A Client Proximity Based Load Balance Algorithm in Web Server Cluster

Pei-rui JIANG, Li-min MENG*, Yu-zhou SHANG and Yang-tian-xiu HU

College of Information Engineering Zhejiang University of Technology Hangzhou,
Zhejiang, China, 310023
*Corresponding author

Keywords: Web server cluster, Client proximity based load balance algorithm, Round robin scheme, Modified random allocation scheme.

Abstract. Nowadays, the traffic to the popular site has increased dramatically. Some websites usually have different servers geographically distributed inconsistently can provide same service and these servers make up a web server cluster. In order to guarantee the QoE of clients, load balance scheme is necessary during the servers of the web server cluster. In this paper a novel load balancing algorithm which considers the proximity between client and server is proposed. We compare the Client Proximity Based Load Balancing algorithm (PBLB) with the Round Robin (RR) scheme and the Modified Random Allocation (MRA) scheme. Through the simulation results, we can see the PBLB algorithm has a better performance.

Introduction

With the rapid popularization of Internet and WWW (World Wide Web) applications, the performance of web server has become one of the key factors that affect the quality of network services. Web server cluster brings new solution for web server system because of its expansibility, high reliability and low cost performance. In the web server cluster multiple servers work simultaneously and thus can have higher response efficiency as well as higher reliability. So more and more Internet information service systems adopt such solution, such as Yahoo, Netscape and so on. The famous network query system Google uses a cluster of thousands of servers to provide services to the outside world [1].

Load balancing is one of the most important problems in cluster systems. The goal of load balancing in a cluster system is to assign tasks commensurate with the performance of the processor to minimize application execution time [2]. For a Web server cluster, it is necessary to assign each server the amount of requests that match its processing capacity based on the load balancing principle. Through the appropriate request load scheduling algorithm, some servers in the cluster can avoid heavy workloads, and some servers have a light workload and reach the goal of short average response time.

At present, the classic load balancing algorithms include Round-Robin algorithm (RR), Weighted Round-Robin algorithm (WRR), Random Allocation algorithm (RA), dynamic feedback scheduling and so on [3~5]. In addition to the classical algorithms, scholars have also figured out many other algorithms. Study [6] proposed a load balancing algorithm based on DNS, firstly the load information of the server nodes is obtained by DNS and then the load balancing is carried out. But this approach requires DNS communicate with the Web server constantly so that increasing the network traffic. Study [7] proposed a dynamic adaptive load balancing algorithm based on distributed cluster architecture, the algorithm has the on-line load forecasting mechanism. By adjusting the sampling mode of load information, the network switching pressure can be effectively reduced and the response speed of the algorithm can be improved. A dynamic priority scheduling based on cognitive trusted model was presented in [8], this approach ensures that tasks run in a secure environment and improves the success rate of cluster task allocation. Study [9] summarized 3 kinds of distributed cloud computing cluster load balancing algorithm: random sampling algorithm, honeybee algorithm and active clustering algorithm. This paper also analyzed and compared the relationship between cluster performance and the number of nodes and the
performance of nodes when using the 3 algorithms. In [10], a weighted minimum connection algorithm based on exponential smoothing prediction is proposed, which dynamically predicts the load of the cluster according to the current task of the cloud computing system, and improves the resource utilization rate of the cluster. Study [11] proposed a predictive dynamic load balancing algorithm which also takes services types into consideration, during the loads allocation by using the predictive algorithm the appropriate server node is chosen to achieve optimal load balancing. Study [12] modified the traditional round robin algorithm which takes the servers’ state into consideration, so that it can achieve better effect of load balancing.

However, most research about load balancing only consider from the perspective of server’s own. In this paper, we develop a new load balancing approach which comprehensively considers the relationship between the server and the client.

Later in this paper, we illustrate the architecture of web server cluster in section II; we also introduce some related load balancing algorithms in section III; we explain the proposed Client Proximity Based Load Balancing algorithm (PBLB) in section IV; simulation results analysis is explained in section V. Finally, section VI is the conclusion of the paper.

Architecture of the Web Server Cluster

Generally speaking server cluster is a technology which connects multiple systems and makes the multiple servers working like a single machine. Although there are many servers in a cluster, each server does the same service. So from the view of the client, the server cluster just likes a single server.

As show in the Figure 1, in a web server cluster there are more than one servers which are connected through a high-speed network to solve user requests and each server machine operate as a node which has its own control system. Although the web server cluster works as a single machine, from the interior of the server cluster system each server works independently. When request loads come, they are assigned to the every servers properly so that improving the overall performance of the system which makes the server cluster work as a high-performance and high-available super server.

Always the server cluster has extremely high reliability, because a server cluster contains several servers with shared data storage space, every server communicates with each other via an internal LAN; when one of the servers fails, the application it runs will be automatically took over by another server. In most case, all computers in the cluster share a common name, and any server in the cluster system can be used by all network users; each server can assume part of the computing task, and because of the performance of clustering multiple servers, the overall computing power of the system will be improved; at the same time, each server can assume certain fault-tolerant tasks, when a server fails, the system can isolate this server from the system under the special software support, and load balancing is achieved by the new load transfer mechanism among servers, and send the alarm signal to the system administrator.
In the server cluster, one of the most important components is the load balancer which can assign the client requests to all the server nodes according to the specific scheme. Always we figure out algorithms of load balancing and deploy the schemes into load balancer to control the loads allocation. By a reasonable load balance plan and the load balancer, client requests can be dispatched to the server nodes evenly and clients can get quick service so that have good QoS.

Related Load-balancing Algorithms

Now, we introduce two related load-balancing algorithms, MRA and RR.

Modified Random Allocation algorithm (MRA) is the improvement of the Random Allocation algorithm (RR) which is the simplest scheme. RR algorithm just assign the load to the available servers randomly without considering the number of loads on each server. The MRA scheme selects the loads randomly and assigns them as far as possible equally. For example there are 1000 request loads and 10 servers, so each server is allocated 100 request loads but the loads are random selected which may lead to a situation of server overload, because of the big loads concentrate distributing on the same server. To some extent this algorithm can achieve load balancing but the effect is not well.

Round-Robin algorithm (RR) is one of the most basic load balancing algorithms being used. The principle of the RR algorithm is to assign the client’s request to the servers one by one, from the first one to the last one and then restart the cycle. For instance there are n servers in the web server cluster \( N = \{N_0, N_1, N_2, \ldots, N_{n-1}\} \) and the client’s request is expressed as \( R_i \) where \( i \) represents the request’s number. Then the RR algorithm can be described through a simple mathematical expression:

\[
x = i \mod n
\]

When request \( R_i \) comes and it will be dispatched to the server \( N_x \).

A Client Proximity Based Load Balance Algorithm

We firstly make the definition of client proximity, it represents the distance relationship between a client and a server. We explain it in two aspects.

1. The number of route hops from a server node to the client which sends a request, indicated by \( h \).
2. The Round-Trip Time (RTT), indicated by \( t \).

So we use the following formula to express the client proximity:

\[
P = h + t
\]

\[
Q = \frac{1}{p}
\]

Value \( Q \) is the intuitive expression of the sever node’s client proximity, it is obviously to see that the greater the \( Q \) value the higher the client proximity.

Algorithm description:

Suppose that there are \( N \) homogeneous sever nodes in web server cluster and the client is single. We use \( S \) to represent the collection of server nodes, \( S = \{S_1, S_2, \ldots, S_N\} \).

Step1: Sending the same request to each sever node of the web server cluster from the single client, by comparing the every sever node’s response time we can get the client proximity of each server node.

Step2: Renumbering and sequencing for the sever nodes. Suppose that a sever node \( \{S_i\} \in S \), \( Q_{S_i} \) represents the client proximity of the sever node \( i \). If the \( Q_{S_i} \) is the maximum value in all sever nodes, using \( S_{\text{max}1} \) to represent it, and if the value \( Q_{S_i} \) is the second, then using \( S_{\text{max}2} \) to represent it. So according the rule of renumbering and the result of step1, we sequence the sever nodes as collection \( \{S_{\text{max}1}, S_{\text{max}2}, S_{\text{max}3}, \ldots, S_{\text{max}N}\} \).

Step3: When requests comes, we assign the first \( N \) requests to the sever nodes one by one based on the sequencing result of step2 (assigning the request to the \( S_{\text{max}1} \) then to the \( S_{\text{max}2} \) in order).
Step4: When the N+i request comes, we calculate the $x \equiv \text{mod}(N+i, N)$. If $x \neq 0$, we consider the loads on sever nodes $S_{\text{max}(x)}$ and $S_{\text{max}(x+1)}$ which belong to collection {$S_{\text{max}1}, S_{\text{max}2}, S_{\text{max}3}, \ldots, S_{\text{max}N}$}. If $L_{S_{\text{max}(x)}} < L_{S_{\text{max}(x+1)}}$, assigning the request to the $S_{\text{max}(x)}$, or else assigning the request to the $S_{\text{max}(x+1)}$ ($L_{S_{\text{max}(x)}}$ means the load on sever node $S_{\text{max}(x)}$). If $x=0$, we consider the loads on sever node $S_{\text{max}N}$ and $S_{\text{max}1}$, if $L_{S_{\text{max}N}} < L_{S_{\text{max}1}}$, assigning the request to the $L_{S_{\text{max}N}}$, or else assigning the request to the $L_{S_{\text{max}1}}$.

So according the steps 1 to 4, we can achieve the load balancing and in the next section we do some simulations to demonstrate the practicability of the algorithm.

**Experiment and Result Analysis**

In order to verify the effectiveness of our approach, we use the MATLAB to make some simulations of the Round-Robin scheduling and the Modified Random-Allocation scheduling to compare with our algorithm. It’s easy to find out the time complexity of the three methods is $O(n)$ where the $n$ is the number of servers. So by calculating the percentage of server utilization, response time and load variance, we can prove the advantage of our approach in contrast to the other two schemes.

The percentage of server utilization indicates the ratio of loads on server to the server’s capacity. We use $P$ to represent it, $l_i$ is the loads on server $i$ and $L_i$ is the capacity of server $i$.

$$P = \frac{l_i}{L_i}$$

The response time is the time that from the client sends a request to receiving the response.

The load variance can reflect the dispersion of the loads between servers, the smaller value of load variance the better effect of load balancing. We can calculate the load variance by using the following formula. The Var means the load variance of all servers; $l_{\bar{s}_i}$ represents the loads on sever $s_i$; $\bar{l}_{s_i}$ represents the mean value of loads on all servers; $n$ means the number of servers.

$$\text{Var} = \frac{\sum_{i=1}^{n}(l_{s_i} - \bar{l}_{s_i})^2}{n}$$

In order to simplify the simulation, we use 4 homogeneous servers and single client. We random generate 100 different loads between 0 and 1 firstly and then using the different approaches to dispatch the loads for 50 times. By comparing the results of these approaches we can see the better performance of our algorithm.

![Figure 2. Percentage of sever utilization using Modified Random Allocation algorithm.](image1)

![Figure 3. Percentage of sever utilization using Round-Robin algorithm.](image2)
From the above graphs, we can easily find out that the loads are distributed more balanced using PBLB algorithm, comparing with the other two approaches.

With the increase number of requests, the response time of PBLB algorithm is smaller and smaller than the other two algorithms. It is easy to know that the shorter response time the better effect of load balancing. So the PBLB approach has better performance.

Summary
A client proximity based load balancing algorithm is presented in this paper. This algorithm comprehensively considers the proper distribution of loads between servers and the proximity of the clients and the server. We compare the algorithm with several previous algorithms. The results show the advantage of our algorithm in load balancing and response time. The algorithm can get better effect of load balancing and shorter response time comparing with the other two algorithms.

In this paper, we only consider the situation of single client, so in the future research we need consider multiple clients situation and find out the better algorithm.

Acknowledgement
This work was supported by the National Natural Science Foundation of China (61372087).

References


