UDC 004.738, DOI: 10.2298/CSIS1001153Z

A Content-based Dynamic Load-Balancing Algorithm for Heterogeneous Web Server Cluster

Zhang Lin[1], Li Xiao-ping[2], and Su Yuan[2]

[1]School of Electricity and Information Engineering, BUCEA, Beijing, 100044, China
zzcmeng0806@sina.com
[2] Department of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081
lxpmx@x263.net

Abstract. According to the different requests of Web and the heterogeneity of Web server, the paper presents a content-based load-balancing algorithm. The mechanism of this algorithm is that a corresponding request is allocated to the server with the lowest load according to the degree of effects on the server and a combination of load state of server. Besides, apply a method of random distributing base-probability to assign each request to an appropriate server in terms of their weight. All the parameters that will be used in the algorithm can be acquired by simulated test. Experimental results suggest that this algorithm can balance the load of web server clusters effectively, make full use of the existing source of software and hardware, highly improve the server's performance, and even make the best use of the web server.

Keywords: Web server cluster; load-balancing; dynamic feedback

1. Introduction

A Web cluster refers to a Web site that uses two or more server machines housed together in a single location to handle user requests. Although a large cluster may consist of dozens of Web servers and back-end servers, it uses one hostname to provide a single interface for users [2].

Request distribution and load balance are key technological means to realize web server clusters, but most existing request distribution algorithms are static ones and they don't have any optimization strategy and don't synthetically consider the dynamic parameters such as the utilization ratio of CPU and memory. With the apply of Jsp、database and multimedia, the request coming from different client will bring different load on web server. So those existing equilibrium algorithms are inefficient and erratic and don't fit to construct heterogeneous Web server clusters. This paper considers the heterogeneous characteristic of hardware and software on every node and puts forward a content-based load-balancing algorithm of application layer—

QSC-Load Balancing algorithm. Experimental results suggest that it can improve the response time and throughput of the system.

2. Analysis of the content-based request distribution strategy

In the cluster system based on TCP/IP request distribution [1], the fore-end is transmitted based on the IP and the target port, without knowing the clients' submission. However the distribution based on the application layer means the target content must be informed beforehand (for instance the URL information carried by the massage head of HTTP). In addition, information will be configured and the request will be distributed to the corresponding back-end server's node according to the matching rules. Take web service as an example, it requests the client and the fore-end to build up a complete TCP connection so as to acquire the head message of HTTP. After receiving the HTTP requirement, the fore-end starts to analyze and apply the suitable distribution strategy in order to transfer the request to the back-end server by which the request will be carried out [6].

As for the content, different distribution strategies have different definitions [3]. Rough divisions can be made according to the types of the document, such as HTML, GIF, and CGI. Relatively detailed contents mean the complete parts of URL in the HTTP's head message, that is the web object's oriented message expected by the client. This paper discusses the latter content, which is the specific URL message [7].

HTTP's text is divided into two parts: the first part describes the format and content of the request message sent by client; the second part describes the format and content of the answer message sent by Web server. This paper is only concerned with the contents of the request message. HTTP request consists of two parts, one is a request head with line structure and the other is request body. The request head includes several parts as follow:

1. Method: located in the first line of request head, pointing out the request operational types. There are four main methods: GET、PUT、 POST and HEAD.
2. URL: located in the first line of request head and after the method, pointing out the location of the target and other information.
3. Protocol version: pointing out the HTTP protocol version number applied by the client browser.
4. Compressed mode: pointing out the data compressed mode supported by the client browser.

In the HTTP request message, the target URL after the GET method in the first line is an important gist applied in the request distribution system.

3. Design of QSC-load balancing algorithm

3.1. Theoretical analysis of QSC-loading balancing

From the logical resource point of view, the web server can be regarded as one composed of several resource nodes providing all kinds of service. It is an important problem of how to share the entire service ability in the entire server, without overloading a single node[5].

According to the degree of influence of the server, this paper divides the request into static request, database operating request and safety request. Each request is endowed with specific weight-value. The calculation formula of request weight-value is defined as follow:

w[i] = CPU process time* a +memory use*p+ disk access time*(1-a-p)

w[i]:relative weight-value of request i; a, p: relative weight-value coefficient, namely the importance of every class resource.

In order to simplify the problem, this paper doesn't consider the practical factors, such as the choke point and the wastage in the internet transmission. Fro the convenience of the research, time is equably divided into discrete time slices(0 T) (T 2T)…(kT (k+l)T)….

Definition 1: the start moment of every time slice indicates this period of time,such as kT represents (kT (k+1)T).

Definition 2: C indicates the processing ability of the server in the unit time.

Definition 3: C(kT) indicates the processing ability of a web server within kT.

Definition 4: S(i,kT) indicates the needed processing ability of mission i within kT.

Definition 5: N(kT) indicates the allowed number of mission within kT.

So we can get an equation as follow:

$$C(kT) = c*T \geq \sum_{i=1}^{N(kT)} S(i,kT) \tag{3-1}$$

When the result of equation (3-1) is true, the request i is allowed to enter the service queue, otherwise it will be refused. This equation shows that the maximum of the operating lag of the request is T and the value of T affects the system efficiency. So we must choose the different T according to the processing ability of the system.

3.2. Description of QSC-load balancing

3.2.1 Performance parameter of server

In order to achieve higher system throughput and shorten the client's feedback time, the algorithm adopted both static and dynamic parameters to reflect the capability of sever [4].

1. Static performance parameter

The parameter of software and hardware of the web server determine its capability. During the processing of the web server cluster, these parameters are changeless, so they are also called static performance parameters. This paper selected the following static performance parameters:

(1) CPU processing ability: in the cluster system, the process of sever monitor inspects the number and type of CPU, and endows, according to previous experience value, the server with a moderate CPU processing ability value.

(2) Memory parameter: the process of sever monitor inspects the size of computer's physical memory and virtual memory, so as to get the memory parameter.

2. Dynamic performance parameter

During the processing of cluster system, each web server's load is changing as time goes on, the system has to estimate the load-balance according to the real-time server load, and these are called dynamic performance parameters. This paper selected the following dynamic performance parameters:

(1) Processor utilization ratio: it can reflect the busyness degree. The process of sever monitor inspects the CPU utilization ratio termly, so as to confirm the CPU's load.

(2)Memory utilization ratio: the size of the server memory changes as the system runs. The process of sever monitor inspects the utilization ration of physical memory termly, so as to confirm the server memory's load.

(3)Network flow: the network data are mainly transferred through TCP mode in cluster system, the process of sever monitor inspects the packages received and sent by server termly, so as to get the server's load.

3.2.2 Division of request types

1. Issuance type: mainly providing static information. This request is the easiest one, including html or other undivided documents.

2. Affair type: mainly coming from the operation on dynamic database, and the condition of operation is provided by user via a dynamic HTML page. Because it needs dense access to the disk, and the reserved operation to obtain the type of the document, the weight-value is large.

3. Dynamic page type: needs to analyze page information dynamically, including obtaining the extended name such as jsp, asp and php, the weight-value is large.

4. Security type: provides static, dynamic and security information transmissions. During the security transmission, encrypting operate will consume large numbers of CPU resource and it needs dense access to the disk, so the weight-value is large.

5. Multi-media type: provides with real-time audio and video services. The weight-value is large.

3.2.3 Description of QSC-load balancing

When the linking number N exceeds the prescriptive limit value $N_{\max}$, which is determined by the server ability, it will get a view. Here the processing time becomes longer and we call it key state, $N_{\max}$ as key number. In order to insure the server performance, N should avoid approaching $N_{\max}$.We can confirm that the server is loaded by the web linking number exceeding the limit value.

In web cluster system, Cluster_Server_Table has requests of n style and servers of m brand, $1 \le i \le N$, each request is endowed with given value w[i] via Service_Maping_Table(SMT), $1 \le i \le N$,so the request load of server j is:

$$Load[j] = \sum_{i=1}^{N} R(i)[j] * w[i] \tag{3-2}$$

R(i)[j] is R(i) requests to connect with server j, so the request load of server j is:

$$Load[j] = \sum_{i=1}^{N} R(i)[j] * w[i] + w[k] \tag{3-3}$$

Server j has disposed request R(i), so the request load of server j is:

$$Load[j] = Load[j] - w[i] \tag{3-4}$$

The judging measure of load-balancing:

Judging method of load-balancing: read in every line of CST, and then write into a record array, thus the array's every record represents all information of one server. Take the first working server's information (assume server M) as benchmark and put it and other working server's (assume server i) information into equation (3-5) to compare:

$$ratio = \alpha \times \frac{\lambda 1_i \times C_i}{\lambda 1_M \times C_M} + \beta \times \frac{\lambda 2_i \times M_i}{\lambda 2_M \times M_M} + \gamma \times \frac{N_i}{N_M} + \theta \times \frac{R_i}{R_M} \tag{3-5}$$

Parameters' meaning are as follow:

$\lambda 1$: CPU processing ability.

$\lambda 2$: memory parameter.

C: CPU utilizing ratio.

M: memory use ratio.

N: network flux.

R: request load.

α: CPU comparative weight-value.

β: memory comparative weight-value.

γ: network comparative weight-value.

θ: request load comparative weight-value.

The initial values of α、β、γ、θ are all 1. According to the cluster running fact we can increase or decrease some weight-value to enhance or weaken the corresponding load performance. The static and dynamic parameters are all considered when we are comparing the performance of CPU, memory and network.

We will have two problems when calculating the lightest load server in the CST. On the one hand, constantly calculating the computer's load will result in reduction of the computer's capability. On the other hand, when two continuous overloads are distributed into the same sever, it may respond quite slowly to the client's. In order to solve those problems, this paper applies a method of random distributing base-probability to assign each request to an appropriate server in terms of their weight. First of all, calculate the comparative weight-value to ensure its probability space via each sever, all the cluster server system space is 1. Fig.1 shows 4 probability spaces of the servers with the relatively weight-value of 0.1, 0.15, 0.25 and 05.

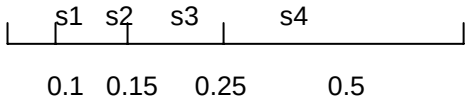

Fig. 1. Probability space of servers

When a new request mission reaches distribution mechanism, it temporarily calculates a random number between [0,1].According to the random number's placement in the probability space, make sure the target server of this transmission. Because the lower weight-value server has larger probability space, it will have higher allocation probability, meanwhile, each transmission is an individual event that can neither be affected by the former transmission, nor be affected by the next transmission operation, so it can reach more average effects. This is the improved algorithm of QSC-load balancing. Here is the description:

Load synchronization ( ) {

For each server j in CST

```
        if request R(i) responses
          set the request load of server j: load[j]'=load[j]- w[i];
        if delay service queue is not null
          Pop the first request in the queue;
          distribute the out request according to request-distribution();
    Heart beat ();//inspect current load and the server's useableness in CST
 }//Load synchronization ()
Request distribution (){
  if request r arriving
    Search Type(); //search in Service_Maping_Table(SMT) and identify the type of request r
    calculate the load of server j:load[j]'=load[j]+ w[i];
    if the number of server exceeds connection number N_max
      set the load of server j:load[j]'= infinite;
     else
       For k=1 to k=n
         if  load[k]= infinite (all servers are all overload)
           put the request into the delay service queue;
         else
           calculate the comparative weight-value of server k and confirm it's probability space R(k);
          rand=random();
          if rand belongs to R(k)
            distribute the request to server k
}//Request_distribution()
Search_Type(){
Switch request type:
  Case issuance type:calculate it's request weight-value; break;
  Case affair type: calculate it's request weight-value; break;
  Case dynamic page type: calculate it's request weight-value; break;
  Case security type: calculate it's request weight-value; break;
  Case multi-media type: calculate it's request weight-value; break;
}
```

3.3. Analysis of QSC-Load Balancing

1. Query of hash table

Since the load equalizer needs to read and write corresponding to records from User_Info_Table, Cluster_Server_Table and Service_Maping_ Table,the operation efficiency will effect the system performance.Hashtable is an important storage and index measure and it can query、insert and delete the record efficiently. This algorithm adopts Hash table with bidirectional double linked list to operate the record.

2. Enactment of $N_{\max}$、T

Enactment of $N_{\max}$、T is the key problem in this algorithm. The content-based distribution strategy is a exiguous granularity load-balancing one and the better of server performance the larger of $N_{\max}$ value. We can set a same $N_{\max}$ in the isomorphic server clusters, but in an isomerous system it is necessary to set a different $N_{\max}$ according to the server performance. We should consider that an improper $N_{\max}$ will make the load unbalanced. The value of T also affects the system performance. The state of each server in the cluster isn't reflected if it is too large, however if it is too small there will be dithering. So the value of $N_{\max}$ and T must be generally considered.

3. Dynamic feedback mechanism

The real-time load and response ability of every node are considered in dynamic feedback mechanism. The load equalizer allocates the load according to the real-time feedback information and improves the system efficiency. In this algorithm, the load equalizer checks the load at T intervals and gets the load state via the dynamic feed mechanism.

3.4. Testing of QSC-Load Balancing

Suppose the request is disposed by the method of first in and first out or share time slice. In simulated program, each physical entity is abstracted a Java class, log analysis is an interface class and can be used to dispose different input log. When the load equalizer receives those requests that are disposed by log program, it will distribute them into different servers. The input stream of simulator is a object request one with mark, which can produce asynchronously or via disposing log in server.

Simulate three different request strategies: least_connections strategy, random strategy, QSC-Load Balancing strategy. The throughput contrast of cluster server system is shown as Fig. 2:

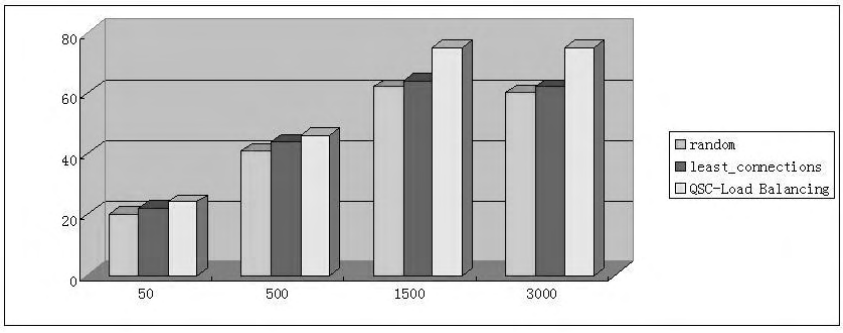

Fig. 1. Throughput contrast of cluster server system

The response time contrast of cluster server system is shown as Fig. 3:

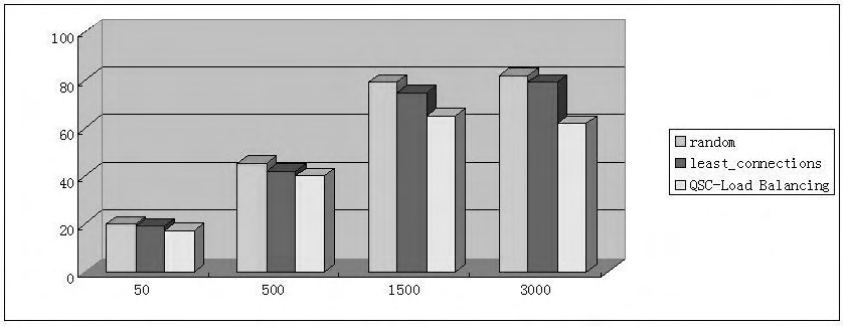

Fig. 3. Response time contrast of cluster server system

Have a research on the load situation of requesting distribution and stimulating server through the analysis of the log on the websites. In the stimulated system every object can be abstracted to a Java class. Log analysis is an interface class, which used to handle different input logs. According to the different request to the load equalizer, this can be equipped within the basic content and maintaining line, will send the request to different server.

4. Conclusions

This paper points out a developed strategy on the basis of the context distribution contra pose to the different requesting content. Show the load condition of every server in the web class more currently through the evaluation of the web content and load server. Load lays the foundation in order to distribute the request balance; introduce the concept of probability-space within the process of distribution to make the load more balance; to eliminate the period related effect due to the adoption of the fixed proportion model. Thus enhance its performance.

5. References

1. Chi-Chung Hui,Samuel T.Chanson.Improved Strategies for Dynamic Load Balancing.IEEE Concurrency,1999.
2. Emiliano Casalicchio,Michele Colajanni,Scalable Web clusters with Static and Dynamic Contents,University of Rome Tor Vergata,ecasalicchio@ing. uniroma2.it.
3. C.-S. Yang, M.-Y. Luo. A content placement and management system for distributed Web-server systems.Proc. of IEEE 20th Int. Conf. on Distributed Computing Systems (ICDCS'2000), Taipei, Taiwan,Apr. 2000.
4. Vivek S. Pai, M. Aron, G. Banga,M. Svendsen, P. Druschel,W. Zwaenepoel, E. Nahum.Locality-aware request distribution in cluster-based network servers.In

Proc. of 8th ACM Conference on Architectural Support for Programming Languages and Operating Systems,San Jose, CA, Oct. 1998.
5. M Arlitt,C Williamson.Web server workload characterization:The search for invariants(extended version).In:ACM SIGMETRICS'96.Marriott,PA,1996.
6. Haakon Brhni A Comparison of LoadBalancing Techniques for Scalable Web Servers University of Oslo EsPen Klovning and ivind Kure,Telenor Research and Development.IEEE Network July/August 2000.
7. Michele Colajanni,Philip S. Analysis of Task Assignment Policies in Scalable Distributed Web-server Systems,IEEE Transactions on Parallel and Distributed Systems,vol.9,No.6,June,1998.

Zhang Lin, Master, lecturer; research fields: network security, software engineering.

LI Xiao-ping, Ph.D., professor; research fields: network security, information processing, multimedia and image processing.

Su Yuan, Master; research fields: network security.

Received: April 10, 2009; Accepted: September 03, 2009.