**Ref: F219**

**Words: 3512**

# Large-scale ASP Replication of Database-driven Portals

Christopher B. Mayer
Dept. of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433

K. Selçuk Candan
Computer Science and Engineering Dept.
Arizona State University
Tempe, AZ 85287

# Large-scale ASP Replication of Database-Driven Portals

Christopher B. Mayer
Dept. of Electrical and Computer Engineering
Air Force Institute of Technology, USA

K. Selçuk Candan
Computer Science and Engineering Dept.
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 server's. 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 system 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 and services 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, 2002; 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 Fig. 1(b) shows a simplified version containing the main components: the application logic and backend database. As will be explained later, the back-end database is important. To emphasize that importance such portal applications will be henceforth called **database-driven applications**, or DAs.

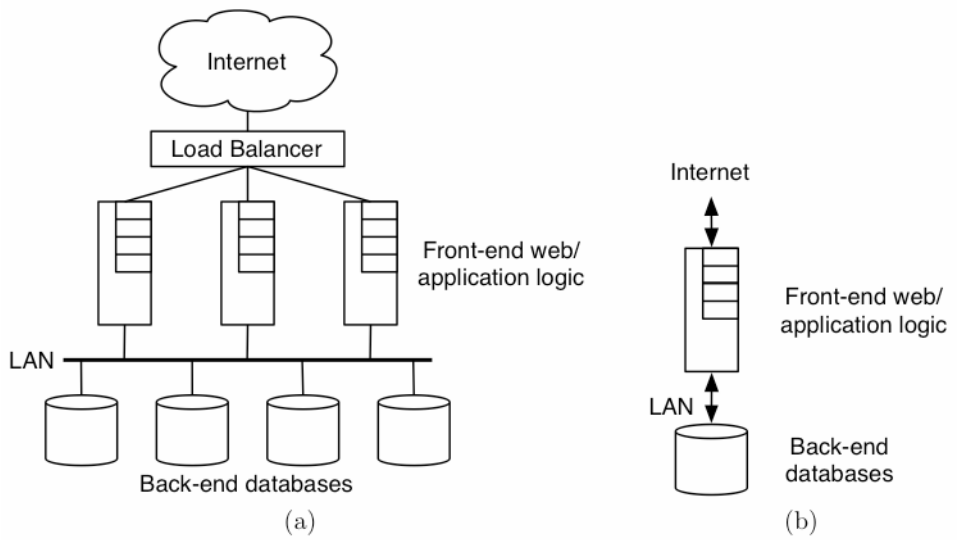


Figure 1. Multi-tiered portal application architectures. In (a) a complex arrangement of front-end servers is connected to a farm of database servers on the back-end. In (b) the arrangement is simplified to just the essentials.

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, processed results, fragmented page design, or all three (Choi & Luo, 2004; Datta, Dutta, Thomas, Vandermeer, & Ramamritham, 2004; Huang, Sebastine, & Abdelzaher, 2004; Li, Po, Hsiung, Candan, & Agrawal, 2003; Luo, Krishnamurthy, Mohan, Pirahesh, Woo, Lindsay, & Naughton, 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 DA 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 three challenges (in the authors' opinion) as outlined below.

- *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 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, 2002; Candan & Li, 2002; Li et al., 2004b; Carney, Lee, & Zdonik, 2003; Majumdar, Ramamritham, Banavar, & Moudgalya, 2004; Olston & Widom, 2002).

- *Database Load and Response Times.* Response times are 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 & Li, 2002; Datta et al., 2004; Labrinidis & Roussopoulos, 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 doing so, the ASP must deal with all the replication issues mentioned in the previous section and some new ones which are discussed next.

A naïve, but infeasible, way for the ASP to meet demand is to pre-position replicas that can be "turned on" as needed. A more reasonable approach is to create and destroy replicas as demand fluctuates. However, since a DA's database can be extremely large, replicas cannot be rapidly established. ASPs will probably have to do some pre-positioning while also forecasting demand so that replicas can be deployed by the time they are needed.

While many portals come in just one version, some may offer several distinct freshness/quality levels in order to meet the needs of different types of users (Bright & Raschid, 2002; Cherniack, Galvez, Franklin, & Zdonik, 2003). Pay-for-content sites such as brokerages or news services are good examples. In each case, users can be grouped into at least two categories: high-quality and low-quality. High-quality users expect either very fresh (timely) content or premium content. Low-quality users, on the other hand, are pleased with default or moderately-fresh content. Regardless of type, users are satisfied only when their quality expectations are met. Figure 2 illustrates this concept.

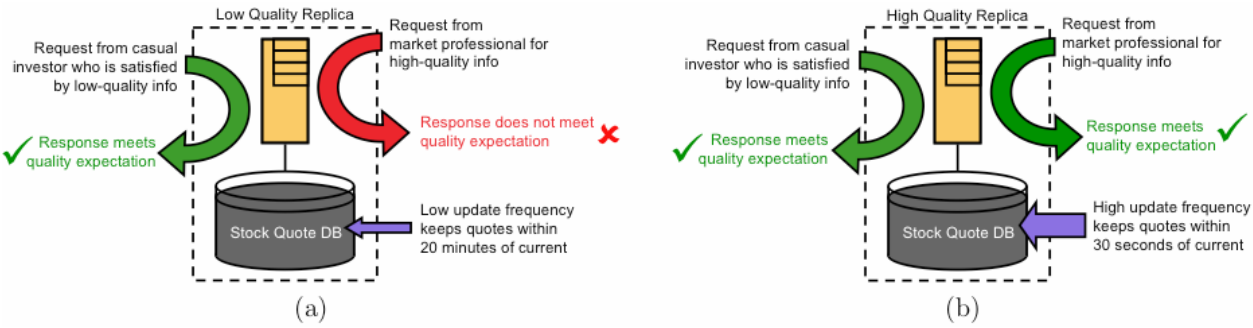


Figure 2. An illustration of low-quality and high-quality replicas. The low-quality replica in (a) can only satisfy users who are happy with older data. The high-quality replica in (b) can service both low- and high-quality users.

Depending on demand for each quality level, not all replicas may need to operate at the highest possible quality. By wisely choosing replica quality levels, an ASP can minimize update overhead. In

order to stay competitive, an ASP replicating quality-differentiated DAs must meet the five requirements listed in Table 1. Although seemingly simple, meeting these five goals has been proven to be NP-hard, even when all applications are single-quality (Mayer, 2005).

Table 1. **The five performance requirements of an Application Service Provider.**

| |
|---|
| 1. Provide enough serving capacity to handle all request load for the hosted applications. Capacity includes request load and the update load required to keep each portal replica fresh. |
| 2. Make replication decisions and deploy and dismantle portal replicas “on-line" (without disrupting existing services). |
| 3. Minimize operating overheads such as network bandwidth, number of servers used, and update load. Note that minimizing update load helps minimize bandwidth and number of servers. |
| 4. Minimize replica movement and service disruptions. |
| 5. Ensure that user demand for each portal application is satisfied at or above the quality level requested. |

The ASP’s replication problem bears a resemblance to other Internet data replication problems seen over the years (Cidon, Kutten, & Soffer, 2001; Kangasharju, Roberts, & Ross, 2002; Rabinovich, Rabinovich, Rajaraman, & Aggarwal, 1999; Wolfson, Jajodia, & Huang, 1997), yet differ from the ASP’s problem in a number of ways. For one, past work typically focused on small chunks of data that can be moved rapidly. Secondly, the data changes rather slowly, resulting in trivial update loads at the replicas. Thus, in the authors’ opinions, the ASP’s replication problem (Table 1) is different enough to warrant novel solutions and frameworks. The ultimate goal would be a replication system which can solve the ASP’s replication problem and manage the deployment of replicas. Requirements for such a system are examined next.

## TOWARDS MATURE DA REPLICATION

Although web technology continues to improve, a mature system for DA management by ASPs has not yet appeared. This section speculates on features that such a system, probably an integrated suite of software, would have. However, before jumping right to the features, two prototype DA management systems, Active CDN and DATE/DASIM, are introduced. Taking cues from these two systems and from topics mentioned elsewhere in this article, a list of features is then extrapolated.

**Active CDN.** ACDN (Active CDN) is a platform for adaptively creating and relocating web applications (Rabinovich, Xiao, & Aggarwal, 2003). ACDN consists of one central replicator and a number of ACDN servers, each of which contains a local replicator. The central replicator tracks the location of DA replicas and helps the local replicators make their replication decisions. Replication decisions (create or delete a replica) are based on server load, bandwidth (load) consumed in deploying and freshening replicas, and use high-load and low-load watermarks as trigger conditions. To facilitate deployment, each DA managed by ACDN is described in a metafile which lists the DA’s files and contains instructions for installing the application. ACDN attempts to place them close to sources of demand. Once placed, ACDN attempts to load-balance servers by routing requests to replicas based on proximity and replica load.

**DATE and DASIM.** The DA AssignmenT Engine (DATE) and the Database Application System Implementer and Manager (DASIM) are two cooperating software systems for ASP DA replication management (Mayer, 2005). DATE considers request load and update load information for each quality of each DA hosted by an ASP. DATE is able to produce low-update overhead assignments that can be implemented “on-line”.

DASIM manages the deployment of real database-driven web applications that adhere to the master/slave database replication scheme. DASIM has an open, flexible script-based design that can be easily changed to accommodate a wide range of DA implementation technologies. DASIM can link with DATE in order to implement replication decisions output by the latter. DASIM’s central

controller application, the System Manager Console (SMC), communicates with daemon applications called System Manager Remotes (SMRs) located on each replica server (Fig. 3).

**Requirements for Robust Replication.** Even though ACDN and DATE/DASIM are prototypes, they represent an important first step toward full DA replication in an ASP-like environment. While the prototypes have some admirable qualities, they fall short in some respects. Below, both prototypes and topics mentioned throughout the article have been used to create a list of requirements for mature portal replication.

- *Replication Decision Making.* Replication decisions must be made somehow, somewhere. Decision making in ACDN is mostly distributed, but does have a centralized component. Conversely, DATE/DASIM is highly centralized. Centralization may be the best option for many ASPs since it can uncover savings that might otherwise escape notice.
- *Operating Cost Minimization.* DATE was designed to minimize the cost of freshening replica databases. ACDN does not minimize any operating costs. However, ACDN's goal of placing replicas close to sources of demand can help reduce network traffic and, hence, indirectly, operating costs.
- *Accurate Application Load Reporting.* ACDN's servers are equipped with a load reporting element. DATE/DASIM does not currently have this feature. The ability to accurately measure and report server load, especially on an application-by-application basis, is essential.

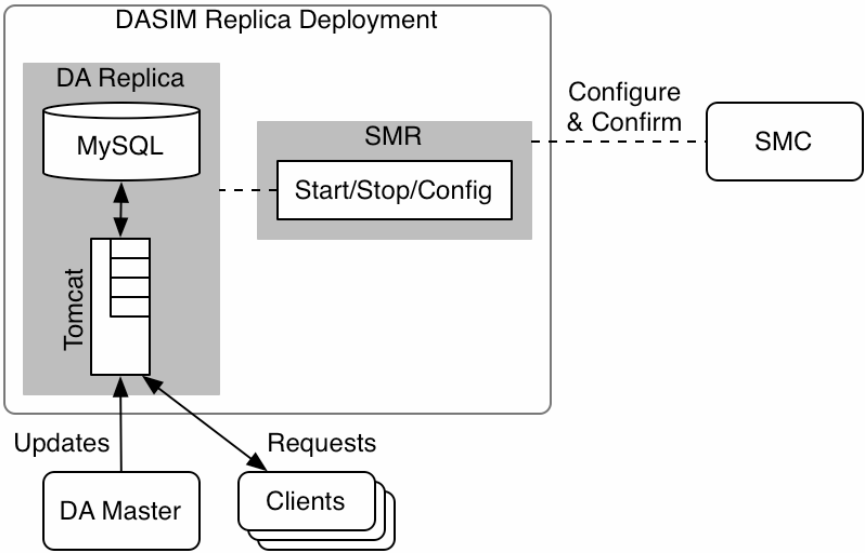


Figure 3. A DA Replica Deployed in DASIM. Solid lines show information flow required by the DA replica. Dashed lines show the flow of information needed to manage the replica and its components.

- *Replication Decision Cycles and Thresholds.* ACDN relies on asynchronous distributed replication decisions based on low- and high-watermarks and considers the time needed to deploy new replicas. DATE/DASIM's centralized approach is conducted in alternating assignment generation and implementation phases. To speed up assignment implementation, DASIM does much of its work in parallel. DATE uses neither watermarks nor any kind of load prediction/reaction scheme at present. Of all the items listed here, selecting a replication decision frequency and triggering events are probably the topics most in need of further evaluation and research.
- *Active Load Balancing and Request Redirection.* ACDN includes a Domain Name Server (DNS) load-balancer/request redirector. ACDN's explicit use of a load balancer makes sense given its preference for placing replicas close to sources of demand. Meanwhile, DATE/DASIM recognizes the need for request redirection, but assumes its presence.
- *Streamlined Replica Deployment and Deletion.* Both ACDN and DASIM make use of application descriptions and script files to streamline the deployment and deletion of replicas. Still, given the wide range of DA technologies and designs, it is doubtful that either ACDN or DASIM are fully

up to the task. What would most ease replica deployment at this point is increased DA uniformity, both of technology and design.

- *Minimal Replica Movement.* DATE addresses burdensome, but necessary, replica movement by explicitly minimizing it when generating replica assignments. Experiments in (Mayer, 2005) show DATE to be quite effective in this regard. ACDN realizes the problems surrounding replica creation and specifically accounts for moving large amounts of data. The ACDN algorithms described in (Rabinovich et al., 2003) appear to limit replica movement, although no direct supporting evidence is provided. In either case, replica movement is troublesome and must be limited by the system somehow.
- *Ability to Handle Quality-differentiated Applications.* Whereas DATE/DASIM pays particular attention to applications offering multiple freshness quality levels and can reduce operating overheads for them, ACDN does not consider such applications at all.

## SUMMARY

This article explained issues surrounding the replication of a single full-scale database-driven portal application: master-slave architecture, the burden of keeping replicas fresh, and the connection between database load and response times. Anticipating a future in which portal applications offer multiple levels of service and large Application Service Providers (ASPs) host and replicate many portal applications on networks of servers, the replication issues and two prototype replica management systems were used to extrapolate features for a mature ASP replica management system.

## REFERENCES


Bright, L., & Raschid, L. (2002). Using latency-recency profiles for data delivery on the web. *Very Large Data Bases (VLDB)* (pp. 550–561).

Candan, K. S., Agrawal, D., Li, W. S., Po, O., & Hsiung, W. (2002). View invalidation for dynamic content caching in multitiered architectures. *VLDB* (pp. 562–573).

Candan, K. S., & Li, W. (2002). *Architectural issues of web-enabled electronic business*, Chap. Integration of Database and Internet Technologies for Scalable End-to-end E-commerce Systems, (pp. 84–112). Idea Group Publishing.

Carney, D., Lee, S., & Zdonik, S. B. (2003). Scalable application-aware data freshening. *International Conference on Data Engineering (ICDE)* (pp. 481–492).

Cherniack, M., Galvez, E. F., Franklin, M. J., & Zdonik, S. (2003). Profile-driven cache management. *ICDE* (pp. 645–656).

Choi, C. Y., & Luo, Q. (2004). Template-based runtime invalidation for database-generated web contents. *Asia Pacific Web Conference (APWeb)* (pp. 755–764).

Cidon, I., Kutten, S., & Soffer, R. (2001). Optimal allocation of electronic content. *IEEE Conference on Computer Communications (INFOCOM)* (pp. 1773–1780).

Datta, A., Dutta, K., Thomas, H. M., Vandermeer, D. E., & Ramamritham, K. (2004). Proxy-based acceleration of dynamically generated content on the world wide web: An approach and implementation. *Transactions on Database Systems (TODS)*, *29*(2), 403–443.

Huang, C., Sebastine, S., & Abdelzaher, T. (2004). An architecture for real-time active content distribution. *Euromicro Conference on Real-Time Systems* (pp. 271–280).

Kangasharju, J., Roberts, J., & Ross, K. W. (2002). Object replication strategies in content distribution networks. *Computer Communications 25*(4), 376–383.

King, A. B. (2003). *Speed Up Your Site: Web Site Optimization* (1st ed.). Berkeley, CA: New Riders Press.

Labrinidis, A., & Roussopoulos, N. (2003). Balancing performance and data freshness in web database servers. *VLDB* (pp. 393–404).

Li, W.-S., Altinas, K., & Kantarcioglu, M. (2004a). On demand synchronization and load distribution for database grid-based web applications. *Data and Knowledge Engineering (DKE)*, *51*(3), 295–323.

Li, W.-S., Hsiung, W.-P., Po, O., Hino, K., Candan, K. S., & Agrawal, D. (2004b). Challenges and practices in deploying web acceleration solutions for distributed enterprise systems. *International World Wide Web Conference* (pp. 297–308).

Li, W.-S., Po, O., Hsiung, W.-P., Candan, K. S., & Agrawal, D. (2003). Freshness-driven adaptive caching for dynamic content web sites. *DKE*, *47*(2), 269–296.

Luo, Q., Krishnamurthy, S., Mohan, C., Pirahesh, H., Woo, H., Lindsay, B. G., & Naughton, J. F. (2002). Middle-tier database caching for e-business. *ACM SIGMOD* (pp. 600–611).

Majumdar, R., Ramamritham, K., Banavar, R. N., & Moudgalya, K. M. (2004). Disseminating dynamic data with QoS guarantee in a wide area network: A practical control theoretic approach. *Real-Time and Embedded Technology and Applications Symposium* (pp. 510–517).

Mayer, C. B. (2005). *Quality-based replication of freshness-differentiated web applications and their back-end databases*. PhD thesis, Arizona State University.

Olston, C., & Widom, J. (2002). Best-effort cache synchronization with source cooperation. *ACM SIGMOD* (pp. 73–84).

Rabinovich, M., Rabinovich, I., Rajaraman, R., & Aggarwal, A. (1999). A dynamic object replication and migration protocol for an internet hosting service. *ICDCS* (pp. 101–113).

Rabinovich, M., Xiao, Z., & Aggarwal, A. (2003). Computing on the edge: A platform for replicating internet applications. *International Workshop on Web Content Caching and Distribution* (pp. 57 – 77).

Vallamsetty, U., Kant, K., & Mohapatra, P. (2002). Characterization of e-commerce traffic. *International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems* (pp. 137–144).

Wolfson, O., Jajodia, S., & Huang, Y. (1997). An adaptive data replication algorithm. *Transactions on Database Systems*, *22*(2), 255–314.

## ACKNOWLEDGEMENTS

Research funded by NSF grant 998404-0010819000

The views expressed in this paper are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

## KEY TERMS AND THEIR DEFINITIONS

**Database-driven portal application.** A web-based application consisting of business logic and a (usually) large database or database containing the application's content. In response to user requests, the logic queries the database, obtains needed content, and then uses that content to build a page (or other result) which is returned to the user. In this article, "portal", "portal application", "database-driven portal application", and "database application (DA)" are synonymous.

**Replication.** Making multiple copies of something in order to increase access to it.

**Quality-differentiated portal application.** A portal application that can store data and make its responses available in multiple temporal freshness levels.

**Application Service Provider (ASP).** A portal hosting service. In this article, ASPs are presumed to specialize in the hosting and replication of DAs.

**ACDN.** Active Content Deliver Network (ACDN) is a management system for the replication of DAs.

**DATE.** Database-driven Application AssignmenT Engine (DATE) is software program that makes replica-to-server assignments for multiple quality-differentiated DAs. The assignments meet user demand and minimize replication overheads.

**DASIM.** The Database Application System Implementer and Manager (DASIM) is a prototype database application replication environment.