Prova d'esame – 28 Giugno 2017

Parte 1

Domanda 1. 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:

A_1) Quanto tempo richiede l'esecuzione del cammino critico di $\mathcal{P}$?

A_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 $\mathcal{P}'$.

A_3) La scelta è corretta? Perché? Oltre a una spiegazione formale, date anche un'intuizione del risultato ottenuto.

A_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. Sia T un albero binario completo con $n = 2^h$ foglie, tale che ogni nodo di T (nodi interni e foglie) ha associato un valore reale arbitrario v. Se x è una foglia, indichiamo con $A(x)$ l'insieme di tutti gli antenati di x, ovvero x stesso, il padre di x, il nonno, e via dicendo fino a risalire alla radice dell'albero. Sia $s(x)$ la somma dei valori v associati ai nodi in $A(x)$.

Progettate un algoritmo parallelo che trovi la foglia per cui il valore $s(x)$ è massimo. Analizzate work e span dell'algoritmo proposto in funzione di n.

Domanda 3. Descrivete l'implementazione parallela dell'algoritmo mergesort: analizzate le prestazioni della parallelizzazione naive (con merge sequenziale), presentate poi l'algoritmo per la fusione in parallelo di due array e discutetene l'integrazione nel mergesort. Analizzate work e span degli algoritmi considerati.

Prova d'esame – 28 Giugno 2017

Parte 2

Domanda 4. Considerate la seguente classe `TreeNode`:

```
public class TreeNode {
    TreeNode parent  = null;
    List children = new ArrayList();

    public synchronized void addChild(TreeNode child){
         if(!this.children.contains(child)) {
             this.children.add(child);
            child.setParentOnly(this);
        }
    }

    public synchronized void addChildOnly(TreeNode child){
        if(!this.children.contains(child){
            this.children.add(child);
        }
    }

    public synchronized void setParent(TreeNode parent)
        this.parent = parent;
        parent.addChildOnly(this);
    }

    public synchronized void setParentOnly(TreeNode parent){
        this.parent = parent;
    }
}
```

Rispondete alle seguenti domande, motivando sempre le risposte:

5a) Che meccanismo di locking viene impiegato nell'implementazione? Se ne discutano vantaggi e svantaggi.

5b) Quanti e quali lock sono coinvolti nell'esecuzione del metodo `setParent`? E nell'implementazione di `setParentOnly`?

5c) Il codice è soggetto a deadlock? Se si, mostrate un esempio, altrimenti dimostrate che è impossibile.

5d) Se il codice è soggetto a deadlock, come modifichereste l'implementazione per evitarlo?

5e) Supponete di usare il codice nel seguente scenario: siano dati 10 diversi nodi $v_0, \ldots v_9$. Ciascun nodo all'inizio è isolato, ma i nodi devono essere collegati a formare una catena in cui v_0 sia la radice e v_{i+1} sia figlio di v_i. Nove thread distinti si occupano di stabilire i collegamenti nella catena. In che ordine si crea l'albero? Si caratterizzino gli interleaving ammissibili.

Domanda 5. Spiegate in modo sintetico cosa è l'hand-over-hand locking e discutete se, utilizzando questo meccanismo di locking, il codice può essere soggetto a deadlock o a stavation.

Domanda 6. Supponete di avere un array A di n valori interi e un parametro intero $k \geq n/2$. Il problema richiede di invertire le coppie di elementi a distanza k (quindi $A[i]$ e $A[k+i]$, per ogni $i < n/2$ tale che $k+i < n$). Come risolvereste questo problema in OpenCL? Mostrate e discutete il codice kernel. Se k fosse minore di $n/2$, quale sarebbe il comportamento del codice presentato?