Aplicando Migração de Processos em Aplicações BSP: Estudo de caso usando Decomposição LU

Rodrigo Righi[1], **Alexandre Carissimi**[1], **Philippe Navaux**[1], **Hans-Ulrich Heiss**[2]

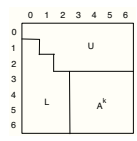
[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)

[2]Kommunikations- und Betriebssysteme - Technische Universität Berlin (TU Berlin)

{rodrigo.righi,asc,navaux}@inf.ufrgs.br, heiss@cs.tu-berlin.de

1. Introdução

Migração de processos é um pertinente mecanismo para oferecer balanceamento de caga, principalmente em ambientes dinâmicos e heterogêneos. Nesse contexto, foi desenvolvido o modelo MigBSP, o qual controla o reescalonamento de processos em aplicações do tipo BSP (*Bulk Synchronous Parallel*) [da Rosa Righi 2009]. MigBSP é especialmente pertinente para obter desempenho nessas aplicações, uma vez que elas são compostas por fases (ou superpassos) síncronas que sempre esperam pelo processo mais lento. Detalhes sobre o desenvolvimento de MigBSP podem ser vistos em [da Rosa Righi 2009] e suas contribuições podem ser resumidas em dois caminhos: (i) combinação de múltiplas métricas para para a seleção de processos candidatos a migração e; (ii) adaptações que agem na frequência do reescalonamento com o intuito de diminuir a sobrecarga de MigBSP sobre a aplicação. No primeiro deles, MigBSP usa uma função de decisão chamada Potencial de Migração (PM) que é atingida da seguinte maneira: $PM = COMP + COM - MEM$. Nessa equação, $COMP$ e COM são as métricas computação e comunicação enquanto MEM representa os custos de migração.



(a)

for k=0 to n-1 do
  for j=k to n-1 do
    $u_{k,j} = a^k_{k,j}$;
  for i=$k+1$ to n-1 do
    $l_{i,k} = \frac{a^k_{i,k}}{u_{k,k}}$;
  for i=$k+1$ to n-1 do
    for j=$k+1$ to n-1 do
      $a^{k+1}_{i,j} = a^k_{i,j} - l_{i,k}.u_{k,j}$;

(b)

for k=0 to n-1 do
  for i=k+1 to n-1 do
    $a_{i,k} = \frac{a_{i,k}}{a_{k,k}}$;
  for i=$k+1$ to n-1 do
    for j=$k+1$ to n-1 do
      $a_{i,j} = a_{i,j} - a_{i,k}.a_{k,j}$

(c)

Figura 1. (a) Matriz no estágio 3; (b) e (c) Diferentes visões do algoritmo LU

Em especial, esse artigo aborda a modelagem da aplicação de decomposição LU[Bonorden 2007] segundo o modelo BSP e sua execução usando MigBSP. A decomposição LU divide uma matriz A no produto de uma matriz triangular superior L e outra inferior U tal que $A = L.U$. LU é empregado para tornar o cálculo de equações lineares mais fácil, visto que a solução de um conjunto triangular de equações é trivial. A Figura 1 (a) mostra a idéia do algoritmo usado. Os valores de L, U e A^k podem ser armazenados no mesmo espaço de memória que A^0 (A^0 representa a matriz original e k o índice da evolução do algoritmo). A Figura 1 (b) apresenta o algoritmo que produz L e U em estágios. A última iteração prepara A^{k+1} para o próximo estágio. Já a Figura 1 (c) mostra o funcionamento do algoritmo anterior usando os mesmos elementos da matriz A.

2. Avaliação e Considerações Finais

A técnica de simulação em três cenários foi usada para a avaliação. Além disso, foi empregado o simulador Simgrid nos seguintes cenários de testes: (i) execução da aplicação LU simplesmente; (ii) aplicação com MigBSP sem habilitar migrações e; (iii) aplicação com MigBSP permitindo migrações. A compração entre os cenários i e ii permite ver a sobrecarga imposta pelos algoritmos de MigBSP. Já aquela que trata os cenários i e iii possibilita verificar se migrações acarretam ganho ou perda de desempenho. Ao avaliar matrizes de ordem 500, 1000 e 2000, notou-se que quanto maior a ordem delas, maior o ganho com a migração de processos. Ao aumentar a matriz, maior a carga computacional para os processos e maior o ganho de desempenho com a relocação deles para processadores mais velozes. Além disso, observou-se que ao aumentar o número de processos, a viabilidade da migração de processos vai decrescendo. Isso deve-se ao fato que ao aumentar o número de processos, as carga sde computação e comunicação dos processos não são capazes de sobrepor os custos de migração segundo o cálculo de PM.

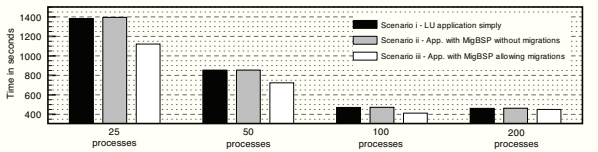


Figura 2. Desempenho para a matriz 5000x5000 nos três cenários testados

Tendo em vista os resultados obtidos até então, foi expandida a ordem da matriz para 5000 para verificar se um maior ganho com migrações era verificado. Os resultados com essa configuração são ilustrados no gráfico da Figura 2. O simples movimento de processos do $cluster$ mais lento para o mais rápido atinge um ganho de 19% ao executar 25 processos. Os testes com 50 processos obteram 852.31s e 723.64s para os cenários i e iii, respectivamente. A hipótese original foi confirmada uma vez que obteve-se 15% (ordem 5000) e 10% (ordem 2000) de ganho na aplicação LU com 25 processos. Como detectado anteriormente, o gráfico apresentado também mostra que um número maior de processos leva a um ganho menor com a relocação dos processos. Em adição, contrário aos tamanhos de matrizes anteriormente testados, a ordem 5000 habilita migração ao usar 200 processos. Isso acontece porque nessa configuração trabalha-se com um grão de computação maior. Por fim, conclui-se que MigBSP pode ser utilizado para almejar desempenho na aplicação LU de maneira sem esforços. Essa expressão é verificada pelo fato que MigBSP atua em nível de $middleware$ (sem mudanças no código da aplicação) e não necessita de execuções complementares para alimenar o modelo de reescalonamento.

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.

da Rosa Righi, R. (2009). *MigBSP: A New Approach for Processes Rescheduling Management on Bulk Synchronous Parallel Applications*. Tese de doutorado em ciência da computação, Universidade Federal do Rio Grande do Sul, Brasil.