

OpenMP

Alessandro Castelli

December 27, 2023

1 Cos'è

OpenMP è un'interfaccia di programmazione di applicazioni che supporta la programmazione multi processore. È nativamente supportata da: **C**, **C++**, **Fortan**. Per capire come funziona guarda la Figura 1, in cui è visibile come agisca attraverso un metodo **Fork-Join**.

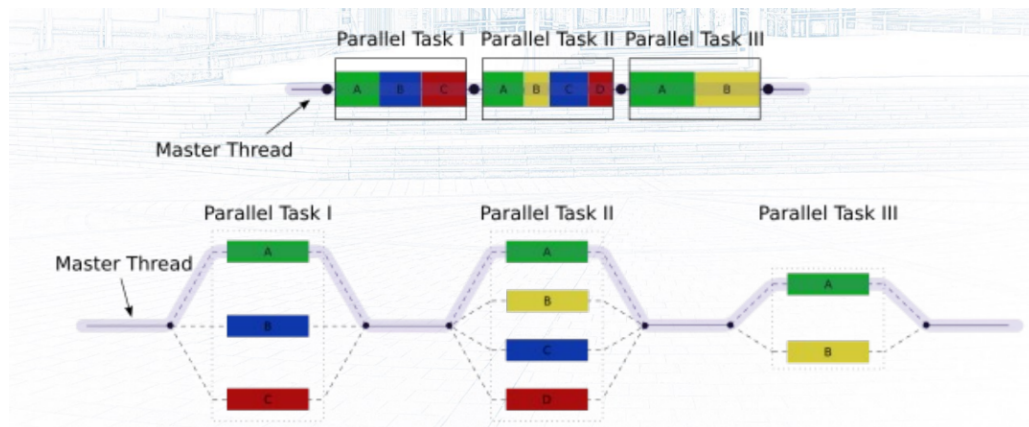


Figure 1: Fork-Join

2 OpenMP memory

- **Shared memory model:** thread comunicano tramite accesso a variabili condivise.
- **La condivisione è definita sintatticamente:**
 - Ogni variabile che è vista da 2 o più threads è **condivisa**
 - Ogni variabile che è vista da un solo thread è **privata**
- **Sono possibili le Race Conditions:** si usa la sincronizzazione per prevenire i conflitti e si cambia il modo di archiviazione dei dati per minimizzare la sincronizzazione

3 Sintassi

È la direttiva che si usa all'inizio della sezione.

```
1 #pragma omp <directive> [clause[,clause]...]
```

3.1 Direttive

- **parallel**: crea un team di thread che eseguono il blocco di codice incluso in parallelo.
- **for**: suddivide un ciclo in iterazioni più piccole che possono essere eseguite in parallelo da thread diversi.
- **sections**: suddivide il blocco di codice incluso in diverse sezioni che possono essere eseguite in parallelo.
- **single**: specifica che un blocco di codice deve essere eseguito da un solo thread.
- **critical**: specifica che un blocco di codice deve essere eseguito da un solo thread alla volta.
- **atomic**: specifica che è necessario accedere a una variabile in modo atomico.

4 Controlling Granularity

```
1 #pragma omp parallel if (expression)
```

Può essere usato per disabilitare la parallel. in alcuni casi.

```
1 #pragma omp num_threads (expression)
```

Controllare il numero di threads usati in questa regione parallel.

5 Data Environment

- **Shared Memory programming model:** la maggior parte delle variabili sono condivise per impostazione predefinita.

```
1 {  
2     int sum = 0;  
3     #pragma omp parallel for  
4     for(int i = 0; i < N; i++) sum += i;  
5 }
```

- Alcune variabili possono essere private

- **private:**
 - A copy of the variable is created for each thread
 - There is no connection between the original variable and the private copies
 - Can achieve the same using variables inside { }
- **firstprivate:**
 - Same, but the initial value of the variable is copied from the main copy
- **lastprivate:**
 - Same, but the last value of the variable is copied to the main copy

```
int i;  
#pragma omp parallel for private(i)  
for (i=0; i<n; i++) { ... }
```

```
int idx=1;  
int x = 10;  
#pragma omp parallel for \  
firstprivate(x) lastprivate(idx)  
for (i=0; i<n; i++) {  
    if (data[i]==x) idx = i;  
}
```

Figure 2: Overriding storage attributes

6 Synchronization Mechanism

6.1 Single/Master execution