

A New Approach for Dynamic Load Balancing Algorithm

Harshal Khandre, Prof. Manasi Kulkarni

Abstract— Mission-critical applications must run 24x7, and networks need to be able to scale performance to handle large numbers of client requests without unwanted delays. A "server cluster" is a group of independent servers managed as a single system for achieving availability and scalability. It consists of two or more servers connected by a network, and cluster management software. The software provides services such as failure detection, recovery, load balancing, and manages the servers as a single system. Load balancing is a technique that allows the performance of a server-based program, such as a Web server, to be scaled by distributing its client requests across multiple servers within a cluster of computers. Load balancing is used to enhance scalability, which improves throughput while keeping response times low.

In this paper we have proposed a dynamic load balancing system in which servers in distributed system uses multicasting to communicate with each other and uses decentralized approach for load balancing.

Index Terms—ASP, Load Balancing, Machine Load

I. INTRODUCTION

1.1 Load Balancing

The process of distributing or redistributing the load of the system equally among all the nodes of the system is called as load balancing. Load balancing helps the system to use its resources effectively and improve the response time of the job.

1.2 The goals of load balancing are:

- To improve the performance of the system
- To maintain the availability of the system.
- To keep system stable.

1.3 Types of Load balancing algorithms

1] Load balancing algorithms are divided into three categories based on who initiated the process of load balancing.

-Client Initiated: If the load balancing algorithm is initialized by the client.

Harshal Khandre, Student at Computer Department, Veermata Jijabai Technological Institute, Mumbai, India, 8983113034.

Manasi Kulkarni, Prof at Computer Department, Veermata Jijabai Technological Institute, Mumbai, India,

-Server Initiated: If the load balancing algorithm is initiated by the server.

-Symmetric: It is the combination of both client initiated and server initiated

2] Depending on the current state of the system, load balancing algorithms can be divided into 2 categories:-

-Static: It doesn't take into account the current state of the system. Past knowledge of the system is needed.

-Dynamic: Decisions on load balancing are based on current state of the system. No past knowledge is needed. So it is better than static approach.

[i] Static Algorithms :-

Static algorithms distribute the traffic evenly between servers. By this approach the traffic on the servers will be disdained easily and as a result it will make the situation more improperly. In this algorithm, based on the past knowledge of performance, master server distributes the load among the slave servers. Static algorithms are non-preemptive.

[ii] Dynamic Algorithms:-

Dynamic algorithms distribute the workload among the nodes at run time. It assigns proper weights on servers and by searching in entire network a lightest server chosen to balance the traffic. However, selecting the lightest server requires real time communication with the networks, which will lead to extra traffic added on system.

1.4 Types of Dynamic Load Balancing Algorithms

1] **Centralized dynamic load balancing:** - Centralized dynamic load balancing takes smaller amount of messages to reach a decision

2] **Distributed dynamic load balancing:** - Each node needs to exchange status information with every other node in the system.

1.5 Policies or Strategies in dynamic load balancing

There are 4 policies:

-Transfer Policy: The policy which selects a job for transferring from a local node to a remote node is referred to as Transfer policy or Transfer strategy.

-Selection Policy: It specifies the processors involved in the load exchange.

-Location Policy: It selects a destination node for a transferring the task.

-Information Policy: It collects information about the nodes in the system.

II. PROPOSED SYSTEM

Before proceeding further we must understand the following two terms regarding our system

1) Machine Load -

The overall workload (utilization) of a machine - in our case, this is the sum of the weighted averages of all performance counters (monitored for load balancing);

2) Fastest machine -

The machine with the least current load.

We proposed Load balancing software which has three parts.

1] Machine Server- Machine server will calculate the load of the machine it is running on and will report it to Master server.

2] Master Server- This server will collect the load information from all machine servers.

3] Library- Library will ask master server to find out least loaded server i.e. fastest server.

The Machine Server on each machine joins a special multicasts group, and sends messages, containing the machine's load to the group's multicast IP address. Because all Master Servers join the same group at startup, they all receive each machine load, so if you run both Machine and Master servers on all machines, they will know each other's load.

All Master Servers store the machine loads in a special data structure, which lets them quickly retrieve the least machine load at any time. So all machines now know which the fastest one is. Each Master Server registers a special singleton object with the .NET Remoting runtime, so the Library can get an instance of that object, and ask it for the least loaded machine. The problem is that LBL cannot ask simultaneously all machines about this, so it should choose one machine (i.e. the machine, it is running on) and will hand that load to the client application that needs the information to perform whatever load balancing activity is suitable.

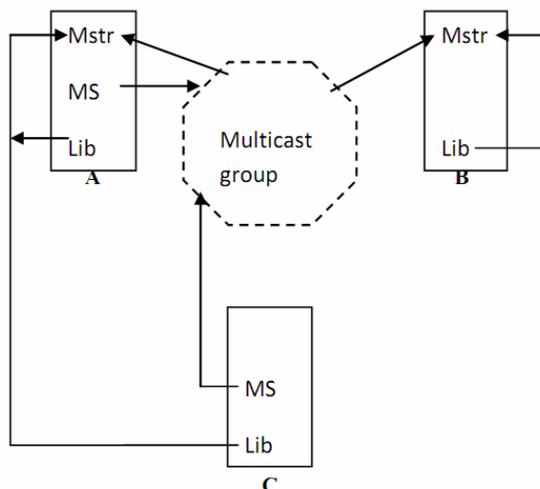


Fig 1: Proposed System architecture

Now look at the "figure". When a machine joins a multicast group, it receives all messages sent to that group, including the messages that the machine has sent. Machine A receives its own load, and the load, reported by C. Machine B receives the load of A and C (it does not report its load, because there's no MS(Machine server) server installed on it). Machine C does not receive anything, because it has not Mstr (Master server) installed. Because the machine C's Lib (library) should connect to an Mstr server, and it has no such server installed, it could connect to machine A or B and query the remote object for the fastest machine. On the "diagram" above, the Lib of A and C communicate with the remote object on machine A, while the Lib of B communicates with the remote object on its machine.

2.1 Metrics for load measurement

1) Process time –

It is the percentage of elapsed time that the processor spends to execute a non-idle thread. It is calculated by measuring the duration of the idle thread is active in the sample interval, and subtracting that time from interval duration. (Each processor has an idle thread that consumes cycles when no other threads are ready to run). This counter is the primary indicator of processor activity, and displays the average percentage of busy time observed during the sample interval. It is calculated by monitoring the time that the service is inactive and subtracting that value from 100%.

2) User Time –

It is the percentage of elapsed time the processor spends in the user mode. User mode is a restricted processing mode designed for applications, environment subsystems, and integral subsystems. The alternative, privileged mode is designed for operating system components and allows direct access to hardware and all memory. The operating system switches application threads to privileged mode to access operating system services. This counter displays the average busy time as a percentage of the sample time.

3) ASP Request time –

It is the number of requests currently executing.

4) Disk Time –

It is the percentage of elapsed time that the selected disk drive was busy servicing read or writes requests

2.2 The Proposed Load Balancing Algorithm:-

1] Machine Server calculate the load of the machine it is running on

$MachineLoad = \text{Sum}(\text{WeightedAverage}(\text{EachCounter}))$

2] The Machine Server on each machine joins a special multicasts group.

3] Master Server receives the load of all the other machines and store the machine loads in special Data Structure called as *priority_queue*. *priority_queue* is nothing but an Array List. Machine with the least *MachineLoad* will be store at the first position and will have higher priority.

4] Load Balancing Library will ask master server to find out least loaded server. Master Server will retrieve the first element from priority queue which is nothing but the Fastest Machine and forwards this information to Load balancing Library.

5] LBL checks if the Fastest Machine is the different Machine than the Machine it is running on, then LBL will forward the request to the URL of fastest Machine.

2.3 Characteristics of Proposed System.

1. Dynamic Load Balancing.
2. Decentralized Approach.
3. Communication between nodes using Multicasting.

REFERENCES

- [1] Ankita Singhal and Padam Kumar, "Seizetoken: A Dynamic Load Balancing Algorithm for Distributed System", *International Journal of Advanced Research in Computer Engineering & Technology*, Volume 1, Issue 3, May 2012.
- [2] Abbas Karimi, Faraneh Zarafshan, Adznan b. Jantan, A.R. Ramli, M. Iqbal b.Saripan, "A New Fuzzy Approach for Dynamic Load Balancing Algorithm", (*IJCSIS*) *International Journal of Computer Science and Information Security*, Vol. 6, No. 1, 2009.
- [3] Valeria Cardellini, Michele Colajanni and Philip S. Yu, "Dynamic Load Balancing on Web-server Systems", *IEEE Internet Computing*, vol. 3, no. 3, pp. 28-39, May-June 1999.
- [4] Sagar Dhakal, Majeed M. Hayat, Jorge E. Pezoa, Cundong Yang, and David A. Bader, "Dynamic Load Balancing in Distributed Systems in the Presence of Delays: A Regeneration-Theory Approach", *IEEE Transaction On Parallel And Distributed Systems*, Vol. 18, No. 4, April 2007.
- [5] Xiao Qin, Hong Jiang, Yifeng Zhu and David R. Swanson, "A Dynamic Load Balancing Scheme for I/O-Intensive Applications in Distributed Systems", In the *Proceedings of 2003 International Conference on Parallel Processing Workshops (ICPP 2003 Workshops)*.
- [6] Urjashree Patil, Rajashree Shedge, "Improved Hybrid Dynamic Load Balancing Algorithm for Distributed Environment", *International Journal of Scientific and Research Publications*, Volume 3, Issue 3, March 2013 | ISSN 2250-3153.
- [7] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, "Performance Analysis of Load Balancing Algorithms", *World Academy of Science, Engineering and Technology* 38 2008.
- [8] Ali M. Alakeel, "A Fuzzy Dynamic Load Balancing Algorithm for Homogenous Distributed Systems", *World Academy of Science, Engineering and Technology* 61 2012.
- [9] Marc H. Willebeek-LeMair, Anthony P. Reeves, "Strategies For Dynamic Load Balancing On Highly Parallel Computers", *IEEE Transaction On Parallel And Distributed System*, Vol. 4. No. 9. September 1993.
- [10] Tarek Helmy, Fahd S. Al-Otaibi, "Dynamic Load-Balancing Based on a Coordinator and Backup Automatic Election in Distributed Systems", *International Journal of Computing & Information Sciences* Vol. 9, No. 1, April 2011.