Comparação entre Métodos de Decomposição de Domínio e Decomposição de Dados na Solução de Sistemas de Equações

Guilherme Galante, Tiarajú Asmuz Diverio
PPGC, Instituto de Informática, UFRGS
CP 15064, 91501-970, Porto Alegre, RS, Brasil
{ggalante,diverio}@inf.ufrgs.br

André Luis Martinotto, Delcino Picinin Júnior, Ricardo Vargas Dorneles
Departamento de Informática, Centro de Ciências Exatas e Tecnologia, UCS
Rua Francisco Getúlio Vargas, 1130, 95001-970, Caxias do Sul, RS, Brasil
{almartin,dpicinij,rvdornel}@ucs.br

Rogério Luis Rizzi
Centro de Ciências Exatas e Tecnológicas, UNIOESTE, Campus de Cascavel
Rua Universitária, 2069, 85801-110, Cascavel, PR, Brasil
rogerio@unioeste.br

Resumo

Neste trabalho é apresentado um estudo comparativo entre as abordagens de decomposição de dados e decomposição de domínio para a resolução em paralelo de sistemas de equações. As duas abordagens mostraram-se eficientes, com bons ganhos de desempenho na resolução de sistemas de equações. Nos testes efetuados a abordagem de decomposição de dados mostrou-se mais adequada para uma quantidade menor de processos, enquanto que a abordagem de decomposição de domínio mostrou-se mais escalável, comportando-se melhor com quantidades mais elevadas de processos.

1. Introdução

A simulação numérica é uma importante abordagem para o desenvolvimento científico e tecnológico, uma vez que esta pode ser utilizada em situações onde experiências reais são impossíveis ou economicamente inviáveis. Em geral, problemas que empregam simulação são baseados em modelos matemáticos que são expressos através de Equações Diferenciais Parciais (EDPs).

As EDPs são definidas em domínios contínuos, e em geral, não possuem solução analítica conhecida, sendo necessário o uso de métodos de aproximação e métodos de discretização, como diferenças finitas ou elementos finitos. No processo de discretização o domínio é dividido em um número finito de pontos, denominado malha. Nesses pontos os termos das EDPs são aproximados, resultando em sistemas de equações que devem ser resolvidos a cada passo de tempo.

Numericamente, a acurácia da solução depende, além do método de aproximação das EDPs utilizado, do número de pontos da malha. Quanto mais refinada for a malha, mais acurada será a solução. Em contrapartida, maior será a ordem do sistema de equações a ser resolvido [9]. Sendo assim, em simulações numéricas realísticas de grande escala espaço-temporal, e de alta acurácia numérica, o tempo computacional necessário para simular tais fenômenos pode ser muito elevado. Para remediar essa situação utiliza-se, cada vez mais freqüentemente, computação de alto desempenho e, em particular, a computação em *clusters*.

Existem, pelo menos, duas grandes abordagens para a resolução de sistemas de equações em paralelo. Em uma delas, chamada de decomposição de dados, gera-se um único sistema de equações para todo o domínio, e este é resolvido através de um método numérico paralelizado. A segunda abordagem consiste na utilização de métodos de decomposição de domínio. Esses são baseados no particionamento do domínio computacional em subdomínios,

de tal modo que a solução global do problema é obtida pela combinação apropriada das soluções obtidas em cada um dos subdomínios. Uma vez que diferentes subdomínios podem ser tratados independentemente em paralelo, tais métodos são atrativos para ambientes computacionais paralelos.

Neste trabalho é apresentado um estudo comparativo entre as abordagens de decomposição de dados e decomposição de domínio para a resolução em paralelo dos sistemas de equações gerados pelo modelo de hidrodinâmica do HIDRA, que é um modelo paralelo com balanceamento dinâmico de carga para a simulação da hidrodinâmica e transporte de substâncias em corpos de água desenvolvido no GPPD (Grupo de Processamento Paralelo e Distribuído) da UFRGS (Universidade Federal do Rio Grande do Sul) [16] [8].

As implementações apresentadas neste trabalho foram desenvolvidas de forma a explorar o paralelismo em *clusters* multiprocessados. Em *clusters* com essas características, além do paralelismo inter-nodos pode-se explorar o paralelismo intra-nodos. Para exploração do paralelismo inter-nodos e intra-nodos foram usadas as bibliotecas MPI (*Message Passing Interface*) e OpenMP, respectivamente.

2. *Clusters* Multiprocessados

Um *cluster* é uma arquitetura baseada na união de um conjunto de máquinas independentes, interconectados por uma rede de interconexão dedicada e rápida, formando uma plataforma de alto desempenho para execução de aplicações paralelas. A utilização de *clusters* em aplicações que requerem uma alta capacidade de processamento só é uma realidade devido ao desenvolvimento e barateamento das tecnologias de redes locais e a evolução na capacidade de processamento dos computadores pessoais [2].

Os nodos de um *cluster* podem ser monoprocessados ou multiprocessados. *Clusters* com nodos multiprocessados podem ser considerados um sistema de memória hibrída, por possuírem, ao mesmo, tempo memória compartilhada e memória distribuída. Em *clusters* multiprocessados existe a possibilidade da exploração do paralelismo intra-nodal em conjunto com a exploração do paralelismo inter-nodal.

Em *clusters* multiprocessados, o paralelismo intra-nodal e inter-nodal pode ser explorado das seguintes formas [2]:

- Somente com o uso de ferramentas de troca de mensagens: troca de mensagens são um padrão no desenvolvimento de aplicações para *clusters*. Quando usado em *clusters* multiprocessados, os processadores de um mesmo nodo comunicam-se entre si através do uso de troca de mensagens, não usufruindo das vantagens presentes em ambientes de memória compartilhada.
- Somente com o uso de memória compartilhada: mecanismos DSM (*Distributed Shared Memory*), implementados em *software* ou *hardware*, são uma alternativa ao uso de troca de mensagens em *clusters*. Mecanismos DSM permitem simular ambientes de memória compartilhada em ambientes com memória distribuída. Desse modo, o programador não precisa se preocupar com a comunicação inter-nodos. Esse tipo de mecanismos introduz um custo adicional para o gerenciamento de endereços e coerência de dados.
- Utilização de troca de mensagens em conjunto com memória compartilhada: para uma melhor exploração do paralelismo em *clusters* multiprocessados é adequado explorar as vantagens de ambas as arquiteturas. Nesse caso é conveniente, para a exploração do paralelismo inter-nodos (memória distribuída) o uso de troca de mensagens e para a exploração do paralelismo intra-nodos o uso de *multithreading*.

Neste trabalho analisou-se o uso de troca de mensagens em conjunto com memória compartilhada (*multithreading*) na paralelização dos dois métodos de solução. Para exploração do paralelismo inter-nodos utilizou-se a biblioteca de troca de mensagens MPI e para a exploração do paralelismo intra-nodos utilizou-se a biblioteca de *threads* OpenMP. Maiores informações sobre o OpenMP podem ser encontradas na página do projeto OpenMP [1]. E para maiores informações sobre o MPI recomenda-se Pacheco [14] e Snir et al [20].

3. Solução Paralela

O método utilizado na discretização do sistema de EDPs foi o de diferenças finitas que, combinado com métodos de aproximação apropriados para as variáveis envolvidas, gera sistemas de equações lineares, simétricos definidos-positivos (SDP), esparsos e de grande porte. Nesse caso, pode-se resolver esses sistemas de equações através do método do gradiente conjugado, que é um dos mais efetivos métodos iterativos no subespaço de Krylov para obter a solução de sistemas com essas características [18].

Como os sistemas de equações gerados pelo modelo HIDRA são de grande porte é conveniente o uso de processamento paralelo. Para a solução em paralelo enfocou-se duas abordagens: a paralelização de métodos numéricos e o emprego de métodos de decomposição de domínio. Na primeira abordagem gera-se um único sistema de equações para todo o domínio que é resolvido através de um método numérico paralelizado. Na segunda abordagem emprega-se um método de decomposição de domínio de modo que a solução global do problema é obtida pela combinação apropriada das soluções obtidas em cada um dos subdomínios. Estas abordagens já foram estudadas em outros trabalhos desenvolvidos no GPPD:

- "Paralelização de Métodos de Solução de Sistemas Lineares Esparsos com o DECK em Clusters de PCs", dissertação de Ana Paula Canal, onde foram implementadas versões paralelas do Método do Gradiente Conjugado (GC) e do Método de Thomas para matrizes do tipo banda [3];
- "Paralelização de Métodos de Solução de Sistemas Lineares em Clusters de PCs com as Bibliotecas DECK, MPICH e Pthreads", dissertação na qual Delcino Picinin Jr. implementou e analisou a paralelização do método do GC e do Método do Resíduo Mínimo Generalizado (GMRES), utilizando MPI, DECK e Pthreads [15];
- A dissertação de André Luis Martinotto e o trabalho de conclusão de curso de Guilherme Galante abordaram a solução paralela de sistemas de equações lineares através de métodos de decomposição de domínio [13] [10].

3.1. Particionamento do Domínio

Para que o problema possa ser resolvido em paralelo, é necessário fazer o particionamento do domínio computacional entre os processadores disponíveis. O particiomento do domínio deve ser feito de forma que a carga de trabalho de cada processador seja proporcional a sua capacidade de processamento. Além disso, em problemas que requerem a troca de informações nas fronteiras dos subdomínios, como no modelo de hidrodinâmica utilizado para realizar os experimentos apresentados neste trabalho, o particionamento deve ser feito de modo a reduzir as fronteiras dos subdomínios e, consequentemente, a comunicação entre os processadores.

De modo geral, o particionamento do domínio, deve ser feito de forma a atender três requisitos básicos [8]:

- A distribuição balanceada da carga computacional entre os processadores disponíveis;
- A minimização da comunicação entre os processadores;
- O tempo de execução do algoritmo de particionamento deve ser menor que o tempo ganho com o seu uso.

O particionamento de domínio pode ser modelado como um problema de particionamento de grafo, onde os vértices do grafo representam as células do domínio e as arestas representam os dados trocados entre os subdomínios. O total de comunicação entre os subdomínios é dado pelo total de arestas entre os subdomínios.

Considerando que um grafo representa o domínio do problema deve-se dividí-lo em k subgrafos, onde k é o número de processadores disponíveis, de modo que cada processador tenha um número proporcional de vértices (células) e que o número de arestas (comunicação) entre as partes seja o menor possível. O problema de dividir um grafo em k subgrafos com as propriedades de maximizar a computação e minimizar as comunicações entre eles é um problema NP-Difícil, existindo apenas heurísticas que encontram uma aproximação da solução ótima [11].

Existem diferentes software que implementam algortimos de particionamento de grafos. Entre essas destacam-se as bibliotecas METIS, Chaco, JOSTLE. Para o desenvolvimento deste trabalho foi utilizado, no particionamento do domínio, o pacote METIS 4.0, que é um pacote de execução seqüencial para o particionamento de grafos, malhas e reordenamento de matrizes esparsas desenvolvido na *University of Minnesota* por George Karypis e Vipin Kumar [12].

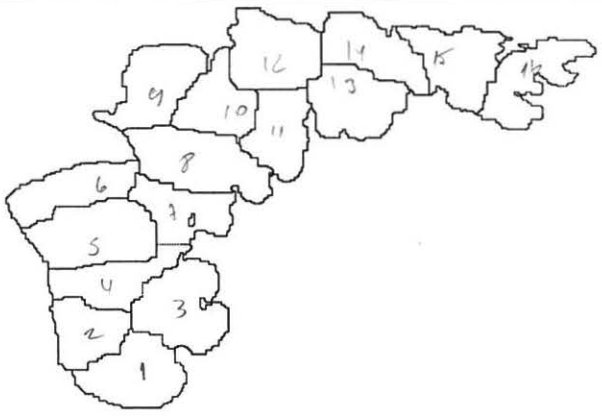

Figura 1. Particionamento do Lago Guaíba em 16 subdomínios

A Figura 1 mostra um exemplo para o particionamento para o domínio do Lago Guaíba, em 16 processos, usando o METIS. No particionamento tomou-se como espaçamento horizontal $\triangle x = \triangle y = 50m$. Com essa escolha tem-se 185.124 vértices (células) e 366.032 arestas.

Após o particionamento do domínio, a geração dos sistemas de equações é realizada em paralelo pelos processos, e a solução desses pode ser obtida através da abordagem de decomposição de dados ou de domínio.

3.2. Decomposição de Dados

Na abordagem de decomposição de dados distribui-se as operações e os dados entre os processadores de forma que possam ser resolvidos independentemente, exceto nos pontos de sincronismo. Nessa abordagem, gera-se um único sistema de equações para todo o domínio computacional que é resolvido de forma distribuída, empregando um método

de solução paralelizado [6]. Neste trabalho paralelizou-se o método do gradiente conjugado (GC).

O GC é um método iterativo não estacionário, pertencente à classe de métodos do subespaço de Krylov, e parte do princípio de que o gradiente, que é um campo vetorial, aponta sempre na direção mais crescente da função quadrática. O algoritmo do GC busca minimizar uma determinada função quadrática, de modo que se a matriz for simétrica definida-positiva o gradiente dessa função $f(x)$ resume-se a $\nabla f(x) = b - Ax$. Como se deve minimizar $\nabla f(x)$, deve-se determinar x tal que $\nabla f(x) = b - Ax = 0$, ou seja, $Ax = b$, significando que ao encontrar a solução do GC encontra-se a solução do sistema de equações.

O GC é composto basicamente de operações de álgebra linear e, nesse caso, o paralelismo é extraído executando em paralelo as operações de subtração de vetores, adição de vetores, multiplicação de vetor por escalar, produto escalar entre vetores e multiplicação de matriz esparsa por vetor. Essa última operação é a de maior custo computacional.

O particionamento dos dados é feito considerando que a matriz de coeficientes, o vetor de incógnitas e o vetor solução são divididos entre todos os processos. Cada processo terá n/p linhas da matriz de coeficientes e i/p elementos do vetor de incógnitas e do vetor solução, onde n é o número de linhas da matriz, i é o número de elementos do vetor de de incógnitas e de solução.

Na operação de produto escalar entre dois vetores é necessário a comunicação entre todos os nodos envolvidos. Cada nodo calcula o produto escalar sobre os dados que possui, e depois é feita uma operação de redução entre todos processos. Na operação de redução o produto escalar total é calculado somando os produtos escalares de todos os nodos.

A operação mais custosa é a de multiplicação matriz por vetor. Em multiplicações de matrizes esparsas, cada processo necessita apenas de uma parte do vetor. Essa parte corresponde à sua porção do vetor e mais alguns elementos adicionais dos nodos vizinhos, logo, necessitando de comunicação para suprir esta necessidade de dados.

Nas demais operações, cada nodo executa a operação sobre as partes dos vetores que lhe cabem, sem que haja necessidade de comunicação entre os nodos envolvidos.

Para cada operação de álgebra linear foram utilizadas t *threads*, onde cada *thread* manipula uma parte dos dados. O valor de t corresponde ao número de processadores existentes em cada máquina. A Figura 2 apresenta a versão do GC descrita por Shewchuk [18].

3.3. Decomposição de Domínio

Métodos de decomposição de domínio (MDD) designam um conjunto de técnicas e métodos matemáticos, numéricos e computacionais para resolver problemas em computa-

$it \Leftarrow 0$
$r \Leftarrow b - Ax$
$d \Leftarrow r$
$\delta_{novo} \Leftarrow r^T r$
$\delta_0 \Leftarrow \delta_{novo}$
enquanto $it < it_{max}$ e $\delta_{novo} > \varepsilon^2 \delta_0$ faça
$q \Leftarrow Ad$
$\alpha \Leftarrow \delta_{novo} / d^T q$
$x \Leftarrow x + \alpha d$
$r \Leftarrow r - \alpha q$
$\delta_{velho} \Leftarrow \delta_{novo}$
$\delta_{novo} \Leftarrow r^T r$
$\beta \Leftarrow \delta_{velho} / \delta_{novo}$
$d \Leftarrow r + \beta d$
$it \Leftarrow it + 1$

Figura 2. Algoritmo do gradiente conjugado

dores paralelos. Um MDD é caracterizado pela divisão do domínio computacional, que é particionado em subdomínios empregando algoritmos de particionamento. A solução global do problema é, então, obtida através da combinação dos subproblemas que são resolvidos localmente. Cada processador é responsável por encontrar a solução local de um ou mais subdomínios, que a ele são alocados e, então, essas soluções locais são combinadas para fornecer uma aproximação para a solução global [19].

Em abordagens paralelas via decomposição de domínio os subdomínios podem ser tratados quase que independentemente e, portanto, tais métodos são atrativos para ambientes de memória distribuída [17].

De fato, alguns dos principais atrativos para o uso de MDDs são [19]:

- A necessidade de pouca comunicação, a qual, em geral, fica restrita às fronteiras dos subdomínios;
- A versatilidade para trabalhar com distintos modelos matemáticos que são definidos em diferentes subregiões do domínio global;
- Podem ser utilizados para a construção de précondicionadores para métodos iterativos.

Neste trabalho, o MDD implementado é o método aditivo de Schwarz. Este método caracteriza-se pela decomposição de um domínio global Ω em n subdomínios Ω_i sobrepostos, conforme ilustrado na Figura 3.

No MDD aditivo de Schwarz os subdomínios utilizam condições de contorno do tipo Dirichlet [16], que são obtidas através do conhecimento dos valores das células adjacentes aos subdomínios vizinhos (região de sobreposição) na iteração anterior. Assim, os sistemas de equações lo-

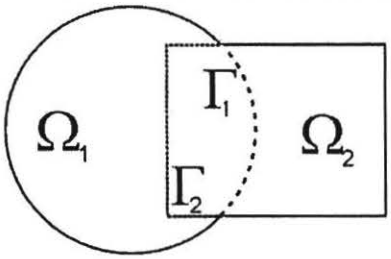

Figura 3. Decomposição do Domínio Ω e Sobreposição

cais podem ser resolvidos independentemente, apresentando um bom grau de paralelismo, necessitando de pouca comunicação.

O método aditivo de Schwarz pode ser escrito, na forma discreta, como:

$$\begin{cases} L_i u_i^n = f_i, u \in \Omega_i \\ u_i^n = g, u \in \partial\Omega_i \backslash \Gamma \\ u_i^n = g^{n-1}, u \in \Gamma \end{cases}$$

onde $L_i u_i^n = f_i, u \in \Omega_i$ representa a solução no interior de Ω; $u_i^n = g, u \in \partial\Omega_i \backslash \Gamma$ representa a solução na fronteira real de Ω; e $u_i^n = g^{n-1}$ a solução na fronteira artificial Γ de Ω.

Note-se que para resolver o método na iteração n é necessário o conhecimento dos valores das células de contorno na iteração anterior $(n-1)$, como pode ser visto em $u_i^n = g^{n-1}$. Assim, para resolver o problema em paralelo em arquiteturas de memória distribuída deve-se trocar informações entre os subdomínios, ou seja, é necessário o intercâmbio das condições de contorno (CC) entre as fronteiras artificiais geradas pelo particionamento. As diferentes possibilidades para a troca de informação determinam o particular MDD.

Na versão aditiva, utilizada neste trabalho, todos os subdomínios usam a solução da última iteração em cada subdomínio como CC para os subdomínios adjacentes, de modo que cada um deles pode ser resolvido independentemente, ficando as comunicações restritas às fronteiras. Além disso, supondo que $\Omega_i \cap \Omega_j \cap \Omega_k \neq \emptyset, \forall i \neq j \neq k$ pode-se mostrar que o algoritmo converge, e a presença de regiões sobrepostas assegura a continuidade da solução e de suas derivadas [7].

O nível de sobreposição mínima requerida depende do particular método usado para aproximar as EDPs (estêncil computacional). O uso do MDD aditivo de Schwarz requer troca de informações a cada iteração do método de Schwarz (ciclo de Schwarz), a fim de estabelecer CC homogêneas. Como as matrizes locais são simétricas definidas positivas (SDP) empregou-se como resolvedor local um GC com múltiplas *threads*, de modo que a cada ciclo do método de Schwarz trocam-se as CCs atualizadas até que seja satisfeito um determinado critério de convergência global. Neste trabalho utilizou-se sobreposições de duas células, visto que é este nível de dependência de dados requerido pelo estêncil computacional.

4. Resultados Obtidos

Neste capítulo são apresentados os resultados obtidos com as paralelizações desenvolvidas neste trabalho. Essas foram implementadas em linguagem C, utilizando o compilador *gcc* 2.95.4 sobre o sistema operacional Debian Linux com *kernel* 2.4.21. Como biblioteca de troca de mensagens foi utilizado o MPICH 1.2.2, para a criação e gerenciamento de *threads* através de OpenMP, utilizou-se pré-compilador Omni 1.4a.

Para a execução e avaliação das implementações desenvolvidas utilizou-se o *cluster* LabTeC do Instituto de Informática da UFRGS. Esse *cluster* faz parte do Laboratório de Tecnologia em *Clusters* (LabTeC), que é um laboratório de pesquisa desenvolvido pelo Instituto de Informática da UFRGS em convênio com a empresa Dell Computadores.

O *cluster LabTeC* é constituído por 21 nodos, onde 20 desses são dedicados exclusivamente para processamento e 1 nodo é servidor. A interconexão dos nodos de processamento é feita através de um *switch Fast Ethernet*, que é interconectado ao nodo servidor através de uma rede *Gigabit Ethernet*.

No que se refere aos nodos desse *cluster*, cada nodo de processamento do *cluster LabTeC* é um *Dual Pentium III* 1.1 GHz, com 1 GB de memória RAM, 512 KB de *cache* e disco rígido SCSI com 18 GB; o nodo servidor é um *Dual Pentium IV Xeon* 1.8 GHz, com 1 GB de memória RAM e disco rígido SCSI com 36 GB.

Nas soluções locais utilizou-se múltiplas *threads* na exploração do paralelismo intra-nodal. Dessa forma requer-se uma menor quantidade de subdomínios, o que resulta em uma melhor taxa de convergência dos métodos e, por conseqüência, um menor volume de troca de mensagens. Com a diminuição do número de subdomínios ocorre a redução da comunicação entre processos.

Para a realização dos testes utilizou-se sistemas resultantes da discretização do Lago Guaíba com 3 diferentes tamanhos de células: $\triangle x = \triangle y = 200m$, que resulta em sistemas de equações com 11.506 equações, $\triangle x = \triangle y = 100m$, que resulta em sistemas de equações com 46.024 equações, e $\triangle x = \triangle y = 50m$, que resulta em sistemas de equações com 184.096 equações. Esses sistemas são chamados, respectivamente, de Guaíba200, Guaíba100 e Guaíba50.

As Figuras 4, 5 e 6 mostram um comparativo dos tem-

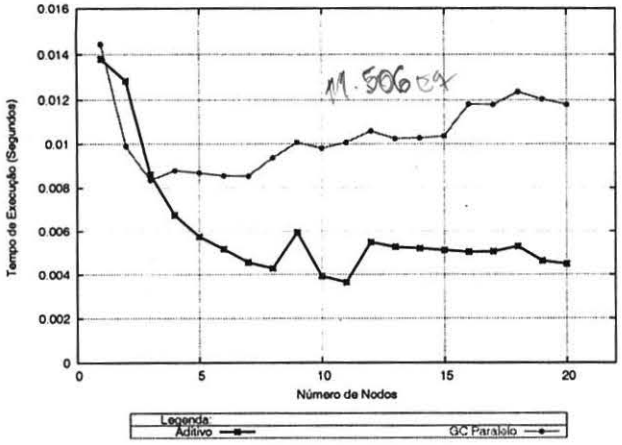

Figura 4. Guaíba200 - Método aditivo de Schwarz Vs. GC paralelo

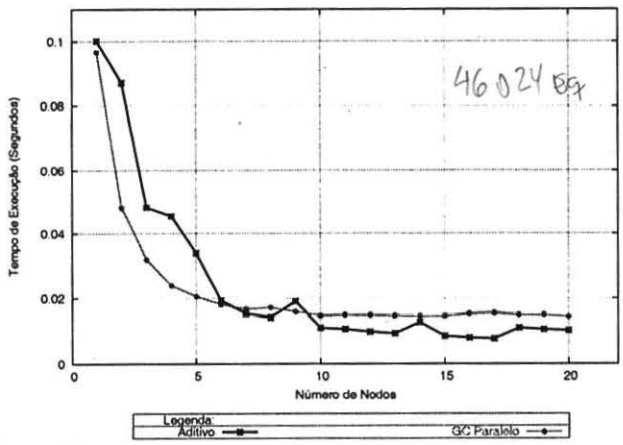

Figura 5. Guaíba100 - Método aditivo de Schwarz Vs. GC paralelo

pos de execução[1] dessas implementações. É possível verificar nessas figuras que para poucos subdomínios o método do GC paralelo possui um melhor desempenho. À medida em que o número de subdomínios aumenta, e conseqüentemente a granularidade do problema diminui, o método aditivo de Schwarz passa a apresentar melhores desempenhos. Isso porque o método aditivo de Schwarz utiliza-se, em grande parte, de dados locais, necessitando de pouca comunicação. Já o método do GC paralelizado exige um

1 tempos exigidos para a convergência de cada um dos métodos utilizados, considerando-se seus critérios de convergência. O critério de convergência dos algoritmos do GC, tanto o paralelo quanto o local, é 10^{-6}, já o critério do ciclo do MDD aditivo de Schwarz é 10^{-3}.

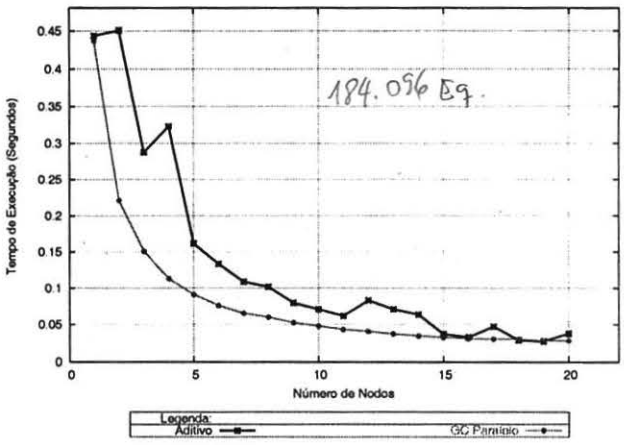

Figura 6. Guaíba50 - Método aditivo de Schwarz Vs. GC paralelo

elevado número de operações de produto escalar a cada iteração e essa operação, quando executada em paralelo provoca sincronização de todos processos. Assim, devido à necessidade de um menor volume de comunicação pode-se concluir que o método aditivo de Schwarz apresenta uma maior escalabilidade.

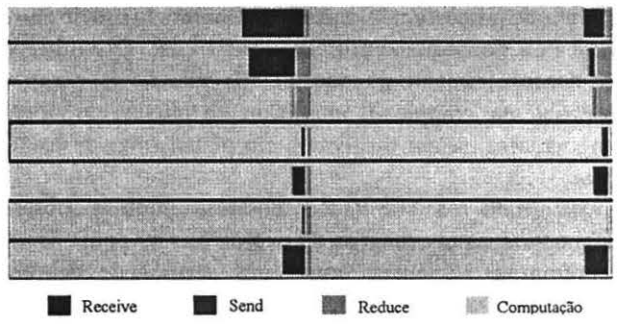

Figura 7. Execução do método aditivo de Schwarz

As Figuras 7 e 8 ilustram as execuções em paralelo dos métodos aditivo de Schwarz e do GC paralelizado, respectivamente, quando executados para o domínio Guaiba100 com 7 subdomínios. Para a geração da figura utilizou-se a biblioteca MPE (*MultiProcessing Environment*) disponível com a distribuição MPICH. Essa biblioteca contém rotinas que permitem a geração de arquivos de LOG, que descrevem as etapas de cada execução de um programa MPI [5]. Para a visualização dos arquivos de LOG utilizou-se a ferramenta Jumpshot-4 [4].

Através da análise das Figuras 7 e 8 é possível identifi-

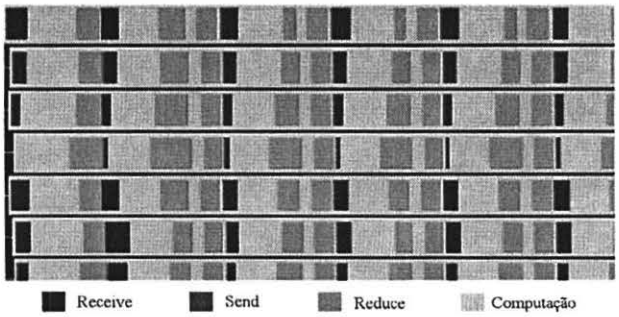

Figura 8. Execução do GC em paralelo

car que o GC paralelo apresenta um pior desempenho em relação ao método aditivo de Schwarz, com o aumento no número de processos (subdomínios), devido à necessidade de um maior volume de comunicações. As operações de *reduce* indicam, em ambos os métodos, uma operação de produto escalar e ocasionam uma sincronização de todos os processos da aplicação e, conseqüentemente, uma queda no desempenho da aplicação.

5. Conclusão

O desenvolvimento de aplicações paralelas é hoje uma abordagem indispensável em áreas que requerem grande capacidade de processamento. A disseminação do uso de *clusters* de PCs, conseqüência direta do baixo custo do hardware e do desenvolvimento de redes de alta velocidade, permite que problemas complexos anteriormente tratados em supercomputadores, sejam resolvidos em plataformas de baixo custo.

Analisando os resultados obtidos, pode-se observar que as duas abordagens de exploração do paralelismo mostraram-se eficientes, com bons ganhos de desempenho na solução de sistemas de equações. O gradiente paralelizado mostrou-se melhor para uma quantidade menor de processos, enquanto o MDD aditivo de Schwarz mostrou-se mais escalável, comportando-se melhor com quantidades mais elevadas de processsadores.

Referências

[1] OPENMP: Simple, Portable, Scalable SMP Programming. Disponível em: http://www.openmp.org/. Acesso em: out. 2003, 2003.

[2] R. Buyya. *High Performance Cluster Computing: Architecture and Systems*, volume 1. Prentice Hall, 1999.

[3] A. P. Canal. Paralelização de Métodos de Resolução de Sistemas Lineares Esparsos com o DECK em Clusters de PCs. Master's thesis, Instituto de Informática, UFRGS, Porto Alegre, 2000.

[4] A. Chan, D. Ashton, R. Lusk, and W. Gropp. Jumpshot-4's User's Guide, 2003. Disponível em http://www-unix.mcs.anl.gov/perfvis/software/viewers/. Acesso em: out. 2003.

[5] A. Chan, W. Gropp, and E. Lusk. User's Guide for MPE: Extensions for MPI programs, 2003. Disponível em http://www-unix.mcs.anl.gov/mpi/mpich/. Acesso em: out. 2003.

[6] A. S. Charão. *Multiprogrammation Parallèle Générique des Méthodes de Décomposition de Domaine*. PhD thesis, Institut National Polytechnique de Grenoble, 2001.

[7] L. Debreu and E. Blayo. On the Schwarz Alternating Method for Solving Oceanic Models on Parallel Computers. *Journal of Computational Physics*, 141:93–111, 1998.

[8] R. V. Dorneles. *Particionamento de Domínio e Balanceamento de Carga no Modelo HIDRA*. PhD thesis, Instituto de Informática, UFRGS, Porto Alegre - RS, 2003.

[9] A. O. Fortuna. *Técnicas Computacionais para Dinâmica dos Fluidos*. Editora da Universidade de São Paulo, 2000.

[10] G. Galante, A. L. Martinotto, R. L. Rizzi, and T. A. Diverio. Solução Paralela de Sistemas de Equações Lineares Através de Métodos de Decomposição de Domínio. In *ERAD 2004*. Pelotas, RS, 2004.

[11] M. R. Garey and D. S. Johnson. *Computer and Intractability: a Guide to the Theory of NP-completeness*. Freeman, San Francisco, 1979.

[12] G. Karypis and V. Kumar. METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-reducing Orderings of Sparse Matrices, 1998. Disponível em: http://www.cs.umn.edu/~karypis. Acesso em: out. 2003.

[13] A. L. Martinotto, D. Picinin, R. L. Rizzi, R. V. Dorneles, and T. A. Diverio. Paralelização de Métodos Númericos para a Solução de Sistemas de Equações Lineares. In *ERAD 2004*. Pelotas, RS, 2004.

[14] P. S. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, San Francisco, 1997.

[15] D. Picinin. Paralelização de Métodos Numéricos em Clusters Empregando as Bibliotecas MPI, DECK e Pthreads. Master's thesis, Instituto de Informática, UFRGS, Porto Alegre, 2003.

[16] R. L. Rizzi. *Modelo Computacional Paralelo para a Hidrodinâmica e para o Transporte de Massa Bidimensional e Tridimensional*. PhD thesis, Instituto de Informática, UFRGS, Porto Alegre, 2002.

[17] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, 1996.

[18] J. R. Shewchuk. An Introduction to the Conjugate Gradient Method without the Agonizing Pain. Disponível em: http://www.cs.cmu.edu/~jrs/jrspapers.html . Acesso em: out. 2003, 1994.

[19] B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Pres, Cambridge, 1996.

[20] M. Snir, S. Otto, S. Huss-Ledermann, D. Walker, and J. Dongarra. *MPI: The Complete Reference*. MIT Press, 1996.