Performance Analysis of Static Load Balancing in Grid

Sherihan Abu Elenin[1,2] and Masato Kitakami[3]

Abstract— **Monitoring is the act of collecting information concerning the characteristics and status of resources of interest. The Grid Monitoring Architecture (GMA) based on simple Consumer/ Producer architecture with an integrated system registry and distinguishes transmission of monitoring data and data discovery logically. We propose grid monitoring system based on GMA. The proposed grid monitoring system consists of producers, registry, consumers, and failover registry. The registry is used to match the consumer with one or more producers, so it is the main monitoring tool. The failover registry is used to recover any failure in the main registry. The structure of proposed grid monitoring system depends on java Servlet and SQL query language. Load balancing (LB) should be added to the system to overcome the message overloaded. Load balancing algorithms can be static or dynamic. This paper evaluates the four types of static load balancing algorithms; Randomized, Round Robin, Threshold, and Central Manager algorithms. We evaluate the performance of the system by measuring the response time, and throughput. Central Manager algorithm introduces the smallest response time and the highest throughput.**

Index Terms— ***Grid computing, monitoring system, load balancing, response time, Security, throughput.***

I. Introduction

Grid computing has emerged as an important new field. It focuses on large-scale resource sharing, innovative applications, and high-performance orientation. The distinguished advantage of Grid against traditional distributed computing is that it can integrate a large number of computing resources as well as data resources to solve some kind of challenges. As more and more web service applications, Grid computing has extended its territory from traditional computing Grid to public and provides Grid services. The purpose of Grid is to realize coordinated resource sharing and problem in dynamic virtual organizations (VOs) [1]. In this environment, the security problem is a hot topic in Grid research due to the dynamics and uncertainty of Grid system. The Grid security issues can be categorized into three main categories: architecture related issues, infrastructure related issues, and management related issues [8].

This paper is focused on Grid management. The different management issues that Grid administrators are worried about are credential management, trust management, and monitoring related issues. The ability to monitor and manage distributed computing components is critical for enabling high performance distributed computing. Monitoring data is needed to determine the source of performance problems and to tune the system for better performance [9]. Monitoring is the act of collecting information concerning the characteristics and status of resources of interest. Monitoring is also crucial in a variety of cases such as scheduling, data replication, accounting, performance analysis and optimization of distributed systems or individual applications, self-tuning applications, and many more [8]. The functions of monitoring are correctness checking, performance enhancement, dependability or fault tolerance, performance evaluation, debugging and testing, control or management, and security.

Most existing monitoring systems work with network or cluster systems. There are several research systems implementing the Grid Monitoring Architecture (GMA) [6]: Autopilot, R-GMA, MDS, etc. Autopilot [11] is a framework for enabling applications to dynamically adapt to changing environments. It aims to facilitate end-users in the development of application. R-GMA [12] combines grid monitoring and information services based on the relational model. Although the robustness of R-GMA, it has three drawbacks: flow of data, loss of control message, and single point of failure. The Monitoring and Discovery System (MDS) [3] of the Globus Toolkit (GT) is a suite of components for monitoring and discovering Grid resources and services. It has many problems such as it is too difficult to install.

In this paper, we focus on monitoring management in Grid system. The proposed Grid monitoring system is also based on the GMA [6]. GMA is the basis of most of monitoring system. The goal of GMA is to provide a minimal specification that will support required functionality and allow interoperability. We design a simple Grid monitoring system. The proposed system components are producers, registry, consumers, and failover registry. The goals of this system are to provide a way for consumers to obtain information about Grid resources as quickly as possible, and to recover any faults in the system. There is no direct relationship between producer and consumer. The monitoring tool is registry. It manages and controls the relationship between all producers and consumers existing

[1,3] Graduate School of Advanced Integration Science,
Chiba University, Japan 263-8522
[2] Faculty of Computers and Information, Mansoura University, Egypt
Emails: [1,2]sherihan@graduate.chiba-u.jp,
[3] kitakami@faculty.chiba-u.jp

in the system. Failover registry is responsible of taking place of main registry in case of failure.

In the proposed Grid monitoring system, we observe that there may be overloaded in Registry if the number of requests is large. So load balancing should be added to the proposed Grid monitoring system in order to get better performance. Load balancing is not a new concept in the server or network space. Load balancing is a technique applied in parallel system that is used to reach optimal system condition, which is workloads are evenly distributed amongst computers, and as its implication will decrease programs execution time. Load balancing is dividing the amount of work that a computer has to do between two or more computers so that more work gets done in the same amount of time and, in general, all users get served faster. Load balancing can be implemented with hardware, software, or a combination of both [5].

Load balancing can be static or dynamic [14]. Static load balancing algorithms are Round Robin algorithm, Randomized algorithm, Central Manager algorithm, and Threshold algorithm. Dynamic load balancing algorithms are Central Queue algorithm, and Local Queue algorithm. In this paper, we apply the static load balancing algorithms in the proposed system to get better performance. Based on Table 1 [14], the dynamic load balancing algorithms aren't suitable for the proposed Grid monitoring system. In the end, we compare the four static load balancing algorithms to select the best one that can work well with the proposed system.

Table 1: Parametric Comparison of Load Balancing Algorithms

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

II. Static Load Balancing Algorithms

In static load balancing, the performance of the processors is determined at the beginning of execution. Then depending upon their performance the work load is distributed in the start by the master processor [4]. The slave processors calculate their allocated work and submit their result to the master. A task is always executed on the processor to which it is assigned that is static load balancing methods are non-preemptive. 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. There are four types of static load balancing: - Round Robin algorithm, Randomized algorithm, Central Manager algorithm, and Threshold algorithm.

Round Robin algorithm [13] 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. Processors choosing are performed locally on each processor, independent of allocations of other processors. Advantage of Round Robin algorithm is that it does not require inter-process communication. In general Round Robin is not expected to achieve good performance in general case.

Randomized algorithm [13] uses random numbers to choose slave processors. The slave processors are chosen randomly following random numbers generated based on a statistic distribution. Randomized algorithm can attain the best performance among all load balancing algorithms for particular special purpose applications.

Central Manager algorithm [14], 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. The central processor is able to gather all slave processors load information, thereof the choosing based on this algorithm are possible to be performed. The load manager makes load balancing decisions based on the system load information, allowing the best decision when of the process created. High degree of inter-process communication could make the bottleneck state.

In Threshold algorithm [14], 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. Two threshold parameters t_under and t_upper can be used to describe these levels. Under loaded: load < t_under , Medium : t_under ≤ load ≤ t_upper , and Overloaded: load > t_upper. Initially, all the processors are considered to be under loaded. When the load state of a processor exceeds a load level limit, then it sends messages regarding the new load state to all remote processors, regularly updating them as to the actual load state of the entire system.

III. Proposed Grid Monitoring System

A. *Overview*

Most of Grid monitoring and information service have shortcomings and not easy to use [10]. First, they are too large and too difficult to install, to configure and to deploy. For example, to use MDS4 you need configure third party monitoring tools such as Ganglia or Hawkeye. Second, some functions they provide may be limited to some specific projects; these functions may be useless to another project and may reduce the performance of this project. Finally, some protocols these tools rely on also have defects.

We design a simpler model after we analyze the requirements of Grid monitoring and information service, and implement it. The proposed Grid Monitoring System is based on the Grid Monitoring Architecture (GMA) [6] as shown in Fig. 1.

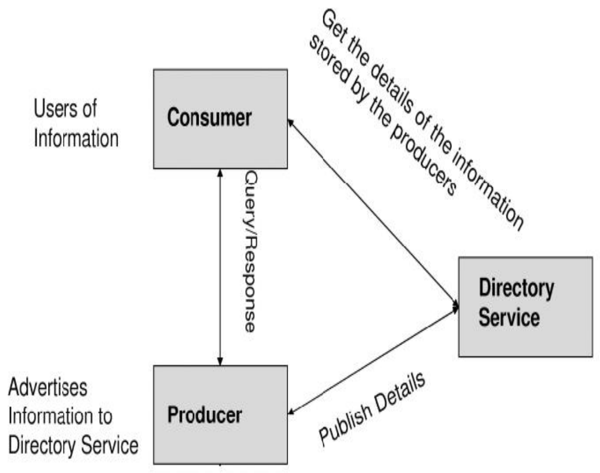

Users of Information
Consumer
Get the details of the information stored by the producers
Query/Response
Directory Service
Advertises Information to Directory Service
Producer
Publish Details

Fig. 1. Grid Monitoring Architecture (GMA)

In order to satisfy the requirement of Grid monitoring, Global Grid Forum (GGF) recommend Grid Monitoring Architecture (GMA) as Grid monitoring mechanism [6]. 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. In GMA, all of monitoring data are events which are based on timestamp for storing and transferring. For example, CPU usages, memory usage, thread status and error information. The Grid Monitoring Architecture consists of three types of components: Directory Service (Registry), Producer and Consumer.

The architecture of proposed Grid monitoring system and the Communications between the Producer and the Consumer is shown in Fig. 2. 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. The structure of proposed Grid monitoring system depends on java Servlet and SQL query language. Java servlets are more efficient, easier to use, more powerful, more portable, and cheaper than traditional Common Gateway Interface (CGI). Structured Query Language (SQL) is a database computer language designed for managing data in relational database management systems (RDBMS), and originally based upon relational algebra. Users are offered all the flexibility that SQL query language brings.

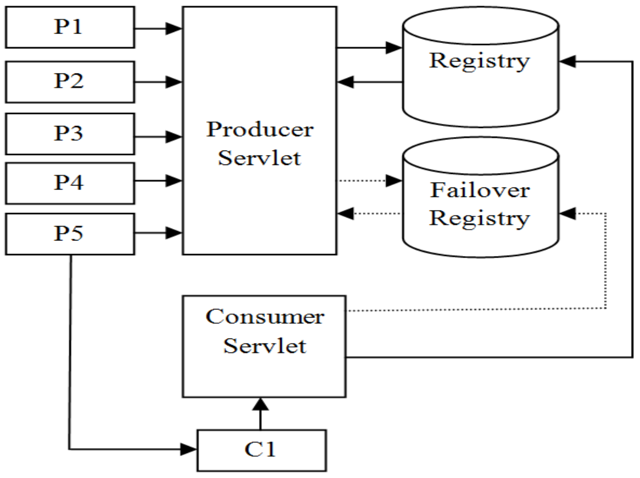
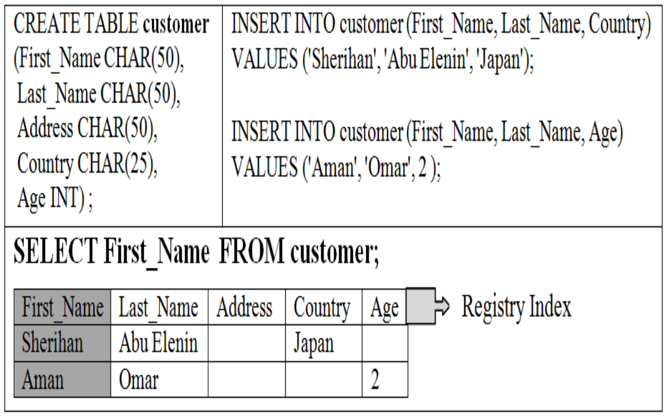


Fig. 2. Proposed Grid Monitoring System

B. *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 as shown in Fig. 3. So the producers in our Grid monitoring system are source of data. Each producer has interface and Servlet. Producer interface communicates with producer Servlet to build data base. The functions that are supported by the producer are creating tables, inserting data into tables, deleting data from tables, and updating data in tables.

Registry acts as a discovery Grid service to find relevant producers matching the query of a consumer. Registry schema consists of four layers: register layer, data layer, service layer, and republish layer. Register layer is responsible of registering all producers and consumers in the system. Data layer as shown in Fig. 4 contains the description of data base exist in all producers. As the example in Fig. 4, The Registry index contains the table name and description of it. For example, if the table "customer" in Fig. 3 exists in producer1, then Data layer in Registry contains "Producer1 has 'customer table' with description {First_Name, Last_Name, Address, Country, Age}. Service layer and republish layer take request and get reply, respectively. The functions that are supported by the registry are registering both producers and consumers, adding entry from producers, updating entries from producers, removing entries from producers, and searching about suitable producer for 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.



```
CREATE TABLE customer
(First_Name CHAR(50),
 Last_Name CHAR(50),
 Address CHAR(50),
 Country CHAR(25),
 Age INT) ;
```

```
INSERT INTO customer (First_Name, Last_Name, Country)
VALUES ('Sherihan', 'Abu Elenin', 'Japan');

INSERT INTO customer (First_Name, Last_Name, Age)
VALUES ('Aman', 'Omar', 2 );
```

```
SELECT First_Name FROM customer;
```

First_Name	Last_Name	Address	Country	Age
Sherihan	Abu Elenin		Japan	
Aman	Omar			2

Registry Index

Fig. 3. SQL Example

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. The function of consumer is sending request to registry to find data by SQL SELECT statement in browser interface.

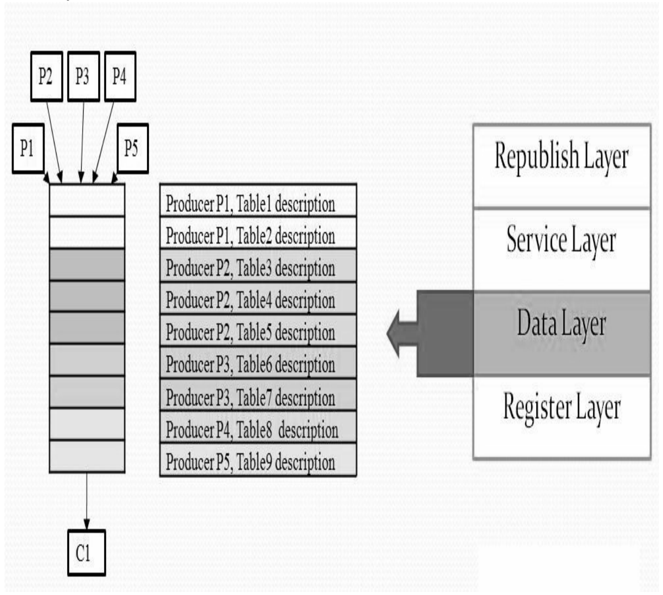

P2
P3
P4
P1
P5
Producer P1, Table1 description
Producer P1, Table2 description
Producer P2, Table3 description
Producer P2, Table4 description
Producer P2, Table5 description
Producer P3, Table6 description
Producer P3, Table7 description
Producer P4, Table8 description
Producer P5, Table9 description
C1
Republish Layer
Service Layer
Data Layer
Register Layer

Fig. 4. Registry Schema

C. *The Overall 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 [7]. 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 proposed Grid monitoring system, we observe that there may be overloaded in Registry or Producers 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.

IV. Evaluation Results

To evaluate the performance of the proposed Grid monitoring system, we pay more attention to the following parameters: response time, and throughput. We measure these two performance metrics twice. One depends on the message sizes in the system, and the other depends on the number of users.

A. *Experimental Platform*

Our Grid platform consists of: 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: Linux Fedora 10, and Tools: Programs written in Java, Apache Ant for Java- based build tool, and Microsoft SQL server 2008.

B. *Domains Structure in the Experimental*

The system is divided into two domains (Domain1 and Domain2) as shown in Fig. 5. Domain1 consists of G1, G2, G3, G4, G5, G6, G7, and G8. Domain2 consists of G1, G2, G3, G4, G5, G6 and G7. Trust Manger (TM) with registry (R1) of Domain1 is existed in G1 and there is back up version called failover registry existed in G8. In Domain2, Trust Manger (TM) with registry (R2) is existed in G1 and there is back up version called failover registry existed in G6. We have two nodes that exist in the two domains; G5 and G7. In the system, always every node is called with its domain name such as G5: Domain2.

Failover registry is an important tool to recovery the failure. For example in Domain2, if G1:Domain2 failed, the request will go to G6:Domain2 (Failover Registry), and it works all functions of Registry. The algorithm will be:

IF (G1:Domain2 == Fail)
THEN Goto (G6:Domain2)
ELSE Get the request from consumer.

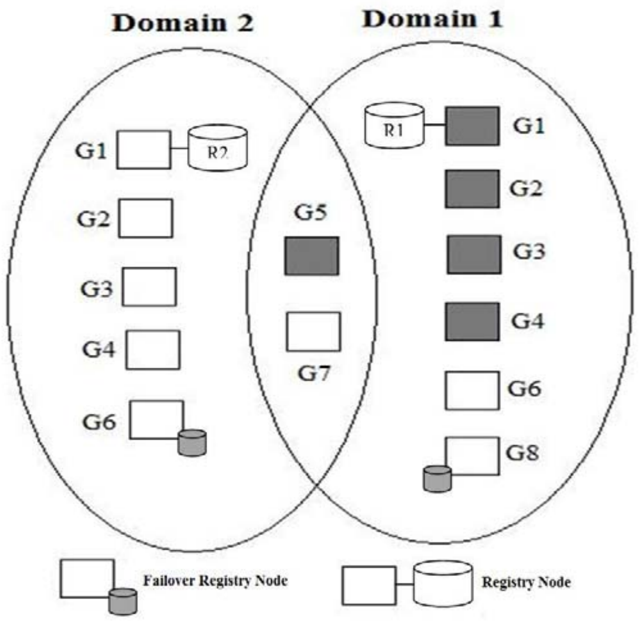


Fig. 5. Domains structure in the experiment

C. Response Time

Response time (RT) 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 twice; one as a function of message size as shown in Fig. 6 and one as a function of number of users as shown in Fig. 7.

We measure response time depending on message size with fixed number of requests; 15 requests. In Fig. 6, Randomized algorithm gives the highest response time in all message sizes. This is because there is no steps or any calculations of loaded in the system. It depends on choosing the producer in random manner. So the response time takes long time. Round Robin algorithm introduces results near to Randomized algorithm. Central Manager algorithm is the best in all results because it introduces the smallest response time and it increases very simply; especially when message size is less than or equal 512KB. When message size is more than 512KB, response time is increased in huge manner. So we recommended working in this proposed system with message size less than 512KB.

In Fig. 7, when number of users is one, all algorithms show the same response time. When number of users is 10, the difference between response times from the four algorithms is slightly small. This is because the number of users is less than the number of nodes in the system. In the remaining results, Randomized algorithm introduces the biggest response time and Central Manager algorithm gives the smallest response time. So Central Manager algorithm is also the best algorithm. Threshold and Round Robin algorithms give mediate results. We observe that the response time in all four algorithms until 300 users is small. When number of users is more than 300, the response time in all algorithms are largely increased. If the number of nodes in the system is increased, the system can serve many users. So we recommended working in this proposed system with number of users less than 300.

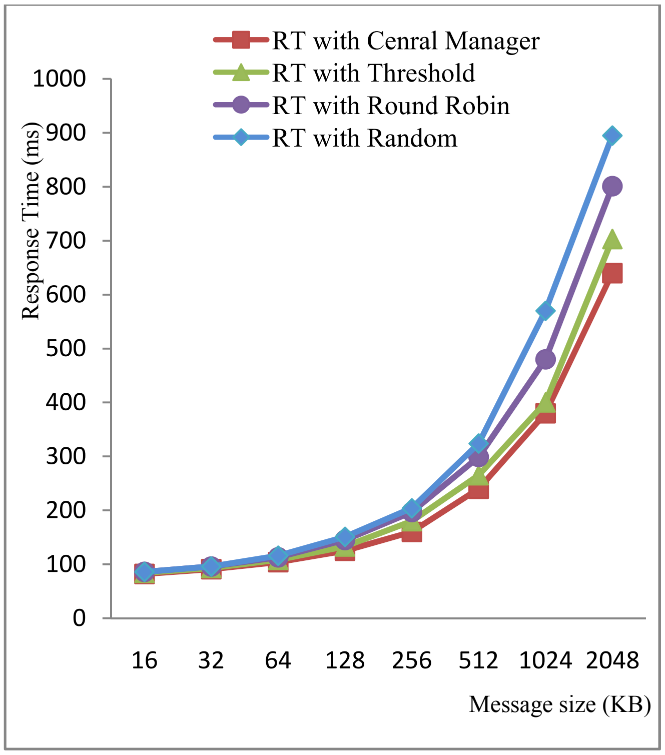


Fig. 6. Comparing Response Time for 4 static load balancing algorithms depending on the message size

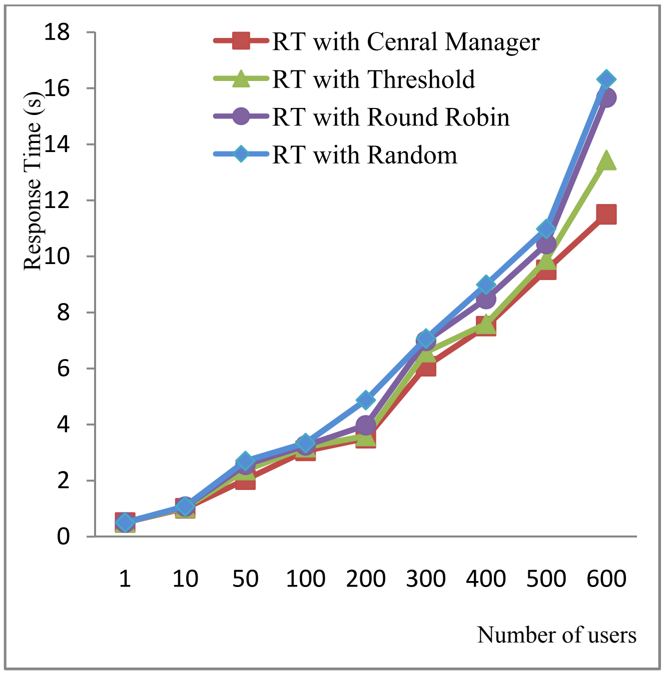


Fig. 7. Comparing Response Time for 4 static load balancing algorithms depending on the number of users

D. *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. A typical method of performing a measurement is to transfer a large file and measure the time taken to do so. 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 Fig. 8 and as a function of number of users as shown in Fig. 9.

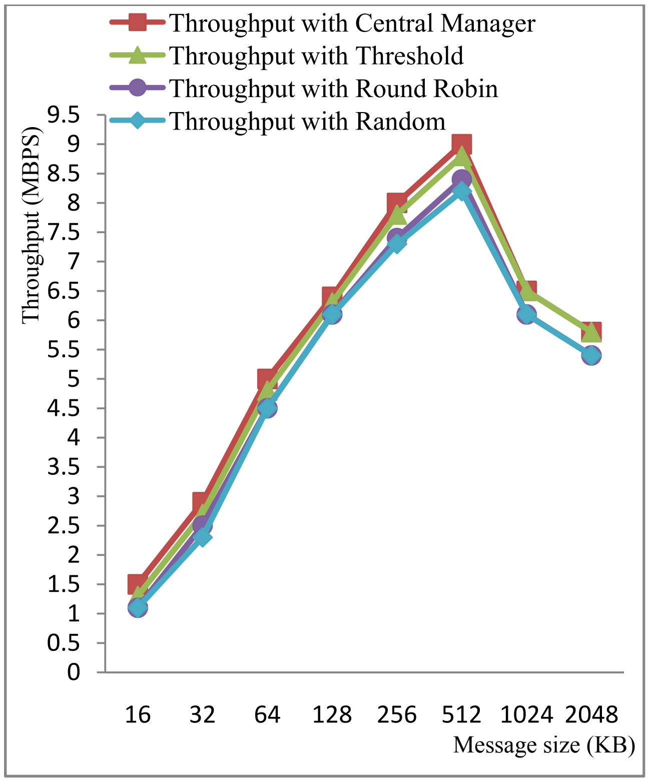


Fig. 8. Comparing Throughput for 4 static load balancing algorithms depending on the message size

In Fig. 8, Round Robin and Randomized algorithms give the smallest throughput. Low throughput means low data flow. Round Robin algorithm depends on Round Robin order, so there is no strategy to know the best loaded node. Randomized algorithm is the same story. We think they can be good in the specific projects. On the other hand, Central Manager algorithm introduces the highest throughput. High throughput means high data flow. This algorithm depends on choosing the producer with smallest load so it chooses the minimal one. We observe that the throughput is increased when message size is less than or equal 512KB in all algorithms. But it decreased when message size is more than 512KB in all algorithms. So we recommended using message size with less than 512 KB when working with the proposed Grid monitoring system to get high performance.

In Fig. 9, when number of users is one, all algorithms show the same throughput. This is because the system is not worked in parallel. When number of users is 10, Central Manager and Threshold algorithms give the same result, and Randomized and Round Robin give the same throughput. In Central Manager algorithm, throughput is increased with number of users until 300 users. It is constant after 300 users. In Threshold algorithm, throughput is increased with number of users until 400 users. It is decreased after 400 users. In Round Robin algorithm, throughput is increased with number of users until 400 users. It is decreased after 400 users. In Randomized algorithm, throughput is increased with number of users until 300 users. It is decreased after 400 users. These show the capacity of each algorithm in number of users.

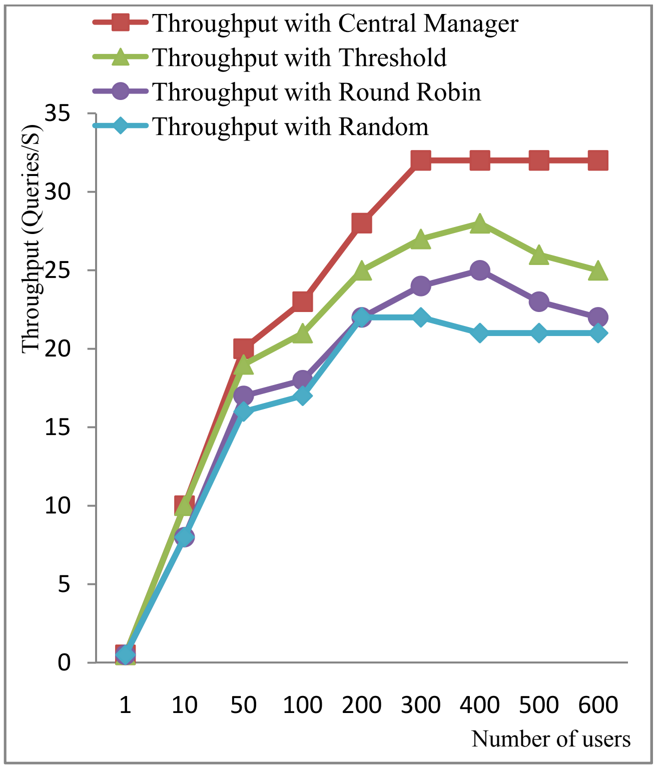


Fig. 9. Comparing Throughput for 4 static load balancing algorithms depending on the number of users

V. Conclusions and Future work

The monitoring system in distributed system is new topic. Previous works over monitoring system is interested in cluster computing, network, or P2P systems. In Grid systems, most of monitoring system is under development and isn't executed in real projects. In the proposed Grid monitoring system, we focus in the system management by controlling the relationship between the producers, consumers, and registry, and its fault tolerance by adding failover registry in every domain. The overloaded is a big problem in the system, so load balancing should be added. The load balancing algorithms are two types: static or dynamic. The performance of four types of static load balancing is evaluated by measuring the response time, and throughput. Round Robin, Randomized, Central Manager, and Threshold algorithms are evaluated in the proposed Grid monitoring system twice from point of view of message sizes and number of users. Central Manager algorithm is the best and has introduced good performance. Randomized algorithm has introduced bad results.

For future work, the dynamic load balancing algorithms should be modified to be suitable with Grid systems. The complete evaluation should be made between all load balancing algorithms in Grid.

ACKNOWLEDGMENT

The authors are grateful to Prof. Hideo Ito for his discussions and advices.

REFERENCES

[1] I. Foster, C. Kesselman, S. Tuecke: "The Anatomy of the Grid: Enabling Scalable Virtual Organizations". International Journal of High Performance Computing Applications, 15(3):200-222, 2001.

[2] Yuanzhe Yao, Binxing Fang, Hongli Zhang, and Wie Wang, "PGMS: A P2P-Based Grid Monitoring System" , Third International Conference of Grid and Cooperative Computing(GCC 2004) China, 2004.

[3] Globus Toolkit: http://www.globus.org/

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

[5] http://www.cyquator.com/Html/load.html

[6] Brian Tierney, R.Aydt, D.Gunter etc. "A Grid Monitoring Architecture". http://www-didc.lbl.gov/ GGF-PERF/GMAWG/papers/GWD-GP-16-2.pdf, 2004.

[7] Sherihan Abu Elenin and 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.

[8] Anirban Chakrabarti, Grid Computing Security, Springer, 1 edition 2007, pages 33-45.

[9] Brian Tierney, Brian Crowley, Dan Gunter, Mason Holding, Jason Lee, Mary Thompson, "A Monitoring Sensor Management System for Grid Environments", Cluster Computing Volume 4, Number 1, March 2001, pp:19-28.

[10] Weibin Pei, Zhongliang Chen, Chunhao Feng, Zhi Wang, "Design and Implementation of a Plain Grid Monitoring and Information Service", Fifth IEEE International Symposium on Network Computing and Applications (NCA'06) 2006, PP: 277-284.

[11] R.L. Ribler, J.S. Vetter, H. Simitci, D.A. Reed, "Autopilot: adaptive control of distributed applications", in: Proceedings of the Seventh IEEE Symposium on High-Performance Distributed Computing, 1998, pp. 172–179.

[12] P. Bhatti, A. Duncan, S. M. Fisher, M. Jiang, A. O. Kuseju, A. Paventhan and A. J. Wilson, "Building a robust distributed system: some lessons from R-GMA ", international Conference on Computing in High Energy and Nuclear Physics (CHEP '07) September 2007, Victoria, Canada.

[13] Hendra Rahmawan, Yudi Satria Gondokaryono, "The Simulation of Static Load Balancing Algorithms", 2009 International Conference on Electrical Engineering and Informatics, Malaysia.

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