Migração de Objetos: Análise de jMigBSP com uma Aplicação CPU-Bound para Compressão de Imagens

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@acm.org, rrrighi@unisinos.br

1. Introdução

A tarefa de alocar processos para recursos em *grids* e *clusters* torna-se muitas vezes um problema que requer um esforço considerável para o programador. A fim de explorar plenamente tais ambientes, deve-se conhecer tanto a arquitetura da máquina paralela quanto o código da aplicação. Além disso, uma nova aplicação poderá exigir uma nova análise para o escalonamento dos processos. Considerando isso, uma possibilidade é explorar o remapeamento deles através de algoritmos que atuam nas bibliotecas de programação. Nesse contexto, estamos desenvolvendo uma biblioteca chamada jMigBSP [Graebin and Righi 2011]. Ela permite a escrita de aplicações Java no estilo BSP (*Bulk Synchronous Parallel*) e se diferencia de trabalhos relacionados por oferecer facilidades de reescalonamento tanto em nível de *middleware* quanto de aplicação. Em adição, jMigBSP possui rotinas para comunicação assíncrona e *one-sided* que permitem ao programador ler e escrever diretamente na memória de um objeto remoto. Esse artigo apresenta os resultados obtidos com a avaliação do recurso de migração explícita de jMigBSP com uma aplicação CPU-Bound. O algoritmo que realiza a compressão de imagens segundo o método de Fractal foi escolhido para a avaliação [Uma et al. 2011]. Tal técnica tem gerado interesse na comunidade em virtude da alta taxa de compressão e da boa qualidade da imagem resultante. Um dos principais problemas na abordagem de Fractal é o alto custo computacional associado a fase de codificação. Embora a compressão seja custosa, o processo de decodificação é rápido e simples [Uma et al. 2011].

2. Metodologia de Avaliação e Resultados

O algoritmo de compressão inicia por dividir a imagem alvo em blocos de referência e blocos de domínio. Para cada bloco de referência, é realizada uma procura na imagem aplicando um conjunto de transformações afins com o objetivo de encontrar blocos de domínio similares. A modelagem paralela do algoritmo considera a variação na quantidade de domínios e no número de processos. A estratégia combina o modelo BSP com a ideia de *pipeline* circular. A Figura 1 (a) apresenta a organização de uma superetapa BSP. Primeiramente, blocos de domínio são distribuídos uniformemente entre os processos. Em seguida, cada processo é designado a calcular um bloco de referência. As melhores combinações de domínio para as referências calculadas são armazenadas no processo. Após a etapa de computação, esse processo envia ao vizinho da direita as combinações, juntamente com o bloco de referência calculado anteriormente e que serão a carga de trabalho do receptor na próxima superetapa. Quando todos os blocos de referência tiverem passado por todos os processos do *pipeline*, a imagem resultante será gerada.

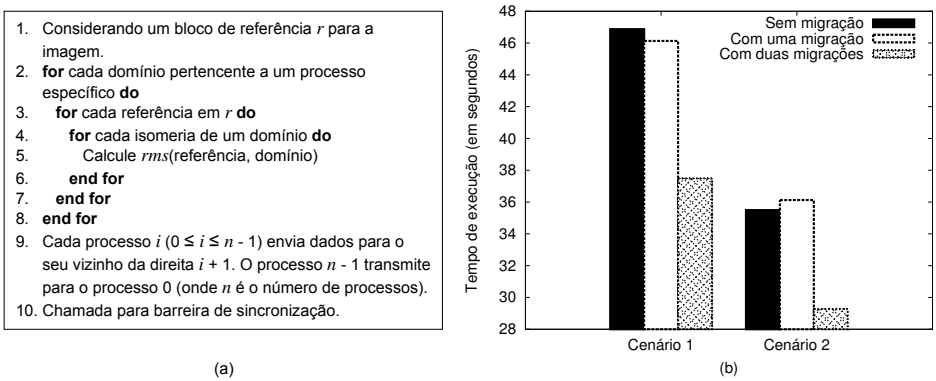

Figura 1. (a) Modelagem de uma superetapa BSP para o problema da compressão de Fractal; (b) Ganhos obtidos ao trabalhar com 4 objetos e migrar 2 deles de nós sobrecarregados para nós levemente carregados.

A avaliação compreendeu o desenvolvimento do algoritmo de compressão utilizando jMigBSP. Os testes foram executados em um *cluster* composto por 18 nós Intel Core 2 Duo de 2,93GHz ligados por uma rede de 100 Mbps. Dois cenários foram montados para uma imagem de 256×256 *pixels*. No primeiro, trabalhou-se com 4096 blocos de domínio e uma taxa de compressão de 68,52%. Já o Cenário 2 possuía 256 blocos de domínio e uma taxa de compressão de 97,76%. Além da execução paralela do algoritmo, a avaliação compreendeu observar os ganhos com a migração de objetos em ambientes sobrecarregados. Para tanto, quatro nós foram reservados e uma sobrecarga foi simulada em dois deles. A simulação de sobrecarga consistiu em limitar o uso máximo da CPU em 33% usando a ferramenta cpulimit[1]. A Figura 1 (b) apresenta os resultados obtidos com essa avaliação. Observou-se ganhos de 20,01% e 17,53% nos Cenários 1 e 2, respectivamente, ao migrar os objetos localizados em nós sobrecarregados para outros levemente carregados. A migração de um objeto não apresenta resultados satisfatórios devido ao sincronismo do modelo BSP. Apesar da transferência de um objeto, temos ainda um outro que permanece em um processador sobrecarregado, limitando o tempo da superetapa.

3. Conclusão e Trabalhos Futuros

A avaliação de jMigBSP compreendeu a execução do algoritmo de Fractal para a compressão de imagens em um ambiente com nós sobrecarregados. Os resultados envolvendo migração explícita mostram a pertinência dessa técnica quando oscilações relacionadas ao estado dos recursos são frequentes. Nesse sentido, trabalhos futuros compreendem o desenvolvimento de uma nova biblioteca chamada LBjMigBSP. Seu foco é oferecer reescalonamento automático em nível de *middleware* para aplicações escritas em jMigBSP.

Referências

Graebin, L. and Righi, R. d. R. (2011). jMigBSP: Object Migration and Asynchronous One-Sided Communication for BSP Applications. In *The 12th Int. Conf. on Parallel and Distr. Comput., Applications and Technologies (PDCAT 2011)*, Gwangju, Korea.

Uma, K., Palanisamy, P., and Poornachandran, P. (2011). Comparison of Image Compression Using GA, ACO and PSO Techniques. In *International Conference on Recent Trends in Information Technology (ICRTIT 2011)*, Chennai, India.

[1]http://cpulimit.sourceforge.net/