



Insper Supercomputação



Aula - 12

- OpenMP

OpenMP

Sections
Tasks



Sections

- A diretiva `sections` divide o trabalho de forma não iterativa em seções separadas, aonde cada seção será executada por uma “thread” do grupo. Representa a implementação de paralelismo funcional, ou seja, por código.
- Algumas observações:
 - A diretiva `sections` define a seção do código sequencial onde será definida as seções independentes, através da diretiva `section`;
 - Cada `section` é executada por uma *thread* do grupo;
 - Existe um ponto de sincronização implícita no final da diretiva `section`, a menos que se especifique o atributo `nowait`;
 - Se existirem mais *threads* do que seções, o OpenMP decidirá, quais *threads* executarão os blocos de `section`, e quais, não executarão.

```
#include <omp.h>
#define N 1000
int main () {
int i, n=N;
float a[N], b[N], c[N];
for (i=0; i < N; i++) a[i] = b[i] = i * 1.0,
#pragma omp parallel shared(a,b,c,n) private(i) {
#pragma omp sections nowait {
#pragma omp section
for (i=0; i < n/2; i++)
    c[i] = a[i] + b[i];
#pragma omp section
for (i=n/2; i < n; i++)
    c[i] = a[i] + b[i];
} /* fim seções*/
} /* fim parallel */
}
```

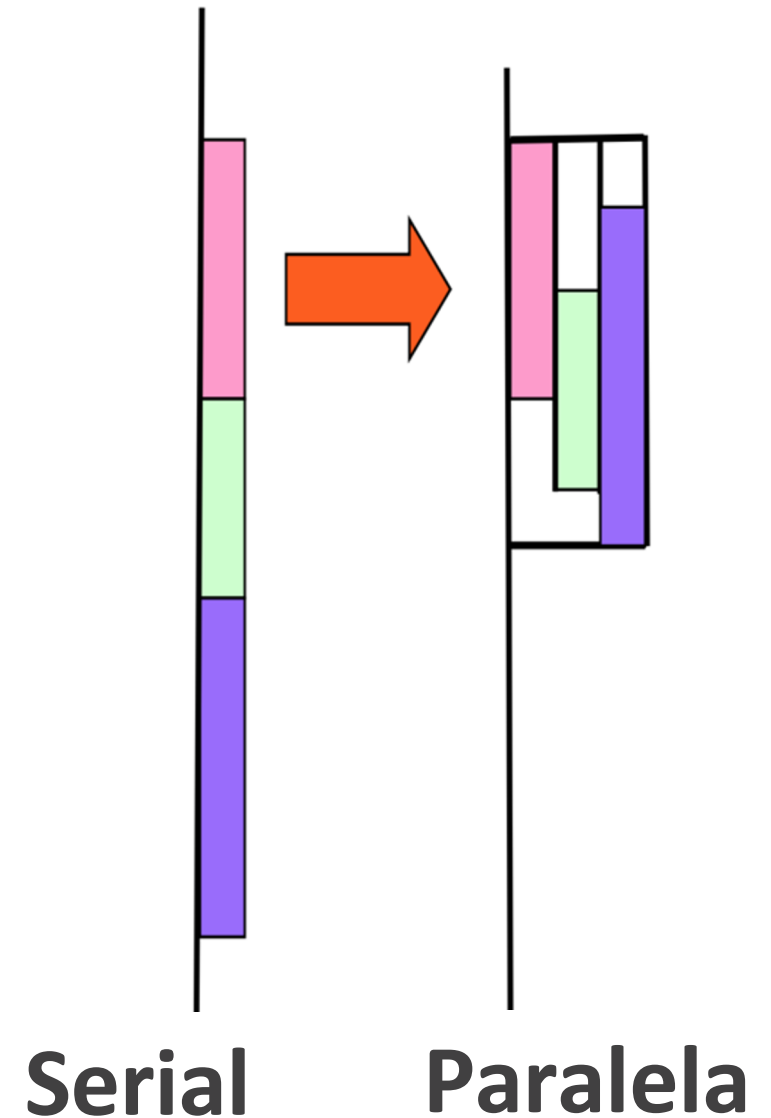
Definição de uma área de seções.

Primeira seção

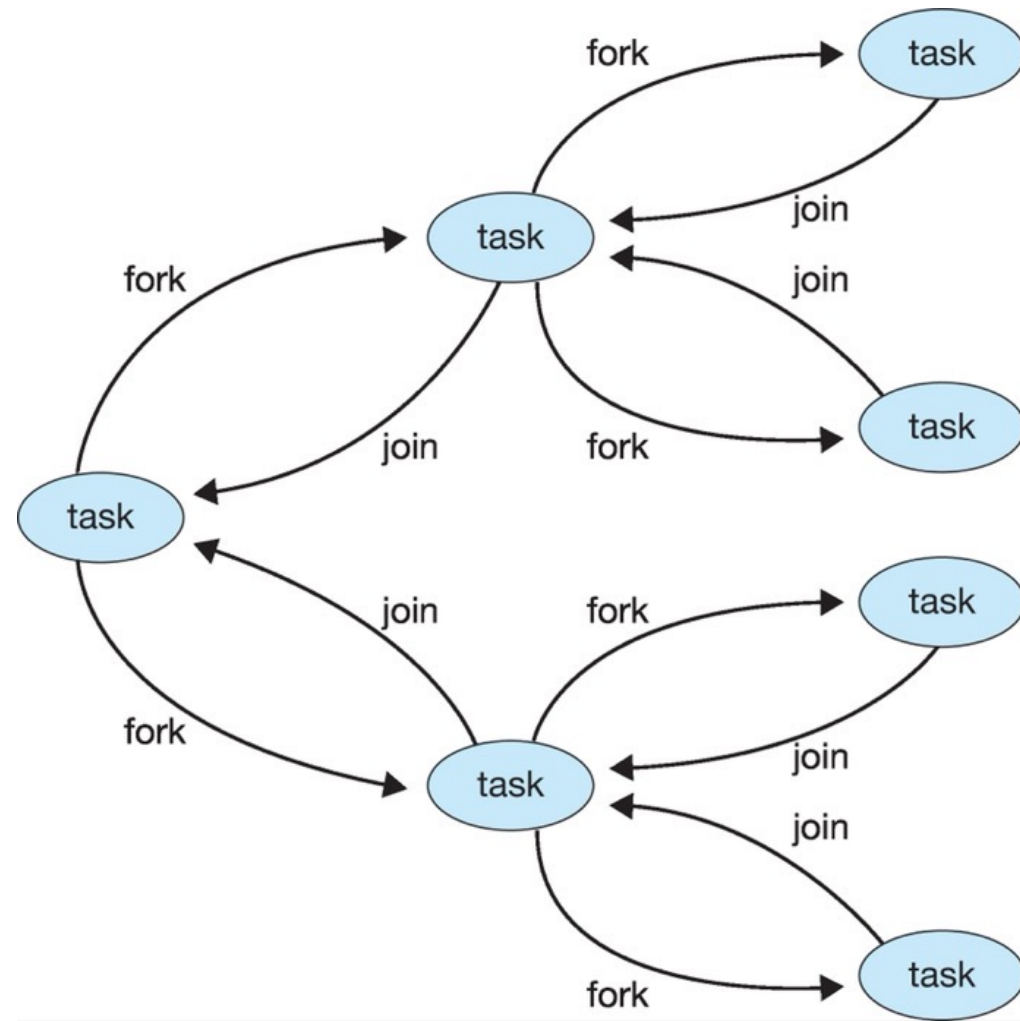
Segunda seção

O que são tarefas (tasks)?

- A tarefa é definida em um bloco estruturado de código
- Tarefas podem ser aninhadas: isto é, uma tarefa pode gerar outras novas tarefas
- Cada thread pode ser alocada para rodar uma tarefa
- Não existe ordenação no início das tarefas
- Tarefas são unidades de trabalho independentes



Tasks / Fork-Join



Tarefas em OpenMP

#pragma omp task[clauses]

```
#pragma omp parallel  
{
```

```
    #pragma omp master  
    {
```

```
        #pragma omp task  
        func1();
```

```
        #pragma omp task  
        func2();
```

```
        #pragma omp task  
        func3();
```

```
    }
```

```
}
```

Crie um conjunto de threads

Thread 0 organiza as tarefas

Tarefas executadas por alguma thread em alguma ordem

Todas as tarefas devem ser concluídas antes que esta barreira seja liberada


Estrutura Padrão

```
#include <stdio.h>
#include <omp.h>
int main()
{ printf("I think");
  #pragma omp parallel
  {
    #pragma omp single
    {
      #pragma omp task
      printf(" car");
      #pragma omp task
      printf(" race");
    }
  }
  printf("s");
  printf(" are fun!\n");
}
```

Esperando (taskwait)

```
#pragma omp parallel
{
    #pragma omp single
    {
        #pragma omp task
        fred();
        #pragma omp task
        daisy();
        #pragma taskwait
        #pragma omp task
        billy();
    }
}
```

fred() and **daisy()**
must complete before
billy() starts



Fibonacci

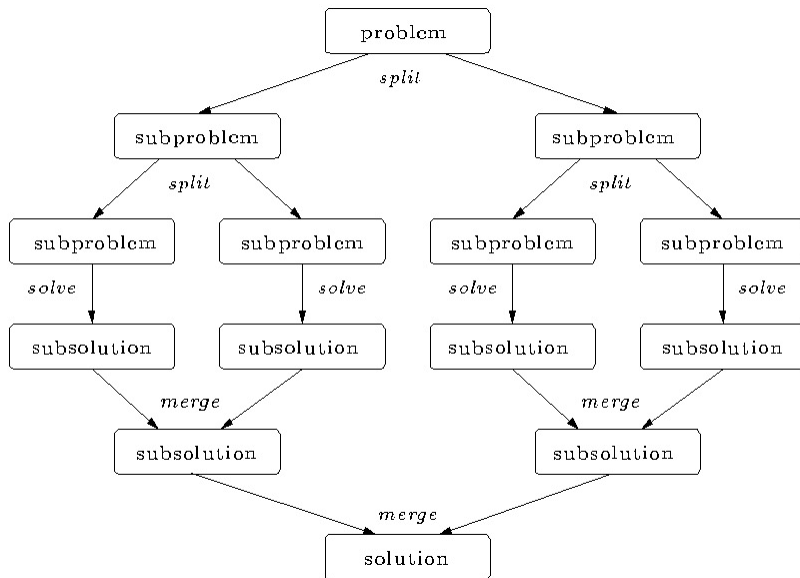
- É possível criar tarefas para esse problema?
- Altere a função fib para que ela chame uma task antes de calcular x e outra task antes de y.
- Lembre-se que quem chama fib (na função main) também precisa de uma task omp do tipo single (a task que dispara as outras tasks).
- Um ponto importante, se $n < 20$, calcule o Fibonacci sem o OpenMP.

```
#include<iostream>
#include<omp.h>
using namespace std;
```

```
int fib (int n) {
    int x, y;
    if(n<2) return n;
    x = fib(n-1);
    y = fib(n-2);
    return (x+y);
}
```

```
int main() {
    int NW = 1000;
    float time = omp_get_wtime();
    fib(NW);
    time = omp_get_wtime() - time;
    cout << "Tempo em segundos : " << time << endl;
}
```

Resolução



```
#include<iostream>
#include<omp.h>
#include <math.h>
using namespace std;

int fib (int n) {
    int x, y;
    if(n<2) return n;
    if (n < 20) {
        return fib(n-1) + fib(n-2);
    } else {
        #pragma omp task shared(x)
        x = fib(n-1);
        #pragma omp task shared(y)
        y = fib(n-2);
        #pragma omp taskwait
        return x+y;
    }
}

int main() {
    int NW = 50;
    float time;
    time = omp_get_wtime();
    #pragma omp parallel
    {
        #pragma omp single
        fib(NW);
    }
    time = omp_get_wtime() - time;
    cout << "Tempo em segundos : " << time << endl;
}
```



OpenMP - Roteiro

- Seguir o roteiro da aula

Mochila Binária - OpenMP



- Sua tarefa:
- Adaptar o código da mochila recursiva por busca exaustiva, para suportar OpenMP.
- Avalie o desempenho