoS requirements and then schedule the task on the suitable resources so that users requirements can be accomplished. Under QoS constraint based algorithms, Deadline constrained algorithm minimizes the execution cost under set deadline constraint and Budget constrained algorithm reduces the execution time under set budget constraint [17].

Analysis Outcome
• Existing Grid Workflow scheduling algorithms count only two factors execution time and cost. They do not deal with energy consumption.

2.3.2 Greedy Heuristic Scheduling Algorithms

Greedy heuristics are based on Dynamic voltage scaling approach (DVS) approach to optimize the energy consumption and makespan [20]. In DVS [21] [22], after allocating the task to a Processing Element (PE), a PE is fixed to the minimum DVS level at which a task can execute under deadline. It includes seven greedy heuristics which are as follows:

(i) Greedy-Min assigns those tasks to the resources first which are having shortest completion times.

(ii) Greedy-Max assigns those tasks to the resources first which are having maximum completion times.

(iii) Greedy-Deadline assigns those tasks to the resources first which are having urgent deadlines.

(iv) Max-Min assigns the tasks to least efficient PE first, so that scheduling of succeeding tasks will become easier.

(v) MinMin StdDev firstly sorts the tasks in ascending order of their completion time and then rearranges the tasks by calculating standard deviation of each row of the tasks. The need behind that is to execute the task first which has most consistent execution time, so that tasks with inconsistent execution time can execute easily.

(vi) MinMax StdDev differs from MinMin StdDev that it rearranges the tasks in descending order after standard deviation calculation.

(vii) ObFun calculates the sufferage value of tasks and then assigns tasks with high sufferage value first. Sufferage value [23] in terms of completion time is the difference between the minimum estimated completion time and second best minimum estimated completion time and in terms of energy-efficiency is the difference between most energy efficient and second most energy efficient processing element.

Analysis Outcome
• Greedy-Min, Greedy-Deadline, Greedy-Max, MinMin StdDev and MinMax StdDev perform well for small-sized problems and ObFun performs best for large-sized problems in terms of mean energy consumption and mean makespan.
• Greedy Heuristics count energy consumption as well as makespan.
• Greedy Heuristics are based on DVS (Dynamic voltage scaling). DVS technique is difficult to implement in global Grids.

2.3.3 Heuristics on Multicore Heterogeneous Grid Computing System

Heuristics on Multicore Heterogeneous Grid Computing System have been proposed by Sergio Nesmachnow et al. [24] that considers energy consumption, makespan

and kbp (k percent best machines) [25]variations. They are categorized into three categories as shown in Figure 2.2:

(i) Combined Heuristics: They are the combination of traditional and greedy heuristics. All these heuristics are applied in two phases. Pair of task and machine is selected, in first phase that satisfies traditional heuristics and in second phase that satisfies greedy heuristics and vice versa. Like MIN-Min first selects a pair of task and machine that minimizes the makespan then from resulting pairs select that pair which minimizes energy consumption [24].

(ii) Alternated Heuristic: They are devised using traditional and combined heuristics. Pair of task and machine is selected using combination of traditional and combined heuristics alternatively. Like Sufferage-MINMin first selects the task having high sufferage value and assign it to the machine that can complete it in minimum time and then next task and machine pair is selected using MINMin combined heuristic [24].

(iii) k percent best Heuristics: In kbp heuristics, k percent best machines are found for each task that can complete the task in minimum execution time. Like kbp-Sufferage applies Sufferage traditional heuristics only on k percent best machines [24].

Analysis Outcome
• From the experimental results, MaxMin-MINMin performs well for both energy and makespan optimization for large size problem. Sufferage and MaxMin also perform well.
• They do not consider communication latency .
• Scheduling of workflows is not considered separately but it selects tasks simply on the basis of heuristics.

2.4 Energy-Efficient Generic Algorithms

Energy-Efficient algorithms are shown in Figure 2.3 and discussed in the consecutive section:

2.4.1 Utility based Scheduling Algorithm

Utility based Scheduling Algorithm is based on maximizing utility of all the tasks while achieving desired consumption of energy. It includes:

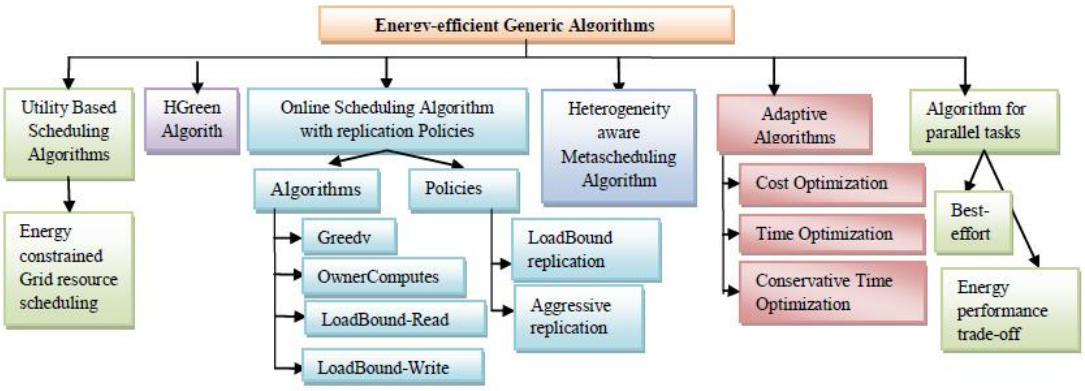

Figure 2.3: Energy-Efficient Generic Algorithms

(i) Energy Constraint Grid Resource scheduling algorithm (EGRS): It is based on non-linear optimization theory. EGRS aims to maximize the utility without surpassing the deadline and set energy budget. Due to complicated calculation, maximizing utility under energy and deadline constraint problem is decomposed into grid user and grid resource provider optimization problem. Grid user optimization problem finds an optimal solution to maximize the users benefit under energy budget and deadline constraint whereas Grid resource provider optimization problem finds an optimal solution of resource allocation to maximize providers revenue. Then Pricing based iterative algorithm is proposed for energy constrained grid scheduling [26].

(ii) In pricing based iterative algorithm, Grid user first calculates the optimal payment under energy and deadline constraint to maximize user satisfaction and sends it to resource providers after that resource providers compute optimal resource allocation to maximize their own revenue and send the updated price to grid users. Grid user receives price and again calculates optimal payment. In this way this algorithm iterates to find optimal solution. This algorithm optimizes both energy consumption and deadline [26].

Analysis Outcome
• Utility based scheduling algorithm maximizes the utility of grid users by counting both the factors energy consumption and time.
• Computational complexity is high in that algorithm.

2.4.2 HGreen Algorithm

HGreen Algorithm schedules the task in which task with highest energy consumption has high priority. Then task with high priority is assigned to the resource

which is highly energy-efficient. SPECpower benchmark [28] is used to define the energy efficiency of resources. The simulation performed on Gridsim [28]and results proved that HGreen reduce 50% power consumption lower than the randomly task assignment [27].

Analysis Outcome
• HGreen algorithm can significantly cut down the consumption of power in global Grids.
• HGreen does not consider the performance parameter.

2.4.3 Online Scheduling Algorithms and Replication Policies

Online Scheduling Algorithms and Replication Policies are proposed to balance the load and to minimize the response time. Online scheduler schedules the request to appropriate host according to the algorithm and replica manager handles the creation of replica, migrate and eliminate the replica when necessary. Online scheduler also categorizes the hosts for each request. DataSourceHost that owns the input data, ComputeHost i.e. a computational resource and DataDestination Host stores the result [30]. There are four online scheduling algorithms:

(i) Greedy: Greedy online scheduling algorithm attempts to schedules the request to the host which completes it in minimum time [31].

(ii) OwnerComputes: It selects a host that can act as both ComputeHost and DataSourceHost as well as completes the request first.

(iii) LoadBound-Read: LoadBound-Read selects a host that completes the task in minimum time and performs better than estimated performance. Performance is estimated on the basis of frequency of replications and load average of the host.

(iv) LoadBound-Write: It selects a host that has the smallest response time.

There are two replication policies:

(i) LoadBound-replication: In this policy, replica manager creates the replica of that data which is accessed more [30].

(ii) Aggressive-replication: In this policy, replica of each job data is generated and then sent to that host which maximizes the performance [30].

Analysis Outcome
• All these scheduling algorithms improve the performance by reducing the response time with the replication policies.
• There is overhead of creating the replicas.

2.4.4 Adaptive Algorithms

Adaptive algorithms adapt the change in resources by updating the resource information and rescheduling them [32]. It comprised of three algorithms:

(i) Cost optimization: It endeavors to complete the task in minimum cost within the limited deadline.

(ii) Time optimization: It tries to complete the task in minimum time with in the limited budget.

(iii) Conservative Time optimization: It attempts to complete the task within the limited deadline and budget.

This algorithm divides the resources into expensive and inexpensive resources. Then task with high task completion time allots to inexpensive resources and vice versa.

Analysis Outcome
• On the basis of QoS requirements, user can choose any adaptive algorithm. If user have limited budget then user can choose cost optimization and if user requires completing the task as quickly as possible then user can go for time optimization algorithm.
• It does not count energy consumption parameter.

2.4.5 Power Aware Scheduling Algorithms for Parallel Tasks

Power Aware Scheduling Algorithms for Parallel Tasks are based on Dynamic Voltage frequency scaling technique [34] [35]. It discusses two approaches:

(i) Best effort scheduling: It attempts to reduce the energy consumption without degrading the performance. It scales down the processor voltage level during the non critical jobs execution because delay in execution of non-critical jobs does not affect the overall execution time [33]. It also scales down the voltage when processing element is not doing any data communication and job execution.

(ii) Energy-performance trade-off scheduling: It applies Green Service Level Agreement (SLA) negotiation between consumers and resource providers. It scales down the processor voltage similarly as best effort but it allows the execution time to expand that is negotiated in green SLA [34]. These allow reducing energy consumption without degrading performance in parallel tasks execution. Green SLA negotiates the task response time, energy consumption and carbon dioxide emission metrics. It schedules the tasks to the resources by assuring all the consider metrics.

Analysis Outcome
• They attempt to reduce the energy consumption without degrading the performance for parallel tasks.
• Energy-performance trade-off can save the more energy when user can pay for extra execution time.

2.4.6 Heterogeneity Aware Metascheduling Algorithm

Heterogeneity Aware Metascheduling Algorithm (HAMA) sorts the jobs with earliest deadline first. Resources are also sorted in terms of energy-efficiency. CPU power and cooling system efficiency have been considered to find energy-efficient resource. Then it assigns the job with earliest deadline to most energy-efficient resource and energy is further reduced by applying DVS approach [36].

Analysis Outcome
• HAMA algorithm reduces the energy consumption and does not consider performance.

2.5 Energy-Efficient Policies

Different energy policies are shown in Figure 2.4 and have been discussed in the consecutive section:

2.5.1 Smart Scheduling Policies in Grid Computing

Smart Scheduling includes different scheduling energy policies [37]as shown in Figure 2.4. These policies handle the resources and are applied on Grid5000 [38]Grid organization in France to find which policy saves how much energy.

(i) Always On: This policy makes the resources available all the time and never in switched off state.

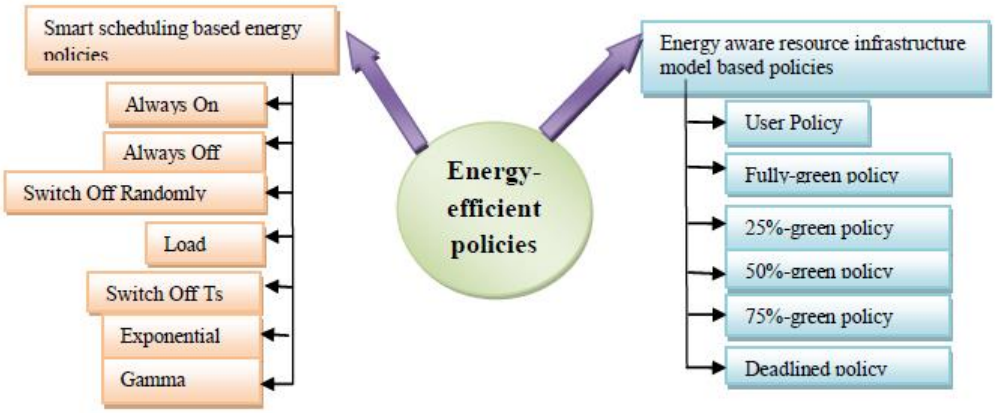

Figure 2.4: Energy-Efficient Policies

(ii) Always Off: This policy allows the resources to remain in switch off state until a job arrives.

(iii) Switch Off randomly: This policy allows the resources to remain idle or in switch off state based on Bernoulli distribution. It is a mid way of Always On and Always Off.

(iv) Load policy: It imparts the resources on the basis of load. If load is below than threshold then resources remain idle otherwise gets switch off.

(v) Switch Off Ts: It tries to minimize the energy spent in booting and shutting down the resources.

(vi) Exponential and Gamma policy: These policies having different probability calculating functions sets a threshold value and then check if probability of arriving of new job is less than that then resource remain idle otherwise gets switch off.

Smart Scheduling also includes Arranging Policies that work together with energy policies. Arranging Policies are Do Nothing and Simple Aggregation of Jobs which are defined below:

(i) Do Nothing works together with Always On and does not change anything.

(ii) Simple Aggregation of Jobs policy takes care that if required resources are available but not assigned and assigned ones are switch off. Then it provides available resources to the job and saves the energy and time. Always Off and Load policy gives good result in terms of energy saving but Switch Off Ts gives good result in energy saving as well as reduce power cycles. Aggregation of Jobs policy is best in energy saving and reducing power cycles [37].

Analysis Outcome
• All these energy policies and arranging polices shows saving in energy consumption when implemented on Grid5000.
• It does not consider performance.
• Heterogeneity of resources is not taken in Grid5000.

2.5.2 Energy Aware Resource Infrastructure (EARI)

EARI is an energy saving model in which scheduler manages the on/off cycle of the resources to reduce the energy consumption. This model furnishes the user with a portal through which user can submit their reservation for resources. Then scheduler sues the users submissions. Then handle the resources and provide access to users according to their reservation. To make the reservation user can view the previous reservation result and can select the reservation accordingly. Some reservations have least booting and shutting of resources so it has low energy consumption as compared to other reservations. Energy consumption by reservations is estimated if it starts at the next possible start time, just after the next possible end time of reservation, l second (l refers to the length of reservation) before the next possible start time and during the slack period. On the basis of estimation, only these reservations are selected which have minimum on and off of the resources [39] [40]. Then few green policies have been proposed to shift the reservation at another date to get better result.

(i) User policy: It allows selecting the reservation that is suitable for user requirements.

(ii) Fully-green policy: This policy allows selecting the reservation that is most energy efficient.

(iii) 25%-green policy: It permits 25% reservations basis on the criteria of fully-green and rest on the basis of User policy.

(iv) Similarly 50%-green and 75%-green policies according to their percentage values.

(v) Deadlined policy: It follows fully-green policy if delay in the fulfillment of user requirements is not more than 24 hours. Otherwise it follows user policy.

Analysis Outcome
• The energy consumption result of the Lyon site proved fully-green policy is most energy efficient energy policy.
• Impact on performance does not taken in to account here.

2.6 Review Analysis

Analysis of various existing scheduling algorithms has been done on the basis of energy and performance in terms of time and cost as shown in Table 2.2. Algorithms are compared on the basis of following metrics:

(i) Energy Consumption (EC): Energy consumption is defined as the amount of energy that is consumed by the resources during the tasks execution.

(ii) Makespan: Makespan is defined as the total execution time from the submission of first task to the completion of last task [41].

(iii) Cost: Cost is defined as monetary cost that is spent in whole scheduling and execution operation.

Table 2.2: Comparison between Existing Scheduling Algorithms

Scheduling Algorithm		**EC**	**Makespan**	**Cost**
Workflow Scheduling Algorithms	Best-Effort based		✓	
	QoS based		✓	✓
Greedy heuristics Scheduling Algorithms		✓	✓	
HGreen algorithm		✓		
Combined, Alternated, kbp based heuristics on multicore heterogeneous based environment		✓	✓	
Utility based energy constraint scheduling algorithm		✓	✓	✓
Adaptive Scheduling algorithms			✓	✓
Online Scheduling algorithm with replication policies			✓	
Heterogeneity aware Metascheduling Algorithm		✓		
Algorithms for parallel tasks		✓	✓	
Smart scheduling based energy policies		✓		
Energy aware resource infrastructure model based energy policies		✓		

2.7 Conclusion

Various Grid scheduling algorithms that focus mainly on energy parameter as well as performance have been reviewed in this chapter. The result of the study was

classified into three categories and analyzed. From the analysis it can be concluded that Greedy heuristics, heuristics on multicore heterogeneous environment and utility based algorithms attempt to optimize the energy and performance simultaneously and perform well.
In the next chapter, research gaps are discussed and Problem statement is formulated.

Chapter 3

Research Gaps and Problem Statement

Based on Literature Survey, research gaps in the existing techniques have been identified and analysis is presented. On the basis of this analysis, Research problem is defined and then objectives are described.

3.1 Gap Analysis

From the literature survey, following algorithms have been identified that focus on both energy and performance factors:

- Greedy Heuristics Scheduling Algorithm,
- Utility Based Scheduling Algorithm.
- Multicore Heterogeneous Environment Heuristics.

Table 3.1 the analysis of these algorithms. From the analysis it can be concluded that an efficient scheduling algorithm that simultaneously optimizes both energy and performance is required.

3.2 Problem Formulation

The energy consumption in Grid Computing system is an important problem and there is always a trade-off between energy consumption and performance. To minimize the energy consumption with no degradation in performance is one of the main challenges in Grid Computing for which an efficient Grid scheduling system is required. This research work intends to propose an algorithm that can simultaneously optimize both energy and performance and which considers communication latency, scheduling of workflows and is also easy in computation.

Table 3.1: Gap Analysis of Selected algorithms

Existing Algorithms	**Analysis**	**Techniques**
Greedy Heuristics Scheduling Algorithms	It takes into account both energy consumption and performance but these are based on Dynamic Voltage Scaling approach i.e. difficult to implement in global Grids.	Dynamic Voltage Scaling(DVS)
Utility based scheduling algorithm	It maximizes the utility of Grid users by counting both energy consumption and time but it is very complex in computation because it uses Lagrangian approach to solve the problem like to complete the execution in limited energy budget. Similarly to complete the execution in limited deadline.	Lagrangian approach
Heuristics on multicore heterogeneous environment	It does not consider communication latency i.e. the time that the tasks takes to communicate with each other.	Traditional and Greedy Heuristics

3.3 Objectives

(i) To study the existing energy and performance aware scheduling techniques of Grid computing.

(ii) To develop an energy aware algorithm for optimizing both energy and performance simultaneously.

(iii) To implement the proposed algorithm in Grid environment.

In the next chapter, proposed architecture and algorithm have been discussed with the help of UML diagrams.

Chapter 4

Proposed Energy-Efficient and High Performance Approach

This chapter analyzes the problem through UML diagrams and presents the design of proposed solution through layered architecture of the proposed technique and its flowchart. In addition, it also proposes a scheduling solution by considering the dependency among the tasks and mapping those tasks to efficient resources.

4.1 Analysis of the Proposed System

In the proposed system the Grid tasks should be analyzed, prioritized and scheduled to the energy-efficient resources that will ensure the performance and minimal energy consumption. For that the proposed system is intended to provision the task and resources by taking both energy and performance in consideration and design a scheduling algorithm so that both energy and performance can be optimized.

Analysis of proposed system is done with the help of Use Case diagram. Different components of the system are identified and their working with each other is presented in UML diagrams. UML diagrams are the best approach to study and analyze the different parts of the system and their working.

• Use Case Diagram

Use Case diagram [42] depicts the overall behavior of proposed system by describing the set of actions that the system can perform in collaboration with external users of the system as shown in Figure 4.1. Actors includes Grid user, Database, Provisioning Manager, Gridsim for simulating the Grid environment and Scheduler. All the actions are described within the system boundary that the system performs which includes submission of tasks by Grid user, analyzing and prioritizing the tasks, performance provisioning energy provisioning by Provisioning

Manger, scheduling the tasks to the resources by scheduler, simulation of Grid environment by Gridsim, checking the statistics of energy, execution time and cost by Grid user. Thus, the proposed approach is analyzed by describing its users and actions in Figure 4.1. The design of the proposed solution has been explained in the next section.

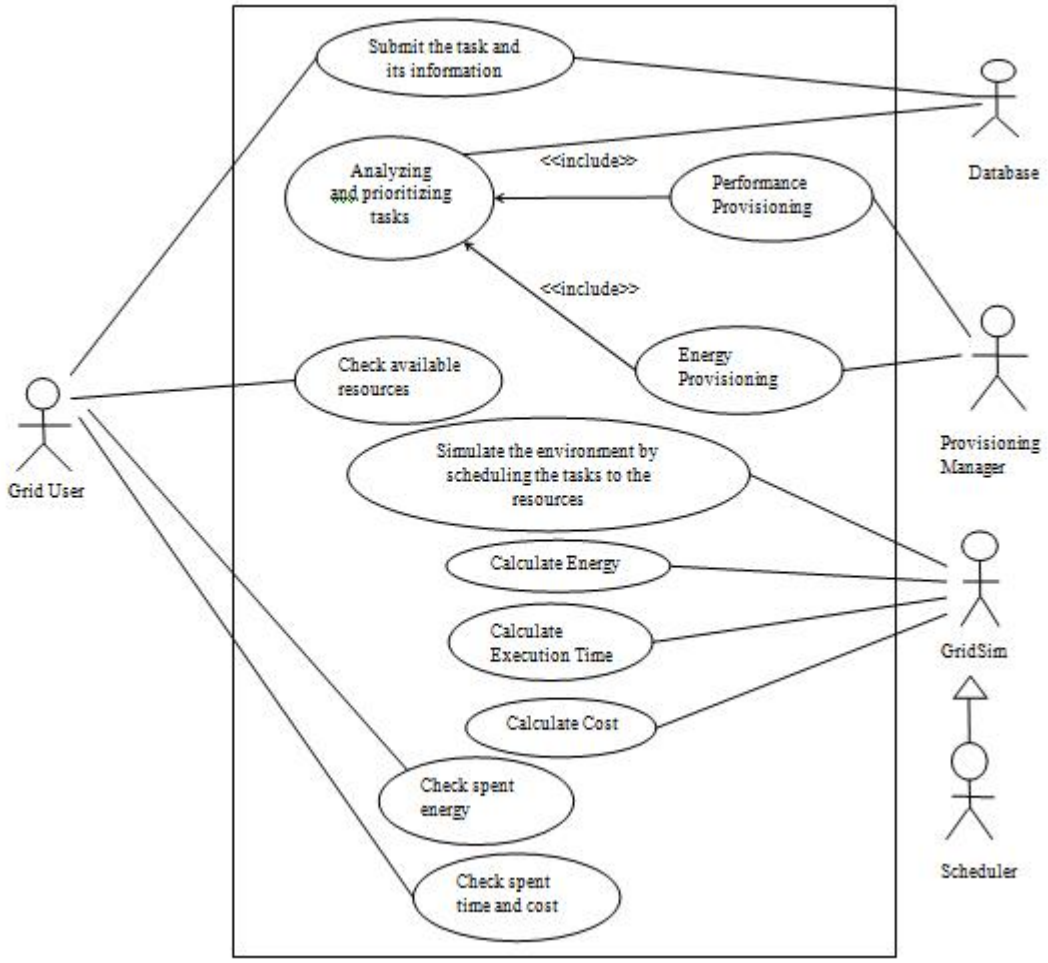

Figure 4.1: Use Case Diagram of Proposed Approach

4.2 Proposed Energy-aware Layered Grid Architecture

Three layered energy and performance aware Grid architecture is proposed as depicted in Figure 4.2. It focuses on reducing the energy consumption by simultaneously improving the performance in Grid environment. The architecture has three layers, Grid Portal, Provisioning Manager and Scheduler.

4.2.1 Grid Portal

Grid Portal provides an interface to the users through which user can submit the task and the information related to the task that is required for its execution. Information gathered from the interface is stored and used by task analyzer to analyze the task complexity and dependency. Grid Portal makes it easy for the Grid users to submit the tasks for execution. Lines of Code (LOC) and Function Point metrics have been used to measure the complexity of tasks. Function Point considers 14 complexity factors that counts processing complexity of the project or a task. To measure the dependency in a task Fan-in, Fan-out and Information Flow Metric have been used.

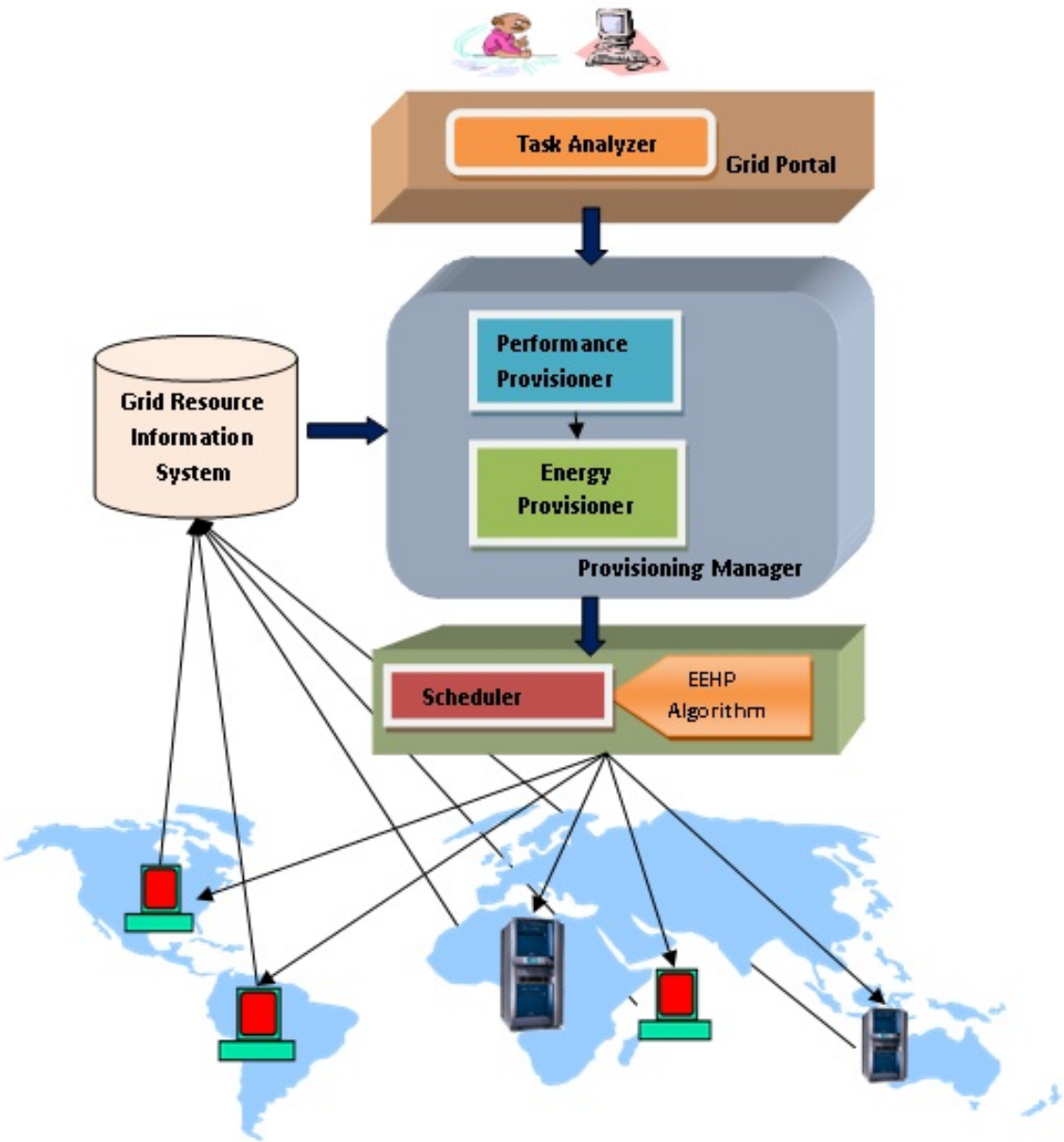

Figure 4.2: Energy-Aware Layered Grid Architecture

- Task Analyzer : Task information like its size, complexity and dependency is collected through Grid Portal.Tasks are then analyzed on the basis of collected information and categorized as heavily loaded, complex or dependent tasks. A set of five tasks have been taken for the experimental purpose as shown in Table 4.1. To calculate the Function Point metric of the tasks, a separate interface is provided to the users. To calculate the LOC, Fan-in, Fan-out and Information Flow Metric of the tasks, C and C++ code counter (CCCC) free software tool for the measurement of source code related metrics is used [43]. These entire submitted tasks shown in Table 4.1 are analyzed on the basis of size and structural metric and sufferage value [45] based on time and energy using the following equation:

$$weight = size * var_{time} * var_{energy} + dependency \tag{4.1}$$

Where size represents the size of a task in terms of LOC or Function Point, var_{time} refers to variance of time i.e. representing the difference between the execution time of a task on most and least efficient resource. var_{energy} refers to variance of energy i.e. representing the difference between the energy consumption of a task on most and least efficient resource, dependency representing the task dependencies in terms of Information Flow Measure.

(i) Size: Size of a task can be calculated using either LOC [44] metric or Function Point metric [45]. To calculate the weight of a task, size metric is used because tasks execution time and energy consumption differs as the size and complexity of task varies. Metrics that are used to calculate the size are described below:
Line of code is a software metric used to calculate the size of a software program by counting the number of lines in the source code of a program except the blank and comment lines [44].
Function Point is software metric which is used to find the size and complexity of a program. It considers 14 technical complexity factors that are used to provide an indication of problem complexity [45] [46]. Higher the complexity or size of task then task requires more time and energy to execute.
LOC of the tasks which are specified in Table 4.1 is calculated using CCCC tool and Function Point is calculated by providing separate interface for its calculation. Here FP has been used during experimentation.

(ii) var_{time}: It refers to the Variance of time i.e. the difference between the

execution time of a task on execution upon the most energy efficient resource and least energy efficient resource. This factor is responsible for maintaining the performance of a task. Task with high value of var_{time} shows that its performance will be affected more if it is not allotted to more energy efficient resource. So the task with high value of var_{time} is of high weight. Value of var_{time} for the tasks is calculated using the past data of the resources i.e. stored in database.

(iii) var_{energy}: It refers to the Variance of energy consumption i.e. the difference between the energy consumption by a task on execution upon the most energy efficient resource and least energy efficient resource. This factor is responsible for saving the energy consumption of a task. Task with high value of var_{energy} shows that it will consume more energy if it is not allotted to more energy efficient resource. So the task with high value of var_{energy} is of high weight. Value of var_{energy} for the tasks is calculated using the past data of the resources i.e. stored in database. var_{time} and var_{energy} [47] factors are based on the idea of sufferage value.

(iv) Dependency: It refers to how many modules of a task are dependent on each other. A program with multiple tasks has dependencies among the tasks. Task having high dependency will consume more energy as compared to the task having less dependency. High dependency task cause the CPU to be idle or free because when the task is busy in doing Input/output activity CPU cannot be assigned to any other task as the dependent task cannot start executing until the task that it depends upon is complete. Whereas, low dependency tasks use the resources in continuous mode and CPU idle time or switching time is negligible.
The CPU is the main power consumer in system devices. Power consumption can be saved significantly by switching the CPU from running state to off or low power state when it is not doing any useful work. But switching the state of CPU also has energy overhead. It is generally assumed that during the execution time of a task, all the system devices whose support is required will stay in their active state. So, energy is wasted even when resources are not doing any useful work.
Therefore, while executing the task with high dependency whether the CPU sits idle or switches into low power state in both the cases energy consumption will be more as compared to the execution of low dependency task. So, it is worth to assign more energy-efficient resource to the task having high dependency.

To measure dependency following metrics have been used:

- *Coupling between Objects*: It refers to the count of other modules of a program which are coupled to the current module either as a client or a provider. High coupling indicates the dependency between the modules is high [46] [48].
- *Fan-in Fan-out*
  - Fan-in: The count of other modules in a program which send information or data into the current module of a program [48].
  - Fan-out: The count of other modules in a program into which the current module sends information [48].
- *Information Flow Measure (IFM)*: It is measured as the square of the product of the fan-in and fan-out of a module to find the structural complexity of a module. So it is a combined measure of Fan-in Fan-out [48].

  To calculate above metrics user can use CCCC tool [44] i.e. C C++ code pointer. It can calculate LOC size metric, Object Oriented metric and Structural metric as shown in Figure 4.3 and 4.4. Thus, the tasks are prioritized and a task with high weight is assigned high priority. After calculating weight, tasks are sorted with task ID using bubble sorting.

Metric	Tag	Overall	Per Module
Number of modules	NOM	4	
Lines of Code	LOC	85	21.250
McCabe's Cyclomatic Number	MVG	3	0.750
Lines of Comment	COM	0	0.000
LOC/COM	L_C	******	
MVG/COM	M_C	------	
Information Flow measure (inclusive)	IF4	0	0.000
Information Flow measure (visible)	IF4v	0	0.000
Information Flow measure (concrete)	IF4c	0	0.000
Lines of Code rejected by parser	REJ	12	

Figure 4.3: Size Metric Calculation

Module Name	Fan-out			Fan-in			IF4		
	vis	con	inc	vis	con	incl	vis	con	inc
Grid	0	0	0	2	1	2	0	0	0
JFileChooser	1	0	1	0	0	0	0	0	0
JFrame	1	1	1	0	0	0	0	0	0
anonymous	0	0	0	0	0	0	0	0	0

Figure 4.4: Structural Metric Calculation

Table 4.1: Task Information

Tasks	**Size (LOC)**	**Size (FP)**	**Dependency (IFM)**	var_{time} **(secs)**	var_{energy} (watts)	**Millions of Instructions**
Gridlet 0	286	28	0	5	3917	222340
Gridlet 1	474	15	0	5	8143	4741236
Gridlet 2	1788	82	1	7	29817	13178798
Gridlet 3	116	30	0	5	936	20000011
Gridlet 4	85	8	0	5	936	12356785

4.2.2 Provisioning Manager

Provisioning Manager handles the provisioning of both tasks and resources to improve the performance and to reduce the energy consumption. Performance can be improved by fulfilling the quality requirements of the users. Provisioning Manager is responsible for analyzing and sorting the tasks by taking energy and performance factors under consideration. It also manages the resources whose information comes from Grid resource information system as shown in Figure 4.2. Provisioning Manager is working into two steps which are described below:

- Performance Provisioner:It provisions the tasks and resources both for Performance. In case of tasks, by using the past information of execution time of the tasks on different resources, variance of time is calculated i.e. a difference between the execution time of a task on the most performance efficient resource and least efficient resource. The task having maximum value of variance of time affects the performance highly as calculated by equation 4.1. So a task with high value of variance of time assigned high priority to schedule and execute first. In case of resources, Joulesort benchmark [49] is used that defines energy and performance efficient resources. These efficient resources use SSD (Solid State Disks) drives that are better in performance [50] [51]. Resources are shown in Table 4.2. The resources that are proved as an energy and performance efficient by Joulesort benchmark have been considered. Mips Rating is a metric which is used to measure the performance of processor [52]. MIPS value of all these processors mentioned in Table 4.2 is calculated using clock rate and No. Of records sorted per joule information [53].

- Energy Provisioner:It provisions the tasks and resources both to reduce the energy consumption. In case of tasks, by using the past information of energy consumption of the tasks on different resources, variance of energy is calculated i.e. a difference between the energy consumption by a task on the most energy efficient resource and next efficient resource. The task having maximum value of variance of energy affects the energy consumption more

as shown in equation 4.1. So a task with high value of variance of energy is assigned high priority to be scheduled and executed first.

Table 4.2: Resources List [50] [51]

Resources	**Configuration**	**Clock Rate**	**Mips Rating**	**Power in watts (10 GB data)**
Intel Core i7-2700K	16GB RAM, 16 x 300 GB Intel 710 Series SSDs, 1 160 GB Intel 510 Series SSD	3.5 GHz	897500	165
Intel Core i5-2400S	16GB RAM,7 x 120 GB Intel 510 Series SSDs	2.5 GHz	873750	93
Intel Xeon L3426	12GB RAM, Fusion-io ioDrive (80GB), 4 x Intel X25-E (3 x 32GB, 1 x 64GB)	1.86GHz	561250	105
Intel Atom 330	4GB RAM,4 x Super Talent UltraDrive GX MLC 256GB	1.6GHz	310000	142
Quad Core AMD Opteron 2373	16GB RAM 80GB FusionIO	2.01GHz	145000	252

In case of resources, Joulesort benchmark is used that defines energy-efficient resources. Joulesort as a benchmark is selected to measure the energy efficiency because sort emphasizes the all core components of a system which are CPU, memory and I/O [49]. Five most energy efficient resources according to Joulesort benchmark have been taken to carry out the simulation in Gridsim as shown in Table 4.2.

4.2.3 Scheduler

It collects all the information from the provisioning manager and schedules the tasks to the resources according to the Energy-efficient and High performance algorithm.

Proposed EEHP Algorithm

EEHP Algorithm: It refers to Energy efficient and High Performance algorithm. It assigns the high priority to the task i.e. complex in size, highly dependent and which is affected more if it is not assigned to energy efficient resource in terms of both energy and execution time. As shown in equation 4.1, task with highest weight has high priority. Then map the high priority task to the most energy and performance efficient resource which is found out by joulesort benchmark. The pseudo-code for EEHP algorithm is described in Algorithm 1.

Algorithm 1 Energy-Efficient High Performance Algorithm

1. //Declarations and Definitions
2. T=t1....tn //set of the tasks
3. R=r1....rn //set of energy and performance efficient grid resources
4. loc(t)
5. fp(t)
6. var_{time} (t)
7. var_{energy} (t)
8. dependency(t)
9. LT[]
10. LR[]
11. pos(L,criterion)
12. weight(t)
13. power(r)
14.
15. // Task Analyzer Phase
16. For each task t in T do
17. weight(t) = analyzer(t)
18. EndFor
19.
20. //Provisioning Phase
21. For each task t in T
22. p=pos(LT,Weight(t))
23. add(t,LT,p)
24. EndFor
25. //Scheduling Phase
26. while $\exists t$ in LT do
27. t=LT.get(0) //the most complex, dependent and power expensive task
28. re=LR.get(0) //the most energy and performance efficient resource
29. schedule(t,re) //assign t task to the re
30. remove(LT,0) //remove first item list
31. EndWhile

Table 4.3: Abbreviations used in Algorithm 1

Abbreviation	Description
loc (t)	Size of tasks in terms of lines of code
fp (t)	Function point value of the tasks
var_{time} (t)	Variance of time
var_{energy} (t)	Variance of energy
dependency(t)	Dependency of modules in a task.
LT[]	List of tasks prioritized by most complex, dependent and decreasing energy waste
LR[]	List of resources which are highly energy and performance efficient resources
pos(L, criterion)	Insert position of new element on the basis of weight criteria at list L
weight(t)	Weight of a task
power(r)	Power consumption rate of resource
analyzer()	Analyze the tasks on the basis of complexity and dependency
add(t, LT, p)	Add the task t to the prioritized list of tasks i.e. LT at position p
LT.get( )	It is a function to get the most complex and dependent task
Schedule(t, re)	This function schedule the task t to the resource re.
Remove(LT, 0)	This function removes the tasks from the list as they get scheduled .

Step by step working of the proposed EEHP scheduling algorithm is shown in Figure 4.5. It elaborates the procedure for submission of tasks, analysis of tasks, sorting of tasks on the basis of weight and scheduling of tasks to the resources In the proposed algorithm, allocation of tasks to the resources is carried out in a manner as described below:

(i) First, List of tasks is created which are submitted by users through Grid Portal for execution and List of energy efficient resources is created which are available on site.

(ii) All the tasks are analyzed on the basis of their complexity (Using Function Point and LoC), dependency (Using Information Flow Measure) and their past history that gives estimate about their energy and time consumption data.Then tasks are prioritized by assigning weight to them.The task having higher weight will have higher priority.

(iii) Tasks are sorted on the basis of weight assigned to them and then scheduled to the resources accordingly. So, the task having higher priority is allocated to the most energy-efficient resource for execution.

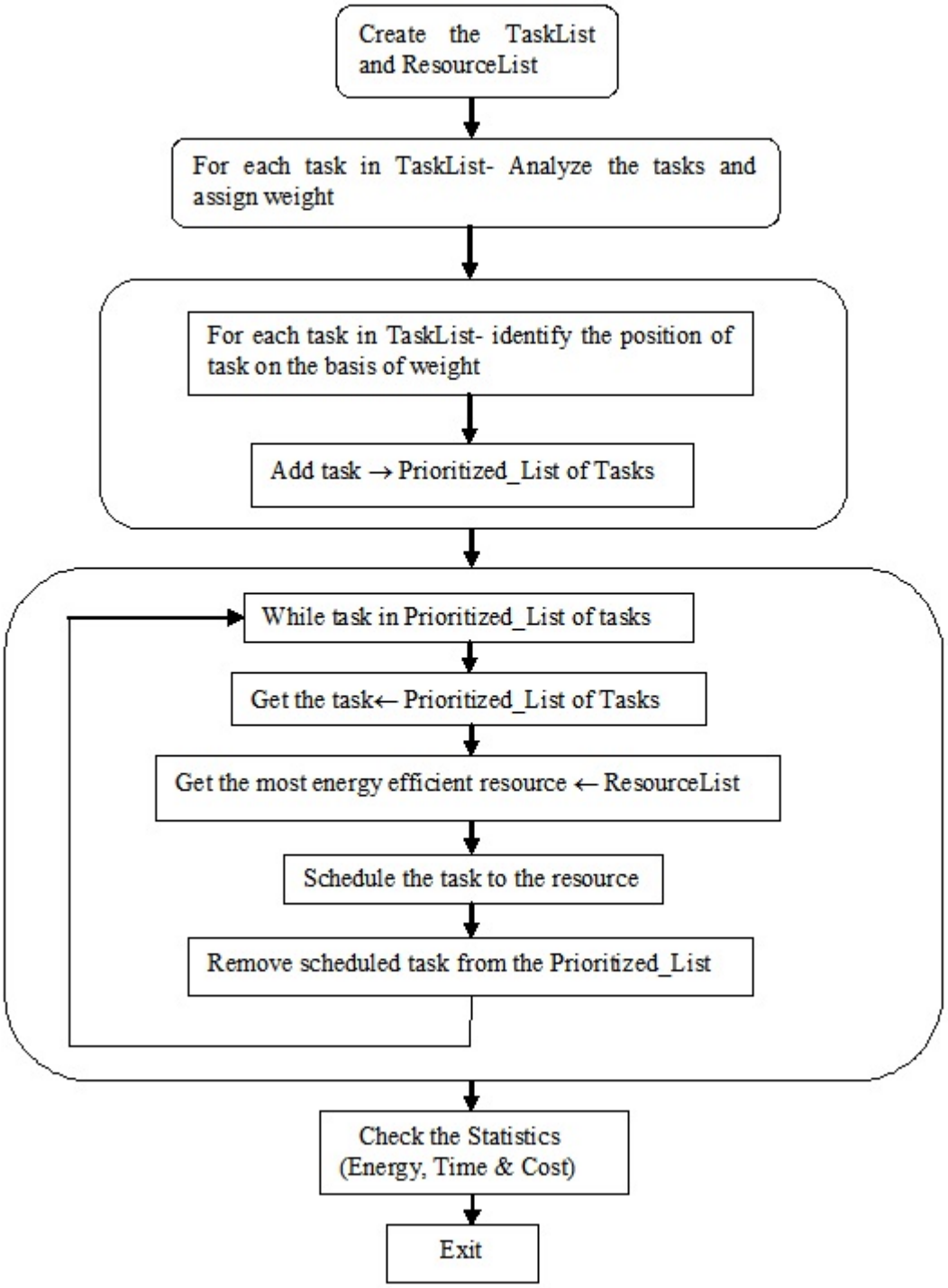

Figure 4.5: Flow Chart of Proposed Algorithm

4.3 Conclusion

In this chapter, Energy aware layered Grid architecture has been proposed to optimize both energy and consumption. Along with this, Energy-efficient and High performance algorithm to schedule the tasks to the resources in such a way so that energy can be reduced has also been proposed and designed. In the next chapter, the implementation of Grid Portal and the proposed algorithm and experimental results have been presented.

Chapter 5

Experimental Results

This chapter discusses tools for setting up Grid environment, implementation of Grid Portal and EEHP Algorithm on Gridsim Toolkit, experimental results of this approach and comparative analysis of this approach with other existing algorithms.

5.1 Tools for setting Grid Environment

Tools required for implementing the EEHP algorithm on Grid are described below.

5.1.1 Gridsim Toolkit

Different types of disseminated systems such as Grids and clusters use Gridsim toolkit to do the simulation of tasks or job scheduling. Information services are used to find the resources and provide interfaces to map the tasks to the resources and handle their execution and completion. These capabilities can be used to measure the performance of different scheduling algorithms or heuristics by simulating the resources and Grid schedulers [29]. It provides following features [29]:

(i) Permits creation of heterogeneous types of resources.

(ii) Potentiality of resources is defined in terms of Millions of Instruction Per Second (MIPS) which are specified in Standard Performance Evaluation Corporation (SPEC) benchmark.

(iii) Time shared or space shared allocation policies are used in Gridsim toolkit for resources.

(iv) Advance reservation policy is provided for booking of resources.

(v) Tasks or jobs can be different and they can be CPU bound or input output bound.

(vi) No limit on the number of tasks that can be submitted.

(vii) Both static and dynamic schedulers simulation is supported.

(viii) Time and cost spent in execution parameters can be recorded and analyzed.

5.1.1.1 Gridsim Architecture

Gridsim modular architecture is shown in Figure 5.1. Java Virtual Machine with Java interface is provided at first layer for different types of processors because its implementation is platform independent. A basic discrete-event infrastructure is built at second layer using interfaces of first layer. At third layer, simulation of resources, applications and so on entities to create higher level entities.

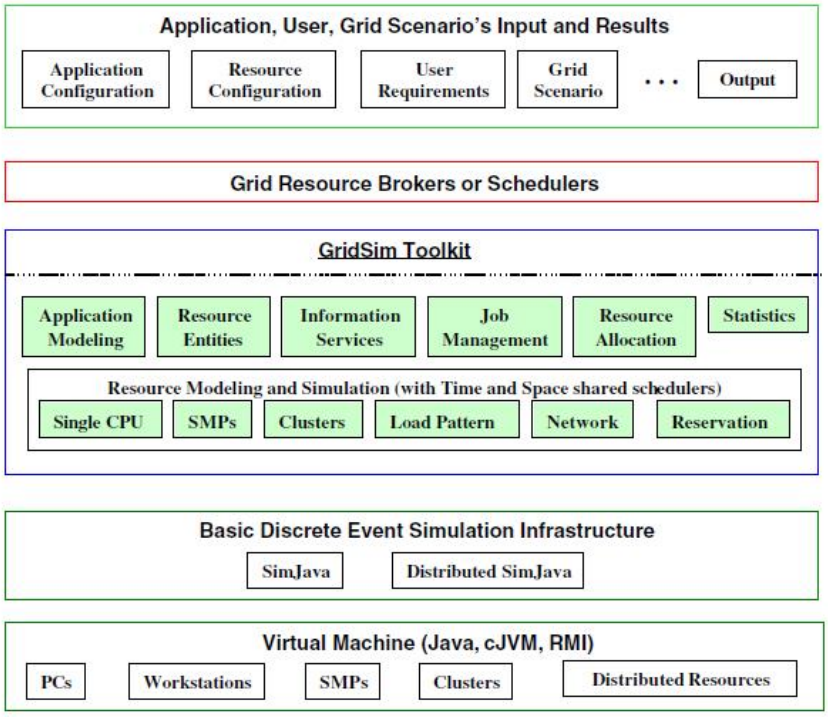

Figure 5.1: Modular Architecture for Gridsim [29]

The fourth layer consists of Grid resource brokers or schedulers for simulation of resources. The final layer is focused on jobs and resource creating with different scenarios for executing scheduling and resource management policies, heuristics, and algorithms.

- **Sequence Diagram**
  In Gridsim simulation, all the entities and the interaction between them by the use of events are shown in Figure 5.2. As Gridsim starts, the resource entities register themselves by sending the events with the Grid information

system service. User entity submits all the tasks and their information to provisioning manager entity. Provisioning manager entity sends an event to Grid Information service entity to inquiry about resource. Grid Information service entity gives the list of all the registered resources and their information. Provisioning manager entity sends an internal synchronous event of task and resource provisioning according to set parameters. On the basis of the resource & task selection according to scheduling policy, the provisioning manager entity sends events (asynchronous) in order to submit the Gridlets to resource entities for execution because the provisioning manager has no need to wait for a resource to complete the allotted task. When the Gridlet execution is finished, the resource entity updates the status of Gridlet and processing time, energy consumed and cost spent and sends it to the provisioning manager by sending an event to show its completion.

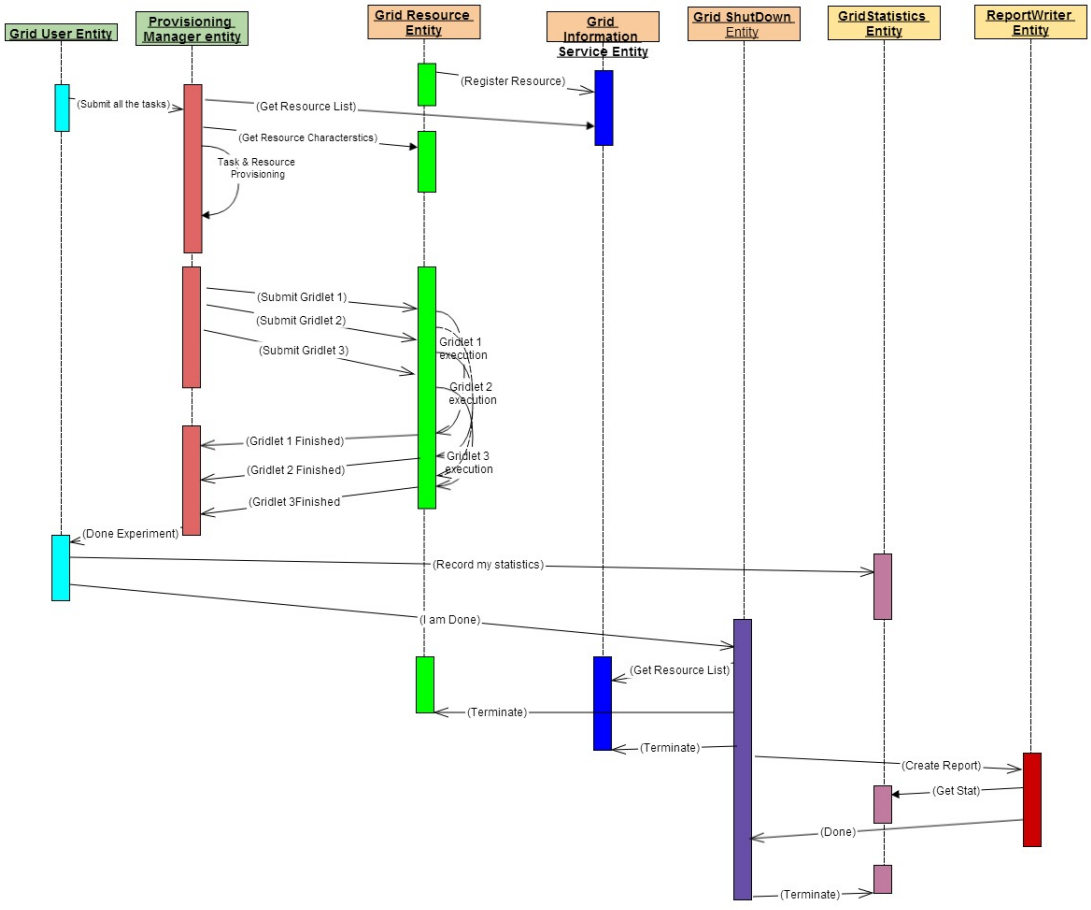

Figure 5.2: Sequence diagram to show working of Gridsim

The Gridsim resource sends asynchronous internal events to simulate the resources or to change the resource according to provisioning and requirement of task. Grid user entity sends an event to Grid statistics entity to record it statistics. GridShutDown entity sends terminate event to Grid resource and

Grid information service entity. GridShutDown entity sends create report event to ReportWriter entity. When it gets report as a result then it sends terminate event to GridStatistics entity [29].

5.1.2 NetBeans

NetBeans is an integrated development environment and also an application platform framework to develop the applications in Java. It can run on different operating systems like Windows, OS X, Linux, Solaris and other platforms. NetBeans provides many features which are menus and toolbars to develop user interface, window and storage management, wizard framework, NetBeans library. NetBeans IDE is free, open source, platform independent and provides support for Java programming language. NetBeans IDE 6.8 has been used for loading and using Gridsim toolkit [55].

5.1.3 Microsoft Access

Microsoft Access is a database management system given by Microsoft. It combines the relational Microsoft Jet Database Engine with a software development tools and graphical user interface. It is the part of Microsoft Office suite of applications [56]. All the data about tasks and resources are stored in this database. Using ODBC drivers Microsoft Access database is connected with Java application to submit the tasks and its information.

5.2 Implementation of the Proposed System

In this first implementation of Grid User Interface (Grid Portal) then implementation of EEHP algorithm is described.

5.2.1 Grid User Interface

Grid Portal is a user interface that offers various services for executing tasks on the Grid. Developers can access to their interface by Login into admin and users can access their interface by Login into User. The Grid portal interface is shown in Figure 5.3. Registered and authorized Users are only allowed to access the services. The services that it offers are Task its Specification Submission Service, Resource Management Service and Simulation Report Service [57].
The services that Grid portal provides are:

Figure 5.3: Grid Portal

5.2.1.1 Task & its Specification Submission Service

Task Submission service is a service for submitting the task and task specifications to execute it on the grid. Task specifications include metrics for defining the task so that its priority can be calculated to determine which type of resources can be used for executing that task. It also offers a functionality that allows a user to delete the record of tasks that are previously submitted by users (as shown in Figure 5.3).

(i) Add Task

Users can add the task by selecting it from the directory to perform it on the grid (as shown in Figure 5.4).

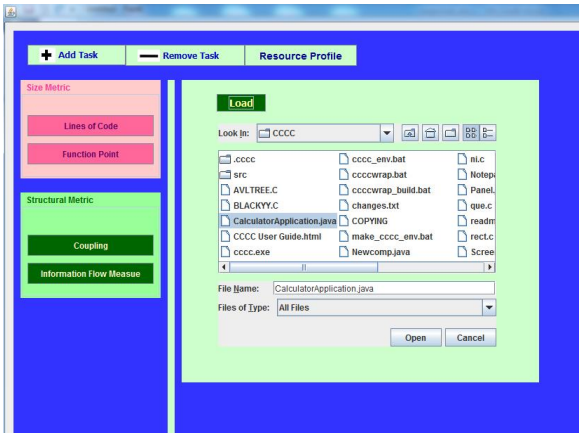

Figure 5.4: Load the Task

(ii) Add Task Specification

Add Task Specification service provides different metrics to define the task and to set its priority. Tasks can be ranked or sorted on the basis of priority to execute them on grid resources so that energy consumption can be reduced as well as performance can also be maintained. Metrics that are required to measure the priority of task are provided on the interface (shown in Figure 5.5).

(i) Size Metrics

- Lines of Code interface allow a user to enter LoC of task as shown in Figure 5.5.

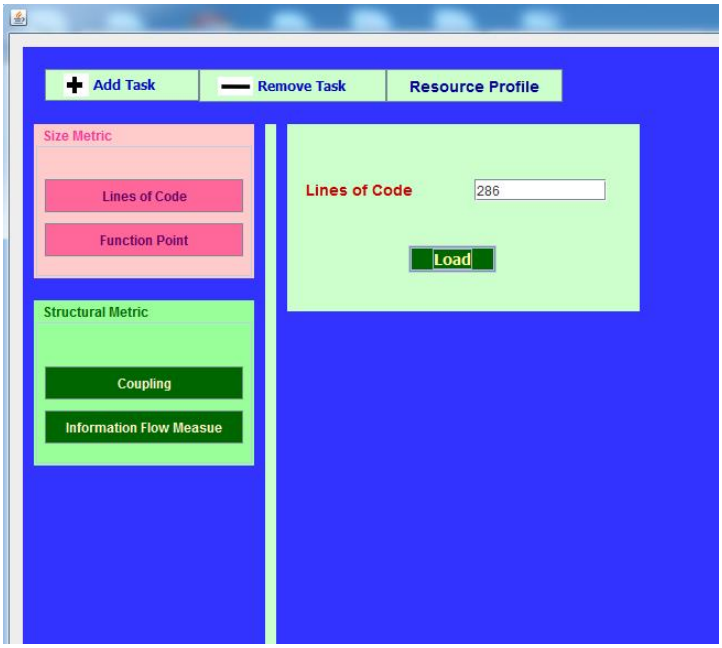

Figure 5.5: Line of Code Interface

- Function Point interfaces allows a user to enter domain characteristics and complexity factors of a task to calculate the function point metric as shown in Figures 5.6 & 5.7

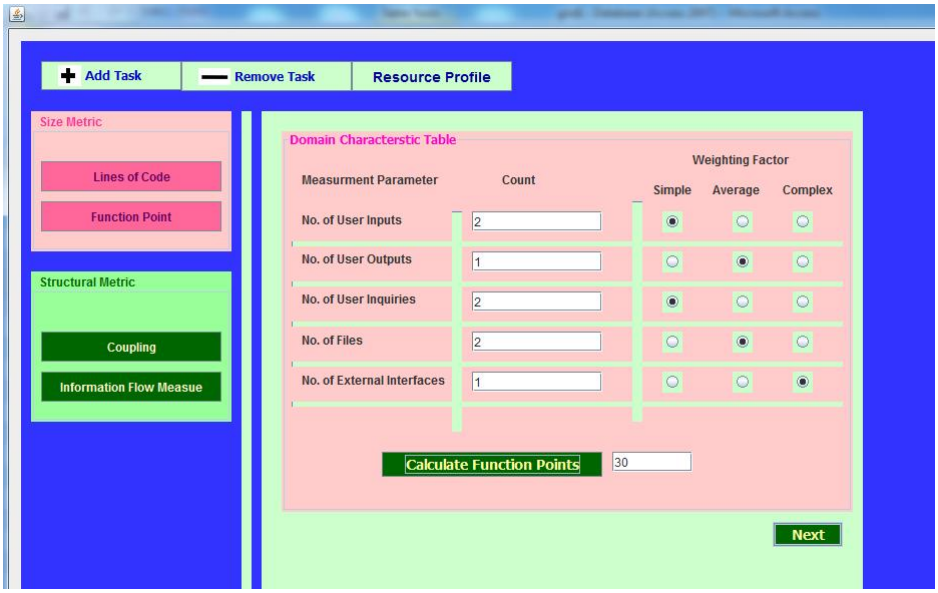

Figure 5.6: Function Point Interface

Figure 5.7: Function Point Technical Complexity Factors Interface

(ii) Structural Metrics: They are based on the relationships of a module with other modules. It includes Information Flow measure which uses Fan-in Fan-out to find a measure of coupling for the module.

- Coupling interface is provided to the user to enter coupling between object metric of a task as shown in Figure 5.8.

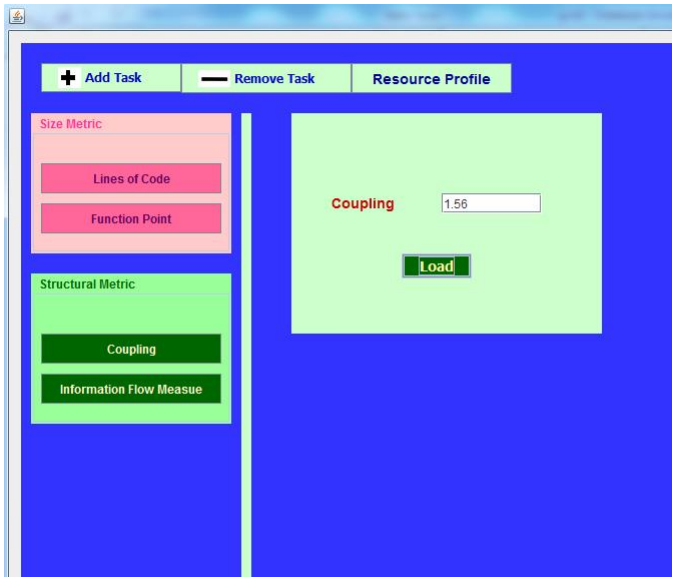

Figure 5.8: Coupling Interface

- Information Flow Measure interface is provided through which user can submit Fan-in, Fan-out and Information Flow. It is shown in Figure 5.9.

Figure 5.9: Information Flow Measure Interface

(iii) Remove Task

It allows a user to delete the record of the tasks that are submitted by user previously. It will update the Grid Portlet database automatically. User can select the task from the list and click on delete button then it will delete the task as shown in Figure 5.10.

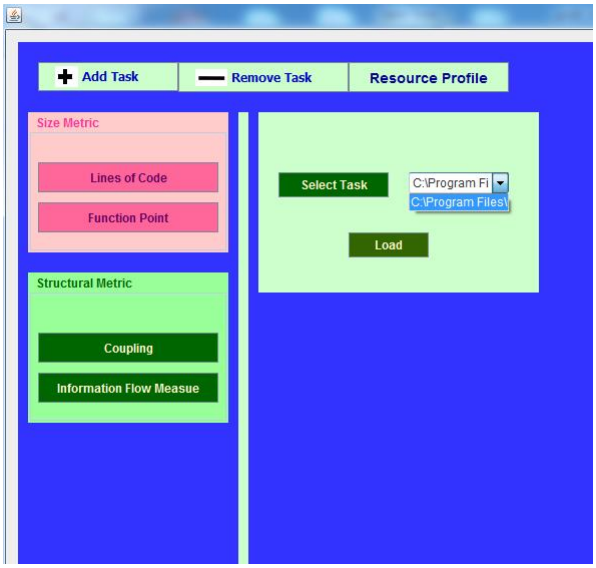

Figure 5.10: Remove Task

(iv) Resource Profile Service

It enables users to view the resources that have been registered on grid as shown in Figure 5.11. All the information of resources like number of cores, Mips Rating, CPE and Power consumption rate is available on clicking resource profile.

Figure 5.11: Resource Profile Interface

(v) Resource Management Service

It allows the developers to manage the resources by modifying, updating and deleting them. It includes following services:

- Modify Resource attributes: It allows the developers to easily customize the information of resources that is displayed to user. It will show a list box to allow the developer to select the particular resource to which they want to customize as shown in Figure 5.12.

Figure 5.12: Modify Resource Attributes Interface

- Remove any Resource: It allows the developers to delete a record of resource that is no more available to provide services to the user. Grid Portlet database will update automatically as it is shown in Figure 5.13.

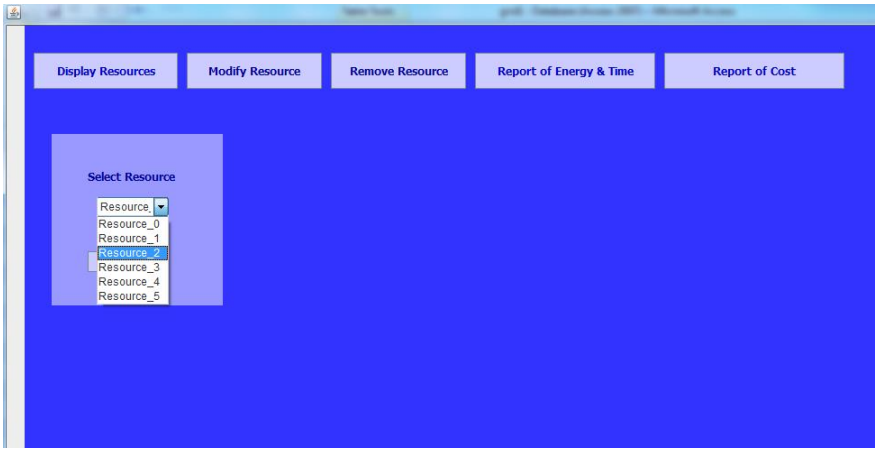

Figure 5.13: Remove Resource Interface

(vi) Simulation Output Service:
Simulation Report service allows the developers to check and see the result of simulation as how much energy is consumed, how much cost is spent and how much execution time is taken by a particular resource. It is shown in Figure 5.14. Allowing the developer to check these statistics will help them to analyze the consumption and they can maintain the tasks and resources according to that.

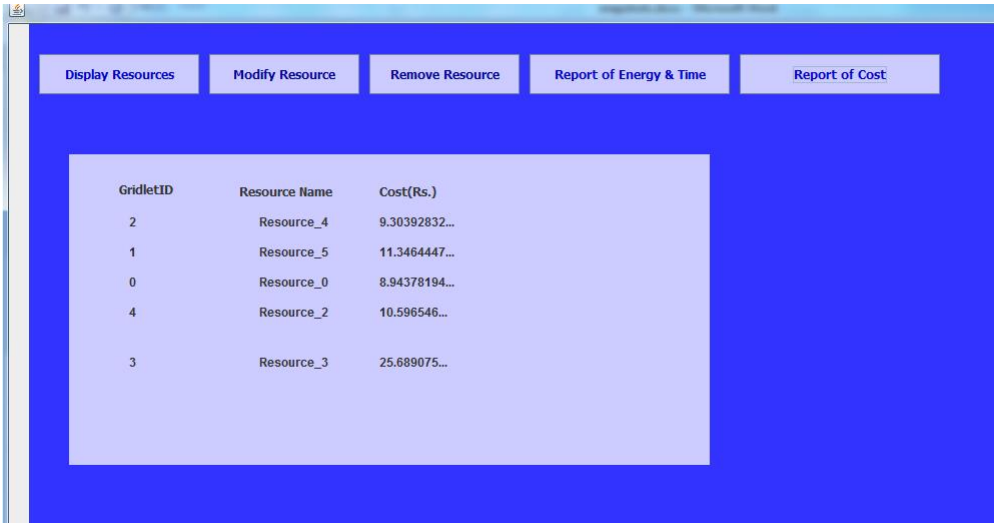

Figure 5.14: Simulation Output

5.2.2 Implementation of EEHP Algorithm

Gridsim [26] is used to simulate the Grid environment consisting of five most energy efficient resources as given in Table 4.2 above. For each resource their configuration, Clock rate, MIPS rating and power consumption rate is given. Joulesort benchmark defined them as most energy and performance efficient resources.

(i) Create Resources

In Gridsim, five resources are created that are mentioned in Table 4.2 as shown in Figure 5.15. For each resource their configuration, clock rate, MIPS rating and power consumption rate are specified as their parameters. createGridResource() function is used in Gridsim to create the resources and then resources are added in the machine list with their corresponding parameters.

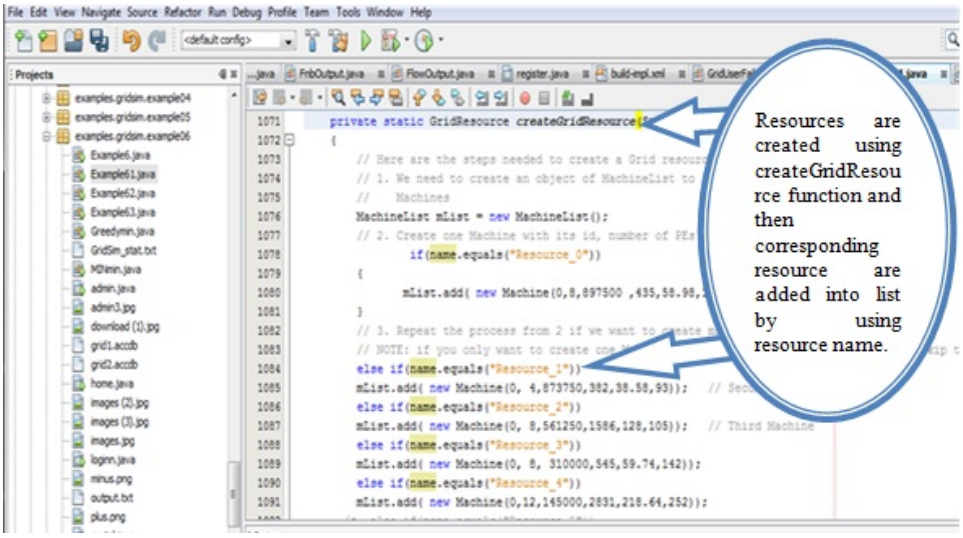

Figure 5.15: Create resources in Gridsim

(ii) Create Gridlets

Gridlets or tasks that are submitted by users through Grid portal are stored in database. Then through database connectivity, Gridsim access the submitted tasks and their information and creates the Gridlets as shown in Figure 5.16.

(iii) Tasks Prioritization

Tasks that are submitted by users for execution are stored in database along with their configuration. All the submitted tasks are accessed and analyzed using parameters defined in equation 1 in chapter 4 and then prioritized on the basis of their calculated weight stored in output_size array by using bubble sorting. Task Prioritization is essential because task i.e. responsible for high energy consumption is needed to be scheduled in that way so that energy consumption can reduce without degrading the performance.

- Tasks Size

  To prioritize the tasks, first of all size of tasks are extracted from the database and saved in the array named output_size[] as shown in Figure 5.17. Size of tasks is calculated using LoC and Function Point.

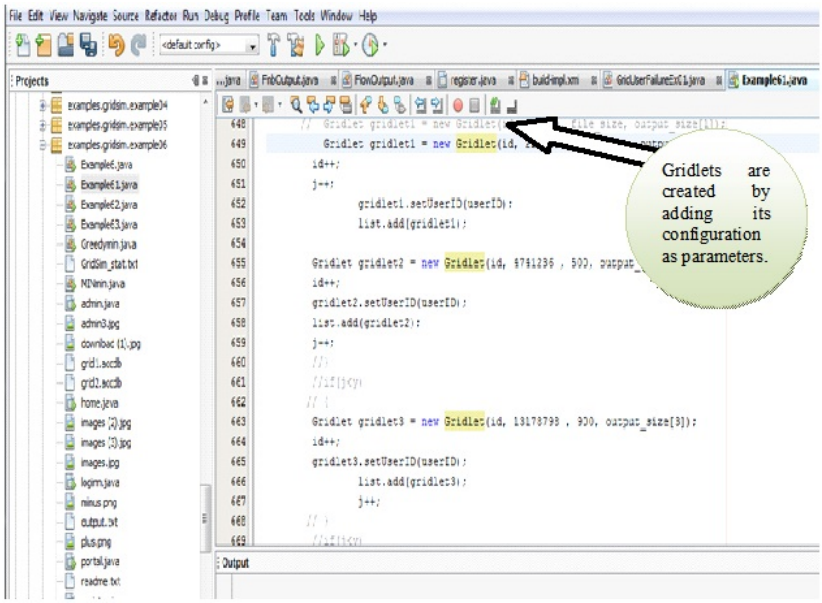

Figure 5.16: Create Gridlets in Gridsim

Function Point also includes technical complexity factors. So complexity of tasks is also considered in the evaluation of priority of tasks.

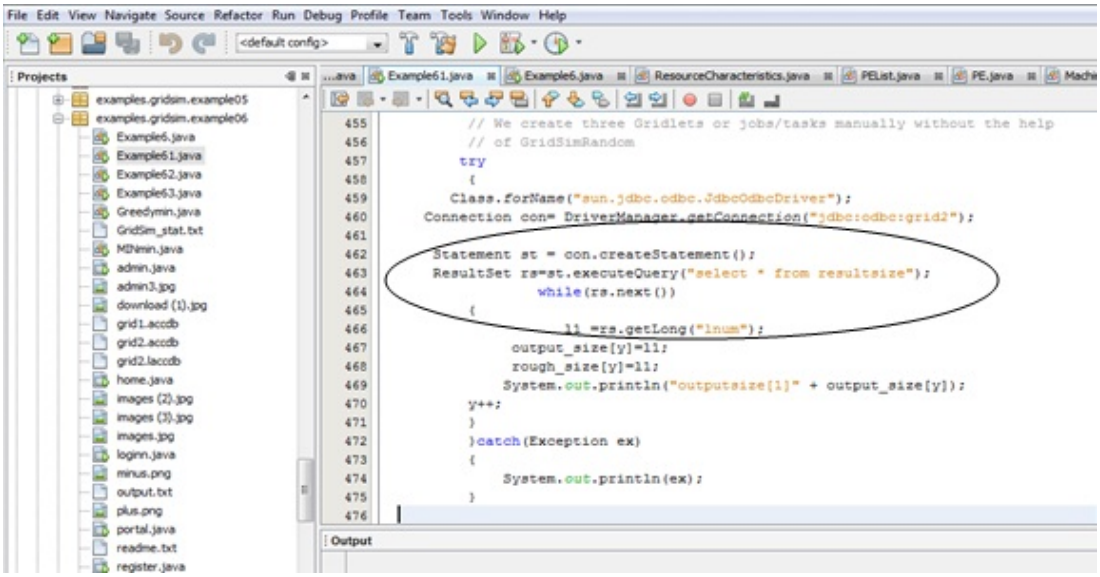

Figure 5.17: Extraction Size of Tasks from Database

- Tasks Dependency

  Tasks dependency i.e. submitted by users as tasks specification using Information flow metric is extracted from the database and added into the already created array named output_size[] as shown in Figure 5.18. In this way, by updating the output_size[] array we are calculating the weight of the tasks as stated in equation 1.

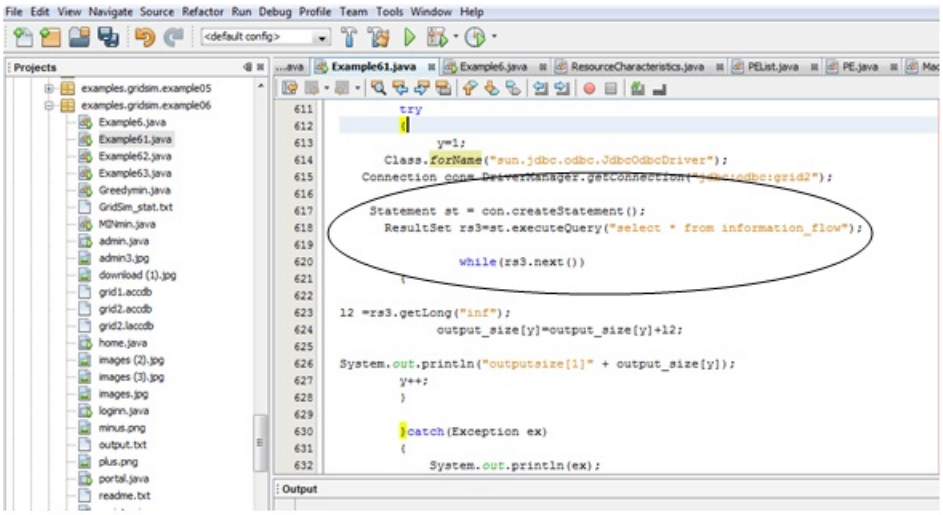

Figure 5.18: Extraction Dependency of Tasks from Database

- Variance of Energy

  Past data of tasks related to energy consumption is stored in the database on the execution of tasks using resources of different efficiency. Variation between the energy consumption value on high efficient resource and least efficient resource is calculated and stored in database. Variation in Energy consumption value of tasks is estimated by matching the tasks configuration. So, variance in energy values for tasks is extracted from database and multiplied with already created array output_size[] according to equation 4.1. It is shown in Figure 5.19.

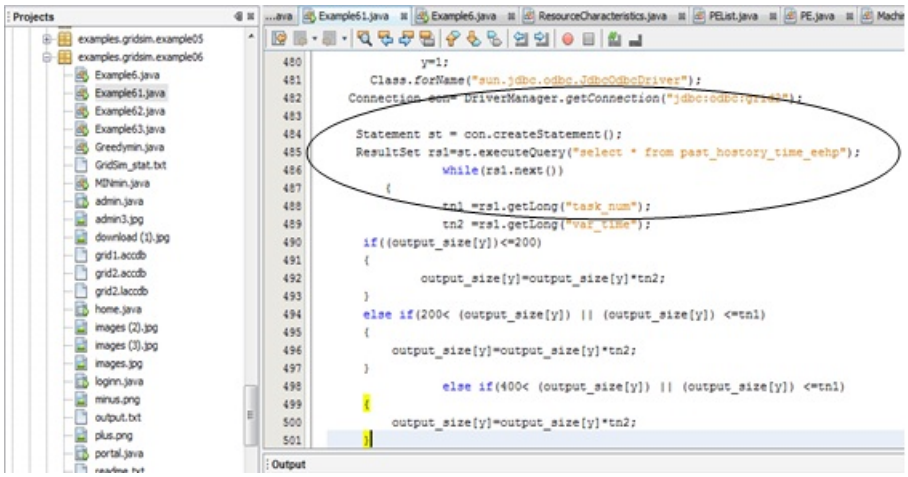

Figure 5.19: Extraction Variance of Energy for Tasks from Database

- Variance of Execution Time

  Past data of tasks related to execution time is stored in the database on the execution of tasks using resources of different efficiency. Variation between the execution time value on high efficient resource and

least efficient resource is calculated and stored in database. Variation in execution time value of tasks is estimated by matching the tasks configuration. So, Variance in execution time values for tasks is extracted from database and multiplied with already created array output_size[] according to equation 4.1. It is shown in Figure 5.20.

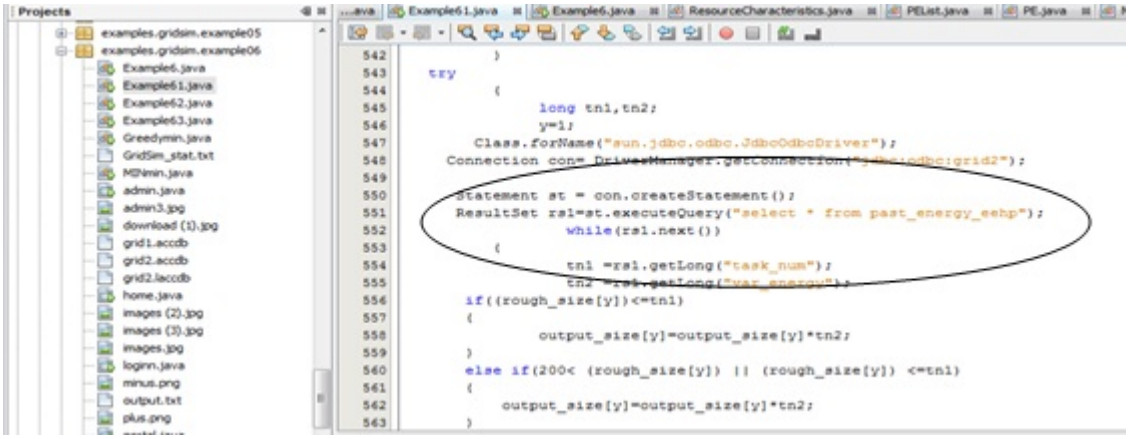

Figure 5.20: Extraction Variance of Time for Tasks from Database

(iv) Sorting of Tasks

Calculated weight of all the tasks are stored in array output_size[]. Using getGridletOutputSize() method calculated weight of tasks are extracted and store as second dimension of two dimension max_arr[][]array. Then tasks are sorted using Bubble sorting as shown in Figure 5.21.

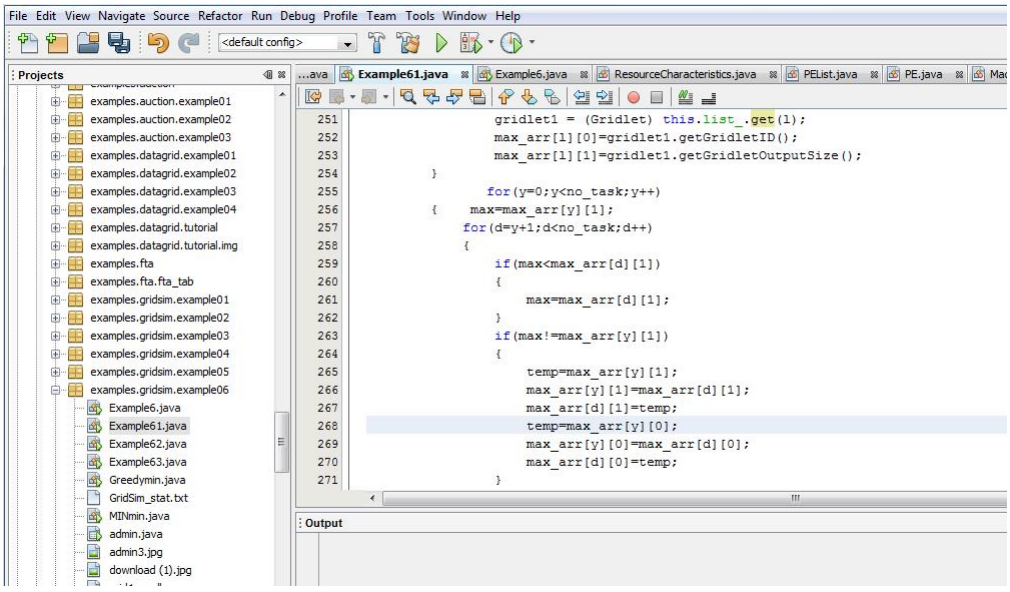

Figure 5.21: Task Prioritization in Gridsim

(v) Mapping of Gridlets (tasks) to the Resources

Task is selected from the sorted list of complex and dependent task. Then

selected task is scheduled to the most efficient resource for execution. It is shown in Figure 5.22.

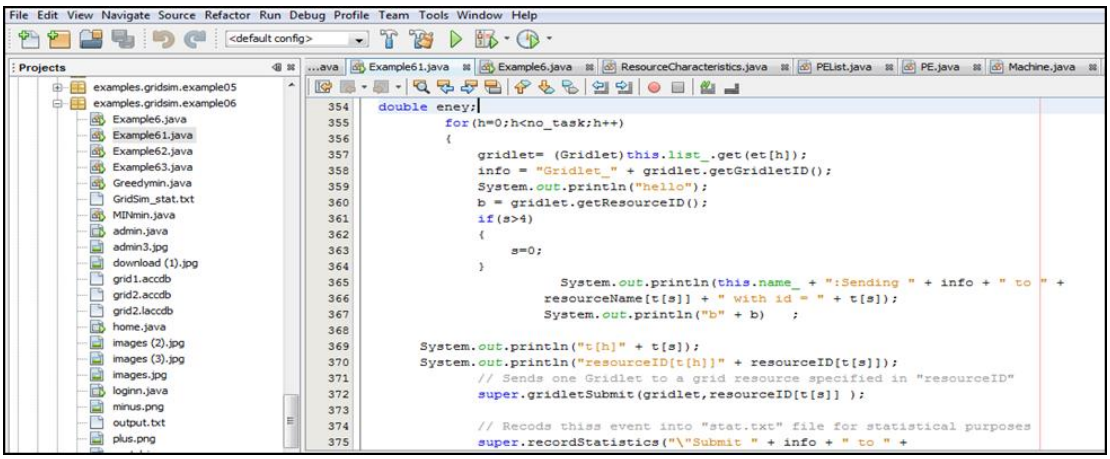

Figure 5.22: Sending Gridlets to Resources

(vi) Energy and Time calculation Execution time is calculated using getActual-CPUTime() function and energy is calculated by multiplying the power with time as shown in Figure 5.23.

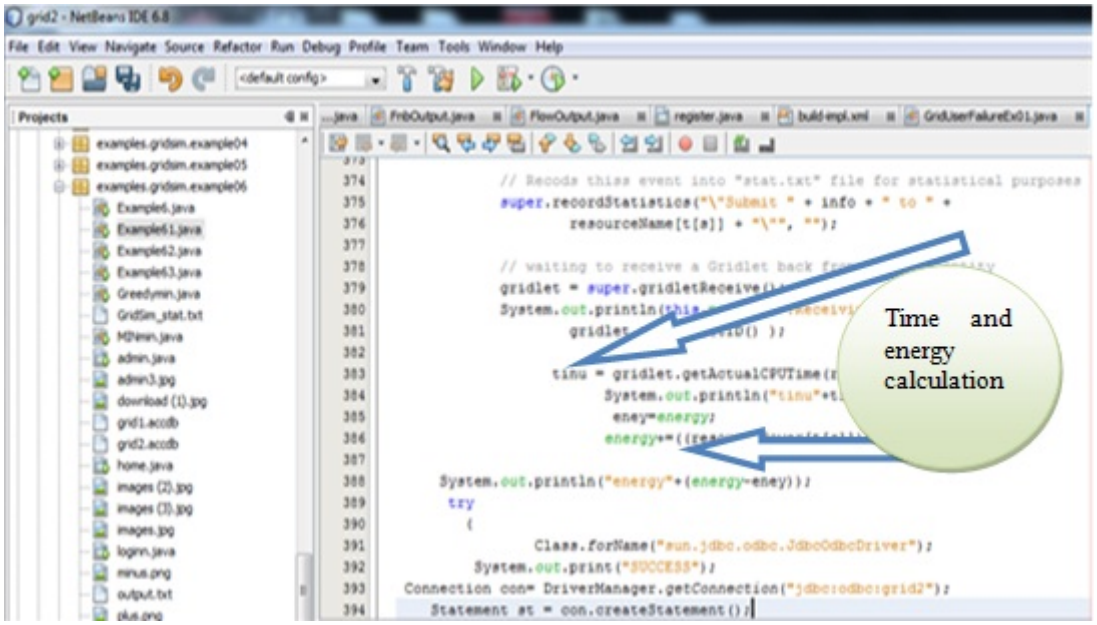

Figure 5.23: Energy and time calculation in Gridsim

5.3 Simulation Result

Tasks that are given in Table 4.1 are prioritized using equation 4.1 and then according to their weight they are mapped to the energy and performance efficient resources which are found out by Joulesort benchmark as given in Table 4.2.

Table 5.1 and Figure 5.24 show the simulation result for EEHP algorithm. It shows the results i.e. energy consumption in joules, execution time and cost spent on the tasks.

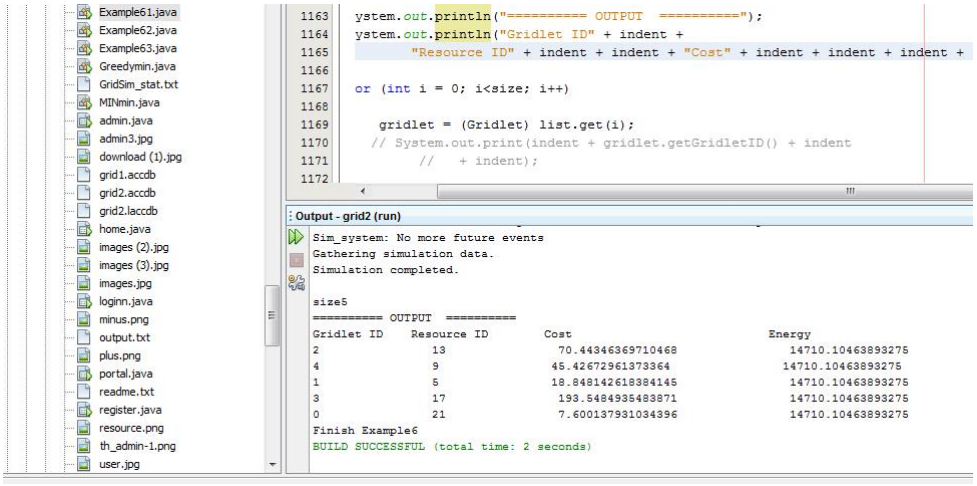

Figure 5.24: Simulation Result in Output Window of Gridsim

Table 5.1 shows the simulation result of EEHP algorithm on Energy-efficient resources found out by Joulesort benchmark. As shown in Table 5.1, Gridlet 2 is the most complex and dependent task and it is scheduled to the most energy efficient Resource 0. Similarly all the Gridlets are scheduled and Execution time is calculated in seconds, Cost in Rupees and Energy in Joules. Table 5.3 shows the simulation result of EEHP algorithm on other resources. On comparing these results it can be concluded that vast amount of energy can be saved as well as performance can be improved by using highly efficient resources.

Table 5.1: Simulation Results for EEHP Algorithm

Tasks	**Execution time (Secs)**	**Cost(Rs)**	**Energy(watt-secs(Joules))**	**Resource**
Gridlet 2	15.683	47.051	2587.843	Resource 0
Gridlet 4	15.1422	45.426	1408.228	Resource 1
Gridlet 1	8.4476	25.342	887.0018	Resource 2
Gridlet 3	64.5161	193.548	9161.295	Resource 3
Gridlet 0	2.5333	7.6001	638.4115	Resource 4
Total	106.3219	318.9671	14682.7793	

5.4 Comparative Analysis of EEHP algorithm with other algorithms

Comparison of EEHP algorithm with other existing algorithms which are HGreen algorithm, Greedy-min algorithm based on DVS approach and combined MINmin

heuristic approach is performed to analyze the energy, time and cost that can be saved by EEHP. For comparing the different algorithms, a set of resources has been considered as shown in Table 5.2. All these algorithms have been executed in Gridsim using these resources and compared on the basis of energy, time and cost.

Table 5.2: Resources List for Comparison [25]

Resources	**CPE**	**Mips Rating**	**Power consumption in watts (SPECpower benchmark)**
Resource 0	435	1174	219
Resource 1	382	1535	172
Resource 2	1586	2548	88
Resource 3	545	1190	139
Resource 4	2831	2902	60
Resource 5	681	1322	155

All these resources are sorted on the basis of energy-efficiency criteria. Resource which is highly energy efficient is assigned high priority to get the tasks first for execution. Sorting of these tasks is shown in Figure 5.25 .

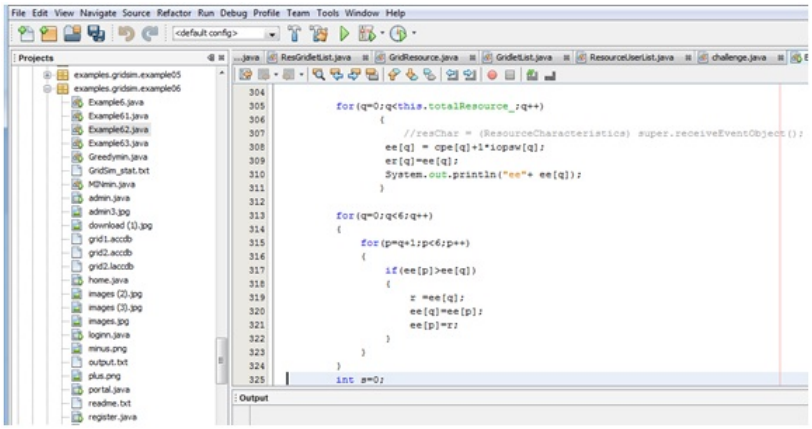

Figure 5.25: Sorting of Resources on the basis of Energy

Table 5.3 shows the simulation result of EEHP algorithm executed using Table 5.2 resources. As per EEHP algorithm complex and dependent tasks are scheduled first to the most energy-efficient resource. So the tasks given in Table 4.1 are scheduled with Table 5.2 resources using EEHP. According to EEHP, Gridlet 2 is the most complex and dependent task and it is scheduled to the most energy efficient resource i.e. Resource 4. Similarly all the Gridlets are mapped and Execution time, cost and energy are calculated.

Table 5.4 shows the simulation result of HGreen algorithm. HGreen algorithm says task with high energy consumption gets higher priority and scheduled first to the energy-efficient resources. Table 4.1 tasks are scheduled to Table 5.2 resources.

Table 5.3: Simulation Results of EEHP Algorithm (With Table 5.2 resources)

Tasks	**Execution time (Secs)**	**Cost(Rs)**	**Execution Energy (watt-secs(Joules))**	**Resource**
Gridlet 2	455.1343	1365.403	27308.063	Resource 4
Gridlet 4	4849.601	14548.804	426764.945	Resource 2
Gridlet 1	3586.411	10759.234	555893.7821	Resource 5
Gridlet 3	16807.731	50423.195	2504352.057	Resource 3
Gridlet 0	190.3867	571.1601	41694.689	Resource 1
Total	25889.264	77667.7961	3556013.536	

According to HGreen algorithm, Gridlet 2 is the highest energy consuming task so it is first scheduled to the most efficient resource. Similarly all the Gridlets are scheduled and all three parameter values are calculated.

Table 5.4: Simulation Results of HGreen Algorithm

Tasks	**Execution time (Secs)**	**Cost(Rs)**	**Execution Energy (watt-secs(Joules))**	**Resource**
Gridlet 2	454.134	1362.403	27248.063	Resource 4
Gridlet 1	1861.767	5585.302	163835.554	Resource 2
Gridlet 0	169.184	507.553	26223.608	Resource 5
Gridlet 3	16806.731	50420.195	2504352.057	Resource 3
Gridlet 4	10525.370	31576.111	2305056.145	Resource 1
Total	29817.186	89451.564	5026715.282	

Table 5.5 shows the simulation result of Greedy-min heuristic i.e. a DVS approach. Greedy-min heuristic approach says task with shortest completion time first gets higher priority and assigned to most energy efficient resource.

Table 5.5: Simulation Results of Greedy-Min

Tasks	**Execution time (Secs)**	**Cost(Rs)**	**Execution Energy (watt-secs(Joules))**	**Resource**
Gridlet 0	77.616	232.8483	4656.967	Resource 4
Gridlet 2	517.228	1551.685	45516.100	Resource 2
Gridlet 1	3586.411	10759.234	555893.782	Resource 5
Gridlet 4	10383.852	31151.558	1547194.088	Resource 3
Gridlet 3	17035.784	51107.353	3730836.804	Resource 1
Total	31600.891	94802.6783	5884097.741	

Table 5.6 shows the simulation result of Combined MINmin heuristic i.e. an approach for multicore heterogeneous Grid Computing environment. It first selects a pair of task and machine that minimizes the makespan then from resulting pairs selects that pair which minimizes energy consumption. All these algorithms attempt to reduce the energy consumption and improve performance.
Figures 5.26, 5.27 and 5.28 show EEHP and other existing algorithms with highly comparable results, namely HGreen algorithm, Greedy-Min heuristic and Com-

Table 5.6: Simulation Results of Combined MINmin Heuristic

Tasks	Execution time (Secs)	Cost(Rs)	Execution Energy (watt-secs(Joules))	Resource
Gridlet 0	77.616	232.8483	4656.967	Resource 4
Gridlet 2	517.228	1551.685	45516.100	Resource 2
Gridlet 1	3088.753	9266.259	531265.532	Resource 1
Gridlet 4	10384.852	31154.558	1547343.088	Resource 5
Gridlet 3	15128.601	45385.804	2344933.211	Resource 3
Total	29197.05	87591.1543	4473714.898	

bined MIN-min heuristic. It depicts the comparative analysis of EEHP algorithm with others in graphical form and also shows how EEHP algorithm is performed better than other algorithms in terms of both energy consumption and performance considered in terms of cost and time. In this comparison (five) tasks are taken on x-axis and corresponding comparing parameter on y-axis.

5.4.1 Comparison in terms of Energy

Figure 5.26 shows the energy consumption for 5 task problems. It can be noticed that EEHP has the lowest energy consumption. Combined heuristic MINmin is the only other heuristic that consumes less energy. But EEHP consumes 20.51 % energy lower than the MINmin heuristic. Greedy-Min consumes highest energy which is 39.56 % more than EEHP algorithm. HGreen consumes 29.29 % more energy than EEHP algorithm.

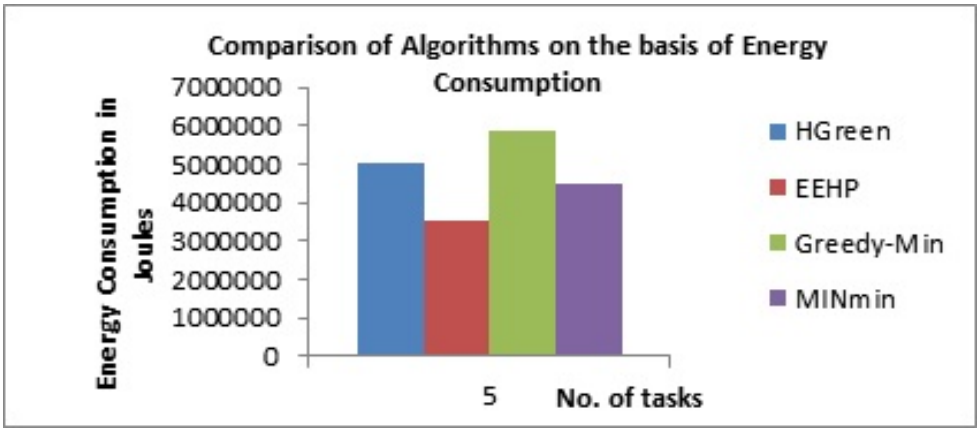

Figure 5.26: Comparison in terms of Energy

5.4.2 Comparison in terms of Execution Time

Figure 5.27 depicts the execution time of EEHP algorithm and other existing algorithms. EEHP is significantly better than others. Greedy-min performed poorly

compared to the other. EEHP performs 12 % better than combined MINmin heuristic and 13.17 % better than HGreen Algorithm in terms of execution time.

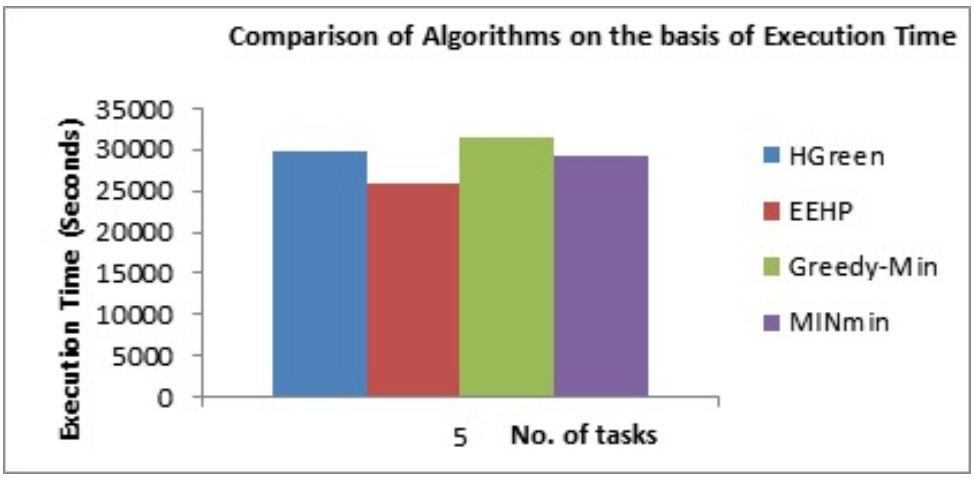

Figure 5.27: Comparison in terms of Execution Time

5.4.3 Comparison in terms of Cost

Figure 5.28 represents comparison of EEHP algorithm with other algorithms in terms of cost. EEHP obtained cost 11.32 % lesser than combined MINmin heuristic and 13.17% lesser than HGreen algorithm. Greedy-Min again performed poor in case of cost.

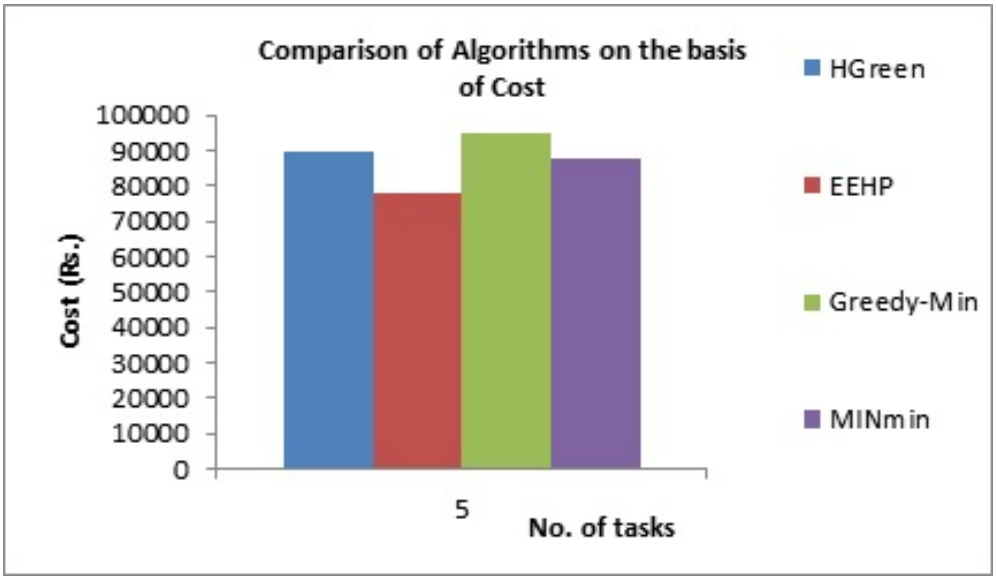

Figure 5.28: Comparison in terms of Cost

5.5 Conclusion

This chapter presented experimental results of implementation of EEHP algorithm. Simulation results and comparative analysis validates that Energy-efficient and High Performance Scheduling algorithm can save vast amount of energy without degrading the performance by using Joulesort benchmark resources.

Chapter 6

Conclusions and Future Scope

This chapter concludes the research work done in this thesis, defines thesis contributions along with future research directions.

6.1 Conclusions

The objective of this work was to reduce the power consumption without degrading the performance in global Grids. For achieving this challenge, resources that are proven as energy and performance efficient by joulesort benchmark have been used and tasks are prioritized by considering complexity of tasks, dependency within the task modules and variance of energy consumption and execution time of task on different resources.
Simulation results have shown that Energy-efficient and high performance algorithm reduces the energy consumption without degrading the performance. On average, EEHP saves 30% energy and performs 12% better than other existing algorithms.

6.2 Thesis Contributions

(i) In this thesis, existing energy and performance aware scheduling algorithms have been analyzed and compared according to energy and performance factors. Performance factors include execution time and cost.

(ii) This thesis work proposes an Energy Aware Layered Grid Architecture and an Energy-efficient and High Performance algorithm (EEHP) for energy and performance effective mapping of tasks to the resources. It is different from other energy aware approaches because of following aspects:

- It uses JouleSort benchmark to select the resources. JouleSort is a

system-level benchmark for energy efficiency that is useful across many types of systems.

- It provides Grid user interface to provide easy access to the users for the submission of tasks.
- It considers performance along with energy consumption because it includes Performance and Energy provisioning both at second layer in Energy Aware Layered Grid Architecture. Execution time and cost are the factors that are considered to measure the performance.
- It considers the dependency within the task to prioritize it.
- It conceives the variance of energy and time i.e. the difference between energy consumption and execution time of tasks by most energy efficient and least energy efficient resources.

(iii) Experimental results which have been collected and compared with other existing algorithms reveal that this approach results in the significant reduction of energy consumption without degrading the performance.

6.3 Future Work

- This work considers complexity and dependency factors to prioritize the tasks so that energy can be reduced. For future work, it is intended to improve the proposed technique by including more efficient factors for task prioritization.
- This work considers execution time to maintain the Quality of Service requirements of users. For future work, it is intended to include more QoS factors like reliability and security to improve it further.
- This work presents the static scheduling algorithm. For future work, it is intended to design dynamic scheduling algorithm which will choose not only the next job to be executed but also the speed at which job is to be executed.
- It is also foreseen to perform real tests instead of simulations.
- Calculation of Dependency of tasks can also be automated.

References

[1] I. Foster,"What is the Grid? A Three Point Checklist," in Argonne National Laboratory University of Chicago, 2002.

[2] F. Dong and S. G. Akl,"Scheduling algorithms for grid computing: State of the art and open problems," School of computing, Queen's University, Kingston, Ontario, 2006.

[3] B. Wilkinson,"Grid Computing," 2004. [Online]. Available: http://www.it.uom.gr/teaching/unc charlottePPG/Grid.htm. [Accessed 20 November 2013].

[4] M. Prashar and C. A. Lee,"Grid Computing: Introduction and Overview," *in Proceedings of the IEEE,special issue on grid computing*, 2005.

[5] "What-is-the-grid," [Online]. Available: http://www.Gridcafe.org. [Accessed 16 Novemeber 2013].

[6] L. M. Bote-Lorenzo, A. Y. Dimitriadis and E. Gomez-Sanchz,"Grid characteristics and uses: a Grid definition," *in Grid Computing, Springer Berlin Heidelberg*, 2004, pp. 291-298.

[7] I. Foster and C. Kesselman,"Computational Grids," *in Cern European Organization for Nuclear Research=Reports-Cern*, pp. 87-114,1998.

[8] M. J. Litzkow, M. Livny, and M. W. Mutka,"Condor-a hunter of idle workstations," *in Distributed Computing Systems, 1988., 8th International Conference on. IEEE*, pp.104-111, 1988.

[9] C. Lee, C. Kesselman, and S. Schwab,"Near-realtime satellite image processing: Metacomputing in CC++," *in IEEE Computer Graphics and Applications*, vol.16, no.4, pp.79-84, 1996.

[10] M. Roussos, A. E.Johnson, J. Leigh, C. A.Vasilakis, C. R. Barnes T. G. Moher,"NICE: combining constructionism, narrative and collaboration in a virtual learning environment," *in COMPUTER GRAPHICS-NEW YORK-ASSOCIATION FOR COMPUTING MACHINERY-31*,pp. 62-63, 1997.

[11] I. Foster and C. Kesselman,"The Anatomy of the Grid: Enabling Scalable Virtual Oganizations," *in International Journal of high performance computing applications*, vol. 15, no. 3, pp. 200-222, 2001.

[12] K. Krauter, R. Buyya and M. Maheswaran,"A taxonomy and survey of Grid resource management systems for distributed computing," *in International Journal of Software Practice and Experience*, vol. 32, no. 2, pp. 135-164, 2002.

[13] W.-C. Chung and R. S. Chang,"A new mechanism for resource monitoring in Grid computing," *in Future Generartion Computer Systems*, vol. 25, no. 1, pp. 1-7, 2009.

[14] G. v. Laszewski,"Power-aware scheduling of virtual machines in dvfs-enabled clusters," *in Cluster Computing and Workshops,2009. CLUSTER'09, IEEE International Conference on*, 2009.

[15] M. Maheswaran,"Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *in Journal of Parallel and distributed computing*, vol. 59, no. 2, pp. 107-131, 1999.

[16] W.Forrest,"How to cut data centre carbon emissions?," 5 December 2008. [Online]. Available: http://www.computerweekly.com/Articles/2008/12/05/233748/how-tocut-data-centre-carbonemissions.htm. [Accessed 4 January 2014].

[17] J. Yu and R. Buyya,"Workflow Scheduling algorithms for grid computing," *in Metaheuristics for scheduling in distributed computing environments, Springer Berlin Heidelberg*, 2008, pp. 173-214.

[18] A. Abraham and R. Buyya,"Nature's heuristics for scheduling jobs on computational grids," *in The 8th IEEE international conference on advanced computing and communications (ADCOM 2000)*, 2000.

[19] F. Xhafa and A. Abraham,"Computational models and heuristics methods for grid scheduling problems," *Future generation computer systems*, vol. 26, no. 4, pp. 608-621, 2010.

[20] P. Lindberg,"Comparison and analysis of greedy energy-efficient scheduling algorithm for computational grids," *in Energy-Efficient Distributed Computing System*, 2011, pp. 189-214.

[21] Y. Chen,"Managing server energy and operational costs in hosting centers," *in ACM SIGMETRICS Performance Evaluation Review*, 2005.

[22] A. Verma and P. Ahuja,"Power-aware dynamic placement of HPC applications," *in Proceedings of the 22nd annual international conference on supercomputing. ACM*, 2008.

[23] M. Maheswaran,"Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," *in Heterogeneous Computing Workshop, 1999. (HCW'99) Proceeding. Eighth. IEEE*, 1999.

[24] S. Nnesmachnow,"Energy-Aware Scheduling on Multicore Heterogeneous Ggrid Computing Systems," *in Journal of Grid Computing*, vol. 11, no. 4, pp. 653-680, 2013.

[25] J.-K. Kim,"Dynamically mapping tasks with priorities and multiple deadlines in heterogeneous environment," *Journal of Parallel and Distributed Computing*, vol. 67, no. 2, pp. 154-169, 2007.

[26] C. Li and L. Li,"Utility-based scheduling for Grid computing under constraints of energy budget and deadline," *Computer Standards Interfaces*, vol. 31, no. 6, pp. 1131-1142, 2009.

[27] F. Coutinho and L. A. V. d. Carvalho,"A workflow scheduling algorithm for optimizing Energy-Efficient Grid resources usage," *in Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on. IEEE,*2011, 2011.

[28] "SPECpower," 2008. [Online]. Available: http://www.spec.org/power_ssj2008. [Accessed 28 December 2013].

[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, no. 13-15, pp. 1175-1220, 2002.

[30] A. Takefusa,"Performance Analysis of Scheduling and Replication Algorithms on Grid Datafarm Architecture for High-Energy Physics Applications," *HPDC*, vol. 3, pp. 34-47, 2003.

[31] M. Maheswaran,"Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *in Journal of Parallel and Distributed computing*, vol. 59, no. 2, pp. 107-131, 1999.

[32] R. Buyya,"Economic-based distributed resource management and scheduling for Grid computing," arXiv preprint cs/0204048, 2002.

[33] L. Wang,"Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS," *in Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on. IEEE*,2010.

[34] J. Kolodziej,"Hierarchical genetic-based grid scheduling with energy optimization," *Cluster Computing*, vol. 16, no. 3, pp. 591-609, 2013.

[35] C. H. Hsu and W.-c. Feng,"A feasibilty analysis of power awareness in commodity-based high-performance clusters," *in Cluster Computing*, 2005. IEEE International IEEE,2005.

[36] S. K. Garg and R. Buyya,"Exploiting heterogeneity in Grid Computing for energy-efficient resource allocation," *in Proceedings of the 17th International Conference on Advanced Computing and Communications*, 2009.

[37] F. Montes and A. ,"Smart Scheduling for saving energy in Grid computing," *Expert Systems with Applications*, vol. 39, no. 10, 2012.

[38] A. Flissi and P. Merle,"A generic deployment framework for grid computing and distributed applications," *in On the Move to Meaningful Internet Systems 2006: CoopIS,DOA,GADA, and ODBASE, Springer Berlin Heidelberg*, 2006, pp. 1402-1411.

[39] A. C. Orgerie and L. Lefevre,"Save watts in your Grid: Green startegies for energy-aware framework in large scale distributed systems," *in Parallel and Distributed Systems, 2008. ICPADS'08. 14h IEEE International Conference on. IEEE*, 2008.

[40] G. D. Costa,"The green-net framework: Energy efficiency in large scale distributed systems," *in Parallel and Distributed Systems, 2009. IPDPS 2009. IEEE, International Symposium on. IEEE*, 2009.

[41] W. Chuliang, and L. Xinda,"Heuristic scheduling for bag-of-tasks applications in combination with QoS in the computational grid," *in Future Generation Computer Systems*, vol. 21, no. 2, pp. 271-280, 2005.

[42] G. Booch, J.Rumbaugh and I.Jacobson, The Unified Modeling Language User Guide, Addison Wesley Longman Publishing Co., Inc., 2005.

[43] "CCCC," [Online]. Available: http://cccc/.sourceforge.net/. [Accessed 21 February 2014].

[44] "LOC," [Online]. Available: http://www.projectcodemeter.com/cost_estimation/help/GL_sloc.htm. [Accessed 15 february 2014].

[45] A. Heydarnoori, "Function Point," [Online]. Available: https://cs.uwaterloo.ca/ apidduck/CS846/Seminars/abbas.pdf. [Accessed 16 February 2014].

[46] R. Pressman, Software Engineering: A practitioner's Approach, McGraw Hill, 2005.

[47] J. Hu and R. Marculescu,"Energy and performance aware mapping for regular NoC architectures," *in Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2005.

[48] P. Goodman, Software Metrics-Best Practices for successful IT management, Rothstein Associates Inc., 2005.

[49] R. Suzanne,"JouleSort: a balanced energy-efficiency benchmark," *in Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM*,2007.

[50] A. Beckmann,"Energy-efficient sorting using solid state disks," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 2, pp. 151-163, 2011.

[51] N. Agarwal,"Design Tradeoffs for SSD Performance," *in USENIX Annual Technical Conference*, 2008.

[52] D. J. Lilja, Measuring Computer Performance: A Practititoner's Guide, Cambridge University Press, 2004.

[53] P. Pillai and M. Kaminsky, "FAWNSort: Energy-efficient sorting of 10GB, 100GB and 1TB," Intel Labs, Carnegie Mellon University.

[54] "Joulesort Benchmark," [Online]. Available: http://sortbenchmark.org/. [Accessed 25 March 2014].

[55] "NetBeans," [Online]. Available: http://en.wikipedia.org/wiki/NetBeans. [Accessed 12 July 2013].

[56] "Microsoft Access," [Online]. Available: http://en.wikipedia.org/wiki/Microsoft_Access. [Accessed 14 October 2013].

[57] M. Russell, J. Novotny and O. Wehrens, "Gridsphere's Grid Portlets," *Computational Methods in Science and Technology*, vol. 12, pp. 89-97, 2006.

Appendix A

Installation of Gridsim 5.2 toolkit

Gridsim 5.2 toolkit is easily downloadable from http://www.cloudbus.org/Gridsim/ . Steps to run Gridsim:

1. Install Netbeans.
2. Open Gridsim 5.2 as a new project in Netbeans.
3. Then add jar folders by right clicking on Library folder and add Gridsim.jar and simjava2.jar folders that contains built in classes which are required for implementation as shown in snapshot below.
4. Then algorithm is implemented. If any changes is made in built in classes then clear and build the jar folders again and then copy that new compiled jar folders in Gridsim folder.

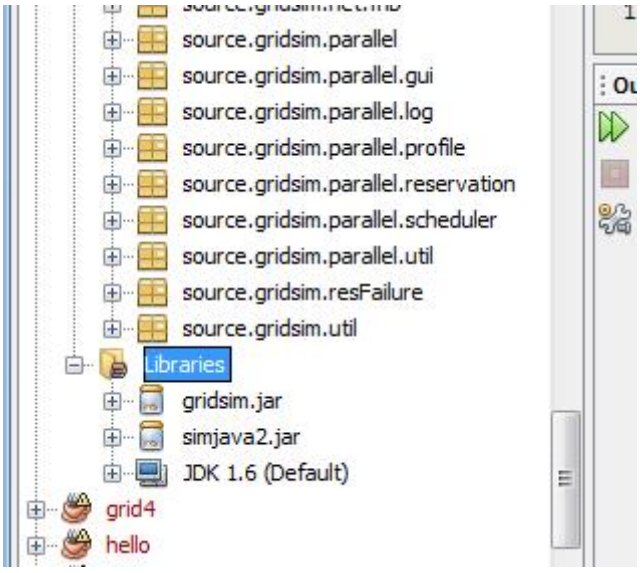

Figure A.1

Appendix B

Installation of CCCC tool to get metrics information of tasks

CCCC i.e. C and C++ code counter tool is easily downloadable from http://sourceforge.net/projects/cccc/.

1. Simply Install CCCC.exe by double clicking on it.
2. Then run the shortcut on desktop, then by using command cccc file_name execute the program on it to get its metrics as shown in figure below.
3. Then it generates the Html reports that contain all metrics information as shown in figure.

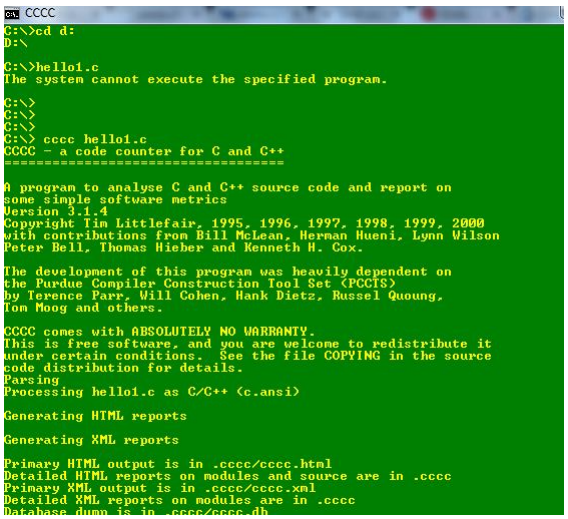

Figure B.1

Module Name	Fan-out			Fan-in			IF4		
	vis	con	inc	vis	con	incl	vis	con	inc
Grid	0	0	0	2	1	2	0	0	0
JFileChooser	1	0	1	0	0	0	0	0	0
JFrame	1	1	1	0	0	0	0	0	0
anonymous	0	0	0	0	0	0	0	0	0

Figure B.2

Metric	Tag	Overall	Per Module
Number of modules	NOM	4	
Lines of Code	LOC	85	21.250
McCabe's Cyclomatic Number	MVG	3	0.750
Lines of Comment	COM	0	0.000
LOC/COM	L_C	******	
MVG/COM	M_C	------	
Information Flow measure (inclusive)	IF4	0	0.000
Information Flow measure (visible)	IF4v	0	0.000
Information Flow measure (concrete)	IF4c	0	0.000
Lines of Code rejected by parser	REJ	12	

Figure B.3

List of Papers

Accepted

[1] Nidhi Jain and Inderveer Chana,"Energy-Efficient Scheduling Techniques in Grid Computing: A Systematic Review," International Conference on Communication and Computing (ICC - 2014), Bangalore, 12-14 June, 2014.

Communicated

[1] Nidhi Jain and Inderveer Chana,"Energy Efficient and High Performance Scheduling Algorithm for Grid Computing," Journal of Engineering with Computers,2014.