Web Services versus Distributed Objects: A Case Study of Performance and Interface Design

William R. Cook, University of Texas at Austin, USA
Janel Barfield, University of Texas at Austin, USA

ABSTRACT

Web services are promoted as a new model for distributed systems, yet many skeptics see them as simply a poor implementation of traditional remote procedure calls (RPC) or distributed objects. Previous comparisons, which show Web services to be significantly slower than RPC, are biased because they use a RPC communication style and ignore the document-oriented style that is recommended for Web services. We compare the straightforward design and implementation of a small file server application implemented using proxy-based RMI and document-oriented Web services. We show that Web services outperform RMI when accessing multiple/deeply nested files, especially over high-latency networks. However, the automatically generated Web service interfaces are awkward to use, so we develop a technique for wrapping the Web service to make it as easy to use as the distributed object implementation. This case study provides a more detailed comparison of the relationship between Web services and distributed objects.

Keywords: case study; distributed objects; performance; RMI; Web services

INTRODUCTION

There is significant debate on the relationship between Web services and distributed objects (Birman, 2004; Fowler, 2002; Vogels, 2003). Concrete case studies can help to clarify and quantify these differences. In this article, we take a step toward this goal by examining the performance and design of an example application using distributed objects and Web services.

Previous approaches to comparing performance of distributed objects and Web services have focused on the ability of Web services to perform traditional RPC–style interactions: invoking a single method whose arguments are simple primitive values or arrays (Davis...
Given the overhead of encoding and decoding XML, it is not surprising that Web services are an order of magnitude slower than distributed object implementations in CORBA (OMG, 1998), DCOM (Microsoft, 2006), or RMI (RMI, 2006). However, these studies are biased because they only measure RPC-style communication. They do not consider the possibilities of document-oriented designs that demonstrate the strengths of Web services.

In this article we do the opposite: consider a scenario that is better suited for a Web service implementation. The example application is a remote file server based on the HTTP protocol (Fielding et al., 1999). The question considered is, What if HTTP had been defined using distributed object middleware (RMI/DCOM/CORBA) or as a Web service? The resulting designs are implemented and evaluated for performance and usability.

Object-oriented analysis and design (Booch, 1994) of this application produces a clean object-oriented interface to the repository of files. The interface is easily converted to remote invocation over RMI using automatic proxies. There is a tight coupling between clients and servers, since the client must also have access to the interfaces used to program the server. More significant, the design results in a large number of round-trips to the server, causing poor performance as client transactions become more complex—for example, in downloading multiple files or increasing path length. Any experienced designer of distributed systems would recognize this as a poor design; yet it is certainly a very natural design using the facilities of RMI. We consider more sophisticated designs in the fourth section.

The Web service is defined by creating java classes implementing a service that processes requests to produce responses. These classes are translated to a WSDL (Web Service Description Language) file and corresponding server-side wrappers using Apache Axis in the Eclipse Web Tools Project (Apache Software Foundation, 2002; Christensen, Curbera, Meredith, & Weerawarana, 2001; Holzner, 2004). The client was created similarly by importing the WSDL. The Web service approach naturally reduces the number of round trips.

We evaluate the performance of each implementation in loading batches of files over networks with a range of latencies. The pure-object RMI implementation is faster for small batches of documents and low-latency networks, but performance degrades rapidly with larger batches and higher latency. The Web service implementation has a high initial cost but shows little or no change with larger batches. Higher latency creates a greater initial cost, but performance is still independent of batch size. As latency increases, the performance benefits of the document-oriented approach increase significantly. This is relevant when in some real world scenarios, latency may even be minutes, hours, or days, as for disconnected or asynchronous workflow processes.

Unfortunately, the Web service client code is awkward to use: The programmer must manipulate request and response structures rather than directly perform operations on server objects as in the RMI implementation.

In short, the most natural designs for distributed objects are easy to use but scale poorly, while Web services have good scaling properties but are awkward to use. To address this problem, we create better client wrappers for the Web service implementation, using the same interfaces implemented in the distributed object model. One additional call is added to the interface, and a change is made to the time at which values are available and exceptions are raised. We also show how these wrappers can be reused to convert the RMI solution to use a document-oriented communication style based on mobile code and value objects. The resulting RMI implementation is very fast—but it is also quite complex, does not make significant use of automatic proxies (it is stateless except for a connection to a singleton server object), and is still platform specific. Both the final Web service and RMI implementations use the communication style that is natural to Web services,
Automatic Adaptable Intentional Service Generating and Publishing Framework using OWL-S Annotation
www.igi-global.com/article/automatic-adaptable-intentional-service-generating-and-publishing-framework-using-owl-s-annotation/193859?camid=4v1a