International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 4 Issue 2, Feb.-2015, pp: (246-252), Impact Factor: 1.252, Available online at: www.erpublications.com

Load balancing in Computer using FCFS algorithm

Kapil Sharma[1], Lokesh Kumar[2]

[1]M. Tech Scholar, RIEM, Rohtak, Haryana
[2]Asst Prof., RIEM, Rohtak, Haryana

ABSTRACT

Now a day's, everything is going online. Web applications plays vital role on internet to provide 24*7 services to customers. When application becomes popular, traffic to that is also in a state of growing. For availability, scalability, performances boosting more and more servers are required. Load balancing is a key issue in these types of large scale situation. Load balancing is to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload. In many scenarios, however, it's not simply a matter of distributing traffic equally across many servers once deploys; other factors also comes into play. Here is some brief introduction to different load balancing strategies, algorithms, methods. By investigating the comparative behavior of load balancing with different parameters, dynamic load balancing is more reliable. So; here we are discussing the implementation of application load balancer by using Dynamic load balancing method. Actually we perform load balancing by integrating more than two physical servers with logical load balancing.

Keywords: Dynamic Load Balancing Library, Load Monitoring Server (LMS), Load Reporting Server (LRS), Remoting, Quality of service (QoS).

I. INTRO DUCTION

In today's world, more or less, every activity belongs to internet. The increase of E-Commerce has leads many businesses to carry out the most of their day-to-day business online transaction on data. As a result of the popularity of the web, providers of web sites and storage space providers want to ensure the availability of access to information for their users and the guarantee that requests are processed as quickly as possible. If a server gets more requests than it can handle, this can be combated by using multiple le hosts to provide the same service. A web farm of multiple physical hosts grouped together will share the total load of the client requests, thus reduce the response time and increase the QoS, ultimately resulting in satisfied customers. Another important issue amongst service providers is their degree of uptime of the servers. This is also refereed as server availability, which in marketing documents are given as a certain number nines. Availability of 99.99 per cent is referred to as an availability of four nines. This means that the server should only be down .1 per cent of the time, which over duration of one year contributes to about 52 minutes of unavailability. Another benefit of having a web farm is redundancy, which helps to achieve both high availability as well as high performance. It is possible to perform upgrades on a host without shutting down the total service. we are specifically giving priority to Dynamic load balancing method rather than Static load balancing. Further we have discussed different balancing algorithms, techniques and their drawbacks. By taking advantage of Dynamic load balancing method, we have deployed one load balancing solution.

I. BRIEF INTRODUCTION TO LOAD BALANCER

One author described Load balancing as "In a distributed network of computing hosts, the performance of the system can depend crucially on dividing up work effectively across the participating nodes". Load balancer is usually a software program that is listening on the port where external clients connect to access services. The load balancer forwards requests 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 attacks on the kernel's network stack or unrelated services running on other ports. If you are load balancing across several servers and one of the servers fails, your service will still be available to your users, as the traffic will be diverted to the other servers in your server farm.

A. NETWORK LOAD BALANCING:

Network load balancing is the distribution of traffic based on network variables, such as IP address and destination ports. It is layer 4 (TCP) and below and is not designed to take into consideration anything at the application layer such as content type, cookie data, custom headers, user location, or the application behavior. It is context-less, caring only about the network-layer information contained within the packets it is directing this way and that.

B. APPLICATION LOAD BALANCING:

Application load balancing is the distribution of requests based on multiple variables, from the network layer to the application layer. It is context-aware and can direct requests based on any single variable as easily as it can a combination of variables. Applications are load balanced based on their peculiar behavior and not solely on server (operating system or virtualization layer) information. Application of load balancing is used by different companies that conduct huge business transactions using internet. It is normally done to ensure that all their clients are able to conduct business activities with ease and accuracy. The difference between the two is important because network load balancing cannot assure availability of the application. This is because it bases its decisions solely on network and TCP-layer variables and has no awareness of the application at all. Generally a network load balancer will determine "availability" based on the ability of a server to respond to ICMP ping, or to correctly complete the three-way TCP handshake. There are three major parameters which usually define the strategy; a specific load balancing algorithm which answer three important questions [6]:

1) who makes the load balancing decision,
2) what information is used to make the load balancing decision, and
3) Where, the load balancing decision is made.

A. Sender-Initiated vs. Receiver-Initiated Strategies:

The question of who makes the load balancing decision is answered based on whether a sender-initiated or receiver - initiated policy is employed. In sender-initiated policies, congested nodes attempt to move work to lightly-loaded nodes.
NLS= No Load Sharing SI= sender RI=Receiver

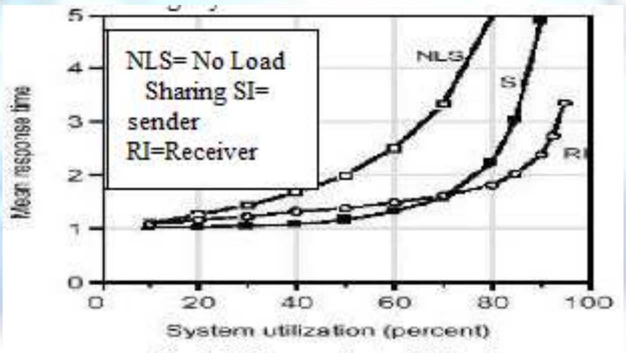

Fig 1: Performance Sender VS Receiver

The sender-initiated policy performing better than the receiver-initiated policy at low to moderate system loads; because the probability of finding a lightly-loaded node is higher than that of finding a heavily-loaded node. In receiver-initiated policies, lightly-loaded nodes look for heavily-loaded nodes from which work may be received. At high system loads, the receiver-initiated policy performs better since it is much easier to find a heavily-loaded node [9].

B. GLOBAL VS. LOCAL STRATEGIES:

Global or local policies answer the question of what in format ion will be used to make a load balancing decision. The choice of a global or local policy depends on the behavior an application will exhibit. In local policies workstations are partitioned into different groups. In a heterogeneous network station, the partitioning is usually done such that each group has nearly equal aggregate computational power. The benefit in a local scheme is that performance profile information is only exchanged within the group. For global schemes, balanced load convergence is faster compared to a local scheme since all workstations are considered at the same time. However, this requires additional communication and synchronization between the various workstations. But the reduced synchronization between workstations is also a downfall of the local schemes if the various groups exhibit it major differences in performance [9].

C. CENTRALIZED VS. DISTRIBUTED STRATEGIES:
A load balancer is categorized as either centralized or distributed, both of which define where load balancing decisions are made. There are tradeoffs associated with choosing one location scheme over the other. In a centralized scheme, the load

balancer is located on one master workstation node and all decisions are made there. So, the reliance on one central point of balancing control could limit future scalability. Additionally, the central scheme also requires an "all-to-one" exchange of profile information from workstations to the balancer as well as a "one-to-all" exchange of distribution instructions from the balancer to the workstations. In a distributed scheme, the load balancer is replicated on all workstations. The distributed scheme helps solve the Scalability problems, but at the expense of an "all-to-all broadcast of profile in format ion between workstations. However, the distributed scheme avoids the "one-to-all "distribution exchanges since the distribution decisions are made on each workstation [9].

TABLE I: PROS AND CONS OF LOAD-BALANCING SCHEMA

Approach	Scheduling	Pros	Cons
Client based	Client Side	No server overhead	Limited applicability
	Distributed	LAN & WAN solution	Medium-coarse-grained balancing
DNS based	Web System Side	No bottleneck	Partial control
	Centralized	LAN & WAN solution	coarse-grained balancing
Dispatcher based	Web System Side	Fine-grained balancing	Dispatcher Bottleneck
	Centralized	Full control	LAN solution packet rewriting Overhead
Server based	Web system side	Distributed control	Latency time increase (HTTP)
	Distributed	Fine-grained balancing LAN and WAN solution	Packet-rewriting overhead

II. LOAD BALANCING ALGORITHMS

Load balancing algorithm directly influences the effect of balancing the server workloads. Its main task is to decide how to choose the next server and transfer a new connection request to it.

There are four basic steps that that algorithms should follow:
1) Monitoring server performance (load monitoring)
2) Exchanging this information between servers (Synchronization with load-balancer)
3) Calculating new distributions and making the balancing decision. (Rebalancing criteria)
4) Actual request serve (fulfilling demand)
The two major categories for load-balancing algorithm ms are
A. Static Load Balancing
B. Dynamic Load Balancing

A. STATIC LOAD BALANCING:

Static load balancing algorithms allocate the tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on an average load of our workstation cluster. The advantage in this sort of algorithm is the simplicity in terms of both implementation as well as overhead, since there is no need to constantly monitor the workstations for performance statistics. The static algorithm is easily carried into execution and takes less time, which doesn't refer to the states of the servers. However, static algorithms only work well when there is not much variation in the load on the workstations. Clearly, static load balancing algorithms aren’t well suited to a NOW environment, where loads may vary significantly at various times in the day, based on the issues discussed earlier.
Different Algorithms to Implement Static load balancer:

1. RANDOM SCHEDULING:

The Random algorithm is self-explanatory. Traffic is directed arbitrarily to any server in your farm. In a random Scheduling, the requests are assigned to any server picked randomly among the group of servers.
Pros: Random m Scheduling load balancing algorithm is simple le to implement.
Cons: It can lead to overloading of one server while under-utilization of others.

2. ROUND-ROBIN SCHEDULING:

Robin Scheduling Algorithm is that the IP sprayer assigns the requests to a list of the servers on a rotating basis. For the subsequent requests, the IP sprayer follows the circular order to redirect the request. Once a server is assigned a request, the server is moved to the end of the list. This keeps the servers equally assigned.
Pros: Better than random allocation because the requests are equally divided among the available le servers in an orderly fashion.
Cons: 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.

3. WEIGHTED ROUND-ROBIN SCHEDULING:

One can assign a weight to each server in the group so that if one server is capable of handling twice as much load as the other, the powerful server gets a weight of 2. In such cases, the IP sprayer will assign two requests to the powerful server for each request assigned to the weaker one.
Pros: Takes care of the capacity of the servers in the group.
Cons: Does not consider the advanced load balancing requirements such as processing times for each individual request.

B. DYNAMIC LOAD BALANCING:

Dynamic load balancing algorithms make changes to the distribution of work among workstations at run-time; they use current or recent load information when making distribution decisions. As a result, dynamic load balancing algorithms can provide a significant improvement in performance over static algorithms. However, this comes at the additional cost of collecting and maintaining load information, so it is important to keep these overheads within reasonable limits [6][8].
The dynamic algorithm is self-adaptive algorithm, which is better than static algorithm. Self-adaptive load balancing system mainly includes two processes: monitoring the load states of servers and assigning the request to the servers. The state supervision, which depends on the load information of each server in the cluster monitored and collected periodically by the front-end balancer, raises the effect of load balance by monitoring load variety, however, this will burden the workload of balancer which is the bottleneck of the cluster system.

III. IDEA TO IMPLEMENT LOAD BALANCING

While using system sometimes user feels that the machine is getting very slow, then launch the task manager and look at CPU utilization. If it is low, then memory is low, and disk must be trashing. Well works if user is around the machine and has one or two mach ines to monitor. When there are more machines, one couldn't monitor machines constantly and even if he somehow manages it but, you can't distribute their workload manually. So, you need load monitoring and load distributing features all together to enhance the whole assembly. For 24*7 running application online, performance of total assembly is more depends on how servers are performing. Idea is to monitor server performance by collecting parameter information of processor, Disk utilization ion, Memory health, User time etc.

User can monitor system performance by launching Task Manager and by looking at the CPU utilization in the Performance tab start monitoring the CPU utilization. Now notice the counter values and values stay almost constant. Now close Task Manager, run media player or any other application, wait about 5 seconds and start it again. A big peak in the CPU utilization should be noticed. In several seconds, may be the peak vanishes. Here if performance counters values are reported instantly one could think that our machine was extremely busy (almost 100%) at that moment. That's why rather than reporting instance values, several samples of the counter's values are collected and will report their average. The CPU utilization is not enough for a realistic calculation of the mach ine's workload; more than one counter should be monitor at a time such as disk utilization, memory usage, I/O etc. Now the machine load will be calculated as the sum of the weighted averages of all monitored performance counters. A counter is set to 0. All parameter values are collected in certain interval till counter value becomes 5. Then sum and average of parameter values are calculated. Depends on that calculated values, less loaded server or high performance system is being selected for load balancing and serving the coming request at best.

IV. BRIEF ARCHITECTURE AND ASSEMBLY

The load balancing comes in three parts: a server that reports the load of the machine on which it running; a server that collects such loads, no matter which machine they come from; and a library "Load Balancing Library" which asks the collecting server which is the least loaded or fastest machine. The server that reports the machine's load is called (LRS) "Machine Load Reporting Server" and the server that collects machine loads is called (LMS) "Machine Load Monitoring Server". These are logical servers i.e. modules in load balancing software which will deploy on physical servers. To understand how the objects interact with each other within an assembly and on different machines, following are some technical terms need to focus on.

1) DELEGATE

A secure type safe way to call a method of a class indirectly, using a reference to that method; A delegate can hold reference/s to one or more functions and invoke them as and when needed.

2) WORKER

Utility class, usually with just one method, which is started as a separate thread; the class holds the data (is the state), needed by the thread to do its job;

3) TCP

A connection-based protocol, that means a communication session between hosts is established before exchanging data. A host is any device on a TCP/IP network identified by a logical IP address. TCP provides reliable data delivery and ease of use. Specifically, TCP notifies the sender of packet delivery, guarantees that packets are delivered in the same order in which they were sent, retransmits lost packets, and ensures that data packets are not duplicated with end-to-end error detection and correction.

4) IP MULTICASTING

Technology which allows data to be sent from m one host and then replicated to many others without creating a copy. IP multicasting relies on a special group of addresses known as multicast addresses. It is this group address that names a given group. For example, if five machines all want to communicate with one another via IP multicast, they all join the same group address. Once they are joined, any data sent by one machine is replicated to every member of the group, including the machine that sent the data. A multicast IP address is a class D IP address in the range 224.0.0.0 through 239.255.255.255.

6) REMOTING

The process of communication between different operating system processes, regardless of whether they are on the same computer. The .NET remoting system is an architecture designed to simplify communication between objects living in different application domains, whether on the same computer or not, and between different contexts, whether in the same application do main or not Above all techno is used for development of load balancer. The Load Reporting Server on each machine designated for the purpose of the load balancing joins a multicasts group and sends messages containing the machine's load to the group's multicast IP address. Because all Load Reporting servers join the same group at startup, they all receive each machine load, so if you run both Load Reporting Server and Load Monitoring Server on all physical machines, they will know each other's load. Well, all Load Monitoring Servers store the machine loads so that they can quickly retrieve the least machine load at any time. So all machines now know which the fastest one is. Each Load Monitoring Server registers a special singleton object with the .NET Remoting runtime, so that Load Balancing Library can get an instance of that object, and ask it for the least loaded machine. Here it should choose one machine and will hand that load to the client application that needs the information to perform whatever load balancing activity is suitable.

For deployment, assembly may consist of two or more servers. Basically two servers are needed because you can't do load balancing on single server. Here we have used three servers for deployment of load balancer.
On server 1:

Load Monitoring Server for monitoring load of its own system, Load Reporting Server which reports load collected by Monitoring Server to Load Balancing Library, Load Balancing Library which accepts data sent by Load Reporting Server and does calculations to find least loaded or high performance machine.
On Server 2 & Server 3:

Load Monitoring Server for collecting machine performance data and Load Reporting Server for reporting that data to server. Here all servers are connected each other through LAN. Concept of server clustering is used to make a server cluster. Clusters are usually deployed to imp rove performance and availability over that of a single computer, while typically being much mo re cost-effective than single computers of comparable speed or availability. Clusters are usually deployed to improve performance and availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

V. BRIEF ASSEMBLY COLLABORATION WITH ALGORITHM

How the LRS reports the machine load to LMS? Load Monitoring Server starts two worker threads. One for collecting machine loads called Collector Worker and one for reporting the minimum load to interested clients called Reporter Worker. Here Collector Worker comes into action; it creates a UDP socket, binds it locally and adds it to a multicast group. Now here collector initializes the code. It then enters a thread loop, periodically polling the socket for arrived requests. When a request comes, the collector reads the incoming data, parses it and enters the load in the machine loads collection of the Load Monitoring Server class.

Load Reporting Server starts one thread for reporting the current machine load. Here Report ing Worker comes into action. The code is initializes and thread starts monitoring the performance counters, creates a UDP socket and makes it a member of the same multicast group that collector worker has joined. In its thread loop, the reporter waits a predefined amount of time, then gets the current machine load and sends it to the multicast group which is received by other members that have joined the multicast group.

Now here comes the Load Balancing Lib rary to find the machine with the least load. It is achieved by .NET Remoting technique. When Load Monitoring Server runs, it registers a class named Server Load Balancer with the Remoting runtime as a singleton. When a request to get the fastest machine comes (Get least machine Load: gets called) the singleton asks Machine Loads Collect ion to return its least load and then hands it to the client object that made the remoted call.

A. FAULT TOLERANCE SOLUTIONS "WHEN SERVER STOPS REPORTING LOAD":

In this load balancing solution machine (physical server) and its respective load is stored in sorted order so that first machine in list is the least loaded machine. Now take a scenario where three servers are clustered and exchanging data with each other. Suddenly server 2 stops informing the machine load to load balancer i.e. server 2 is down. It the least loaded machine among others for say 10 min. In between if request comes from the user then request is forwarded to server 2. But server 2 is down, so here comes a fault. Solution to this problem; after receiving data from server 2, Collector runs timer for.

B. EXECUTING AND BALANCING A WEB FARM:

One application is built just as simple .NET Web application, which uses LBL to perform a load balancing in a web farm. Application shows fastest machine name telling that next request is being processed by that machine. Every time when user hit link page is refreshes and again shows fastest machine name. Some classes are written to wrap from Load Balancing Library and implemented it as a singleton. Then the Session On Start method of the Global class is used to redirect every new session's first HTTP request to the most available machine. In the sample web page, a dynamically build URLs are used for further processing, replacing the local host again with the fastest machine.

CONCLUSION

The efficient load balancing can clearly provide major performance benefit. By having advanced load balancing models and algorithms that can dynamically adapt to situations on servers to which they actually forward traffic, most of the overloading problems caused by bad load balancing decisions can be avoided. From the experiments and considering rapid growth in web-farm, Dynamic algorithms are efficient than static algorithms. We have deployed the Load Balancing Application and tested, found performing good. Now we are planning to go for rigorous testing and comparing with other load balancing techniques. In Future work, we are planning to go for load distribution policies means transfer load from highly loaded machine to least loaded machine.

REFERENCES

[1] Li Wenzheng, Shi Hongyan: "Novel Algorithm for Load Balancing in Cluster Systems" Publidhe by IEEE Proceedings of the 2010-978-1-4244-6763-1/10.

[2] Remi Badonnel, Mark Burgess: " Dynamic Pull-Based Load Balancing for Autonomic Servers" Published by IEEE 978-1-4244-2066-7/08.

[3] Valeria cardellini, University of Rome Tor Vergata, Michele Colajanni, University of Modena, Philip S. Yu, IBM T.J. Watson Research Center: "Dynamic load balancing on web-server systems" Publidhe by IEEE Internet Computing 1089-7801/99.

[4] Sewook Wee, Huan Liu:" Client-side Load Balancer using Cloud" Published by SAC'10 March 22-26, 2010, Sierre, Switzerland Copyright 2010 ACM 978-1-60558-638-0/10/03.

[5] Der-Chiang Li, Fengming M. Chang: " An In–Out Combined Dynamic Weighted Round-Robin Method for Network Load Balancing" Published by Advance Access publication, The Computer Journal Vol. 50 No. 5, 2007.

[6] Satoru Ohta and Ryuichi Andou : "WWW Server Load Balancing Technique Based on Passive Performance Measurement" Published by IEEE 978-1-4244-3388-9/09.

[7] Jiubin Ju, Gaochao Xu, Kun Yang: "An Intelligent Dynamic Load Balancer for Workstation Clusters" Published by ACM.

[8] Shahzad Malik: " Dynamic Load Balancing in a Network of Workstations" from Research Report.