Análise do processamento paralelo em Clusters multi-core na simulação de escoamento míscivel implementado pelo método dos elementos finitos

Adriana C. Barbosa [1], Lucia Catabriga [2], Alberto F. De Souza [3], Andréa M.P. Valli [4]
Laboratório de Computação de Alto Desempenho - LCAD
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514, Vitória, ES
[1] adriana@lcad.inf.ufes.br, [2] lucia@inf.ufes.br, [3] alberto@inf.ufes.br, [4] avalli@inf.ufes.br

Resumo

*Neste trabalho analisamos o desempenho paralelo de um código de elementos finitos em Clusters multi-core considerando duas alternativas de particionamento do job paralelo MPI entre os cores: single-core, na qual apenas um processo é enviado para cada máquina multi-core; e fill-up, na qual todos os cores de cada máquina do Cluster recebem processos. A aplicação envolve o escoamento miscível de fluido incompressível em meio poroso, mais especificamente, a simulação de traçadores no contexto da Engenharia de Petróleo. Nós medimos o tempo de processamento e o speedup obtidos em dois Clusters de computadores multi-core: um composto por 31 máquinas, cada uma com uma Unidade Central de Processamento (*Central Processing Unit - CPU*) dual-core (Enterprise 2); e outro composto por 29 máquinas, cada uma com uma CPU quad-core (Enterprise 3). Além disso, analisamos a eficiência dos algoritmos de escalonamento no Enterprise 3 em termos da comunicação externa entre as CPUs e a concorrência dos cores pela hierarquia de memória das CPUs. Nossos resultados mostraram que, muito embora máquinas mais modernas como o Enterprise 3 possuam um número maior de cores e estes sejam mais rápidos, a competição destes cores pela hierarquia de memória tem grande impacto no desempenho final das aplicações.*

1. Introdução

O modelo matemático que descreve o problema de escoamento miscível de fluido incompressível em meio poroso consiste em um sistema acoplado de equações diferenciais parciais e não lineares de natureza elíptica, composto pela equação da pressão, equação da concentração e lei de Darcy, que relaciona a velocidade e a pressão através da permeabilidade [20]. A solução analítica desse modelo dificilmente pode ser obtida, assim é necessário utilizar métodos numéricos estáveis o suficiente para se produzir uma solução aproximada que represente satisfatoriamente o fenômeno físico. Nesse trabalho é empregada uma formulação semi-discreta estabilizada do método dos elementos finitos [3]. A formulação variacional semi-discreta conduz a um sistema de equações não lineares que pode ser solucionado por meio da resolução de uma sequência de sistemas lineares associados à pressão, velocidade e concentração. Uma vez que as características dos meios porosos requerem malhas bastantes refinadas para problemas práticos de interesse, os sistemas lineares resultantes são de grande porte. Neste caso, a classe de métodos iterativos não estacionários é essencial para garantir a obtenção da solução. Dentre os métodos iterativos não estacionários destaca-se o método dos Gradientes Conjugados para matrizes simétricas e o método do Resíduo Mínimo Generalizado (GMRES) para matrizes não simétricas [21]. As principais operações desses métodos são: produto matriz vetor e produto escalar.

As matrizes geradas pela discretização de elementos finitos são esparsas e, devido às particularidades desse método, tal esparsidade é aleatória, ou seja, os elementos não nulos ocorrem em posições arbitrárias. Visando armazenar somente os elementos diferentes de zero dessas matrizes e com isso reduzir o consumo de memória e de operações de ponto flutuante, várias estratégias podem ser usadas dentro do contexto do método dos elementos finitos, como, por exemplo, elemento por elemento (EBE) [9], adotada nesse trabalho, aresta por aresta (EDE) [4], *compressed storage row* (CSR) [21], dentre outras. Apesar da utilização de uma estratégia especial de armazenamento e solucionadores eficientes, ainda assim os problemas de escoamento de fluidos demandam um grande tempo de processamento e também um alto consumo de memória. Uma alternativa que tem se mostrado valiosa na resolução dessa classe de problemas é o processamento paralelo.

O objetivo deste trabalho é analizar o desempenho paralelo da implementação via elementos finitos do problema de escoamento de fluidos em meios porosos ou, mais especificamente, a simulação de traçadores no contexto da Engenharia de Petróleo em Clusters de máquinas com Unidade Central de Processamento (*Central Processing Unit - CPU*) multi-core. Para esse propósito, os sistemas de equações provenientes das equações governantes são solucionados através de métodos iterativos não estacionários utilizando uma versão paralela da estratégia de armazenamento elemento por elemento. Um estudo do desempenho da aplicação com relação ao algoritmo de escalonamento dos jobs nas CPUs é realizado visando observar o comportamento do código em CPUs single-core e multi-core. Nossos resultados mostram que, muito embora máquinas mais modernas possuam um número maior de cores por CPU e estes sejam mais rápidos, a competição destes cores pela hierarquia de memória das CPUs tem grande impacto no desempenho final da aplicação.

2. Formulação de elementos finitos

O problema de escoamento miscível de fluido incompressível em meio poroso é descrito por um sistema acoplado de equações diferenciais parciais, não lineares, formado pela equação da pressão (1), lei de Darcy (2) e equação de concentração (3) [20]:

$$\nabla.(-\mathbf{A}(c)\nabla p) = q \tag{1}$$

$$\mathbf{v} = -\mathbf{A}(c)\nabla p \tag{2}$$

$$\phi\frac{\partial c}{\partial t} + \mathbf{v}.\nabla c - \nabla.(\mathbf{D}(\mathbf{v})\nabla c) = 0 \tag{3}$$

onde **v** é a velocidade, q representa os termos fontes e sumidouros, c é a concentração, p é a pressão, ϕ é a porosidade do meio. O tensor $\mathbf{A}(c)$ é igual a $\frac{\mathbf{K}}{\mu(c)}$, sendo **K** o tensor de permeabilidade e $\mu(c)$ a viscosidade da mistura. O comportamento da viscosidade do fluido μ é não linear e pode ser obtido por leis empíricas, como, por exemplo, a apresentada em [23]:

$$\mu(c) = (1 - c + M^{0.25}c)^{-4}\mu_r \tag{4}$$

Como a lei da viscosidade está escrita em termos da mobilidade M e essa lei acopla o sistema, então, quanto maior o valor de M, maior é o acoplamento do sistema e maior a ocorrência das oscilações numéricas.

O tensor de difusão-dispersão adotado, **D(v)**, é o mesmo utilizado em [19], e depende da velocidade **v**, do coeficiente de difusão molecular (α_m), do coeficiente de dispersão longitudinal (α_l) e do coeficiente de dispersão transversal (α_t). Para que o problema fique completamente descrito, um conjunto apropriado de condições de contorno e condições iniciais devem ser especificados. As condições de contorno adotadas são:

$$\mathbf{v}.\mathbf{n} = 0 \text{ em } \Gamma_n, \forall t \in [0,T] \tag{5}$$

$$\mathbf{D}(\mathbf{v})\nabla c.\mathbf{n} = 0 \text{ em } \Gamma_n \tag{6}$$

$$c(x,t) = \widetilde{c} \text{ em } \Gamma_i \tag{7}$$

onde **n** representa a normal exterior ao domínio do problema Ω, sendo Γ_n e Γ_i partes do contorno de Ω. As condições inicias empregadas são:

$$p(x,0) = p_0 \text{ em } \Omega \tag{8}$$

$$c(x,0) = c_0(x) \text{ em } \Omega \tag{9}$$

$$\mathbf{v}(x,0) = 0 \text{ em } \Omega \tag{10}$$

A discretização de elementos finitos no espaço considera a divisão do domínio do problema Ω em nel elementos finitos de forma que $\Omega = \bigcup_{e=1}^{nel}\Omega^e$ e $\bigcap_{e=1}^{nel} = \oslash$, sendo nel o número de elementos totais da malha. Na discretização espacial de elementos finitos é importante considerar as particularidades e características de cada equação para se escolher uma formulação adequada, aumentado dessa forma a qualidade da aproximação.

Como o comportamento da equação da pressão (1) é predominantemente difusivo, é utilizada a formulação clássica de Galerkin que gera boas aproximações. A equação de concentração (3), por sua vez, apresenta um comportamento convectivo, logo é utilizada a formulação estabilizada SUPG que acrescenta uma quantidade necessária de difusão na direção do escoamento para eliminar as instabilidades [3]. Apesar da formulação SUPG gerar uma solução numérica estável, não evita oscilações próximas a regiões com elevados gradientes. Logo, é usado o operador de captura de descontinuidade CAU para aumentar a dissipação numérica onde a solução não é suave [8]. Todas as grandezas envolvidas no processo de solução do problema devem ter a mesma ordem de precisão para se obter um resultado satisfatório. Assim, são usados esquemas de pós-processamento para a velocidade uma vez que seu cálculo diretamente pela lei de Darcy ocasiona diferenças de precisão [15].

Como resultado das aproximações por elementos finitos é gerado um sistema acoplado, composto de três subsistemas. O primeiro diz respeito a equação da pressão e é dado por:

$$\mathbf{Kp} = \mathbf{Q} \tag{11}$$

O segundo está relacionado a equação da velocidade e é representado por:

$$\overline{\mathbf{M}}\mathbf{v} = \mathbf{F} \tag{12}$$

O terceiro refere-se a equação da concentração e é representado por:

$$\widetilde{\mathbf{M}}\mathbf{a} + \widetilde{\mathbf{C}}\mathbf{c} = 0 \tag{13}$$

sendo **p** o vetor de incógnita das pressões, **v** o vetor de incógnita das velocidades, **c** o vetor de incógnita das

concentrações e $\mathbf{a} = \frac{\partial \mathbf{c}}{\partial t}$ o vetor que contém os valores da derivada de **c**. Esses sistemas foram obtidos a partir das contribuições de todos os elementos do domínio discretizado. Montar um sistema global com as contribuições dos elementos é uma operação muito custosa. Para minimizar os impactos da esparsidade é utilizada a estratégia elemento por elemento. Assim, as matrizes **K**, $\overline{\mathbf{M}}$, $\widetilde{\mathbf{M}}$ e $\widetilde{\mathbf{C}}$ e os vetores **Q** e **F** podem ser calculados por uma soma inteligente das contribuições de cada elemento em um processo chamado de *assembling*.

Identificando por **A** a estrutura matriz e por **b** a estrutura vetor, o armazenamento elemento por elemento pode ser representado por:

$$\mathbf{A} = \mathop{\mathsf{A}}_{e=1}^{nel}(a^e) \tag{14}$$

$$\mathbf{b} = \mathop{\mathsf{A}}_{e=1}^{nel}(b^e) \tag{15}$$

onde A representa o *assembling*. As estruturas locais a^e e b^e representam, respectivamente, a matriz do elemento e e o vetor local associado. Para malhas bidimensionais de elementos triangulares lineares, utilizadas nesse trabalho, a matriz a^e é de ordem 3×3 para as matrizes referentes a pressão e concentração, e de ordem 6×6 para à matriz referente a velocidade.

3. Resolução dos sistemas

Ao final da discretização espacial é necessário solucionar os sistemas da pressão (11), concentração (13) e velocidade (12). Uma vez que a formulação variacional semi-discreta está sendo utilizada, as derivadas da concentração são discretizadas por diferenças finitas. Para tal, é utilizado um algoritmo membro da família de métodos da regra trapezoidal denominado preditor/multicorretor [10] amplamente adotado na resolução de problemas de escoamento [5, 6, 7, 17, 19].

Sendo n o contador de passos de tempo, as aproximações para $p(t_n), a(t_n), v(t_n)$ e $c(t_n)$, representadas por p_n, a_n, v_n e c_n, Δt o passo de tempo, Δa o incremento da derivada temporal da concentração e i o contador das multicorreções (iterações não-lineares), o algoritmo pode ser resumido nas seguintes etapas:

Bloco 1: Resolve a equação da pressão

$$\mathbf{K}(\mathbf{c}_{n+1}^{i})\,\mathbf{p}_{n+1}^{i+1} = \mathbf{Q}_{n+1} \tag{16}$$

Bloco 2: Calcula o campo de velocidade

$$\bar{\mathbf{M}}(\mathbf{c}_{n+1}^{i})\,\mathbf{v}_{n+1}^{i+1} = \mathbf{F}(\mathbf{p}_{n+1}^{i+1}) \tag{17}$$

Bloco 3: Resolve a equação da concentração

$$\mathbf{M}_{n+1}^{*}\Delta\dot{\mathbf{c}}_{n+1}^{i+1} = \mathbf{G}(\mathbf{p}_{n+1}^{i+1}, \mathbf{v}_{n+1}^{i+1}, \mathbf{c}_{n+1}^{i})_{n+1} \tag{18}$$

com

$$\mathbf{c}_{n+1}^{i+1} = \mathbf{c}_{n+1}^{i} + \alpha\,\Delta t\,\Delta\dot{\mathbf{c}}_{n+1}^{i+1}$$
$$\dot{\mathbf{c}}_{n+1}^{i+1} = \dot{\mathbf{c}}_{n+1}^{i} + \Delta\dot{\mathbf{c}}_{n+1}^{i+1}.$$

onde $\mathbf{M}_{n+1}^{*} = \tilde{\mathbf{M}} + \alpha\,\Delta t\,\tilde{\mathbf{C}}$ é denominada matriz efetiva e $\mathbf{G}_{n+1} = -\tilde{\mathbf{M}}(\dot{\mathbf{c}}_{n+1}^{i})\,\dot{\mathbf{c}}_{n+1}^{i} - \tilde{\mathbf{C}}(\mathbf{v}_{n+1}^{i+1}, \mathbf{p}_{n+1}^{i+1}, \mathbf{c}_{n+1}^{i})\,\mathbf{c}_{n+1}^{i}$.

O processo iterativo é finalizado quando se atinge um critério de convergência pré-determinado, que é avaliado empregando-se medidas de erros adequadas para pressão, velocidade e concentração que podem ser vistas com detalhes em [5, 19]. No algoritmo, bloco iterativo preditor-multicorretor, a cada iteração não-linear são resolvidos três sistemas lineares de equações, cujas principais operações (produto matriz vetor e produto interno) foram implementadas em paralelo utilizando as estratégias apresentadas na próxima seção.

4. Paralelização do método dos elementos finitos

A paralelização do sistema composto pelas equações (11), (12) e (13) discretizado por formulações do método dos elementos finitos utiliza a técnica de decomposição de domínios descrita em [11, 21]. Inicialmente, o domínio deve ser particionado de forma balanceada, ou seja, deve-se manter em cada partição o mesmo número de elementos e nós sempre que possível. Também é importante reduzir ao máximo os nós que pertencem à fronteira de comunicação entre as partições, o que representará uma diminuição na comunicação entre os cores.

Para realizar o particionamento foi utilizado o conhecido particionador de malhas METIS [12]. Com o domínio particionado, é possível montar as contribuições das matrizes **K**, $\overline{\mathbf{M}}$, $\widetilde{\mathbf{M}}$ e $\widetilde{\mathbf{C}}$ e dos vetores **Q** e **F** concorrentemente em cada core [14, 18, 21]. Considere a Fig. 1, que representa uma malha com 50 nós e 74 elementos particionada em quatro subdomínios, cada qual direcionado a um core. Na Fig. 1 os nós foram numerados de 1 a 50 em sentido anti-horário. Primeiro foram numerados os cantos da malha, depois as bordas e posteriormente o seu interior. Os elementos foram numerados de 1 a 74 e sua numeração está representada por números no interior de pequenos círculos. As partições obtidas possuem três tipos de nós, denominados por *IntNodes*, *IBNodes* e nós de valor conhecido. Os nós *IntNodes* são os nós incógnitas que não estão na fronteira de particionamento, ou seja, são os nós que pertencem ao interior da partição. Os nós *IBNodes* são os nós incógnitas que pertencem, simultaneamente, a mais de uma partição, ou seja,

estão posicionados na fronteira do particionamento. Para ilustrar estas denominações, considere os nós 28 e 49. Eles são nós *IntNodes* dos cores 4 e 3, respectivamente. Já o nó 27 é um nó *IBNodes* para o core 1, assim como o nó 3 é um nó *IBNodes* para o core 4 [14].

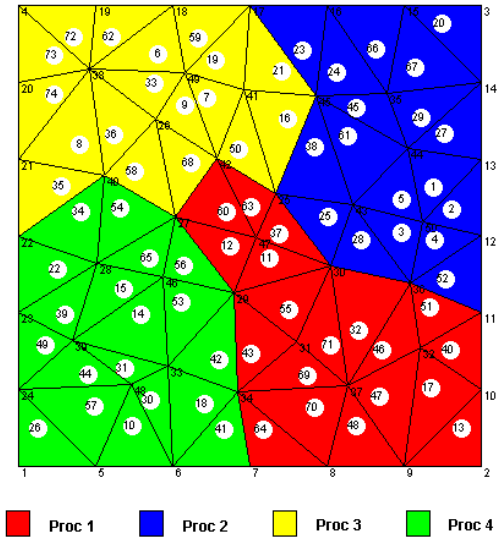


Figura 1. Malha com 4 partições

Essa forma de particionamento faz com que as incógnitas dos vetores soluções do sistema fiquem distribuídas ao longo dos p subdomínios gerados no particionamento de forma que parte do vetor pode pertencer unicamente a um core ou ser compartilhado entre vários cores. Considerando esse particionamento do domínio, os sistemas lineares resultantes da discretização, denominados aqui por $Ax = b$, são descritos da seguinte forma [14, 13]:

$$Ax = \begin{bmatrix} A_1 & & & & B_1 \\ & A_2 & & & B_2 \\ & & \ddots & & \vdots \\ & & & A_p & B_p \\ C_1 & C_2 & \dots & C_p & A_s \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \vdots \\ \underline{x}_p \\ \underline{x}_s \end{bmatrix} = \begin{bmatrix} \underline{b}_1 \\ \underline{b}_2 \\ \vdots \\ \underline{b}_p \\ \underline{b}_s \end{bmatrix} = b \tag{19}$$

sendo que os blocos A_i, B_i, C_i e A_s, com $i = 1, ..., p$, armazenam as contribuições dos elementos destinados originalmente à matriz A. O primeiro bloco, A_i, representa as contribuições dos nós *IntNodes* nos nós *IntNodes* pertencentes a partição i. O bloco B_i armazema as contribuições dos nós *IntNodes* sobre os nós *IBNodes* na partição i. O bloco C_i é similar ao bloco B_i, uma vez que relaciona nós *IBNodes* com nós *IntNodes*, mas o bloco C_i representa as contribuições dos nós *IBNodes* da partição i nos nós *IntNodes* da partição i. O bloco A_s, representa a reunião de vários blocos ao longo das p partições, de forma que, $A_s = \bigcup_{i=1}^{p} A_{s_{(i)}}$, sendo que cada uma das $A_{s_{(i)}}$ armazena as contribuições dos nós *IntNodes* sobre os nós *IntNodes* da partição i. Os vetores $\underline{x}$ e $\underline{b}$, são divididos em dois blocos, o primeiro representa as incógnitas relativas aos nós *IntNodes* da partição i, ou seja, $\underline{x}_i$ e $\underline{b}_i$, com $i = 1, ..., p$. O segundo bloco está relacionado às incógnitas dos nós *IBNodes* para a partição i isto é, $\underline{x}_s$ e $\underline{b}_s$, formados pelo arranjo de vários blocos ao longo das p partições, isto é, $\underline{x}_s = \bigcup_{i=1}^{p} \underline{x}_{s_{(i)}}$ e $\underline{b}_s = \bigcup_{i=1}^{p} \underline{b}_{s_{(i)}}$ [13].

Na resolução dos sistemas (19) duas operações básicas que tem um grande impacto no desempenho da solução é o produto matriz vetor e o produto interno entre dois vetores. No particionamento utilizado, considere uma matriz $\mathbf{A}$, um vetor $\mathbf{u}$. O produto $\mathbf{Au} = \mathbf{v}$ pode ser definido da seguinte forma [14]:

$$\mathbf{v}_i = \mathbf{A}_i\mathbf{u}_i + \mathbf{B}_i\mathbf{u}_{s_{(i)}} \tag{20}$$

na partição i, para $i = 1, ..., p$ e

$$\mathbf{v}_{s_{(i)}} = \mathbf{C}_i\mathbf{u}_i + \mathbf{A}_{s_{(i)}}\mathbf{u}_{s_{(i)}} \tag{21}$$

O cálculo do produto interno pode causar uma significante perda de desempenho como resultado do custo da comunicação no processamento em paralelo, visto que, utiliza todos os elementos de um vetor e deve ser comunicado a todos as partições. Assim, o cálculo do produto interno necessita de uma comunicação global. Considere os vetores $\mathbf{u}$ e $\mathbf{v}$ distribuídos por p partições. O produto interno entre esses dois vetores pode ser expresso por:

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{p} (\mathbf{u}_i \cdot \mathbf{v}_i + \mathbf{u}_{s_{(i)}} \cdot \mathbf{v}_{s_{(i)}}) \tag{22}$$

5. Experimentos numéricos

A simulação de injeção de traçadores é muito útil para se obter informações sobre o meio poroso, como barreiras ao escoamento, direção do escoamento, dispersividade, etc. No contexto da Engenharia de Reservatórios de Petróleo, traçadores são utilizados para monitorar o movimento do fluido a fim de adquirir informações sobre as propriedades do reservatório, uma vez que o tempo de chegada do traçador aos poços produtores e a sua concentração são informações muito úteis [2]. Neste trabalho, na simulação do problema do traçador foi utilizado o esquema de injeção conhecido como *five-spot*.

Considere um reservatório de petróleo hipotético de geometria quadrada constituído por um poço produtor localizado no centro do reservatório e quatro poços injetores dispostos nos vértices do reservatório. Neste arranjo, o escoamento é simétrico com relação às duas direções; portanto,

o domínio computacional pode ser representado apenas por um quarto do domínio.

Em nossas simulações, o lado do reservatório mede $L = 1000.0ft$, o poço injetor está localizado no canto inferior esquerdo ($x = y = 0$) e o poço produtor no canto superior direito ($x = y = 1000$). O meio poroso é homogênio com permeabilidade $\mathbf{K} = k\mathbf{I}$, sendo $k = 100mD$, e possui porosidade $\phi = 0.1$. Os níveis de dispersão do traçador são $\alpha_l = 10.0$, $\alpha_t = 1.0$ e $\alpha_m = 0.0$, e a viscosidade do fluido residente é $\mu(1) = 1.0cP$.

No problema de injeção de traçador, um volume de traçador é injetado no poço injetor e é movimentado, por exemplo, pelo escoamento de água, sendo retirado no poço produtor. Em nossas simulações, o volume do traçador injetado é de 0,25% do volume poroso, o que equivale a uma injeção realizada por um período de 5 dias a uma taxa de $200ft^2/dia$. Nas simulações, a viscosidade do traçador é a mesma do fluido residente, assim, a razão de mobilidade M é unitária. Portanto, o problema é linear e o escoamento livre de qualquer instabilidade física ou numérica e, além disso, a orientação da malha não influência o resultado [1].

O escoamento do traçador foi observado durante 800 dias. O domínio considerado é representado por um malha estruturada com 80×80 células, onde cada célula está dividida em dois triângulos, resultando em 263169 nós e 524288 elementos. A tolerância utilizada para o GMRES é 10^{-3} e a base é composta por 5 vetores de Krylov, GMRES(5). A Fig. 2 apresenta as isocurvas de concentração da solução obtida, que estão de acordo com a literatura [16].

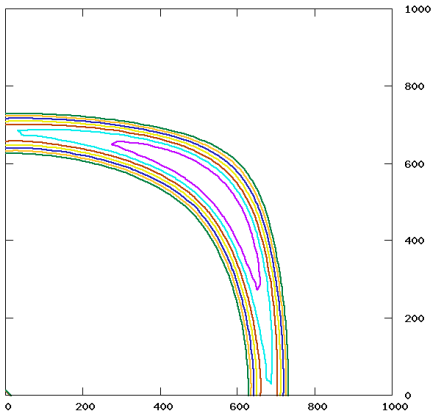


Figura 2. Isocurvas de concentração do traçador

As simulações foram realizadas em 2 Clusters do Laboratório de Computação de Alto Desempenho da Universidade Federal do Espírito Santo. O primeiro Cluster, denominado Enterprise 2, é composto por 31 máquinas, cada uma com uma CPU dual-core, totalizando 62 cores. As CPUs são Intel Core 2 4300, que possuem frequência de clock igual a 1.8 GHz e cache L2, compartilhada entre os cores, de 2 MB. O segundo Cluster, denominado Enterprise 3, é composto por 29 máquinas, cada uma com uma CPU quad-core, totalizando 116 cores. As CPUs de Enterprise 3 são Intel 2 Q6600, que possuem frequência de clock igual a 2.4 GHz e cache L2 compartilhada de 4 MB.

A distribuição de processos (jobs) entre os cores de Enterprise 2 e Enterprise 3 é realizada por meio da ferramenta Sun Grid Engine (SGE), da Sun [22], que permite variar o algoritmo de distribuição de jobs. No presente trabalho, foram utilizados dois algoritmos de distribuição: *fill-up* e *single-core*. No *fill-up*, o SGE distribui os jobs de forma a ocupar todos os cores de uma CPU para então passar para a CPU seguinte. No *single-core*, o SGE considera que há apenas um core por CPU. Nós examinamos o desempenho da aplicação considerando esses dois algoritmos de distribuição.

O particionamento do domínio foi feito de forma balanceada pelo particionador de malhas METIS [12], que toma o cuidado de numerar as partições próximas com números consecutivos. Isso permite garantir que, na distribuição dos processos em ambiente multi-core, processos vizinhos fiquem em cores de uma mesma CPU, reduzindo o tempo de comunicação via interface de rede, uma vez que o MPI usa comunicação através da memória entre cores de uma mesma CPU multi-core. A Fig. 3 ilustra um particionamento realizado pelo METIS de uma malha com 256×256 células de elementos triangulares (16 partições).

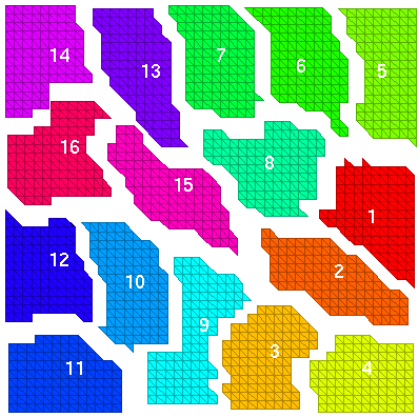
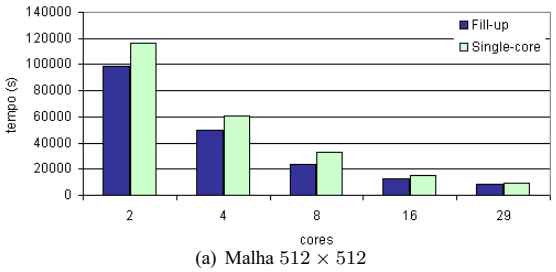


Figura 3. Malha 256×256 particionada pelo programa METIS em 16 partições

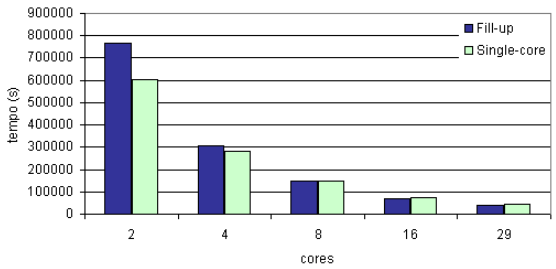
O desempenho paralelo dos algoritmos desenvolvidos foi testado em duas malhas, uma de tamanho pequeno, com 512×512 células, e uma outra grande, com 2048×2048

células, ambas com 2 elementos triangulares por célula. A malha pequena possui 524288 elementos e 263169 nós, enquanto que a malha grande possui 8388608 elementos e 4198401 nós. Considerando que (i) serão resolvidos três sistemas lineares, um para pressão, um para velocidade e um para concentração, (ii) a velocidade possui dois graus de liberdade, e (iii) são necessários vetores auxiliares para os métodos GMRES(5) e Gradiente Conjugado, pode-se estimar a quantidade de memória usada pela aplicação em ambas as malhas. Na malha pequena, o programa utiliza em torno de 382 MB, enquanto que, na malha grande, 1,5 GB.

A Fig. 4 mostra o tempo de processamento para as malhas 512×512 (Fig. 4(a)) e 2048×2048 (Fig. 4(b)) no Cluster Enterprise 3, considerando os dois algoritmos de escalonamento e diferentes números de cores. Nos gráficos da Fig. 4, o eixo y apresenta o tempo de processamento e o eixo x o número de cores empregado; as barras mais escuras indicam o tempo de execução para o escalonamento *fill-up* e as mais claras o tempo de execução para o *single-core*.



(a) Malha 512×512



(b) Malha 2048×2048

Figura 4. Tempo de processamento na Enterprise 3 - Problema do Traçador

Nos gráficos da Fig. 4, o par de barras mais à esquerda (2 cores) apresenta o tempo de execução com o domínio dividido em 2 sub-domínios, sendo que a barra mais à esquerda do par (*fill-up*) mostra o tempo de execução com os jobs responsáveis por ambos sub-domínios rodando em cores de uma única CPU (comunicação via memória), enquanto que a barra mais à direita do par (*single-core*) mostra o tempo de execução com os jobs executando em cores de CPUs distintas (comunicação via rede); o segundo par de barras (da esquerda para a direita) mostra o tempo de execução com 4 jobs (responsáveis por 4 sub-domínios) rodando em 4 cores de uma mesma CPU (barra mais à esquerda do par) ou 4 jobs rodando em 4 cores de CPUs distintas (barra mais à direita); e assim sucessivamente.

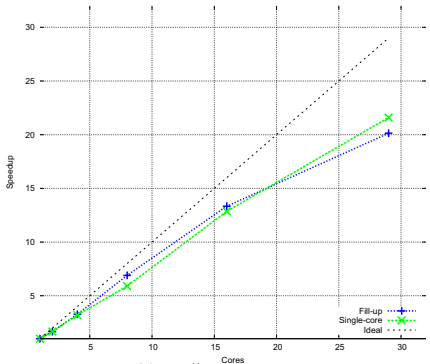
Como a Fig. 4(a) mostra, na malha menor (512×512), o escalonamento *fill-up* proporciona tempo de processamento menor que o escalonamento *single-core* para todos os números de cores empregados. Isso era esperado, já que, no escalonamento *fill-up*, toda (com 2 e 4 cores) ou boa parte (com 8, 16 e 29 cores) da comunicação é feita via memória ao invés de via rede. Na malha maior (2048×2048, Fig. 4(b)) o mesmo não ocorre—o escalonamento *single-core* proporciona tempo de processamento menor ou equivalente ao escalonamento *fill-up* para 2, 4 e 8 cores. Isso acontece porque, com subdomínios maiores, os benefícios obtidos com a menor quantidade de comunicação do escalonamento *fill-up* são sobrepujados pelo custo da competição dos cores pela hierarquia de memória das CPUs que os abriga; concomitantemente, a razão comunicação / computação também diminui e, com ela, os benefícios do escalonamento *fill-up*. No entanto, a medida que aumentamos o número de cores, o tamanho dos sub-domínios diminui, o custo da competição pela hierarquia de memória diminui, e a razão comunicação / computação aumenta; tudo isso beneficia o escalonamento *fill-up*, que supera (menor tempo de execução) o *single-core* para 8 ou mais cores (Fig. 4(b)).

Fig. 5 mostra o speedup na Enterprise 3 para os dois tipos de escalonamento. O cálculo do speedup foi feito dividindo-se o tempo de execução em um core pelo tempo de execução em dois ou mais cores; o mesmo código foi usado em todos os casos. É importante destacar que não executamos o código em um único core para a malha maior (2048×2048) devido ao tempo de processamento ser muito grande. Assim, neste caso, o speedup foi calculado dividindo-se o tempo de execução em 2 cores multiplicado por 2, pelo tempo de execução em 2 ou mais cores.

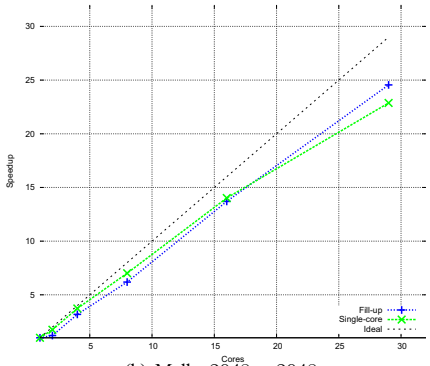
Como a Fig. 5 mostra, o desempenho do código escala razoavelmente bem para ambas as malhas, muito embora escale melhor para a malha maior (2048×2048) e com escalonamento *fill-up* (Fig. 5(b)). Isso era esperado, já que, com a malha maior, a razão comunicação / computação é menor.

No último experimento analisamos o escalonamento *fill-up* nos Clusters Enterprise 2 e Enterprise 3 para a malha maior (2048×2048), e considerando uma quantidade maior de cores. A Fig. 6 apresenta o tempo de processamento nos dois clusters neste caso; ela segue o mesmo padrão da Fig. 4, sendo que as barras escuras mostram o tempo de processamento da Enterprise 2 e as claras o da Enterprise 3.

Como a Fig. 6 mostra, com 4 cores o tempo de proces-



(a) Malha 512×512



(b) Malha 2048×2048

Figura 5. Speedups no Enterprise 3

samento do Enterprise 2 é surpreendentemente menor que o do Enterprise 3, muito embora as máquinas que compõem o Enterprise 3 sejam superiores. Isso ocorre porque, com 4 cores, a competição destes pela hierarquia de memória no Enterprise 3 impacta mais no tempo de processamento que a comunicação via rede no Enterprise 2 e o hardware inferior deste. No entanto, com 8 cores a situação se inverte, pois a redução do tamanho dos subdomínios neste caso reduz a competição pela hierarquia de memória, o que beneficia o Enterprise 3. A vantagem do Enterprise 3 diminui à medida que o número de cores aumenta, já que o aumento da razão comunicação / computação passa a afetá-lo em medida similar ao Enterprise 2.

A Fig. 7 mostra o speedup nos dois Clusters para 4, 8, 16, 32 e 62 cores na malha 2048×2048 e com escalonamento *fill-up*. Como a figura mostra, o speedup escala bem nos dois clusters até 32 cores (um pouco melhor em Enterprise 2 do que em Enterprise 3 devido à menor competição pela uso da hierarquia de memória). Contudo, acima deste número de cores o speedup passa a não escalar bem para ambos os clusters devido ao aumento da razão comunicação / computação.

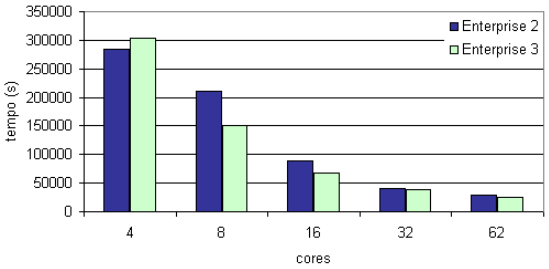


Figura 6. Tempo de processamento *Fill-up* - Malha 2048×2048

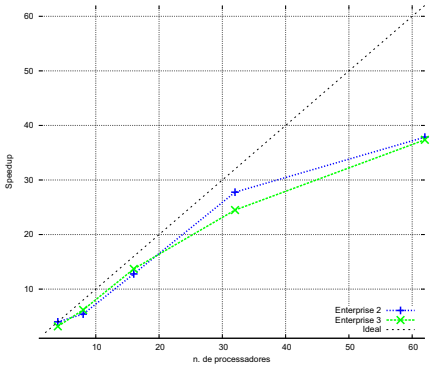


Figura 7. Speedups *fill-up* - Malha 2048×2048

6. Conclusão

Este trabalho teve como objetivo analizar o desempenho paralelo da implementação via elementos finitos do problema de escoamento de fluidos em meios porosos, especificamente na simulação de traçadores no contexto da Engenharia de Petróleo em Clusters multi-core. Para esse propósito, os sistemas de equações proveniente das equações governantes foram solucionados através de métodos iterativos não estacionários utilizando uma versão paralela da estratégia de armazenamento elemento por elemento. O estudo do desempenho da aplicação paralela MPI com relação ao algoritmo de escalonamento dos jobs em cores foi realizado visando observar o comportamento do código em CPUs single-core e multi-core. Foram considerados dois algoritmos de escalonamento: *Fill-up* e *Single-core*. Nossos resultados mostraram que, muito embora máquinas mais modernas possuam um número maior de cores por CPU e estes sejam mais rápidos, a competição destes cores pela hierarquia de memória tem grande impacto no desempenho final da aplicação.

Como trabalhos futuros iremos investigar mecanismos para reduzir o impacto da hierarquia de memória em nos-

sos algoritmos por meio da melhor utilização de eventuais localidades no acesso a dados.

Agradecimentos

Agradecemos à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de mestrado concedida à primeira autora. Os demais autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio recebido dentro do escopo do Projeto CNPq 620185/2008-2. Agradecemos também o apoio técnico fornecido pelo suporte do Laboratório de Computação de Alto Desempenho da UFES.

Referências

[1] A. C. Barbosa, L. Catabriga, A. M. P. Valli, S. M. C. Malta, and L. M. de Lima. Experiments using a finite element formulation of incompressible miscible displacements in porous media. In *Congresso Nacional de Matemática Aplicada e Computacional*, 2009.

[2] M. R. Borges. *Injeção de traçadores em reservatórios de petróleo: modelagem multi-escala e simulação numérica*. PhD thesis, IPRJ/UERJ, Rio de Janeiro, RJ, Fevereiro 2006.

[3] A. N. Brooks and T. J. R. Hughes. Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Comput. Methods Appl. Mech. and Engrg.*, 32:199–259, 1982.

[4] L. Catabriga, M. D. A. Martins, A. L. G. A. Coutinho, and J. L. D. Alves. Clustered edge-by-edge preconditioners for non-symmetric finite element equations. In *4th World Congress on Computational Mechanics*, 1998.

[5] A. Coutinho and J. Alves. Parallel finite element simulation of miscible displacement in porous media. *SPE Journal*, 4(1):487–500, 1996.

[6] A. Coutinho and J. Alves. Finite element simulation of nonlinear viscous fingering in miscible displacement with anisotropic dispersion and nonmotonic viscosity profiles. *Computational Mechanics*, 423:108–116, 1999.

[7] M. C. Dias. *Técnicas de integração reduzida para simulação de problemas não-lineares de transporte pelo método dos elementos finitos*. PhD thesis, COPPE/UFRJ, Rio de Janeiro, RJ, Novembro 2001.

[8] A. Galeao and E. Carmo. A consistent approximate upwind petrov-galerkin method for convection-dominated problems. *Computer Methods in Applied Mechanics and Engineering*, 68:83–95, 1988.

[9] T. J. R. Hughes. *The Finite Element Method.* Prentice-Hall International, 1987.

[10] T. J. R. Hughes. *The Finite Element Method - Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall International, Inc., 1987.

[11] P. K. Jimack and N. Touheed. Developing parallel finite element software using mpi, 2000.

[12] G. Karypis and V. Kumar. Metis - unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, 1995.

[13] L. Lima, B. Melotti, L. Catabriga, and A. Valli. Estratégias de armazenamento para implementacões paralelas do método dos elementos finitos. In *Computational Mechanics*, 2004.

[14] M. L. Lima. Estratégias de armazenamento paralelas aplicadas ao método dos elementos finitos. Master's thesis, UFES, Espírito Santo, ES, Dezembro 2004.

[15] S. Malta, A. Loula, and E. Garcia. A post-processing technique to approximate the velocity filed in miscible displacement simulations. *Matemática contemporânea*, 8:239–268, 1995.

[16] S. Malta, A. Loula, and E. Garcia. Numerical analysis of a stabilized finite element method for tracer injection simulations. *Computer Methods in Applied Mechanics and Engineering*, 187:119–136, 1998.

[17] A. L. Mendonça. *Simulação numérica de escoamentos incompressíveis bifásicos de fluidos não-newtonianos e imiscíveis em meios porosos via método dos elementos finitos*. PhD thesis, COPPE/UFRJ, Rio de Janeiro, RJ, Outubro 2003.

[18] S. A. Nadeem. Parallel domain decomposition preconditioning for the adaptive finite element solution of elliptic problems in three dimensions, 2001.

[19] W. G. Ney. Um estudo comparativo sobre formulações estabilizadas e adaptatividade na simulação de desolcamento miscíveis em meios pororos pelo método dos elementos finitos. Master's thesis, COPPE/UFRJ, Rio de Janeiro, RJ, Dezembro 2002.

[20] D. Peaceman. *Fundamentals of Numerical Reservoir Simulation*. Elsevier Scientific Publishing Company, Amsterdam, The Netherlands, third edition, 1982.

[21] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, 1995.

[22] Sun. *Sun Grid Engine: Enterprise Edition 5.3 Administration User's Guide*. Sun Microsystems, 2002.

[23] L. Young. A finite element method for reservoir simulation. *Society of Petroleum Engineers, SPE*, 4, 1981.