20º POSMEC

Simpósio do Programa de Pós-graduação em Engenharia Mecânica

Universidade Federal de Uberlândia

Desenvolvimento e Implementação de Metodologia de Paralelização para Malhas Adaptativas Bloco Estruturadas

Autor(es):
Rafael Sene de Lima

Orientador:
Aristeu da Silveira Neto

Considerável progresso tem sido observado no uso de malhas adaptativas para a solução numérica de equações diferenciais parciais utilizando diferenças finitas. A ideia principal seria aumentar a densidade da malha onde ocorrem fenômenos físicos importantes, afim de capturá-los na simulação numérica. Dentre os vários métodos para refinamento local adaptativo, no presente trabalho optou-se por utilizar a técnica de refinamento adaptativo bloco estruturado.

Proposta por Berger & Oliger (1984), esta técnica consiste em uma aproximação para utilização de malhas adaptativas, em que faz-se uso de uma sequência de blocos de malhas estruturadas devidamente posicionados. Tais malhas são formadas por pontos onde o erro provindo da malha mais grosseira é suficientemente elevado. Este tipo de refinamento adaptativo foi estendido para escoamentos incompressíveis por Almgren *et al.* (1998). Roma, Peskin & Berger (1984) acoplaram esta técnica ao Método da Fronteira Imersa. Recentemente, Griffith (2005) aplicou esta metodologia na simulação de escoamentos sanguíneo em um coração utilizando o framework SAMRAI (Structured Adaptive Mesh Refinement Application Infrastructure). Villar (2007) fez um estudo detalhado sobre escoamentos bifásicos utilizando o método da fronteira imersa e malhas adaptativas refinadas localmente.

No presente trabalho, propõe-se desenvolver de uma metodologia de paralelização para malhas bloco estruturadas, utilizando o padrão MPI (*Message Passing Interface*) para comunicação de dados em paralelo. O balanço de carga e o particionamento de domínio são implementados utilizando-se o pacote

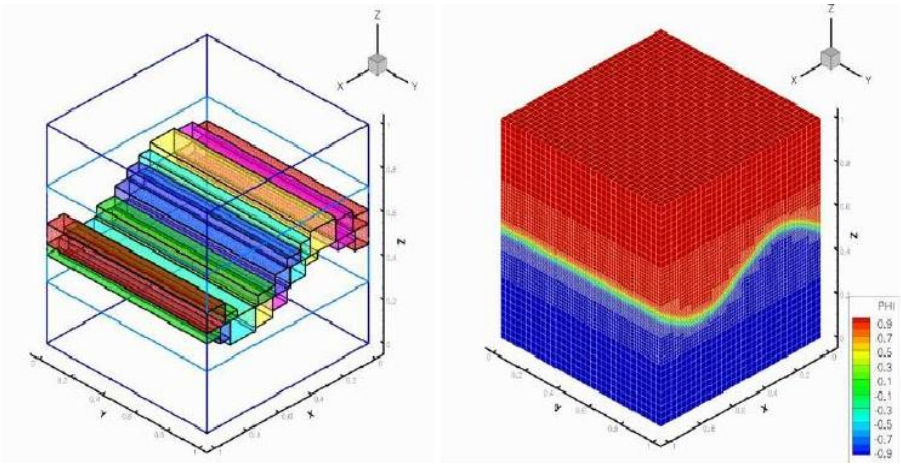
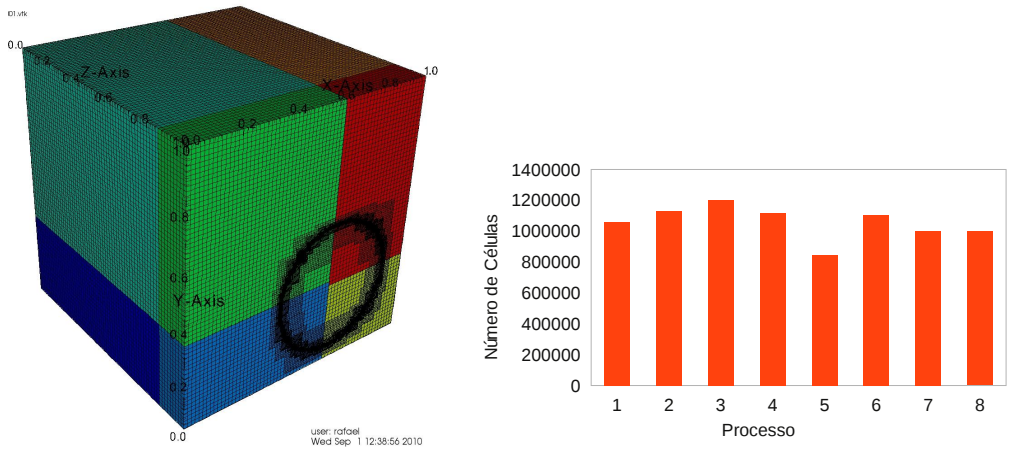


Figura 1: Exemplo de malha adaptativa bloco estruturada.

Zoltan, o qual é constituído de um conjunto de ferramentas para balanço de carga e gerenciamento de dados em paralelo, desenvolvido pela *Sandia National Laboratories*.

O particionamento de domínio é feito utilizando-se o método das bisseções recursivas, ou recursive coordinate bissection (RCB). Este é um método simples e intuitivo, em que o algoritmo escolhe uma direção de corte para a malha, fazendo um corte perpendicular apropriado. O domínio deve ser cortado de tal forma que a soma das cargas, ou volume de dados, em cada partição seja a mesma. O algoritmo é reutilizado recursivamente até qu o número de partições seja igual ao número de processos. Nas Figuras 2(a) e 2(b) mostram-se , respectivamente, o particionamento de domínio para uma malha com 4 níveis de refinamento em 8 processos e a distribuição da carga em cada partição. Na Figura 2(a), cada cor representa uma partição de domínio ou processo.



(a) (b)

Figura 2: Malha com 4 níveis, nível de base 64x64x64 e 8 processos. a) visualização das partições. b) Distribuição da carga entre os processos.

Um aspecto importante na utilização de malhas adaptativas bloco estruturadas é a comunicação

entre os blocos. Cada bloco possui uma camada extra de células ao seu redor, denominadas células fantasmas.As condições de contorno para as células fantasmas podem ser fornecidas de três diferentes formas, dependendo do tipo de célula fantasma que está sendo considerada. Na primeira forma, um procedimento de interpolação envolvendo valores da malha grossa e da malha fina são empregados para determinar os valores de células fantasmas que não pertencem a nenhuma outra malha do mesmo nível. Na segunda forma, células fantasmas pertencentes a uma malha irmã tem seu valores determinados através de importações dos valores já previamente determinados na malha irmã, esse processo é conhecido como injeção. Finalmente, a terceira forma consiste em substituir os valores das células fantasmas que tocam o domínio por valores reais das condições de contorno.

Na paralelização de um código envolvendo malhas adaptativas bloco estruturadas, o cálculo das células fantasmas é de importância fundamental para todo o funcionamento do código em paralelo. O cálculo em paralelo das células fantasmas é feio em 5 passos. No primeiro passo faz-se uma varredura em busca dos blocos que tocam as fronteiras dos processos. No segundo passo envia-se os dados geométricos dos blocos selecionados para os processos vizinhos via MPI_SEND. No terceiro passo faz-se uma varredura dos blocos selecionados no primeiro passo, à procura das células que tocam as fronteiras. No quarto passo envia-se as células selecionadas no terceiro passo para os processos vizinhos via MPI_SEND. Finalmente, aplicam-se os mesmos algoritmos implementados em serial, porém utilizando os dados recebidos na vizinhança.

Devido à importância das células fantasmas em paralelo, é necessário que os cálculos sejam devidamente validados. A validação é feita por meio da comparação entre uma função f_g, imposta nas células fantasmas de cada bloco e o valor calculado da célula fantasma, f. Observa-se que os desvios $|f_g - f|$ são no mínimo da ordem do dx^2 de cada nível, para $dx = dy = dz$. Os resultados apresentados na Tab. 1 representam os desvios absolutos obtidos no cálculo das células fantasmas para a malha da Fig. 2(a). Dentre os desvios de cada processo, são mostrados somente os valores máximos na Tabela 1. Observa-se ainda na Tab. 1, que não são mostrados desvios para o nível 1, pois este corresponde ao nível base, no qual não são necessárias interpolações. A função utilizada, neste trabalho, para a determinação dos desvios absolutos no cálculo em paralelo das células fantasmas é dada por:

$$f(x, y, z) = f_g(x, y, z) = x^2 + y^2 + z \tag{1}$$

Tabela 1: Desvios absolutos obtidos no cálculo em paralelo das células fantasmas.

Nível	$\|f_g - f\|$	dx^2
2	0.1220E-03	0.9765E-03
3	0.3814E-05	0.2441E-03
4	0.3476E-03	0.6103E-04

Para os testes de Speedup no cálculo das células fantasmas foi utilizada a malha da Fig. 2(a). O cálculo do Speedup ou aumento de velocidade tem o objetivo de determinar o ganho em velocidade de execução com o aumento do número de processos utilizados. Este é dado por:

$$S = \frac{T_s}{T_p} \tag{2}$$

sendo T_s o tempo em serial e T_p o tempo em paralelo.

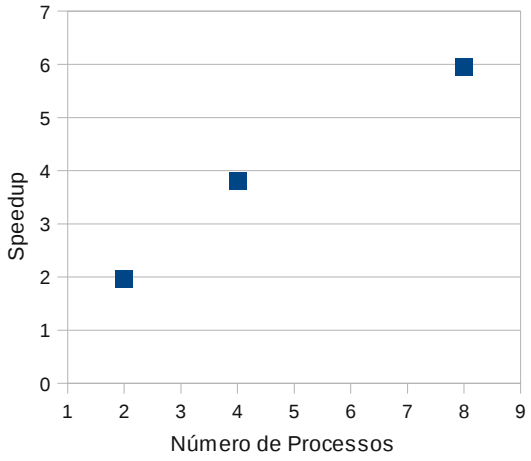


Figura 3: estes de Speedup para a configuração de malha da Fig. 2(a)

Como os testes foram feitos em um core2-quad, observa-se no resultados apresentado na Fig. 3, uma queda de desempenho após 8 processos, pois o número de processos ultrapassou o número de núcleos disponíveis no computador utilizado.

Referências

ALMGREN, A. *et al.* A conservative adaptive projection method for the variable density incompressible navier-stokes equations. *J. Comput. Phys*, v. 142, p. 1–46, 1998.

BERGER, M. J.; OLIGER, J. A conservative adaptive projection method for the variable density incompressible navier-stokes equations. *J. Comput. Phys*, v. 53, p. 484–512, 1984.

GRIFFITH, B. *Simulating the blood-muscle-valve mechanics of the heart by an adaptive and parallel version of the immersed boundary method*. Tese (Doutorado) — New York University, 2005.

ROMA, A. M.; PESKIN, C. S.; BERGER, M. J. An adaptive version of the immersed boundary method. *J. Comput. Phys*, v. 153, p. 509–534, 1984.

VILLAR, M. M. *Análise numérica detalhada de escoamentos multifásicos bidimensionais*. Tese (Doutorado) — Universidade Federal de Uberlândia, 2007.