International Journal of Information Technology and Knowledge Management
December 2012, Volume 6, No. 1, pp. 53-60

COMPARATIVE STUDY ON LOAD BALANCING TECHNIQUES IN DISTRIBUTED SYSTEMS

P. Beaulah Soundarabai*[1], Sandhya Rani A.[1], Ritesh Kumar Sahai[1], Thriveni J.[2], K.R. Venugopal[2] and L.M. Patnaik[3]

Abstract: Scalability becomes the basic need for distributed systems. With the increase of users, load on application servers also keep increasing. This requires a critical action to balance the load on servers. Load balancing is the concept of balancing load on servers using various load balancing techniques. In this paper we discuss various load-balancing techniques that are currently available. We discuss inherent characteristics, operational process, advantages and disadvantages of various static load-balancing techniques and dynamic load-balancing techniques. Also we made a comparative study on various parameters of the load-balancing techniques.

Categories and Subject Descriptors: Distributed Systems, Load Balancing; Load Balancing Techniques; Static Load Balancing Techniques; Dynamic Load Balancing Techniques; Performance;

General Terms: Load Balancing Algorithms, Reliability, Performance Measures

Keywords: Load Balancer; Availability; Scalability; Load Balancing Techniques; Static Load Balancing Techniques; Dynamic Load Balancing Techniques;

1. INTRODUCTION

A server program that is installed on a targeted system has limited resources. These resources include system memory, hard disk space, and processor speed. Since the server capacity is limited, so it can handle only certain number of clients. With more number of clients a server will be overloaded and that may lead to slow down of the performance, hang and crash issues. So, it is crucial to balance the load on server and this can be achieved by keeping copies of servers and distribute the load among them.

Load balancing is the process of making a group of servers participate in the same service and do the same work. The general purpose of load balancing is to increase availability, improve throughput, reliability, maintain stability, optimize resource utilization and provide fault tolerant capability. As the number of servers grows, the risk of a failure anywhere increases and such failures must be handled carefully. The ability to maintain unaffected service during any number of simultaneous failures is termed as high availability.

[1] Department of Computer Science, Christ University, Hosur Main Road, Bangalore, India
[2] Department of Computer Science and Engineering, Univesity Visvesvaraya College of Engineering, Bangalore University, Bangalore, India
[3] Honarary Professor, Indian Institute of Science, Bangalore, India.
* *beaulah.s@christuniversity.in*

Load balancing is also provided by few operating systems. Microsoft's Network Load Balancing (NLB), a software-based solution that allows you to effortlessly cluster multiple machines [3]. There are variety open source load balancers and load balancing software available for Linux, such as Linux Virtual Server, Ultra Monkey, Red Hat Cluster Suite, High Availability Linux (Linux HA); which can be used efficiently with most of the network services, including FTP, HTTP, DNS, SMTP, POP/IMAP, VoIP, etc.

2. LOAD BALANCING CONCEPT

"Load balancing" means an even distribution of the total load amongst all serving entities [2].

Load balancing is very essential in distributed computing systems to improve the quality of service by managing customer loads that are changing over time. The request demands of incoming requests are optimally distributed among available system resources to avoid resource bottlenecks as well as to fully utilize available resources [5]. Load balancing also provides horizontal scaling e.g., adding computing resources in order to address increased loads.

2.1 Client Server Architecture

In typical client server architecture, client entities make request to the servers which in turn process the request and send the response back to the requested client. Such a request initiated from the client entity is referred as the "Work

Load". The volume of such work load transmitted between client and server is the "Traffic". In general the servers are responsible for:

- Receiving the request from client and validating.
- Processing the client request.
- Sending the response back to the requested client.

Generally most of the servers spend at least 50% of time in processing the client request based on their functionality. Quality of service is degraded by the growing workloads and traffic bottlenecks. Although there are options to upgrade the servers with the high-end system peripherals, however it is not the optimal solution. Hence there is a need for balancing the load on the servers and it is the right approach to handle work load issues and avoid the congestion.

2.2 Load Balancer

Simple load balancer architecture is depicted in the Figure 1. Client's requests for services terminate at *Load Balancer* [1], which in turn forwards the requests to the servers, based on the various load balancing algorithms and mechanisms.

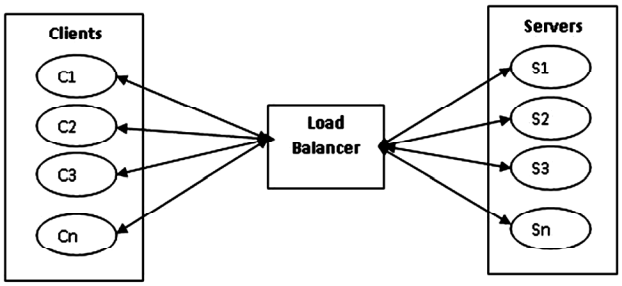

Figure 1: General Load Balancer

In today's redundant high-availability computer systems it is common that the incoming network traffic is distributed on network level by using any one of the commonly used network load balancing algorithms (like: random allocation, round-robin allocation, weighted round-robin allocation, etc.) [4]. These algorithms predominantly use network parameters of incoming traffic to make decisions where to forward traffic, without considering any other information like current load of application or database servers.

3. TYPES OF LOAD BALANCING

Load balancing of servers can be implemented in different ways. These methods of load balancing can be set up in the load balancer based on available load balancing types. There are various algorithms used to distribute the load among the available servers.

Load balancing techniques are normally used for balancing the workload of distributed systems. Load balancing techniques are broadly classified into two types namely "Static load balancing" and "Dynamic load-balancing". The main purpose of load balancing techniques, either static or dynamic, is to improve performance by redistributing the workload among available server nodes. Dynamic load balancing techniques reacts to the current system state, whereas static load balancing techniques depends only on the average behaviour of the system in order to balance the workload of the system [6]. This makes dynamic techniques necessarily more complex than static one. But, dynamic load balancing policies have been believed to have better performance than static ones [7].

3.1 Static Load Balancing Techniques

In this category of static load balancing techniques, the performance of the server nodes is determined at the initial stage of process. Then based on their performance the work load is distributed by the master node. The slave nodes evaluate their allocated work and then send their result to the master node.

A job is always executed on the server node to which it is assigned that is static load balancing techniques are non-preemptive. The main intention of static load balancing method is to reduce the overall execution time of a concurrent program while minimizing the communication delays. A usual disadvantage of all static methods is that the final selection of a server node for process allocation is made when the process is initially created and it cannot be changed during process execution in order to make variations in the system load.

3.1.1 Random Allocation

In a random allocation method, the client requests for example HTTP requests are assigned to any server picked randomly among the group of available servers. In such a case, one of the servers may be assigned more requests to process, while the other servers are sitting idle. However, on an average, each server gets its share of the client load due to the random selection.

3.1.2 Round-Robin Allocation

In a round-robin algorithm, the load balancer master node assigns the requests to a list of available servers on a rotating basis. The first request is assigned to a server which picked randomly from the list of available servers and for the subsequent requests, the master node follows the circular order to redirect the incoming request. Once a server is assigned with a request, the server is moved to the end of the list. This makes the load to be equally shared among available servers.

3.1.3 Weighted Round-Robin Allocation

Weighted Round-Robin method is an extended version of the round-robin method which overcomes the demerits of the normal round robin algorithm. In this weighted round-robin method, one can allocate a weight to each server in the list so that if one server is capable of handling twice as much load as the other, the powerful server gets a weight of two. In such cases, the master node allocates two requests to the powerful server where as single request is allocated to the weaker servers.

Table 1 compares the basic static load balancing algorithms with their advantage and limitations.

Table 1
Comparison of Basic Static Load Balancing Algorithms

Algorithms	*Comparisons*	
	Advantage	*Limitation*
Random Allocation	Simple to implement.	Can lead to overloading of one server while under-utilization of others.
Round-Robin Allocation	Better than random allocation because the requests are equally divided among the available fashion.	Round robin algorithm is not enough for load balancing based on processing overhead required and if the server specifications are not identical to each other in the server group.
Weighted Round-Robin Allocation	Takes care of the capacity of the servers in the group.	Does not consider the advanced load balancing requirements such as processing times for each individual request.

3.1.4 Central Manager Allocation

In this central manager allocation algorithm [13], a central manager node selects the host for new process. A server node which is least loaded when compare to all other nodes in the group of servers is selected when process is created. In order to select the least loaded server, the load manager uses the server's current system load information which will be locally stored on the manager node. This information is updated by all the remote server nodes which are in the group by sending a message every time when the load on them changes. Since all the server nodes have to update their load status to central manager, there will be more inter-process communication which may lead to network congestion state.

3.1.5 Threshold Based Allocation

In this algorithm, the processes are allocated to the server nodes as soon as they are created. Nodes for new processes are selected locally without sending remote messages. Every server node keeps a local copy of the system's current load. The current load of a server node could be in one of the following three levels: under loaded, medium and overloaded. Two threshold parameters tunder and tupper can be used to describe these levels [8].

Under loaded - load < tunder

Medium - tunder ≤ load ≤ tupper

Overloaded - load > tupper

At the initial stage, all the server nodes are assumed to be under loaded. When the load state of a server node crosses the load level limit, then it sends messages to all other remote nodes to inform its new load state.

If the load state is not in the overloaded level then, the process is assigned locally. If it is overloaded then, a remote under loaded server node is selected and assigned the process. This algorithm has less inter process communication but more number of local process allocations. The later decreases the overhead of remote process allocations and the overhead of remote memory accesses, which leads to improvement in performance [8]. The main drawback of this algorithm is that all processes are assigned locally when all remote server nodes are overloaded. A load on one overloaded node can be higher than the other overloaded nodes. This interrupts the load balancing operation which leads to increase in the execution time of an application.

3.2 Dynamic Load balancing Techniques

Dynamic load balancing techniques depend on recent system load information and determine the job assignments to server nodes at run time. In case of dynamic approach, the load balancing decisions are based on the current state of the system and hence work loads are allowed to shift dynamically from an overloaded node to an under-loaded node to get faster response from the server nodes. This ability to respond to variations in the system is the main advantage of the dynamic load balancing. But, since the load balancer has to continuously monitor the current system load on all the nodes, it becomes an extra overhead as monitoring consumes CPU cycles. So, a proper decision must be taken when to invoke the monitoring operation by the load balancer.

In the below section, we discuss the various policies of dynamic load balancing and also discuss about different types of existing dynamic load balancing techniques along with their relative merits & demerits.

3.2.1 Policies in Dynamic Load Balancing Algorithms

Load balancing algorithms can be defined by their implementation of the following policies [10][13]:

3.2.1.1 Information Policy

Information policy is mainly responsible for collection of system state information. This policy specifies what kind of workload information to be collected, when it has to be collected and from where. In case of local state information, the state information of neighboring nodes is collected where as in case of global state information; state information of all the nodes in the system is collected for making scheduling decisions better.

3.2.1.2 Triggering Policy

Triggering policy specifies the appropriate time period to initiate a load balancing operation.

3.2.1.3 Resource Type Policy

Resource policy deals with the specifying of a resource as a server node or receiver of workloads according to its availability status.

3.3.1.4 Location Policy

Location policy uses the results of the resource type policy to find an appropriate partner for a server node or a receiver node. Location policy is mainly responsible for selecting the best server node among all the available nodes in the system. Resource availability and service availability are some of the factors which need to be considered while selecting a server node for work load execution.

3.2.1.5 Selection Policy

Selection policy deals with the selection of a task to be transferred. This policy defines the tasks that are to be transferred from overloaded nodes to most idle nodes. While transferring a task a basic criterion must be satisfied. This criterion is that the overhead incurred in the transfer of the task should be compensated by response time reduction.

3.2.2 Centralized and Distributed Based Dynamic Load Balancing Techniques

This section describes the different dynamic load balancing techniques based on the location of decision making, the information used for the decision making process, scalability factor, and the overhead of exchanging the profile information [10].

3.2.2.1 Centralized Dynamic Load Balancing Technique

In this load balancing technique, a master node will have the responsibility of making the load balancing decision and the information used for the load balancing is obtained from the remaining slave nodes on either on demand basis or after a predefined fixed time interval. The load information may also be gathered only when there is any change in the system's current state. The advantage of this load balancing technique is that, since the load information is not send on arbitrarily, the inter process communication is reduced thereby avoiding network overhead. But, on the other hand, this technique has limited scalability feature.

3.2.2.2 Distributed Non-cooperative Dynamic Load Balancing Technique

In this load balancing technique, the responsibility of distributing the work load is shared among all the working server nodes instead of using a master node. The current work load information is collected based on the on demand criteria. In case if any of the server node changes its current working state to overloaded state, then that specific node must distribute its current load information to all other nodes so that the current work load can be redistributed in order to balance the entire system load efficiently. This method provides moderate scalability as compare to the centralized method. But in case if any working node is overloaded then, since that overloaded node has to distribute its current load information to all other working nodes before rescheduling the system load, it may increase the network traffic due to inter process communication.

3.2.3 Queue Based Dynamic Load Balancing Technique

In the queue based algorithms, predominantly queues are maintained in the load balancer system in order to store the incoming workloads. In this section, we have discussed five different algorithms which mainly use queuing mechanism for balancing the workloads among available server nodes.

3.2.3.1 Central Queue Algorithm

In this Central Queue Algorithm [16], the new work load requests and the unfulfilled requests are stored in a cyclic FIFO queue on the main node. Each new work load request which arrives at the queue manager is inserted into the queue. Then, whenever a request for work load is received

by the queue manager, it removes the first work load from the queue and sends it to the requesting node. If there are no available workloads in the queue then, the request is buffered, until a new work load is available. If a new work load arrives at the queue manager and at the same instance, there are un-serviced requests in the queue then, first such request is removed from the queue and the new work load is allocated to it.

In case if any of the working node reaches to under loaded state then, the local load manager sends a request for a new work load to the central load manager. The central load manager responds the request immediately if a work load is available in the queue or queues the request until a new work load arrives.

3.2.3.2 Local Queue Algorithm

This Local Queue algorithm [16], is featured with the support of dynamic process migration. The basic functionality of this algorithm is static assignment of all new processes with process migration initiated by a host when its load falls under threshold limit. This is a user-defined parameter of the algorithm. This parameter defines the minimal number of ready processes the load manager tries to allocate on each processor.

At the initial stage, the new processes which are created on the main host are assigned on all under loaded hosts. The number of parallel activities created by the first parallel construct on the main host is usually sufficient for allocation on all remote hosts.

From there after all the processes created on the main host and all other hosts are allocated locally. When the host gets under loaded, the local load manager tries to get several processes from remote hosts. It randomly picks up the local ready processes and sends requests with the number of local ready processes to remote load managers. When such requests are received by the load manager, it compares the local number of ready processes with that of the received number. If the local number is greater than the received number then, few running processes are shifted to the requester and a confirmation is sent with the number of processes transferred.

3.2.3.3 Queue Adjustment Policy (QAP)

In this Queue Adjustment Policy method [13], the scheduler which is the load manager is placed immediately after the queue. The algorithms which are following this policy attempts to balance the workloads in the queues of the nodes. When a work load arrives at node *i*, if the queue is empty, then the work load will be assigned to processor directly. If queue is not empty then, the work load is made to wait in the queue. The scheduler of node *i* periodically checks the queue lengths of other nodes that node *i* is interested. When there is any imbalance in the queue lengths *i.e.* few queues may be too long and few may be too short, the scheduler will make a decision on how many workloads in the queue should be transferred and where each of the workloads should be sent to. By doing the queue adjustment in this way, the load manager or the scheduler could balance the entire load in the system.

3.2.3.4 Rate Adjustment Policy (RAP)

In this Rate Adjustment Policy method [13], the scheduler is immediately placed before the queue. When a work load arrives at node *i*, the scheduler makes a decision on where the work load should be sent, whether it needs to be sent to the queue of node *i* or to other nodes in the system. Once the work load enters the queue, it will be processed by the respective assigned node and will not be transferred to other nodes in the system.

3.2.3.5 Hybrid Policy: Combination of Queue and Rate Adjustment Policy

In this Hybrid Policy method [13], the scheduler is allowed to adjust the incoming work load rate and also allowed to adjust the queue size of node i in some conditions. In some cases, when we use Rate Adjustment Policy method, the queue size may exceed a predefined threshold value due to which load imbalance may happen. When such situation happens, Queue Adjustment Policy method starts to work and guarantees that the workloads in the queues are balanced in the entire system. In this method, the rate adjustment can be considered as a coarse adjustment and the queue adjustment can be considered as a fine adjustment.

3.2.4 Primary and Centralized Node Based Dynamic Load Balancing Techniques

3.2.4.1 Primary Node Based Load Balancing Technique

In this method [12], either at the initial stage, processes can be stored in queue or the processes can be allocated to the nodes as they arrive. If processes are placed in queue then, each process is allocated one by one to the primary nodes of the load balancing system. During the course of time, in case if load imbalance happens in the system then, processes are migrated from heavily loaded nodes to lightly loaded nodes. Process migration makes a great impact on the network bandwidth and work load.

In order to reduce the network traffic during process migration, nodes are grouped into clusters. At first, a lightly loaded node is checked in the same cluster, if a node is found then the process is transferred to the node which was found within the same cluster. If suitable node is not found within the same cluster then, it searches the nearby cluster and once a suitable node is found, the process transfer takes place.

3.2.4.2 Centralized Node Based Load Balancing Technique

There are some situations where in, a heavily loaded node will not be able to find the lightly loaded node in its own cluster and due to network traffic; the node fails to search a node which would present in a remote cluster. It would be suitable if heavily loaded node finds a temporary node in its own cluster to handle the over load.

So, in this Centralized approach [12], for every cluster a centralized node is provided and this node is not assigned with any work load initially. Whenever a primary node is over loaded, it first searches for the other lightly loaded primary nodes in its own cluster; if such primary node is available, the work load is transferred to the found node and the load is balanced in the system. If no other lightly loaded primary node is found within the same cluster then, the work load is assigned to the centralized node which is present within the same cluster. A Centralized node will have some better configuration structure when compared to other nodes in the cluster. The network traffic between centralized node and all other primary nodes are kept minimum in order to avoid network delays. Due to this reason, any overloaded node can easily reach the centralized node in case if it is heavily loaded and transfer the load easily.

4. COMPARATIVE MEASURES

The performance of various load balancing techniques is measured by the following parameters [8] [14].

4.1 Overhead Associated

This parameter is related with determining the amount of overhead involved while implementing a load-balancing technique. It is comprised of overhead due to movement of workloads, inter-processor communication and inter-process communication.

4.2 Fault Tolerant

This parameter is used to indicate whether the load-balancing technique has the ability to with stand to the faults or not. This feature makes an algorithm to continue operating properly even during the event of some failure.

4.3 Reliability

This parameter is used to indicate which algorithm is more reliable in case if some host failure is happened.

4.4 Stability

This parameter can be related to the exchange of present workload state information among processors. Stability can be featured in terms of the delays in the transfer of work load information between processors and the gains in the load balancing algorithm by achieving faster performance by a specified amount of time.

4.5 Centralized or Decentralized

This parameter is used to indicate whether a centralized node is used in the load-balancing technique or not. Centralized approaches store the global information within a designated centralized node. All other sender or receiver nodes access the designated node to determine the amount of workload-transfers and also to check what tasks are to be sent to or received from.

4.6 Adaptability

This parameter is used to indicate whether the load-balancing algorithm is adaptive to varying or changing situations *i.e.* situations which are of dynamic nature.

4.7 Cooperative

This parameter indicates whether the nodes share information between them while doing the process allocation decision. It indicates the extent of independence that each node has in deciding how it can utilize its own resources. In the situation of cooperativeness, all nodes will have the responsibility to carry out their own assigned job, but all nodes work together in a cooperative manner to achieve better efficiency in the load-balancing system. In the non-cooperative situation, individual nodes act independently and make decisions about the use of their resources without causing any effect on remaining nodes of the system.

4.8 Process Migration

This parameter indicates whether transfer of work load happens within the nodes of the load-balancing system or not. It provides information on when a system makes decision to transfer a work load. It determines whether a process could be processed locally or it needs to be handled by a remote node of the system.

4.9 Resource Utilization

This parameter indicates the resource utilization of the server node. Automatic load balancing can be achieved by Resource utilization. A distributed environment might have more number of processes that demand for more processing power. If the load-balancing algorithm is capable enough to utilize the available resources then, those resources can be shifted to under loaded processors more efficiently.

4.10 Response time

This parameter indicates the amount of time taken by the load-balancing technique to respond to the distribution of work load among the nodes of the system.

5. COMPARISON AMONG VARIOUS LOAD BALANCING TECHNIQUES

In this section, we discuss the comparison between various load balancing techniques starting with the high level comparison between static load balancing techniques and dynamic load balancing techniques. Our discussion continues further by comparing various static load balancing algorithms that we discussed in the previous section considering the various comparison parameters. Finally we provide the comparison metrics among various dynamic load balancing techniques which were discussed in earlier sections.

5.1 Static Vs Dynamic Techniques

Various comparisons between static load balancing techniques and dynamic load balancing techniques are listed in the table 2.

Table 2
Static Vs Dynamic

Parameters	*Static*	*Dynamic*
Reliability	Less	More
Complexity	Less	More
Performance	Less	More
Flexibility	More	Less
Stability	More	Less
Implementation	Easy	Difficult
Communication Overhead	Less	More
Resource Utilization	Less	More
Adaptability	Less	More
Response Time	Less	More

5.2 Comparison of Static Techniques

Various comparisons among static load balancing techniques are listed in the below table 3.

Table 3
Comparison of Static Load Balancing Algorithms

Parameters	*Random*	*Round-Robin*	*Weighted Round Robin*	*Central Manager*	*Threshold*
Fault Tolerant	No	No	No	Yes	No
Complexity	Less	Less	Less	More	More
Performance	Less	Less	Better	More	Good
Flexibility	More	More	More	Less	Less
Resource Utilization	Less	Less	More	Less	Less
Communication Overhead	No	No	No	More	Less
Cooperative	No	No	No	Yes	Yes
Centralized	No	No	No	Yes	No

5.3 Comparison of Dynamic Techniques

Various comparisons among dynamic load balancing techniques are listed in the below table.

Table 4
Comparison of Dynamic Load Balancing Techniques

Techniques	*Centralized*	*Communication Overhead*	*Scalability*	*Resource Utilization*	*Process Migration*
Centralized	Yes	Less	Less	Less	No
Distributed Non-cooperative	No	More	More	More	No
Central Queue-Based	Yes	Less	Less	Less	No
Local Queue-Based	No	Less	More	More	Yes
Queue-Based adjustment	Yes	Less	Less	More	Yes
Rate-Based adjustment	Yes	Less	Less	More	No
Hybrid	Yes	Less	Less	More	Yes
Primary approach	No	More	More	More	Yes
Centralized approach	No	Less	More	More	Yes

6. CONCLUSION

In this paper, we started the discussion from the very basic features of client-server architecture and load balancer characteristics. Then we discussed about various techniques present in the static load balancing and dynamic load balancing. We discussed the advantages and

disadvantages of each of the load balancing techniques. Also, we made comparative study on various parameters among the static load balancing techniques and dynamic load balancing techniques.

Even though every load-balancing technique has its own merits & demerits, the selection of a best load balancing technique is purely based on the type of application which looks forward for its smooth processing of the work loads.

For example, for the web related applications, a simple round robin technique would be sufficient to handle the download requests for the static web pages. Where as in dynamic applications for example in case of real time interactive applications, where the requests take longer than the others to process, advanced load balancing algorithms such as dynamic load balancing techniques could be used.

Whatever might be the load balancing technique we choose for an application, the chosen technique should be an efficient technique to handle the work load of the application in an efficient way. So in order to make a load balancing system efficient, the primary load balancer module should be featured with the capability of providing intellectual monitoring service in order to monitor the current workloads on the group of servers and also the load balancer module must be capable enough to distribute the workloads to the best available servers that are capable of handling the workloads better than the others in the group of servers.

REFERENCES

[1] Network Load Balancing in Microsoft Windows 2000 Advanced Server and Datacenter Server Operating Systems, http://technet.microsoft.com/en-us/library/bb742455.aspx.

[2] Ramana Kumar K., Mahesh V. Ghatage, "Load Balancing of Services with Server Initiated Connections", ICPWC, 2005.

[3] Yixin Diao, Chai Wah Wu, Joseph L. Hellerstein, Adam J. Storm, Maheswaran Surendra, Sam Lightstone, Sujay Parekh, Christian Garcia-Arellano, Matthew Carroll, Lee Chu, and Jerome Colaco, "Comparative Studies of Load Balancing With Control and Optimization Techniques", *2005 American Control Conference Portland*, OR, USA, June 8-10, 2005.

[4] Tony Bourke, "Server Load Balancing", O'ReiIly Publicationsm, 2001.

[5] Branko Radojevic, Mario Žagar, "Analysis of Issues with Load Balancing Algorithms in Hosted Cloud Environments", Opatija, Croatia, MIPRO 2011, May 23-27, 2011.

[6] Hisao Kameda, El-Zoghdy Said Fathy, Inhwan Ryu, Jie Li, "A Performance Comparison of Dynamic vs. Static Load Balancing Policies in a Mainframe - Personal Computer Network Model", *Proceedings of IEEE Conference on Dedsion and Control Sydney*, Australia December, 2000.

[7] J. Li, C. Kim and Y. Zhang, "Optimal Load Balancing in Distributed Computer Systems", Springer, 1997.

[8] Zeng Zeng and Bharadwaj Veeravalli, "Rate-Based and Queue-Based Dynamic Load Balancing Algorithms in Distributed Systems", ICPADS'04 .

[9] H. Kameda, J. Li, C. Kim and Y. Zhang, "Optimal Load Balancing in Distributed Computer Systems", Springer, 1997.

[10] Ardhendu Mandal and Subhas Chandra Pal, "An Empirical Study and Analysis of the Dynamic Load Balancing Techniques Used in Parallel Computing Systems", ICCS 2010.

[11] William Leinberger, George Karypis, Vipin Kumar, "Load Balancing Across Near-Homogeneous Multi-Resource Servers", *IEEE*, 2000.

[12] Parveen Jain and Daya Gupta, "An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Nodes by Exploiting the Interrupt Service", *International Journal of Recent Trends in Engineering*, **1(1)**, May 2009.

[13] P.L. McEntire, J.G. O'Reilly, and R.E. Larson, "Distributed Computing: Concepts and Implementations". New York: IEEE Press, 1984.