OpenMP

lunedì 23 settembre 2019 11:53

PARALLEL

Il pragma parallel si usa per lavorare in parallelo.

```
#pragma omp parallel num_threads(nw)
```

PARALLEL FOR

Le iterazioni devono essere del tutto indipendenti tra loro, tutti i thread eseguono lo stesso codice

```
#pragma omp parallel for nthreads(n)
```

Posso anche indicare come vengono schedulati i dati, possiamo scegliere tra una divisione precisa del vettore in blocchi, una divisione dinamica in cui ogni volta chiedo ai worker parte del lavoro d'aiuto (dipende dal compilatore)

```
#pragma omp for schedule(argument)
```

SINGLE

Pragma che serve per eseguire quello che troviamo nelle parentesi in un singolo thread

```
#pragma omp single
```

ATOMIC

Pragma che serve per usare in modo atomico le variabili tra le parentesi:

```
#pragma omp atomic
```

SECTIONS

Pragma che viene usato per eseguire in due thread due codici differenti:

```
#pragma omp sections {
    #pragma omp section {
    } → codice 1
    #pragma omp section {
    } → codice 2
}
```

BARRIER

Pragma che serve per sincronizzare due thread che sono dipendenti tra loro.
Un thread arriva alla barrier e si ferma, quando anche l'altro arriva alla barrier allora parte anche l'altro che viene risvegliato.

```
#pragma omp barrier
```

TASK

Pragma usato per garantire che una parte di codice sia eseguito in modo del tutto indipendente dal resto.

C'è anche taskWait che mi esegue il codice in modo indipendente ma poi prima di fare altro devo attendere che tutti i thread dichiarati funzionino.

DIPENDENZE

Posso dire che un thread dipende da una certa variabile in input e in output, verrà eseguito in parallelo andando però a rispettare le dipendenze.

```
#pragma omp task depend(in: x) depend(out: y)
```