Paper presented at the 22nd Asian Conference on Remote Sensing, 5 - 9 November 2001, Singapore.

The Design and Implementation of Dynamic Load Balancing for Web-Based GIS Services

Myung-Hee Jo* · Yun-Won Jo* · Jeong-Soo Oh** · Si-Young Lee**

* Department of Geodetic Engineering, Kyungil University
33 Buho-ri, Hayang-up. Kyungsan-si, Kyungsang-bukdo, 712-701, Korea
Tel)+82-53-850-7312, Fax)+82-53-854-1272
mhjo@bear.kyungil.ac.kr, chogerry@yahoo.com

** Div. Forest Environment, Korea Forest Research Institute
Tel)+82-2-9612-744, Fax)+82-2-9612-543
jsoh@foa.go.kr, LSY925@chollian.net

KEY WORDS: Load Balancing, Agent, Web GIS, Forest Fire Information Management System

ABSTRACT: In the current server system architecture over web, one of the most important concerns is to handle load coming to Web Server dynamically, so that the server has to enable its Web sites service without any delay or failure or shutdown. Therefore, there is fundamental need to develop technology for dynamic load balancing on web side. In this study, the agent is designed on dispatcher in a Web Server cluster and implemented to distribute load dynamically by considering information related to load coming to the Web Server, especially the connection to the Map Server. Moreover, the experimental result shows that our system architecture has less traffic for Web-GIS services comparing previous architecture. Finally, agent implemented on dispatcher will select the web server in cluster, which has the lowest traffic.

INTRODUCTION

The Internet has grown so rapidly over the last few years and continues to expand at amazing paces. This situation has result in heavy demands on Web Server and has big concerns in terms of performance, scalability and availability of Web services because single server cannot control its load from client's requests.

Especially, in case the information related to natural disaster such as forest fire or flood damage is served on Web, the first thing that should be concerned is about how fast it is presented on the Web sites without delay or shutdown or bottleneck. Because these sites are supposed to have a number of hits from clients simultaneously when the disaster happens, the way to handle load dynamically on Web Server side should remain to be really big concerns.

In other words, if this problem is not properly handled yet, clients become frustrated by slow respond time or refuse connections, and perhaps never visit this site again. (Chris Gago, 1999) Moreover, this site can become unstable or even turned out fail under critical load conditions. Finally, the main goal of the site to present information about disaster within seconds is gone.

To avoid such a problem in your own networks, you need to ensure maximum scalability and availability with a solution that balances the load effectively and protects the user from these bad experiences. Therefore, you are

probably looking for a way to handle your share of that traffic, and even to grow your share. (Chris Gago, 1999)

Nowadays in order to solve this problem, cluster-based services architecture is considered as proper and effective alternative. In addition, the agent proposed in this paper is designed on dispatcher in a Web Server cluster and implemented to distribute load dynamically while current another server typically has led bottlenecks in the server node.

In order to select right one, the agent needs to collect the information from server side such as the CPU load, the Disk load, the memory load and especially the access to GIS database. Considering information related to all above load coming to the Web Server, the agent can select right server by keeping load balancing dynamically so that it is possible to set servers to share the workload effectively. Finally, this agent will reduce the probability that a client gets error message when server fails to work at the Web sites. Thus, highly maintainable and expandable Web sites are continuously available to clients.

Implementation issues as well as the necessary model needed to support the system are briefly discussed. The rest of parts are organized as followings. Chapter 2 discusses the network architecture and agent design. Chapter 3 shows the experimental result is presented. Finally, conclusion remarks appear in section 4.

THE NETWORK ARCHITECTURE AND AGENT DESIGN

In this section, the agent-based dynamic load balancing method is designed and simulated considering Web Server clustering shared with Map Server architecture.

Network architecture

In order to respond faster to a client's request, the way to select best-suited server has been considered as very important job. Recently there have been discussed in several mechanisms to maintain load balancing. In this study, the load balancing mechanisms using dispatcher is introduced to react to a client's request in real time based on information about load balancing such as the connection and between a Map Server and Web Servers.

The basic operation of load balancing method preformed on dispatcher, which is working with the Map Server, is shown in figure 1. As you see here, Web Sever is accessed to database through Map Server. The reason to use this architecture is to reduce processing load at Web Server and Web Server keeps the nature of previous functionality.

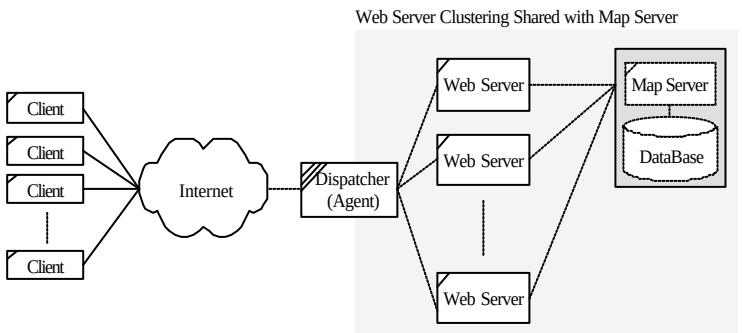


Figure 1. The system architecture of web server cluster shared with map server

Figure 2 shows the message processing in case client requests desired map in figure 1. When the client requests map, dispatcher selects the best-suited server based on load information at each server sides. Then, Web Server accepts desired map from Map Server and returns to client the result of processing.

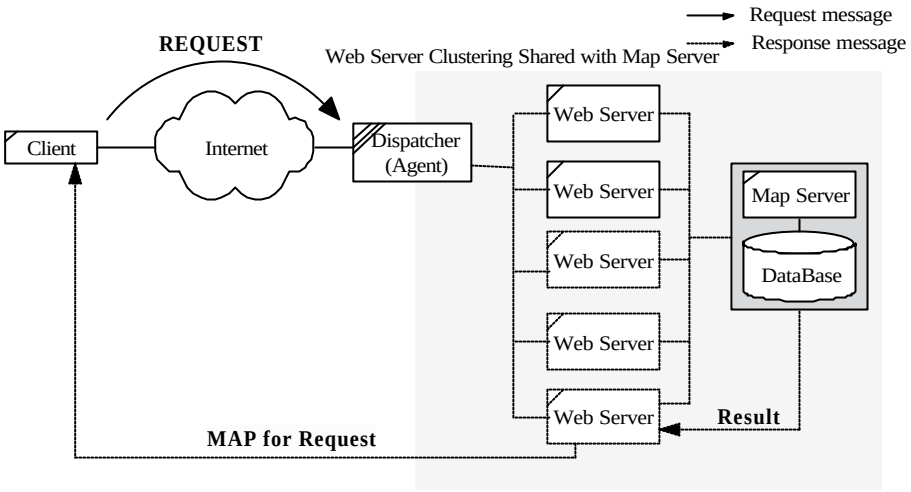


Figure 2. The overview of messages operation

Agent

This paper proposed the ideal agent over Web GIS. The agent proposed in this study are a new kind of agent that aims at prediction of the load and selection of the most proper Web Server in a Web Server cluster for Web GIS service.

In this section, the concept of agent to reduce the Internet traffic is discussed. It outlines the architecture of agent's technology and gives the performance for this architecture. The agent's policies proposed in this paper are based their decision on the estimated load rates.

Figure 3 summarizes that the additional software components needed for an efficient agent-based load balancing method. The agent on a dispatcher has an alarm monitor and a load monitor. These monitors track the feedback from servers to avoid assigning requests to an overloaded server. When the agent is in learning mode, the data is generated as the load information. In order to compute load on Web side, agent collects CPU Load, memory Load, Disk using, the connection between Map Server and Web Server from each Web server. This above information is sent to a dispatcher periodically from Web Server. Agent on dispatcher becomes aware of this load information delivered from server at alarm monitor and gathers them at load monitor.

As shown in figure 3, set of components in one Web Server cluster is consisted in HTTPD, Load Checking and Connection Counter. Besides the HTTPD, the proposed agent requires Load Checking and Connection Counter. The Load Checking tracks the server utilization of CPU Load, memory Load, Disk using. The Connection Counter estimates the number of connection between Map Server and Web Server. That information estimated from Load Checking and Connection Counter is periodically provided to agent.

The basic operation rule is that dispatcher selects sever using load information when a client requests map. The selected server will be the one that have probably the lowest load value. If the proper Web Server is selected, the

message is forward to server. If client's request message is requesting map, Web Server is accepting the result from Map Server then responding it to client.

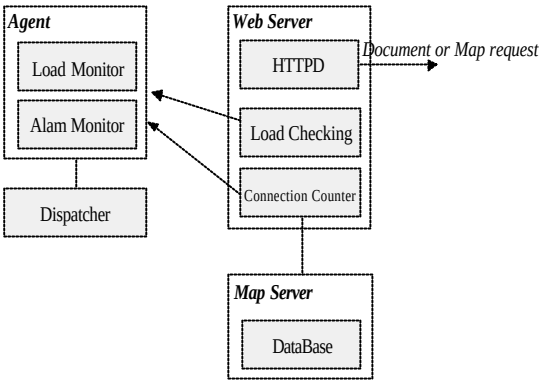


Figure 3. Action among Web Server and Map Server and Agent

The formula to compute load to select the best-suited server for its web services is following.

$$\text{Total\_Load} = \text{weight} \times (\text{cpu\_Load} + \text{memory\_Load} + \text{disk\_using} + \text{Map\_Server\_connection}) \quad (1)$$

In the above formula (1), CPU_load, memory_load, and disk_using refer to the estimation carried out using various Web Server information that is the utilization of each Web Server side. Map_Server_connection means the percent between Map server and each Web Server.

Here, Map_Server_connection is defined to divide total connection between a Web Server cluster and Map Server by number of connection between each Web Server and Map Server.

Here, the weight can be granted differently depending on each load. For the convenience in computing load, cpu_Load, memory_Load, disk_using, Map_server_connection are only considered in this paper. The sum of weight is 1($0 \le \text{weight} \le 1$). The formula to grant Weight differently depending on each load is following.

$$\text{Total\_Load} = W_{cpu} \times \text{cpu\_Load} + W_{mem} \times \text{mem\_Load} + W_{disk} \times \text{disk\_using} + W_{conn} \times \text{Map\_server\_connection} \quad (2)$$

$$(W_{cpu} + W_{mem} + W_{disk} + W_{conn} = 1)$$

Figure 4 shows the result of simulation that agent selects the server having the lowest load value based on total load information at the one time after agent learns load information.

In the figure 4, there is divided in two parts. The upper part presents that client's request goes to the server that is selected by agent on dispatcher and the lower shows the chart that results in computing load information acquired from formula (2). Here, the weight of cpu_load, memory_load, disk_using, Map Server_connection are defined in 1/4, 1/4, 0, 1/2, respectively.

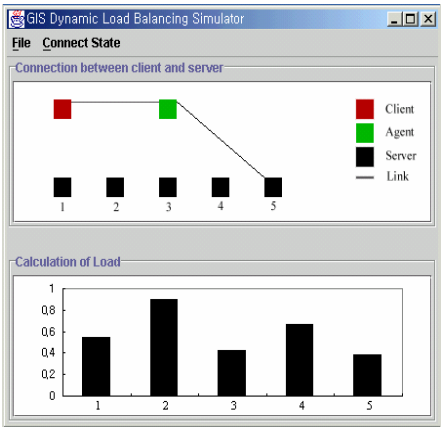


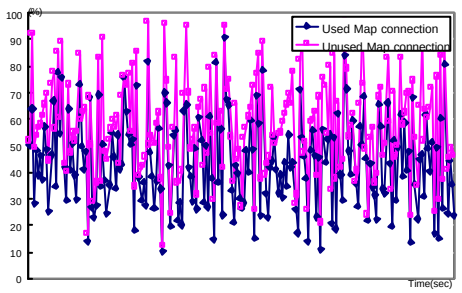
Figure 4. The result of agent simulation

EXPERIMENTAL RESULTS

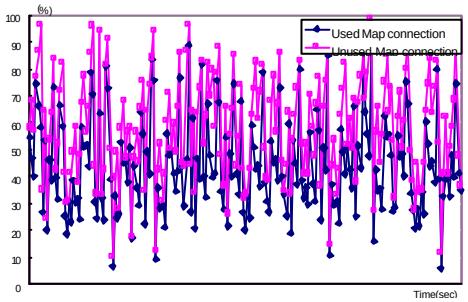
The main goal of this section is to investigate that our proposal has better impact by avoiding that any server becomes overloaded for Web-GIS services. The performance is evaluated through the maximum server utilization observed during the simulation run, especially comparing whether connection to the map server is considered.

The system was run for 12 minutes. Following figure 8 shows the comparison between total load considering the value of map server connection and total load without it. The computation of total load is based on formula (2). The axis of x and y coordinates indicate time (seconds) and the percent (%) of total load observed at every 3.6 seconds, respectively.

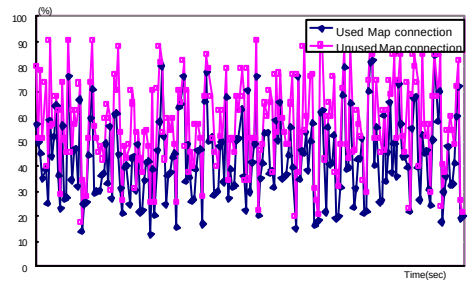
As the result, the individual trace graph of five servers in a Web Server cluster demonstrates that the agent, which has information about cpu_load, memory_load, disk_using, and especially Map_Server_connection, have generally lower traffic at the same time comparing agent, which doesn't consider load information about Map_Server_connection. Finally, the agent on dispatcher will select one proper server in Web server cluster, which have the lowest load, then keep the load balancing dynamically at server side.



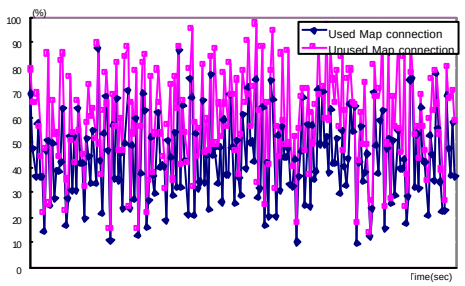
(a) Server 1



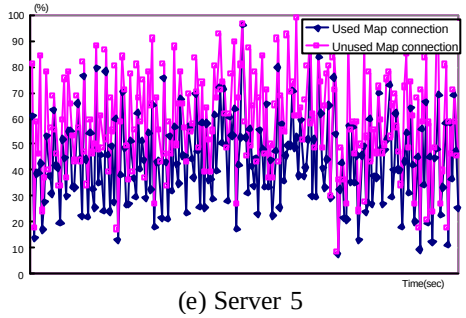
(b) Server 2



(c) Server 3



(d) Server 4



(e) Server 5

Figure 8. The comparison of traffic observed at five servers in Web Server cluster

CONCLUSIONS

The information system implemented over Internet has a big demand to present information on Web within seconds without bottleneck. In this study agent-based load balancing is designed and simulated considering Web Server cluster shared with Map Server. In this system architecture, agent watches load coming to each Web Server side and gathers it. Then, the agent is able to select the best-suited Web Server in the cluster by maintaining load balancing. Also, the experimental result to compare traffic observed at five servers in cluster show that our architecture considering map connection has better effect for Web-GIS services than previous normal Web-GIS architecture.

In the future, the work done here can be extended for implementation in larger networks with agent performing more useful and efficient tasks. Also, the dynamic load balancing should be considered in a large network system, which consists of several web server clusters shared with map server.

REFERENCES

Athanasios, K., Ziliaskopoulos, S., and Travis Waller, 2000, An Internet-based geographic information system that integrates data, models and users for transportation application. *Transportation Research Part C*, Vol. 8, pp. 427-444.

A. Bouguttaya, M.Ouzzani, B. Benatallah, L. Hendra, and J. Beard, 2000, Supporting Dynamic Interactions among Web-Based Information Sources. *IEEE Transactions on knowledge and data engineering*, Vol.12, No. 5. pp. 779-801.

Chris Gago, 1999, Scaleability, Availability andLoad-balancing for TCP/IP applications. *IBM Secure Way Network dispatcher Version 2.1, White Paper*.

G. Q. Huang, J. Huang, and K. L. Mak, 2000, Agent-based workflow management in collaborative product development on the Internet. *Computer-Aided Design*, Vol. 32, pp. 133-144.

J. P. Kimmance, M. P. Bradshaw, and H. H. Seetoh, 1999, Geographic Information System (GIS) application to construction and geotechnical data management on MRT construction projects in Singapore. *Tunnelling and Underground Space Technology*, Vol. 14, No. 4, pp. 469-479.

V. Cardellini, M. Colajanni, and Philip S.Yu, 1999, DNS Dispatching Algorithms with State Estimators for Scalable Web-Server Clusters. *Woeld Wide Web Journal Baltzer Science*, Vol. 2, No. 2, pp. 1-25.