Avaliação das tecnologias para objetos distribuídos: JAVA x .NET

Bethania Primon Candeia, Marcos José Brusso

Universidade de Passo Fundo, ICEG
Campus I, BR 285, Bairro São José
Passo Fundo (RS) – Brasil – CEP: 99001-970 – Caixa Postal: 611
Telefone: (54) 316-8354, Fax: (54) 316-8346
bethania@lci.upf.br, brusso@upf.br

Introdução

Para suprir a necessidade de criação e invocação de objetos remotos de uma maneira transparente, isto é, sem levar em conta, principalmente a localização física dos mesmos, é que adveio a idéia de objetos distribuídos (ODs). Dessa maneira, a orientação a objetos, bem como a arquitetura cliente/servidor, fazem-se presentes em um sistema de objetos distribuídos para que seja possível a invocação de métodos de um objeto, mesmo este estando em uma máquina diferente do objeto cliente [FAR, 98].

Atualmente, as tecnologias que mais se ouve falar e que permitem o desenvolvimento de aplicações distribuídas baseadas em ODs são RMI e Java IDL, oferecidas pela plataforma JAVA, e o *Remoting*, através da plataforma .NET. Sendo assim, o propósito deste artigo é mostrar um pouco do estudo feito destas tecnologias, bem como os resultados obtidos e validados através da avaliação de alguns critérios com base na performance das mesmas.

Objetos Distribuídos

A idéia básica de um sistema de objetos distribuídos (ODs) é a habilidade de criar e invocar objetos remotos, interagindo com eles como se fossem locais.

Um objeto pode ser considerado como uma instância de uma classe, isto é, de um conjunto de objetos que compartilham estrutura e comportamentos comuns. A interação entre eles se dá através da troca de mensagens pela invocação de um determinado método, que é composto basicamente pelo nome do objeto, nome do método e seus respectivos parâmetros [LIM, 01]. Já um OD, além dessas características, possui uma interface bem definida, permitindo que este possa ser acessado por outro de uma forma transparente.

JAVA RMI

A tecnologia JAVA RMI (R*emote Method Invocation)* está disponível para as plataformas J2SE, bem como J2EE e J2ME. Através dela, um objeto ativo em uma máquina virtual JAVA pode interagir com outros de outras máquinas virtuais JAVA independentemente da localização delas. Contudo, ambos os objetos devem ser escritos

na linguagem JAVA, pois esta tecnologia utiliza um protocolo de comunicação nativo, denominado JRMP (*Java Remote Method Protocol*) [WOL, 04].

O RMI possui um mecanismo chamado *RMIRegistry* (servidor de nomes) que roda no servidor com informações disponíveis sobre os objetos. Isto é necessário para que um cliente possa localizar um objeto servidor. Além disso, uma das características consideráveis do RMI é a habilidade de fazer o *download* de *bytecodes* das classes dos objetos caso esta não esteja disponível na máquina virtual do cliente. A comunicação entre o cliente e o servidor é feita através de *stubs* e *skeletons*, respectivamente. Estes são objetos intermediários, gerados pelo compilador *rmic*, responsáveis, junto com o JRMP, pela interação entre o cliente e o servidor garantindo, assim, a transparência entre os mesmos.

JAVA IDL

Java IDL é outra tecnologia Java para objetos distribuídos que adiciona características CORBA ao JAVA, fornecendo padrões baseados na interoperabilidade e conectividade. A IDL (*Interface Definition Language*) é uma característica CORBA onde cada linguagem que a suporta tem seu próprio mapeador IDL [INS 04]. Sendo assim, o Java IDL suporta esse mapeamento para o JAVA. Basta que a interface seja definida pela linguagem OMG IDL para que o compilador *idlj*, responsável pelos *stubs* e *skeletons*, mapeie para a linguagem JAVA Devido a isto, esta tecnologia permite a interação não somente entre objetos JAVA como também entre objetos escritos em linguagens que suportam CORBA.

Para que a interação entre objetos localizados em programas separados seja possível, o Java IDL disponibiliza o ORB (*Object Request Broker*) que é uma biblioteca de classes Java que permite a comunicação de baixo nível entre aplicações Java IDL e outras aplicações que suportam CORBA.

Remoting

A *Microsoft* disponibilizou a tecnologia *Remoting* através do .NET *Framework* para o desenvolvimento de aplicações distribuídas utilizando ODs. Esta comunicação é possível entre objetos de domínios de aplicação diferentes em um mesmo processo, entre objetos localizados em diferentes processos em um mesmo *host*, assim como entre objetos localizados em *hosts* diferentes [MEN, 03].

O *Remoting* traz os canais TCP e HTTP para a transmissão de mensagens, sendo que a codificação das mensagens fica a cargo dos *Formatter*s (*Binary Formatter* e *Soap Formatter* respectivamente). Além disso, suporta dois tipos de ativação de objetos: *server activation*, que são ativados no servidor e *client server*, ativados no cliente, bem como suporte ao tempo de vida do objeto remoto através do objeto *lease*.

Diferentemente do RMI e do Java IDL, as aplicações *Remoting* não ficam limitadas a uma linguagem de programação. Isto porque as aplicações .NET são compiladas para um código de linguagem intermediária (MSIL) que quando executado é convertido para o código de máquina nativo específico para a plataforma de operação. Sendo assim, pode-se utilizar mais de vinte linguagens de programação, sendo a C# escolhida para este trabalho.

Avaliação da performance

Para que o estudo das tecnologias em questão auxilie na tomada de decisão por uma delas, estas foram comparadas e avaliadas de acordo com sua performance. Os critérios adotados para fundamentar a avaliação foram: obtenção da referência ao objeto remoto, chamada de método remoto (RTT), e transferência de dados com arrays de tamanhos variáveis [JUR et. al.]. Em todos os critérios o tempo foi medido em milisegundos (msec) e para a transferência de dados foi medido o tempo de passagem e de retorno de arrays de bytes. Para que os resultados fossem validados, foi considerado 1% da média como margem de erro e um grau de confiança de 95%.

Resultados obtidos

As aplicações foram desenvolvidas utilizando o J2SDK 1.4.2 e o *Microsoft* .NET *Framework* 1.1. Os resultados são correspondentes às execuções feitas no dia 10 e 11 de novembro de 2004 nos computadores do Laboratório de Pesquisa da Universidade de Passo Fundo, sobre o sistema operacional *Windows 98*.

Sendo assim, foi constatado que o RMI possui o melhor tempo para a chamada de método remoto, ficando o *Remoting* em segundo e o Java IDL em último. As diferenças entre o RMI e o Java IDL foram grandes, como pode ser visto na figura 1. Na transferência de arrays como parâmetro o RMI também foi mais rápido, seguido do *Remoting* e do IDL. Cabe ressaltar aqui que o *Remoting* foi o que apresentou um maior aumento de tempo à medida que o tamanho do array aumentou, principalmente entre 2180 bytes e 20480 bytes. Durante o retorno de parâmetros, as diferenças entre os tempos das tecnologias entre si foram menores se comparados com os tempos gastos na passagem de parâmetros. O RMI obteve o melhor resultado (estes foram semelhantes aos obtidos na passagem de parâmetro), seguido do *Remoting* e do Java IDL, como pode ser percebido na figura 2. Em resumo, demora-se mais tempo na passagem de array do que no retorno dele, principalmente no Java IDL e no *Remoting,* respectivamente. Já na obtenção de referência ao objeto remoto, o *Remoting* obteve o melhor desempenho, seguido do Java IDL e do RMI, sendo que o desempenho do *Remoting* foi bem melhor que das outras tecnologias.

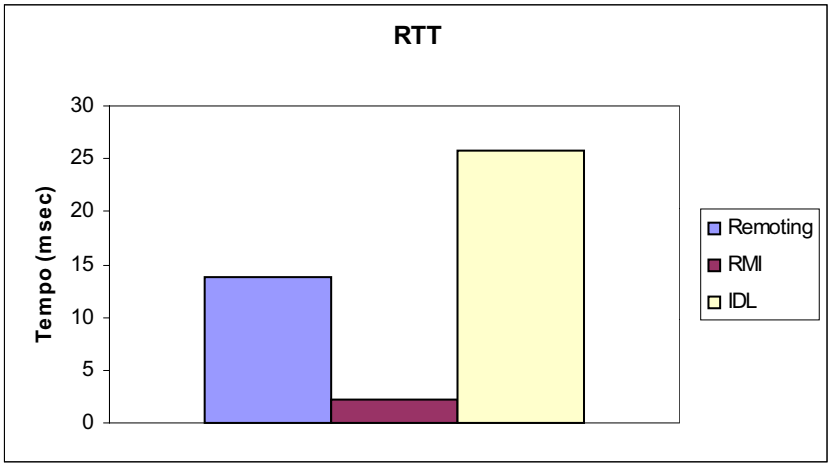

Figura 1 – RTT

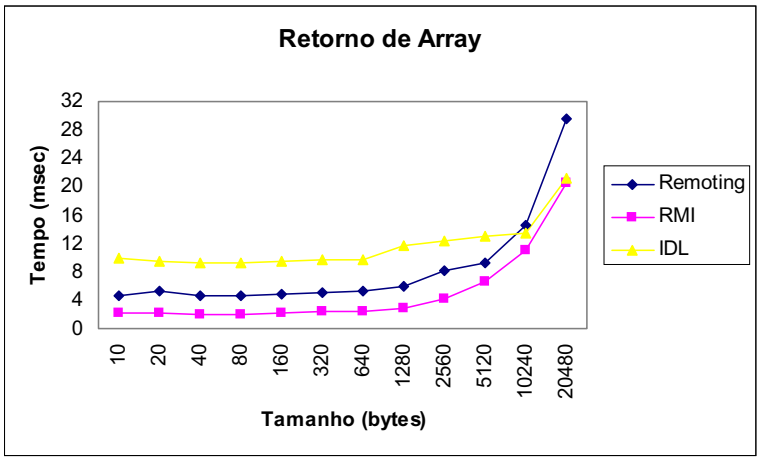

Figura 2 – Tempo de retorno de array

Conclusão

Este trabalho apresentou um estudo sobre Objetos Distribuídos nas tecnologias RMI, Java IDL e *Remoting*. Tanto o RMI quanto o Java IDL, são independentes de plataformas. O primeiro fica limitado à linguagem de programação JAVA, sendo que o Java IDL é compatível com as linguagens suportadas pelo CORBA. O *Remoting* é dependente do *Windows*, porém não se limita a uma linguagem de programação.

Através dos testes realizados, o RMI apresentou melhor desempenho em quase todos os critérios avaliados, ficando com o pior apenas na obtenção de referência. Sendo assim, esta é a tecnologia mais indicada para as aplicações que consideram tais critérios.

Referências

[FAR 98] FARLEY, Jim. **Java distributed computing**. 1. ed., cap. 3, jan, 1998. Disponível em: <http://www.oreilly.com/catalog/javadc/chapter/ch03.html>. Acesso em: 17 nov. 2003.

[LIM, 01] LIMA, Luiz Augusto de Paula Júnior. Objetos distribuídos. In: ESCOLA DE INFORMÁTICA DA SBC-SUL, 9, 2001, Passo Fundo. **Livro texto**...Porto Alegre: Instituto de Informática UFRGS, 2001. p. 145-173.

[WOL, 04] WOLLRATH, Ann; WALDO, Jim. **Trail***:* RMI. Disponível em: <http://java.sun.com/docs/books/tutorial/rmi/>. Acesso em: 18 mar. 2004.

[INS, 04] INSCORE, Jim. **Trail**: IDL. Disponível em: <http://java.sun.com/docs/books/tutorial/idl/intro/intro.html>. Acesso em: 19 março 2004.

[MEN, 03] MENDES, Fernando Vasconcelos. .NET Remoting: aplicações distribuídas no .NET. **Revista Clube Delphi**, ed 44, p. 6-11, 2003.

[JUR, 04] JURIC, Matjaz B. et al. **Performance assessment framework for distributed object architectures**. Disponível em: <http://lisa.uni-mb.si/~juric/PerfAssessmentFrw.pdf>. Acesso em: 25 maio 2004.