Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 3, No.7, 2012

An Efficient Scheduling Policy for Load Balancing Model for Computational Grid System

Mukul Pathak[1] Ajeet Kumar Bhartee[2] Vinay Tandon[3]
1, 2.Department of Computer Science & Engineering, Galgotias College of Engineering & Technology, Greater Noida (U.P.), India
1mukul.pathak2501@gmail.com
2ajeetkbharti@gmail.com
3.Department of Master of Computer Application, Aligarh College of Engineering & Technology, Greater Noida (U.P.), India
3vnay.tandon@gmail.com

Abstract
Workload and resource management are two essential functions provided at the service level of the Grid system. To improvement in global throughput need, effective and efficient load balancing are fundamentally important. We also check that what type of scheduling policy is used by that algorithm, because an efficient scheduling policy can utilize the computational resources efficiently by allowing multiple independent jobs to run over a network of heterogeneous clusters. In this paper, a dynamic grid model, as a collection of clusters has been proposed. An efficient scheduling policy is used, and its comparison with the other scheduling policy has been presented.

1. INTRODUCTION. In order to fulfil the user expectations in terms of performance and efficiency, the Grid system needs efficient load balancing algorithms for the distribution of tasks [1]. A load balancing algorithm attempts to improve the response time of user's submitted applications by ensuring maximal utilization of available resources. The main goal is to prevent, if possible, the condition where some processors are overloaded with a set of tasks while others are lightly loaded or even idle [2]. Although load balancing problem in conventional distributed systems has been intensively studied, new challenges in Grid computing still make it an interesting topic and many research projects are under way. This is due to the characteristics of Grid computing and the complex nature of the problem itself. Load balancing algorithms in classical distributed systems, which usually run on homogeneous and dedicated resources, cannot work well in the Grid architectures. In this chapter we define the motivation of this research and then identify the research questions. This chapter also discusses overall organization of thesis.

2. CHARACTERISTICS OF GRID

There are these main issues that characterize computational Grids [3, 4]:

• *Heterogeneity*: A Grid involves a multiplicity of resources that are heterogeneous in nature and might span numerous administrative domains across wide geographical distances.

- Resources are heterogeneous
- Resources are administratively disparate
- Resources are geographically disparate
- Users do not have to worry about system details (e.g., location, operating system, accounts).
- Resources are numerous.
- Resources have different resource management policies.
- Resources are owned and managed by different, potentially mutually distrustful organizations and individuals that likely have different security policies and practices.

• *Scalability:* A Grid might grow from few resources to millions. This raises the problem of potential performance degradation as a Grids size increases. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant.

• *Dynamicity or Adaptability*: In a Grid, a resource failure is the rule, not the exception [5]. In fact, with so many resources in a Grid, the probability of some resource failing is naturally high. The resource managers or applications must tailor their behaviour dynamically so as to extract the maximum performance from the available resources and services.

• *Parallel CPU execution*: One of most important feature of Grid is its scope for massive parallel CPU capacity. The common attribute among such uses is that the applications have been written to use algorithms that can be partitioned into independently running parts. A CPU intensive Grid application can be thought of as many smaller "sub jobs," each executing on a different machine in the Grid.[5] To the extent that these sub jobs do not need to communicate with each other, the more "scalable" the application becomes. A perfectly scalable application will, for example, finish 10 times faster if it uses 10 times the number of processors.

• *Virtual organizations*: The users of the Grid can be organized dynamically into a number of virtual organizations, each with different policy requirements [6, 7]. These virtual organizations can share their resources collectively as a larger Grid.

• *Resource balancing*: A Grid contains a large number of resources contributed by individual machines into a greater total virtual resource. For applications that are Grid enabled, the Grid can offer a resource balancing effect by scheduling Grid jobs on machines with low utilization [8].

• *Reliability and Management*: High-end conventional computing systems use expensive hardware to increase reliability. A Grid is an alternate approach to reliability that relies more on software technology than expensive hardware [51]. The goal to virtualized the resources on the Grid and more uniformly handle heterogeneous systems will create new
Opportunities to better manage a larger more disperse IT infrastructure.

3. GRID ARCHITECTURE

Architecture identifies the fundamental system components, specifies purpose and function of these components, and indicates how these components interact with each other. Grid architecture is protocol architecture, with protocols defining the basic mechanisms by which VO [9, 10, and 11] users and resources negotiate, establish, manage and exploit sharing relationships. Grid architecture is also a services standards based open architecture that facilitates extensibility, interoperability, portability and code sharing. The components that is necessary to form a Grid.

Grid Fabric: It comprises all the resources geographically distributed (across the globe) and accessible from anywhere on the Internet. They could be computers (such as PCs or Workstations running operating systems such as UNIX or NT), clusters (running cluster operating systems or resource management systems such as LSF, Condor or PBS), storage devices, databases, and special scientific instruments such as a radio telescope.

Grid Middleware: It offers core services such as remote process management, collocation of resources, storage access, information (registry), security, authentication, and Quality of Service (QoS) such as resource reservation and trading.

Grid Development Environments and Tools: These offer high-level services that allow programmers to develop applications and brokers that act as user agents that can manage or schedule computations across global resources.

Grid Applications and Portals: They are developed using Grid-enabled languages such as HPC++, and message-passing systems such as MPI. Applications, such as parameter simulations and grand-Challenge problems often require considerable computational power, require access to remote data sets, and may need to interact with scientific instruments. Grid portals offer web-enabled application services i.e., users can submit and collect results for their jobs on remote resources through a web interface.

4 LOAD BALANCING APPROACHES

Load balancing problem has been discussed in traditional distributed systems literature for more than two decades. Various algorithms, strategies and policies have been proposed, implemented and classified [12].

4.1 STATIC LOAD BALANCING ALGORITHM

Static load balancing algorithms allocate the tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on an average load of our workstation cluster. The decisions related to load balance are made at compile time when resource requirements are estimated. The advantage in this sort of algorithm is the simplicity in terms of both implementation as well as overhead, since there is no need to constantly monitor the workstations for performance statistics.

4.2 DYNAMIC LOAD BALANCING ALGORITHM

Dynamic load balancing algorithms make changes to the distribution of work among workstations at run-time; they use current or recent load information when making distribution decisions. Multicomputers with dynamic load balancing allocate/ reallocate resources at runtime based on no a priori task information, which may determine when and whose tasks can be migrated.

5. LOAD BALANCING STRATEGIES

There are two major strategies which usually used in load balancing algorithm will employ [17].

SENDER-INITIATED VS RECEIVER-INITIATED STRATEGIES

The question of who makes the load balancing decision is answered based on whether a sender-initiated or receiver-initiated policy is employed [13]. In sender- initiated policies, congested nodes attempt to move work to lightly-loaded nodes. In receiver-initiated policies, lightly-loaded nodes look for heavily-loaded nodes from which work may be received.

Figure shows the relative performance of a sender-initiated and receiver-initiated load balancing algorithm. As can be seen, both the sender-initiated and receiver-initiated policies perform substantially better than a system which has no load sharing.

The sender-initiated policy performing better than the receiver-initiated policy at low to moderate system loads. Reasons are that at these loads, the probability of finding a lightly loaded node is higher than that of finding a heavily-loaded node. Similarly, at high system loads, the receiver initiated policy performs better since it is much easier to find a heavily loaded node. As a result, adaptive policies have been proposed which behave like sender initiated policies at low to moderate system loads, while at high system loads they behave like receiver-initiated policies.

6. LOAD BALANCING POLICIES

Load balancing algorithms can be defined by their implementation of the following policies [14-15]:

- Information policy: specifies what workload information to be collected, when it is to be collected and from where.
- Triggering policy: determines the appropriate period to start a load balancing operation.
- Resource type policy: classifies a resource as server or receiver of tasks according to its availability status.
- Location policy: uses the results of the resource type policy to find a suitable partner for a server or receiver.
- Selection policy: defines the tasks that should be migrated from overloaded resources (source) to most idle resources (receiver).

7. PROBLEM STATEMENT

In grid environments, the shared resources are dynamic in nature, which in turn affects application performance. Workload and resource management are two essential functions provided at the service level of the Grid software infrastructure. To improve the global throughput of these environments, effective and efficient load balancing algorithms are fundamentally important. The focus of our study is to consider factors which can be used as characteristics for decision making to initiate Load Balancing. Load Balancing is one of the most important factors which can affect the performance of the grid application.
This thesis work analyzes the existing Load Balancing modules and tries to find out performance bottlenecks in it. All Load Balancing algorithms implement five policies [16]. The efficient implementation of these policies decides overall performance of Load Balancing algorithm.

The main objective of this paper is to propose an efficient Load Balancing Algorithm for Grid environment. Main difference between existing Load Balancing algorithm and proposed Load Balancing is in implementation of Scheduling policies: selected by Selection Policy. In my thesis we make a scheduling policy, which can be used more reliably to make decision about selection of job for migration from heavily loaded node to lightly loaded node.

8. PROPOSED METHODOLOGY

Load balancing is defined as the allocation of the work of a single application to processors at run-time so that the execution time of the application is minimized. This chapter is going to discuss the design of proposed Load Balancing algorithm.

9. PROPOSED GRID MODEL COMPONENTS

Cluster-Level consists of a collection of computing nodes. Cluster-Level manager (CM) can fully control the computing nodes within it, but cannot operate the computing nodes of other clusters directly. The computing nodes within the cluster are referred as friends. CM maintains the load information along with registration information of its computing nodes. In Cluster-Level, each friend runs a CM. CM role is to balance the intra-cluster workload. A designated friend with highest CPU speed in each cluster is treated as the cluster server or master. Cluster System Monitor (CS) determines the load index of computing nodes and provides this information to CM.

Grid-Level consists of a collection of interconnected clusters. Grid-Level manager (GM) is responsible for load control among its clusters as shown in. GM maintains the load information along with registration information of neighbouring masters in the grid. Neighbours for each cluster are formed in terms of communication costs. GM calculates the minimum communication cost of sending or receiving jobs to/from remote clusters based on the information collected in the last exchange interval. Master of each cluster also runs the GM. K denotes the number of clusters in Grid-Level. Grid System Monitor (GS) determines the load index of masters and provides this information to GM.

Client Interface provides a graphical user interface to the user for the submission of jobs. Scheduler is responsible for scheduling of submitted jobs. Dispatcher performs the dispatching of jobs to other masters. Collector is in charge of capturing jobs from other masters. Each neighbour of a cluster is responsible for completing the jobs assigned to them by their master.
A decentralized job scheduling approach is used since the jobs generated by users are directly submitted to master.

Scheduler runs as a sub-component of CM

9. DESIGN OF LOAD BALANCING MODEL

Load balancing should take place when the load situation has changed. There are some particular activities which change the load configuration in Grid environment. The activities can be categorized as following Arrival of any new job and queuing of that job to any particular node.

- Completion of execution of any job.
- Arrival of any new resource
- Withdrawal of any existing resource.

Whenever any of these four activities happens activity is communicated to master node then load information is collected and load balancing condition is checked. If load balancing condition is fulfilled then actual load balancing activity is performed.

SI-LB algorithm gets the load information of cluster and communicating this information to GM via mutual information feedback. Based on the load information SI-LB chooses most suitable node for each job, thereby minimizing job execution time and maximizing system throughput. Load information, generally defined in term of load index is necessary condition of SI-LB algorithm. The load at each computing node contributes to the overall load of the cluster and can be determined as: CPU utilization, CPU speed and queue length. The load index is determined dynamically and the weighted sum of squares method is used to calculate the load at each computing node. Figure represents the logical structure of a processing cluster. Any cluster in the grid can be a processing cluster.
In Cluster-Level load balancing, depending on the current workload of its associated cluster, estimated from its own friends in NLIST, each Cluster-Level manager (CM) decides whether to start or not a load balancing operation. If it decides to start a load balancing operation, then it tries to load balance the workload among its under-loaded friends in NLIST. If any friend in the cluster at any instant of time is under loaded or over-loaded, it requests or allots jobs to/from other friends in (NLIST) with minimal load using the symmetric initiated approach to load balancing. If the master is unable to load balance the workload among its friends, then the jobs are transferred to the under-loaded masters in (CLIST) with minimum load and minimum communication delay.
In Grid-Level load balancing, the load balancing is performed only if CM fails to load balance their workload among their associated friends. In this case, jobs of overloaded clusters are transferred to under loaded ones in (CLIST) regarding the minimum communication delay and load. The chosen under loaded clusters are those which need minimal communication delay for transferring jobs from overloaded clusters.

10. IMPLEMENTATION DETAILS AND EXPERIMENTAL RESULTS

To implement the proposed Load Balancing Algorithm, an application has been developed, which is executed in simulated grid environment. The application has been developed using J2EE and Alea 3 simulator.

EXPERIMENTAL RESULTS

In this section we show the performance of the Alea 3 simulator through several experiments. All experiments were performed on Intel Core I3 2.27GHz Laptop with 3GB of RAM. Unless otherwise indicated, the JVM (Java Virtual Machine) was limited by 1GB of available RAM.

The experiment involved 103,656 jobs 14 clusters having 806 CPUs.
Through the experiment, we have compared three different scheduling algorithms: FCFS, EASY-BF and Random+CONS scheduling policy. Figure presents graphs depicting the average machine usage per cluster (left) and the number of waiting and running jobs per day (right) as were generated by the Alea 3 during the experiment. These graphs nicely demonstrate major differences among the algorithms. Concerning the machine usage as expected FCFS generates very poor results. FCFS is not able to utilize available resources when the first job in the queue requires some specific and currently unavailable machine(s). At this point, other more flexible" jobs in the queue can be executed increasing the machine utilization. This is the main goal of the EASYBF algorithm. As we can see, EASY-BF is able to increase the machine usage by using the backfilling approach. Still, EASY-BF does not allow delaying the execution start of the first job in the queue, which restricts it from making more aggressive decisions that would increase the utilization even more. Random+CONS algorithm does not apply such restrictions and thanks to the application of a more efficient schedule-based approach it produces the best results.

In case of the second criteria, similar reasons as in the previous example caused that FCFS is not able to schedule jobs fluently, generating huge peak of waiting jobs during the time. For the same reason, the resulting make span is also much higher than in the remaining algorithms. EASY-BF is capable of a higher resource utilization and reduction of the number of waiting jobs through the time. ESG again produces the best results. As can be seen, these and several other graphical outputs such as those presented in Figure help the user to understand and compare the scheduling process of different scheduling algorithms.

11. CONCLUSION AND SCOPE OF FUTURE WORK

Every Load Balancing algorithm implements five policies. The efficient implementation of these policies decides overall performance of Load Balancing algorithm. In this work we analyzed existing Load Balancing algorithm and proposed an enhanced algorithm which more efficiently implements three out of five policies implemented in existing Load Balancing algorithm. These three policies are: Information Policy, Triggering Policy and Selection Policy. Proposed algorithm is executed in simulated Grid environment.

12. FUTURE DIRECTIONS

- More complex models such as nesting of clusters need to be investigated.
- Additional factors like network bandwidth that may affect the performance of the algorithm need to be studied.
- Experiments could be tried in a real environment.

REFERENCE

[1] Krishnaram Kenthapadi, Stanford University , kngk@cs.stanford.edu and
Gurmeet Singh Mankuy , Google Inc., manku@google.com,Decentralized Algorithms using both Local and Random Probes for P2P Load Balancing.

[2] B. Yagoubi , Department of Computer Science, Faculty of Sciences, University of Oran and Y. Slimani , Department of Computer Science, Faculty of Sciences of Tunis, Task Load Balancing Strategy for Grid Computing .

[3] Hans-Ulrich Heiss and Michael Schmitz, Decentralized Dynamic Load Balancing: The Particles Approach.

[4]Junwei Cao1, Daniel P. Spooner, Stephen A. Jarvis, and Graham R. Nudd, Grid Load Balancing Using Intelligent Agents.

[5]Ian Foster, Argonne National Laboratory & University of Chicago, What is the Grid? A Three Point Checklist.

[6]Jennifer M. Schopf, Mathematics and Computer Science Division, Argonne National Lab, Department of Computer Science, Northwestern University, Grids: The Top Ten Questions.

[7]Karl Czajkowski, Ian Foster and Carl Kesselman, Resource Co-Allocation in Computational Grids.

[8]Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury and Steven Tuecke, The Data Grid:Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets

[9]Foster, I., C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations". International Journal of Supercomputer Applications, 2001.

[10]Jean-Christophe Durand, "Grid Computing a Conceptual and Practical Study", November 8, 2004

[11]Clovis Chapman1, Paul Wilson2, "Condor services for the Global Grid: Interoperability between Condor and OGSA", Proceedings of the 2004 UK e-Science All Hands Meeting, ISBN 1-904425-21-6, pages 870-877, Nottingham, UK, August 2004 http://www.cs.wisc.edu/condor/doc/condor-ogsa-2004.pdf

[12]Javier Bustos Jimenez, Robin Hood: An Active Objects Load Balancing Mechanism for Intranet.

[13]Shahzad Malik, Dynamic Load Balancing in a Network of Workstations, 95.515F Research Report, November 29, 2000.

[14]Menno Dobber, Ger Koole, and Rob van der Mei, Dynamic Load Balancing for a Grid Application, http://www.cs.vu.nl/~amdobber

[15]Guy Bernard, A Decentralized and Efficient Algorithm for Load Sharing in Networks of Workstations. [51] D. Klus_a_cek, L. Matyska, and H. Rudov_a. Alea { Grid scheduling simulation environment. In 7th International Conference on Parallel Processing and Applied Mathematics (PPAM 2007), volume 4967 of LNCS, pages 1029{1038. Springer, 2008}}.

[16] Francois Grey, Matti Heikkurinen, Rosy Mondardini, Robindra Prabhu, "Brief History of Grid", http://Gridcafe.web.cern.ch/Gridcafe/Gridhistory/history.html.

[17] Gregor von laszewaski, Ian Foster, Argonne National Laboratory, Designing Grid Based Problem solving Environments www-unix.mcs.anl.gov/~laszewsk/papers/cogpse- final.pdf.

Mukul Pathak

The author is pursuing Post Graduation in engineering from Galgotias College of Engineering & Technology, Greater Noida (U.P.). He had completed engineering from College of Engineering & Technology; Moradabad (U. P.) affiliated to Gautam Buddh Technical University in 2010.

Ajeet Kumar Bhartee

The author currently working in Galgotias College of Engineering & Technology, Technology, Greater Noida (U.P.), and have 9+ year experience. He had completed engineering from Madan Mohan Malvia Engineering College; Gorakhpur (U. P.) affiliated to Uttar Pradesh Technical University in 2001 and completed his Masters from CDAC Noida (U.P) in 2007.

Vnay Tandon

The author currently working in Aligarh college of engineering and technology, Aligarh (U.P.), and have 13 year experience. He had completed Master in Computer Application; and Pursuing his M-Tech.

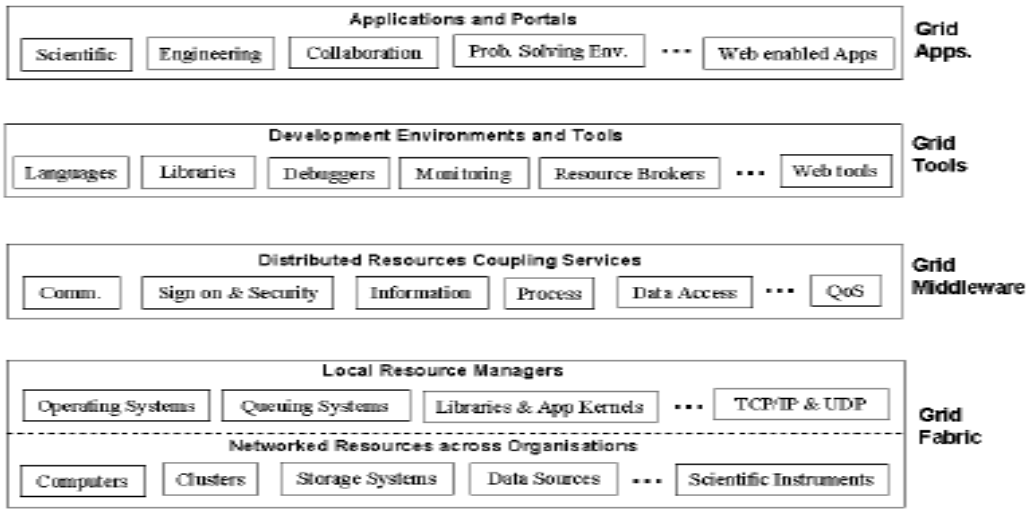

Figure-1*: Grid Architecture*

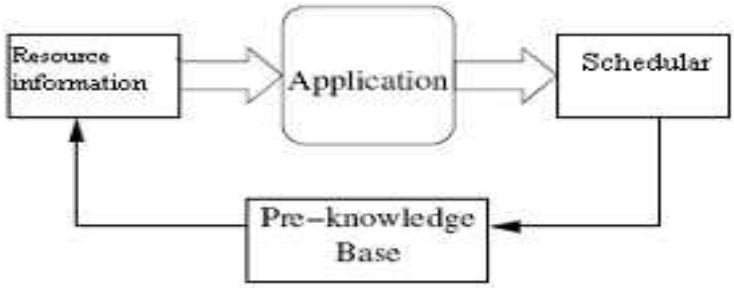

Figure-2*: Static Load Balancing [12]*

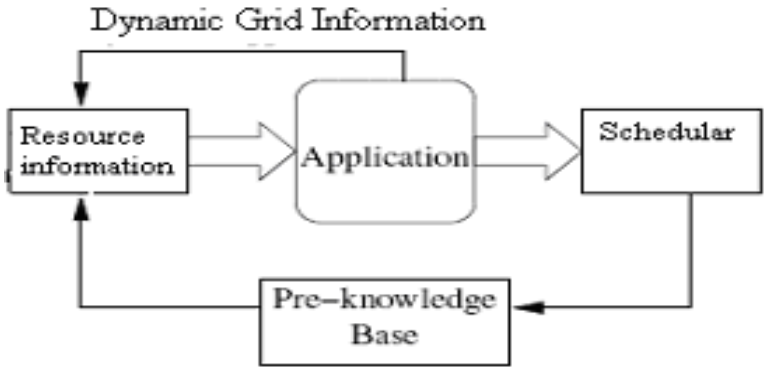

Figure 3: Dynamic Load Balancing [12]

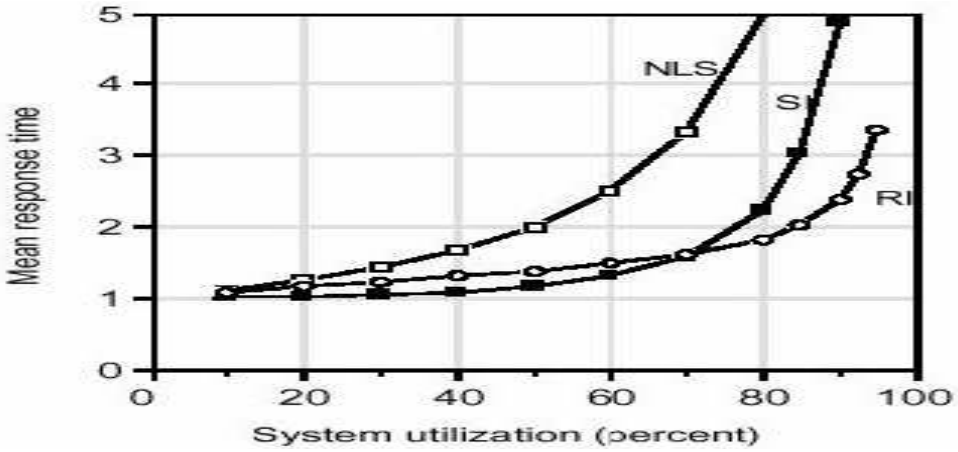

Figure 4: System Utilization of Sender initiated, receiver initiated and No load

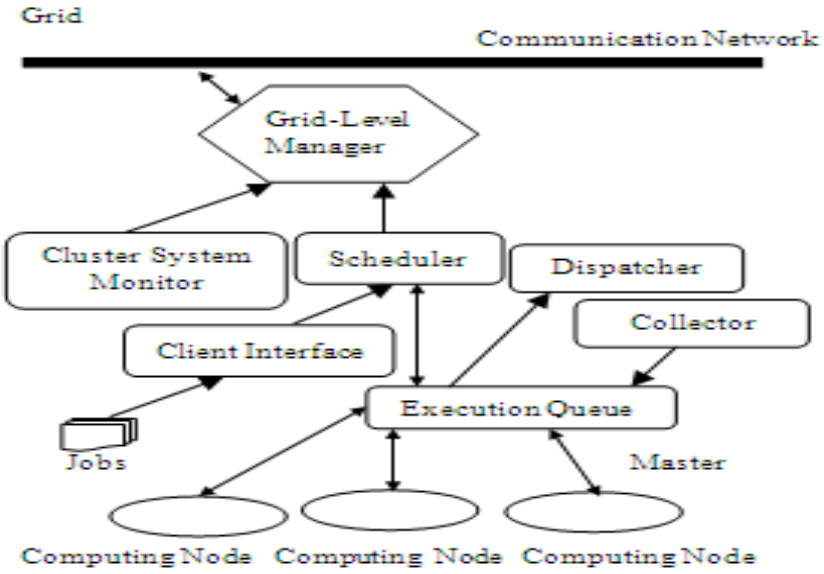

Figure 5: Design of Load Balancing Model

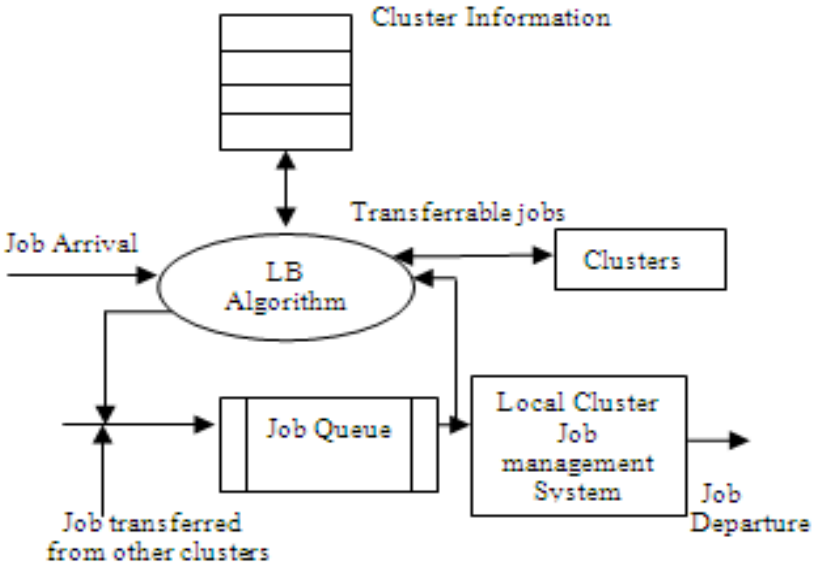

Figure 6: Logical Structure of Processing Cluster

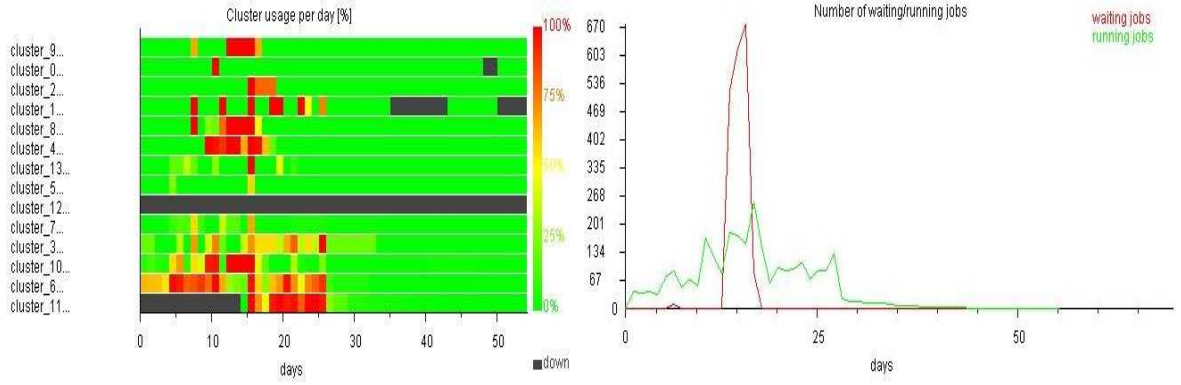

Figure 7: FCFS Scheduling Result

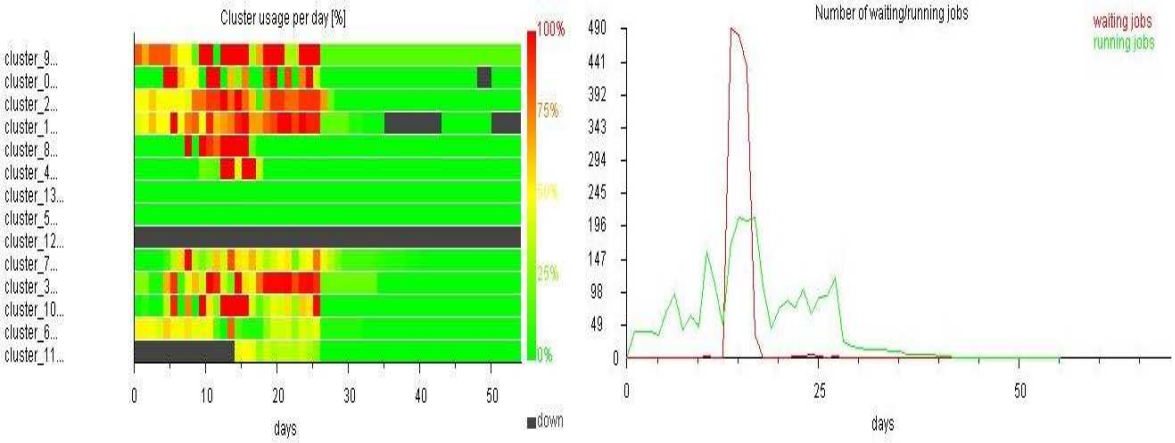

Figure 8: EASY-BF Scheduling Result

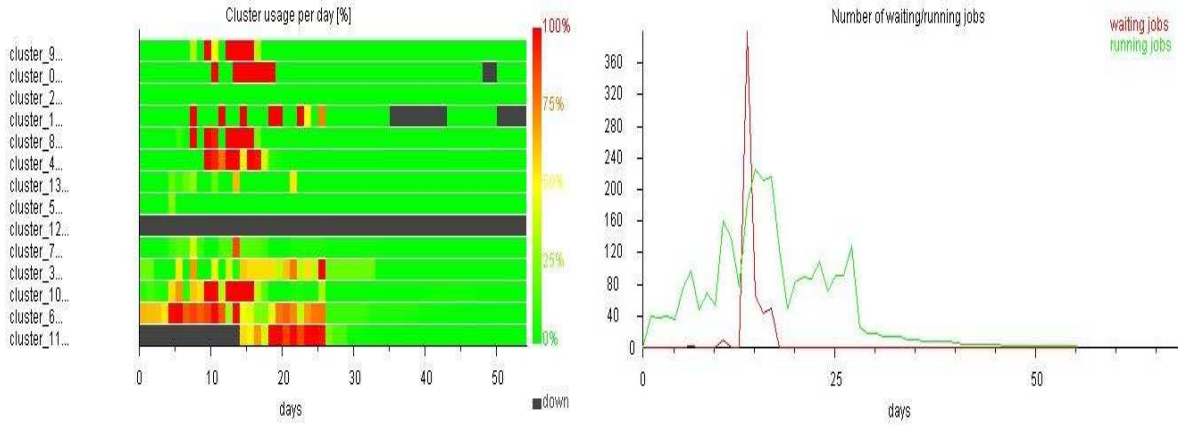

Figure 9: CONS+RANDOM Scheduling Result

This academic article was published by The International Institute for Science, Technology and Education (IISTE). The IISTE is a pioneer in the Open Access Publishing service based in the U.S. and Europe. The aim of the institute is Accelerating Global Knowledge Sharing.

More information about the publisher can be found in the IISTE's homepage:
http://www.iiste.org

The IISTE is currently hosting more than 30 peer-reviewed academic journals and collaborating with academic institutions around the world. **Prospective authors of IISTE journals can find the submission instruction on the following page:** http://www.iiste.org/Journals/

The IISTE editorial team promises to the review and publish all the qualified submissions in a fast manner. All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Printed version of the journals is also available upon request of readers and authors.

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar