Analisando a Migração de Processos MPI para seu Emprego em Aplicações BSP

Laércio Lima Pilla[1], **Rodrigo da Rosa Righi**[1], **Philippe Olivier Alexandre Navaux**[1]

[1]Grupo de Processamento Paralelo e Distribuído
Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{laercio.pilla, rodrigo.righi, navaux}@inf.ufrgs.br

Resumo. *O desbalanceamento de carga (processos) de uma aplicação entre recursos é um problema relevante em sistemas paralelos e distribuídos. Ele pode resultar na queda de desempenho da aplicação, pois esta acaba por não tirar um melhor proveito dos recursos disponíveis. Tal problema é ainda mais marcante em aplicações desenvolvidas segundo o modelo de programação BSP, onde o tempo de cada superetapa é proporcional ao tempo do processo mais lento. Este artigo apresenta uma análise da migração de processos que poderá ser usada como forma de rebalanceamento de carga em aplicações BSP.*

1. Introdução

O uso de programas e ambientes distribuídos dinâmicos tem se tornado cada vez mais freqüente para a obtenção de alto desempenho. O tratamento eficiente da dinamicidade é um desafio nessa área, pois informações estabelecidas de antemão podem não ser comprovadas em tempo de execução de uma aplicação [Bhandarkar et al. 2000]. Isto porque é possível que alguns processadores se tornem sobrecarregados e redes fiquem congestionadas nesse meio tempo. Quanto a própria aplicação, os processos que a compõe podem requisitar mais processamento em um dado momento ou mesmo alterar o comportamento de comunicação entre si.

A dinamicidade em sistemas paralelos pode levar ao problema de desbalanceamento da carga de processos entre os recursos disponíveis, principalmente quando se consideram aplicações em fases, como aquelas desenvolvidas segundo o modelo BSP (*Bulk Synchronous Parallelism*) [Bonorden 2007]. Este modelo pode ser usado em aplicações do tipo MPI e trabalha com a noção de fases síncronas, nas quais o tempo de conclusão de cada uma depende da execução do processo mais lento. Uma alternativa para melhorar o desempenho neste sentido, é proporcionar o reescalonamento de processos através da migração deles para novos recursos. Essa técnica pode ser disparada em função do tratamento da dinamicidade com o intuito de obter maior desempenho. No caso de aplicações BSP, pode-se utilizar esta técnica para oferecer rebalanceamento de carga, aumentando o aproveitamento dos recursos e minimizando o tempo das suas fases.

Neste contexto, este artigo aborda testes iniciais de migração em uma aplicação sintética MPI. A idéia é avaliar preliminarmente o custo de migração de um processo MPI que varia a quantidade de dados que aloca. Os testes guiarão a viabilidade do desenvolvimento de um rebalanceador de carga para aplicações BSP confeccionadas conforme a norma MPI. Por fim, aplicações do tipo BSP são o alvo final do nosso trabalho e esse modelo foi escolhido porque é bastante utilizado na modelagem de aplicações científicas como aquelas que tratam de dinâmica de fluídos e eletromagnetismo [Schepke 2007].

2. Modelo de Programação BSP

Programas que seguem o modelo BSP possuem uma estrutura horizontal e outra vertical. A horizontal advém da concorrência que existe na execução de um número fixo de processos. Tais processos podem ser mapeados para processadores de forma arbitrária e existe comunicação todos-para-todos entre eles. A estrutura vertical provem do progresso da computação através do tempo e é uma composição seqüencial de superetapas. As fases de uma superetapa estão ilustradas na Figura 1. A separação das fases de computação, comunicação e sincronização permite computar o tempo máximo de cada superetapa e por conseqüência predizer o desempenho da aplicação BSP como um todo. Uma aplicação BSP pode ser escrita usando bibliotecas de comunicação bem conhecidas como PVM e MPI, assim como aquelas específicas para esse modelo como a BSPlib.

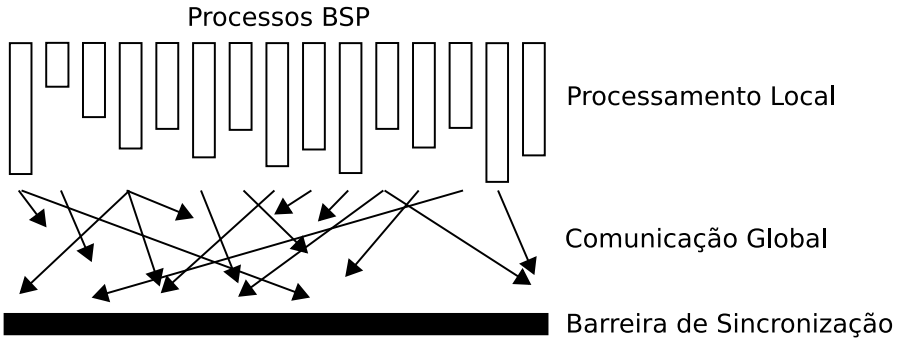

Figura 1. Visão de uma superetapa no modelo BSP

Embora BSP tenha sido originalmente proposto para ser usado em um ambiente de computação paralela confiável, homogêneo e dedicado, ele pode também ser usado em ambientes dinâmicos e não previsíveis de antemão. Camargo, Kon e Goldman [Camargo et al. 2005] apresentam três estratégias para minimizar o tempo de execução de um programa BSP: (i) Rebalancear a computação local; (ii) Rebalancear a comunicação; e (iii) Diminuir as superetapas. O item (i) trata do balanceamento da computação dos processos entre os recursos, uma vez que a barreira de sincronização de cada superetapa tem a necessidade de esperar pelo processo mais lento. Esse balanceamento faz-se pertinente principalmente num ambiente heterogêneo, com processadores de diferentes velocidades e que podem ter flutuações na sua carga. O item (ii) refere-se ao balanceamento da comunicação entre os processos. O equilíbrio da comunicação é relevante quando existem comunicações por ligações congestionadas ou que usam a Internet. Dessa forma é importante aproximar processos com elevado padrão de comunicação para uma mesma rede local (LAN) ou de sistema (SAN). Por fim, o item (iii) dedica-se a diminuição do número de fases percorridas pela aplicação. Analisando tais itens, mostra-se possível viabilizar ambas idéias de rebalanceamento através da migração automática de processos BSP. É possível realizar a migração após a barreira de sincronização de uma superetapa e antes de começar a próxima. Nesse ponto tem-se informações tanto da parte de computação, quanto da comunicação entre os processos BSP.

3. Resultados Experimentais

A aplicação MPI utilizada nos testes é sintética e desenvolvida em linguagem C com a biblioteca AMPI (Adaptive MPI) [Huang et al. 2007]. A idéia com a sua execução é observar o custo de migração de um processo variando a memória que é alocada nele. Para

a execução da aplicação são reservadas três máquinas e criados dois processos. O primeiro destes processos aloca uma quantidade de bytes e chama a diretiva de migração AMPI_Migrateto(destino). Além disso, ele envia duas mensagens ao segundo processo, uma antes e a outra logo depois da migração. O segundo processo é criado na máquina não envolvida na migração do primeiro processo. Ele captura o tempo entre as mensagens enviadas pelo primeiro processo através da diretiva MPI_Wtime(). Deste tempo é descontado o atraso entre o envio e o recebido da segunda mensagem, resultando no tempo de migração do primeiro processo.

Na biblioteca AMPI existem duas formas de se manter válidas as referências aos dados alocados dinamicamente após uma migração. A primeira delas redefine a chamada malloc() para utilizar a diretiva chamada isomalloc(). Ela faz com que os dados sejam automaticamente migrados. Esta abordagem é mais simples para o programador, pois o tratamento dos dados é realizado de forma totalmente transparente. Ela garante que os dados do processo terão os mesmos endereços virtuais independente de processador. Para isso, é alocado um intervalo de mesma área de endereço virtual entre os processadores. Esta área é dependente do número de processadores, podendo esta abordagem tornar-se inviável quando a quantidade deles for grande. A segunda abordagem requer a especificação pelo programador de uma função que trata do empacotamento e desempacotamento de dados. Para isso, AMPI oferece uma função de registro que recebe como parâmetros a função e os dados alocados. Esta função será chamada automaticamente antes e depois de uma chamada para migração. A aplicação desenvolvida utilizou a segunda abordagem de tratamento dos dados.

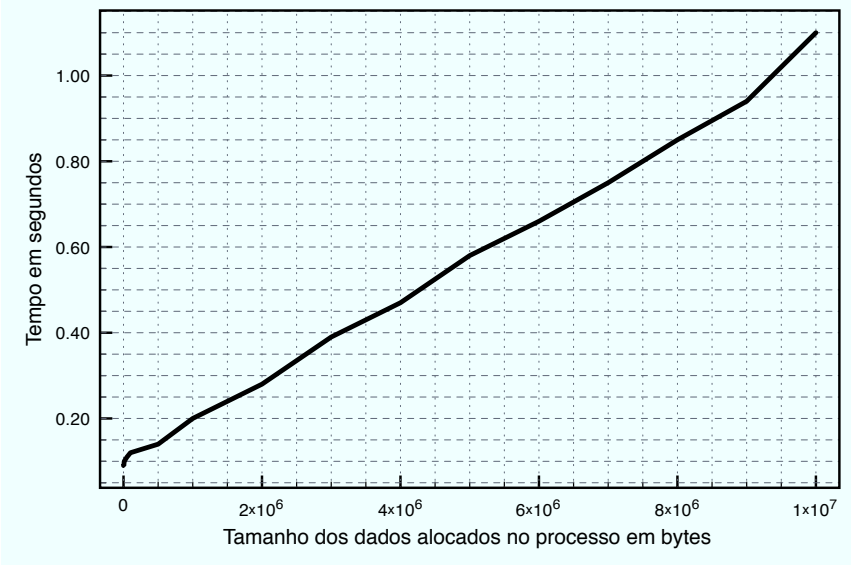

Figura 2. Tempo de migração de um processo variando os dados que aloca

Os testes foram executados em três nós Pentium III 1.2 GHz biprocessados do Cluster Corisco do Instituto de Informática da UFRGS, ligados por uma rede de 100 Mbit/s. O tamanho dos dados alocados pela aplicação variam de 50 Bytes até 10 MBytes. Para cada tamanho testado, a aplicação foi executada 50 vezes e a média aritmética dos resultados é apresentada na Figura 2. O gráfico apresenta-se praticamente linear, onde um aumento nos dados acarreta em um aumento proporcional no tempo. Ao migrar um

processo que aloca 10 MBytes, alcança-se um tempo de 1.1 segundo, o que representa aproximadamente 73% da capacidade da rede. Conclui-se que esse resultado deve-se ao fato que o teste é executado em um ambiente dedicado, onde não há interferências com congestionamentos ou com a execução de outras aplicações nos nós reservados. Assim, espera-se que o tempo de migração seja maior em ambientes dinâmicos e não dedicados.

4. Conclusões

Este artigo apresentou testes iniciais de migração de um processo MPI. Ele também abordou um estudo teórico sobre o modelo BSP e migração em seu contexto. Os resultados experimentais mostram um caminho viável do uso da biblioteca AMPI para a migração. Isto a torna uma opção para rebalanceamento de carga usando migração em programas BSP. Os experimentos mostraram uma relação memória$\times$tempo em um ambiente dedicado. Visto que aplicações paralelas podem ter seus tempos de execução na ordem de minutos e até horas, o tempo de aproximadamente 1,1 segundo para se migrar 10 MBytes pode ser considerado pequeno. Os resultados num ambiente dinâmico e não dedicado à aplicação tendem a serem inferiores aos apresentados, sem uma proporção certa.

O próximo passo da pesquisa envolve a utilização de AMPI e seu suporte a migração na execução da aplicação MPI que computa o método de Lattice Boltzman. Esta aplicação foi desenvolvida segundo o modelo BSP no Grupo de Processamento Paralelo e Distribuído [Schepke 2007]. Baseado no funcionamento do módulo AMPI que trata de gerentes de balanceamento de carga, nós planejamos construir um gerente particular. Esses gerentes controlam e efetuam o reescalonamento de processos e são chamados através de diretivas AMPI_Migrate() (sem parâmetros) na própria aplicação. Assim, pode-se colocar essa diretiva após cada barreira de uma superetapa BSP, segundo o modelo de rebalanceamento de carga em aplicações BSP descrito em [Righi et al. 2007].

Referências

Bhandarkar, M. A., Brunner, R., and Kal, L. V. (2000). Run-time support for adaptive load balancing. In *IPDPS '00: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, pages 1152–1159, London, UK. Springer-Verlag.

Bonorden, O. (2007). Load balancing in the bulk-synchronous-parallel setting using process migrations. In *21th Parallel and Distributed Processing Symposium*, pages 1–9.

Camargo, R. Y. D., Kon, F., and Goldman, A. (2005). Portable checkpointing and communication for bsp applications on dynamic heterogeneous grid environments. In *17th International Symposium on Computer Architecture and High Performance Computing, 2005*, pages 226–234.

Huang, C., Zheng, G., and Kalé, L. V. (2007). Supporting adaptivity in mpi for dynamic parallel applications. Technical Report 07-08, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign.

Righi, R., Carissimi, A., and Navaux, P. O. A. (2007). On the dynamic load-rebalancing of bsp application using process migration. In *V Workshop em Processamento Paralelo e Distribuído (WSPPD 2007)*, pages 31–36.

Schepke, Claudio; Maillard, N. (2007). Performance improvement of the parallel lattice boltzmann method through blocked data distributions. In *19th International Symposium on Computer Architecture and High Performance Computing, 2007*, pages 71–78.