Utilizando MPI na Paralelização do Método de Decomposição de Domínio para Resolução Numérica da Equação de Poisson

Adriano M. Santos, Dany S. Dominguez, Esbel T. V. Orellana

Laboratório de Computação Científica – Universidade Estadual de Santa Cruz
LCC/UESC – Rodovia Ilhéus/Itabuna km 16 – Ilhéus – BA - Brasil
adrimedeiros123@gmail.com, dsdominguez@gmail.com, evalero@uesc.br

Abstract. In recent years the simulations of science and engineering problems have been increased demand for a higher capacity of computational resources. In this scenario the numeric resolution of partial differential equations (PDE) on larger heterogeneous regions is relevant in function of the long processing time consume. An alternative to solve this issue is the use of parallel processing techniques. In this meta-paper was used the paradigm of message passing interface on the development to efficient parallel algorithm that uses techniques of domain decomposition to solve the Poisson mixed equation in two-dimensional cartesian geometry. It was calculate measurements of performance on the parallel algorithm for different situations demonstrating the feasibility in using parallelism to solve larger simulations problems. Furthermore, it was perform computational tests to evaluate algorithm behavior between a distributed-memory parallel computer and a shared-memory parallel computer.

Keywords: Parallel processing, MPI, distributed-memory, shared-memory

1 Introdução

Numerosos fenômenos em diversas áreas das ciências e as engenharias são modelados matematicamente através de equações diferenciais parciais (EDPs). Uma EDP envolve uma função incógnita de várias variáveis independentes e suas derivadas. De forma geral, EDPs descrevem fenômenos com dependência espacial ou dependência espacial e temporal e podem ser classificadas em elípticas, parabólicas ou hiperbólicas [7]. Na maioria dos casos a resolução numérica destes modelos demanda um alto custo computacional.

Nos últimos anos a demanda de capacidade de processamento para resolver problemas maiores e mais complexos tem impulsionado o desenvolvimento das técnicas de processamento paralelo. A utilização de maquinas paralelas, clusters ou máquinas multiprocessadas, é uma tendência que visa aumentar o desempenho computacional na resolução de problemas de grande porte. Entre as técnicas de paralelismo mais difundidas podemos mencionar o paradigma de intercâmbio de mensagens [8] em arquiteturas de memória distribuída e o paradigma multithread [3, 4] em arquiteturas de memória compartilhada.

O objetivo deste trabalho é utilizar técnicas de processamento paralelo na resolução de problemas de simulação computacional de grande porte modelados por equações em derivadas parciais (EDPs), especificamente a modelagem de reservatórios utilizando a equação mista de Poisson em geometria X,Y. Pretende-se avaliar o desempenho em arquiteturas de memória distribuída e compartilhada.

Para resolver a equação mista de Poisson em grandes regiões heterogêneas utilizamos o método de decomposição de domínios [6] com elementos finitos Raviart-Thomas [9]. A versão serial do algoritmo numérico foi implementada em linguagem C, e em seguida foram implementadas versões paralelas utilizando o padrão MPI (Message Passing Interface) para 2, 4 e 8 processadores. Visando avaliar o desempenho dos algoritmos desenvolvidos foram realizados experimentos computacionais para diferentes grades espaciais e diferentes arquiteturas de maquinas paralelas. Com os resultados obtidos foi possível calcular a eficiência e o speedup [8] dos algoritmos implementados.

O speedup (S) é o ganho obtido pela versão paralela sobre a versão serial definido como $S(p) = T_s / T_p$, onde T_s é o tempo de execução do melhor algoritmo seqüencial, e T_p é o tempo de execução em uma máquina paralela com p processadores. A eficiência (E) é a medida de utilização dos processos em um programa paralelo em relação ao programa serial definida como $E(p) = S(p)/p$. De forma geral o desenvolvimento de algoritmos paralelos visa obter speedup linear e valores de eficiência próximos da unidade.

Na próxima seção apresentamos os fundamentos matemáticos da equação de Poisson e comentamos o método de decomposição de domínio. Na seção 3, apresentamos o algoritmo seqüencial para à resolução numérica da equação de Poisson. Na seção 4 abordamos os detalhes da paralelização do algoritmo. Os resultados da análise de desempenho em diferentes arquiteturas paralelas são ilustrados na seção 5. Finalmente, as conclusões do trabalho e comentários sobre trabalhos futuros são oferecidas na seção 6.

2 Equação de Poisson e Método de Decomposição de Domínio

A equação mista de Poisson bidimensional representa um processo difusivo estacionário, portanto, pode ser utilizada em simulações de exploração de campos petrolíferos. Na figura 1 ilustramos um problema clássico de recuperação avançada de petróleo, o problema dos cinco poços (*5-spot problem*). Neste tipo de problema temos um domínio retangular (campo de extração) com um poço central e quatro poços periféricos. Introduzindo uma pressão positiva nos poços periféricos, a través da injeção de um fluído ou gás, aumentamos a pressão do reservatório o que permite extrair o petróleo em sua forma bruta no poço central. Considerando condições de simetria é possível simular apenas um quarto do domínio, onde desejamos conhecer a pressão e a velocidade dos fluídos em cada ponto.

A equação mista de Poisson pode ser escrita na forma

$$\vec{q}(x,y) = -k(x,y)\vec{\nabla}p(x,y) \ , \tag{1.1}$$

$$\nabla\vec{q}(x,y) = f(x,y) \ , \tag{1.2}$$

$$\vec{q}(x,y)\bullet n_{|r} = 0 \ . \tag{1.3}$$

Onde $\vec{q}(x,y)$ é a velocidade do fluxo, $p(x,y)$ é a pressão, $k(x,y)$ é o coeficiente de permeabilidade do meio poroso, e $f(x,y)$ é a função fonte. A Eq. (1.3) representa a condição de contorno e caracteriza um domínio no qual não existe fluxo pelas fronteiras.

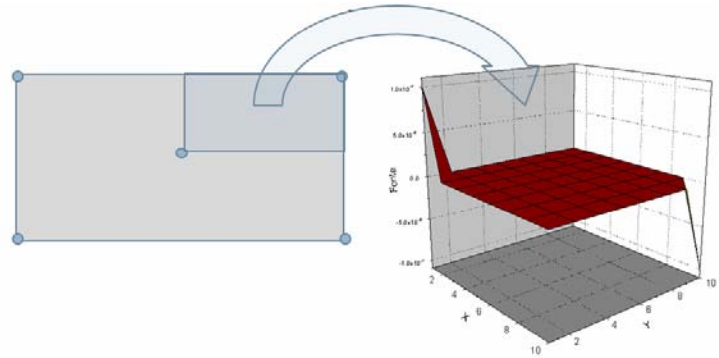

Fig. 1. Representação do problema dos cinco poços (*5-spot problem*) encontrado em campos de extração de petróleo e a função fonte $f(x,y)$ num quarto do domínio.

Para resolver o sistema de Eqs. (1) discretizamos o domínio de cálculo utilizando o método de elementos finitos de Raviart-Thomas [6, 9], para isso dividimos o domínio em células quadradas de dimensão *h*. Através do processo de discretização convertemos o sistema de EDPs (1) em um sistema de equações lineares e algébricas que pode ser resolvido numericamente. Para cada célula da grade espacial (vide Figura 2) as variáveis $p(x,y)$ e $\vec{q}(x,y)$ são discretizadas e as Eqs. (1.1 e 1.2) podem ser escritas na forma

$$q_r + q_u + q_l + q_b = f \cdot h \ , \tag{2.1}$$

$$q_\alpha = -\frac{2k}{h}(l_\alpha - p) \ , \tag{2.2}$$

$$l_\alpha = \beta_{\alpha,\bar{\alpha}} \left(q_\alpha + q_{\bar{\alpha}} \right) + l_{\bar{\alpha}} \ , \tag{2.3}$$

onde p é a pressão média na célula, l_α com (α = *r*, *u*, *l*, *b* - right, up, left e bottom) são os multiplicadores de Lagrange, e q_α são as velocidades médias dos fluidos nas arestas das células espaciais. A condição de contorno, Eq. (1.3), se transforma em

$$q_\alpha = 0 \ , \tag{3}$$

o problema proposto não possui solução única é necessário impor uma restrição adicional, isto é, impomos que a média da pressão seja igual à zero

$$\sum_{i,j=1}^{n} p_{i,j} = 0 \ . \tag{4}$$

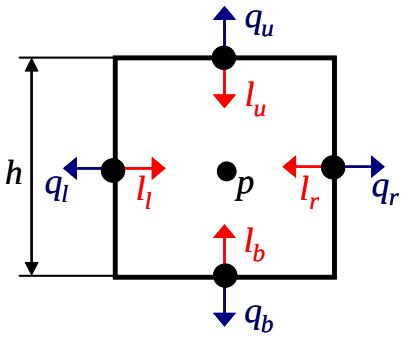

Fig. 2. Célula espacial e variáveis discretizadas pelo método de elementos finitos de Raviart-Thomas.

Ao resolvermos este problema em regiões heterogêneas de grandes dimensões, geralmente da ordem de quilômetros, desejamos conhecer as incógnitas em escalas reduzidas, da ordem de metros. A principal limitação deste enfoque é o elevado número de células espaciais e conseqüentemente o alto custo computacional.

3 Algoritmo Numérico de Resolução da Equação de Poisson

O esquema numérico proposto para resolver o sistema Eqs. (2 a 4) é um método iterativo de Gauss-Seidel [2]. A partir de uma estimativa inicial (chute) se realizam os cálculos de pressão e velocidade dos fluídos em cada célula da malha, de forma iterativa, enquanto um determinado critério de convergência não for satisfeito. Um

passo importante a ser tomado antes de iniciarmos às iterações é definir os tipos de célula, isto é a influencia que em uma célula exercem as células vizinhas segundo sua distribuição espacial. Com base neste critério temos nove tipos de células diferentes conforme Figura 3.

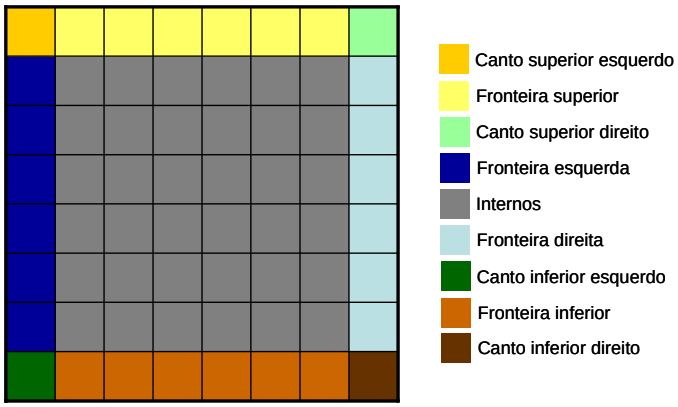

Fig. 3. Tipos de células segundo a discretização espacial e a influência das células vizinhas.

Definidos os tipos de células inicia-se o processo de cálculo. O primeiro passo é inicializar os fluxos (q_α), os multiplicadores de Lagrange (l_α) e a pressão (p) em cada uma das células com uma estimativa inicial. No segundo passo são realizados cálculos para obter novas estimativas dos fluxos e a pressão conforme as Eqs. (2.1 – 2.3) utilizando os fluxos e as pressões da estimativa anterior. Em seguida impomos a condição de normalização dada pela Eq. (4) à distribuição de pressão. Por último, é avaliado o critério de convergência das iterações que estabelece que a distancia entre duas estimativas sucessivas da distribuição de pressão não supere um valor ε. Caso o critério não seja satisfeito o algoritmo retorna ao segundo passo para computarmos uma nova estimativa dos fluxos e a pressão em cada célula. O algoritmo proposto foi implementando em linguagem C.

4 Paralelização do Algoritmo Utilizando MPI

As técnicas de paralelização baseadas em troca de mensagens foram desenvolvidas na década do 70, e orientadas a execução de algoritmos paralelos em arquiteturas de memória distribuída. Nesse tipo de máquinas cada processador tem sua memória independente, caso um processador precise de um dado armazenado na memória de outro processador um intercambio explicito de informações (troca de mensagem entre processos) é necessário. O padrão mais utilizado para troca de mensagens é o MPI (Messasge Passing Interface), com implementações disponíveis para as principais linguagens de programação.

A implementação do MPI utilizada neste trabalho foi a LAM-MPI [10] que é baseada em *daemons* e utiliza especificações tanto do MPI-1.2 quanto do MPI-2.0. O

seu funcionamento consiste inicialmente na criação de *daemons* pelo programa *lamboot* [10] em cada processador. Esses *daemons* permanecem inativos nos nós até que eles recebam uma mensagem para carregar um arquivo executável MPI e iniciar a execução.

A paralelização do algoritmo seqüencial da equação de Poisson foi realizada a partir da divisão da malha discretizada em partes iguais para cada processador seguindo o modelo SPMD (Single Program-Multiple Data) [5] onde cada processador encarrega – se de calcular a pressão e velocidades dos fluídos em sua região sobre um conjunto diferente de dados. Para a definição de tipos de células segundo sua distribuição espacial no algoritmo paralelo não devemos apenas considerar uma visão isolada de cada processador e sim uma visão em conjunto com todos os processadores (Vide Figura 4); por exemplo, o canto inferior direito do processador 3 é realmente uma fronteira direita.

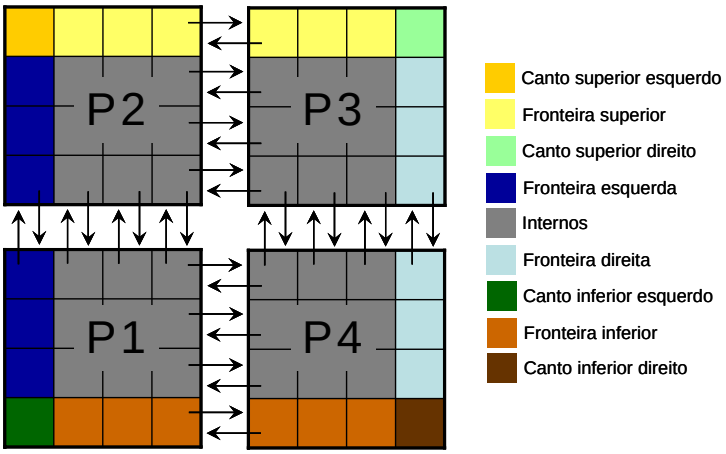

Fig. 4. Divisão do domínio e representação do intercambio de mensagens para o algoritmo paralelo com 4 processos.

Estabelecidos os tipos de células entramos no processo de cálculo com a estimativa inicial e iniciamos o processo iterativo. No final de cada iteração avaliamos o critério de convergência parcial em cada processo, em seguida obtemos a convergência do problema total através de uma operação coletiva de redução. Caso o critério não seja satisfeito cada processador se comunica com os processadores responsáveis pelas regiões vizinhas, enviando os fluxos $\left(q_\alpha\right)$ e os multiplicadores de Lagrange $\left(l_\alpha\right)$ das células fronteiriças para serem utilizadas nos cálculos das próximas estimativas (vide Figura 4). Este procedimento é repetido até que o critério de convergência seja satisfeito. Neste trabalho foram implementadas três versões do algoritmo paralelo, para 2, 4 e 8 processadores, que foram executadas em um cluster de 8 nós.

Nos últimos anos o desenvolvimento de processadores multicore tem permitido o acesso a arquiteturas de memória compartilhada com baixos custos. O desempenho destas maquinas ao executarmos programas MPI deve ser caracterizado para cada algoritmo em particular, as implementações feitas foram executadas em uma máquina

com 8 núcleos. Na próxima seção apresentamos resultados de experimentos numéricos ao executar o algoritmo descrito em máquinas de memória distribuída e compartilhada.

5 Resultados

Os testes computacionais foram realizados em um cluster Beowulf heterogêneo composto por um servidor (máquina mestre) mais oito máquinas clientes (nós) e em uma máquina de memória compartilhada com dois processadores de quatro núcleos cada. Desta forma foi possível caracterizar o algoritmo numérico de resolução da Equação de Poisson apresentado neste trabalho em diferentes arquiteturas paralelas.

Especificamente, o cluster é composto por um nó servidor que possui dois processadores Intel Xeon CPU 3050 (2.13 GHz, 2 MB de cache) com 2GB de memória RAM; um nó com um processador Intel Pentium D (2.80 GHz , 2 MB de cache) com 2 GB de RAM; três nós com processadores Intel Core 2 Duo CPU E6550 (2.33 GHz, 4 MB de cache) com 2 GB de RAM; 2 nós com processadores Genuine Intel CPU 2160 (1.80GHz, 2MB de cache) com 2 GB de RAM e 2 nós com processadores Intel(R) Pentium(R) Dual CPU E2140 (1.60 GHz, 1MB de cache) com 2 GB de RAM. O sistema operacional instalado em cada máquina é o Fedora 9 para 64 bits (x86_64). O compilador utilizado para o código serial foi gcc version 4.3.0 e para o código paralelo utilizou – se o LAM-MPI 7.1.4.

A máquina multicore é composta por 2 processadores Intel Xeon Quad-Core E5410 (2.33 GHz, 6 MB cache) com 4 GB de memória RAM. O sistema operacional instalado é o Fedora 10 para 64 bits (x86_64). O cluster e a máquina multicore foram disponibilizados pelo Núcleo de Biologia Computacional e Gestão de Informações Biotecnológica (NBCGIB) da Universidade Estadual de Santa Cruz (UESC).

Tabela 1. Número de iterações para diferentes tamanhos da malha até o critério de convergência ($\varepsilon = 1E\text{-}05$) ser satisfeito.

Tamanho da Malha	Número de Iterações
128×128	11595
256×256	29252
512×512	58144
1024×1024	81502

Foram realizados testes em 4 tipos diferentes de cenários com malhas de 128×128, 256×256, 512×512, 1024×1024. A partir dos experimentos computacionais foi possível observar que o aumento do número de pontos na malha leva a um aumento significativo na quantidade de iterações necessárias para satisfazer o critério de convergência. O incremento do número de iterações aumenta o volume de operações matemáticas e a troca de mensagens entre os processadores. A quantidade de iterações necessárias para satisfazer o critério de convergência considerado ($\varepsilon=1E\text{-}05$) em cada malha pode ser vista na Tabela 1. Os valores de speedup e eficiência no

cluster e na máquina multiprocessada para os experimentos numéricos realizados são mostrados nas Figuras 5 e 6.

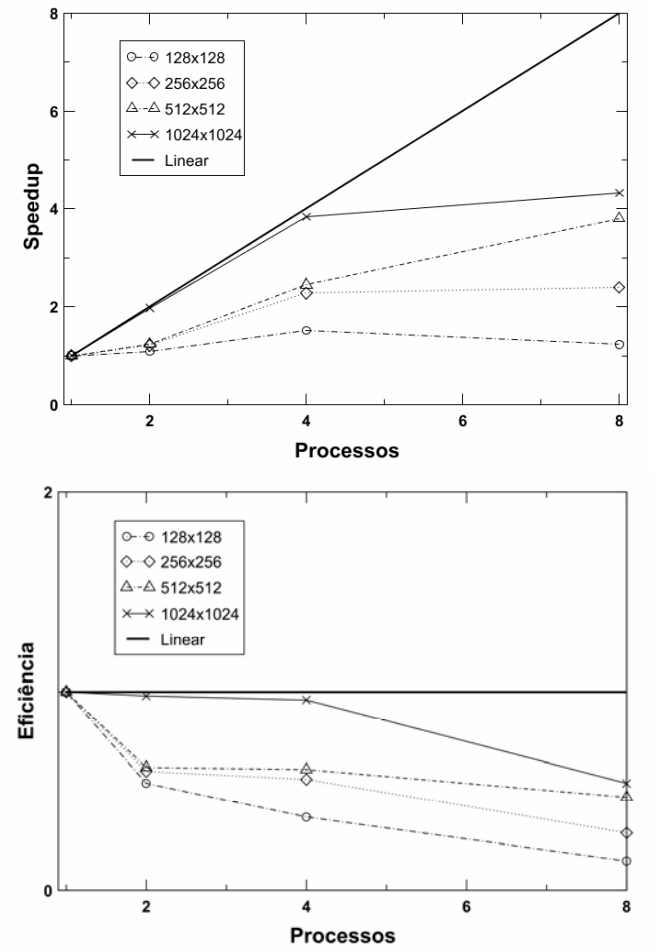

Fig. 5. Speedup e eficiência no cluster Beowful heterogêneo nas malhas consideradas (128**×**128 – 1024**×**1024).

Analisando os resultados de speedup e eficiência obtidos para o cluster (Figura 5), podemos concluir que para os dois tipos de malhas menores (128×128, 256×256) o algoritmo paralelo não teve um ganho de desempenho significativo (próximo do speedup linear), isto se deve a sobrecarga provocada pela comunicação entre processos que em cenários pequenos é maior que o ganho da paralelização. Para uma malha média (512×512) o ganho de desempenho é levemente melhor em relação às malhas menores e para cenários grandes (1024×1024) o speedup obtido é satisfatório estando próximo do speedup linear. Observamos ainda que o aumento no número de processos mantendo constante o tamanho do problema (dimensões da malha) leva a uma queda no speedup o que concorda com a interpretação da lei de Amhdal [1].

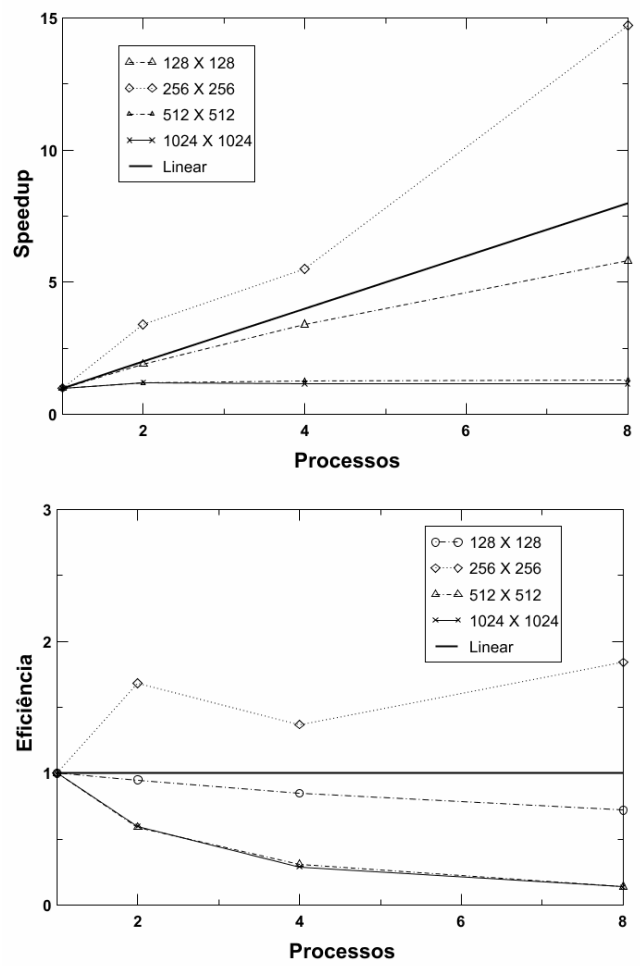

Fig. 6. Speedup e eficiência na maquina multiprocessada para as malhas consideradas (128×128 – 1024×1024).

Analisando a Figura 6, que ilustra os resultados de speedup e eficiência na maquina multiprocessada podemos concluir que para malhas pequenas temos um desempenho aceitável, inclusive temos speedup super-linear em malhas de 256×256. Para malhas medias e grandes observamos uma queda significativa no desempenho , atribuímos este fato aos mecanismos de acesso à memória compartilhada e aos algoritmos de coerência de memória cachê presentes nesta máquina.

6 Conclusões e Trabalho Futuros

A partir dos testes computacionais realizados podemos concluir que na resolução numérica da equação mista de Poisson via método de decomposição de domínio e elementos finitos de Raviart-Thomas é recomendável à utilização de técnicas de processamento paralelo para diminuir o tempo de processamento. Ademais, é importante destacar as diferenças no desempenho da implementação paralela proposta entre arquiteturas de memória distribuída e arquiteturas de memória compartilhada para as malhas consideradas. O cluster oferece melhor desempenho para malhas grandes, entretanto a maquina multiprocessada oferece melhor desempenho com malhas pequenas.

Como trabalhos futuros, propomos a implementação de um código paralelo para resolver numericamente a equação de Poisson baseada em threads utilizando OpenMP [3]. Com isto será possível comparar o desempenho do paradigma de intercâmbio de mensagens e do paradigma multithread na resolução de EDPs usando a técnica de decomposição de domínios.

Agradecimentos: Os autores do trabalho agradecem ao Conselho Nacional de Pesquisa (CNPq) pelo suporte financeiro parcial dado ao desenvolvimento deste trabalho. Agradecemos ao Núcleo de Biologia Computacional e Gestão de Informações Biotecnológicas (NBCGIB) da Universidade Estadual de Santa Cruz (UESC) por fornecer a infra-estrutura computacional necessária ao desenvolvimento deste trabalho.

Referências

1. Benner, R.E., Gustafson, J.L., and Montry, G.R., Development and analysis of scientific application programs on a 1024-processor hypercube, SAND 88-0317, Sandia National Laboratories (1988).
2. Burden, R. e Faires, D.F. Análise Numérica. Thomson Learning, São Paulo (2003).
3. Chandra R, Paralell Programing in OpenMP, Academic Press, 2001.
4. Chapman B., Jost G. e Van der Pas R., Using OpenMP, Portable Shared Memory Paralell Programming, MIT Press (2008).
5. Darema F., SPMD model: past, present and future, Recent Advances in Parallel Virtual Machine and Message Passing Interface: 8th European PVM/MPI Users' Group Meeting, Santorini/Thera, Greece, 2001. Lecture Notes in Computer Science 2131, p. 1, (2001).
6. Douglas JR, Pereira Felipe, Yeh Li Ming, Leme P. J. P.; Domain decomposition for immiscible displacement in single porosity systems. Lecture Notes In Pure And Applied Mathematics, v. 164, p. 191-199, 1994.
7. Fritz John, Partial Differential Equations, (4th ed.), Springer (1982).
8. Pacheco Peter, Paralell Programing with MPI; Acedemic Press (1996).
9. Raviart P. A. e Thoma J. M. s, A Mixed Finite Element Method for Second Order Elliptic Problems, vol. 606 of Springer Lecture Notes in Mathematics, Springer-Verlag, New York (1977).
10. Squyres J. M. e Lumsdaine A., Boot A Component Architecture for LAM/MPI, Proceedings, 10th European PVM/MPI Users' Group Meeting, Lecture Notes in Computer Science, 2840, (2003).