Corso di Laurea in Informatica, A.A. 2023/2024

Architetture degli Elaboratori e Sistemi Operativi (AESO corso A e B)

Primo Appello – 12 Maggio 2025

(compito per studenti del vecchio ordinamento)

Si vuole realizzare un programma in C che, dato in input un vettore di interi di lunghezza N, lo ordini sfruttando $P \geq 1$ thread "Worker" (numerati da 0 a $P-1$) e una coda concorrente Q, che offre le operazioni *push* e *pop*. All'avvio, il thread 0 (Worker 0) calcola una serie di partizioni disgiunte dell'intero array (si assuma che il numero di partizioni è pari a *P*) e per ciascuna di esse inserisce nella coda Q la coppia di indici (start,end) che ne definisce l'intervallo. Dopo aver caricato tutti i task nella coda, tutti i Worker prelevano un task alla volta dalla coda concorrente eseguendo il sorting della partizione associata al task (per il sorting sequenziale di una partizione può essere usata la funzione di libreria qsort).
Ogni thread ripete questa operazione finché Q non è vuota, quindi tutti i Worker si sincronizzano mediante una barriera, garantendo che ogni partizione sia stata ordinata prima di passare alla fase di merge.
Superata la barriera, inizia la fase di merge, che viene implementata in $\log_2 P$ passi eseguiti in modo sincrono. Nel passo 1 soltanto i primi $\frac{P}{2}$ Worker effettuano in parallelo il merge delle coppie di blocchi adiacenti (<0;1>, <2;3>, …, <P-2; P-1>), mentre gli altri rimangono inattivi in una barriera. Al termine di ogni step tutti i thread si sincronizzano nella barriera prima di iniziare lo step successivo. Nel passo 2 solo i primi $\frac{P}{4}$ Worker fondono in parallelo le partizioni già unite al livello precedente; e così via, dimezzando il numero di thread attivi a ogni step, fino a quando il solo Worker 0 esegue l'ultimo merge, ottenendo un'unica partizione di lunghezza N completamente ordinata. Al termine dell'ultima barriera, l'intero vettore risulta ordinato ed i thread Worker terminano.

Lo studente dovrà fornire, entro il tempo assegnato ed utilizzando pseudo codice C, la definizione dei tipi di dato utilizzati ed una implementazione della funzione eseguita dai Worker. La sincronizzazione tra thread avviene tramite mutex e variabili di condizione. Si supponga che esista e sia già implementata la funzione *barrier* e si supponga anche che la coda Q sia già implementata e che siano disponibili i metodi *push* e *pop*.
Successivamente lo studente dovrà fornire una implementazione funzionante dell'algoritmo in C (includo il main, la funzione del generico thread Worker, l'implementazione della coda concorrente, l'implementazione della funzione barrier -- ad esempio usando pthread_barrier_wait--, …) unitamente al Makefile per compilare il programma ed eseguire i test (definiti dallo studente). Solo il codice sorgente di questo mini progetto dovrà essere spedito via email a massimo.torquati@unipi.it in un file ZIP dal nome sort_<NomeCognome>.zip entro e non oltre il **13 Marzo ore 23:59**.
Il programma dovrà gestire le seguenti opzioni:

- Numero di thread Worker (minimo 1).
- Dimensione dell'array contenente gli elementi da sommare (N)

Se il condice consegnato non compila utilizzando un compilatore gcc moderno oppure compila ma non esegue l'ordinamento in modo corretto al variare del numero di thread worker e del numero di partizioni, il codice non verrà valutato.