International Journal of Innovative Computing, Information and Control
Volume **5**, Number **6**, June **2009**

ICIC International ©2009 ISSN 1349-4198
pp. 1689–1698

DESIGNING THE BURST WEB LOAD BALANCING ARCHITECTURE USING FUZZY DECISION

Jian-Bo Chen and Tsang-Long Pao
Department of Computer Science and Engineering
Tatung University
Taipei, Taiwan
jbchen@mcu.edu.tw; tlpao@ttu.edu.tw

Received December 2007; revised March 2008

Abstract. *In order to reduce the cost and increase the performance of load balancing systems when dealing with burst traffic, we propose the concept of service-on-demand servers, such as bringing DNS servers or Mail servers to join the load balancing system during the burst traffic period. Due to the different capacities that these servers have, we propose using a fuzzy decision mechanism to dispatch the client requests to the appropriate backend server. The CPU idle percentage, the memory available, and the number of connections for each backend servers are the input parameters of our fuzzy decision mechanism. Based on these input features, the fuzzification, rule evaluation, and defuzzification are the three steps that can be used to calculate the final crisp value, and then the most appropriate backend server can be determined. The experimental results show that using service-on-demand servers can achieve acceptable performance and eliminate the need of using another dedicated server. In addition, the fuzzy decision mechanism can also achieve higher performance than other dispatching algorithms.*

Keywords: Fuzzy decision, Load balancing, Burst traffic, Service-on-demand

1. **Introduction.** Due to the growth of the World Wide Web, the traffic of popular web sites has grown far beyond the capacity of a single web server. Most popular web sites adopt a distributed or parallel architecture to alleviate the load for the single server [5]. These sites can provide higher performance for a large number of client requests [4]. Although the load balancing architecture consists of a number of backend servers, they act as a single unit. User transparency is implemented to allow clients to issue the requests to the central unit without knowing the load balancing architecture the web site has implemented. Meanwhile, clients do not need to make any configuration modifications when they connect to the load balancing systems.

There are various methods used to build a load balancing system. These methods include the dispatcher-based approach [7,13,19], the parallel filtering approach [16], the Round Robin Domain Name Server(RR-DNS) approach [2,18,20], and so on. However, in parallel filtering and RR-DNS schemes, the exact workload of each individual backend server in the system may not take the factors into consideration which might lead to an unbalanced load situation. For the RR-DNS, another major problem is the DNS query result caching in the intermediate DNS server and the client itself [1]. In this case, requests from hosts in the same domain may all be served by the same backend server and may drive that server into an overload state. So, the best solution for the load balancing architecture is the dispatcher-based approach.

In the dispatcher-based approach, the central unit is the dispatcher which is responsible for dispatching the client requests to the backend servers. In the server-state dispatching architecture [9], the dispatcher must collect the status of all the backend servers and