Chapter XI
Object Grouping and Replication on a Distributed Web Server System

Amjad Mahmood  
*University of Bahrain, Kingdom of Bahrain*

Taher S. K. Homeed  
*University of Bahrain, Kingdom of Bahrain*

**ABSTRACT**

Object replication is a well-known technique to improve performance of a distributed Web server system. This paper first presents an algorithm to group correlated Web objects that are most likely to be requested by a given client in a single session so that they can be replicated together, preferably, on the same server. A centralized object replication algorithm is then proposed to replicate the object groups to a cluster of Web-server system in order to minimize the user perceived latency subject to certain constraints. Due to dynamic nature of the Web contents and users’ access patterns, a distributed object replication algorithm is also proposed where each site locally replicates the object groups based on the local access patterns. The performance of the proposed algorithms is compared with three well-known algorithms and the results are reported. The results demonstrate the superiority of the proposed algorithms.

**INTRODUCTION**

The phenomenal growth in the World Wide Web (Web) has brought about a huge increase in the traffic to popular Websites. This traffic occasionally reaches the limits of the sites’ capacity, causing servers to be overloaded (Chen, Mohapatra, & Chen, 2001). As a result, end users either experience a poor response time or denial of a service (time-out error) while accessing these sites.
Since these sites have a competitive motivation to offer better service to their clients, the system administrators are constantly faced with the need to scale up the site capacity. There are generally two different approaches to achieving this (Zhuo, Wang, & Lau, 2003). The first approach, generally referred to as hardware scale-up, is the use of powerful servers with advanced hardware support and optimized server software. While hardware scale-up relieves short-term pressure, it is neither a cost effective nor a long-term solution, considering the steep growth in clients’ demand curve. Therefore, the issue of scalability and performance may persist with ever increasing user demand.

The second approach, which is more flexible and sustainable, is to use a distributed Web-server system (DWS). A DWS is not only cost effective and more robust against hardware failure, but it is also easily scalable to meet increased traffic by adding additional servers when required. In such systems, an object (a Web page, a file, etc.) is requested from various geographically distributed clients. As the DWS spreads over a metropolitan area network (MAN) or wide area network (WAN), movement of documents between server nodes is an expensive operation (Zhuo, Wang, & Lau, 2003). Maintaining multiple copies of objects at various locations in a DWS is an approach for improving system performance, such as latency, throughput, availability, hop counts, link cost, and delay (Kalpakis, Dasgupta, & Wolfson, 2001; Zhuo, Wang, & Lau, 2003).

There are two techniques used in maintaining multiple copies of an object: caching and replication. In Web caching, a copy of an object is temporarily stored at a site that accesses the object. The intermediate sites and proxies also may cache an object when it passes through them en route to its destination site. The objective of Web caching is to reduce network latency and traffic by storing commonly requested documents as close to the clients as possible. Since Web caching is not based on users’ access patterns, the maximum cache hit ratio achievable by any caching algorithm is bounded under 40-50% (Abrams, Standridge, Abdulla, Williams, & Fox, 1995). In addition, cached data have a time to live (TTL), after which the requests are brought back to the original site. Object replication, on the other hand, stores copies of an object at predetermined locations to achieve a defined performance level. The number of replica to be created and their locations are determined by users’ access patterns. Therefore, the number of replicas and their locations may change in a well-controlled fashion in response to changes in the access patterns.

In most existing DWS, each server keeps the entire set of Web documents/objects managed by the system. Incoming requests are distributed to the Web server nodes via domain name system (DNS) servers or request dispatchers (Cardelino, Colajanni, & Yu, 1999; Colajanni & Yu, 1988; Kwan, MeGrath, & Reed, 1995; Baker & Moon, 1999). Although such systems are simple to implement, they could easily result in uneven load among the server nodes, due to caching of IP addresses on the client side. To achieve better load balancing as well as to avoid disk wastage, one can replicate part of the documents on multiple server nodes, and requests can be distributed to achieve better performance (Li & Moon, 2001; Karlsson & Karamanolis, 2004; Riska, Sun, Smimi, & Ciardo, 2002). Choosing the right number of replicas and their locations is a nontrivial and nonintuitive exercise. It has been shown that deciding how many replicas to create and where to place them to meet a performance goal is an NP-hard problem (Karlsson & Karamanolis, 2004; Tenzakhti, Day, & Olud-Khaoua, 2004). Therefore, all the replica placement approaches proposed in the literature are heuristics that are designed for certain systems and work loads.

This article proposes two algorithms for replicating objects in a DWS environment. The first algorithm is centralized in the sense that a central site/server determines the allocation of objects. The distributed algorithm (DA), however, does