Modelo de refinamento em paralelo para malhas de elementos finitos

Abner Franco Hermsdolf[1]
José Jeronimo Camata[1]

[1]Departamento de Ciências da Computação
Universidade Federal de Juiz de Fora (UFJF)

abner.franco@estudante.ufjf.br, jose.camata@ufjf.br

Abstract. *This paper presents a parallel mesh refinement model for finite element simulations on large-scale problems. The proposed methodology employs hash tables to map and synchronize newly created nodes at partition interfaces, ensuring global mesh consistency. The initial element distribution is performed with the METIS library, while interprocess communication is managed through MPI. The approach was tested on a mesh with more than five million tetrahedra, executed on 16 cores distributed across four computing nodes. Results demonstrate that the model preserves load balance among processes during refinement, confirming its suitability for high-performance computing applications.*

Resumo. *Este trabalho apresenta um modelo paralelo de refinamento de malhas de elementos finitos voltado para simulações de larga escala. A metodologia emprega tabelas hash para mapear e sincronizar os nós criados nas interfaces entre partições, garantindo a consistência global da malha. A distribuição inicial dos elementos é realizada com a biblioteca METIS, enquanto a comunicação entre processos é gerenciada via MPI. O método foi avaliado em uma malha composta por mais de cinco milhões de tetraedros, processada em 16 núcleos distribuídos em quatro nós de computação. Os resultados indicam que o modelo preserva o balanceamento de carga entre processos durante o refinamento, confirmando sua viabilidade para aplicações em ambientes de alto desempenho.*

1. Introdução

O refinamento de malhas de elementos finitos é uma técnica fundamental para aumentar a precisão de simulações numéricas, pois permite representar o domínio com maior resolução espacial [Bangerth and Rannacher 2003]. Na literatura, grande parte das abordagens está voltada para o refinamento adaptativo de malha (AMR), no qual regiões específicas do domínio são refinadas de acordo com estimadores de erro ou critérios físicos. Embora poderoso, o AMR envolve algoritmos complexos de decisão e pode demandar operações adicionais de balanceamento dinâmico de carga.

Neste trabalho, entretanto, o foco está no refinamento uniforme de malha (UMR), em que todos os elementos são subdivididos de forma homogênea a cada etapa de refinamento. Essa escolha simplifica a implementação e é particularmente útil em estudos exploratórios, benchmarks e simulações de larga escala. Abordagens semelhantes foram discutidas por Houzeaux et al. [Houzeaux et al. 2013], que propuseram um algoritmo de

parallel uniform mesh multiplication aplicado a um solver de Navier–Stokes, demonstrando boa escalabilidade até 16.384 processadores.

Apesar da simplicidade conceitual, o refinamento uniforme em ambiente paralelo apresenta desafios significativos relacionados à consistência da malha distribuída, mapeamento de nós nas interfaces entre partições e renumeração global eficiente. Para enfrentar esses desafios, este trabalho propõe um modelo paralelo de refinamento uniforme baseado em tabelas hash para mapear os nós de interface. O método assegura a integridade global da malha particionada e mantém o balanceamento de carga entre processos. A distribuição inicial é realizada pela biblioteca METIS[1], enquanto a comunicação entre processos é gerenciada por MPI[2].

2. Metodologia

O método proposto realiza o refinamento uniforme da malha de elementos finitos em ambiente paralelo. A cada etapa, todos os elementos são subdivididos de forma homogênea, aumentando a resolução espacial do domínio. O processo é estruturado em três componentes principais: refinamento geométrico dos elementos, sincronização de nós nas interfaces e renumeração global.

2.1. Refinamento dos elementos

O refinamento é realizado pela subdivisão das arestas de cada elemento, criando novos nós intermediários. A partir desses nós, são gerados elementos filhos que preservam as propriedades geométricas e topológicas do elemento pai (Figura 1). O procedimento segue o esquema descrito em Bey [Bey 2000], garantindo que a malha refinada permaneça conforme. Por se tratar de um refinamento uniforme, todos os elementos do domínio passam pelo mesmo processo de subdivisão.

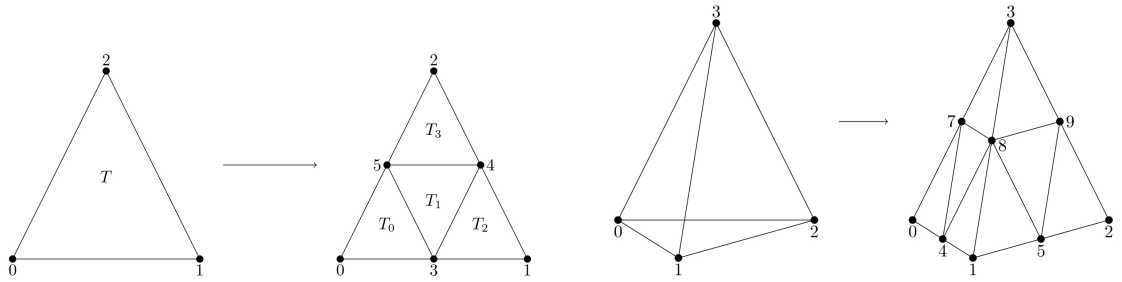

Figura 1. Modelos de Refinamento

2.2. Sincronização via função hash

Um dos desafios do paralelismo é a criação de nós compartilhados entre partições. Para resolver esse problema, cada novo nó criado em uma aresta/face é identificado por uma chave única, calculada a partir dos numeração global dos nós que definem a aresta/face. Essa chave é obtida com uma função de *hash* eficiente [Jenkins 2006]. Assim,

[1]https://github.com/KarypisLab/METIS
[2]https://www.open-mpi.org/

quando diferentes processos refinam a mesma aresta/face, o nó intermediário é identificado de forma consistente em todas as partições. Esse mecanismo evita a criação de duplicatas e garante a integridade da malha global. Além disso, define-se que o proprietário do nó em uma interface paralela será o processo de maior *rank*.

2.3. Renumeração global dos nós

Após a criação dos novos nós, cada processo numera localmente os seus nós internos. Em seguida, é realizado um procedimento de soma prefixada paralela (`MPI_Scan`) entre os processos, no qual cada partição acumula o total de nós internos dos processos de menor rank. Esse valor é usado para atualizar a numeração local, assegurando que todos os nós da malha refinada possuam identificadores globais únicos. Por fim, os processos que compartilham nós de interface comunicam os índices globais corretos, completando a etapa de sincronização. Esse mecanismo mantém a consistência da conectividade entre elementos e preserva o balanceamento de carga inicial.

3. Resultados

As simulações foram realizadas no cluster do Laboratório Integrado de Modelagem Computacional (LIMC), vinculado ao Programa de Pós-Graduação em Modelagem Computacional (PPGMC) da UFJF. Os quatro nós utilizados são do modelo Intel(R) Xeon(R) E5620, operando a um clock 2.40GHz, CPU(s) scaling entre 85% a 90% , com Frequency boost ativo ativado, permitindo que a frequência opere entre um mínimo de 1600 MHz e um máximo de 2401MHz. Cada nó dispõe de 16 Gb de memória RAM.

Para a avaliação do modelo de refinamento paralelo, empregou-se uma malha inicial de 660.992 tetraedros, distribuída em 16 núcleos de processamento, organizados em 4 nós (4 núcleos por nó). Após a aplicação de um nível de refinamento uniforme, cada tetraedro foi subdividido em 8 elementos filhos, resultando em uma malha com 5.287.936 tetraedros.

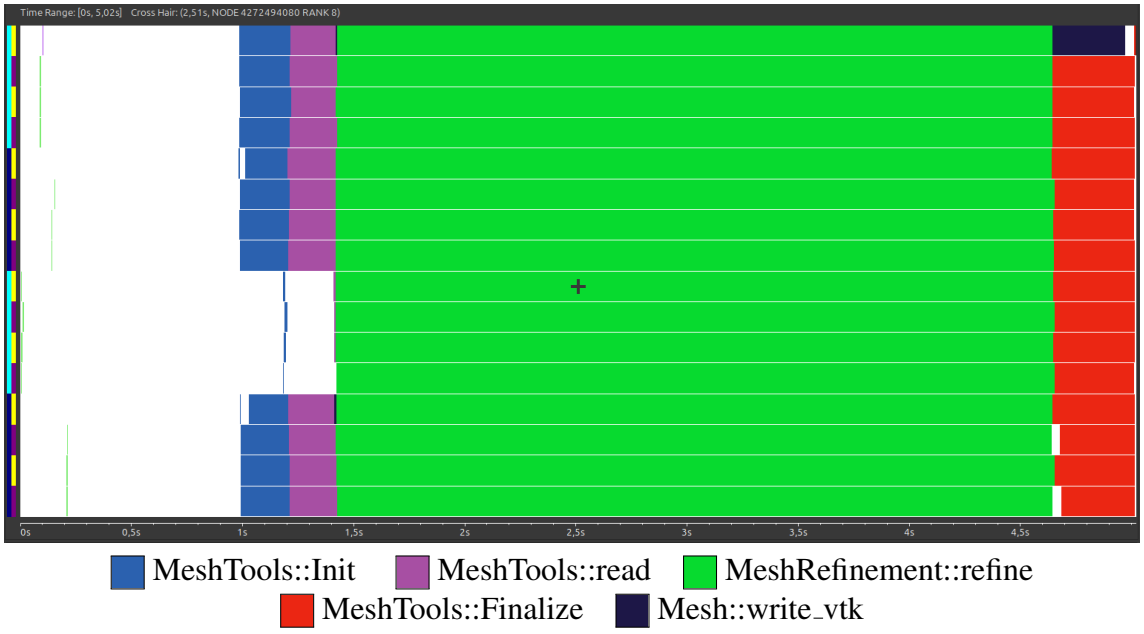

Figura 2. Análise Traceview gerada pelo hpcviewer

A análise de desempenho foi conduzida com a ferramenta HPCToolkit [Adhianto et al. 2010], tendo como foco a verificação do balanceamento de carga entre os processos. A Figura 2 apresenta o trace das funções executadas por processo após uma etapa de refinamento. Nota-se que a carga de trabalho da rotina de refinamento permanece bem distribuída, com variação mínima entre os processos mais e menos sobrecarregados. Esse resultado confirma a eficácia do método proposto, demonstrando que a criação dos novos elementos não compromete o balanceamento de carga inicial da malha paralela.

4. Conclusão

Neste trabalho, foi apresentado um modelo para o refinamento paralelo de malhas de elementos finitos de grande escala. A metodologia proposta utiliza a comunicação entre processos, gerenciada por tabelas hash, para sincronizar os nós de interface e garantir a consistência global da malha distribuída. Conforme analisado, o modelo manteve um excelente balanceamento de carga entre os processos durante o refinamento da malha, confirmando a viabilidade da abordagem para aplicações de alto desempenho, onde a escalabilidade é um fator crucial. Como perspectivas futuras, pretende-se avaliar o desempenho do modelo em arquiteturas com número maior de núcleos, bem como integrá-lo a códigos de dinâmica de fluidos computacional (CFD) e de elementos finitos, ampliando sua aplicabilidade em simulações científicas e de engenharia.

Agradecimentos

Os autores gostariam de expressar seus agradecimentos à FAPEMIG (APQ-02513-22), FINEP (SOS Equipamentos 2021 AV02 0062/22) e UFJF pelo financiamento deste trabalho.

Referências

[Adhianto et al. 2010] Adhianto, L., Banerjee, S., Fagan, M., Krentel, M., Marin, G., Mellor-Crummey, J., and Tallent, N. R. (2010). Hpctoolkit: tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, 22(6):685–701.

[Bangerth and Rannacher 2003] Bangerth, W. and Rannacher, R. (2003). *Adaptive Finite Element Methods for Differential Equations*, volume 2106 of *Lecture Notes in Mathematics*. Birkhäuser.

[Bey 2000] Bey, J. (2000). Simplicial grid refinement: on freudenthal's algorithm and the optimal number of congruence classes. *Numerische Mathematik*, 85:1–29.

[Houzeaux et al. 2013] Houzeaux, G., de la Cruz, R., Owen, H., and Vázquez, M. (2013). Parallel uniform mesh multiplication applied to a navier–stokes solver. *Computers Fluids*, 80:142–151. Selected contributions of the 23rd International Conference on Parallel Fluid Dynamics ParCFD2011.

[Jenkins 2006] Jenkins, B. (2006). lookup3.c: A Hash Function for Hash Table Lookup. https://www.burtleburtle.net/bob/c/lookup3.c. Código-fonte em domínio público. Acessado em: 23 de setembro de 2025.