Reescalonamento de Processos em Aplicações BSP Irregulares Empregando MigBSP*

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. *Este artigo apresenta uma avaliação do reescalonamento de processos sobre uma aplicação irregular do tipo BSP (Bulk Synchronous Parallel). Tal aplicação é baseada em programação dinâmica e sua irregularidade se apresenta através da variação da carga computacional entre as células da matriz. O reescalonamento é controlado pelo modelo MigBSP. MigBSP combina múltiplas métricas para decidir sobre a migração de processos e utiliza-se de adaptações para reduzir seu impacto na execução da aplicação. A migração automática de processos representou uma técnica para obter desempenho sem esforços para o programador, com ganhos de até 10% sobre uma arquitetura multi-cluster.*

1. Introdução

Dinamismo e heterogeneidade são palavras-chave nos cenários atuais de computação paralela e distribuída. O dinamismo pode ser visto tanto em nível de máquina (flutuações na carga e disponibilidade de processadores e rede) quanto em nível de aplicação (mudanças nos padrões de comunicação e quantidade de computação). Como exemplo de aplicações dinâmicas e irregulares, pode-se citar algoritmos baseados em programação dinâmica. Esta técnica é popular na resolução de problemas onde a solução ótima é formada por subestruturas ótimas com superposição. Numa aplicação desse tipo a carga de trabalho em uma matriz pode aumentar conforme a direção da computação. Nesta situação, alternativas para o ganho de desempenho incluem o mapeamento dos processos mais carregados para os processadores mais rápidos, a definição do particionamento dos dados durante a execução e a adição de estratégias de balanceamento de carga dentro da própria aplicação. Entretanto, estas técnicas requerem esforço no reconhecimento de características de novas arquiteturas alvo e aplicações. Além disso, as propriedades estabelecidas inicialmente podem não se provar verdadeiras durante a execução da aplicação. Assim, uma alternativa pertinente para este problema é a migração automática e dinâmica de processos.

Neste contexto, tal questão é endereçada através do controle dinâmico da migração de processos sobre ambientes de grid. Foi desenvolvido um modelo chamado **MigBSP** [Righi et al. 2008] que controla o reescalonamento de processos em aplicações BSP (*Bulk Synchronous Parallel*) [Valiant 1990]. MigBSP atua através do reescalonamento de processos entre os recursos sem necessitar de alterações nem conhecimento prévio sobre a aplicação. Este artigo apresenta uma avaliação do impacto de MigBSP sobre a execução do algoritmo de Smith-Waterman de alinhamento de sequências. O interesse aqui é observar a influência da densidade computacional na escolha dos processos à migrar.

*Pesquisa parcialmente apoiada pelo CNPq e Microsoft

2. MigBSP: Modelo de Reescalonamento de Processos BSP

O reescalonamento de processos BSP ocorre de forma dinâmica de acordo com o estado do ambiente. A idéia do modelo MigBSP é o ganho de desempenho através da redução do tempo de execução das superetapas. A duração de cada superetapa é igual ao tempo do processo mais lento. Assim, o modelo busca migrar os processos com maiores cargas de computação para processadores mais rápidos, além de aproximar processos que se comunicam muito. Uma descrição detalhada de MigBSP pode ser vista em [Righi et al. 2008].

A escolha dos processos candidatos à migração é baseada no Potencial de Migração (PM) de cada processo para um dado Conjunto. Um Conjunto é a abstração de uma rede e pode ser entendido como sendo um cluster, uma rede local ou um supercomputador. Cada um possui um gerente responsável pelo escalonamento. O PM é encontrado através da combinação das métricas Computação, Comunicação e Memória. Computação e Comunicação agem a favor da migração, enquanto a Memória trabalha no sentido contrário, trazendo a idéia de custo. A métrica Computação considera informações sobre o grau de desempenho do Conjunto considerado, a predição do tempo de computação do processo e o Padrão de Computação do mesmo. O Padrão de Computação mede a regularidade do processo quanto as instruções que realiza em cada superetapa. O seu valor é próximo a 1 se o processo é regular e próximo a 0, caso contrário. Padrões foram empregados pois espera-se que o processo desempenhe um comportamento similar no lugar de destino. Assim como a métrica Computação, a Comunicação computa o Padrão de Comunicação. A cada superetapa, esse padrão considera a quantidade de dados recebidos provenientes de processos de determinado Conjunto. Além disso, essa métrica usa a predição do tempo de comunicação do processo. A métrica Memória trabalha com a memória do processo, os custos de migração e a taxa de transferência entre o processo e o Conjunto alvo. Com o PM calculado, é possível definir quais processos são candidatos a migração. Antes de migrá-los, verifica-se a viabilidade da migração. Isto é feito através da simulação da execução do processo em um novo recurso no Conjunto alvo. Caso a execução no novo recurso adicionado ao tempo gasto na migração seja maior que o tempo de execução no recurso atual, a migração do processo é cancelada.

A decisão de remapeamento de processos é tomada após o final de uma superetapa, pois neste momento é possível analisar dados de todos os processos BSP sobre suas fases de computação e comunicação. Contrário a biblioteca PUBWCL [Bonorden 2007], MigBSP não realiza o reescalonamento a cada superetapas mas sim de acordo com o estado do ambiente. Para isso, foram aplicadas adaptações que gerenciam o valor de α, onde α representa o número de superetapas entre duas chamadas de reescalonamento. As idéias gerais das adaptações são: (i) atrasar as chamadas se o sistema estiver balanceado ou torná-las mais frequentes, caso contrário; (ii) atrasar as chamadas se o sistema apresentar um padrão sem migrações nas últimas ω tentativas. A segunda adaptação é importante para cenários onde os processos estão desbalanceados e migrações são inviáveis.

3. Aplicação Irregular

Programação dinâmica é um método de resolução para problemas que exibem as propriedades de subestruturas ótimas com superposição. Aqui tratamos do algoritmo de Smith-Waterman [Smith and Waterman 1981], utilizado para encontrar o alinhamento ótimo entre duas sequências. Ele determina a similaridade entre duas sequências de nucleotídeos ou proteínas. O algoritmo age sobre uma matriz $n \times m$, onde n e m são os

tamanhos das sequências a serem comparadas. Cada célula i,j da matriz, onde $1 \leq i \leq n$ e $1 \leq j \leq m$, depende da computação das células $i-1,j$ e $i,j-1$. Portanto, a computação paralela somente ocorre entre células na mesma anti-diagonal. Além disso, a densidade da computação aumenta de acordo com a elevação de n,m, tornando o algoritmo irregular.

As colunas da matriz são mapeadas para processos. A execução do algoritmo acontece de forma que cada anti-diagonal da matriz representa uma superetapa. Cada processo computa os dados referentes a uma célula de sua coluna e os envia ao seu próximo vizinho. Não há comunicação dentro de uma mesma coluna, pois as células desta estão alocadas ao mesmo processo. Caso a matriz for quadrada de ordem n, cada processo se envolve com n superetapas, em um total de $2n - 1$. É comum em aplicações BSP o fato de nem todos processos estarem envolvidos em todas as superetapas [Valiant 1990].

4. Resultados Experimentais

O principal objetivo desta avaliação experimental é observar o impacto do modelo MigBSP em uma aplicação irregular. Para isso, foram simulados três cenários diferentes na ferramenta SimGrid: (i) Apenas execução da aplicação; (ii) Execução da aplicação com MigBSP mas sem migrações; e (iii) Execução da aplicação com MigBSP com migrações. Aqui serão apresentados os resultados da simulação com matrizes de 50×50 e 200×200, as quais executam por 99 e 399 superetapas, respectivamente. O ambiente simulado está ilustrado na Figura 1. Os testes foram executados com α inicial igual a 2, 4, 8 e 16. Em todos os casos, o sistema se encontra balanceado e o valor de α sempre aumenta. Além disso, os processos que apresentam PM$>MAX$(PM)$\times 0.80$ são candidatos à migração. Os resultados da simulação são ilustrados na Figura 2, normalizados pelos cenários i.

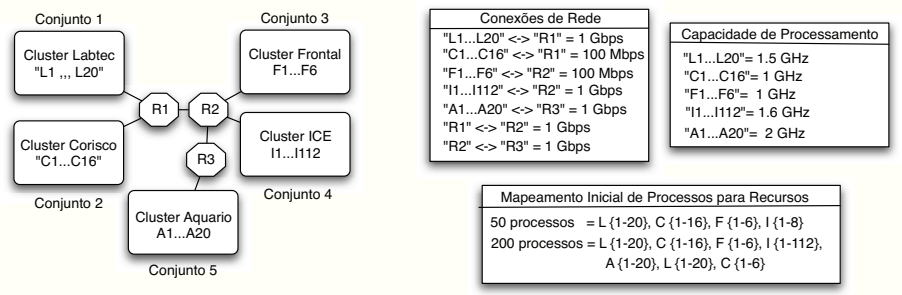

Figura 1. Ambiente simulado e mapeamento inicial de processos para recursos

A execução de 50 processos ocorre quando a matriz 50×50 é testada. Os processos ocupam 800 KBytes em memória e as comunicações envolvem 50 KBytes. Um sobrecusto de até 3% é percebido na comparação dos cenário i e ii, como ilustrado na Figura 2(a). Além disso, sempre se obteve um ganho de desempenho nos cenários iii. Os processos sempre acabam migrando para o cluster Aquario. Quando utilizado $\alpha = 2$, cinco chamadas de reescalonamento são feitas. Nenhuma migração ocorre nas três primeiras chamadas. Com $\alpha = 4$ se observa um ganho maior do que com $\alpha = 2$. Isso acontece devido a última chamada de reescalonamento ocorrer na superetapa 60, tendo assim mais duas superetapas para usufruir de processadores mais rápidos. O melhor desempenho na simulação acontece com $\alpha = 8$. Neste caso, apenas 3 chamadas de reescalonamento acontecem. Além disso, na última chamada, todos os processos do cluster Frontal são migrados. Por fim, ao considerar $\alpha = 16$, um dos processos que executam no cluster Frontal acaba não sendo migrado, resultando em um desempenho pior do que $\alpha = 4$ e 8.

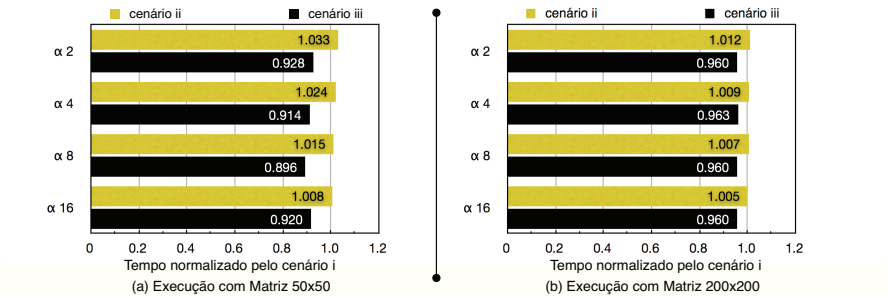

Figura 2. Tempos de execução em diferentes cenários: (a) matriz 50×50; (b) matriz 200×200

O gráfico na Figura 2(b) apresenta o desempenho na simulação de 200 processos. Cada processo ocupa 725 KBytes em memória e as comunicações envolvem 25 KBytes. O gráfico mostra um sobrecusto menor do que 2% em todos os casos. Além disso, resultados satisfatórios foram obtidos com a migração de processos. Não há problemas em ter mais de um processo mapeado para um mesmo processador, pois nem todos executam em cada superetapa. Considerando α inicial igual a 2, 8 chamadas para o reescalonamento de processos aconteceram. Todas migrações acontecem para o cluster Aquario. Nas três primeiras vezes nenhum processo é migrado. Na superetapa 62, todos processos do cluster Frontal são migrados. Além disso, nenhuma migração é viável na superetapa 254. Por fim, 12 processos são migrados na última chamada de reescalonamento. Neste momento, todos os outros processos alocados ao cluster estão inativos e a migração é viável. Porém, a aplicação executa apenas mais 10 superetapas para amortizar os custos da migração.

5. Conclusão

Este artigo apresentou brevemente o modelo MigBSP, assim como a avaliação de seu sobrecusto e desempenho na execução de uma aplicação irregular. Inicialmente, MigBSP foi desenvolvido para tratar processos com comportamento regular. Porém, ele permite, através de parâmetros flexíveis, o tratamento da irregularidade de uma maneira eficiente. Nos casos simulados, o modelo impõe uma baixa sobrecarga na execução da aplicação quando migrações não acontecem (custo inferior a 3%). Além disso, obteve-se ganhos de até 10% com migrações no ambiente simulado.

Referências

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

Righi, R., Pilla, L., Carissimi, A., and Navaux, P. O. A. (2008). Controlling processes reassignment in bsp applications. In *20th International Symposium on Computer Architecture and high Performance Computing (SBAC-PAD 2008)*, pages 37–44. IEEE.

Smith, J. M., Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195-197.

Valiant, L. G. (1990). A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111.