Article can be accessed online at http://www.publishingindia.com

Resource Cost Optimization for Dynamic Load Balancing on Web Server System

Harikesh Singh*, Shishir Kumar**

Abstract

The growth of technology increases user expectations at a fast pace in terms of performance and efficiency of web servers. Such user's expectations are required the number of efficient and dynamic computational resources and mechanism. Several scheduling policies have been already working on multiple web servers, but still web servers get overloaded while the access of resources has been increased. Now-a-days, to increase the number of resources gets costly for the organizations so there is a need for an efficient scheduling policy which can optimize the cost of the resources as per the user's expectation. In this paper, an efficient dynamic load balancing policy based on the activity of the processes has been proposed for reducing the cost of Grid resources and simulated on the Grid-Sim simulator to compare with existing scheduling policies. Therefore, costs of resources have been optimized through the proposed dynamic load balancing policies on web servers.

Keywords: Dynamic Load Balancing, Web Server, Grid, Grid-Sim, Scheduling Policy, Resource Costs, Condor Scheduler

1. INTRODUCTION

In computer networking, load balancing is a technique to distribute workload evenly across two or more servers, network links, CPUs, hard drives, or other resources in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overloading (Kamarunisha, 2011). To improve the performance of web servers, resources can be replicated on several servers and requests can be distributed to a suitable replica, normally the server with the lightest load. The rapid development in computing resources has enhanced the performance of web servers and reduced their costs (Kenthapadi & Manku, 2005).

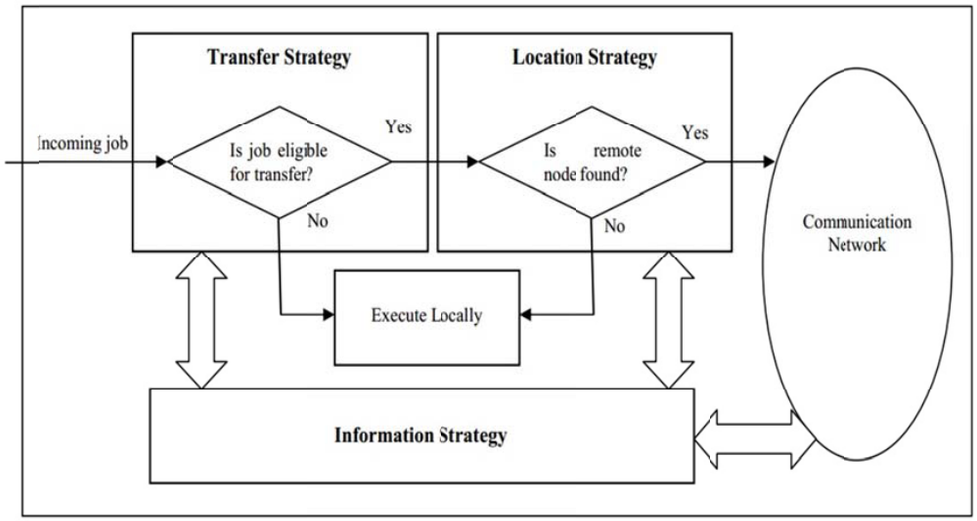
A dynamic load balancing strategy must provide a mechanism through which a destination node can determine the transferred job. There are three main components used in the dynamic load balancing approach as: information, transfer, and location strategies. As shown in figure-1 (Foster, Kesselman, & Tuecke, 2001), incoming jobs can be intercepted by the transfer strategy through which it can be decided that jobs should be transferred or not over a remote machine for load balancing. When the transfer strategy decides about the transfer of a job, the location strategy is triggered to find the remote node for that job. All the essential information which required in taking decisions by both transfer and location strategies are performed by Information strategy.

In general all load balancing algorithms perform on homogeneous and dedicated resources, however, it may not perform efficiently on Grid architectures (Yagoubi & Slimani, 2007). Grids have several explicit characteristics, like heterogeneity, autonomy, scalability, adaptability and resource computational data separation, which generates several complexity results in the load balancing. Load balancing (Buyya & Murshed, 2002) scheduler is responsible for resource discovery, resource trading, resource selection, and job assignment and the dispatcher trigger the appropriate actuators deploy agents

* Assistant Professor, Department of Computer Science and Engineering, Jaypee University of Engineering and Technology, Guna, Madhya Pradesh, India. E-mail: harikeshsingh@yahoo.co.in

** Professor Department of Computer Science and Engineering, Jaypee University of Engineering and Technology, Guna, Madhya Pradesh, India. E-mail: dr.shishir@yahoo.com

Figure 1. Dynamic Load Balancing Strategies for a Job



on Grid resources and schedule one of the resources for the execution of jobs. The schedule advisor maintains a schedule for the complete processing of jobs based on the user requirements, the dispatcher deploys jobs on the resources periodically based upon the resource load and available number of CPUs.

An application level resource broker in the web server Grid has integrated three adaptive approaches for resource controlled scheduling:

- Cost optimization, within time and resource control
- Time optimization, within time and resource control
- Conservative time optimization, within time and resource control

The functions of deadline and resource control in scheduling and objectives of different scheduling approach are given in Table-1. Therefore, cost of execution and execution time should be minimized for the cost optimization of resources.

Table 1: Adaptive Scheduling Algorithms Constraints and Objectives

Adaptive Scheduling Algorithms	*Execution Time (not beyond deadline)*	*Execution Cost (not beyond budget)*
Cost Optimisation	Limited by deadline	Minimize
Time Optimisation	Minimize	Limited by resources
Conservative Time Optimisation	Limited by deadline	Limited by resources

2. RELATED WORK

In web servers Grid, individual users can access several computational resources and data visibly without considering the position, operating system, and other details. Such web servers are highly effective to solve complex scientific problems and in the achievement of expected outcome. Presently, all wide applications are utilizing the web servers Grid with following reasons (Ferreira et al., 2003):

2.1 Exploit Unused Resources

There are several organizations in which most of the computing resources are underutilized. Many desktop machines are busy less than 25% of the time and even the server machines can often be fairly idle. In such cases Grid computing offers a structure for exploiting these underutilized resources and also supports to increase the efficiency of resource usages (Ferreira et al., 2003).

This is one of easiest use of Grid computing would be to run an existing application on various resources. In general, during the execution of any application the resources get busy and execution of the task would be delayed. Grid computing enables the jobs to be run on an idle resource elsewhere on the network.

2.2 Increase Computation

To provide users with more computational power, some crucial areas have to be considered. These areas are: hardware improvement, periodic computational needs, capacity of idle machines, and sharing of computational effects.

2.3 Characteristics of Web Servers Grid

There are three main issues that characterize computational web servers Grids (Nieuwpoort, Kielmann, & Bal, 2001): heterogeneity, scalability, dynamicity or adaptability of parallel CPU execution.

2.4 Grid Architecture

It is protocol architecture with protocols defining the basic mechanisms by which users and resources, negotiate, build, manage and exploit sharing relationships. Grid architecture is also a services standards based open architecture that facilitates extensibility, interoperability, portability, and code sharing (Genaud, Giersch, & Vivien, 2003).

The components that are necessary to form a Grid as: Grid Fabric, Grid Middleware, Grid Development Environments and Tools, Grid Applications and Portals as shown in figure-2 (Genaud, Giersch, & Vivien, 2003).

2.5 Web Servers Grid Challenges

Resource management becomes quite difficult because of the dynamic quality of the web servers. Several publications have concentrated upon the dynamic quality of the web servers Grid as: discovery of resources, selection and position, active monitoring, relocation during execution time, the resources of the networks (Yagoubi & Slimani, 2006).

2.6 Load Balancing Model

(Yagoubi & Slimani, 2006) model is based on an incremental tree. In this model, each machine generates a two-level sub-tree and the sub-tree leaves communicated to the machine about its computing elements. The root

Figure 2. General Grid Architecture

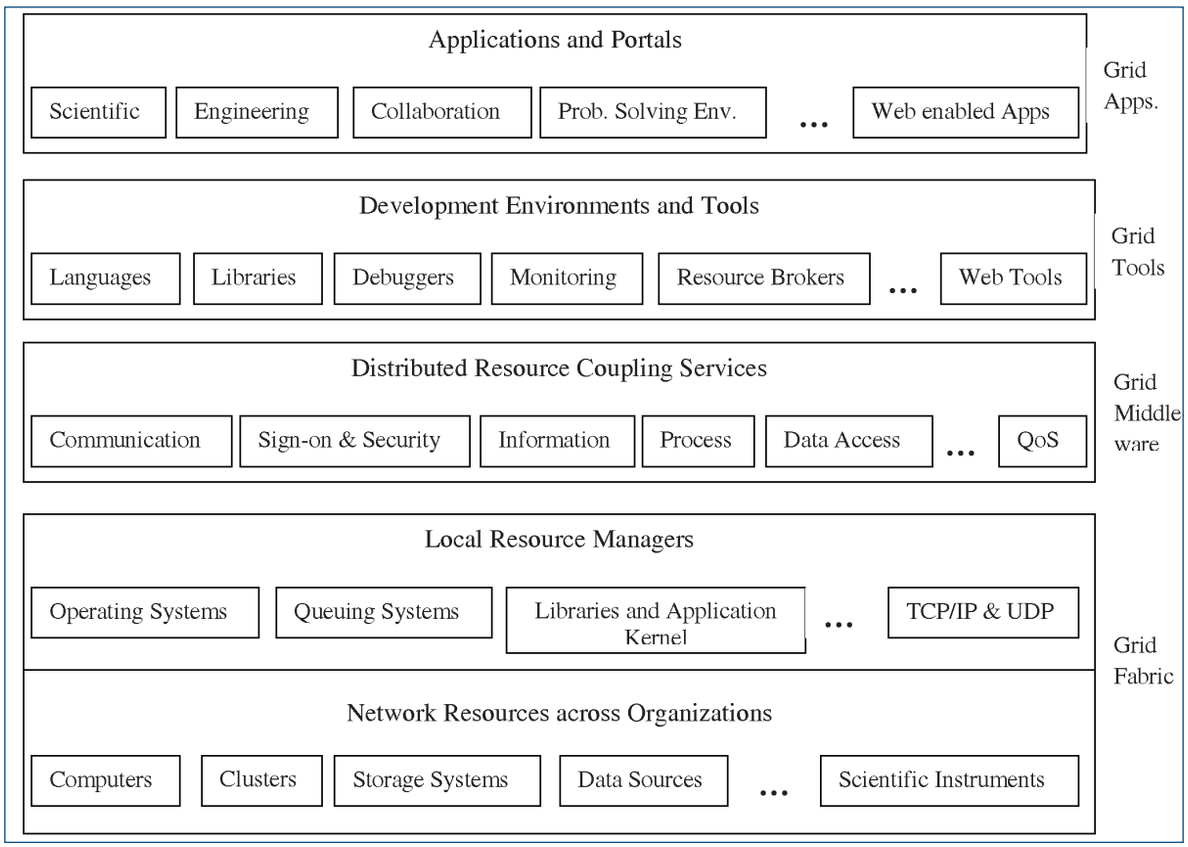
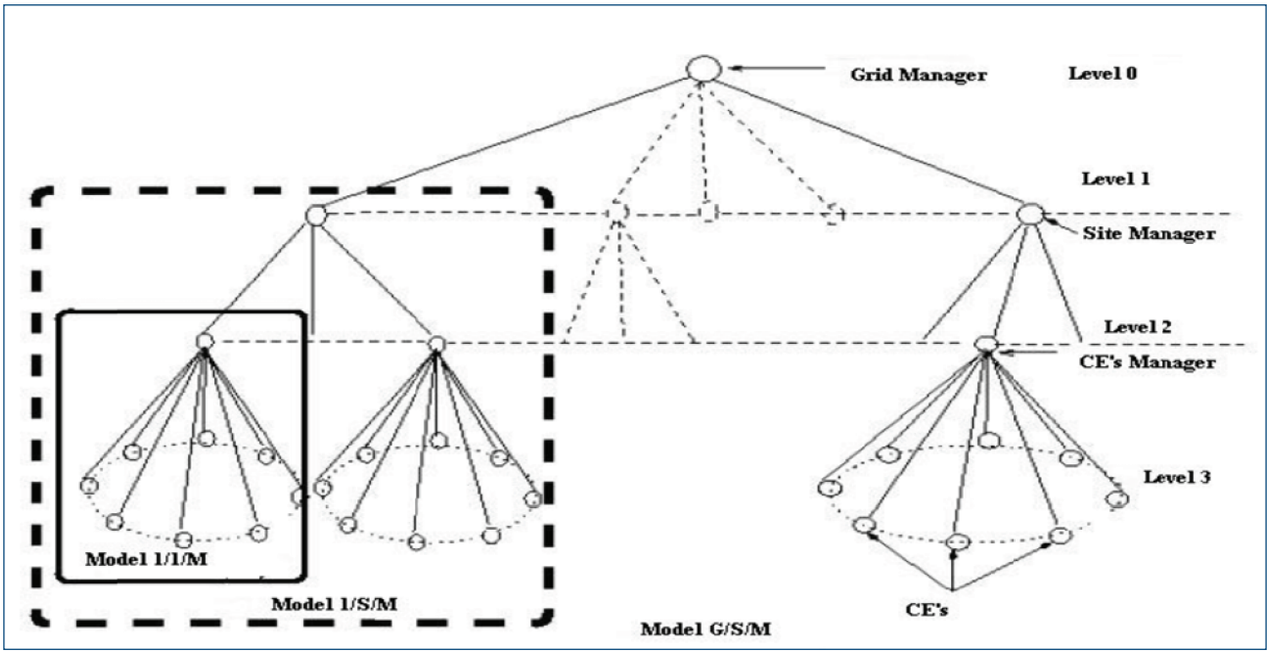


Figure 3. Generic Model for Load Balancing



of the tree as a virtual node associated with the machine (Yagoubi & Slimani, 2006).

The machines of a resources have been communicated by these sub-trees and then combined to develop a three-level sub-tree. Eventually, these sub-trees are linked together to generate a four-level tree called load balancing generic model as shown in figure-3 (Yagoubi & Slimani, 2006).

2.6.1 Level-0 (Grid Manager/Broker (GB))

Grid Manager/Broker in the first level of the tree act as a virtual node which corresponds to the root of the tree. It performs the following roles in the web server Grid (Yagoubi & Slimani, 2006):

(i) To collect the information regarding different resource managers in its geographical areas.

(ii) To manage the workload information of the Grid.

(iii) To take decisions based upon receiving tasks from users and these jobs can be assigned based on the demands of the user and the existing load of the Grid.

(iv) To allocate the task and load balancing procedure in the Grid.

(v) To do a global load balancing between the Grid resources.

(vi) To send the load balancing decisions over the resources of level-1 execution.

2.6.2 Level-1 (Site Manager/ Resource Manager (RM))

In this level, "G" virtual nodes called as resource manager or site manager and each one is related to a physical cluster of the Grid which is responsible for organizing the pool of machines. Its role is to collect the information about the various active processing elements in its pool which includes speed of CPU and other hardware specifications. These virtual nodes are responsible for the followings (Yagoubi & Slimani, 2006):

- Preserving the information relating to each machine's load.
- Estimating its associated machine workload and allocating the incoming jobs to any machine in its pool.
- Local load balancing has managed and called as resource-level load balancing.
- The decision has taken to invoke GB-level load balancing algorithm.

2.6.3 Level-2 (CE or Machine Manager (MM))

This level finds S nodes associated with physical sites of all clusters of the Grid and manage the workload of their physical computing elements. Any machines may connect to the Grid system by registering within any RM to offer

its computing resources to be utilized by the Grid users (Yagoubi & Slimani, 2006).

More than one PE's can have the resources and each PE's have some specified CPU speed in terms of MIPS (million instructions per second) rating. PE's are responsible for actual implementation of jobs or Gridlets. When any Gridlet is submitted to the machines, MM will assign this Gridlet to the available PEs. At this level the migration of Gridlet has been done and each MM is responsible for the followings (Yagoubi & Slimani, 2006):

- Maintaining workload information of its associated PEs.
- Estimating workload of all PEs.
- Managing a local load balancing, called as machine-level load balancing.
- Deciding whether to invoke resource-level load balancing algorithm.

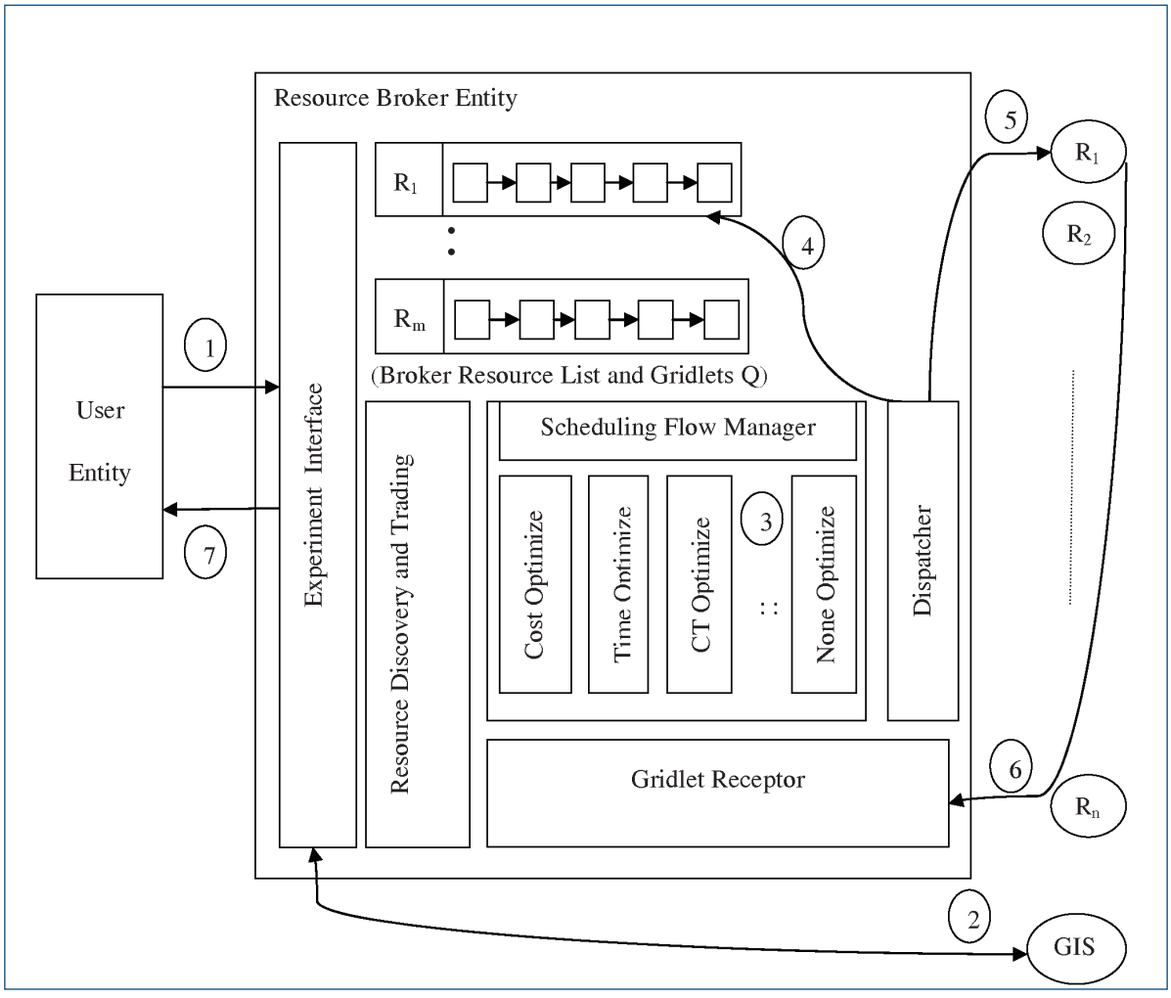
2.6.4 Level-3 (Processing Elements)

"M" computing elements or processing elements of the Grid linked to their respective sites and clusters in the last level of the tree,.

2.7 Economic Grid Resource Broker:

The GridSim toolkit has been used to simulate the Grid environment of web servers having multiple web resources and user entities with different requirements. The resource broker entity architecture along with its interaction flow diagram with other entities has shown

Figure 4. Architecture of Economic Grid Resource Broker



in the figure-4. The functionality of resource broker components and their interaction are given below (Buyya, Murshed, & Abramson, 2001):

(i) The user entity generates an experiment having a list of Gridlets for processing and experimental interface generates user requirements to the broker.

(ii) The resource discovery and trading module interaction with the resource broker of GridSim GIS entity classify the information of resources (R_1, R_2,…., R_m) and then interacts with the resources to establish their configuration and access cost.

(iii) The scheduling flow manager opts a suitable scheduling algorithm based upon the user requirements for mapping Gridlets.

(iv) In every resource, "Q" number of Gridlets selected by the dispatcher and as per the usage policy it can be staged for the execution for avoiding resource's overload with single user jobs.

(v) Gridlets have been submitted to resources using the GridSim's asynchronous service by the dispatcher.

(vi) After completion of Gridlet processing, the resource returns back to the broker's Gridlet receptor module for measures and updates of the runtime parameter of resources.

(vii) From the steps (iii) to (vi), continue until all the Gridlets are processed or the broker exceeds deadline limits of execution.

3. PROPOSED LOAD BALANCING POLICY

All load balancing approaches have followed five policies as:

- **Information Policy:** In this policy, the collected load information has been specified, when it is to be collected and from where.
- **Triggering Policy:** This policy is used for determining the right instant to initiate a load balancing process.
- **Resource Type Policy:** This policy classifies a resource as a server or receiver of tasks according to its availability status and capabilities.
- **Location Policy:** This policy uses the results of the resource type policy to find a suitable partner for a server or receiver.
- **Selection Policy:** In this policy, tasks has been defined that should be migrated from overloaded resources to idlest ones.

The effective implementation of these policies decides the overall execution of load balancing algorithms. The main difference between existing and proposed load balancing approach is in the implementation of three policies: Information policy, Triggering policy and Selection policy. For implementation of Information policy all existing load balancing algorithm uses a periodic approach, which is very time consuming. Therefore, the proposed approach uses an activity based approach for implementing Information policy and Triggering policy uses two parameters which decide the load index. On the basis of load index, load balancer decides to activate the load balancing process. For implementation of Selection policy, the proposed approach uses server queue-length as a parameter, and the task is migrated based on available resources which can be used more reliably to make decisions about selection of job for migration from heavily loaded to lightly loaded nodes as shown in Table-2.

Table 2: Policy Parameters of Dynamic Load Balancing

	Selection Policy	*Information Policy*	*Triggering Policy*
Existing Condor Load Balancing Approach (Kamarunisha, Ranichandra, & Rajagopal, 2011)	Load balancing information is collected using periodic approach	Load Balancer is triggered based on queue-length	The task is selected for migration using the job-length as criteria.
Proposed Approach	Load balancing information is collected using activity based approach	Load Balancer is triggered based on server queue-length	The task is selected for migration based on available resources

The proposed load balancing technique uses an activity based approach in selection policy. It takes into account the current load as well as the incoming load for Information policy and selects the job for migration based upon the incoming load hierarchies. The proposed approach is using the sender initiated strategy in which machines or resources are required to transfer and then Gridlets start searching for the under-loaded processing elements, machines or resources. The proposed approach also works on three levels: broker, resource and machine level (level-0, level-1, and level-2 respectively). When a new Gridlet arrives at the machine, it submits it to a processing element which is lightly loaded and after this activity of Gridlet, it starts checking of load of all the processing elements and then classifies them into overloaded, normal loaded, and under-loaded. Whenever any Gridlet arrives, activity happens and communicated to master node then load information is collected and load balancing condition is checked. If the load balancing condition is fulfilled, then the actual load balancing activity is performed. In accordance with the structure of the proposed model, the load balancing strategy is also hierarchical and differentiate between three load balancing levels:

- **Intra-Site Load Balancing:** On the first level, action of load balancing has started based on the current load of each site. In this case, the site tries in precedence to balance its workload among its computing elements.
- **Intra-Cluster Load Balancing:** At the second level, load balancing has performed in only one cluster among the clusters of a Grid. This kind of load balance is achieved only if some sites fail to load balance its workload among their respective computing elements.
- **Intra-Grid Load Balancing:** If clusters also fail to balance the load, then load balancing has performed at this level.

4. SIMULATION AND EXPERIMENTAL ANALYSIS

To simulate the application scheduling in GridSim environment using the economic Grid resource broker, it requires the modeling and creation of GridSim resources and applications that modeled jobs as Gridlets. In this section, the costs of resources used for simulation have been given in Table-3 as follows:

The cost of the resources has been calculated and compared through proposed load balancing w.r.t existing condor load balancing policy. The total cost of processing over the Grid resources are as follows (Buyya & Murshed, 2002):

Table 3: Web Server Grid Testbed Resources, Simulated using Grid-Sim (Buyya & Murshed, 2002)

Resource Name in Simulation	*Simulated Resource Characteristics (Vendor, Resource Type, Node OS, No of PEs)*	*Resource Manager Type*	*Price (G$/PE time unit)*
R0	Compaq, AlphaServer, CPU, OSF1, 4	Time-shared	8
R1	Sun, Ultra, Solaris, 4	Time-shared	4
R2	Sun, Ultra, Solaris, 4	Time-shared	3
R3	Sun, Ultra, Solaris, 2	Time-shared	3
R4	Intel, Pentium/VC820, Linux, 2	Time-shared	2
R5	SGI, Origin 3200, IRIX, 6	Time-shared	5
R6	SGI, Origin 3200, IRIX, 16	Time-shared	5
R7	SGI, Origin 3200, IRIX, 16	Space-shared	4
R8	Intel, Pentium/VC820, Linux, 2	Time-shared	1
R9	SGI, Origin 3200, IRIX, 4	Time-shared	6
R10	Sun, Ultra, Solaris, 8	Time-shared	3

$$Processing\ cost = (actual\ CPU\ time\ of\ resource) * (resource\ cost\ per\ second) \quad (1)$$

Several processes have waited in response of resources due to waiting in the job queue. This waiting time of the process has also been calculated by the following equation (Cakanyildirim, 2014):

$$Waiting\ time = Activity\ time * \left(\frac{utilization}{1 - utilization}\right) * Variability\ factor \quad (2)$$

Where the activity time depends on the execution of the resources and variability factor can be computed based upon the coefficient of variation. The utilization rate of the resources is the fraction of the time that the resource was being utilized by the activity. If there is more than one resource required for any process and if the resources are identical with respect to processing capability then it is required to add up all of the average utilization rates and divide by the number of resources. In another condition where resources are not similar with respect to processing capability it is required to collect an average utilization rate data for each individual resource (Chung, 2004):

$$\text{Average Resource Utilization} = \frac{\int_0^T B\,dt}{T} \quad (3)$$

Where, B=0 for idle or 1 for busy, dt=length of time that B is observed, T=total length of time for the simulation. In order to implement the proposed load balancing model, the Grid environment has been simulated for web servers Grid using GridSim simulator. The proposed approach has been implemented using Java programming language on the GridSim 5.0 toolkit.

Case-1: Comparative Analysis of Costs with Different Processing Elements in the Resources

In this case, the proposed approach has considered for two different scenarios in which first scenario includes 1 resource with 1 machine and 1 processing element, whereas second scenario contains 1 resource with 1 machine and 3 processing elements. Comparison of costs has been performed using 10 Gridlets of different sizes as in Table-4 and the respective graph has shown in figure-5.

It has been observed from figure-5, the costs of the resources decreases as the number of processing elements increases using the proposed approach.

Table 4: Comparison of Costs with Different Processing Elements (PE)

S. No.	Gridlets Size (lines of code)	Costs with PE=1 (G$/PE time unit)	Costs with PE=3 (G$/PE time unit)
1	110	160.7	25.08
2	150	236.97	26.28
3	175	237.77	27.12
4	200	199.09	27.96
5	240	109.64	28.92
6	245	239.99	29.28
7	270	110.59	30.12
8	300	241.75	31.08
9	310	242.07	31.44
10	350	243.34	39.84

Case-2: Comparative Analysis of Costs when Number of Machines Increases Inside the Resource

In this case, two different scenarios have taken where first scenario includes 1 resource with 1 machine and 1 processing element, whereas second scenario contains 1 resource with 2 machines and 1 processing element in each. Comparison of costs has been performed using 10 Gridlets of different sizes as using proposed approach in Table-5 and figure-6 shows the comparative analysis of costs.

Table 5: Comparison of Costs with Different Machines

S. No.	Gridlets Size (lines of code)	Costs with no. of machines=1 (G$/PE time unit)	Costs with no. of machines=2 (G$/PE time unit)
1	110	160.7	34.58
2	150	236.97	34.05
3	175	237.77	37.01
4	200	199.09	35.65
5	240	109.64	37.76
6	245	239.99	37.08
7	270	110.59	34.87
8	300	241.75	38.83
9	310	242.07	37.83
10	350	243.34	40.3

Figure 5. Comparison of Costs with Different Processing Elements

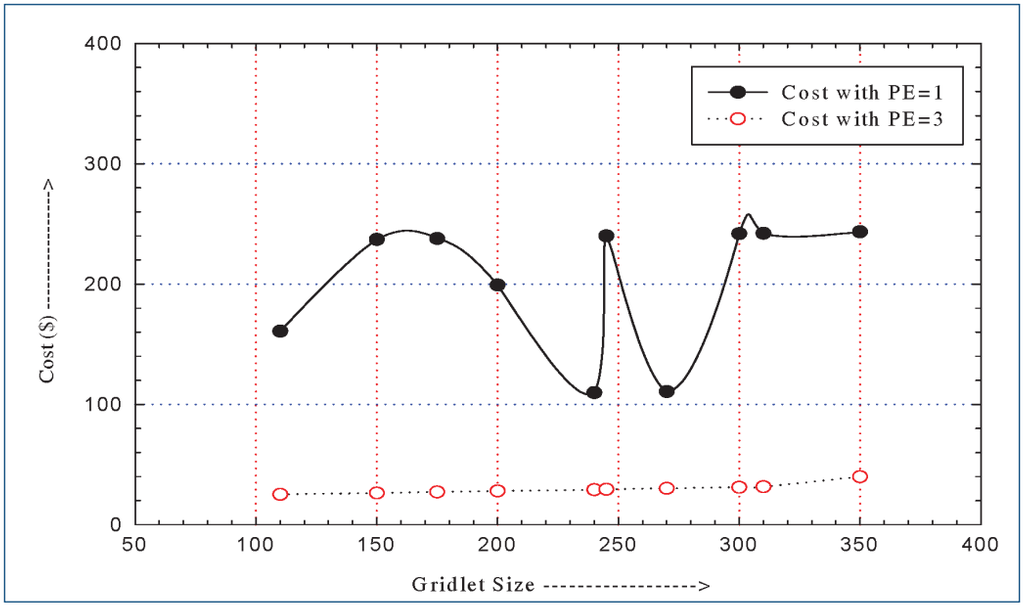
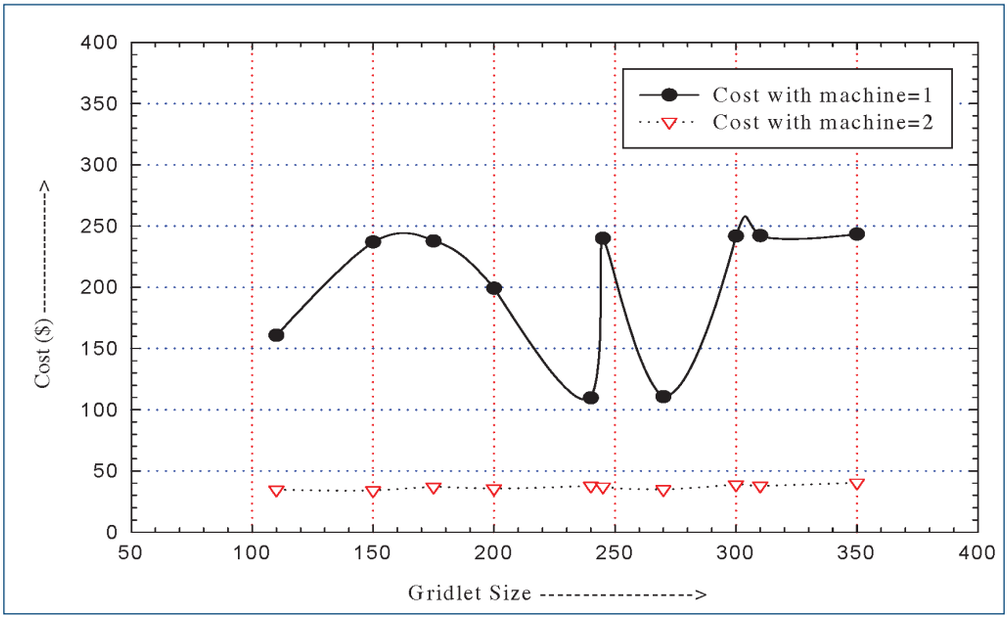


Figure 6. Comparison of Costs with Different Machines



It has been observed from figure-6, the costs of the resources decreases as the number of machines increases using the proposed approach.

Case-3: Comparison of the Resource Costs with the Periodic Approach

In this case, two different scenarios have taken where first scenario includes the resources with computing elements executed using a Periodic approach, whereas second scenario contains resources with processing elements executed using Activity based approach. Comparison of costs has been performed using 10 Gridlets of different sizes as in Table-6 and figure-7 shows the comparison of costs.

It has been observed from figure-7, the costs of the resources decreases using the proposed approach in comparion of existing periodic approach (kamarunisha et al., 2011).

Table 6: Comparison of Costs with Periodic Approach

S. No.	Gridlets Size (lines of code)	Costs using a Periodic approach (G$/PE time unit)	Costs using Proposed Approach (G$/PE time unit)
1	110	73.33	25.08
2	150	38.9745	26.28
3	175	20.0458	27.12
4	200	41.75	27.96
5	240	55.63	28.9198
6	245	54.768	29.28
7	270	53.98	30.1196
8	300	51.581	31.08
9	310	14	31.44
10	350	47.2601	39.8396

Case-4: Comparison of the Waiting Time with the Periodic Approach

In this case, two different scenarios have taken where first scenario includes resources with computing elements executed using a Periodic approach, whereas second scenario contains resources with processing elements executed using the proposed approach. Comparison of costs has been performed using 10 Gridlets of different sizes as in Table-7 and the figure-8 has shown the comparison of waiting Time.

Table 7: Comparison of Waiting Time with Periodic Approach

S. No.	Gridlets Size (lines of code)	Waiting Time by Periodic Approach (seconds)	Waiting Time by Proposed Approach (seconds)
1	110	11.6542	13.39777
2	150	15.76008	19.747877
3	175	17.2989	19.81419
4	200	20.2985	16.5905
5	240	24.0347	9.13664
6	245	25.10893	19.999867
7	270	27.1289	9.21618
8	300	30.02637	20.145755
9	310	31.23677	20.1722811
10	350	35.02176	20.2783819

As shown in figure-8, the waiting time of processes have been minimized using the proposed approach in comparison of existing periodic approaches (Kamarunisha et al., 2011).

5. CONCLUSIONS AND FUTURE WORK

The proposed approach has focused on the problem of load balancing in the web servers Grid. Every load balancing approach in the Grid environment has implemented five policies. The efficient implementation of these policies decides the overall performance of the load balancing

Figure 7. Comparison of Resource Costs with Periodic Approach

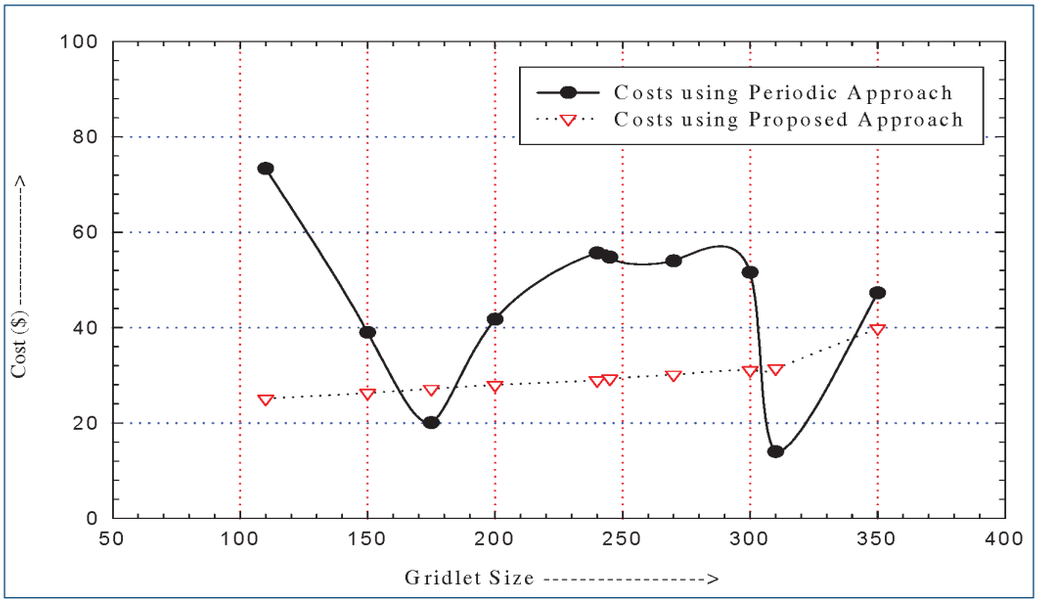
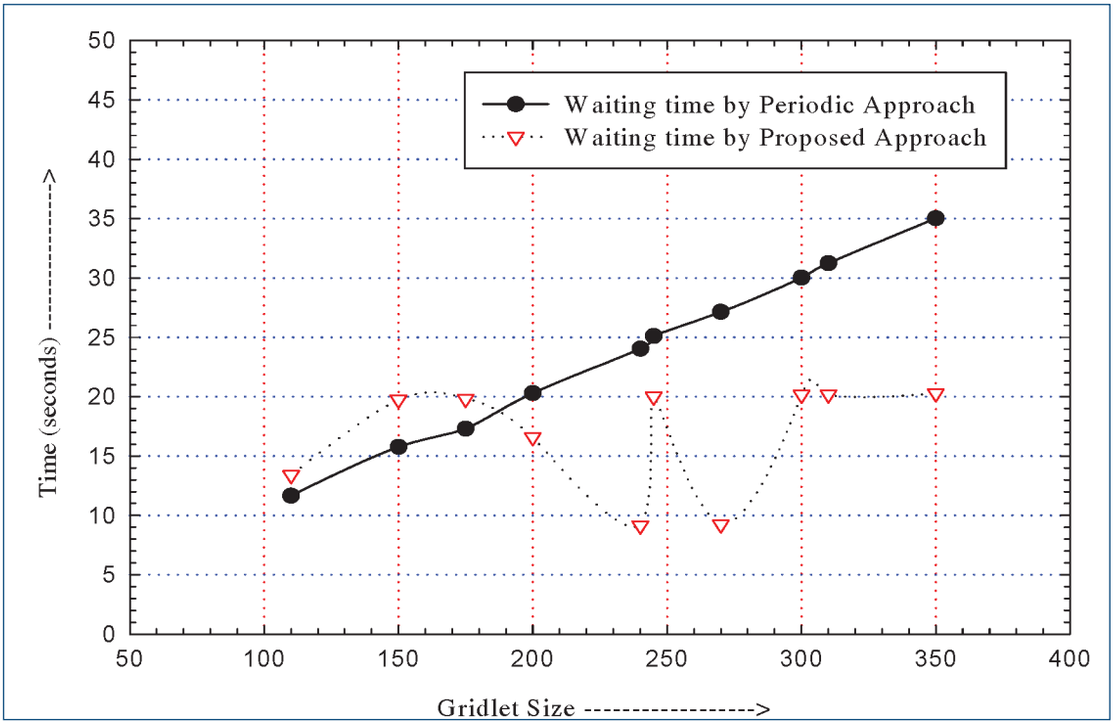


Figure 8. Comparison of Waiting Time with Periodic Approach



strategy. We have analyzed the existing load balancing approaches and proposed an approach which works more efficiently. In the proposed approach, three out of five policies have been implemented in comparison to the existing load balancing approach. These three policies are: Information policy, Triggering policy and Selection policy. The proposed approach has been simulated in Grid environment of web servers and the results of simulation have been compared with existing approaches. As a result, the reduction in costs of the resources as well as in waiting time has been observed with various scenarios.

Therefore, we can conclude that the proposed policies are better than the previous ones, but the performance of Grid applications remains a challenge in a dynamic Grid environment. Resources can be added to Grid and can be extracted from Grid at any moment. This characteristic of Grid makes load balancing one of the critical features of Grid infrastructure. This activity based proposed approach can be used as the basis for an improved load balancing module in the Condor scheduler. This approach can not only improve the performance of Grid application, but also makes it more powerful, reliable and capable of handling more complex and large problems in Grid environments. It can also be extended in making the load balancing module as a middleware independent module.

REFERENCES

[1] Buyya, R. & Venugopal, S. (2005). A gentle introduction to grid computing and technologies. *CSI Communications,* 29(1), 1-19.

[2] Buyya, R. & Murshed, M. (2002). GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. *Concurrency and Computation: Practice and Experience,* 14(3-15), 1175-1220.

[3] Buyya, R., Murshed, M. & Abramson, D. (2001). *A Deadline and Budget Constrained Cost-Time Optimization Algorithm for Scheduling Task Farming Applications on Global Grids*. Proceeding of International Conference on Parallel and Distributed Processing Techniques and Applications (pp. 1-12).

[4] Cakanyildirim, M. (2014). *Waiting Times*. Retrieved from the University of Dallas. Retrieved from http://www.utdallas.edu/ ~metin/Or6302/Folios/omqueue.pdf

[5] Caol, J., Spooner, D. P., Jarvis, S. A. & Nudd, G. R. (2005). Grid load balancing using intelligent agents. *Future Generation Computer Systems,* 21(1), 135-149.

[6] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. & Tuecke, S. (2000). The data grid: Towards an architecture for the distributed management

and analysis of large scientific datasets. *Journal of Network and Computer Applications: Special Issue on Network-Based Storage Services,* 23(3), 187-200.

[7] Chung, C. A. (2004). *Simulation Modeling Handbook: A Practical Approach*. CRC Press LLC, University of Houston: Washington, D.C.

[8] Czajkowski, K., Foster, I. & Kesselman, C. (1999). *Resource Co-Allocation in Computational Grids*. Proceeding of 8th IEEE International Symposium on High Performance Distributed Computing (pp. 219-228).

[9] Ferreira, L., Berstis, V., Armstrong, J., Kendzierski, M., Neukoetter, A., Takagi, M., Bing-Wo, R., Amir, A., Murakawa, R., Hernandez, O., Magowan, J. & Bieberstein, N. (2003). *Introduction to Grid Computing with Globus*. Redbook, IBM Corporation.

[10] Foster, I., Kesselman, C. & Tuecke, S. (2001). The anatomy of the grid enabling scalable virtual organizations. *International Journal of High Performance Computing Applications,* 15(3), 200-222.

[11] Foster, I. (2002). What is the Grid? A Three Point Checklist. Argonne National Laboratory & University of Chicago. Retrieved from http://dlib.cs.odu.edu/WhatIsTheGrid.pdf.

[12] Genaud, S., Giersch, A. & Vivien, F. (2003). Load-balancing scatter operations for grid computing. *Parallel Computing,* 30(8), 179-186.

[13] Gu, D., Yang, L. & Welch, L. R. (2005). *A Predictive, Decentralized Load Balancing Approach*. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium.

[14] Heiss, H. U. & Schmitz, M. (1995). Decentralized dynamic load balancing: The particles approach. *Journal of Information Sciences-Informatics and Computer Science,* 84(1-2), 115-128.

[15] Kamarunisha, M., Ranichandra, S. & Rajagopal, T. K. P. (2011). Recitation of load balancing algorithms in grid computing environment using policies and strategies-An Approach. *International Journal of Scientific & Engineering Research,* 2(3), 1-7.

[16] Kenthapadi, K. & Manku, G. S. (2005). *Decentralized Algorithms using both Local and Random Probes for P2P Load Balancing*. Proceeding of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures, Las Vegas, Nevada, USA, (pp. 135-144)

[17] Krauter, K., Buyya, R. & Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience,* 32(2), 135-164.

[18] Laszewaski, G. V., Foster, I., Gwaor, J., Lane, P., Rehn, N. & Russell, M. (2001). *Designing Grid Based Problem Solving Environments and Portals*. Proceeding in 34th Annual Hawaiian International Conference on System Science (pp. 3-6)

[19] Luther, A., Buyya, R., Ranjan, R. & Venugopal, S. (2006). *Peer-to-Peer Grid Computing and A .NET-Based Alchemi Framework*. Proceeding in High-Performance Computing: Paradigm and Infrastructure (eds. L. T. Yang & M. Guo), John Wiley & Sons, Inc., Hoboken, NJ, USA. (pp. 1-21).

[20] Nieuwpoort, R. V. van, Kielmann, T., & Bal, H., E. (2001). *Efficient Load Balancing for Wide Area Divide and Conquer Applications*. Proceeding of the 8th ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming, *36*(7), 34-43, 18th -20th June, Snowbird, Utah, USA.

[21] Schopf, J. M. & Nitzberg, B. (2002). Grids: The top ten questions. *Journal of Scientific Programming,* 10(2), 103-111.

[22] Shoshani, A., Sim, A. & Gu, J. (2002). *Storage Resource Managers: Middleware Components for Grid Storage*. Proceeding in the 19th IEEE Symposium on Mass Storage Systems, Maryland. (pp. 209-224). Retrieved from http://storageconference.org/2002/presentations/d02-arie.pdf.

[23] Yagoubi, B. & Slimani, Y. (2006). *Dynamic Load Balancing Strategy for Grid Computing*. Proceeding of World Academy of Science, Engineering & Technology, (13, pp. 90-95).

[24] Yagoubi, B. & Slimani, Y. (2007). Task load balancing strategy for grid computing. *Journal of Computer Science,* 3(3), 186-194.