Large-Scale ASP Replication of Database-Driven Portals

Christopher B. Mayer
Air Force Institute of Technology, USA

K. Selçuk Candan
Arizona State University, USA

INTRODUCTION

Web portal applications that dynamically generate results in response to user requests are more popular than ever. Such portal applications usually consist of a business logic component and a very large database, or databases that hold the portal's content. Despite efforts to speed up response generation, ever-rising user demand means that replication of a portal's logic *and* database will be needed at some point as other methods to keep up with demand (faster databases and content caching for example) have limits.

This article explains issues surrounding the replication of a single full-scale database-driven portal application. In addition to the issues of replicating a single, unsophisticated application, it also anticipates a future in which portal applications offer multiple levels of service to users, and large application service providers (ASPs) host and replicate many portal applications on networks of servers. An ASP must replicate complex portal applications in order to satisfy user demand while minimizing operating costs. ASPs will need mature tools for making replication decisions and deploying and otherwise managing their replication system. To that end, this article highlights two prototype software packages, ACDN and DATE/DASIM, that address some aspects of replica management by ASPs. Using the two prototypes as a starting point and recalling single portal replication issues, a set of features for a mature ASP replication management systems are proposed.

DATABASE APPLICATIONS AND REPLICATION ISSUES

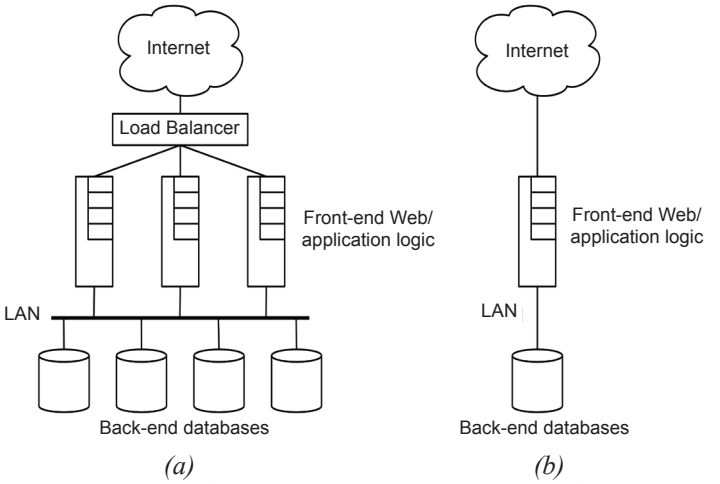
More and more companies and organizations are discovering the benefits of providing services over the Internet through portal applications. In order to provide a more profitable, responsive, and flexible user experience, these portal applications generate responses dynamically and provide a customizable experience for each user. That is, the content they provide to the user is generated on demand in response to user requests. These applications usually have two components: (1) front-end business logic that interacts with the user and provides the portal's look and feel and (2) a back-end database or databases containing the portal's true content (Candan & Li, 2002a; Li, Hsiung, Po, Hino, Candan, & Agrawal, 2004b; Vallamsetty, Kant, & Mohapatra, 2002). Based on user requests, the front-end logic queries the back-end databases, obtains needed content, and then uses that content to build a page (or other result), which is returned to the user. Figure 1(a) shows a diagram of a complex database-driven portal application while Figure 1(b) shows a simplified version containing the main components: the application logic and backend database. To emphasize the importance of the back-end database, such portal applications will be henceforth called *database-driven applications*, or DAs.

In order to meet user expectations and availability requirements, a DA must be able to return results quickly. Historically, this has been achieved by moving fragments of a DA's logic and its back-end database closer to end-users at the "Internet's edge" by caching raw data, caching results, or using fragmented page design (Choi & Luo, 2004; Datta, Dutta, Thomas, Vandermeer, & Ramamritham, 2004; Huang, Sebastine, & Abdelzaher, 2004; Li, Po, Hsiung, Candan, & Agrawal, 2003; Luo et al., 2002). These methods are generally compatible with dynamic Web page markup languages such as Java Server Pages and Edge Side Includes. However, such improvements can only provide so much relief. They may be attractive for some small or low-demand applications, but are not, in the authors' opinion, the ultimate solution for complex high-demand applications with large amounts of supporting data. Instead, full replication of the application and a significant part of its backend database will be needed in some cases.

Although it has many positive aspects, full replication has at least four challenges (in the authors' opinion) as outlined next.

- **Application Masters and Replica Slaves:** An emerging trend in DA architecture and replication is the use of master and slave versions of the portal application (Li, Altinas, & Kantarcioglu, 2004a). In this scheme,

Figure 1. Multi-tiered portal application architectures. In (a) a complex arrangement of front-end servers is connected to a farm of database servers on the back-end. In (b) the arrangement is simplified to just the essentials.



the DA is viewed as an aggregate of all its instances, be they a master or a slave (replica). In operation, there is a single master and zero or more slaves. Both masters and slaves contain a database component and can respond to user requests. However, only the master can handle database updates. When a slave needs to update the database, the slave contacts the master, which processes the update and propagates changes to the slaves. Oracle's Database Cache and IBM/DB2 (Luo et al., 2002) support master/slave replication.

- **Large Databases:** In some cases, a DA's database may be so large (multiple gigabytes or more) that replicas cannot be rapidly established in response to changing demand.
- **Keeping Replicas Fresh:** Database replicas have to be regularly updated so that their content is timely or *fresh*. Assigning a DA replica to a server induces a continuous update load on the server's database component due to the frequent updates required to maintain the replica's service quality. In general, a higher quality of service requires more frequent synchronization. Update load is parasitic as it reduces the replica's capacity for handling end-user requests and prohibits creating more replicas than demand warrants. Understandably, update load mitigation has been the subject of much research (Candan, Agrawal, Li, Po, & Hsiung, 2002b; Candan et al., 2002a; Li et al., 2004b; Carney, Lee, & Zdonik, 2003; Majumdar, Ramamritham, Banavar, & Moudgalya, 2004; Olston & Widom, 2002).
- **Database Load and Response Times:** Response times are a prime concern, especially for e-commerce applications, since poor response times translate into unhappy customers and lost revenue (King, 2003; Labrinidis & Roussopoulos, 2003; Vallamsetty et al., 2002). As has been repeatedly observed, DA response times depend greatly on database load, and not necessarily on the placement of replicas close to users or network delays (Candan et al., 2002a; Datta et al., 2004; Labrinidis et al., 2003; Vallamsetty et al., 2002). The database load of a DA replica has two components: request load and update load. Request load results from queries stemming from user requests. Synchronizing a replica's slave database with its master's database causes update load. Generally, if a replica's aggregate update and request loads do not overload the database, then response times will be fine.

APPLICATION SERVICE PROVIDERS AND MULTI-QUALITY APPLICATIONS

Although a DA is a great asset for its owners, there are several drawbacks, especially when replication is needed. Chief among them is the monetary expense of maintaining enough capacity (servers and bandwidth) to handle demand surges. To help alleviate this problem, a new entity called the application service provider (ASP) has emerged. ASPs like Akamai and ASP-One specialize in hosting database-driven applications on behalf of owners and maintain a large heterogeneous pool of servers for that purpose. The ASP provides the expertise, bandwidth, processing capacity, and global presence that few portal owners could afford on their own.

Marketplace competitiveness means that an ASP has to both satisfy the portal owners and keep costs low. Owner satisfaction primarily means having enough serving capacity on hand to meet the demand generated by end-users. In