Designing Responsive Database-Driven Web Sites

Vlad Wietrzyk
School of Computing and Information Technology
University of Western Sydney
v.wietrzyk@uws.edu.au
Australia

Bill Grosky
University of Michigan
Dearborn
USA

Abstract

With the wide availability of content delivery networks, many e-commerce Web applications utilize edge cache servers to cache and deliver dynamic content at locations much closer to users, avoiding network latency. Replication of Web documents can improve both performance and reliability of the Web service. Response time is a key differentiation point among electronic commerce (e-commerce) applications. We choose to develop a framework for a secure, highly available distributed workflow architecture since interworkflow is anticipated as a major supporting mechanism for Business-to-Business Electronic Commerce. In this paper, we analyze the factors that impact the performance and scalability of a database-driven Web site.

1. Introduction

Responsiveness (high availability and timeliness) is a crucial issue in the design of a Web service. From the service user perspective, a poorly responsive service can be virtually equivalent to an unavailable service.

The general architecture of a database-driven Web-based e-commerce site is depicted in Fig. 1. This type of architecture overcomes the inconsistencies associated with the distributed business logic location in the simpler client-server model. It also helps resolve inconsistencies such as overloading, which causes performance problems.

Web services technology is changing the Internet, augmenting the `eyeball web` with capabilities to produce a `transactional web`. The eyeball web is dominated by `program-to-user` business-to-consumer (B2C) interactions. The transactional web will be dominated by `program-to-program` business-to-business (B2B) interactions. This transformation is being fueled by the program-to-program communication model of Web services built on existing and emerging standards. Recent studies suggest that more than 85 percent of the traffic on the Internet backbone is HTTP-related [2]. In the rest of the paper, we present and discuss several elements of the advanced data distribution strategies. Data distribution strategies are presented in section 2. Replicated servers selection algorithms are tested and discussed in section 3. Section 4 describes an architecture for a secure, highly available distributed workflow since interworkflow is anticipated as a major supporting mechanism for Business-to-Business Electronic Commerce. Implementation of the secure distributed workflow is presented in section 5. Section 6 describes experimental study. Finally, section 7 concludes the paper with a summary and future work.

2. Data Distribution Strategies

Web services are becoming data-intensive when their primary goal is to make large amounts of data accessible to a variety of users. A typical application scenario is electronic commerce on the Web.

When the application runs in a distributed environment and with a significant load, additional mechanisms are needed to provide reasonable performance. If a centralized repository is used, the repository is likely to become a bottleneck and a single point of failure.

To solve this impasse, we propose to use the semantic knowledge embedded in the documents handling mechanisms, which can be, for example the workflow engine. By exploiting this knowledge, a much "looser" (asynchronous) distributed document management policy can be implemented. For this purpose the replication mechanisms can be used.

3. Replicated Servers Selection Algorithms

Many Web sites employ a replication of their sites. We considered the following algorithms in the comparison:

Fixed Host: Client sends all requests to a fixed server. This corresponds to the current scenario on the Web with no

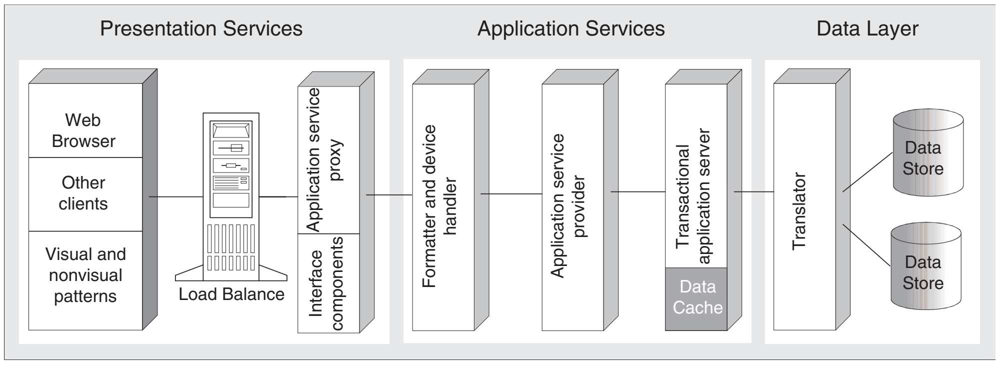


Figure 1. Complex Three Tier Client/Server Architecture with DB2

server replication.

Hops: Client sends all requests to a server that has the closest distance from the client in terms of the number of hops.

Refresh: The server with the minimal HTTP request is chosen.

Parallel: Client sends requests to all servers replicating the document and waits for the first response.

Ping: Client periodically sends a ping request to each server and redirects all HTTP requests to the server with minimal ping round trip time.

Probabilistic: The probability that a given server is selected is given by: $Prob_i = K / lat_i$,
where lat_i is the average `HTTP` request latency on server i and K is normalizing constant defined as $K = 1 / \Sigma_i (1 / lat_i)$.

To analyze the behavior of these algorithms we have conducted two sets of experiments, one with 30 servers and another one with 3 servers chosen out of the first set of 30 servers. The results of the experiments for the 3-server replication group are depicted in Figure 2, while the results for 30-server replication group are given in Figure 3.

4. Business-to-Business Architecture

Global information management strategies based on a sound distributed architecture are the foundation for effective distribution of complex applications that are needed to support ever changing operational conditions across security boundaries.

We present the fully distributed architecture for implementing a Workflow Management System (WFMS). A workflow distributed database consists of a set $\mathcal{N}$ of sites, where each site $N \in \mathcal{N}$ is an MLS database. The sites in

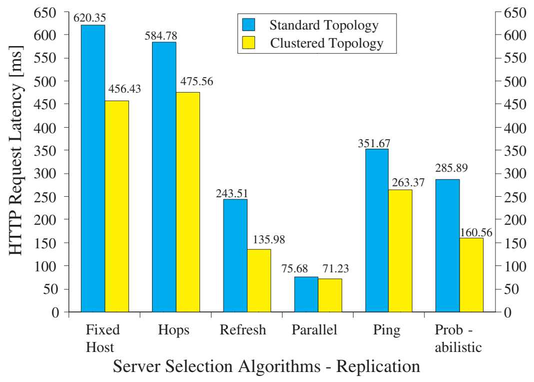


Figure 2. HTTP Request Latencies for 3-Server Clustered Replication Group

the workflow system are interconnected via communication links over which they can communicate. The WFMS architecture operates on top of a Common Object Request Broker Architecture (CORBA) implementation. A CORBA's Interface Definition Language (IDL) is used to provide a means of specifying workflows. Also we assume that communication links are secure — possibly using encryption. This distributed workflow transaction processing model describes mainly those components necessary for the distribution of a transaction on different domains.

Domain is a unit of autonomy that owns a collection of flow procedures and their instances.

If a transaction should be dstributed on several domains — a global transaction, in every domain there must exist the following components, (see Fig.4).

- TM - Transaction–Manager. The transaction manager plays the role of the coordinator in the respective domain. If a transaction is initiated in this do-

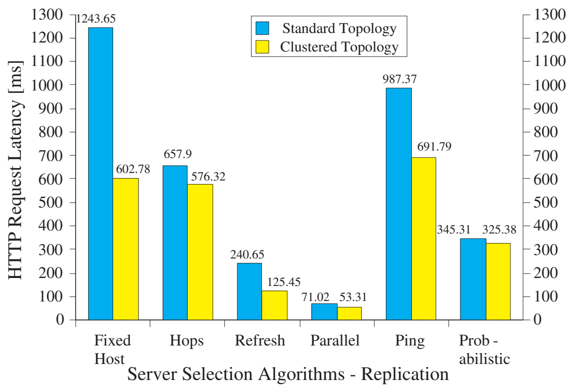


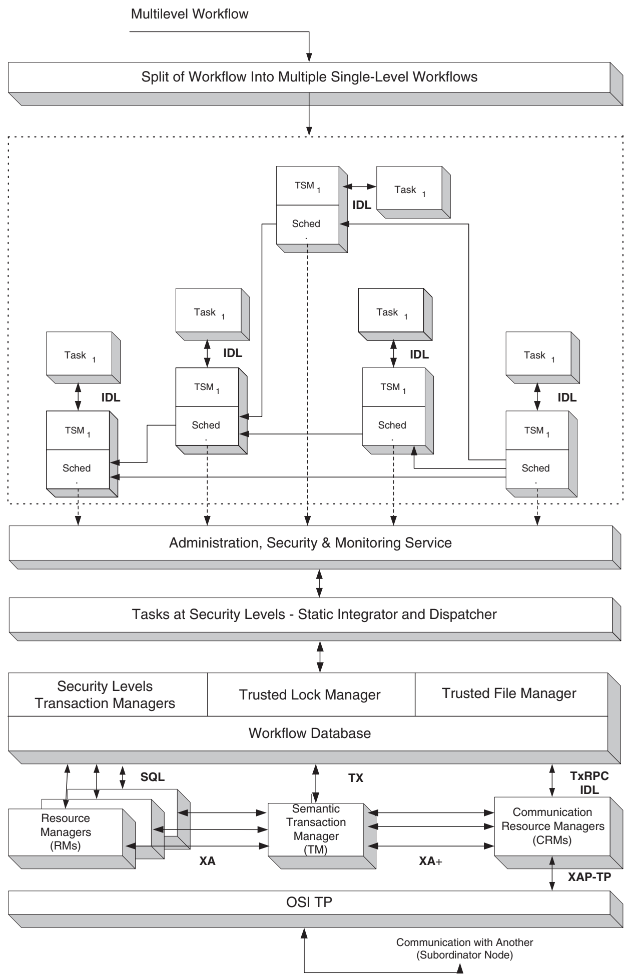
Figure 3. HTTP Request Latencies for 30-Server Clustered Replication Group

main, the TM assigns a globally unique identifier for it. The TM monitors all actions from applications and resource managers in its domain. In every domain involved in the distributed workflow transaction environment there exists exactly one TM.

- CRM - Communication–Resource–Manager. Multiple applications in the same domain talk with each other via the CRM. This module is used by applications but also other management components for inter-domain communication. CRM is the most important module with respect to the transactional support for distributed workflow executions. Our model specifies the T*RPC as a communication model, which supports a remote procedure call (RPC) in the transactional environment.

- RM - Resource–Manager. An accountable performer of work. A resource can be a person, a computer process, or machine that plays a role in the workflow system. This module controls the access to one or more resources like files, printers or databases. The RM is responsible for the ACID properties on its data records. A resource has a name and various attributes defining its characteristics. Typical examples of these attributes are job code, skill set, organization unit, and availability.

- AMS - Administration–Monitoring–Service. The monitoring manager is used to control the workflow execution. In our approach, there is no centralized scheduler. In the figue, each Task Manager - designated as TSM, is equipped with a conditional fragment of code which determines if and when a given task is due to start execution. The scheduler communicates with task managers using CORBA's asynchronous Interface Definition Language(IDL) interfaces. Task managers communicate with tasks using synchronous IDL interfaces as well. AMS module is also responsible for the coordination of the different sites in case of an abort that involves multiple sites. Individual task managers communicate to monitoring manager their internal states, as well as data object references - for possible recovery.



Multilevel Workflow
Split of Workflow Into Multiple Single-Level Workflows
TSM 1
Task 1
IDL
Sched
Administration, Security & Monitoring Service
Tasks at Security Levels - Static Integrator and Dispatcher
Security Levels Transaction Managers
Trusted Lock Manager
Trusted File Manager
Workflow Database
SQL
TX
TxRPC IDL
Resource Managers (RMs)
Semantic Transaction Manager (TM)
Communication Resource Managers (CRMs)
XA
XA+
XAP-TP
OSI TP
Communication with Another (Subordinator Node)

Figure 4. The MLS Distributed Workflow Architecture

The distributed architecture suits the inherent distributional character of workflow adequately in a natural way.

This approach also eliminates the bottleneck of task managers having to communicate with a remote centralized scheduler during the execution of the workflow. This architecture also posseses high resiliency to failures — if any one node crashes, only a part of the workflow is affected.

5. Implementation of the Secure Distributed Workflows

The architecture of our multilevel secure workflow transaction processing system can be divided into a trusted, an untrusted, and a receptive module as shown in Fig. 4.

The trusted component consists of two functioning modules: Trusted Lock Manager and Trusted File Manager. The untrusted component is a collection of Transaction Managers (TMs), one for each security level. The receptive component is the Workflow Database - its File Store component, which may be physically partitioned according to the security levels. Fig. 4 shows the interfaces between the different components of our implementation. Except for the native interface between the RMs and the Workflow Database, all interfaces are defined as an API in the X/Open standard.

The major advantage of this implementation is the standardisation of the interfaces. Due to the modular concept, all components and applications can be easily extended and replaced. So cooperation between components of different vendors is possible.

Note that in order to accomodate relevant commercially available DBMS to support our approach, the assumption about Lock Manager being trusted can be relaxed by providing one Lock Manager for each security level. If that particular option is taken, a Trojan Horse inside some untrusted Lock Manager can compromise the correct execution of Concurrent workflow transactions, but cannot violate security. Additionally, as the whole body of a standard Lock Manager, written with all the requisite defensive programming, exception handlers, optimizations, deadlock detectors, etc. comes to about a thousand lines of actual code, it is easily verifiable. Thus our assumption of a Trusted Lock Manager, which jeopardises neither security nor integrity, is justified.

Similar implementations can be found in the context of extended transaction models, for example the reflective transaction framework. Once a workflow is selected from the repository, several steps are carried out that instantiate a workflow instance that is able to run within our execution environment.

6. Experimental Study

We have adapted the hypermodel benchmark [3] to test our algorithms. The benchmark itself comprises operations for:

- *creating* the initial test database with clustering
- *range queries*
- *incremental modifications* of attributes at a number of randomly selected nodes
- *closure operations*
- *reference lookups* (so-called group lookups)

We present in this section only a few experimental results concerning our experiences with the hypermodel benchmark.

We found that the size of workstation buffer pool has no effect on the performance of the sequential scan query and the results are not included.

6.1. Processing of Multiple Queries Concurrently

The main objective in a heavily loaded system is to improve the average response time of queries under a given system throughput [1]. The throughput is established for a given mix of queries by the arrival rate of queries per second. The issue concerning us is to define how the policy of multi-page requests determines the average response time when multiple queries are concurrently processed at a time.

A *query signature* is defined by one or several query types. Every query type is characterized by the parameters listed in table 1.

Table 1. Query signature parameters.

QS_F	query signature.
P_T	the number of target pages.
C_F	the number of cylinders occupied
	by a file F.
ξ	the average number of
	queries which arrive in one second.
QT_i	i-th query type.
$p\omega_i$	probabilty of a query being of
	the i-th query type.
P_{T_i}	the number of target pages of
	query type $QT-i$.
C_i	the number of file cylinders of
	query type QT_i.

Additionally to the query types, a query signature is characterized by the arrival rate of the queries. In our experiments we assume that the arrivals of queries follow an exponential distribition where ξ designates the average number of queries which arrive in one second time interval. Generally, a query signature QS_F can be described by the following string of parameters:

$$QS_F = (\xi,\, p\omega_1,\, QT_1\,(P_{T_1},\, C_1), \ldots \\ ,p\omega_i,\, QT_k\,(N_k,\, C_k))$$

QT_i is the i-th query type which executes streams of queries with probability $p\omega_i$, where $1 \leq i \leq k$.

First, we performed experiments on the following query signatures:

$$QS_{F1}(\xi) = (\xi,\, 1.0,\, QT(15,\, 5))$$

We varied the values of parameter ξ from 1 to 20. Fig. 5 shows the average response time in ms. As can be seen from the equation above, when $\xi > 5$ the average response time per query dramatically increases for queries using single-page requests. However, multi-page requests still offer low response time for the queries.

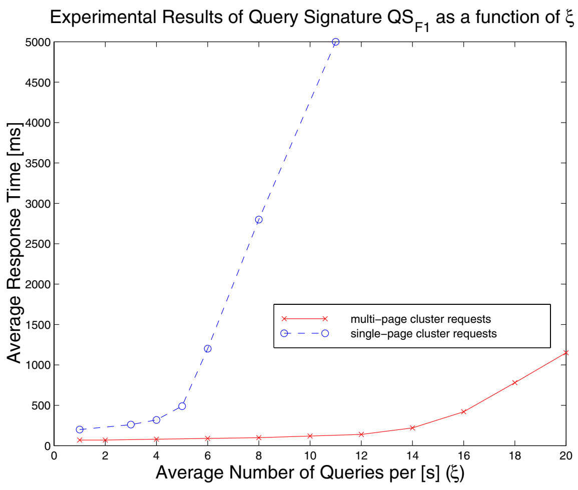


Figure 5. Experimental Results of Query Signature QS_{F1} as a function of ξ

In the next group of experiments, the query signatures are examined where parameter ξ is varying in the range between $1 \leq \xi \leq 30$, each of them comprising four different query types. It is assumed that the query type that will access only one page will occur with probability 60%, however probability of a query that will access 100 target pages is only 10%.

$$QS_{F2}(\xi) = (\xi,\, 0.6,\, QT(1,\, 1),\, 0.1, QT(20,\, 5), \\ 0.1, QT(50,\, 5),\, 0.2, QT(100,\, 10))$$

We studied an important problem related to how the average response times of small queries perform when all queries are executed using multi-page clusters with a varying number of pages.

We addressed this question for our query signature $QS_{F2}(\xi)$, by plotting the average response time of the queries of type $QT(1,\, 1)$ in Fig. 6. The query of type $QT(1,\, 1)$ cannot take advantage from multi-page cluster request. Therefore, response time shows similar results on both graphs for a small value of ξ - a lowly loaded system. In case of a heavily loaded system, the average response time of $QT(1,\, 1)$ queries type is distinctly lower when multi-page cluster requests are executed in comparison to using single-page cluster queries.

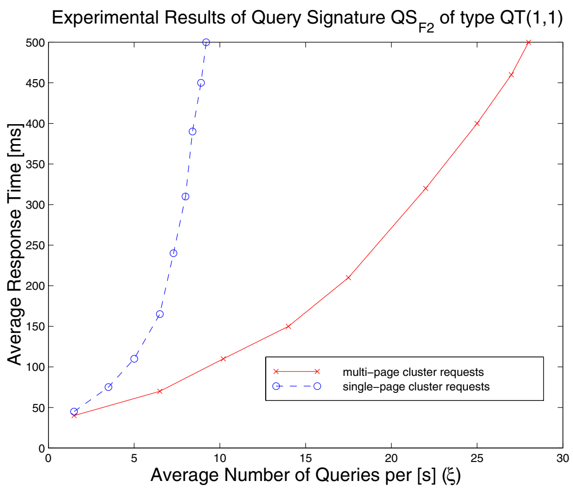


Figure 6. Experimental Results of Query Signature QS_{F2} of type $QT(1,1)$

7. Conclusion and Future Work

As can be seen from the results of our experiments examining several algorithms for selection of replicated servers on the Web, the Parallel algorithm operating on clustered topology provides the best latency in all cases. This paper has discussed issues related to the scalability and performance of back-end database servers used in e-business applications. The distinguishing feature of the workflow transaction support system proposed is the ability to manage the arbitrary distribution of business processes over multiple workflow management systems. The major advantage of our implementation is the standardisation of the interfaces.

We covered additional operations that can be performed on instances in the cache. A shared implementation cache is most useful when we use nontransactional access or optimistic transactions. It is becoming increasingly clear that companies will need to be able to constantly monitor the performance of their Web sites and fine tune all aspects of the Web site to ensure better end user experience.

References

[1] M. B. Seeger, P. A. Larson. Reading a set of disk pages. In *VLDB*, pp. 35-44 Dublin, Ireland, 1993.

[2] G. M. K.Claffy and K. Thompson. The nature of the beast: Recent traffic measurements from an internet backbone. In *Cooperative Assoc. for Internet Data Analysis (CAIDA)*, pp. 25-54 Warszawa, Poland, 2003.

[3] M. M. e. a. T. L. Anderson, A.J. Berre. The hypermodel benchmark. In *Advances in Database Technology*, pp. 8-20 New York, USA, 1990.