An Evaluation of the Java-Based Approaches to Web Database Access*

Stavros Papastavrou[1], Panos Chrysanthis[1], George Samaras[2], Evaggelia Pitoura[3]

[1] Dept. of Computer Science, University of Pittsburgh
{stavrosp, panos}@cs.pitt.edu
[2] Dept. of Computer Science, University of Cyprus
cssamara@cs.ucy.ac.cy
[3] Dept. of Computer Science, University of Ioannina
pitoura@cs.uoi.gr

Abstract. Given the undeniable popularity of the Web, providing efficient and secure access to remote databases using a Web browser is crucial for the emerging cooperative information systems and applications. In this paper, we evaluate all currently available Java-based approaches that support persistent connections between Web clients and database servers. These approaches include Java applets, Java Sockets, Servlets, Remote Method Invocation, CORBA, and mobile agents technology. Our comparison is along the dimensions of *performance* and *programmability.*

1 Introduction

Providing efficient and secure access to remote databases using a Web browser is crucial for the emerging cooperative information systems, such as Virtual Enterprises. A number of methods for Web database connectivity and integration have been proposed such as CGI scripts, active pages, databases speaking http, external viewers or plug-ins, and HyperWave [6]. These methods enhance the Web server capabilities with dynamic functionality for interactive and cooperative applications to create database connections, execute queries and transactions, and generate dynamic Web pages. However, there is an increasing interest in those that are Java-based due to the inherent advantages of Java, namely, platform independence support, highly secure program execution, and small size of compiled code, combined with a simple database connectivity interface (JDBC API) that facilitates application access to relational databases over the Web at different URLs [8].

Several Java-based methods are currently available that can be used for the development of Web cooperative information systems but in the best of our knowledge, there is no quantitative comparison of them in a database context. Existing studies either primarily focused on the various server side scripting mechanisms to support database connectivity (e.g., [5, 9]), or evaluated the Java client/server communication

* This work was partially supported by NSF IRI-9502091 and IIS-9812532, and AFOSR F49620-98-1-043 awards.

O. Etzion and P. Scheuermann (Eds.): CoopIS 2000, LNCS 1901, pp. 102–113, 2000.
© Springer-Verlag Berlin Heidelberg 2000

paradigm without any database connectivity or lengthy computations (e.g., [11]). This experimental paper contributes a comparison of the six Java-based approaches, specifically, Java applets using JDBC (Applet JDBC), Java Sockets [13], Java Servlets [4], Remote Method Invocation (RMI) [3], CORBA [10], and Java Mobile Agents (JMA) [2]. We focus on these methods because of their support for *persistent* database connections, which are essential for cooperative environments with long, and repeated data retrievals and updates.

For our evaluation, we used each approach to implement a Web client accessing and querying a remote database. Each approach differs in the way the client establishes connection with remote database servers with the help of a middleware and the implementation of the middleware. Depending on the way the client establishes connection with the middleware, the approaches can be classified as (1) *non-RPC* ones, that do not provide for remote method invocation mechanisms, (2) *RPC* ones with clear remote method invocation semantics, and (3) *RPC-like* ones involving mobile agent technology.

We compared the behavior of the different approaches along the following two dimensions: (1) *performance* expressed in terms of response time under different loads, and (2) *programmability* expressed in terms of the number of system calls at the client and the server site. The two salient results of our study are: (1) Best performance is not always achievable with high programmability and low resource requirements, and (2) the mobile agent technology needs to improve its programmability while giving particular emphasis in its infrastructure.

In the next section, we first discuss our experimental testbed and then elaborate on the implementation details of the six approaches under evaluation. In Section 3, we discuss our performance evaluation results whereas in Section 4, we compare the different approaches from programmability point of view.

2 The Experimental Testbed

We use each Java method to implement a Web client querying a remote database. Our testbed is structured along a three-tier *client/middleware/database* model. Two design principles were adopted in the selection of the various components during the development of the testbed. First, our Web clients should be lean for allowing fast downloads, and therefore increasing support for wireless clients. Second, no a-priori configuration of the Web client should be necessary to run the experiments in order to maintain portability, and therefore, support arbitrary clients. Thus, our Web client is a Java applet stored on a Web server. When the Java applet is downloaded and initialized at a client computer, queries can be issued through the applet's GUI to be executed on the remote database server (Figure 1). Our remote database server, a 3-table Microsoft Access, is on the same machine with the Web server.

The role of the middleware is to accept client requests, execute them on the database server, and return the results back to the client. Due to security restrictions of Java applets, part of the middleware has to execute on the Web server machine.