IOSR Journal of Engineering (IOSRJEN)
e-ISSN: 2250-3021, **p**-ISSN: 2278-8719
Vol. 3, Issue 3 (Mar. 2013), ||V2 || PP 45-50

Comparative Study of Load Balancing Algorithms

Jyoti Vashistha[1], Anant Kumar Jayswal[2]
[1](Amity School of Engineering & Technology Noida India)
[2](Amity School of Engineering & Technology Noida India)

Abstract: In today's world, more or less every activity belong to internet. The increase of E-Commerce has leads many business activity to carry out most of their day to day business online transaction on data such as financial transaction, database access, corporate internet and other key function must run 24 hours a day ,seven days a week and network need to ability to scale the performance to handle the large volume of client request without creating unwanted delays. For availability and scalability, performances boosting more and more servers are required. Load balancing is a key issue in these type of large scale situation. Load balancing is to achieve optimal resources, maximize throughput, minimize response time and avoid the overload. Load balancing ensures that all the processor in the system or every node in the network does approximately the equal amount of work at any instant of time. The objective of this paper firstly, to compare the static load balancing and dynamic load balancing algorithm by parameter performance and secondly, to compare the dispatcher based algorithms through a simulation to evaluate their performance under different conditions and workloads

Keywords: *Static Load balancing, Dynamic Load Balancing, Dispatcher Based Load Balancing, Parametric Performance*

I. INTRODUCTION

Network load balancing is a cluster technology in Microsoft windows 2000 advanced server and datacenter server operating system, enhance the scalability and availability of mission-critical TCP/IP-based services such as Web terminal services ,virtual private networking and streaming media servers. Network load balancing distributes the IP traffic to multiple copies of a TCP/IP services such as Web server each running on the host within the cluster. Network load balancing transparently partitions the client request among the hosts and lets the client access the cluster using one or more virtual IP address. Load Balancer is usually a software program that is listening on the port where external clients connect to the access services. The load balancer forward the request to one of the backend servers, which usually replies to the load balancer which may have security benefits by hiding the structure of the internal network and preventing attack on the kernel's network stack or unrelated services running on the other ports. If you are load balancing across several severs And one of the server fail, your service will still be available to your users as the traffic will be delivered to the other server in your server farm There are two important features that used in load balancing in web server system and provide the cluster communication among themselves.

A. Scalability

Network load balancing scales the performance of the server-based program such as the webserver, by distributing the client requests across the multiple server within the cluster. As traffic increase, additional servers can be added to the cluster.

B. High availability

Network load balancing provide the high availability by automatically detecting the failure of a server and re-partitioning the client traffic among the remaining servers within ten seconds, while providing users with continuous service

.

II. TYPES OF SCHEDULING

A. Local Scheduling

Local scheduling is performed by the operating system for the uniprocessor

B. Global Scheduling

Global scheduling decides where to execute a process in a multiprocessor system. Global scheduling may be carried out by a single central or master processor, or it may be distributed among the processors. Global scheduling is further classified into static and dynamic scheduling.

III. STATIC LOAD BALANCING

In static load balancing[1][2],no dynamic information is used and the performance of the processor is determined at the beginning of the execution. Depending on their performance such as arrival time execuction time ,amount of resources needed ,the workload is distributed in the start by master processor. The slave processor calculated their allocated work and submit their result to the master .The goal of SLB method is to reduce the overall execution time of a concurrent program and minimizing the communication delays. The SLB algorithm are Round robin Algorithm, Randomized Algorithm, Central Manager Algorithm.

A. Round Robin Algorithm

Round Robin algorithm [1] 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 interprocess communication. In general Round Robin is not expected to achieve good performance in general case.

B. Randomized Algorithm

Randomized algorithm [1] 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.

C. Central Manager Algorithm

Central Manager Algorithm [3], 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.

D. Threshold Algorithm

In Threshold algorithm [3], 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 thesystem'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 underloaded. When the load state of a processor exceeds a load level limit, thenit sends messages regarding the new load state to all remote processors, regularly updating them as to the actual load state of the entire system.

IV. DYNAMIC LOAD BALANCING

In dynamic load balancing the work load is distributed among the processor at the run time In which the master assign the new processor to the slave based on new information collected [4][5], Unlike the static algorithm dynamic algorithm bufferes the process in the queue on the main node and allocated dynamically upon request from remote nodes. As a result, dynamic load balancing algorithm can provide a significant improvement in performance over static algorithm. However ,this comes at the additional cost of collecting and maintaining load information, So it is important to keep this overheads with reasonable time. Dynamic algorithm can be classified into different categories:.

A. Sender Initiative

In this type the load balancing algorithm is initialized by the sender. In this type of algorithm the sender sends request messages till it finds a receiver that can accept the load.

B. Receiver Initiative

In this type the load balancing algorithm is initiated by the receiver. In this type of description algorithms the receiver sends request messages till it finds a sender that can get the load.

C. Symmetric and Periodically Exchanged

It is the combination of both sender initiated and receiver initiated .The simple way to decentralized exchange load information periodically is by every node send its load level to all other nodes periodically

D. Central Queue Algorithm

Central Queue Algorithm [6] 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. 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. If there are no ready activities in the queue, the request is buffered, until a new activity is available. If a new activity arrives at the queue manager while there are unanswered requests in the queue, the first such request is removed from the queue and the new activity is assigned to it. When a processor load falls under the threshold, the local load manager sends a request for a new activity to the central load manager. The central load manager answers the request immediately if a ready activity is found in the process-request queue, or queues the request until a new activity arrives.

E. Local Queue Algorithm

Main feature of this algorithm [6] is dynamic process .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. It randomly sends requests with the number of local ready processes to remote load managers. When a load manager receives such a request, it compares the local number of ready processes with the received number. If the former is greater than the latter, then some of the running processes are transferred to the requester and an affirmative confirmation with the number of processes transferred is returned.

V. PARAMETRIC PERFORMANCE

A. Overload Rejection

If Load Balancing is not possible additional overload rejection measures are needed. When the overload situation ends then first the overload rejection measures are stopped. After a short guard period Load Balancing is also closed down.

B. Fault Tolerant

This parameter gives that algorithm is able to tolerate tortuous faults or not. It enables an algorithm to continue operating properly in the event of some failure. If the performance of algorithm decreases, the decrease is proportional to the seriousness of the failure, even a small failure can cause total failure in load balancing.

C. Forecasting Accuracy

Forecasting is the degree of conformity of calculated results to its actual value that will be generated after execution. The static algorithms provide more accuracy than of dynamic algorithms as in former most assumptions are made during compile time and in later this is done during execution.

D. Stability

Stability can be characterized in terms of the delays in the transfer of information between processors and the gains in the load balancing algorithm by obtaining faster performance by a specified amount of time.

E. Process Migration

Process migration parameter provides when does a system decide to export a process? It decides whether to create it locally or create it on a remote processing element. The algorithm is capable to decide that it should make changes of load distribution during execution of process or not.

F. Recourses Utilization

Resource utilization include automatic load balancing A distributed system may have unexpected number of processes that demand more processing power. If the algorithm is capable to utilize resources, they can be moved to under loaded processors more efficiently

G. Throughput

Throughput is the amount of data moved successful from one place to another from a given period of time.

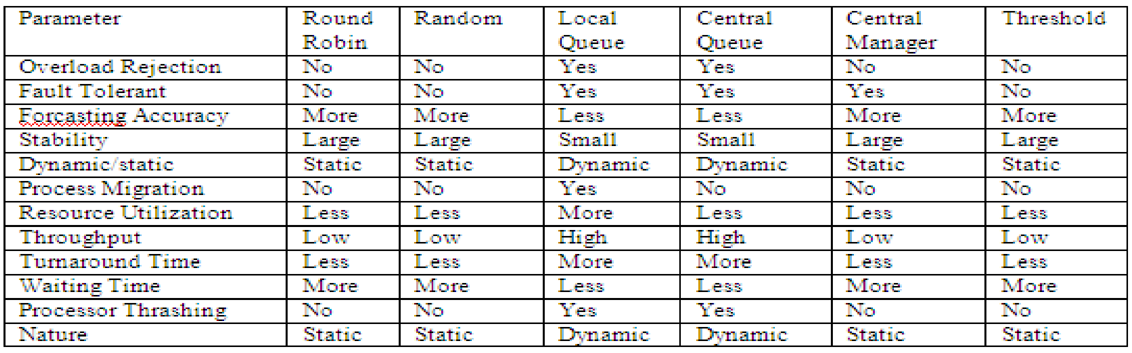
H. Turnaround Time
The **t**urnaround time is the time of submission of process to completion of process is the turnaround time.

I. Waiting Time
Waiting time is the sum of period spent waiting in ready queue is the waiting queue.

J. Processor Thrashing
Processor thrashing occurs when most of the processors of the system are spending most of their time migrating processes without accomplishing any useful work in an attempt to properly schedule the processes for better performance. Static load balancing algorithms are free from Processor thrashing as no relocation of tasks place. Dynamic load balancing algorithms incurs substantial processor thrashing.

K. Nature of Load Balancing Algorithm
This factor is related with determining the nature or behavior of load balancing algorithms that is whether the load balancing algorithm is of static or dynamic nature, pre-planned or no planning..

Parametric Comparison of Load Balancing Algorithm



Parameter	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
Forcasting Accuracy	More	More	Less	Less	More	More
Stability	Large	Large	Small	Small	Large	Large
Dynamic/static	Static	Static	Dynamic	Dynamic	Static	Static
Process Migration	No	No	Yes	No	No	No
Resource Utilization	Less	Less	More	Less	Less	Less
Throughput	Low	Low	High	High	Low	Low
Turnaround Time	Less	Less	More	More	Less	Less
Waiting Time	More	More	Less	Less	More	More
Processor Thrashing	No	No	Yes	Yes	No	No
Nature	Static	Static	Dynamic	Dynamic	Static	Static

VI. PERFORMANCE COMPARISON OF DISPATCHER BASED ALGORITHM

A. Random Scheduling Algorithm
In random scheduling, an incoming request is sent to a randomly selected host. This policy equalizes the expected number of tasks at each host [7]. The algorithm is very fast and over a period of time it ensures that the requests are fairly distributed. However, if the numbers of requests are small, the probability of imbalance will be very high.

```
RECEIVE_REQUEST
WEB_SERVER_ID=RANDOM(1TO N)FORWARD_REQUEST_TO(WEB_SERVER_ID).
```

B. .Round Robin Scheduling Algorithm
The round-robin scheduling algorithm sends each incoming request to the next server in it's list. Thus in a three server cluster (servers A, B and C) request 1 would go to server A, request 2 would go to server B, request 3 would go to server C, and request 4 would go to server A, thus completing the cycling or 'round-robin' of servers. It treats all real servers as equals regardless of the number of incoming connections or response time each server is experiencing.The scheduling granularity of Virtual Server is network connection-based, and it is much superior to round-robin DNS due to the fine scheduling granularity.

```
RECEIVE_REQUEST
WEB_SERVER_ID= (LAST_SERVER_ID + 1) MOD N
 FORWARD_REQUEST_TO(WEB_SERVER_ID)
```

C. Weighted Round Robin
The weighted round-robin scheduling is designed to better handle servers with different processing capacities. Each server can be assigned a weight (an integer value) that indicates the processing capacity of the server. In the implementation of the weighted round-robin scheduling, a scheduling sequence will be generated according to the server weights after the rules of virtual server are modified. Actually, the round-robin scheduling is a special instance of the weighted round-robin scheduling, in which all the weights are equal [8].

D. Least Connection Scheduling Algorithm

The least-connection scheduling algorithm [9] directs network connections to the server with the least number of established connections. This is one of the dynamic scheduling algorithms; because it needs to count live connections for each server dynamically. For a virtual server that is managing a collection of servers with similar performance, least-connection scheduling is good to smooth distribution when the load of requests vary a lot.

```
RECEIVE_REQUEST WEB_SERVER_ID = 0
FOR I = 1 TO
IF (SERVER(I).CONNECTIONS <
SERVER(WEB_SERVER_ID).CONNECTIONS)   WEB_SERVER_ID = I
FORWARD_REQUEST_TO(WEB_SERVER_ID)
```

E. Least Loader Scheduler Algorithm

The least loaded server policy is somewhat similar to the least connection scheduling. It is also a dynamic policy and requires calculating the size of the load on each server at the time of forwarding a request. As the least loaded server policy shares the request based on the actual physical load on the server, it should give an optimum load balancing [10].

```
RECEIVE_REQUEST WEB_SERVER_ID = 0
FOR I = 1 TO N
IF (SERVER(I).LOAD <
SERVER(WEB_SERVER_ID).LOAD)
WEB_SERVER_ID=FORWARD_REQUEST_TO(WEB_SERVER_ID)
```

VII. SIMULATION OF DISPATCHER BASED ALGORITHM

A. Effect of Number of Server

To study the effect of the number of servers on the performance of the algorithms, we kept all the parameters constant and we varied the number of servers in the counts 2, 4. 7,10,15 and 20. The simulation results show that as we increase the number of servers, the overall imbalance on the system decreases. This is mainly due to the fact that as we increase the number of servers, the load on the servers also decreases The results show that the least loaded sever algorithm gives us the best balance and the random is the worst. However, we notice that sometimes Random algorithm tends to perform better than the round robin. This is specially noticed when exceptionally large requests arrive on the system. The results indicate that round robin performs well if the incoming requests are of similar size but if the file sizes differ significantly, it causes the system to imbalance. The least connection, being a dynamic policy tends to adapt itself to the imbalance but as it is not aware of the request size, it does not perform equally as good as the least loaded server algorithm. Figure 1 shows the effect of changing the number of servers on the average latency for the different algorithms. It is clear that the average latency improves with the increasing number of servers. This change is very clear when changing from 2 to 10 servers but there is no significant improvement in the average latency when changing from 10 to 20 servers. This is due to the fact that the web servers are no longer overloaded and all systems have similar performance.

B. Effect of Arrival Rate

Figure 2 shows the effect of the arrival rate on the average latency of the algorithms. We see that the average latency increases with the increase in the arrival rate. We also notice that the average latency of the different algorithms is similar with slight variation when the number of requests is about100 requests/sec.

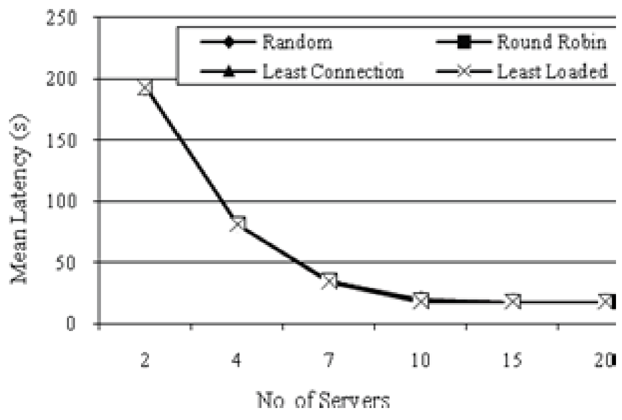


Figure 1 Average latency of Different Algorithm At Different No of Server

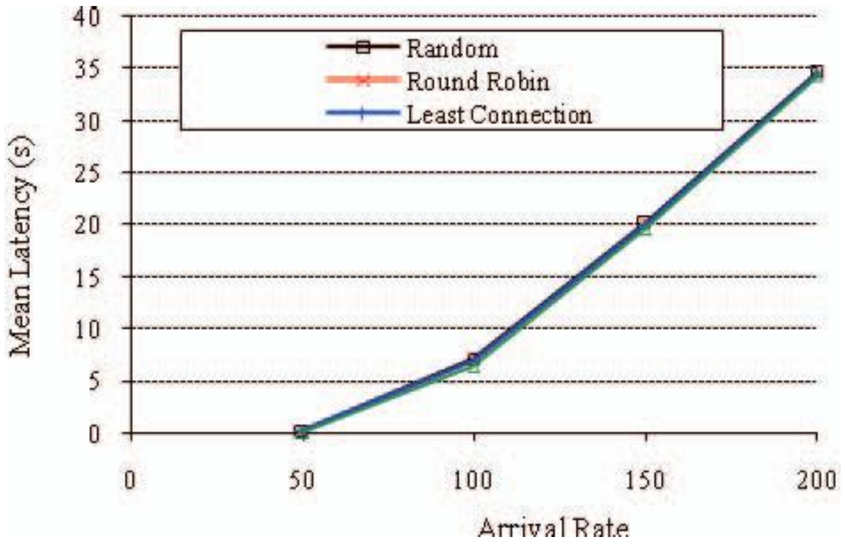


Figure 2. Average Latency of Different Algorithms at different arrival rate

VIII. CONCLUSION

Load balancing algorithms work on the principle that in which situation workload is assigned, during compile time or at runtime. The above comparison shows that static load balancing algorithms are more stable in compare to dynamic and it is also ease to predict the behavior of static, but at a same time dynamic distributed algorithms are always considered better than static algorithms. The dispatcher based approach is a centralized scheduling method in which the dispatcher has the overall control over the balancing process and achieves a fine grained balancing. We simulated four different dispatcher-based scheduling algorithms: random, round robin, least connection scheduling and the least loaded server in our work. The random algorithm performs the worst in some of the conditions but at others it seemed to work well. The random algorithm gives unpredictable results. Among the round robin and the least connection scheduling, the least connection scheduling algorithm functions well

REFERENCES

[1]. Wang, Y., Morris, R.: Load Balancing in Distributed Systems. In: IEEE Transactions on Computing, C-34, pp. 204-217, (1985)

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

[3]. 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.

[4]. Mahk, S.: Dynamic Load Balancing in a Network of Workstation. Research Report, (2000).

[5]. Chow, Y.C., Kohler, W.: Models for Dynamic Load Balancing in a Heterogeneous Multiprocessor System. In IEEE Transactions on Computers, Vol. C-28, pp. 334 – 361, (1979)

[6]. William Leinberger, George Karypis, Vipin Kumar, "Load Balancin Across Near-Homogeneous Multi-Resource Servers", 0-7695-0556- 2/00, 2000 IEEE.

[7]. B. Haakon, E. Kloving and Ø. Kure. "A Comparison of Load Balancing Techniques for Scalable Web Servers." IEEE Network, vol. 14, no. 4, pp. 58-64, 2000.

[8]. M. Harchol-Balter, M. E. Crovella and C. D. Murtaz. "On Choosing a Task Assignment Policy for a Distributed Web-Sever System." J. Parallel Distrib. Computing, vol. 59, no. 2, pp. 204-228, 1999.

[9]. Least Connection Scheduling, http://kb.linuxvirtualserver/wiki/leastconnection-scheduling.

[10]. K. H. Yeung, K. W. Suen and Y. K. Wong. "Least Load Dispatching Algorithm for Parallel Web Server Nodes." IEE Proceedings on Communications, vol. 149, no.4, pp. 223- 226, 2002.