jMigBSP: Migração de Objetos e Comunicação One-Sided em Java para Aplicações BSP

Lucas Graebin[1], **Rodrigo da Rosa Righi**[1]

[1]Programa Interdisciplinar de Pós-Graduação em Computação Aplicada (PIPCA)
Universidade do Vale do Rio dos Sinos (UNISINOS)
Av. Unisinos, 950 – 93.022-000 – São Leopoldo – RS – Brazil

lgraebin@gmail.com, rrrighi@unisinos.br

1. Justificativa de Pesquisa e Sistema Proposto

A quantidade de problemas que demandam alto poder de processamento é crescente. Para dar suporte à solução deles, são construídas arquiteturas de máquinas específicas para alto desempenho e aplicações que consigam usufruir ao máximo o paralelismo e a distribuição por elas oferecidos. Nesse sentido, pesquisam-se modelos de programação paralela que sejam independentes da arquitetura de computadores subjacentes. Como resultado de tais esforços, têm-se a crescente aceitação do modelo *Bulk Synchronous Parallel* (BSP) [Valiant 1990]. Ele é caracterizado por sua portabilidade, podendo ser implementado em diferentes arquiteturas de computadores. O modelo BSP divide as aplicações em superetapas, cada qual contendo fases de computação e comunicação, seguidas de uma barreira de sincronização. Como exemplo de sistemas que o implementam, pode-se citar o BRAMS o qual calcula a previsão atmosférica para o Brasil. Nesse contexto, está em desenvolvimento a interface de programação **jMigBSP**, que possibilita a escrita de aplicações do tipo BSP em Java. A escolha dessa linguagem é aparada pela facilidade no desenvolvimento de aplicações para agregados e pelo seu caráter multiplataforma. O desenvolvimento de jMigBSP é suportado pela biblioteca ProActive [Caromel et al. 1998]. O ProActive apresenta uma interface simples para a escrita de aplicações distribuídas, oferecendo um alto grau de abstração em tarefas como a migração de objetos.

O jMigBSP tira proveito de três ideias do ProActive: (i) a facilidade de mapeamento de objetos para recursos; (ii) o modelo de programação *Object-Oriented SPMD* (OOSPMD) e (iii) a migração de objetos. A primeira ideia permite que os desenvolvedores indiquem os recursos computacionais disponíveis na rede e aqueles necessários para a execução das aplicações. O mapeamento deve ser descrito em arquivos no formato XML e elimina a necessidade de o desenvolvedor especificar nomes de máquinas e protocolos de comunicação no código fonte. Assim, é possível executar qualquer aplicação em diferentes ambientes sem alterações no código das mesmas. No que tange o modelo de programação OOSPMD, o ProActive implementa a técnica SPMD (*Single Program Multiple Data*) em um contexto orientado a objetos. A biblioteca em questão utiliza comunicação em grupo e permite o lançamento e a execução simultânea de objetos ativos em diferentes máquinas ou processadores. Nesse cenário, cada objeto membro do grupo possui um identificador, o que possibilita a comunicação entre eles. Por fim, o recurso de migração habilita a transferência de objetos para outras máquinas do sistema distribuído, permitindo transferi-los para um novo nó. Para tal, é deixado um procurador na máquina origem que é usado para realizar chamadas sobre o objeto de forma transparente.

Considerando o contexto oferecido pela linguagem Java e pelo ProActive, o

jMigBSP busca oferecer uma interface simples e robusta para a construção de aplicações BSP. Nessa linha, para escrever um programa utilizando a interface em pauta, o usuário precisa apenas estender a classe `jMigBSP`. Entre os métodos implementados nessa classe, destacam-se o `bsp_put()` e o `bsp_sync()`. O método `bsp_put()` permite a cópia de dados para o *buffer* de um objeto remoto, habilitando assim a abstração de comunicação *one-sided*. Já o método `bsp_sync()` implementa uma barreira de sincronização e identifica o término de uma superetapa e o início da próxima. Esse método garante que os dados recebidos durante uma superetapa estarão disponíveis para uso somente no início da próxima. O jMigBSP é desenvolvido inteiramente utilizando as classes padrões da linguagem Java, não impondo modificações na máquina virtual ou a necessidade de compiladores especiais ou pré-processadores. Em adição, ele difere de outras interfaces de programação BSP [Gu et al. 2001] por implementar métodos que permitem ao desenvolvedor realizar a migração de processos de forma explícita.

2. Avaliação Preliminar e Considerações Finais

A avaliação consistiu em implementar a versão paralela do algoritmo da soma de prefixos. Nesse algoritmo, dado um conjunto de n valores $a_0, a_1, ..., a_{n-1}$ e uma operação associativa $\oplus$, a computação da soma de prefixos é dada por $[a_0, (a_0 \oplus a_1), (a_0 \oplus a_1 \oplus a_2), ..., (a_0 \oplus a_1 \oplus ... \oplus a_{n-1})]$. A implementação paralela do algoritmo opera em ciclos, onde cada ciclo efetua a soma de elementos de acordo com o padrão de comunicação entre os objetos. Na implementação em questão assume-se que, tendo n processadores, cada um deles possuirá um valor do conjunto. O resultado final da soma de prefixos aparece no processador $n - 1$. O programa foi executado no agregado de computadores do PIPCA da UNISINOS, que é composto por 10 nós. Os testes variaram com conjuntos de 4, 8 e 10 valores, tendo, respectivamente, 2, 3 e 4 superetapas. Todas as avaliações resultaram na execução correta da aplicação. O próximo passo para a pesquisa envolve implementar um protótipo de reescalonador de processos no jMigBSP. Para tanto, será adotado o modelo MigBSP [da Rosa Righi et al. 2010], que trabalha com múltiplas métricas e com adaptações com o intuito de propor novos escalonamentos que reduzam o tempo das superetapas. Portanto, aplicações jMigBSP poderão usufruir de balanceamento de carga através da migração implícita de seus objetos. O código do reescalonador será executado no método de barreira de um objeto. Nele são feitos os cálculos de escalonamento, a captura de dados dos demais objetos e a relocação quando pertinente. Para tal, utilizar-se-á a chamada de migração escrita em jMigBSP para mover o objeto chamador para outro nó. Além disso, trabalhos futuros incluem a análise de desempenho de jMigBSP.

Referências

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

da Rosa Righi, R., Pilla, L. L., Maillard, N., Carissimi, A., and Navaux, P. O. A. (2010). Observing the impact of multiple metrics and runtime adaptations on bsp process rescheduling. *Parallel Processing Letters*, 20(2):123–144.

Gu, Y., Lee, B.-S., and Cai, W. (2001). Jbsp: A bsp programming library in java. *Journal of Parallel and Distributed Computing*, 61(8):1126 – 1142.

Valiant, L. G. (1990). A bridging model for parallel computation. *Commun. ACM*, 33:103–111.