10º Congreso Nacional de Mecatrónica
Noviembre 3 y 4, 2011. Puerto Vallarta, Jalisco.

Implementación de Servicios de Computo Distribuido Mediante Java-RMI, CORBA y Web-Services

Sotomayor Olmedo Artemio[1], Barriga Rodriguez Leonardo[2], Jiménez Hernández Hugo[2], Moya Morales J. Carlos[1] y Pedraza Ortega Jesús Carlos[1]
artemiosotomayor@gmail.com

[1]Universidad Autónoma de Querétaro (UAQ)
[2]Centro de Ingeniería y Desarrollo Industrial Querétaro (CIDESI-QRO)

Resumen

El uso de sistemas de computo han ampliamente utilizado en aplicaciones de índole, científica, fabril y de negocios. La idea principal de los sistemas de computo distribuidos es utilizar una arquitectura cliente-servidor en combinación con la invocación de métodos remotos. La invocación de estos métodos consiste en enviar datos de un cliente hacia un servidor para su proceso, y posteriormente el servidor devuelve los datos al cliente para su presentación. En general las plataformas distribuidas Java-RMI, CORBA y Web Services, proveen una serie de métodos para comunicarse con objetos distribuidos en equipos remotos. Este trabajo presenta un estudio comparativo(mediciones de uso de procesador, memoria, paquetes sobre la red, etc.) entre: Java-RMI, CORBA y Web Services, implementando dos servicios distribuidos, una calculadora y el envío de una cadena de caracteres.

Palabras clave: sistemas de computo distribuidos, invocación de métodos remotos, Java-RMI, CORBA y Web Services.

1. Introducción

Los servicios de computo distribuido son sistemas de software que permiten el intercambio de datos y funcionalidad entre aplicaciones sobre una red. Esta soportado en diferentes estándares que garantizan la interoperabilidad de los servicios. Los servicios web utilizan como su gran insumo el lenguaje extensible de marcado XML(*Extensible Markup Languaje*) y se basa en una arquitectura en la que se define el servicio web a través de uno de los lenguajes estándar se publica en un directorio donde se halla la descripción anteriormente hecha y se utiliza de acuerdo a las expectativas de resolver una necesidad de acuerdo con la descripción provista.[1] Mientras que CORBA(*Common Object Request Broker Architecture*,)[3] utiliza transformaciones, repositorios. Java-RMI implementa un esquemas sencillo de llamadas a métodos remotos mediante el uso de *Helpers (*o ayudantes*)*[2]. La arquitectura que mejor se ha adaptado al mundo de los servicios de computo distribuido es SOA (*Service Oriented Architecture*) brindando un enfoque que ha adoptado los negocios y ha incrementado el intercambio electrónico de datos en el comercio electrónico y procesos fabriles[5][6]. Se plantea como problema la ausencia de arquitecturas que permitan a los sistemas distribuidos medianos y pequeños ingresar sus organizaciones al esquema de Orientación a Servicios y Procesos de Negocios.

2. Java-RMI

Java-RMI por sus siglas en ingles *Java Remote Method Invocation*, es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java.[3]

RMI se caracteriza por la facilidad de su uso en la programación por estar específicamente diseñado para Java; proporciona paso de objetos por referencia (no permitido por SOAP), recolección de basura distribuida (Garbage Collector distribuido) y paso de tipos arbitrarios (funcionalidad no provista por CORBA).[3]

A través de RMI, un programa Java puede exportar un objeto, con lo que dicho objeto estará accesible a través de la red y el programa permanece a la espera de peticiones en un puerto TCP. A partir de ese momento, un cliente puede conectarse e invocar los métodos proporcionados por el objeto.[5]

3. CORBA

CORBA por sus siglas en ingles *Common Object Request Broker Architecture*, provee un conjunto Interfaces de programación, un protocolo de comunicaciones y los mecanismos necesarios que permiten la interoperabilidad entre aplicaciones, lenguajes y plataformas heterogéneas, aspectos fundamentales en la computación distribuida.[2]

EL lenguaje de definición de interfaces (IDL) utilizado por CORBA permite especificar las interfaces y servicios que los objetos cliente-servidor ofrecerán, esto en función de un lenguaje determinado, especificando cómo los tipos de dato deben ser utilizados en las implementaciones cliente-servidor. Algunos lenguajes soportados por CORBA son:, C, C++, , Java, Python, Perl, Ruby y Tcl [2][9].

Al compilar una interfaz en IDL se genera código para el cliente y el servidor. El código del cliente sirve para poder realizar las llamadas a métodos remotos y es conocido como *stub*. El código del servidor conocido como *skeleton* debe implementar todos los métodos que serán invocados remotamente.

CORBA, en de manera general es considerado un middleware pues también provee mecanismos de seguridad y transacciones.

4. Web Services

La World Wide Web Consortium (W3C) lo define como "...un sistema de software diseñado para soportar interacción interoperable máquina a máquina sobre una red[1]. Este tiene una interface descrita en un formato procesable por una máquina. Otros sistemas interactúan con el servicios web en una manera prescrita por su descripción usando mensajes *SOAP*, típicamente enviados usando *HTTP* con una serialización *XML* en relación con otros estándares relacionados con la web"[1][8]. Actualmente los *web services* han sido ampliamente utilizados gracias es la adopción de protocolos y estándares abiertos. Al conjunto de estándares y protocolos para los servicios web se le conoce como: *Web Services Protocol Stack* y permite definir, localizar, implementar y comunicar un servicio web con otro y está conformado por: Servicio de transporte, mensajería xml, descripción del servicio y descubrimiento de servicios.

4.1 Servicio de transporte

Es el servicio responsable del transporte de los mensajes entre aplicaciones sobre la red e integraa varios protocolos del nivel de aplicación como: *http(HyperText Transfer Protocol), smtp (simple mail transfer protocol), BEEP(Block Exensible Exchange Protocol),* etc.

4.1.1 HTTP

Es el protocolo que define la sintaxis y la semántica utilizada en la arquitectura web. En el contexto de los *web services* es utilizado para la transferencia de las transacciones XML en la red utilizando los mismos principios del XHTML/HTML.

4.1.2 SMTP

Es el estándar de la capa de aplicación utilizado para el envío de mensajes de correo electrónico a través de Internet y que requiere como cliente software de tipo POP3 o IMAP.

4.1.3 BEEP

Protocolo de nivel aplicación , también conocido como *BXXP*, diseñado para la interacción asíncrona punto a punto sobre una red *TCP/IP* y provee un marco para dirigir las conexiones punto a punto, centrándose en aspectos de autenticación, transporte de mensajes y manejo de errores.

4.2 Mensajería XML

Componente responsable de la codificación de los mensajes en XML estándar permite la interpretación de cualquier mensaje en los equipos de la red. Los componentes más utilizados en la mensajería xml son: *REST(Representational State Transfer), RPC(Remote Procedure Calls), XML-RPC.*

4.2.1 REST

En resumen, es un conjunto de principios para el diseño de redes, que es utilizado comúnmente para definir una interfaz de transmisión sobre HTTP de manera análoga a como lo hace SOAP. Aunque REST como tal no es un estándar, posee un conjunto de estándares tales como HTML, URL, XML, GIF, JPG y tipos MIME.

4.3 Descripción del servicio

Es una interfaz pública la cual es descrita por un formato llamado *WSDL (Web Services Descripción Languages).*

4.4 Descubrimiento de Servicios

UDDI es un marco independiente de la plataforma para describir servicios, negocios e integrar servicios de negocios. La estructura de UDDI está basada sobre los servicios estándares de la web, lo que quiere decir que UDDI es accesible como otros servicios web. UDDI es un esfuerzo de la industria iniciada en Septiembre de 2000 por Ariva, IBM, Microsoft y otras 33 compañías [9]. Los propietarios de los Servicios Web los publican en el registro UDDI. Una vez publicados se mantienen allí apuntadores a la descripción del Servicio Web y al servicio. UDDI permite a los clientes buscar tal registro, encontrar el servicio deseado y extraer sus detalles. Estos detalles incluyen el punto de invocación así como otras características del servicio y su funcionalidad.

5. Arquitectura Orientada a Servicios en Sistemas distribuidos

En general para hablar de arquitectura orientada a servicios y sistemas distribuidos se deben cumplir lo siguiente[4][5]:

Vista lógica: Es una vista que proporciona una imagen de los componentes del sistema tales como bases de datos, procesos de negocios, programas, etc., explicando que hace cada uno de ellos, normalmente llevándolos a la operación del nivel del negocio.

Orientación al mensaje: Se define el servicio en términos de los mensajes intercambiados por el agente solicitante y el agente proveedor. En SOA es abstraído algunas características de los agentes tales como la estructura de la base de datos, sus lenguajes de implementación, estructuras de procesos, etc. SOA se preocupa por los detalles que son expuestos en la descripción del servicio.

Orientación a la descripción: Un Servicio Web es descrito por metadatos procesables por maquina. La descripción debe soportar la naturaleza pública de la SOA. La semántica del servicio debe ser definida completamente en su descripción.

Granularidad: Los servicios deben tener la tendencia a realizar un pequeño número de operaciones con una gran cantidad de mensajes.

Orientación a la red: Los Servicios Web deben conservar la tendencia de ser concebidos para ser usados sobre una red, sin embargo no es un requerimiento absoluto.

Plataforma Neutral: Los mensajes deben ser creados para una plataforma neutral, utilizando un lenguaje estándar a través de las interfaces. XML es el lenguaje que mejor cumple con esta restricción.

6. Caso de Estudio

Las tecnologías en este caso de estudio fuerón:

- JAXRPC from Sun's Java Web Services
- Development kit (JWSDP) [7];
- Tomcat 7 (as incorporated in JWSDP) [8];
- Java RMI from Java 1.6 SDK [2];
- Java CORBA from Java 1.6 SDK [9];
- Whireshark network traffic analysis program [10].

Las pruebas fuerón realizadas en una LAN(Local Area Network) con una velocidad de 100Mb/s. Las pruebas de estrés y rendimiento se llevarón a cabo con un equipo de escritorio DELL Optiplex 740(2.3Ghz), 2Gb RAM y sistemas opetativo GNU/Linux Ubuntu 10.04 haciendo las veces de cliente cliente y como servidor se utilizo una macbook pro 5,5, a 2.3Ghz, 8Gb RAM y sistemas opetarivo OS-X Lion.

Estas mediciones fueron realizadas en un laboratorio realizando exclusivamente las pruebas correspondientes con cada una de las tecnologías de forma aislada, de tal forma que el procesador, memoria y latencia en los mensajes no fuese interrumpida por procesos ajenos a este experimento.
Las mediciones a realizar son:

- Memoria utilizada en el cliente.
- Uso de procesador en el cliente.
- Uso de procesador en el servidor.
- Latencia en una petición sencilla
- Latencia en peticiones múltiples.
- Total de bytes transferidos.
- Conteo de paquetes

Las aplicaciones utilizadas en este trabajo son:

- Una calculadora simple.
- El envió de una cadena de texto.

Estas dos aplicaciones fuerón elegidas por su simple implementación y representan la base de transacciones y cálculos más complejos. Finalmente la implementación sigue los pasos de la figura 1, donde en el rectángulo de "*Implementation*" se añaden las implementaciones particulares de cada tecnología.

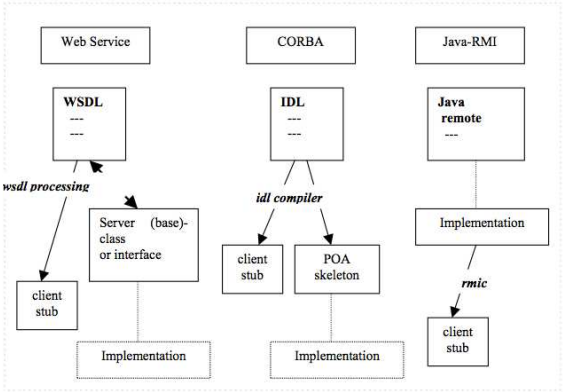

Fig. 1. Arquitectura general de Web Services, CORBA y Java-RMI.

7. Resultados

A continuación se muestras las mediciones obtenidas a partir de la implementación de los casos de estudio de una calculadora y el envió de una cadena de caracteres.

Tabla 1: Resultados de Latencia y tráfico en una petición simple.

Tecnología	Latencia Total en segundos	Total Paquetes	Total Bytes
WS	0.11s	16	3338
CORBA	0.5s	8	1111
Java-RMI	0.3s	48	7670

Tabla 2: Resultados de tecnología y trafico, con casos de estudio.

Ejemplo	Tecnología	Total Paquetes	Total Bytes
Calculadora	**WS**	48,931	10,360,814
	CORBA	10,007	1,400,851
	Java-RMI	10,098	1,017,477
Cadena	**WS**	55,617	16,053,312
	CORBA	10,007	2,236,661
	Java-RMI	10,500	2,451,790

Tabla 3: Medición de uso de Procesador y Memoria.

Ejemplo	Calculadora		Cadena	
Tecnología	**CPU Cliente**	**CPU Servidor**	**CPU Cliente**	**CPU Servidor**
WS	15.0s	6.0s	22.8s	8.4s
CORBA	2.3s	0.8s	4.0s	1.1s
Java-RMI	3.2s	1.9s	3.6s	2.1s

8. Conclusiones

En este trabajo se presento un estudio comparativo entre tres alternativas tecnológicas libres y gratuitas para el desarrollo de sistemas distribuidos. Los resultados demuestran que en entornos locales (entiéndase intranet), la alternativa más viable es el uso de JAVA-RMI, pues posee el mejor desempeño con el menor en cuanto a uso de memoria, procesador y ancho de banda. Ahora bien si las necesidades de sistemas distribuidos son orientados al uso de internet, los *web services* la solución que ofrece los mejores resultados, y su integración escalabilidad y flexibilidad son superiores a CORBA y Java-RMI. Si bien CORBA, permite integrar diversos lenguajes

Referencias

[1] W3C Organizations Web Services definitions, http:// www.w3.org/2002/ws/
[2] Java: Remote Method Invocation; http://java.sun.com/j2se/1.6.2/docs/guide/rmi/index.html
[3] CORBA: Common Object Request Broker Architecture, http://www.omg.org.
[4] Elfwing, R., Paulsson, U., and Lundberg, L, Performance of SOAP in Web Service Environment Compared to CORBA, In Proceedings of the Ninth Asia- Pacific Software Engineering Conference, IEEE, 2002.
[5] Davis, D., Parashar, M., Latency Performance of SOAP Implementations, In Proceedings of 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE, 2002.
[6] Gorton, I., Liu, A., and Brebner, P., "Rigorous evaluation of COTS middleware technology", Computer, IEEE, March 2003, pp. 50-55.
[7] JWSDP Java Web Services Developer Pack http://java.sun.com/webservices/jwsdp/index.jsp
[8] Tomcat application server 7 http://jakarta.apache.org/tomcat/ index.html
[9] Java 1.6 CORBA implementation; http://java.sun.com/j2se/1.6.0/docs/guide/corba/index.html
[10] Wireshark, protocol analyzer 1.6.1 http://www.wireshark.org/