Distributed remote architectures CORBA

HASRAT JAHAN¹ and MANMOHAN SINGH²

Department of Biotechnology, Bhopal Degree College Bhopal, (India).
Department of Computer Science, BIST, Bhopal (India).

(Received: April 12, 2010; Accepted: June 04, 2010)

ABSTRACT

An emerging trend in the Signal and Image Processing (SIP) community is the appearance of middleware and middleware standards that can be readily exploited for distributed computing applications by the SIP community. High performance computing and High Performance Embedded Computing (HPEC) applications will benefit significantly from highly efficient & portable computational middleware for signal & image processing. Open middleware standards such as VSIPL, MPI, CORBA, encoding and SOAP-based messaging protocols. More specifically, we will be focused on the appropriate use of such technologies for implementing new SIP applications, or extending legacy applications through the use of these technologies. The three middleware standards we have selected all have certain commonalities. All are based around the concept of a client application using the services available on a remote machine, or server. A remote executable object that implements one or more exposed interfaces provides these services. The object’s interface represents a contract between the client and the server. This interface is written as a Java interface for Java RMI, in IDL for CORBA, and in WSDL for web services. In the latter two cases, the more generic descriptions can be translated into specific language implementations.

Keywords: Remote Architectures, CORBA.

INTRODUCTION

Transport Layer makes the connection between JVM’s. All connections are stream-based network connections that use TCP/IP. Even if two JVM’s are running on the same physical computer, they connect through their host computer’s TCP/IP network protocol stack.

Naming Remote Objects
Now you’ll find the answer to that question. Clients find remote services by using a naming or directory service. This may seem like circular logic. How can a client locate a service by using a service? In fact, that is exactly the case.

Distributed Garbage
One of the joys of programming for the Java platform is not worrying about memory allocation. The JVM has an automatic garbage collector that will reclaim the memory from any object that has been discarded by the running program.

Serializing Remote Objects
When designing a system using RMI, there are times when you would like to have the flexibility to control where a remote object runs. Today, when a remote object is brought to life on a particular JVM, it will remain on that JVM. You cannot “send”
Mobile Agent Architectures
The solution to the mobile computing agent using RMI is, at best, a work-around. Other distributed architectures have been designed to address this issue and others. These are collectively called mobile agent architectures.

CORBA pros and cons
CORBA is gaining strong support from developers, because of its ease of use, functionality, and portability across language and platform. CORBA is particularly important in large organizations, where many systems must interact with each other.

Common Object Request Broker Architecture
Pros
- Services can be written in many different languages, require the use of an interface definition language (IDL) which must be learned.
- Implementing or using services require an IDL mapping to your required language - writing one for a language that isn’t supported would take a large amount of work.

Cons
- With IDL, the interface is clearly separated from implementation, and developers can create different implementations based on the same interface.
- CORBA supports primitive data types, and a wide range of data structures, as parameters.
- CORBA is ideally suited to use with legacy systems, and to ensure that applications written now will be accessible in the future.

Remote method invocation
Pros
- Portable across many platforms
- Can introduce new code to foreign JVMs
- Java developers may already have experience with RMI (available since JDK)

Cons
- Tied only to platforms with Java support
- Security threats with remote code execution, and limitations on functionality enforced by security restrictions
- Learning curve for developers that have no RMI experience is comparable with CORBA
The availability of 2-tier application builders has helped perpetuate this philosophy. The 2-tier model is not a “bad thing,” but there are cases in which the 3-tier model would be a better choice.

CONCLUSIONS

CORBA doesn’t reveal an optimum solution - one is not “better” than the other. The properties of these two technologies lend themselves to different situations. A comparison of RMI and CORBA helps to highlight individual strengths and weaknesses, but the applicability of one technology over the other depends largely on the purposes for which it is to be used, the experience of the developers who will design, implement and maintain the distributed system, and whether non-Java systems are intended to access the system now or in the future.

REFERENCES