

Programação Multicore

OpenMP

Demetrios A. M. Coutinho - NADIC/IFRN

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

12-13 de maio de 2023



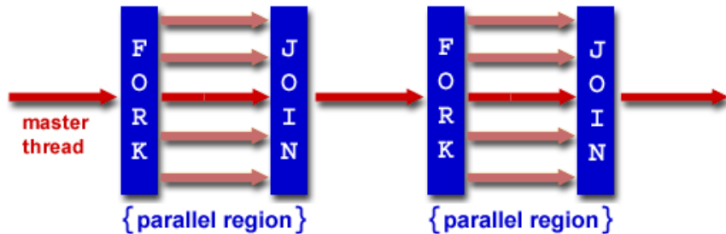
Next

- 1 Região Paralela
- 2 Divisão de Tarefas
- 3 Escopo de variáveis
- 4 Sincronização
- 5 Resumo

OpenMP

- ▶ OpenMP (*Open Multi-Processing*) é uma API multiplataforma para multiprocessamento, usando memória compartilhada.
- ▶ Composta por um conjunto de diretivas para o compilador, funções de biblioteca e variáveis de ambiente as quais especificam a implementação de um programa paralelo em C/C++.

Região Paralela



- ▶ O programa começa com uma única thread: **master thread**.
- 🔗 fork: o **master thread** inicia um conjunto de threads.
- 🔗 join: somente o **master thread** continua a executar.

Região Paralela

```
/* sequential, master thread ... */  
#pragma omp parallel [clause...]  
{  
/* parallel, all threads ... */  
}  
/* sequential, master thread ... */
```

- ▶ A estrutura é sempre **#pragma omp diretiva clausulas**.
- ▶ A diretiva **Parallel** cria novas threads.
- ▶ A quantidade de threads é igual ao número de núcleos, quando não especificado a quantidade de threads.

Hello World

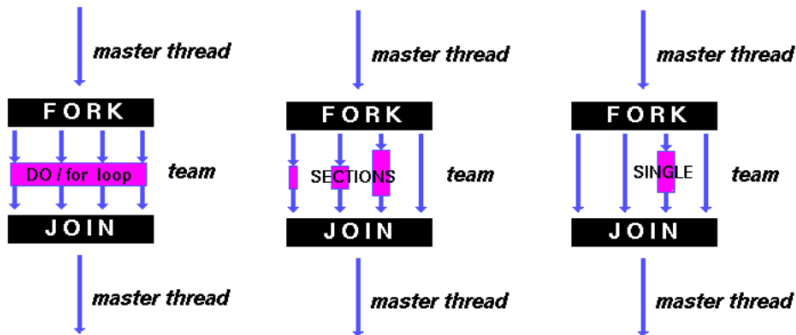
```
//gcc -fopenmp hello.c -o hello
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
    omp_set_num_threads(4);

    #pragma omp parallel
    {
        int id = omp_get_thread_num();
        printf("Hello, world! This is thread %d out of
↪ %d.\n", id, num_threads);
    }
```

Next

- 1 Região Paralela
- 2 Divisão de Tarefas
- 3 Escopo de variáveis
- 4 Sincronização
- 5 Resumo

Divisão de Tarefas



Divisão de Tarefas com OpenMP For

```
#pragma omp for [clause...]  
for (init;test;incr){  
    /* ... */  
}
```

- ▶ Deve estar dentro de um bloco paralelo.
- ▶ Restrições sobre a variável de laço e expressões de controle:
 - ▶ Variável do laço deve ser **int**, **iterator** ou **ponteiro**.
 - ▶ O teste deve ser um dos $<$, \leq , $>$, \geq .
 - ▶ O incremento pode ser $++$, $-$, $+=$, $-=$, $+$, $-$.
 - ▶ O teste e o incremento devem ser **invariáveis** dentro do laço.
 - ▶ Podem haver vários laços aninhados.
 - ▶ O programador é responsável por **dependências** dentro das iterações.

Exemplos

Vamos ver alguns exemplos de divisão de tarefas.

Next

- 1 Região Paralela
- 2 Divisão de Tarefas
- 3 Escopo de variáveis
- 4 Sincronização
- 5 Resumo

Escopo de Dados

- ▶ **Definição de Escopo de Dados:** controle do acesso dos threads a dados compartilhados.
- ▶ **Dados Privados:** variáveis que devem ser alocadas separadamente para cada thread, cada thread terá uma cópia privada da variável.
- ▶ **Dados Compartilhados:** variáveis que são acessadas por todos os threads em uma região paralela.
- ▶ **As cláusulas `private` e `shared`:** permitem controlar o escopo dos dados em uma região paralela.

```
int i, sum = 0;
#pragma omp parallel for private(i) shared(sum)
for (i = 0; i < N; i++) {
    sum += a[i];
}
```

Next

- 1 Região Paralela
- 2 Divisão de Tarefas
- 3 Escopo de variáveis
- 4 Sincronização
- 5 Resumo

OpenMP barrier

```
/* before */  
#pragma omp barrier  
/* after */
```

- ▶ Nenhuma thread cruza barreira até as outras threads atingirem a barreira.
- ▶ Algumas diretivas omp possui uma barreira implícita.
- ▶ A cláusula nowait pode remover a barreira implícita.

OpenMP critical

```
#pragma omp critical [name]
{
  /* ... */
}
```

- ▶ Declara uma região crítica.
- ▶ Todas as regiões críticas com o mesmo `name` são mutuamente exclusivos.
- ▶ Regiões críticas sem nome indicam a mesma região crítica.

Práticas

Mão na massa.

Next

- 1 Região Paralela
- 2 Divisão de Tarefas
- 3 Escopo de variáveis
- 4 Sincronização
- 5 Resumo

O que vimos

Introdução à programação paralela com OpenMP

- </> API multiplataforma para multiprocessamento.
- ⚙️ Uso de memória compartilhada.

Conceitos de regiões paralelas e divisão de tarefas

- ≡ Regiões paralelas e laços for paralelos.
- 🔗 Divisão de tarefas entre threads.

Escopo de variáveis: privado e compartilhado

- 🔒 Variáveis privadas para cada thread.
- 🔓 Variáveis compartilhadas entre threads.

Sincronização: barreira e seções críticas

- ⌚ Barreira para sincronizar threads.
- 🔒 Seções críticas para acesso exclusivo a recursos.

Agenda

Obrigado pela atenção 🙋

Próximos tópicos

Part 1 - ~~Contexto e Motivação~~

Part 2 - ~~Paralelismo de Hardware e Software~~

Part 3 - ~~Computação Paralela com OpenMP~~

Part 4 - **Computação Paralela em CUDA**