Prova Intermedia – 8 Novembre 2016

Traccia A

Domanda 1. Rispondete alle seguenti domande, motivando sempre la risposta.

A) Un programma contiene una funzione f perfettamente parallelizzabile. In una esecuzione su 4 processori, la metà del tempo viene spesa eseguendo codice inerentemente sequenziale, e l'altra metà eseguendo *in parallelo* il codice di f *su tutti e quattro i processori a disposizione.*

- A_1) Se l'esecuzione su 4 processori richiede 100 secondi, quanto durerebbe l'esecuzione su un unico processore?
- A_2) Qual è la percentuale di istruzioni eseguite in parallelo?
- A_3) Qual è lo speedup su 16 processori secondo la legge di Amdahl?
- A_4) Sarebbe possibile ottenere speedup 6? Spiegare il perché.

B) Per risolvere un certo problema, alcuni sviluppatori implementano un programma $\mathcal{P}$ ed una sua versione ottimizzata $\mathcal{P}'$. Eseguono poi alcuni test dei due programmi, osservando che:

- l'esecuzione di $\mathcal{P}$ e di $\mathcal{P}'$ su un unico processore richiede 2048 e 1024 secondi, rispettivamente;
- l'esecuzione di $\mathcal{P}$ su 32 processori richiede 65 secondi;
- $\mathcal{P}'$ ha span di 8 secondi.

Rispondete alle seguenti domande:

- B_1) Quanto tempo richiede l'esecuzione del cammino critico di $\mathcal{P}$?
- B_2) In base ai test eseguiti, $\mathcal{P}'$ è realmente una versione ottimizzata di $\mathcal{P}$? Ovvero, l'esecuzione di $\mathcal{P}'$ su 32 processori è piú veloce rispetto a quella di $\mathcal{P}$ sulla corrispondente piattaforma?

Quando viene comunicato agli sviluppatori che l'architettura target su cui dove essere risolto il problema in fase di produzione ha 512 processori, il lead software developer decide di abbandonare lo sviluppo di $\mathcal{P}'$.

- B_3) La scelta è corretta? Perché? Oltre a una spiegazione formale, date anche un'intuizione del risultato ottenuto.
- B_4) Per quale numero di processori i due programmi sarebbero ugualmente veloci?

Per entrambi i programmi potete assumere esecuzioni di caso peggiore utilizzando uno scheduler greedy.

Domanda 2. Data una matrice M di dimensioni $n \times n$, ne dovete calcolare la trasposta M^T, ovvero la matrice di dimensioni $n \times n$ tale che $M^T[i,j] = M[j,i]$. Ad esempio:

$$M = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \qquad M^T = \begin{pmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{pmatrix}$$

Un programmatore progetta il seguente algoritmo fork-join per risolvere il problema:

1. divide la matrice in quattro quadranti M_1, M_2, M_3, M_4, ciascuno di dimensione $\frac{n}{2} \times \frac{n}{2}$; ad esempio:

$$M_1 = \begin{pmatrix} a & b \\ e & f \end{pmatrix} \qquad M_2 = \begin{pmatrix} c & d \\ g & h \end{pmatrix} \qquad M_3 = \begin{pmatrix} i & j \\ m & n \end{pmatrix} \qquad M_4 = \begin{pmatrix} k & l \\ o & p \end{pmatrix}$$

2. calcola in parallelo la trasposta di ciascun quadrante, scrivendola in output nel quadrante opportuno della matrice M^T.

Il passo base si raggiunge quando la matrice da trasporre contiene un solo elemento.

a) Descrivete la struttura del codice fork-join dell'algoritmo (potete mostrare lo pseudocodice corrispondente alla funzione `compute`).

b) Spiegate in modo sintetico come ottimizzare il codice e quale impatto ha ciascuna ottimizzazione sul numero di thread.

c) Disegnate il DAG di esecuzione per $n = 4$ (matrice di 16 elementi), etichettando i nodi del DAG in modo da individuare il thread associato a ciascun nodo e la porzione di matrice su cui il thread opera. Assumete un sequential cutoff su matrici 2×2.

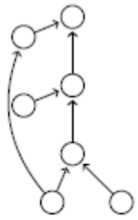
d) Analizzare work e span in funzione di n. Quali sono le equazioni di ricorrenza associate?

e) Se $n = 2^{20}$, quanti thread fork-join sono creati dal codice del punto a)? Applicando un cut-off sequential di 512, quanti thread fork-join sarebbero creati?

f) Supponete ora di voler memorizzare la matrice trasposta *in loco*, ovvero di non avere una matrice di output ma di voler modificare direttamente la matrice di input. Come modifichereste il codice in tale scenario? Descrivete le modifiche ad alto livello.

Domanda 3. Descrivete ed analizzate l'algoritmo parallelo per il partizionamento degli elementi di un array attorno ad un elemento scelto come perno.

Domanda 4 (bonus). Una computazione (non basata sul modello fork-join) dà origine al seguente DAG:



Ogni nodo rappresenta un task e gli archi indicano le dipendenze tra task. Ogni task richiede un'unità di tempo per essere eseguito su un singolo processore. Qual è la differenza tra il peggiore e il migliore tempo di esecuzione se i task sono schedulati in modo greedy su due processori? Mostrate due esempi di scheduling, uno per il caso migliore ed uno per il caso peggiore.