Alternative Approaches to Distribute An E-Commerce Document Management System[*]

Shahram Ghandeharizadeh
Computer Science Department
University of Southern California
Los Angeles, California 90089

Abstract

This experience paper describes newsblaster.com, a commercial document management system that facilitates exchange of news worthy documents between two user communities: a) editor of newspapers and magazines who seek to publish articles, and b) member companies who want to disseminate information such as new product offerings, product recalls, alliances with other companies, etc. We focus on the Internet connection of a single node web server as the system bottleneck and discuss three alternative approaches to distribute the application across multiple nodes. The first, termed vertical, analyzes the semantics of the application and distributes it across multiple nodes based on its community of users and how they access the system. The second, termed horizontal, declusters the application's data across multiple nodes with the objective to distribute client requests evenly across these nodes. Finally, hybrid considers a horizontal partitioning of services identified by a vertical partitioning. We describe the application of each approach to newsblaster.

1 Introduction

Digital documents are at the core of many academic, governmental, and commercial institutions. One may find the following key services common to almost all document management systems:

1. authorized producers of content to upload their documents,
2. authorized consumers of content to access, review, and download documents,
3. querying capabilities so that users can identify relevant documents.

A specific application will almost certainly require additional services such as tools that translate a document into a format desirable for the purposes of a user, browsing capabilities, etc.

After a rapid growth in the last several years, the World-Wide-Web (WWW) has established itself in the mainstream of document management system. With a few mouse clicks, one may find a large number of online newspapers, technical papers, and many different kinds of reports ranging from cooking recipes to business plans and SEC disclosures. Several studies have repeatedly identified the slow response time of WWW as its main limitation [PK95, PK96][1]. A related challenge of E-commerce Document Management (EDM) sites is the availability of data in the presence of failures. This relates to slow response time because a user may perceive a slow EDM to have failed and unavailable to provide service. Both limitations are undesirable and may cause an EDM site to incur a significant revenue loss.

This experience paper reports on our insights gained from the development of an EDM site that specializes in uploading, distribution, and dissemination of company press releases. With this applica-

[*]This research was supported in part by the National Science Foundation grant IIS-0091843.

[1]Visit http://www.gvu.gatech.edu/user_surveys/survey-1998-10/ for the results of their most recent survey.

tion, the network connection contributes to the slow system response time. There are a host of alternative solutions to this problem. One approach is to purchase a higher bandwidth connection to the server. An alternative approach may distribute this EDM across multiple geographically distributed servers in order to minimize network contention at a single server, termed hot spots. This paper focuses on the later approach because it provides the added benefit of continued operation in the presence of one or more node failures. It describes three approaches to distribute the application: vertical, horizontal and a hybrid of these two approaches.

The rest of this paper is organized as follows. Section 2 provides an overview of Newsblaster and its EDM services. Section 3 describes the alternative approaches to distribute this EDM and their tradeoffs. Section 4 describes related research. Section 5 offers brief conclusions and future research directions.

2 Newsblaster: An Overview

Newsblaster.com is a startup[2] that provide document management services to two complementary community of users: (1) editors of magazines, newspapers, trade publications with interest in publishing news worthy information related to specific industries, and (2) member companies who want to announce news worthy information, e.g., a new product or service, a product recall, changes to a product, etc. A member company is provided with a portal to post press releases, review their previous postings and see how many editors have read each press release. Posting a press release entails:

1. uploading its related text and attachments, e.g., images, audio and video clips, etc.
2. specifying one or more target industries for this press release.

Based on the target industries, the system accesses a database of magazine editors to identify those with matching industry interests (this is detailed in the following paragraphs). When the press release is approved by a Newsblaster document administrator for dissemination, the system generates an email to each editor notifying them of the availability of this press release and a URL for this press release. This URL corresponds to the editor's portal that is generated dynamically when the editor references the URL. This dynamic HTML page is driven by a table that maintains the press releases queued up for each editor.

Each member's portal includes an activity log for their submitted press releases. When an editor references a press release, the system memoizes this information in order to provide the member company with feedback on how many editors picked up their press release. While the identity of the editor is kept anonymous (an editor may changes their setting so that they are not anonymous), the system shows the name of publication affiliation of this editor.

Each editor is provided with a dynamicly generated portal to pickup their press releases. Using this portal, an editor may either delete press releases that are not of interest or leave them queued up for future reference. In addition, the portal empower the editor to either adjust their existing profiles or setup new profiles. A profile pertains to an industry that is of interest to this editor. In essence, it is a filter that tells the system which press releases should be targeted to this editor. (Recall that a press release must be targeted towards one or more industries.) An additional attribute of this filter is a counter that specifies how many new press releases must be queued up for this profile before the system generates an email to prompt him or her to visit their portal. This way, an editor can control the system so that it generates neither too many nor too few email messages.

To prevent a member company from submitted unacceptable press releases (containing profanity, nudity, or not news worthy), all press releases must be approved. When the member submits a press release, the system queues it for approval. An administrator reviews each press release. When a press release is approved, the system: a) sends email messages to editors, and b) makes this press release visible to its all editors (when they search on a certain topic).

[2]Newsblaster.com was acquired by biz2biz.com in September of 2000. While the URL "newsblaster.com" is functional as of this writing, the URL is expected to change to biz2bizpress.com.

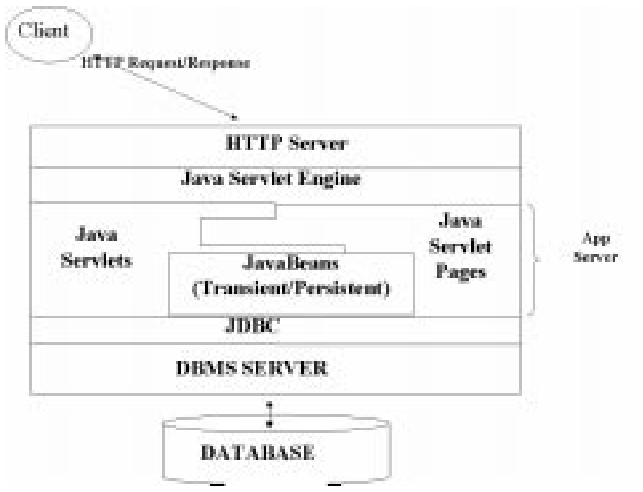

Figure 1. Software components of an application constructed using Sun/Apache Java Servlet Engine.

Query type	Frequency (%)
1. Exact match select referencing primary key	31.5%
2. Exact match select referencing non-primary key	19.5%
3. Range select query	0.06%
4. Join/Aggregate query	0.14%
5. Insert for individual records	22.2%
6. Updates using a selection predicate	25.99%
7. Delete using a range selection predicate	0.6%

Table 1. Query types that constitute Newsblaster's workload.

2.1 Implementation of Newsblaster

Newsblaster is implemented using dynamic HTML. We had a choice of software components for this implementation: Microsoft's Active Server Pages, PHP Hypertext Preprocessor, Sun/Apache's Java Servlet Engine, etc. We decided to use Sun/Apache's Java Servlet Engine. Figure 1 shows the components of the system:

- Web server: this is the front-end of the site that communicates with the clients' browsers using the HTTP protocol [FGM$^+$99].
- Application server: implements Newsblaster's functionality. This implementation is based on the Java Servlet Engine and the JavaBeans framework.
- Database management system (DBMS): provides access to document data stored on mass storage devices. Similar to numerous other web sites, Newsblaster utilizes a relational database management system. Java Development Kit provides JDBC class definition to interface with a DBMS using SQL. JDBC enables the application server to pose queries against remote database servers. Almost all commercial DBMSs include JDBC drivers for their products.

Newsblaster also includes a CyberCash component to facilitate processing of electronic transactions, membership dues. This component sits at the same abstraction layer as the DBMS with its own programming interface.

We used an ER data model to describe Newsblaster's database at a conceptual level. Next, we performed logical database design in order to produce a set of tables that constitute the database. These tables have been normalized and fine tuned for the purposes of this application. Table 1 shows the workload of our database from August 4 to September 14, 2000. Approximately 50% of the workload consists of exact match selection queries. This pertains primarily to editors picking up press releases, member companies logging in to the system, member companies looking up the activity log of a press release. Approximately 22% of the workload consists of insert commands. These reflect the following tasks performed by the system: maintain a list of editors that a press release is targeted towards, insert a press release into the mail box of an editor, log an editor picking up a press release. Approximately 25% of the workload consists of updates that manipulate a single tuple using an exact match predicate. This is attributed to the following. For each press release targeted towards an editor's profile, the system must increment the number of press releases queued up for this profile. If this number equals the threshold specified by the editor then the system generates an email message to the editor.

We wanted to ensure that Newsblaster can scale to accommodate millions of users. The following section entails our design strategies for accomplishing this objective.

3 Distributing Newsblaster

There are three ways to distribute our target web application:

1. Vertical - Analyze its semantics and separate its services based on its community of users, i.e., its workload. A good example is Microsoft's home page. It offers its visitors a host of information services: a) e-mail services, Hotmail, b) chat rooms, c) financial information, etc. A vertical distribution assigns each of these to a different node.
2. Horizontal - Decluster the application's data across multiple nodes with the objective to distribute client requests evenly, preventing formation of hot spots and bottlenecks. As detailed in Section 3.2, this strategy requires special measures in support of join and aggregate operations.
3. Hybrid - After a vertical distribution, apply horizontal distribution to each service.

When compared with vertical, Horizontal is more effective in distributing Newsblaster. Hybrid takes the best elements of these two strategies, distributing the application further.

3.1 Vertical Distribution

This approach exploits the semantics of the application by identifying the different services and assigning them to a different node. For example, newsblasters provides two distinct servers: a) members posting press releases, and b) editors reading the submitted press releases. One may assign each to a different node[3], termed $Node_m$ and $Node_e$, respectively. Logically, these two sets are connected by the press release entity set. One may assign this entity set in different ways:

1. press releases can be assigned to a different node, termed $Node_p$. This forces $Node_e$ and $Node_m$ to make remote JDBC connections whenever they need to access a press release.
2. assign press releases to either $Node_e$ or $Node_m$, forcing the other node to make remote JDBC calls to access this data.
3. assign press releases to one of the two nodes (say $Node_e$) with a replica assigned to the other node ($Node_m$). This requires the system to keep replicas consistent. (This does not apply to newsblaster because press releases are not updated; they are ready-only.)

With the 3rd approach, both $Node_e$ and $Node_m$ are still required to make remote JDBC references to access shared tables. To illustrate, we outline an example scenario that requires $Node_e$ to make a remote connection to $Node_m$. Recall that the member's activity logs shows the editor activity for each of their posted press releases. This is implemented using a table named "EditorLog" that maintains which editor reference a press release at what time. It is natural to assign this table to $Node_m$ because members are anxious to see who is reading their press releases (and will access this table frequently). At the same time, $Node_e$ must make a remote JDBC connection to insert an entry into this table when an editor references a press release.

With a vertical partitioning, client requests are separated into those from editors and members. Each is directed to its own node. This decision is made at the client site. For example, with editors, the email containing the URL of a press release references $Node_e$. Similarly, all members who want to login to view the status of their press release utilize $Node_m$ because the advertised URL (www.newsblaster.com) is tied to this node's network interface card.

If either $Node_e$ or $Node_m$ fails, the system can continue operation with partial service loss. Of course, one can implement measures to compensate and prevent such a loss. For the sake of brevity, we use an example to illustrate this point and eliminate a discussion of each strategy and its implementation

[3]A node is a computer consisting of one or more CPUs, some memory, several magnetic disks, and potentially a tape device.

requirements from a vertical distribution perspective. For example, if $Node_m$ fails then a member is no longer able to login to either post new press releases or view the status of their posted press releases. At the same time, if an editor references a press release, $Node_e$ can no longer insert an entry into the "EditorLog" table that resides on the failed node, $Node_m$. Both are examples of functionality loss. One may prevent this loss by implementing a buffering scheme at $Node_e$ that caches the relevant information when $Node_e$ is down. Once this node becomes operational, the contents of this cache are transmits to $Node_m$ and its "EditorLog" table is brought up to date.

3.2 Horizontal Distribution

With a horizontal distribution, we add S slave servers to the existing web server. The total number of nodes in the system is N (equals to $S + 1$). The original web server contains the entire database and is termed the master server. (The role of master and slave servers is detailed in the following paragraphs.) The database is horizontally declustered [RE78, LKB87, DGS+90] across the slave servers. Each piece of a relation on a slave server is termed a fragment. Fragments are created using a hash partitioning strategy that employs the primary key of each table. For example, both the editors and members are hash partitioned using their userid, the primary key of each table. Similarly, press releases are hash partitioned using their PRID, press release identifier which is a composite key consisting of a member's primary key and the system clock.

One of the N nodes must have its network card binded with the advertised URL and execute a dispatcher process (a Java Servlet). This node might be either be the master server or one of the slave servers. Once a client references the advertised URL, the dispatcher redirects it to one of the N nodes for processing, say $Node_t$. From then on, all interactions between the client and the system takes place with $Node_t$ (and maybe other nodes) without participation of the dispatcher. The dispatcher maps a client to $Node_t$ based on the following information embedded in the HTTP request by the programmer: (relation, attribute, value). By looking at this information, the dispatcher can route the request to the node containing the relevant information. For example, with the login page, the programmer can embed: (User, login, Shahram). Based on the hash partitioning of the User table, the dispatcher utilizes a hash function to determine $Node_t$ containing the record corresponding to "Shahram". If either the programmer fails to provide the required information or the provided attribute is not the partitioning attribute, the dispatcher can route the request to a random node for processing.

Once the dispatcher directs a client to $Node_t$, this node processes all requests generated by the client. $Node_t$ might make remote JDBC calls to other nodes to access data or update their fragments. For example, after logging in, a member might submit a press release. The PRID of this press release might map on to $Node_u$. This means that $Node_t$ must make a remote access to $Node_u$ to preserve the integrity of hash partitioning. All client updates performed on a slave server are propagated to the master server. These updates are termed internal updates. Similarly, all client updates performed on the master server translate into internal updates directed to the slave server containing the relevant fragment of data.

The master server's primary responsibility is to process join and aggregate functions. Slave servers cannot process joins in isolation because records of tables residing on two different nodes might join with one another. A server must have the entire copy of the database in order to perform arbitrary join predicates and the master server satisfies this requirement. The slave servers can process a star join predicates[4] that reference the primary keys of participating relation. Newsblaster's workload does not have a single star query.

With a horizontal distribution, the service becomes unavailable when the node containing the dispatcher process fails. When the master server fails, the system can continue operation with degraded service: it cannot perform the updates and joins. When a slave server fails, the system may not be able to perform those updates that impact the data fragments residing on the failed server. One may design buffering schemes that enable the master server to accumulate such updates and propagate them to the failed slave server once it becomes operational. This would enable the system to provide a complete ser-

[4]A star join predicate employs one attribute of all participating tables.

vice in the presence of failed slave servers.

3.3 Hybrid Distribution

With the hybrid approach, the application is first distributed in a vertical manner. Subsequently, each vertical service is horizontally distributed across multiple slave servers. If the application provides M services, then each service can be dispersed across an arbitrary number of slave servers. One may either employ a master server for each service or utilize one for the entire system. We prefer the later approach because it simplifies system implementation.

With one master server per service, the system must ensure that all have the same snap shot of the database. One approach is to designate one of the masters as the primary and the remaining ones as secondary. The master server would be responsible for propagating updates to the secondary servers. Moreover, in the presence of failures, the design of the system must prevent the replicated master servers from falling out of synchronization with one another [GHOS96]. The overhead of maintaining multiple master servers might outweigh its benefits. Hence, we postpone it as a future research direction.

4 Related Work

Document management has been an active area of research for more than a decade. The novelty of this study pertains to both the WWW and emerging components such as Java Servlets to construct document management systems. As such, we believe that the following two studies are most relevant [BRD00, BCC$^+$00]. We start this section with SIFT, a document management system with features similar to Newsblaster. Next, we focus on the web related studies.

Newsblaster as a document management system shares similarities with the Stanford Information Filtering Tool, SIFT [YGM99]. With SIFT, each user submits a profile consisting of a number of standing queries to represent his or her information needs. The system then continuously collects new documents from the underlying information sources, filters them against the user profile, and delivers relevant information to the user. A user of SIFT may submit to the system a document of interest. The system would analyze this document and accordingly adjust the weights of the words in the user's profile, making the future delivered documents more relevant. This study recognizes the importance of making its services distributed and employs replication to support multiple dissemination servers. It describes three alternative techniques to realize a distributed algorithm to match documents to profiles: a) Majority consensus arrangement [YGM95], b) Grid protocol that is related to [CAA90], and c) hierarchical organization [RST92].

Our study is different because we assume that the industry interests for both a press release and an editor is presented as meta-data[5]. (The industry interest of an editor is similar to a profile in SIFT.) This meta-data is stored in a table that is dispersed across multiple nodes. Moreover, in order to identify the editors that should be contacted, the system issues a select query with a conjunctive predicate against the table that contains the editor profiles and their industries. This enables us to use declustering with one replica (instead of multiple replication of SIFT).

Bistro [BCC$^+$00] provides a framework to address the hot spots that occur when uploading files on the Internet. Example applications include NSF's FastLane system, conference paper submission, etc. The Bistro framework is a simple, yet elegant, framework that consists of three stages: time-stamp, commit, and data transfer. Using multiple Bistros, this framework can implement a number of extreme cases for document uploading. For example, it can support Newsblaster's framework where a member company may store their documents on an arbitrary server (either master or slave). It can also support other paradigms where documents must be propagated to a single server. Our proposed solutions are different because they consider the meta-data that is associated with an uploaded document and how to process queries.

Distributed file systems can facilitate sharing of content among many web servers. With large scale systems, e.g., IBM's Olympics web site [Tra99], sequential consistency may cause Web servers to observe incomplete changes to file data that can result in HTML parsing errors at clients [BRD00]. Pub-

[5]We do not attempt to derive this meta-data by analyzing a document.

lish consistency model, introduced by [BRD00], extends Web servers session to files with semantics in order to prevent such errors. Moreover, it introduces producer-consumer locks to reduce the messages required to distribute updates to file system clients, minimizing network load. These techniques are suitable for sharing of documents by the servers (both master and slave). However, they do not apply to the processing of the meta-data files. In particular, it would be a mistake to share the master server's database using a distributed file system to enable slave servers to perform query processing. This is because queries are selective and it is more efficient to ship the query to the master server (instead of shipping the data to the slave servers) for query processing. This minimizes the amount of network traffic and enhance system performance.

5 Conclusion and Future Research Directions

We described a web-based document management system that accumulates and disseminates press releases. It is implemented using dynamic HTML based on Java Servlet Engine. Based on this framework, we explored three alternative approaches to distribute this application across multiple geographically distributed applications: vertical, horizontal, and hybrid. The approaches apply to the document meta-data information that is maintained by a relational database management system. They strive to balance the system load evenly across a geographically set of distributed servers while employing the Java Servlet Engine framework. In addition, they provide a certain degree of autonomy so that the system can continue operation in the presence of failures.

Several research topics constitute the focus of our future investigations. First, the definition of a "failed" server is somewhat ambiguous in a geographically distributed environment. A server might appear failed because of partitioned network. This has the greatest impact on both horizontal and hybrid because one or more slave servers can be separated from the rest of the environment due to failures that partition the network. A system operating in failed mode must consider these realities [GMB85, Gif79]. With horizontal, the discussions of Section 3 can handle this scenario because they prevent a slave server from updating its fragments when it cannot communicate with the master server. However, this requires further investigation with a hybrid distribution that consists of multiple master servers, see Section 3.3. Second, the performance of these alternative solutions needs to be characterized. We intend to start with analytical models of these alternatives prior to implementing them in the system. Finally, we are investigating the feasibility of a general purpose library system that can horizontally distribute a web application in a seamless manner. It appears that based on the Java Servlet framework, it is possible to realize such a capability.

References

[BCC+00] S. Bhattacharjee, W. Cheng, Chou C., Golubchik L., and Khuller S. Bistro: A Framework for Building Scalable Wide-Area Upload Applications. Performance Evaluation Review, 28(2):29–35, 2000.

[BRD00] R. Burns, R. Rees, and Long D. Consistency and Locking for Distributed Updates to Web Servers Using a File System. Performance Evaluation Review, 28(2):29–35, 2000.

[CAA90] S. Cheung, M. Ammar, and A. Ahamad. The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data. In Conference on Data Engineering. IEEE, 1990.

[DGS+90] D. DeWitt, S. Ghandeharizadeh, D. Schneider, A. Bricker, H. Hsiao, and R. Rasmussen. The Gamma database machine project. IEEE Transactions on Knowledge and Data Engineering, 1(2), March 1990.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol - HTTP/1.1. Technical report, RFC 2616, 1999. http://ftp.isi.edu/in-notes/rfc2616.txt.

[GHOS96] Jim Gray, Pat Helland, Patrick E. O'Neil, and Dennis Shasha. The dangers of replication and a solution. In ACM Sigmod Conference, pages 173–182, May 1996.

[Gif79] D. K. Gifford. Weighted Voting for Replicated Data. ACM SIGOPS SOSP, December 1979.

[GMB85] H. Garcia Molina and D. Barbara. How to Assign Votes in a Distributed System. Journal of ACM, 32(4), October 1985.

[LKB87] M. Livny, S. Khoshafian, and H. Boral. Multi-Disk Management Algorithms. In Proceedings of the 1987 ACM SIGMETRICS Int'l Conf. on Measurement and Modeling of Computer Systems, pages 69–77, May 1987.

[PK95] J. E. Pitkow and C. M Kehoe. Results from the Third WWW User Survey. The World Wide Web Journal, 1(1), 1995.

[PK96] J. E. Pitkow and C. M Kehoe. Emerging Trends in the WWW User Population. Communications of the ACM, 39(6):106–108, 1996.

[RE78] D. Ries and R. Epstein. Evaluation of distribution criteria for distributed database systems. UCB/ERL Technical Report M78/22, UC Berkeley, May 1978.

[RST92] S. Rangarajan, S. Seita, and S. Tripathi. Fault Tolerant Algorithms for Replicated Data Management. In Conference on Data Engineering. IEEE, 1992.

[Tra99] Transarc. Transarc's DFS Provides the Scaleability Needed to Support IBM's Global Network of Web Servers. Technical report, Transarc, 1999. http://www.transarc.com/Solutions/ Studies/EFS_Solutions/ DF-SOly/dfsolympic.htm.

[YGM95] T. W. Yan and H. Garcia-Molina. Duplicate Removal in Information Dissemination. In Very Large Data Bases, September 1995.

[YGM99] T. W. Yan and H. Garcia-Molina. SIFT Information Dissemination. ACM Transactions on Database Systems, December 1999.