Nginx cluster distribution strategy based on request load information

MA YuanLong¹,a, JIANG Yi¹,b, ZHANG ZhenWei²,c

¹ College of Computer Science and Technology, Chongqing University of Post and Telecommunications, Chongqing 400065, China
² ZTE Chongqing Research Institute, Chongqing 400065, China

a) 562497276@qq.com, b)jiangyi@cqupt.edu.cn, c) 1172200695@qq.com

Keyword: Nginx; Request load information; Threshold value; Load balancing strategy

Abstract. According to the different characteristics of load values of the user’s request, this paper proposes a Nginx cluster load distribution strategy based on the load information request. According to the relationship between the average load information of server cluster and the threshold value, we select the load balancing strategy and the back-end server for different request distribution. We realize the reasonable distribution of the load and maximize utilization of server resources. Experiments and results show that in this paper the Nginx cluster distribution strategy request based on load information in two aspects of average throughput and average delay are better than the existing load balancing strategies.

Introduction

The rapid growth of Internet users to bring huge benefits to network operators but also bring them great network load [1]. Now we use server cluster to solve this problem. Nginx adopts the following two load balancing categories [2,3]: one category such as the weighted round robin strategy [4], but it does not take the differences in request load information and the back-end server cluster performance into account [5]; the other category such as the literature [6] proposed "the load balancing technology based on content", although this strategy classify user requests by the content, this strategy does not take the information and the back-end server cluster hardware load request consisting of different situations into account.

Nginx cluster distribution strategy based on request load information

The classification of Web requests.

This paper refers to CAP [7,8] Web for classification: (1) “N” request: W publishing service of The static and slightly dynamic; (2) “DB” request: Hard disk intensive service; (3) “CB” request: Intensive use service of CPU-based; (4) “DCB” request: The intensive use service of the hard disk and the CPU.

Server load

Server load evaluation

We use the server load evaluation standard proposed by literature [9]. The specific calculation method as shown in equation (1).

\[
\text{Load} = 1 - (1 - \lambda_c) \times (1 - \lambda_m) \times (1 - \lambda_p) \times (1 - \lambda_b) \times (1 - \lambda_d)
\]  

(1)

The \( C \) is said the utilization rate of CPU, \( M \) is said the utilization rate of memory, \( P \) is said the utilization rate of connections number, \( B \) is said the utilization rate of network bandwidth and \( D \) is said the utilization rate of disk utilization rate of I/O.

The reporting period of load

The load balancer according to the load information of the back-end server as a basis to distribute requests, but there are two problems about the load period \( T \): one is the selection of \( T \) is too short, the other one is the selection of \( T \) is too long. We know the value of \( T \) will influence the performance of Cluster, from experiment we find that it’s best when the value of \( T \) is 10s.
**Load reporting strategy**

In $T$ time not all of the load information of the server will change. In this paper we introduce formula (2) to show the server’s load change rate of $\Delta t$ to control the report server load.

$$\Delta t = \frac{\text{load}(t_i) - \text{load}(t_j)}{t_j - t_i} (t_j > t_i) \quad (2)$$

$\Delta t$ is said this servers’ load change rate; $\text{load}(t_i)$ is said the servers’ load value at the $t_i$ moment, $\text{load}(t_j)$ is said the servers’ load value at the $t_j$ moment. Through the experiment, it’s the best result when the value of $T$ is 10s.

**Request load**

For different types of user requests, it consumes server resources is not the same. Through experiments with different types of request load values we get the value of “N” request is 1, the value of “DB” request is 2, the value of “CB” request is 2 and the value of “DCB” request is 3.

**Selection of threshold value**

We compare the weighted round robin with literature [6] proposed "content based load balancing technology" in average delay. The experiment results show that the switching threshold value of load balancing strategy is one-tenth of the maximum load value of what the server can withstand.

**List form**

This paper we use a binary sort tree to manage the cluster server list. Load balancer need to maintain a binary sort tree for each request access cluster.

**The distribution strategy based on request load information**

**Architecture**

Depending on the load value of the request type request and considering the back-end heterogeneous cluster at the same time. Figure 2.2 gives the structure of the load balancer. In this structure, the load balancer contains the request classification, request distribution and load feedback acquisition module. And we add load information rate module on each server in the cluster server.

![Figure 2.2 The architecture diagram](image)

**The initialization phase**

1. **Initialize cluster partition**
   
   First of all, we let the back-end heterogeneous server cluster initialization for different types of requests cluster according to the back-end server performance metrics for each server in the cluster as well as the amount of each type of request. Such as $C_i \{S_{j}, S_{j}, S_{j}, \cdots \}$, the $C_i$ is said the request type cluster and $S_{j}$ is said the $j$ server in cluster $C_i$.

2. **Initialize the server threshold**
   
   According to the experimental method above. The average value of each server in the cluster as the cluster load average value. And take one-tenth of the average value of cluster load as threshold.

3. **Initialize the load feedback acquisition module**
   
   Load feedback acquisition module according above, establish the type of request processing cluster and the structure of the server nodes. The node is defined as $\text{Node}_i = \{IP_j, Load_j, Max_j, Type_j\}$, $i$ is said the corresponding cluster type; $j$ is said a server in $i$ cluster; $IP_j$ is a IP of a server in
\(S_i; \text{Load}_i\) is the server’s load calculated by formula (1), and initial binary sort trees according to this value; \(\text{Max}_i\) is the maximum load value server can bear; \(\text{Type}_i\) is the type of a request of \(S_i\) processing.

The load balancing module in each type of request processing maintains a node \(\text{Head}_i = \{\text{Load}_i, \text{Threshold}_i\}\) in a cluster, which \(\text{Load}_i\) represents the average load value of the \(i\) server in the cluster and \(\text{Threshold}_i\) represents the average threshold value of the \(i\) server in the cluster.

**Service request processing phase**

The request distribution process consists of eight steps:

1. **Request(url):** The user input url to the load balancer;
2. **Receive(url):** Request classification module receives the user request url, then deliver the url and request type type to the request distribution module in the form (url, type).
3. **Query(url, type):** The request distribution module receives the data (url, type), and then deliver the request type type to the load feedback acquisition module to ask the back-end server ip.
4. **Inquire(type):** The load feedback acquisition module receives the request type type, then asking the value of \(\text{Load}\) in corresponding request classification cluster node Head is more than \(\text{Threshold}\), if not to the step 5 or turn to 6.
5. **Use_WRR:** The request will use Nginx weighted polling to select a server, then to the step 7.
6. **MinCost:** The load feedback acquisition module according to load values load type of the request type to find a maximum server node in the corresponding binary sort tree load type + Load \(\leq\) Max, then to the step 7.
7. **Return:** The feedback module returns the server ip to request load distribution module.
8. The request distribution module receives the server ip, then distributes the user requests url to the server for processing.

**Experiment and result analysis**

In this paper we compare three different load balancing strategies which are the weighted round robin, "load balancing technology based on content" is putted in literature [6] and the improved load balancing strategy we proposed in the two aspects of the average throughput and average delay by experiments [10-12].

We select different number of requests per second, average throughput of four load balancing strategy as shown in Figure 3.1 and Figure 3.2. Experiments show that when the number of concurrent requests is less, as our load balancing strategy use weighted polling strategy and weighted polling strategy, our strategy is superior to based on the content strategy in average throughput and average delay. With the increase in the number of concurrent requests, our load balancing strategy is superior to the existing load balancing strategy based on content in two aspects of average throughput and average delay.

**Summary**

In this paper, we propose a distribution strategy based on request load information. We divide the back-end server cluster into multiple request processing cluster according to user request type. At the same time the load balancer will consider the request load value when distribute the request. The load
Figure 3.1 The average throughput
Figure 3.2 The average delay
Balancer will choose a suitable load balancing strategy for load distribution. According to the server load information is whether or not more than the threshold. Next step we will solve the problems of single point load balancer based on our study.

Acknowledgements

This work was financially supported by the Chongqing Municipal Education Commission scientific and technological research projects (KJ1400414) and the Chongqing University of Posts and Telecommunications, Dr. startup fund, A2015-18.

References