Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp183-188)

Enhanced Dynamic Web Page Allocation using Fuzzy Neural Network

Y. K. LIU, L.M. CHENG, L.L.CHENG
Department of Electronic Engineering
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong SAR
HONG KONG

Abstract: Due to the limited cache size in each server, the traditional neural network techniques applied to improve the cache hit rate of scheduling algorithm in load-balancing web server cannot provide a good performance real web site because it cannot balance the server workload properly. Here, we propose a fuzzy neural network technique by feeding back the real-time system usage with an updating mapping rules based on different requested objects categorized into different servers groups with different cache size and according to their input frequency to enhance the cache hitting rate of scheduling, simulation result shows that the proposed technique keeps 92% to 99% cache hit rate and in parallel finely balances backend server resource usage.

Key-Words: Neural Network, Load-Balancing, Fuzzy, Competitive Learning, Caching, Clustering Techniques, Online Learning

1 Introduction

As web sites gain popularity and its traffic increases, single web server will no longer be able to handle the increased requests. Multiple servers must be used in web sites with heavy traffic load. This topology facilitates the sharing of information among servers with network storage such as distributed file system (DFS).

The deployment of cluster web server is quite complex as all the servers serve the requests of the same web site. The requests have to be distributed to each server depending on its current status.

Recently, the computational approach to artificial intelligence has undergone a significant evolution. Neural network is involved in the solutions of different practical problems such as face recognition [1] and control system [2]

The main objective of this paper is to study the server cache hit rate and distributing the incoming object requests according to the real-time servers' information using a fuzzy-neural network. The simulation results indicate that the server cache hit rate has significantly been improved and the backend server resource usage has been optimized.

2 Load-Balancing Algorithms in Web Service Provider

The common architecture used in a multiple server web site is shown in Fig.1. The requests from client are received by a load-balancer and redistributed to different servers [3].

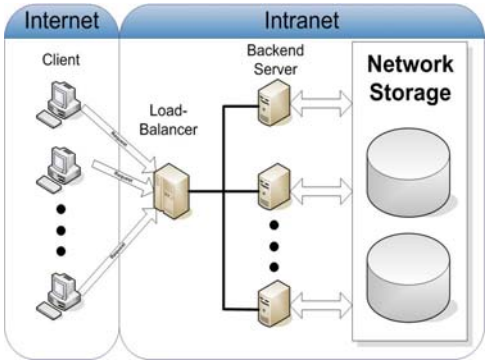

Fig.1: A common topology of web site with multiple servers. All the client requests will be collected and forwarded to the selected backend servers. And the server responses to the incoming request with the data stored in the network storage.

There are four different approaches to distribute the client requests to the servers – client-based, DNS-based, dispatcher-based and server-based. Dispatcher-based approaches can achieve fine-grained load-balancing to maximize the performance of systems [3]. The key of high performance in dispatcher-based approach is the distributing algorithm of the load-balancer. It centralizes the servers' information and gives the best solution for the incoming requests.

2.1 Neural Networks for Load-Balancing System

Neural Network (NN), which provides organizing, correlating and self-learning properties, is fulfilling the requirements of distributing algorithm in load-balancing system. NN can analyze the current system environment and give a suitable decision to the incoming requests.

The Secure Socket Layer (SSL) is a technology that creates a private data pipe in network with a secret session key. It is commonly used in the Internet application such as online bank transaction. However, session keys are expensive to be generated, so their lifetime is as long as about 100 seconds [4]. Thus, it is simple for a system to route the same client connection to a single load balancer server. The competitive learning method of a neural network is naturally for assigning a new input request to a server, and the learning rule strengthens the connection between the input and the server.

3 Conceptual Framework

The conceptual framework model balances the workload among servers and ensures high cache high rate at the same time. Although caching data at client side can improve the web site performance by reducing the servers' loading [5], in the real situation, there are many limitations such as capacity limit of the cache at clients and the reload problem in the web pages with dynamic content. To enhance the performance of the system, we focus on the caching data process at the server side and it is used to test the improvement of performance in web systems [4]. In a load-balancing system, client requests are distributed to a group of computers. To optimize the cache hit-rate in the servers, a Fuzzy-Neural Network technique is used.

The modified conceptual model is shown at Fig.2. The load-balancer consists two parts— Neural Network and Fuzzy Logic Controller. The Neural Network chooses the most suitable server to serve the incoming client request, so that the hit-rate and the system resources can be efficiently used. The resources usage in each server will be send to the load-balancer periodically, however, the measurement of usage level cannot be accepted directly by the Neural Network to modify its weight in order to enhance its performance. A Fuzzy Logic Controller is used to standardize the server resources usage and send the new weights to improve the Neural Network.

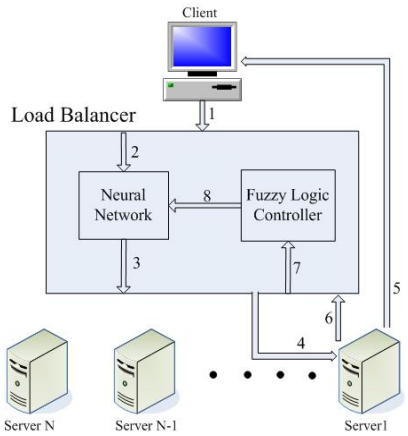

Fig.2: A conceptual model of the framework. 1: A client sends an object request to the Load-Balancer. 2: The request is passed to the Neural Network. 3: The Neural Network decides which server response to the request. 4: Forward the request. 5: The server responses to the client. 6. Each server sends their resources usage to the load-balancer periodically. 7: The received information is passed to the fuzzy logic controller. 8: The NN learns the distribution of system usage from the result of the fuzzy logic controller.

3.1 Structure of Framework

In the proposed framework, a one-layer neural network is used. There are P nodes in this layer, where each server S is assigned a neuron N. Q selected inputs in the input set of K size are fully meshed with each neuron. The weight w_{ij} connects input i and neuron j and represents the strength of relationship between them. Fig.3 illustrates this architecture.

3.2 Balance between Cache Hitting Rate and System Usage

In the learning rule, we add a load-balancing factor, which collects the knowledge from the system environment. By the "rich-get-richer" property in the competitive learning, the network will tend to forward the objects request to the server the same as last time. Therefore, a load-balancing factor has to be included in the neural network to balance the workload in the system.

In the V. V. Phoha algorithm [4], a weight averaging factor is added in the learning rule. The values in the weight vector of neurons are moving towards the average of it when the NN is training. However, it cannot balance the workload effectively because it only averages the probability of each server to receive requests, but does not put focus on unbalancing workload in a real site.

In our proposed algorithm, we add a load-balancing item which is related to the usage of the backend servers in the learning rule. Therefore, the NN can distribute the system workload appropriately by learning the information of servers' resources usage. To get the resource information of each backend server in the system, a Fuzzy Logic Controller is included in the load-balancer. It is used to sum up the usage of different resources such as processor, network and memory usage. Finally, the resulting value will be defuzzyified and send to NN. Fig.4 shows the structure of the Fuzzy Logic Controller.

4 Learning Rule of the NN

Under the new learning rule, NN groups the backend servers to handle different input requests and modifies the size of them according to the input frequency of each object type. At the same time, it collects the usage information from each server and

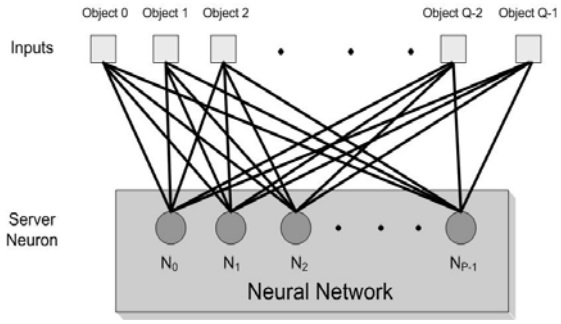

Fig. 3: The architecture of the Neural Network.

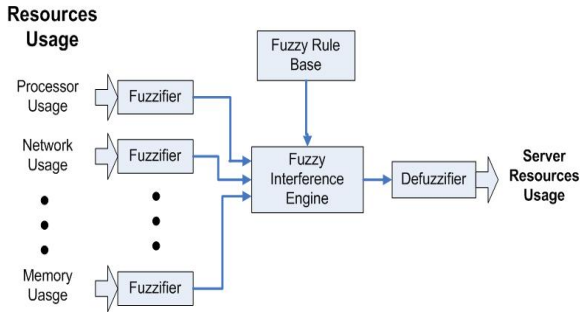

Fig. 4: The structure of the Fuzzy Logic Controller

rearranges the members of each group. We have three objectives:

- Maximize the cache hit rate of the backend servers. This can enhance the performance of the whole system by reducing the IO operations of the backend servers.
- Balance the resources usage in the system.
- Accommodate the sudden change in the network traffic pattern in the running time.

A request of object r from client is received by the load-balancer and it selects backend server N to follow this request.

The input of the NN is represented by:

$$R_i = \begin{cases} 1 & if \quad i = r \\ 0 & if \quad i \neq r \end{cases} \text{ where } i \in K \qquad (1)$$

The neuron with the highest output value will be chosen to handle the client request. So that:

$$\Omega\left(\sum_K R_i w_{iN}\right) = \max\left[\Omega\left(\sum_K R_i w_{ij}\right)\right] \qquad (2)$$

where $j \in [1, P]$ and

$$\Omega(v) = \frac{1}{1+e^{-\rho v}} \qquad (3)$$

is the output function

In each learning step, the weight of each neuron will be updated with Δw:

$$w[n] = w[n-1] + \Delta w \qquad (4)$$

where

$$\Delta w_j = \begin{cases} a_j + \Psi(b_j)(0.9 - (w_{ij} + a)) & if \quad b_j >= 0 \\ a_j + \Psi(b_j)(w_{ij} + a) & if \quad b_j < 0 \end{cases} \qquad (5)$$

where $j \in [1, P]$, $i \in K$

and

$$\Psi(v) = \frac{1-e^{-dv}}{1+e^{-dv}} \qquad (6)$$

Variable a_j is a competitive learning factor which reinforces the request forwarding memory. On the other hand, b_j is the load-balancing factor which provides the knowledge on the usage of the backend servers.

Equation of a:

$$a_j = \begin{cases} \eta H \cdot (R_i - w_{ij}) & if \quad i = r \\ 0 & if \quad i \neq r \end{cases} \qquad (7)$$

where

$$H = e^{-\frac{\varepsilon^2}{2\gamma^2}}$$

and

$$\gamma = \lambda e^{-tf_r} + \beta$$

and

ε is the ranking of the neuron output values,

f_r is the ratio of the request frequency of object r to all objects

.Equation of b:

$$b_j[n] = \alpha(U_{avg} - U_j) + K \cdot b[n-1] \qquad (8)$$

where

U_j is the resources usage of server j,

U_{avg} is the average resources usage of all servers in the system

The parameters η, γ, K and α determine the rate of learning and the weight of the load-balancing factor.

	Load-Balancer Computer	Server Computer Group 1	Server Computer Group 2
No. of Computers	1	12	4
Processor	Pentium 4 2.8C (2.8GHz)	Pentium 4 2.8C (2.8GHz)	Pentium 3 800E (800MHz)
Memory	512MB	512MB	256MB
NIC	100Mbps	100Mbps	100Mbps

Table 1: The equipments of the computers in the experiment.

Total Number of Servers	Number of Group 1 Computers	Number of Group 2 Computers
4	2	2
8	4	4
16	12	4

Table 2: The combination of server computers in the experiment.

5 Features of the algorithm

In the learning rule, we add a history term in the load-balancing part. It minimizes the memory clearance effect of the load-balancing factor and boosts the number of incoming requests to the backend servers, which have lower resources usage than the average of the servers in the system. Thus, the server performance can be enhanced.

One-time training mode is one of the limitations of traditional neural network. In this algorithm, online learning mode is used instead of one-time training. Online neural network can study the inputs and modify the weight at the running time. The NN can response to the change of resources usage and the request traffic pattern to modify the weight of the network [7]. Thus, the NN can provide the best solution to the system at anytime.

6 Experimental Results

To incorporate the characteristics of Web traffic $P[X > x] \sim x^{-\gamma}$ as $x \to \infty$ for $0 < \gamma < 2$. Pareto distribution is used in our simulation:

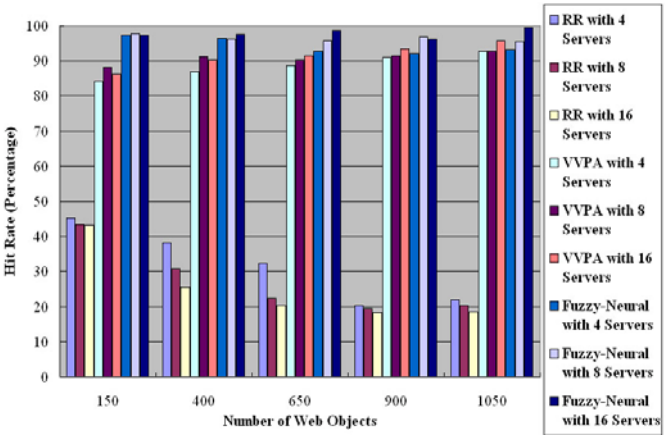

Fig.6: Server cache hit rate of various scheduling algorithms are shown. The proposed algorithm (Fuzzy-Neural) gives the highest cache hit rate which is between 92% and 99%. And the next highest is V.V. Phoha's Algorithm (VVPA) which is about 90%.

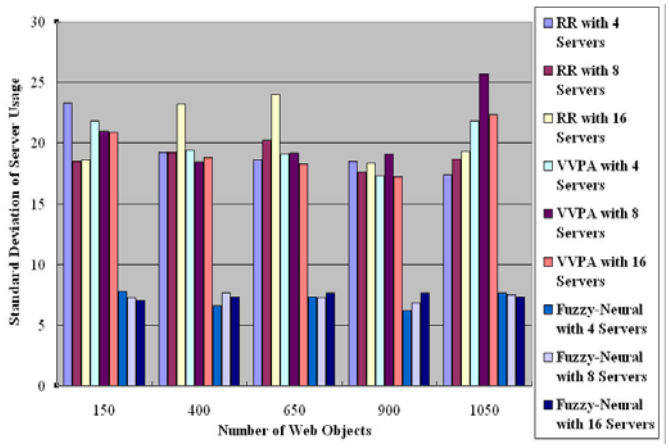

Fig.7: The performance of load-balancing is inversely proportional to the standard deviation of the resources usage. The proposed algorithm (Fuzzy-Neural) keeps the value of standard deviation at a low level.

$$p(x) = \gamma k^{\gamma} x^{-\gamma-1} \qquad (9)$$

where $\gamma = 0.9$ and $k = 0.1$

The simulation environment consists of 17PCs 2.8GHz P4 at Digital System Laboratory in City University of Hong Kong. Details are listed in Table 1 and Table 2. Load-balancer and client program are written in C#.net. The client request generator and the neural network are built on the

load-balancer program.

In the simulation, the performance of round-robin (RR) and V. V. Phoha's Algorithm (VVPA) [4] are compared.

The charts in Fig.6 and Fig.7 show the comparison of the cache hit rate and the standard deviation of the servers' usage in the system. The standard deviation of usage is inversely proportional to the performance of the load-balancing function in each scheme.

Fig. 6 shows that both the Fuzzy-Neural and VVPA give high cache hit rates in the experiment. And, the cache hit rate in Fuzzy-Neural ranges between 92% and 99%. On the other hand, the hit rate of the RR is less than 44% and decreases with the number of web objects due to limitation of the server cache size.

In the Fig. 7, the standard deviation of servers' usage is shown and represents the performance of load-balancing. In the chart, the value of standard deviation of VVPA and RR ranges between 18 and 20, whereas the Fuzzy-Neural algorithm gives the value around 7. It means Fuzzy-Neural algorithm can effectively balance the incoming client requests according to the resources in each backend server.

In the experiment, the proposed algorithm can provide a high cache hit rate and finely adjusted load in each server according to their capability.

7 Conclusion

A load-balancing scheduling algorithm with fuzzy-neural network is proposed. Fuzzy–neural network using Kohonen's algorithm can provide a high sever cache hit rate. We also propose a new learning rule which study the distribution of servers' resources usage. A Fuzzy Controller analyzes the servers' resources usage and the NN learns the usage level of each backend server to balance the system's loading. Online learning is used in the NN to modify the forwarding pattern according to the real-time system's usage information. A promising result is produced in the simulation experiment under the environment containing two groups of servers with different processing power.

References:

[1] H. A. Rowley, S. Baluja, T. Kanade, "Neural network-based face detection", IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1): 23-38 Jan 1998.

[2] K. J. Hunt, D. Sbarvaro, R. Zbikowski, "Neural Networks for Control-Systems - A Survey", Automatica 28 (6): 1083-1112 Nov 1992.

[3] V. C. Cardellini, M. Colajanni, P. S. Yu, "Dynamic load balancing on Web-server systems", IEEE Internet Computing 3 (3): 28-+ May-Jun 1999.

[4] V. V. Phoha, S. S. Iyengar, R. Kannan, "Faster Web page allocation with neural networks", IEEE Internet Computing 6 (6): 18-26 Nov-Dec 2002.

[5] C. Aggarwai, J. L. Wolf, P. S. Yu, "Caching on the World Wide Web", IEEE Transactions on Knowledge and Data Engineering 11 (1): 94-107 Jan-Feb 1999.

[6] T. Kohonen, "The Self-Organizing Map", Proceedings of the IEEE 78 (9): 1464-1480 Sep 1990

[7] J. Tanomaru, S. Omatu, "Process-Control by Online Trained Neural Controllers", IEEE Transactions on Industrial Electronics 39 (6): 511-521 Dec 1992