Proposal of Dynamic Load Balancing Algorithm in Grid System

Sherihan Abu Elenin
Faculty of Computers and Information
Mansoura University, Egypt

Abstract
This paper proposed dynamic load balancing for Grid systems in order to quickly render information to consumer requests. The proposed algorithm is based on migration the requests from Registry to failover registry if there are overloaded at Registry or Registry is failed. The new point in this algorithm is that the tasks will be ordered in ready queue of Registry or failover registry depending on the priority values (trust values of consumers). The proposed dynamic load balancing algorithm is evaluated by measuring response time, system utilization, and throughput. Finally, we compare all the types of LB algorithms with the proposed dynamic load balancing algorithm. It successes in reducing response time and increasing throughput.

Keywords: *Grid computing, trust management, monitoring system, dynamic load balancing.*

1. Introduction

Load Balancing (LB) is not a new concept in the server or network space. Several products perform different types of load balancing [1]. However, load balancers have emerged as a powerful solution for mainstream applications to address several areas, including server farm scalability, availability, security, and manageability [1].
The main goal of load balancing algorithm 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 [3]. Grids have a lot of specific characteristics, like heterogeneity, autonomy, scalability, adaptability and resources computation-data separation, which make the load balancing problem more difficult [4].

2. Related Work in Load Balancing

Load balancing algorithms can be classified into two categories: static or dynamic.

2.1 Static Load Balancing Algorithms

In these algorithms, the performance of the processors is determined at the beginning of execution [7]. The goal of static load balancing method is to reduce the overall execution time of a concurrent program while minimizing the communication delays. A general disadvantage

of all static schemes is that the final selection of a host for process allocation is made when the process is created and cannot be changed during process execution to make changes in the system load.

Static load balancing algorithms [8] are Round Robin algorithm, Randomized algorithm, Central Manager algorithm, and Threshold algorithm.

Robin algorithm [8] distributes jobs evenly to all slave processors. All jobs are assigned to slave processors based on Round Robin order, meaning that processor choosing is performed in series and will be back to the first processor if the last processor has been reached. Randomized algorithm [8] uses random numbers to choose slave processors. The slave processors are chosen randomly following random numbers generated based on a statistic distribution. Central Manager algorithm [8], in each step, central processor will choose a slave processor to be assigned a job. The chosen slave processor is the processor having the least load. In Threshold algorithm [8], the processes are assigned immediately upon creation to hosts. Hosts for new processes are selected locally without sending remote messages. Each processor keeps a private copy of the system's load. The load of a processor can characterize by one of the three levels: underloaded, medium and overloaded.

2.2 Dynamic Load Balancing

It differs from static algorithms in that the work load is distributed among the processors at runtime. The master assigns new processes to the slaves based on the new information collected [9]. Unlike static algorithms, dynamic algorithms allocate processes dynamically when one of the processors becomes under loaded. Instead, they are buffered in the queue on the main host and allocated dynamically upon requests from remote hosts.

Dynamic load balancing algorithms [10] are Central Queue algorithm and Local Queue algorithm. Central Queue Algorithm works on the principle of dynamic distribution. It stores new activities and unfulfilled requests as a cyclic FIFO queue on the main host. Each new activity arriving at the queue manager is inserted into the queue [16]. Then, whenever a request for an activity is received by the queue manager, it removes the first activity from the queue and sends it to the requester. The basic idea of the local queue algorithm is static allocation of all new processes with process migration initiated by a host when its load falls under threshold limit, is a user-defined parameter of the algorithm. The parameter defines the minimal number of ready processes the load manager attempts to provide on each processor [16].

3. Proposed Grid Monitoring System

3.1 Overview

The proposed Grid Monitoring System is based on the Grid Monitoring Architecture (GMA) [12]. The GMA specification sets out the requirements and constraints of any implementation. It is based on simple Consumer/ Producer architecture with an integrated system registry and distinguishes transmission of monitoring data and data discovery logically.

The architecture of proposed Grid monitoring system is shown in Figure 1 [5], [6]. The proposed Grid monitoring system consists of producers (P), registry, consumers (C), and failover registry. The main aim of proposed system is to provide a way for consumers to obtain information about Grid resources as quickly as possible. It also provides fault tolerance system supported by failover registry. The solid line is the normal communication between consumer and registry. The dotted line

is the replacement communication in case of registry failure.

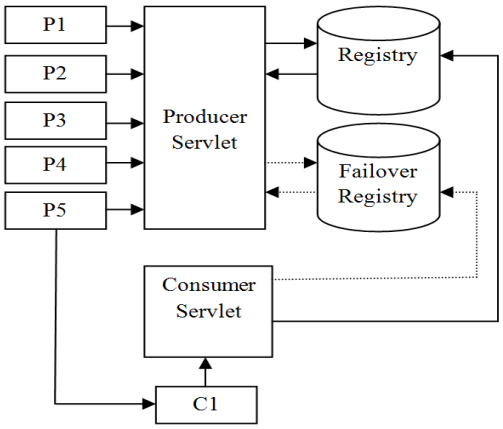

Figure 1. Proposed Grid Monitoring System

3.2 Components of Proposed Grid Monitoring System

Producers are Grid services which register themselves in registry, describe the type and structure of information by SQL CREATE TABLE and SQL INSERT TABLE, and reply to the query of consumer.

Registry acts as a discovery Grid service to find relevant producers matching the query of a consumer. The overall purpose of the registry is to match the Consumer with one or more Producers. This is achieved by that Producers publish information about themselves and then Consumers search through the registry until they find the relevant match and then the two communicate directly with each other. The registry is not responsible for the storage of database, but only the index of it.

Failover registry is a backup version of all layers in registry. It acts like registry in the situation of failure of registry. It also has all the functions of registry.

Consumers can be software agents or users that query the Registry to find out what type of information is available and locate Producers that provide such information.

3.3 Trust and Monitoring System

Our Grid system is divided into Grid domains (GDs). GD consists of application domain (AD), resource domain (RD), client domain (CD), and Trust Manager (TM). TM's operations consist of Trust Locating, Trust Computing, and Trust Updating. This system was proposed and tested in [13]. Every client has a trust level value. This value is one point real value from 0 to 1 to measure the trust value for every client. We add another operation to TM. This operation is Registry to manage the relationship between producers and consumers.

Every domain can have any number of producers and consumers. But it has one TM with Registry; this makes management, and one failover registry node; this makes failure recovery. The domain can have any number of nodes that is intersection with other domains or not.

After analyzing the architecture of the proposed trust model and Grid monitoring system, we observe that there may be overloaded in Registry if the number of requests is large. So Load Balancing (LB) should be added to the proposed Grid monitoring system to get better performance. It is important in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload.

We analyzed and evaluated the four types of static load balancing algorithms in [6]. In this paper, we propose a dynamic load balancing in order to know which will get better performance result.

4. Proposed Dynamic Load Balancing

The most proposed load balancing algorithms were developed in mind, assuming

homogeneous set of sites linked with homogeneous and fast networks [14]. If this assumption is true in traditional distributed systems, it is not realistic in grid architectures because following properties that characterize them [15]:

- Heterogeneity: A Grid involves multiple resources that are heterogeneous in nature and might span numerous administrative domains across a potentially global expanse.
- Scalability: A Grid might grow from few resources to millions. This raises the problem of potential performance degradation as the size of a Grid increases.
- Adaptability: In a Grid, a resource failure is the rule, not the exception. That means that the probability of some resources fail is naturally high.

These properties make the load balancing problem more complex than in traditional parallel and distributed systems, which offer homogeneity and stability of their resources [11]. Also interconnected networks on grids have very disparate performances and tasks submitted to the system can be very diversified and irregular. These various observations show that it is very difficult to define a load balancing system which can integrate all these factors.

Already we have Grid system based on trust and monitoring management. Therefore, we take into consideration when we propose the dynamic load balancing algorithm the properties of proposed trust and monitoring systems.

The steps of proposed dynamic load balancing are as shown in Figure 2:

Step 0: All the requests come from the consumers to Registry of TM. LB of Registry (primary) takes these queries in Task_list in the waiting queue.
Step 1: TM checks the number of tasks (queries) in Task_list. If they are less than or equal 7, then there is no overloaded and go to step 2. But if they are more than 7, then there is overloaded and the first seven tasks will be executed at Registry. The next seven tasks will be executed at Failover Registry in the same time. Therefore, step 2 and step 5 will be executed in parallel.
Step 2: LB of Registry (primary) enters the requests to the ready queue after ordering them depending on the priority values (trust values of consumers) i.e., the request of consumer with higher trust value will be executed before the request of consumer with lower trust value.
Step 3: TM updates all databases.
Step 4: TM checks Task_list again, if it is empty then the algorithm is finished else go to step 1.
Step 5: TM migrates the next seven tasks (task 8 to task 15) from the waiting queue of LB of Registry (primary) to the waiting queue of LB of Failover registry (secondary).
Step 6: LB of Failover registry (secondary) enters the requests to ready queue after ordering them depending on the priority values (trust values of consumers). Then got to step 3.

The proposed dynamic load balancing algorithm is embedded with the proposed Grid monitoring system as shown in Figure 3. Firstly, Registry takes all requests from the consumers in the context. Registry is the primary load balancing in the system. It has two queues: waiting queue and ready queue. Secondly, if LB (primary) of Registry has overloaded, then the proposed dynamic load balancing will be loaded. If Registry is failed, Failover registry will replace it. Failover registry is the secondary load balancing in the system. It also has two queues: waiting queue and ready queue. LB (secondary) of Failover registry will work in the same time with LB (primary) of Registry in case of overloaded. Thirdly, Registry or Failover Registry or both will choose the suitable producers for the requests of the consumers. Finally, the producers send the replies to the consumers.

The proposed dynamic load balancing algorithm has some Characteristics:

- Security: It works with the proposed trust model depending on the trust value of every user.
- Management: It provides a way for consumers to obtain information about Grid resources as quickly as possible by Registry.
- Dependability: It recovers any failures in Registry by using Failover registry.
- Scalability: Any number of nodes or users can be added or deleted.
- Efficiency: It is used Java servlets. Java servlets are more efficient, easier to use, more powerful, and more portable.
- Flexibility: It is used Structured Query Language (SQL).
- Load balancing: It distributes the load in the system between two nodes; Registry and Failover registry.

5. Evaluation Results

5.1 Experimental Platform

Our Grid platform consists of as shown in Figure 4: 1) Hardware Components: Nodes: 5 PCs (Intel Pentium4 2.2 GHz processor, Intel RAM 256 MB) and 10 PCs (Intel Atom 1.66 GHz processor, Intel RAM 2 GB), and Interconnection Network: Gigabit Ethernet 1000Mbps. 2) Grid Middleware: Globus Toolkit 4.2.1. 3) Software Components: Operating System in all nodes: Linux Fedora 10, and Tools: Programs written in Java, Apache Ant for Java- based build tool, and Microsoft SQL server 2008.

This platform is heterogeneous because it has different hardware. But the software is homogeneous in all nodes. Every node has Linux Fedora 10 and Globus Toolkit 4 and programming interface (Java, Ant, and SQL).

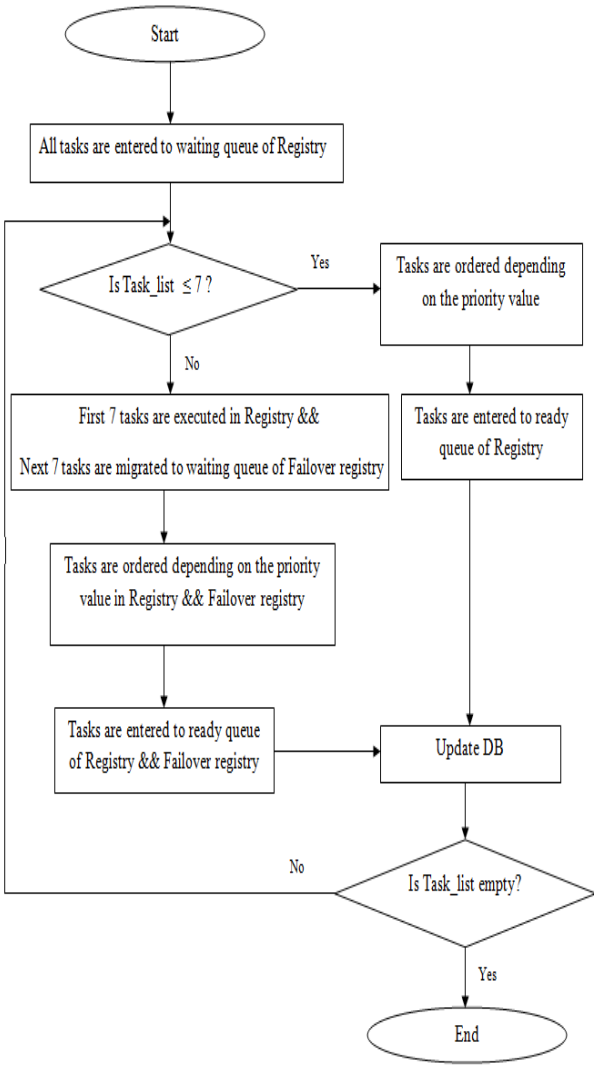

Figure 2. Flowchart of Proposed Dynamic Load Balancing

The proposed Grid monitoring system uses hundreds of databases that exist in Chiba University, Japan. Every producer has tens of databases about students, staffs, published papers, laboratory contents. The consumer can send any query after entering the system by his performance and recovering failure and published in [5].

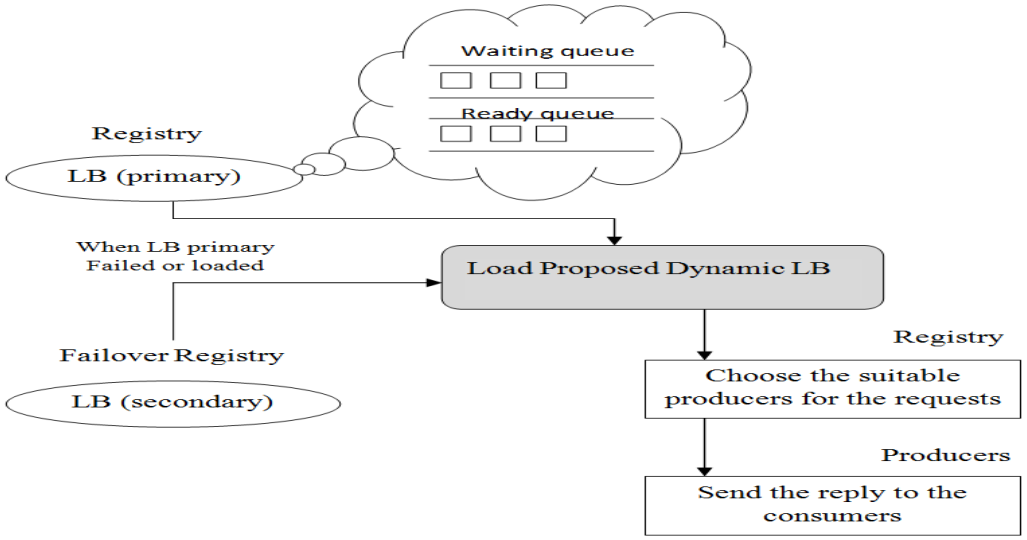

Figure 3. The proposed dynamic LB embedded with the proposed Grid monitoring system

Table 1. Comparison of Load Balancing Algorithms

Parameters	Round Robin	Random	Central Manager	Threshold	Local Queue	Central Queue	Proposed Dynamic LB
Overload Rejection	No	No	No	No	Yes	Yes	Yes
Fault Tolerant	No	No	Yes	No	Yes	Yes	Yes
Forecasting Accuracy	More	More	More	More	Less	Less	-----
Stability	Large	Large	Large	Large	Small	Small	Small
Centralized/ Decentralized	D	D	C	D	D	C	D
Dynamic/static	S	S	S	S	Dy	Dy	Dy
Cooperative	No	No	Yes	Yes	Yes	Yes	Yes
Process Migration	No	No	No	No	Yes	No	Yes
Resource Utilization	Less	Less	Less	Less	More	Less	More

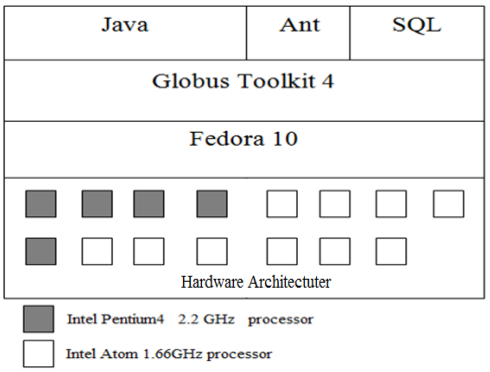

Figure 4. Experimental platform

5.2 Response Time (RT)

Response time is the average amount of time from the point a consumer sends out a request till the consumer gets the response. We measure response time depending on message size with fixed number of requests; 15 requests. We measure response time three times; one without load balancing (i.e. there may be overloaded), one with static load balancing (Central Manager algorithm), and one with the proposed dynamic load balancing. The result is shown in Figure 5.

All results of proposed dynamic LB that are less than or equal 512KB are slightly less than the results of both no load balancing and static load balancing. However, when the message size is more than 512 KB, the response time of static and dynamic LB is largely less than of that without load balancing.

5.3 System Utilization

System utilization is the ratio of time a system is busy (i.e. working for us); divided by the time it is available. Utilization is a useful measure in evaluating performance. For all the results discussed here, the number of requests is 15 requests. The utilization of producers is measured over time slice; every 60 seconds as shown in Figure 6. The utilization of proposed dynamic LB is larger than the static LB until the second 600. After this, the utilization of proposed dynamic LB is the same or slightly larger than the static LB.

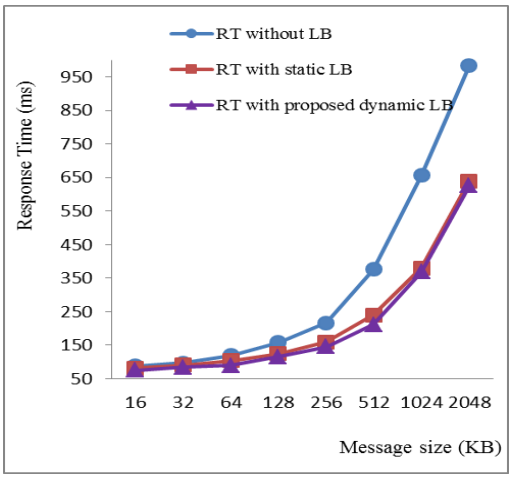

Figure 5. Response Time of three cases in the Proposed Grid Monitoring System

5.4 Throughput

Throughput is the amount of data transferred in one direction over a link divided by the time taken to transfer it, usually expressed in bits or bytes per second. People are often concerned about measuring the maximum data throughput rate of a communications link. The throughput is then calculated by dividing the file size by the time to get the throughput in megabits, kilobits, or bits per second. We measure the throughput as a function of data (message size) in Mega Bytes Per Second (MBPS) as shown in Figure 7.

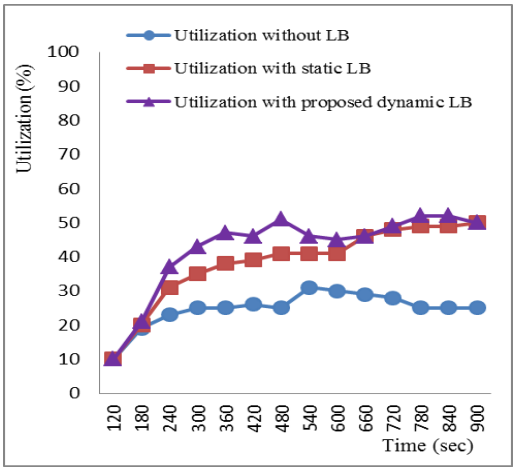

Figure 6. Utilization of three cases in the Proposed Grid Monitoring System

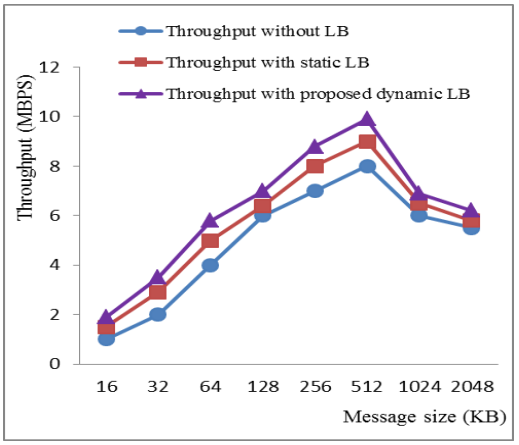

Figure 7. Throughput of three cases in the Proposed Grid Monitoring System

In case of proposed dynamic load balancing system, we note that the results are higher than of both no loading balancing system and static load balancing system. This is because the loaded at Registry is divided and the queries are served by Registry and Failover registry in dynamic environment; i.e. loaded is distributed at run time. All requests will be served depending on the priority value. So the transferred data will be high. We get the highest throughput when message size is 512 KB. So we recommended using message size with less than or equal 512 KB when working with the proposed Grid monitoring system to get high performance.

6. Comparison between all LB Algorithms

6.1 Response Time

We measure the response time twice; one as a function of message size as shown in Figure 8 and one as a function of number of users as shown in Figure 9. Since we could employ at most fifteen machines, it was impossible for us to actually implement hundreds of users. Instead, we used multiple user processes running on each machine. For example, to simulate the traffic generated by 300 users in the real world, we ran 20 traffic-generating processes on each of the fifteen machines. We measure response time depending on message size with fixed number of requests; 15 requests.

6.2 Throughput

We measure the throughput as a function of data (message size) in Mega Bytes Per Second (MBPS) as shown in Figure 10 and as a function of number of users as shown in Figure 11. Since we could employ at most fifteen machines, it was impossible for us to actually implement hundreds of users. Instead, we used multiple user processes running on each machine. For example, to simulate the traffic generated by 300 users in the real world, we ran 20 traffic-generating processes on each of the fifteen machines.

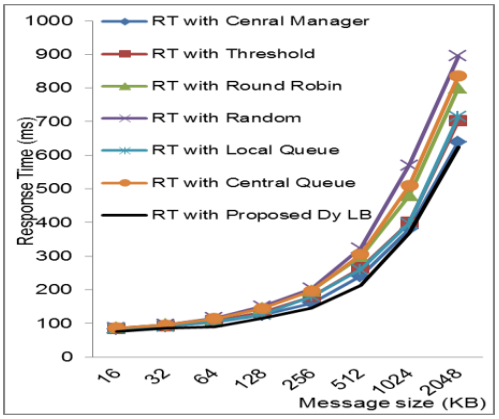

Figure 8. Comparing Response Time for 7 load balancing algorithms depending on the message size

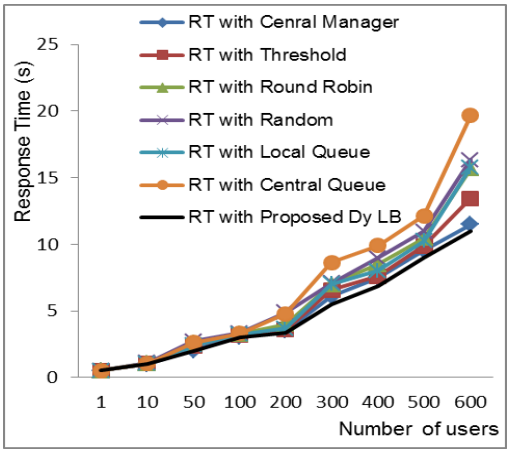

Figure 9. Comparing Response Time for 7 load balancing algorithms depending on the number of users

6. Conclusions

Load balancing is important in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload. In this paper, we integrate load balancing with proposed trust model and proposed Grid monitoring system. The proposed Grid monitoring system controls the relationship between the producers and consumers by registry, and recovers the failure by failover registry. The performance of proposed dynamic LB algorithm is evaluated by measuring the response time, utilization, and throughput depending on message size of query. The experiment results show the effectiveness of proposed dynamic load balancing in reducing the response time, maximizing system utilization, and increasing throughput.

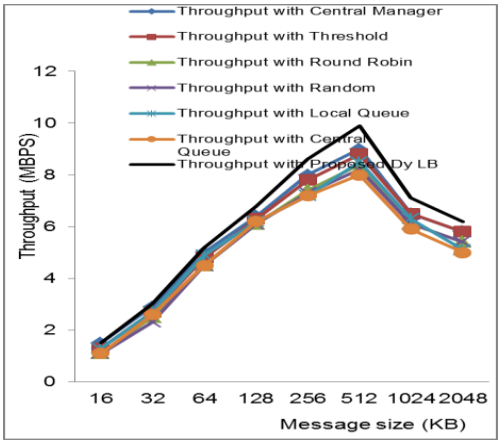

Figure 10. Comparing Throughput for 7 load balancing algorithms depending on the message size

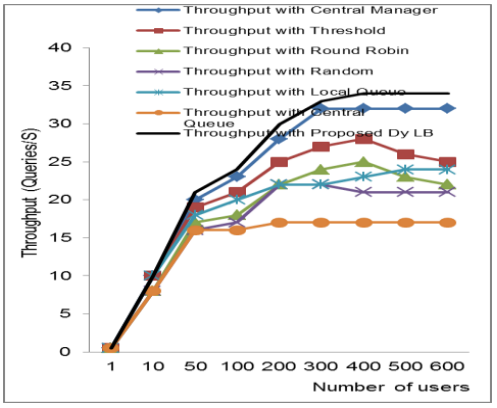

Figure 11. Comparing Throughput for 7 load balancing algorithms depending on the number of users

References

[1] Ch.Kopparapu: "Load Balancing Servers, Firewalls, and Caches", Wiley Computer Publishing, Edition 2002, pp. 1-7 (2002)

[2] Cheng-Zhong Xu, Francis Lau :" Load Balancing in Parallel Computers: Theory and Practice", Kluwer Academic Publishers, Boston (1997)

[3] Fran Berman, Geoffrey Fox , Tony Hey : "Grid Computing: Making the Global Infrastructure a Reality", Wiley Series in Comm. Networking & Distributed System (2003)

[4] Y. Zhu : "A survey on grid scheduling systems", Technical report, Department of Computer Science, (2003)

[5] Sherihan AbuElenin, Masato Kitakami : "Proposal of Grid Monitoring System with Fault Tolerance", Journal of Information Processing. 20 (2), pp.366-377 (2012)

[6] Sherihan Abu Elenin , Masato Kitakami :" Performance Analysis of Static Load Balancing in Grid", International Journal of Electrical & Computer Sciences IJECS/IJENS, Vol.11, Issue: 03, 2011, pp.57-63 (2011)

[7] L.Eager Derek, Edward D. Lazowska , John Zahorjan : "Adaptive load sharing in homogeneous distributed systems", IEEE Transactions on Software Engineering, v.12 n.5, p.662-675 (1986)

[8] Hendra Rahmawan, Yudi Satria : "The Simulation of Static Load Balancing Algorithms", International Conference on Electrical Engineering and Informatics, Malaysia (2009)

[9] Y. Wang and R. Morris:"Load balancing in distributed systems,"IEEE Trans. Computing. C-34, no.3, pp. 204-217

[10] Sandeep Sharma, Sarabjit Singh, , Meenakshi Sharma : "Performance Analysis of Load Balancing Algorithms", Academy of science, engineering and technology, issue 38, February 2008, pp. 269-272 (2008)

[11] Belabbas Yagoubi , Yahya Slimani : "Dynamic Load Balancing Strategy for Grid Computing", World Academy of Science, engineering and technology, (19) 2006, pp. 90-95 (2006)

[12] B.Tierney, R.Aydt,D.Gunter:"A Grid Monitoring Architecture". http://www-didc.lbl.gov/GGFPERF/GMAWG/papers/GWD-GP-16-2.pdf (2004)

[13] Sherihan Abu Elenin , Masato Kitakami : " Trust Management of Grid System Embedded with Resource Management System", IEICE Transaction Information System, vol. E94-D, No.1, 2011, pp. 42-50 (2011)

[14] W. Leinberger, G. Karypis, V. Kumar, and R. Biswas : " Load balancing across near-homogeneous multi-resource servers. In 9th Heterogeneous Computing Workshop, pages 60–71 (2000)

[15] F. Berman, G. Fox, and Y. Hey :" Grid Computing: Making the Global Infrastructure a Reality", Wiley Series in Communications Networking & Distributed Systems (2003)

[16] William Leinberger, George Karypis, Vipin Kumar, "Load Balancing Across Near-Homogeneous Multi-Resource Servers", 9th Heterogeneous Computing Workshop, Mexico (2000)