Least-Connection Algorithm based on variable weight for multimedia transmission

YU SHENGSHENG, YANG LIHUI, LU SONG, ZHOU JINGLI
College of Computer Science
Huazhong University of Science & Technology,
1037 LuoYu Road, Wuhan, 430074
P.R.CHINA

Abstract: A least-connection algorithm based on variable weight is presented in this paper in order to satisfy the requirement of multimedia transmission, which is based on the analysis of the existing cluster architecture, algorithms of the load distribution and balancing of network nodes. A validating trial has been performed and the results show that our algorithm has effective load balancing in one central control node scenario.

Key words: multimedia transmission, cluster, load balancing

1 Introduction

Network-based video transmission and its applications have emerged as promising technologies with the development of computer network technology. With the continuous increase of network bandwidth, applications of multimedia server supported storage and playing of video and audio in Video-on-Demand network video transmission and digital monitoring system have been widely accepted.

Cluster is an approach that can meet the ever-increasing capacity and reality requirement of data transmissions.

Multimedia servers can be classified in two categories: monolithic server which commonly appears as a high performance server for the purpose of storage, transmission and management of video data stream, and cluster server which is a cluster of workstations and PCs. Traditional high performance server is actually a monolithic server employing symmetrical Multi-processor (SMP). It has been limited by its fixed system architecture that cannot be tuned with performance requirement and the high price. Cluster server has its advantages in terms of ratio of performance to price, flexibility, and capability of supporting different architectures, hence it has received increasing attention recently.

A typical cluster server consists of a group of processing nodes, where each node has one or more disks, or even disk array. All the nodes are connected via high-bandwidth switch or network, which provides flexible solution to meet the requirement of real-time multi-media streams.

The most significant parameters are I/O bandwidth and storage capacity. I/O bandwidth determines the number of clients that a sever can support, and the storage capacity determines the number of media streams that a server can store. In monolithic server, client's request will not be blocked as long as there is surplus I/O bandwidth [1]. But this will not be true in cluster server, where the I/O bandwidth of a server is completely distributed. For example, a request can still be blocked at one particular node although other nodes are free at the request arrival time. Therefore the realization and management of cluster server are more complicated than that of the monolithic server. There are four basic questions that need to be addressed [2]:

1) Load balancing: Because cluster server is composed of a number of nodes, its computing

task depends on the corporation of each node. The system can get optimized performance only when every node in the system maintains balancing.

2) Single System Image: Cluster server must make all nodes transparent to users by providing abstract user interface as a single system.
3) Scalability: Because of high requirement of multimedia applications to server and big expenditure, users might divide the investment into several parts. The server is also required to have high scalability, e.g., to add new node, in order to satisfy the future increased demands.
4) Availability: Clusters are susceptible to partial failures. And probability of partial failure increases with the increasing of the size of system resources. The implementation of clusters must take partial failures into account to maintain high availability and reliability.

The goal of this paper is to introduce least-connection algorithm based on variable weight for multimedia transmission. The remainder of the paper is organized as following, section 2 gives a literature survey of the existing load balancing algorithms. In section 3 we proposed our improved algorithm. The experiment result was shown in section 4, and in section 5 we conclude.

2 Existing load balancing algorithm

The main purpose of load balancing is to distribute load among a number of nodes to optimize the utilization of the computation capability of every node and reduce the average task response time as well, this is equivalent to maximize the system throughput. The modus operandi is a special computer(also called request distributor) that receives and distributes all task requests to every server in the cluster according to some rules.

There are scheduling algorithms in the literature [3,4,5,6]: Round-Robin Scheduling, Weighted Round-Robin Scheduling, Least-Connection Scheduling, and Weighted Least-Connection Scheduling.

Round-Robin: Round-robin Scheduling, in its word meaning, directs the request received from network to the different node in a round-robin manner. It treats all nodes as equals regardless of number of connections. The scheduling granularity is node-based, this will lead to significant dynamic load imbalance among the nodes[3].

Weighted Round-Robin: The weighted round-robin scheduling can treat the nodes of different processing capacities. Each node can be assigned a weight, an integer value that indicates the processing capacity. The default weight is 1. For example, three nodes, A, B and C, have the weights, 4, 3, 2 respectively, a good scheduling sequence will be ABCABCABA in a scheduling period (mod sum(Wi)). In the implementation of the weighted round-robin scheduling, a scheduling sequence will be generated according to the node weights after the rules of node are modified. Therefore, scheduling the request is no longer in a round-robin manner[4].

The weighted round-robin scheduling doesn't need to count the request connections for each node, and the overhead of scheduling is smaller than other dynamic scheduling algorithms, it can have more nodes. However, it may lead to dynamic load imbalance among the nodes if the load of requests vary highly. In short, there is possible that most of long requests may be directed to one node.

The round-robin scheduling is a special instance of the weighted round-robin scheduling, in which all the weights are equal. The overhead of generating the scheduling sequence after modifying the node rules is trivial, and it doesn't add any overhead in real scheduling. So, there is unnecessary to implement the round-robin scheduling alone.

Least-Connection: The least-connection scheduling algorithm directs requests received from network to the node with the least number of established connections. This is one of dynamic scheduling algorithms; because it needs to count live connections for each node dynamically. At a node where there is a collection of nodes with similar performance, the least-connection scheduling is good to smooth distribution when the load of requests vary a lot, because all long requests won't have chance to be directed to a node[5].

At a first look, the least-connection scheduling can also perform well even when there are nodes of various

processing capacities, because the faster node will get more connections. In fact, it cannot perform very well because of the TCP's TIME_WAIT state. The TCP's TIME_WAIT is usually 2 minutes, between this 2 minutes a busy web site often get thousands of connections, for example, the node A is twice as powerful as the node B, the node A has processing thousands of requests and kept them in the TCP's TIME_WAIT state, but the node B is crawling to get its thousands of connections finished. So, the least-connection scheduling cannot get load well balanced among nodes with various processing capacities.

Weighted Least-Connection: The weighted least-connection scheduling is a superset of the least-connection scheduling, in which you can assign a performance weight to each node. The nodes with a higher weight value will receive a larger percentage of live connections at any one time. The node administrator can assign a weight to each node, and network connections are scheduled to each node in which the percentage of the current number of live connections for each node is a ratio to its weight[6]. The weighted least-connections scheduling works as follows:

Supposing there are n nodes, each node i has weight Wi (i=1,..,n), and alive connections Ci (i=1,..,n), ALL_CONNECTIONS is the sum of Ci (i=1,..,n) • $\sum C_j$ •, the next network connection will be directed to the node j, in which

$$\frac{C_j / ALL\_CONNECTION\_S}{W_j} = \min\left(\frac{C_j / ALL\_CONNECTION\_S}{W_j}\right)$$

•i•1,…,n• •1•

Since the ALL_CONNECTIONS is a constant in this lookup, there is no need to divide Ci by ALL_CONNECTIONS, it can be optimized as

$$\frac{C_j}{W_j} = \min\left(\frac{C_j}{W_j}\right) \quad \bullet i \bullet 1,\ldots,n \bullet \quad \bullet 2 \bullet$$

3 Least-Connection Algorithm based on variable weight

In this section, we proposed an improved algorithm – “Least-Connection algorithm based on variable weight”. As its name implies, the contribution of this algorithms is that it takes variable weight into account.

Necessity of variable weight: Looking back above Weighted Least-Connection, we noticed that: in equation (2),the way of finding the node j with least load in a cluster is to calculate the ratio of connection number(C_j) and the fixed weight(W_j) in turn, and choose the node with minimum value. However, this did not take into account that servers with different content TCP connection have different overhead, hence this algorithm will lose efficiency when the overhead difference between servers cannot be ignored. In the existing networks with multimedia servers, the overhead of TCP connections can be significant enough that the affect it makes to the Weighted Least-Connection algorithm must be taken into account, in other words, equation 2 is not enough to reflect the real load of every node. And this is the motivation of bringing variable weight into account in our algorithm.

Components of weight: generally speaking, the parameters that reflect the processing ability of a node server are the processing speed of CPU’s, idle rate, size of memory and capacity of I/O. We define node weight of a node as following:

$$W_j = \sqrt{\left(\rho_{mem} \times Q_{mem} \times R_{mem}\right)^2 + \left(\rho_{cpu} \times V_{cpu} \times R_{cpu}\right)^2}$$

•3•

where• R_{mem} denotes the idle rate of node memory• R_{cpu} denotes the idle rate of node CPU • Q_{mem} denotes the size of memory•unit is K• V_{cpu} denotes the speed of CPU(unit is Mhz), ρ_{mem} and ρ_{cpu} denote the proportion factor•and ρ_{mem} • ρ_{cpu} •1•

introduction of algorithm: after getting the equation of weight, we can get the algorithms of Least-Connection basing on variable weight.

Supposing there are n nodes, each node i has weight

Wi (i=1,..,n), and alive connections Ci (i=1,..,n), the next network connection will be directed to the node j, and node j has the below character:

$$\frac{C_j}{W_j} = \min\left(\frac{C_j}{W_j}\right) \bullet i \bullet 1,\ldots,n \bullet \qquad \bullet 4 \bullet$$

In which W_j is gotten by equation(3).

It must be pointed that in equation (3) the CPU's speed and size of memory are basically constant, things need consideration are the memory and CPU's idle rate which varies all the time.

4 Experiments and Result

We measured the performance of our load scheduling algorithms by using a test platform which consists the following: (1) one computer running Windows2000 server version and load scheduling program as load balancing server. (2) three computers running Windows2000 server version and video service program as multimedia servers. (3) a number of computers running either Windows98, Windows NT or Windows2000 and playing program as video client end.

The work flow is described as follows: (1) request data input: the client requests the load balancing server for playing connection. (2)request processing: the load balancing server separately selects a multimedia server according to different scheduling algorithms, and then directs it the client's connection request. (3)video data output: the multimedia server transmits the video stream to the client and the client plays the received video stream.

We employ C++ as the programming language and the programming kit used is Microsoft Visual C++ 6.0. In order to achieve the reusability and encapsulization of code, all the key algorithms are implemented by OOP: load balancing server uses BalanceServer class, multimedia server uses VideoServer class, both the classes are inherited from Server class.

Experiment is divided into two steps: (1) In the first step, the "Weighted Least-Connection" is selected in the load scheduling program, The number of client connections is increased in steps, and the effect of clients playing is recorded. When the number of client increased to 15, the playing become snatchy with mosaic. Refer Fig 1. (2) In the second step, the "Least-Connection basing on variable weight" is selected in load scheduling program, and the number of client connection is increased gradually in steps. The effect on client's playing is then recorded. From Fig. 2 we can see that the client still plays fluently when the number of clients is increased to 15. When the number is increased to 20, the playing become snatchy and accompanied with mosaic.

Fig 1: Weighted Least-Connection•# of clients:15•

Fig 2: Least-Connection basing on variable weight•# of clients:15•

Table 1: result of experiment

# of client connection	Weighted Least-Connection	Least-Connection basing on variable weight
10	Playing fluently	Playing fluently
15	7-8 clients playing slowly	3 • 5 clients playing slowly
20	More than half of clients playing slowly, and the last client requested for connection had to wait for response for a moment	8-9 clients playing slowly, 1-2 clients become snatchy
25	Most of clients had to wait for response, 1-2 even lost their connection.	half of clients playing slowly, and the last client requested for connection had to wait for response for a moment

The results of the experiment are presented in table 1.

It was shown in the above result that with certain number of clients, 15 for example, half of the playing using Weighted Least-Connection was snatchy, whereas the play using Least-Connection based on variable weight was still fluent till 20. It can be inferred that the load balancing server using Least-Connection based on variable weight can distribute the load to multimedia servers more efficiently, depending on the real-time alteration of performance of each servers, due enhanced effect when playing MPEG stream simultaneously on certain number of clients, 15 for example. In other words, the algorithms of Least-Connection based on variable weight performs better that the algorithms of Least-Connection basing on fixed weight.

With increased number of clients, the bottleneck of nodes in cluster and limitation of network bandwidth have to be taken into account.

5 Conclusion

In this paper, we studied the technology of multimedia cluster server, and based on the study, we propose an improved algorithm, namely the Least-Connection based on variable weight. Through experiments we showed that our algorithm has a better performance than Least-Connection based on fixed weight.

The load scheduling and balancing policy we employed in the multimedia cluster server is based upon one central control node. With the increasing dimension of multimedia transmission, the number of multimedia servers must increase too, then the bottleneck of central control node cannot be ignored any more. Therefore, it is a new research task and requires further study on how to change the policy from central control to a policy that involves negotiation among the multimedia servers.

References

[1] O.Rose, Simple and Efficient Models for Variable Bit Rate MPEG Video Traffic, Performance Evaluation, Vol.30, 1997, pp69-85

[2] Renu Tewari, Architectures and Algorithms for Scalable Wide-area Information Systems, Dissertation of Doctor of Philosophy in the University of Texas at Austin, August 1998

[3] H.Schulzrinne, RTP Profile for Audio and Video Conferences with Minimal Control, Internet Draft, draft-ietf-avt-profile-new-01.ps, Internet Engineering Task Force, Jan., 1998, Work in Progress.

[4] C.Reummler & D.Wilkes, An Introduction to Disk Drive Modeling IEEE Computer Vol.27, No.3 March, 1994, pp17-29

[5] E.Shriver, Performance Modeling for Realistic Storage Devices, PhD Thesis , May, 1997, Univ. New York

[6] D.M.Jacobson & J.Wilkes, Disk Scheduling Algorithms Based on Rotational Position technical Report , HPL-CSP-91-7,Rev1,HP