Esercitazione 08

25 Novembre 2024

Questa esercitazione è divisa in più parti, che consistono nel rendere paralleli diversi algoritmi visti in precedenza e di implementare una versione sia sequenziale che parallela della ricerca del vettore più simile ad un vettore dato.

Mergesort

Viene fornita un'implementazione di mergesort. Viene richiesto di implementare, a partire dalla precedente, la funzione

```
void merge_sort_parallel(int * v, int len)
```

che parallelizzi l'ordinamento. Notate che l'implementazione sequenziale è iterativa ma non tutti i cicli possono essere parallelizzati. In particolare si ricorda che prima di iniziare le operazioni di merge di blocchi dell'array di una data dimensione è necessario aver completato l'ordinamento dei blocchi più piccoli.

Moltiplicazione di matrici

Si richiede di parallelizzare i diversi modi di moltiplicare due matrici visti a lezione, definendo, a partire dalle versioni sequenziali fornite, le seguenti funzioni:

```
void omp_simple_multiply(float * A, float * B, float * C, int n)

void omp_transposed_multiply(float * A, float * B, float * C, int n)

void omp_blocked_multiply(float * A, float * B, float * C, int n)
```

Si ricorda che è possibile collassare parte dei cicli (ma non necessariamente tutti!).

Similarità tra vettori

Una procedura molto utile è quella di trovare in un insieme di vettori quale sia il più simile ad un altro vettore dato. Questo è usato, ad esempio, per trovare quali siano i documenti simili ad un documento dato, una volta che questi sono rappresentati come vettori. Il concetto di "similarità" utilizzato è quello di *cosine similarity*, ovvero si misura l'angolo che formano due vettori e si considera come misura di similarità il coseno di quell'angolo. In pratica indica quanto due vettori puntino in direzioni simili (sarà massimo se puntano nella stessa direzione e minimo se puntano in direzioni opposte).

La cosine similarity tra due vettori $x = (x_1, \ldots, x_n)$ e $y = (y_1, \ldots, y_n)$ si può calcolare come:

$$\text{cos\_sim}(x, y) = \frac{\sum_{i=1}^{n} x_i, y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}}$$

ovvero il prodotto scalare dei due vettori diviso il prodotto delle loro lunghezze.

Dati un vettore `v` di `float` ed una matrice `M` di `float`, in forma row major, in cui ogni riga rappresenta un diverso vettore (i.e., la matrice rappresenta l'insieme di vettori di cui trovare il più simile), si vogliono scrivere due funzioni:

```
int most_similar(float * v, float * M, int nrows, int ncols)

int omp_most_similar(float * v, float * M, int nrows, int ncols)
```

che ritornano l'indice della riga della matrice `M` corrispondente ad un vettore di similarità massima rispetto a `v`. La prima funzione dovrà eseguire in modo sequenziale, mentre la seconda dovrà fare uso di OpenMP per parallelizzare la ricerca del massimo.

Note

- Si ricorda che la funzione `sqrtf` in `math.h` implementa l'operazione di radice quadrata con argomento di tipo `float`.