Comparação de Desempenho Entre Objetos Distribuídos, RMI e RMI assíncrono*

Edmar Pessoa Araujo Neto, Elton Nicoletti Mathias,
Benhur de Oliveira Stein

Universidade Federal de Santa Maria - Laboratório de Sistemas de Computação
Faixa de Camobi, Km 9 - CEP 97105-900 - Santa Maria - RS - Fone/Fax: (55) 220 8523
{araujo,emathias,benhur}@inf.ufsm.br

Introdução

O desenvolvimento de algoritmos distribuídos é mais complexo que o de algoritmos seqüenciais. Por isso, é comum que os desenvolvedores busquem maneiras para abstrair essa complexidade. Algumas linguagens orientadas a objetos como *Java* [MIC 2003] e *Objective-C* [MAR 2003] possuem ambientes de desenvolvimento que facilitam ao programador a distribuir ou paralelizar suas aplicações de forma simples.

A linguagem Objective-C possui um ambiente de desenvolvimento de aplicações chamado *GNUstep* [MAR 2003], que trata a comunicação entre vários fluxos de execução através da abordagem de *Objetos Distribuídos* [SUD 95]. Já a linguagem de programação Java possui um pacote que implementa a invocação de métodos remotos (*RMI - Remote Method Invocation* [MIC 2002]). Ambas serão explicadas em seções posteriores.

Este trabalho apresenta uma comparação de desempenho na troca de mensagens através de rede entre Objetos Distribuídos e a invocação remota de métodos RMI.

Objetos Distribuídos

Programação orientada a objetos é um paradigma onde a maior parte do tráfego de informações é feita através de troca de mensagens entre objetos. Expandindo esta idéia, poderia se ter um sistema que permita que mensagens possam ser mandadas a objetos que estejam situados em outros espaços de endereço (outros processos) ou até mesmo em máquinas diferentes. Para que isto ocorra é necessário que este sistema possa estabelecer comunicações entre objetos remotos, reconhecer quando uma mensagem é direcionada a um deles e transferir dados de um espaço de endereçamento a outro.

O ambiente de desenvolvimento GNUstep fornece ao programador uma arquitetura de Objetos Distribuídos que permite que mensagens sejam enviadas para objetos que estão em outros processos ou ter métodos executados em outras *threads* da mesma tarefa.

Para se enviar uma mensagem remota, primeiro a aplicação deve criar uma conexão com o receptor remoto. O estabelecimento da conexão fornece à aplicação um *proxy* que assume a identidade do objeto remoto e pode ser tratado como um objeto qualquer, sem nenhuma diferença de sintaxe. As mensagens direcionadas ao objeto remoto são passadas através do *proxy* para o objeto real situado em outro espaço de endereçamento, como ilustra a figura 1(a).

*Financiamento: CNPq

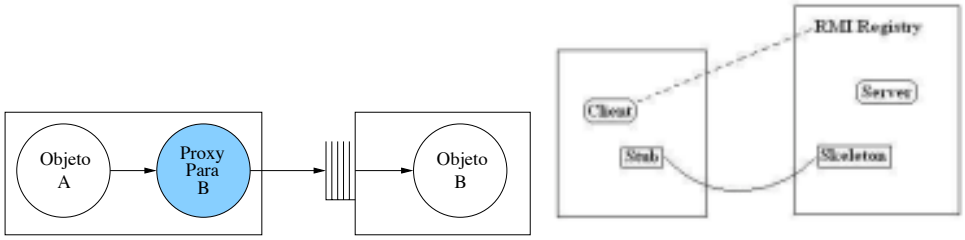

Figura 1: (a) Mensagens Remotas OD (b) Mensagens Remotas RMI

RMI (*Remote Method Invocation)*

O pacote RMI permite a objetos Java a realização de processamento distribuído e comunicação entre os mesmos através da chamada remota de métodos.

Para a utilização de RMI, objetos remotos são definidos através de uma interface remota, que é registrada em um servidor de nomes (*RmiRegistry*). A partir daí, referências a essa interface remota podem ser exportadas para clientes

Da parte do cliente, basta que ele encontre referência a uma determinada interface, para que possa, então, efetuar chamada de método remoto da mesma forma que faz com objetos locais. A figura 1(b) mostra melhor como funciona esse processo.

Pode-se notar que esta interface é bastante simples e intuitiva. Por outro lado, ela pode apresentar desempenho insatisfatório em algumas aplicações. O baixo desempenho do RMI decorre principalmente da pouca otimização nos mecanismos de serialização e transferência da linguagem Java. Outro fato limitante para a utilização de RMI em processamento de alto desempenho é que a implementação dessa interface oferece apenas invocação de métodos remotos de maneira síncrona. Dessa forma, fica impedida a execução de código concorrente.

RMI Assíncrono

Com a finalidade de explorar a possibilidade de execução concorrente, é necessário uma interface que provenha o assincronismo ao RMI. Dessa forma, torna-se possível a utilização do tempo de espera do cliente para realizar processamento concorrente nos objetos remotos distribuídos.

O RMI assíncrono utilizado nos testes comparativos [MAT 2004] foi construído completamente em Java na forma de uma interface sobre o RMI original. Essa implementação garante a chamada assíncrona de métodos através da criação de *threads* responsáveis pela chamada de método remoto. Para a recepção de resultados torna-se necessário efetuar *polling*, até que o retorno esteja disponível.

Comparação de Desempenho

Foram feitos testes de desempenho que medem a latência e a vazão do envio de mensagens a objetos remotos utilizando os Objetos Distribuídos do ambiente GNUstep e a chamada de métodos remotos do pacote Java RMI. Para isso foi implementado um

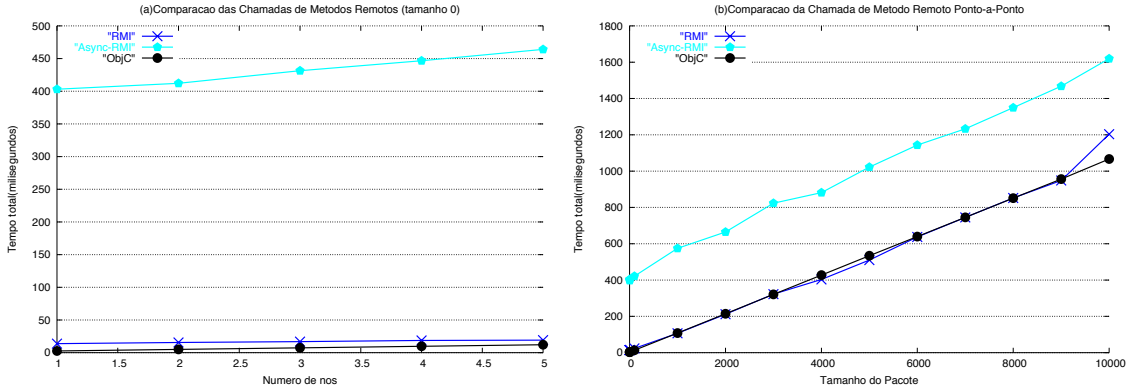

Figura 2: (a) Chamada de métodos remotos para múltiplas máquinas com parâmetro de 0KB. (b) Chamada de um método remoto entre duas máquinas variando o tamanho do parâmetro

sistema onde um processo servidor é lançado em várias máquinas e um processo cliente se comunica com os servidores através de chamada de métodos remotos. Estes métodos remotos recebem um objeto como argumento. Através deste objeto foi possível controlar o tamanho de *bytes* que seriam transmitidos nas mensagens.

Este sistema de testes foi executado em cinco computadores *Pentium III Dual* 1Ghz, com 1GB de memória e *cache* de 512KB. Todas as máquinas utilizaram adaptadores de rede *3Com Fast Ethernet*, ligadas em um comutador *3Com SuperStack 2*. Além disso todos os nós utilizavam GNU/Linux Gentoo com o núcleo do sistema operacional compilado especificamente para esta arquitetura.

Resultados

Após a execução dos testes observou-se que a latência do RMI assíncrono é muito superior a do RMI síncrono e dos Objetos Distribuídos (figura 2(a)). Isso acontece porque o RMI assíncrono cria estruturas adicionais para o controle das chamadas e recepção dos resultados. Além disso há gasto de tempo na criação e sincronização de *threads*. Na figura 2(b) observa-se que o RMI assíncrono leva mais tempo para executar uma troca de mensagens simples (ponto-a-ponto) pois não tem a oportunidade de explorar o paralelismo. Percebe-se também que os Objetos Distribuídos e o RMI síncrono possuem um comportamento semelhante, exceto quando o tamanho dos dados transmitidos excede os 9000KB onde o RMI síncrono passa a ter uma sobrecarga maior na serialização do objeto transmitido.

Na figura 3(a) percebe-se que o *overhead* do RMI assíncrono ainda é maior que do RMI síncrono e dos Objetos Distribuídos, este gráfico também ilustra que à medida que o número de máquinas aumenta, a sobrecarga dos Objetos Distribuídos é maior que a do RMI, isso ocorre porque para cada objeto remoto é criado um proxy local, diferentemente do RMI que usa apenas referência para o controle dos destinos das mensagens. Na figura 3(b) podemos ver que a sobrecarga para cada nova comunicação do RMI assíncrono é menor, pois a medida que o número de máquinas aumenta observa-se que ele tem

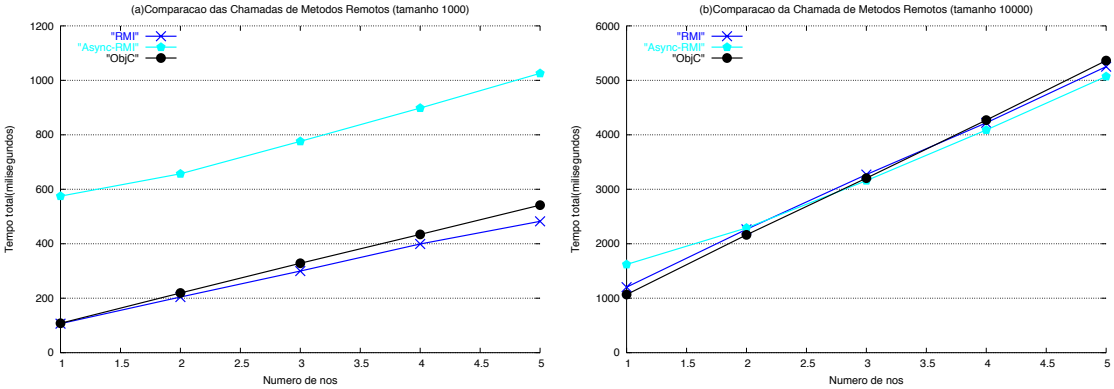

Figura 3: Chamada de métodos remotos para múltiplas máquinas com parâmetros de 1000KB e 10000KB

uma perda de desempenho menor que a dos outros métodos.

Conclusão

As linguagens orientadas a objetos proporcionam uma maior facilidade para a construção de aplicações paralelas ou distribuídas, pois este paradigma de programação baseia-se na troca de mensagens entre objetos. Assim, algumas linguagens que seguem este modelo, como Java e Objective-C, proporcionam ambientes de desenvolvimento que auxiliam ao programador na concepção de aplicações que utilizam objetos remotos para divisão de tarefas entre vários fluxos de execuções.

Este trabalho apresentou uma comparação de desempenho entre a arquitetura de Objetos Distribuídos do GNUstep e o pacote de invocação de métodos remotos RMI síncrono e uma interface que permite a chamada de métodos assíncronos com linguagem Java. Como trabalhos futuros pretende-se utilizar uma aplicação real para comparação de desempenho, bem como a utilização de outros sistemas de chamada de métodos remotos como o *CORBA* e o *ProActive*.

Referências

[MAR 2003] MARCOTTE, L. Programming under GNUstep — an introduction. **Linux Journal**, v.108, Apr. 2003.

[MAT 2004] MATHIAS, E.; PASIN, M. Alto desempenho utilizando RMI assincrono. **ERAD 2004. Escola Regional de Alto Desempenho**, Pelotas, RS, Brasil, 2004.

[MIC 2002] MICROSYSTEMS, S. **Java remote method invocation specification(revision 1.8)**. [S.l.]: Sun, 2002.

[MIC 2003] MICROSYSTEMS, S. Java programming language. **http://java.sun.com/aboutJava/**, 2003.

[SUD 95] SUDAMA, R. Get Ready for Distributed Objects. **Datamation**, v.41, n.18, p.67–76, 1995.