Análise do Sobrecusto de ProActive em Relação a Sockets Java na Transferência de Objetos pela Rede*

Diogo Pinheiro de Araújo, Lucas Graebin, Rodrigo da Rosa Righi

[1]Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos (UNISINOS)
Caixa Postal 93.022-000 – São Leopoldo – RS – Brasil

diogo_pinheiro13@hotmail.com, lgraebin@gmail.com, rrrighi@unisinos.br

Resumo. *Os clusters de computadores estão cada vez mais presentes nas computações que necessitam de grande quantidade de recursos. Técnicas de balanceamento de carga, como migração de processos, tornam-se pertinentes para que se obtenham os resultados em um tempo aceitável enquanto é mantida a precisão desejada. Tendo em vista esse contexto, este artigo apresenta uma análise inicial do custo de migração de objetos Java criados com a biblioteca ProActive [Caromel et al. 1998]. Os resultados mostram a sobrecarga dela ao comparar os seus tempos com aqueles obtidos com Sockets Java. Além disso, eles destacam situações onde as migrações são interessantes ao relacionar a região de dados alocada com o tempo final de migração.*

1. Introdução

O balanceamento de carga é pertinente para melhorar o uso dos recursos computacionais e assim, diminuir o tempo da aplicação. Especificamente no ramo de tratamento de processos, a migração deles é um exemplo típico do emprego dessa técnica. A migração busca reorganizar um mapeamento processos-recursos aplicado previamente, com o intuito de manter ou mesmo melhorar um determinado nível de desempenho da aplicação. A implementação da migração de processos/objetos pode ser uma tarefa complicada, uma vez que exige conhecimentos técnicos como detalhes da arquitetura do executável, uso de checkpoints e reorganização dos canais de comunicação. Dado esse contexto, a construção de ferramentas eficientes e de fácil utilização para prover essa funcionalidade é fundamental para o uso confiável da migração de processos. Nessa linha, o ProActive é uma biblioteca Java para sistemas largamente distribuídos que vem se destacando pela sua interface simplificada e pela sua abstração em tarefas como a migração de objetos. Por mais que exista um sobrecusto ao efetuar a migração de um objeto utilizando o ProActive em relação a Sockets, a biblioteca se torna útil para efetuar migrações de médio/longo período de duração, pois esse sobrecusto tende a diminuir com o aumento do tamanho do objeto.

Este artigo avalia o custo de migração de objetos Java utilizando a biblioteca ProActive em relação a Sockets. Ele está organizado em 5 seções, as quais são descritas da seguinte forma. A seção 2 descreve brevemente a biblioteca ProActive e suas principais características. A metodologia de avaliação é apresentada na seção 3. A seção 4 destaca os resultados obtidos. A seção 5 reforça as conclusões obtidas com esse trabalho.

*Essa pesquisa é parcialmente financiada pelo CNPq

2. Biblioteca ProActive

O ProActive [Caromel et al. 1998] é uma biblioteca Java para o desenvolvimento de aplicações paralelas e distribuídas. Ela oferece uma interface mais simples para a construção de aplicações se comparada com Java RMI. O principal conceito inerente ao desenvolvimento do ProActive está relacionado ao tratamento de objetos ativos. Um objeto ativo é similar a um objeto remoto. Ele pode ser acessível remotamente e é composto de um objeto padrão Java e de um fluxo de execução, chamado de corpo. Um objeto ativo pode ser instanciado por dois métodos disponibilizados pelo ProActive. Os métodos newActive() e turnActive() da classe PAActiveObject têm por finalidade, respectivamente, instanciar uma classe convencional do Java como um objeto ativo e tornar um objeto convencional já instanciado em um ativo. O ProActive também oferece o recurso de migração de objetos para outras máquinas do sistema distribuído. Ele permite transferi-los para um novo nó ou para um nó no qual outro objeto resida no momento.

3. Metodologia e Ambiente de Execução

Foi desenvolvida uma aplicação hipotética com o intuito de mensurar o sobrecusto da migração de objetos criados com o ProActive. A ideia com a sua execução é observar o custo de migração variando a memória que é alocada nos objetos. Para tal, são reservadas três máquinas para a execução da aplicação. No experimento, a aplicação que executa na máquina $m1$ cria um objeto ativo que executa na máquina $m2$ e aloca uma quantidade de bytes em seu construtor. Em seguida, a aplicação invoca a diretiva de migração do ProActive que irá transferir o objeto para uma máquina de destino $m3$. O tempo do sistema é computado antes e após a chamada da diretiva de migração, resultando assim no tempo total de transferência do objeto.

Uma segunda aplicação Java utilizando Sockets foi desenvolvida para o experimento. Seu objetivo é usar as diretivas de envio/recebimento de Sockets para transferir objetos convencionais Java de diferentes tamanhos entre máquinas. Para cada tamanho de objeto testado, a transferência foi realizada duas vezes e a média aritmética dos tempos resultou no custo de transferência. Foi utilizada a versão 4.3.1 do ProActive. Os testes foram executados em dois nós bi-processados Intel Xeon 2.4 GHz com 2 GB de memória principal do laboratório de pesquisa do PIPCA da Unisinos, ligados por uma rede de 10 Mb/s. Ambas aplicações transferem um objeto entre máquinas. Tal objeto possui simplesmente um array do tipo primitivo byte do Java, o qual é alocado com tamanhos que variam de 1 KB até 32 MB.

4. Resultados Obtidos

Os experimentos foram realizados em um ambiente dedicado não havendo interferências de rede tampouco de processamento de outras aplicações. A tabela 1 apresenta os tempos obtidos com a execução das aplicações desenvolvidas. Ao migrar um objeto Java de aproximadamente 1 KB, alcança-se o tempo de 1,42 segundos com o ProActive e 0,13 segundos com a transferência do mesmo via Sockets. A figura 1 ilustra o gráfico com os resultados obtidos. Esse gráfico apresenta um comportamento praticamente linear, onde um aumento na quantidade de dados alocados acarreta em um aumento proporcional no tempo de migração e transferência.

Tabela 1. Detalhamento dos tempos de migração/transferência de objetos Java

Tamanho do objeto (KB)	Tempo (s)	
	ProActive	Socket
1	1,42	0,13
1024	2,19	1,01
2048	3,12	1,89
4096	4,90	3,67
8192	8,50	7,29
16384	15,79	14,45
32768	30,32	28,90

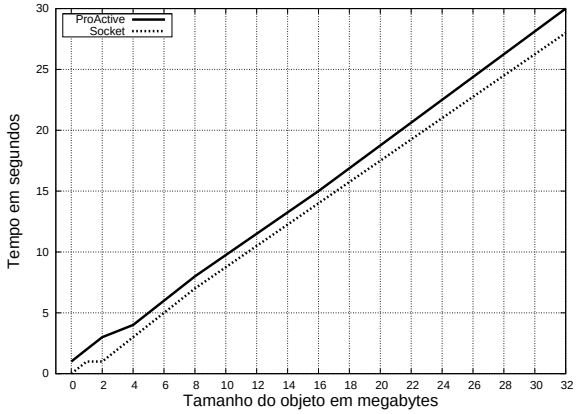

Figura 1. Comportamento da migração/transferência de objetos Java

Com os resultados apresentados na tabela 1 foi analisado o sobrecusto de migração de objetos do ProActive. A ideia nesse ponto é observar a percentagem paga pelo ProActive para oferecer a migração de objetos de maneira abstrata para o programador. A tabela 2 apresenta o resultado dessa análise. Nela, observa-se que 91,20% do tempo de migração de um objeto de 1 KB utilizando o ProActive é considerado sobrecarga em relação a transferência do mesmo via Sockets. Também é possível observar que a tendência é que esse sobrecusto seja menor com o aumento do tamanho do objeto. Tal sobrecusto justifica-se pelas operações que a biblioteca ProActive realiza para garantir o acesso transparente ao objeto migrado. Dentre essas operações, pode-se citar a criação de um objeto procurador e a serialização de todos os objetos ligados ao objeto ativo. Utilizando Sockets o objeto é simplesmente transferido entre máquinas.

O gráfico da figura 2 destaca que a taxa de transferência máxima é próxima de 8 Mb/s. Essa taxa foi encontrada uma vez que o laboratório está interligado por um hub cuja vazão nominal é 10 Mb/s. Por fim, os resultados enfatizam a viabilidade do uso de ProActive para migração em aplicações com moderado/longo período de duração. Nesse sentido, o uso da migração em uma aplicação com objetos de 4 MB em memória e cuja execução demore minutos levaria 4.9s para serializar e transferir um objeto entre diferentes nós.

Tabela 2. Sobrecusto de migração de objetos com o ProActive

Tamanho do objeto em kilobytes	Sobrecusto
1	91,20%
1024	53,92%
2048	39,49%
4096	25,12%
8192	14,16%
16384	8,44%
32768	4,71%

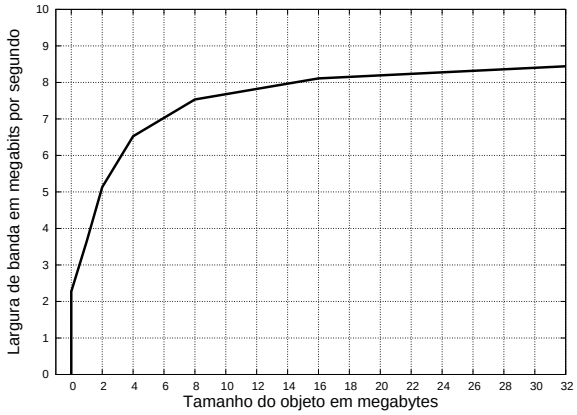

Figura 2. Largura de banda de migração de objetos com o ProActive

5. Conclusão

Este artigo apresentou testes de migração de objetos Java utilizando o ProActive. Essa biblioteca oferece uma interface com alto nível de abstração quanto a funcionalidade de migração. Os testes foram limitados por uma rede de 10 Mb/s existente no laboratório. Assim, espera-se que o tempo de migração seja menor em ambientes com maior largura de banda. O próximo passo para a pesquisa envolve o uso de ProActive no desenvolvimento de um *middleware* para o reescalonamento de aplicações BSP conforme o modelo MigBSP [Righi et al. 2009]. Além disso, sugere-se avaliar o custo de serialização de objetos Java, mensurando a quantidade exata de informações que são agregadas neles e o tempo necessário para serializar e desserializar os mesmos.

Referências

Buyya, R. (1999). *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Caromel, D., Klauser, W., and Vayssière, J. (1998). Towards seamless computing and metacomputing in Java. *Concu. Practice and Experience*, 10(11–13):1043–1061.

Righi, R. d. R., Pilla, L. L., Carissimi, A., Navaux, P., and Heiss, H.-U. (2009). Migbsp: A novel migration model for bulk-synchronous parallel processes rescheduling. In *HPCC '09: IEEE Conference on High Performance Computing and Communications*, pages 585–590, IEEE.