Modelo de Refinamento em Paralelo para Malhas de Elementos Finitos

Abner Franco Hermsdolf José Jeronimo Camata

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

7 de novembro de 2025

Sumário

Contexto

Refinamento de Malhas

- Técnica fundamental para aumentar a precisão de simulações numéricas
- Permite representar o domínio com maior resolução espacial
- Essencial em simulações de elementos finitos de larga escala

Refinamento Adaptativo (AMR)

- Regiões específicas
- Baseado em critérios
- Mais complexo

Refinamento Uniforme (UMR)

- Toda a malha
- Subdivisão homogênea
- Foco deste trabalho

Motivação

Desafios do Refinamento Paralelo

- Consistência da malha distribuída
- Sincronização de nós nas interfaces entre partições
- Renumeração global eficiente
- Manutenção do balanceamento de carga

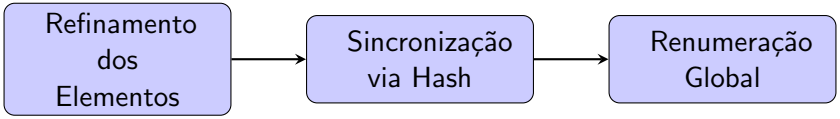
Proposta

Modelo paralelo de refinamento uniforme baseado em **tabelas hash** para garantir consistência global e balanceamento de carga

Objetivos

1. Desenvolver um modelo de refinamento uniforme em paralelo
2. Garantir a consistência global da malha distribuída
3. Sincronizar nós criados nas interfaces entre partições
4. Preservar o balanceamento de carga durante o refinamento
5. Validar em malhas de grande escala (milhões de elementos)

Metodologia - Visão Geral



Ferramentas Utilizadas

- **METIS**: Distribuição inicial dos elementos
- **MPI**: Comunicação entre processos
- **Função Hash**: Identificação única de nós

Refinamento dos Elementos

Processo

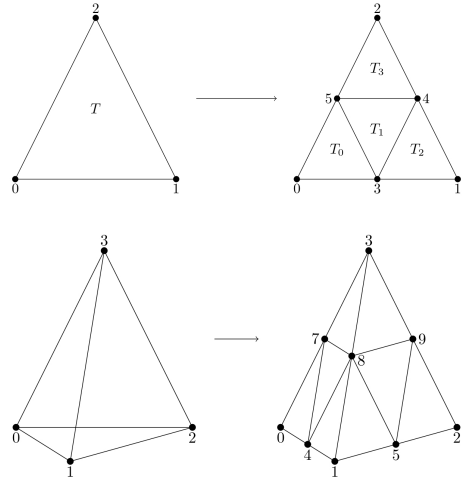
- Subdivisão das arestas
- Criação de nós intermediários
- Geração de elementos filhos
- Preservação das propriedades geométricas

Triângulo

1 elemento $\rightarrow$ 4 elementos

Tetraedro

1 elemento $\rightarrow$ 8 elementos



Sincronização via Função Hash

Desafio

Nós compartilhados entre partições devem ser criados de forma consistente

Solução

- Cada novo nó é identificado por uma **chave única**
- Chave calculada a partir da **numeração global** dos nós que definem a aresta/face
- Função hash eficiente (Jenkins, 2006)
- Processos diferentes geram a mesma chave para a mesma aresta

Regra de Propriedade

O proprietário do nó na interface é o processo de **maior rank**

Processadores que compartilha o Nó

Processo

1. Processo P_3 refina aresta e cria nó n

$$h_{key} = hash(A^{68}); \quad H_{sn}(h_{key}) \leftarrow n$$

2. Identificar processos que compartilham o nó:

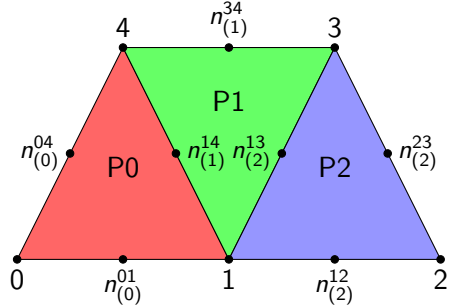
$$ppn(6) = \{2, 3\}; \quad ppn(8) = \{0, 1, 2, 3\}$$
$$ppn(n) = ppn(6) \cap ppn(8) = \{2, 3\}$$

3. Processo P_2 refina a mesma aresta: consulta a tabela hash
4. Evita duplicação de nós

Renumeração Global dos Nós

Procedimento

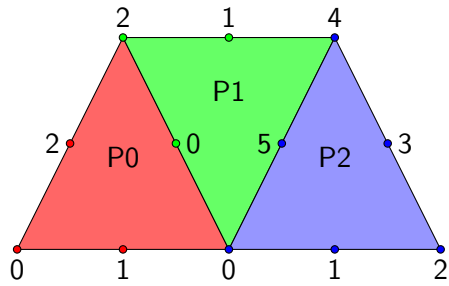
1. Cada processo numera localmente seus nós internos
2. Soma prefixada paralela (`MPI_Scan`)
   - Acumula total de nós de processos de menor rank
3. Atualização da numeração local com offset global
4. Comunicação de índices nas interfaces



Renumeração Global dos Nós

Procedimento

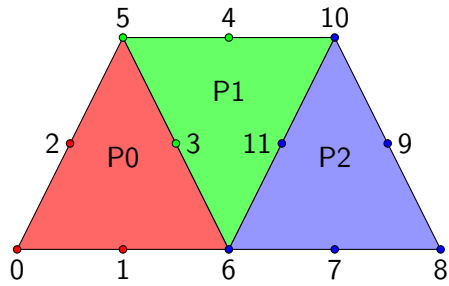
1. Cada processo numera localmente seus nós internos
2. Soma prefixada paralela (`MPI_Scan`)
   - Acumula total de nós de processos de menor rank
3. Atualização da numeração local com offset global
4. Comunicação de índices nas interfaces



Renumeração Global dos Nós

Procedimento

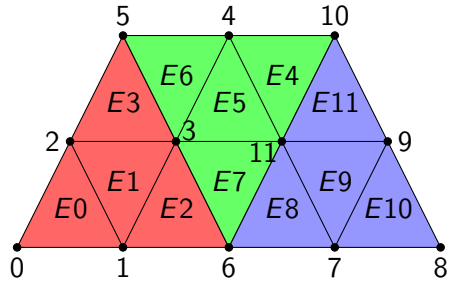
1. Cada processo numera localmente seus nós internos
2. Soma prefixada paralela (`MPI_Scan`)
   - Acumula total de nós de processos de menor rank
3. Atualização da numeração local com offset global
4. Comunicação de índices nas interfaces



Renumeração Global dos Nós

Procedimento

1. Cada processo numera localmente seus nós internos
2. Soma prefixada paralela (`MPI_Scan`)
   - Acumula total de nós de processos de menor rank
3. Atualização da numeração local com offset global
4. Comunicação de índices nas interfaces



Configuração Experimental

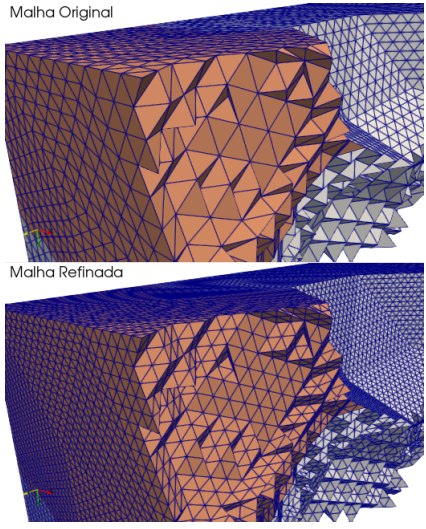
Infraestrutura

Cluster LIMC/PPGMC - UFJF

- 4 nós computacionais
- Processadores: Intel Xeon E5620 @ 2.40GHz
- 16 GB RAM por nó
- 4 núcleos por nó (16 núcleos totais)
- Frequency boost: 1600 MHz - 2401 MHz

Malha de Teste

- **Inicial**: 660.992 tetraedros
- **Após refinamento**: 5.287.936 tetraedros ($8 \times$ maior)
- Distribuída em 16 núcleos



Análise de Desempenho

Tabela: Tempo de Processamento e *Speeup*

Processos	CPU Time (ms)	*speedup*
2	69196,5	1,00
4	26638,5	2,60
8	11654,0	5.90
16	4816,3	14.3

Obs: O valores para um processo não foram considerados pois é uma versão não otimizada (uso desnecessário da hash introduzindo sobrecusto e piora a localidade de memória). Uma versão serial otimizada precisa ser implementada.

Análise de Desempenho

Figura: Traceview gerado pelo HPCToolkit. Bloco verde refere-se a rotina `refine()`

Resultado Principal

Balanceamento de carga na função de refinamento, com variação mínima entre processos.

Balanceamento de Carga

Observações

- Carga de trabalho bem distribuída entre os 16 processos
- Variação mínima entre processos mais e menos carregados
- Distribuição inicial (METIS) preservada após refinamento
- Sincronização via hash não compromete o balanceamento

Conclusões

Contribuições

- Modelo paralelo de refinamento uniforme de malhas FEM
- Uso de tabelas hash para sincronização de nós de interface
- Garantia de consistência global da malha distribuída
- Preservação do balanceamento de carga

Observações

- Testado em malha com 5+ milhões de elementos
- 16 núcleos em arquitetura distribuída.
- Balanceamento confirmado por HPCToolkit.

Trabalhos Futuros

1. Avaliar o desempenho em arquiteturas com **maior número de núcleos**
2. Integrar o modelo a códigos de:
   - Dinâmica de fluidos computacional (CFD)
   - Simulações de elementos finitos
3. Estender para **refinamento adaptativo**
4. Estudos de escalabilidade forte e fraca
5. Aplicações em simulações científicas e de engenharia

Agradecimentos

Agradecimentos à:

- FAPEMIG (APQ-02513-22)
- FINEP (SOS Equipamentos 2021 AV02 0062/22)
- UFJF

Perguntas?

abner.franco@estudante.ufjf.br