① 5ª Aula

Programação com OpenMP

Modelos de programação para memória compartilhada:

- Threading explícito: Posix threads
- Diretivas de ~~compilação~~: OpenMP
- Troca de mensagens: MPI
- Linguagem paralela

②

- Threading explícito
  O desenvolvedor cria explicitamente múltiplas threads dentro de um mesmo processo e divide também o trabalho a ser realizado.
- Diretivas de ~~compilação~~ (Threading implícito)
  O desenvolvedor utiliza diretivas de compilação, que são inseridas no código sequencial, para informar ao compilador as regiões que devem ser paralelizadas.

③ Open MP

→ Open MP significa "Open specifications for Multi Processing"

→ Open + MP = Padrão aberto + Máquinas multiprocessadas

→ É uma API para programação paralela e não uma linguagem

→ Definida para ser utilizada em programas C/C++ e Fortran

→ Permite criação de programas paralelos com compartilhamento de memória, através de threads.

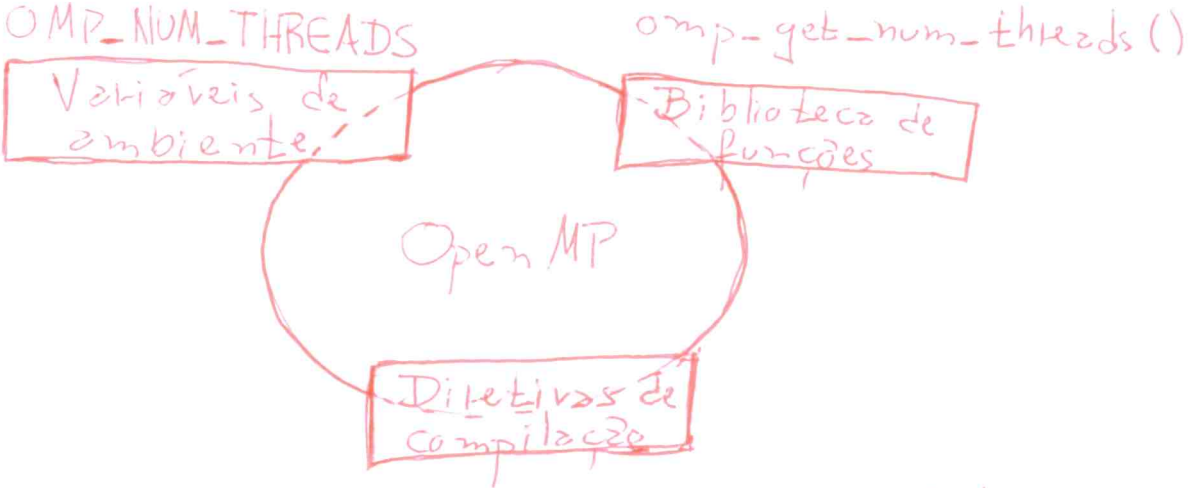
④

→ Estabelece um conjunto simples e limitado de diretivas

→ Permite paralelização incremental de programas sequenciais

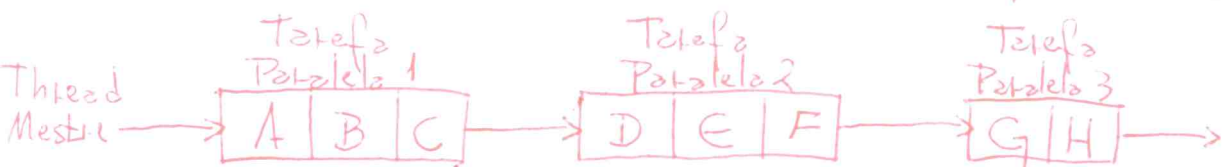
→ Produzir implementações eficientes em problemas de granularidade fina, média e grossa

5 Componentes do Open MP

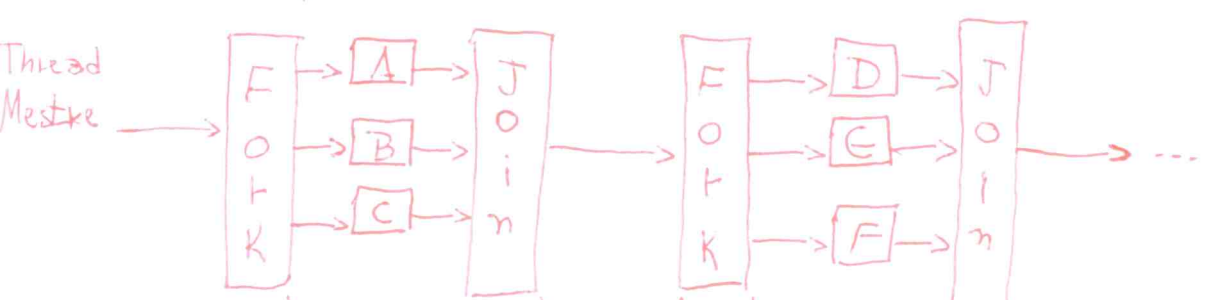


#pragma omp parallel

6 Modelo de execução do Open MP



Conjunto de threads



Sequencial Paralelo Sequencial Paralelo Sequencial

→ Modelo de execução Fork-Join

Estrutura de um programa Open MP

```
#include <omp.h> // incluir a biblioteca Open MP
...
int main(){
... // região sequêncial executada pela thread mestre
#pragma omp parallel // construtor paralelo do Open MP
{ // thread mestre cria o grupo de threads
 ... // região paralela executada por todas as threads
}// grupo de threads sincroniza com o mestre e termina
... // região sequêncial executada pela thread mestre
return 0;
}
```

Meu primeiro programa paralelo

```
#include <stdio.h>
#include <omp.h>
int main(){

#pragma omp parallel
{
  printf("Hello world da thread %d, número de
  threads: %d \n", omp_get_thread_num(),
  omp_get_num_threads());
}
return 0;
}
```

```
gcc -fopenmp teste.c -o teste
```

9

Diretivas e Cláusulas do OpenMP

→ Diretivas: São linhas de código com significado "especial" para o compilador.

→ Cláusulas: Definem comportamento de regiões paralelas e das variáveis aos quais estão associadas

```
#pragma omp directive [clause, ...]
```

opcional

10

Meu "segundo" programa paralelo

```
#include <stdio.h>
#include <omp.h>
int main(){
#pragma omp parallel num_threads(4)
 {
  ...
 }
 return 0;
}
```

parallel → diretiva

num_threads → cláusula

11

Blocos sintáticos

- Diretivas são aplicadas a blocos sintáticos
- Bloco sintático composto

```
#pragma omp directive [clause, ...]
{
  ... instruções
}
```

- Bloco sintático simples

```
#pragma omp directive [clause, ...]
  ... instrução
```

12

Diretivas de compilação em OpenMP

- Construtor paralelo
  - `#pragma omp parallel`
- Construtores de trabalho
  - `#pragma omp for`
  - `#pragma omp single`
  - `#pragma omp sections`
- Construtores de sincronização
  - `#pragma omp critical`
  - `#pragma omp atomic`
  - `#pragma omp barrier`
  - `#pragma omp flush`
  - `#pragma omp ordered`
  - `#pragma omp threadprivate`

13

Cláusulas em OpenMP

num_threads
shared
private
firstprivate
lastprivate
schedule
default
ordered
copyin
copyprivate
if
nowait
reduction

14

Compilação condicional

→ Para escrever código que funciona com e sem OpenMP

→ Protege chamadas de funções OpenMP com a diretiva #ifdef _OpenMP

Exemplo::

```
#ifdef _OPENMP // compilação condicional
    n = omp_get_num_threads(); // função OpenMP
#else
    n = 1;
#endif
```

15

Constutor paralelo fundamental

→ Diretiva #pragma omp parallel

→ O número de threads que irá executar a região paralela depende dos fatores:

(1) Apenas a thread mestre se existir uma cláusula do tipo if(exp) em que exp é falsa

(2) Número de threads definida pela cláusula num_threads(expr)

(3) Número de threads definido na última chamada a função omp_set_num_threads()

16

(4) Número de threads definido pela variável de ambiente OMP_NUM_THREADS

(5) O número de processadores disponíveis.

17

Cláusulas para o constructor parallel

if (expressão lógica)
private (lista de variáveis)
shared (lista de variáveis)
firstprivate (lista de variáveis)
default (shared | none)
copyin (lista de variáveis)
reduction (operador: lista de variáveis)
num_threads (variável inteira)

18

Cláusula if

if (expressão lógica)

→ Se a expressão lógica for verdadeira a região paralela será executada por mais de uma thread (região ativa)

→ Caso contrário, será executada apenas pela thread mestre

Exemplo:

```
#pragma omp parallel if(n > 100*var)
{
...
}
```

Meu "terceiro" programa paralelo

```
#include<stdio.h>
#include<omp.h>
int main(){
int nthreads, tid;
#pragma omp parallel private(nthreads, tid)
{
 tid = omp_get_thread_num();
 printf("Hello world da thread %d \n", tid);
 if (tid = 0){ //Apenas a thread mestre faz isso!
    nthreads = omp_get_num_threads();
    printf("Numero de threads = %d \n", nthreads)
} //if
}
return 0;
}
```