98-DO

Proceedings

36th Annual Southeast Conference

Marietta, Georgia
April 1–3, 1998

K. N. King, Editor

Session II-B: Computational Science

Session II-C: Networks and Computer Architecture

Session III-A: Artificial Intelligence

Session III-B: Web Technology

Key: (C) Concise paper (G) Graduate student paper (U) Undergraduate student paper

Development of an Open and Scalable Web-based Information Publishing System *

Chung-Min Chen, Naphtali Rishe
High-Performance Database Research Center
School of Computer Science
Florida International University
Miami, FL 33199

Abstract- This paper describes the design principles of the Web-based Information Publishing System (WIPS) that is currently under development at the Florida International University. The goal of the project is to facilitate efficient querying of databases through the Web and rapid application development. We examine several design issues related to performance and functionality. We highlight an important technique called *query packing*, which is used to improve the performance under heavy loads. The ultimate goal of WIPS is to serve the need of the Regional Application Center (RAC) at the Florida International University (FIU), a collaborative effort of NASA and FIU, in disseminating and enabling search of large volumes of scientific data to a diverse community of earth and environmental researchers through the Internet.

1 Publishing Database Data on the Web

The World Wide Web (the Web for short) has been growing explosively and becoming the most popular means for information dissemination and discovery through the Internet [1]. Publishing data through the Web is of great interest to many organizations, because of the Web's ease-to-use interface and the Internet as a cost-effective communication infrastructure. However, as the Web was originally designed for delivering data stored in flat files, it provides no direct support to access structured data stored in database systems, which host a large portion of the world's business data. To bridge the connection gap between the Web and the back-end database servers, various software solutions have been proposed.

One of the solutions is based on the Common Gateway Interface (CGI) – a standard interface through which a Web server may invoke an external program [6]. Using this approach one must write a CGI program that performs the following jobs: interpret the user request passed over from the Web server, form the corresponding SQL query and deliver it to the database server, and return the results to the user. A CGI program can be written in a programming language (e.g. C++) as well as in a script language (e.g. Perl). The major difference between a CGI and a regular program is that the standard input and output of a CGI program is the Web server, rather than a terminal.

CGI-based connections are simple to develop and can be ported to work with virtually all kinds of Web and database servers. However, CGI systems have long been criticized for their non-optimal use of system resources: each request will invoke a separate CGI process with its own address space. And usually a separate connection is established between each CGI process and the database server. During a heavy load of requests, the communication overhead, especially the connection set-up and initialization cost, may cause contention and become the performance bottleneck.

In recognition of the performance problem of CGI, some commercial Web and database vendors have provided alternative solutions: either by extending the Web server with an API set (e.g., Netscape's NSAPI and Microsoft's ISAPI) or by enabling the database server to handle the HTTP protocols (e.g. Oracle's Web Application Server). Products of this kind eliminate the resource problem of CGI and are expected to yield better

*This research was supported in part by NASA (under grants NAGW-4080, NAG5-5095, and NRA-97-MTPE-05), NSF (CDA-9711582, IRI-9409661, and HRD-9707076), ARO (DAAH04-96-1-0049 and DAAH04-96-1-0278), DoI (CA-5280-4-9044), NATO (HTECH.LG 931449), and State of Florida.

performance. However, they gave up the nice property of openness: the development of the applications is either subject to the proprietary APIs or the server architecture. The consequence is that an application program developed on one vendor's APIs or servers can not run on another vendor's server platform.

Another alternative is to use the Java Applet [3]. Using this approach, the Web server will, in response to a request, dispatch a client program (written as a Java Applet) to the requesting user. After arriving at the user's computer, the client program establishes direction connection with the database server for further interactions. While this approach eases application development (as it goes by the Web server and CGI), it incurs Applet dispatch overhead. Although such overhead may pay off for long user transactions, it causes unnecessary response time delay for single-query transactions.

In the rest of the paper, we overview the Web-based Information Publishing System (WIPS): our implementation towards an open architecture for Web-based publishing systems that offers scalable performance. In particular, we examine a *query packing* mechanism that is devised to improve the system performance.

2 System Architecture of WIPS

Figure 1 shows the system architecture of the WIPS. The openness requirement motivates us to adopt the CGI mechanism since it is the only standard. In the WIPS, the CGI processes interact indirectly with the database server through a *database gateway* software. The only job of a CGI process is to pass parameters to and receive results from the database gateway, thereby consuming much less resources than the pure CGI approach.

There are three major components in the database gateway. The Application Repository (AR) hosts a collection of applications. The applications can be brought up on-demand and terminated in absence of pending requests. A running application is called an *application server*. The Request Manager (RM) is a daemon process that awaits requests from the CGI processes. It parses the request and passes associated parameters to the target application server. The Database Access Module (DBAM) maintains an all-time connection with the back-end database server. It issues and optimizes accesses (in SQL) to the database server on behalf of the application servers.

The reason for using a three-tier approach (CGI, database gateway, and database server) is twofold: First, it eases application development by isolating business logic from the CGI programs and thus avoids duplicated application images. Second, with a centrally controlled rather than multiple independent connections to the database server, the DBAM incurs less communication overhead and enables optimization on query and result delivery scheduling.

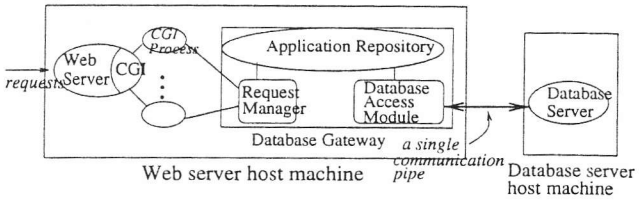

Figure 1: System Architecture of WIPS

3 The Query Packing Mechanism

The overall performance of a Web-enabled database system depends not only on the DBMS's query optimizer but also on the scheduling of queries and results . To lessen WIPS's performance degradation under heavy request loads, we have incorporated a *query packing* mechanism in the DBAM module. Instead of sending queries in separate messages to the database server for processing, the query packing mechanism dynamically packs multiple queries in a single message. The benefit is that the message transfer overhead is now amortized among several queries. For example, if we pack 100 requests into one query, then only two messages need to be exchanged between the DBAM and the database server (one for the query, the other for the returned result). Without packing, a total of 2*100 message exchanges, two for each request, will be needed.

The query packing idea can be further extended to combine several selection queries that are issued against the same table into a single selection query (called the *packed query*). The packed query will have a selection condition that is a disjunction of those of the composing queries. A Result Multiplexer (RX) is therefore needed to evaluate the records in the result set of the packed query and direct them to the corresponding CGI processes.

Consider for example the following two independent queries:

Q_1: **select** * **from** books
where book.title = "Calculus"
Q_2: **select** * **from** books
where book.author = "Eric Johnson"

They can be packed into a single query as:

Q: **select** * **from** books
where book.title = "Computers" **or**
book.author = "Eric Johnson"

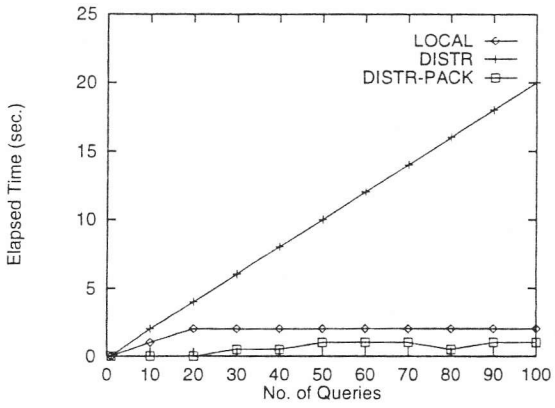

Figure 2: Query Response Time vs. Request Loads

Let $R(Q)$ be the result for query Q, the Result Multiplexer must perform the following after receiving $R(Q)$ from the database server:

for each record r in $R(Q)$ **do** {
  if $r.title$ = "Calculus" **then** add r to $R(Q_1)$;
  if $r.author$ = "Eric Johnson" **then** add r to $R(Q_2)$}

In the following we provide a preliminary experimental result which shows the performance benefit of the query packing technique. The experiment compares the elapsed time of running a sequence of Web requests using three different architectures:
LOCAL: pure CGI model, Web and database servers on the same machine
DISTR: WIPS model without query packing, Web and database servers on different machines
DISTR-PACK: WIPS model with query packing, Web and database servers on different machines

All requests correspond to an SQL query of the form "**select** * **from** R **where** $R.A = c_i$" against an ORACLE database. Table R is indexed on attribute $R.A$. The search value c_i is generated randomly. For the DISTR-PACK configuration, all queries are packed into a single SQL query with a "**where** $R.A$ **in** $(c_1, \ldots, c_n)$" clause. Figure 2 shows the results. The significant increase in elapsed time from LOCAL to DISTR is attributed to the communication overhead. However, by applying the query packing technique, DISTR-PACK managed to constrain the communication overhead to a minimum amount. Note that since both DISTR-PACK and DISTR transfer the same amount of data, the superior of DISTR-PACK is actually attributed to the reduction in the total number of messages transferred. It is also interesting to see that DISTR-PACK yields even better elapsed time than LOCAL. The explanation is that by packing several queries into one, the query processing and database access overhead is also reduced.

4 Related and Future Work

A brief survey and taxonomy on database gateway architectures were first given in [5]. Work addressing efficient architectural design for Web-Database connection can be found in [2, 4]. Our work differs from those systems in that we provide a generic framework for application development and perform optimization at query scheduling level.

The WIPS is an ongoing project with more functions to be added over the course of the development. Currently we have implemented an operational proof-of-concept prototype that incorporates a primitive query packing mechanism. There are several functions that we plan to add to the WIPS prototype in the next stage:

- Load Control: Refining the query packing mechanism to determine the most efficient packing size and scheduling of queries.
- Caching: Exploring caching technique to cache data at the database gateway so as to reduce data access and movement cost.
- Support of Transactional Applications: Incorporating a State Monitor (SM) into the DBAM to keep track of the state of a user transaction that requires successive services from the database servers. This mechanism is needed as the HTTP protocol is stateless.

References

[1] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann. World Wide Web: The information universe. *Electronic Networking: research, applications and policy*, 1(2), 1992.

[2] D. Eichmann, T. McGregor, and D. Danley. Integrating structured databases into the Web: the MORE system. In *Proc. of the 1st Int'l WWW Conference*, 1994.

[3] James Gosling, Frank Yellin, and The Java Team. *The Java Application Programming Interface Volume 2: Window Toolkit and Applets.* Addison Wesley, 1996.

[4] S. Hadjiefthymiades and D. Martakos. Improving the performance of CGI compliant database gateways. In *Proc. of the 6th Int'l World Wide Web Conference*, Santa Clara, CA, 1997.

[5] P.-C. Kim. A taxonomy on the architecture of database gateways for the web. In *Proc. of The 13th Int'l Conf. on Advanced Science and Technology (ICAST97)*, Schaumburg, IL, 1997.

[6] D. Robinson. The WWW Common Gateway Interface v1.1, 1996. Internet draft.