Performance comparison of remote method invocation and web services as communication technologies to build distributed applications

DANIEL F. GARCIA, CHRISTIAN URIA, JULIO MOLLEDA, IVAN PETEIRA
Department of Informatics
University of Oviedo
Campus de Viesques, 33204 Gijón
SPAIN

Abstract: - In order to communicate the processes that constitute the current distributed applications, remote object technologies are currently used, but recently, XML Web services arises as a new and promising technology. This article focuses on comparing the relative performances of both communication technologies.

Key-Words: Distributed Computing Environments, Communication Performance, RMI, Web Services

1 Introduction

The computing industry relies on two predominant technologies to communicate the processes of distributed applications. The first technology is called Remote Invocation Method (RMI), and it is commonly implemented in the form of middleware (as CORBA), or integrating it in virtual machines (as Java or .NET). The second technology is known as XML Web Services and it is frequently implemented within virtual machines.

In general, the RMI technology allows a versatile and highly customizable communication between remote objects, allowing the use of different transport protocols and serialization formats. But the building of complex distributed applications with this technology is not easy. The designers of the client objects must obtain “manually” detailed information of the server objects in order to build correct invocations of the remote methods exposed by the server objects.

On the other hand, Web Services technology has supports fewer possibilities for customizing the communications. But it provides powerful mechanisms for the designers of client objects find the remote methods in the Web and learn how to use them using a Web browser. The designers can also retrieve the description of the server objects and generate “automatically” all the necessary code to make remote invocations.

The development of distributed applications using Web Services technology is easier and faster than using the other technologies, but developers claim that the performance of distributed applications based on Web Services is worse.

This paper presents a comparison of these technologies to asses the relative performance of Web Services versus the Remote Method Invocation.

2 Related Work

The performance evaluation of the communication mechanisms for distributed applications is a permanent open issue. The initial research works were focused on RPC technologies. Further research focused on communication middleware, and particularly in CORBA, covering multiple aspects: the formalization of good benchmarking principles and practices [1], the development of benchmark suites [2], the correct interpretation of benchmarking results [3] and the utilization of benchmarking to detect regression in the functionality of distributed applications [4].

In recent years, the stand-alone communication middleware, as CORBA, is losing interest for researchers, which are focusing their attention in the communication technologies [5] integrated into the two predominant development platforms for distributed applications, .NET and Java.

Some research works focused on the evaluation of the performance of Web Services, with emphasis on the SOAP layer [6] [7], or on the encoding layer [8].

As Web Services, or SOAP, shown a lower performance than other specific communication technologies, important research efforts have been oriented to evaluate the applicability of SOAP to specific application domains, like scientific computing [9][10] or trading systems [11].

Multiple research efforts have been oriented to compare the communication methods (remote method invocations, web services, etc.) in Java platforms [12] and equivalent comparisons have been developed in the .NET platform [13] in order to choose the more appropriate technology to build a particular distributed application [14]. There are also research works that consider the two main development platforms in the same evaluation [15].

Although there are multiple research works about the performance of distributed applications, most of them do not evaluate jointly all aspects of the communication technologies currently used.

The work presented in this paper differs from previous research in that it is totally focused on the comparison of communication mechanisms used to build distributed applications. To reach this goal, the experimental procedure has been designed to minimize the effect of all factors non-related to the communications on the performance metric.

3 Experimental platform and design

In this section we present the design of experiments and the computing platform used. The primary goal of the experiments is the evaluation of the performance differences between the two communication technologies used to build distributed applications.

In this evaluation work we have selected the invocation latency as the response metric. It reflects very clearly the communication latency because the processing involved in each invocation is minimal and the storage work is null.

As the factors capable of affecting the latency we have selected the following:

1. The size of the variables interchanged between a client and the server. It is expected that the latency increases linearly in relation to the size of the variables.
2. The formatter type used to codify the variables to pass into a stream. Generally, two formatters can be used: text formatters and binary formatters. The Binary Formatter serializes data to a compact, proprietary format, although it does not compress the data. The text (SOAP) Formatter serializes data to a SOAP message, which is a cross-platform XML-based plain-text format. Because clients on other platforms can create and send SOAP messages, this formatter can allow cross-platform communication although it generates larger size messages. It is expected that binary formatters perform better (faster) than text formatters, mainly when the variables passed are numerical types.
3. The channel type used to transmit the stream. Two types are commonly used: TCP channels and HTTP channels. The TCP channel uses the connection-based TCP protocol and it is ideal for internal networks. The HTTP channel uses the HTTP protocol, which is ideal to communicate across the Internet. HTTP is based on the TCP/IP protocol, but it does not require a continuous connection. Instead, it communicates using request and response messages. It is expected that TCP channels perform better (faster) than HTTP channels.
4. The concurrency level supported by the client-server system. This factor is the number of concurrent requests (invocations) that are being processed in the system. As the clients do not wait any time between two successive requests, the concurrency level will be, approximately, equal to the number of active threads in the client machine.
5. The mechanism used to serve the request. For example, a server can create an object for each received request. It can also create an object for each client (user) which will serve all the requests of that client. The server can also use a unique object to serve all the requests of all clients. Finally, the server can be built hosting the server object into a Web server.

These five factors are called the primary factors in the evaluation procedure. The quantification of their effects on the response metric is of primary interest for the designers of distributed systems.

Of course, there are many secondary factors that could affect significantly the communication latency. Many of then are related with the computing platform used to carry out the evaluation. For example, the processor type and its speed, the network type and its bandwidth, the operating system, the virtual machine used, etc.

However, we are interested in the variation of latencies when the values of the primary factors change. Although these variations could be different for different values of the secondary factors, we will develop the evaluation work using common (average) values for these secondary factors.

We have selected one, and only one, computing and development platform (with its associated virtual machine) to carry out the experimental work. In this evaluation we selected the .NET platform, but the same experimental design could have been developed using Java or CORBA.

The client and server machines are identical, using a Pentium III 850 Mhz processor, and running the Windows 2000 operating system. The 100 Mbps Ethernet adapters of both machines are directly connected with a twisted Ethernet cable. By this manner, they have available the full Ethernet bandwidth during the experiments.

The server application hosts an object and it creates a single or multiple copies of the object to serve the requests of the clients. The object has a single method that receives a variable-length array of integers, inverts them and finally, returns the array (to the client). Note the total absence of access to data on disk, in a format of files or databases. This is intentionally done to focus the evaluation on the communication mechanisms, trying to avoid the presence of experimental factors that could affect the response metric much more significantly than the communication mechanisms.

The client application is composed by a set of independent threads. Each thread makes a sequence of 1000 invocations of the remote method. The system time is measured just before and just after the invocation of the remote method, using the high-resolution performance counter, which has a typical resolution of one microsecond. The latency of each invocation (the difference between the second and the first measurements) is recorded in memory, and when the load experiment finishes they are stored on disk. There is no think time between the successive requests, so the real concurrency level supported by the server is, approximately, equal to the number of active threads in the client application. This is an essential aspect of the evaluation work presented in this article. Many other evaluation works use a think time, and therefore, the load rate injected in the distributed system is totally dependent of this parameter.

To carry out the experimental work a full factorial experimental design has been accomplished. Firstly, we evaluated the RMI technology using all possible combinations of the values for the categorical or qualitative factors and a selection of proper values for quantitative factors. Secondly we evaluated the Web Services technology using the combination allowed by this technology.

The table 1 summarizes the factors and the values selected for experimentation.

Factor	**Values**
Formatter type	Text (SOAP), Binary
Channel type	TCP, HTTP
Server object type	SingleCall, Singleton, CAO, WS
Data payload (Int32)	1, 10, 10^2, 10^3, 10^4, $5x10^4$, 10^5
Concurrency level	1, 5, 10, 20, 30 40, 50

Table 1 Factors and levels of the experimental design

The single response metric selected is the latency of invocations. Other comparison and evaluation works also use an additional throughput metric. However the throughput is highly influenced for multiple factors different from communication mechanisms, such as the power of client and server machines and the type of service provided by the object server. As this work focuses in the communication subsystem, all the comparison will be done using the communication latency exclusively.

A single value of latency is obtained from each measurement experiment averaging all measured latencies, typically 500 requests multiplied by the number of threads used in the experiment. The initial and final measured latencies are not considered in the calculation of the average latency.

The latency values obtained from 637 experiments are represented in figure 1, which is composed by 13 graphs. Each graph contains 7 curves that represent the latency as a function of the number of integers transmitted (marshaled), one curve for each concurrency level considered.

Within the figure 1, the graphs are organized like in a matrix. The columns indicate the type of server object: Left column for “Singleton”, center column for “SingleCall” and right column for “ClientActivated”. The rows indicate combinations of formatter + channel types. The first row for “Binary + TCP”, the second row for “Binary + HTTP”, the third row for “Text(SOAP) + TCP” and the fourth row for “Text(SOAP) + HTTP”. This matrix-like organization is valid for the RMI technology. The web services only operate with “Text + HTTP” in a manner similar to SingleCall. Therefore its graph has been placed just below the most similar RMI configuration.

4 Performance comparison

In this section we compare the performance of the different configurations of RMI, between them and with the web services, using the latency metric.

Firstly we carry out a qualitative comparison based on a visual analysis of the graphs of figure 1 and later a quantitative comparison based on analytical models obtained from the graphs.

In figure 1, the configurations are ordered from the minimum latency (at the top) to the maximum (at the bottom). It is very interesting to notice that the two faster configurations use the binary formatter and the two slower configurations use the SOAP formatter, and this independently of the type of communication channel used. These results allow deducing that the type of formatter used has much more influence on the latency than the type of the channel. Furthermore, for the same formatter, the latency is always lower with the TCP channel than with the HTTP channel. These conclusions are obtained comparing the elements of figure 1 along the vertical direction.

Figure 1 also shows that the influence of the type of object used to provide the services requested by the clients have very little influence in the latency. This conclusion is obtained comparing the elements of figure 1 along the horizontal direction.

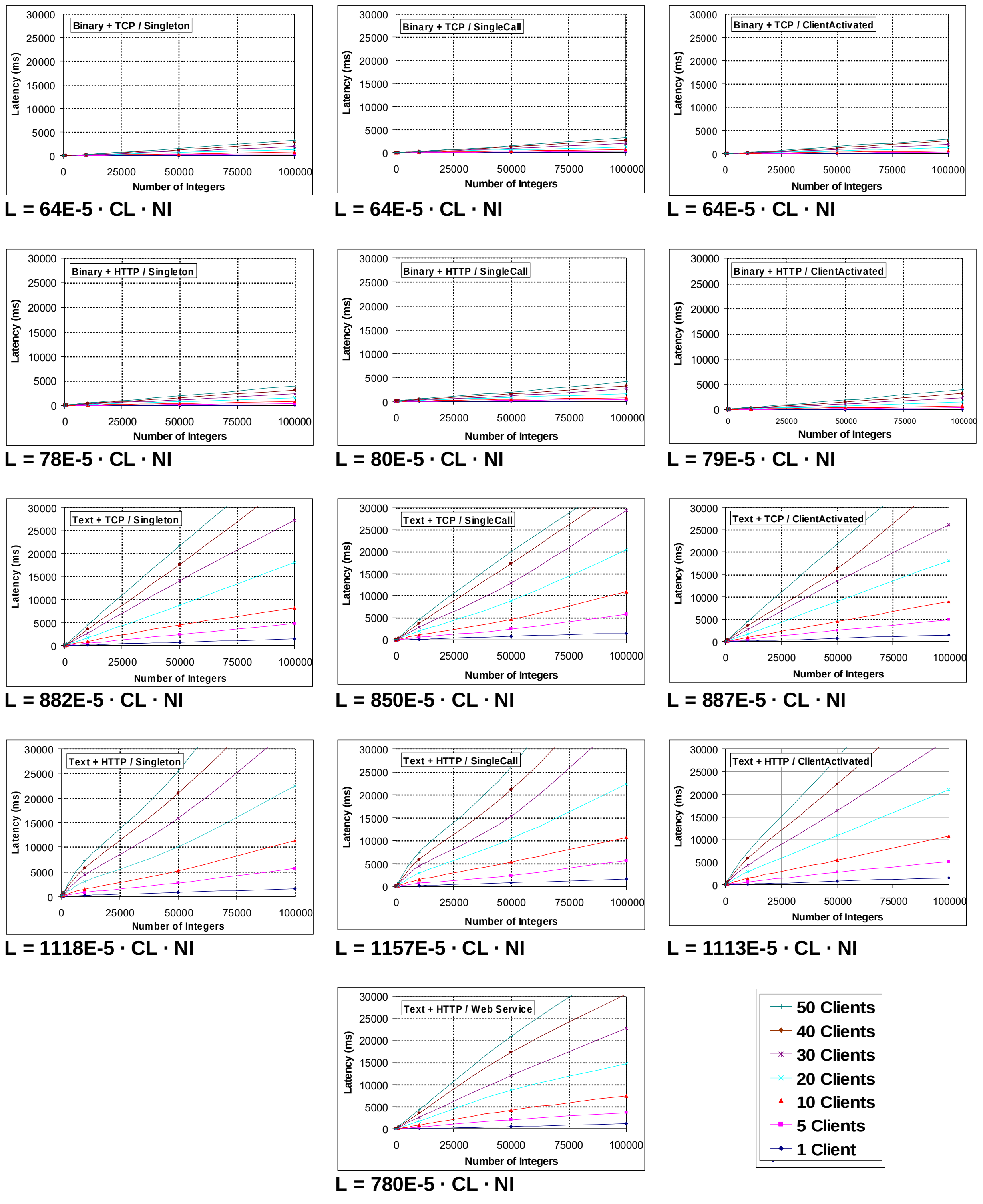

Fig.1 Average latency of invocations as a function and the effective concurrency level (CL) supported in the system and of the number of integers transmitted (NI).

In order to compare the measurements represented in figure 1 qualitatively, an analytical model for the latency has been developed. The model was developed following two successive steps.

In the first step, seeing the curves of the graphs of figure 1, we can conclude that the evolution of latency as a function of the number of integers transmitted seems to follow a linear relationship. Therefore the linear model (1) is proposed, in which, the latency L is directly proportional to the number of integers marshaled, NI.

$$L = C \cdot NI \qquad (1)$$

A different constant C for each curve of each graph of the figure 1 is obtained using a linear regression technique. The coefficient of determination, R^2, was higher than 0.99 in the 91 regressions, indicating this, that equation (1) fits the data very well.

In the second step, we represented the seven C constants of each graph of figure 1 as a function of the concurrency level. All the representations are totally linear and start from the origin. Therefore the new linear model represented in equation (2) was proposed.

$$C = K \cdot CL \qquad (2)$$

A specific parameter K for each graph of the figure 1 is obtained using a linear regression technique. In this step, 13 regressions were carried out, and all of them showed a coefficient of determination, R^2, higher than 0.98, assuring that equation (2) fits the data very well.

Combining the two linear models, we obtain a very simple, but very accurate, multiplicative model represented by the equation (3).

$$L = K \cdot CL \cdot NI \; (3)$$

The factor K has a specific value for each graph of figure 1. Therefore, there is a model for each combination of the two main factors of the communication technology (formatter type + channel type) and for the type of server object used. The model obtained for each graph of the figure 1 has been included in this figure just under its correspondent graph.

As the analytical models always provide an accurate prediction of the latency, we can compare the different configurations using the models.

Firstly, we analyze the influence of the server object type on the latency. Comparing the constants of the three models of each horizontal line in the figure 1 (same formatter and channel) we can see that the differences are negligible. Therefore a designer of a distributed application can select the most appropriate object type for a particular application without considering in detail the implications of the selection on performance.

Secondly, we analyze the joint influence of the formatter and channel types on latency. The main goal is to obtain an index of the relative latencies obtained when these configuration factors change. Absolute values are much less useful because they depend on the computational power of the platform used to carry out the measurements.

As the fastest communication technology is RMI configured with the Binary formatter on a TCP channel, it is used as the comparison base for the latencies obtained with other configurations.

Dividing the constants of each model between the constant of the fastest model, we can obtain directly the latency increment of each configuration with respect to the fastest one. Figure 2 shows the results.

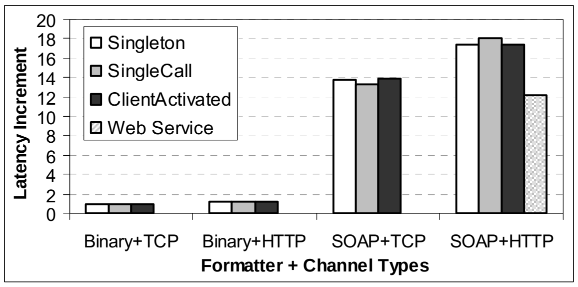

Fig.2 Latency increment as a function of the main configuration parameters of communications

Whit the binary formatter, the use of the HTTP channel instead of the TCP channel only multiplies the latency by 1.23. The programmer can select the appropriate channel without taking special care about the performance effects.

But the largest increment of latency occurs when SOAP formatter is used. In this case the latency is multiplied by a factor of about 14 using the TCP channel and about 18 using the HTTP channel. The decision of which type of formatter must be used for an application affect drastically the performance of the final design and must be considered carefully.

With the SOAP formatter, the utilization of the HTTP channel instead of the TCP channel only multiplies the latency by 1.29. In this case, the designer of a distributed application can select the appropriate channel freely if an increment of 30% in latencies can be tolerated.

The most surprising conclusion obtained in this work is the latency observed in web services in relation to the latency of RMI (SOAP + HTTP).

The latency of RMI operating with SOAP+HTTP is 1.45 times the latency of web services, which also operate with SOAP+HTTP. Furthermore, the latency of RMI operating with SOAP+TCP is 1.12 times the latency of web services. Therefore, web services are the best communication technology when a SOAP formatter must be used.

Figure 3 summarizes the relative performance of all configurations of the communication technologies evaluated on this work.

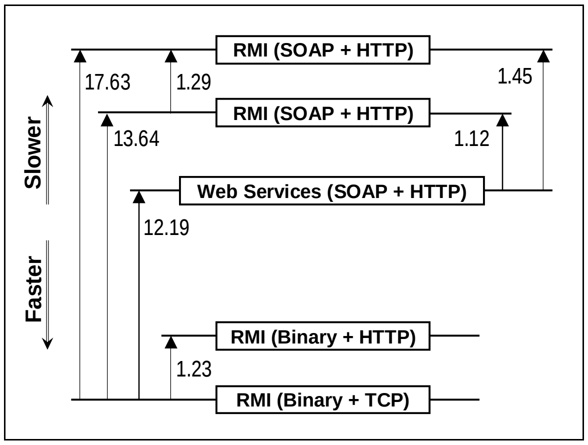

Fig.3 Performance ratios of all configurations

5 Conclusion

A simple but effective manner of comparing the performance of Remote Method Invocation and Web Services technologies has been proposed and implemented. The conclusions are condensed in a few performance ratios, which allow the designer of distributed applications to asses the effect of selecting a particular communication technology on the performance of communications.

The conclusions of this work will be extended, evaluating the effect of using different data types and other computing platforms.

References:

[1] Object Management Group, *White Paper on Benchmarking*, OMG, Framingham, USA, 1999.

[2] Petr Tuma, Adam Buble, Open CORBA Benchmarking, *Int. Symp. on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'01)*, Florida, USA, 2001.

[3] Adam Buble, Lubomir Bulej, Petr Tuma, CORBA Benchmarking: A Course with Hidden Obstacles, *International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003, pp.279-284.

[4] Lubomir Bulej, Tomas Kalibera, Petr Tuma, Regression Benchmarking with Simple Middleware Benchmarks, *Int. Workshop on Middleware Performance (IPCCC'04)*, Phoenix, AZ, USA, 2004, pp.771-776.

[5] Peter Hrastnik, Comparison of Distributed System Technologies for E-Business, *2nd Int. Interdisciplinary Conf. on Electronic Commerce*, Gdansk, Poland, 2002, pp.49-56.

[6] Dan Davis, Manish Parashar, Latency Performance of SOAP Implementations, *IEEE Cluster Computing and the Grid (CCGRID'02)*, Berlin, Germany, 2001, pp.771-776.

[7] Alex Ng, Shiping Chen, Paul Greenfield, An Evaluation of Contemporary Commercial SOAP Implementations, *5 Australasian Workshop on Software and System Architectures (AWSA'04)*, Melbourne Australia, 2003, pp.64-71.

[8] Min Cai, Shahram Ghandeharizadeh, Rolfe Schmidt, Saihong Song, A Comparison of Alternative Encoding Mechanisms for Web Services, *DEXA (DEXA'02)*, Aix-en-Provence, France, 2002.

[9] Madhusudhan Govindaraju, Aleksander Slominski, Venkatesh Chopella, Randall Bramley, Dennis Gannon, Requirements for and Evaluation of RMI Protocols for Scientific Computing, *ACM-IEEE Conf. on Supercomputing*, Dallas, TX, USA, 2000, Article 61.

[10] Kenneth Chiu, Madhusudhan Govindaraju, Randall Bramley, Investigating the Limits of SOAP Performance for Scientific Computing, *11th IEEE Int. Symposium on High Performance Distributed Computing (HPDC'02)*, Edinburgh, Scotland, 2002, pp.246-254.

[11] Christopher Kohlhoff, Robert Steele, Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems, *12 Int. World Wide Web Conference (WWW2003)*, Budapest, Hungary 2003, Paper 872.

[12] Christophe Demarey, Gael Harbonnier, Romain Rouvoy, Philippe Merle, Benchmarking the Round-Trip Latency of Various Java-Based Middleware Platforms, *Component & Middleware Performance Workshop (OOPSLA'04)*, Vancouver, Canada, 2004.

[13] Priya Dhawan, Performance Comparison: .NET Remoting vs. ASP.NET Web Services, On-line in http://msdn.microsoft.com, 2002.

[14] Priya Dhawan, Tim Ewald, ASP.NET Web Services or .NET Remoting: How to Choose, On-line in http://msdn.microsoft.com, 2002.

[15] Mario Stefanec, Sinisa Srbljic, Ivan Skuliber, Performance Evaluation of Distributed Object Platforms for Public Information System Implementation, *7th World Multiconference on Systemics Cybernetics and Informatics (SCI'03)*, Orlando July, 2003.