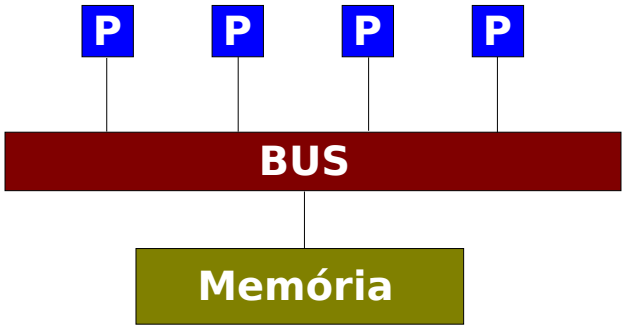
Introdução ao OpenFOAM

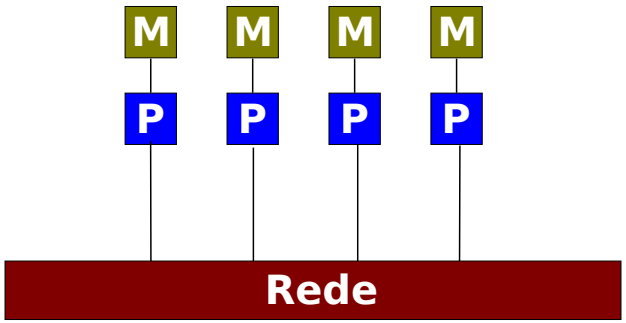
Aulas 05 e 06 - Paralelização, function objects e gmsh

Dr. Diogo Nardelli Siebert
Juan P. L. C. Salazar, Ph.D.
Fredi Cenci

17 a 27 de outubro de 2022

Tipos de sistemas paralelos





Memória compartilhada

As unidades de processamento compartilham uma mesma memória. Exemplo: CPUs multicore e GPUs.

Memória distribuida

Cada unidade de processamento tem sua memória, sendo necessário a troca de informação via rede. Exemplo: Cluster computacionais.

Paralelização no OpenFOAM

A parelização em OpenFOAM é realizada através da Message Passaging Interface (MPI), um padrão para a comunicação de computação paralela em sistemas de memória distribuída.

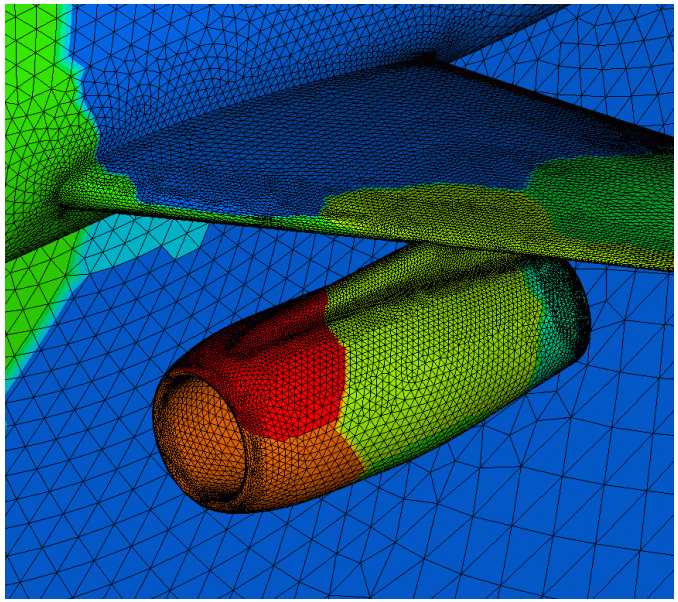
Importante

Um sistema de memória compartilhada pode funcionar como um sistema de memória distribuída, com a única diferença que a memória comum será segmentada e a cada parte é atribuída a uma unidade de processamento. Logo a troca de informação ocorrerá internamente e não via rede.

Paralelização no OpenFOAM

Código em MPI

- Cada unidade de processamento ficará encarregada de uma parte do domínio.
- Nos cálculos que envolvem dados de volumes que estão em outra unidade de processamento, o OpenFOAM envia tais dados utilizando a biblioteca MPI.



Decomposição

Etapas para solução de um caso em paralelo

- Geração da Malha (pode ser paralelizada);
- **decomposição do domínio/malha**;
- simulação;
- **reconstrução**;
- pós-processamento.

Decomposição

A decomposição é realizada pelo aplicativo **decomposePar**, configurado através do arquivo *decomposeParDict*, localizado na pasta *system*:

```
numberOfSubdomains NUMERO_DE_PROCESSOS;

method             NOME_DO_METODO;

DICIONARIO_DE_OPCOES
{
    OPCAO1                  VALOR_DA_OPCAO_1;
    OPCAO2                  VALOR_DA_OPCAO_2;
}

distributed     no;

roots           ( );
```

Métodos disponíveis

- **simple:** Divide a geometria em fatias nos diferentes planos.

```
simpleCoeffs
{
        n                   (2 2 1);
        delta               0.001;
}
```

- **hierarchical:** Similar ao simple, porém permite escolher a ordem em que a divisão será feita.

```
hierarchicalCoeffs
{
      n                   (1 1 1);
      delta               0.001;
      order               xyz;
}
```

Métodos disponíveis

- **scotch:** Utiliza a biblioteca scotch de particionamento de grafos, malhas e matrizes.

```
scotchCoeffs
{
        processorWeights        (1 1 1 1);    // opcional
        strategy                ESTRATEGIA;   // opcional
}
```

- **metis:** Utiliza a biblioteca metis para particionamento de grafos e malhas. Não disponível por padrão.

```
metisCoeffs
{
        processorWeights        (1 1 1 1);    // opcional
}
```

Métodos disponíveis

- **manual:** Especifica manualmente através de um arquivo quais são as células atribuidas a cada processador.

```
manualCoeffs
{
        dataFile        "filename.txt";
}
```

Observação

Existe ainda uma opção denominada **ptscotch**, que faz a própria decomposição em paralelo.

Execução do código em paralelo

Atenção!

Para utilizar o OpenFOAM em paralelo é necessário ter a biblioteca openMPI instalada no sistema.

Para executar o código em paralelo utilizamos o script mpiexec.

Comando

```
$ mpiexec -np 6 icoFoam -parallel
```

Reconstrução do caso

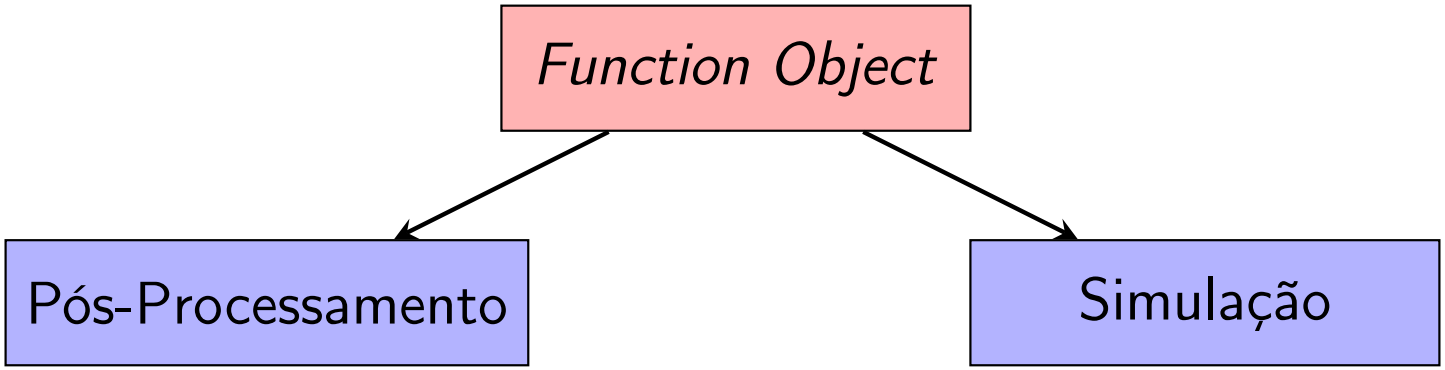
Após a simulação, os dados estarão divididos nos seus respectivos processadores. Para unir os arquivos de saída é necessário executar a reconstrução utilizando o comando

Comando

```
$ reconstrucParMesh
```

Pós-processamento com o OpenFOAM

O *OpenFOAM* possui uma ferramenta, denominada *function ojbects*, que permite a extração de dados e cálculo de parâmetros, tanto após quanto durante a simulação.

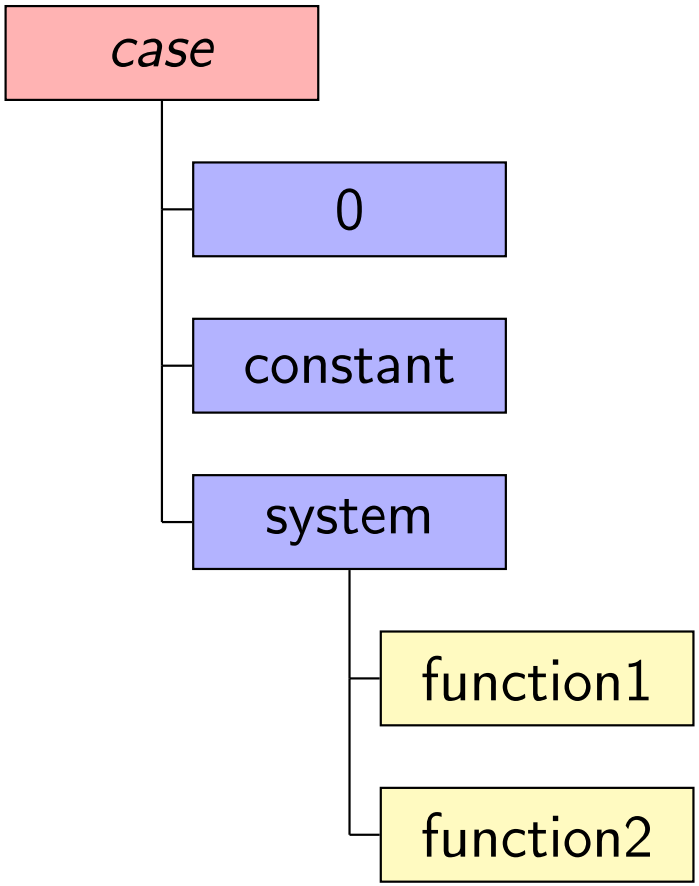


Estrutura de uma function object

```
<nome definido pelo usuario>
{
    type          <tipo do objeto>;
    libs          (<lista de arquivos que contem os objetos>);
    option1       <parametro da configuracao 1>;
    option2       <parametro da configuracao 2>;
    ...
}
```

- OpenFOAM.com guide for functon objects

Adicionando as function objects



Recomenda-se adicionar as function objects como arquivos separados dentro da pasta *system* para uma melhor organização.

Como utilizar durante a simulação

Para que ela seja executada ao longo da simulação deve-se inserir a mesma dentro de um dicionário *functions* no arquivo *controlDict*:

controlDict

```
functions
{
    #includeFunc function1
    #includeFunc function2
}
```

Como utilizar no pós-processamento

Para que ela seja executada após a simulação, usando os campos salvos, utiliza-se a ferramenta *postProcess*

Comando

```
case$ postProcess -func function1
case$ postProcess -func function2
```

Observação

Se a simulação foi executada em paralelo, executar o postProcess em paralelo

```
case$ mpiexec -np 6 icoFoam -parallel
case$ mpiexec -np 6 postProcess -func function1 -parallel
```

Utilizando o gmsh

O que é o gmsh?

Gmsh é

- um gerador de malhas de elementos finitos 3D;
- um programa CAD;
- um pós-processador.

Gmsh é um software livre (GPL2).

Missão

É projetado com o objetivo de prover uma ferramenta de geração de malhas rápida, leve e amigável, com entrada paramétrica e capacidade de visualização avançada.