Rendering di cerchi

Silvia Dani e Niccolò Niccoli

Introduzione

- L’obiettivo è generare cerchi di diverso diametro e colore caratterizzati da trasparenza e coordinate x, y, z e renderizzarli.
- I cerchi vanno disegnati dal più in profondità al più in superficie
- La coordinata z viene rappresentata come l’ordinamento dei cerchi in un array

Progettazione

- Alla base c’è una struct Circle che contiene raggio, centro e colore
- Si crea un array per i cerchi e lo si popola
- Si divide l’array in sottogruppi
- Per ogni sottogruppo si disegnano in ordine i cerchi presenti in esso su una matrice trasparente
- Si uniscono le matrici risultanti dal punto precedente una alla volta partendo da quella contenente i cerchi più in profondità con una matrice bianca opaca

Parallelizzazione con OpenMP

Per parallelizzare il codice senza creare i threads necessari più volte si rendono shared le seguenti variabili

```
# pragma omp parallel default ( none ) shared ( images , circles , nCircles , numProcs , minNumCirclesPerImg , white
, imageHeight , imageWidth )
```

Per effettuare la parallelizzazione per disegnare i cerchi presenti nei sottogruppi dell'array si è utilizzato

```
# pragma omp parallel
# pragma omp for
```

Per unire due immagini analizzando pixel per pixel invece

```
# pragma omp for
for ( int threadIdx = 0; threadIdx < numProcs ; threadIdx ++)
      for ( int i = 0; i < numProcs ; i ++)
            overlayImage (& white , & images [ i ] , tileHeight * threadIdx , tileHeight * ( threadIdx + 1));
```

Esperimenti

La versione sequenziale e quella parallelizzata sono state messe a confronto ed è stato misurato il tempo necessario a renderizzare un'immagine al variare di:

- numero di cerchi (200, 1000, 10000, 100000)
- dimensione della matrice che si vuole riempire (256x256, 512x512, 1024x1024)
- numero dei processori coinvolti (2, 4, 8).

Risultati

Come si vede tutte le immagini sono identiche e quindi non vi è perdita di informazioni tra la versione sequenziale e quella parallelizzata.

Sequenziale

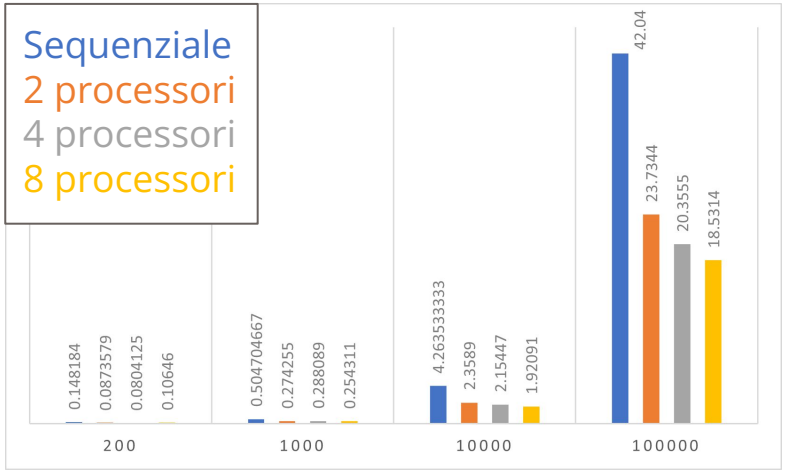
Parallelizzata, 2 processori

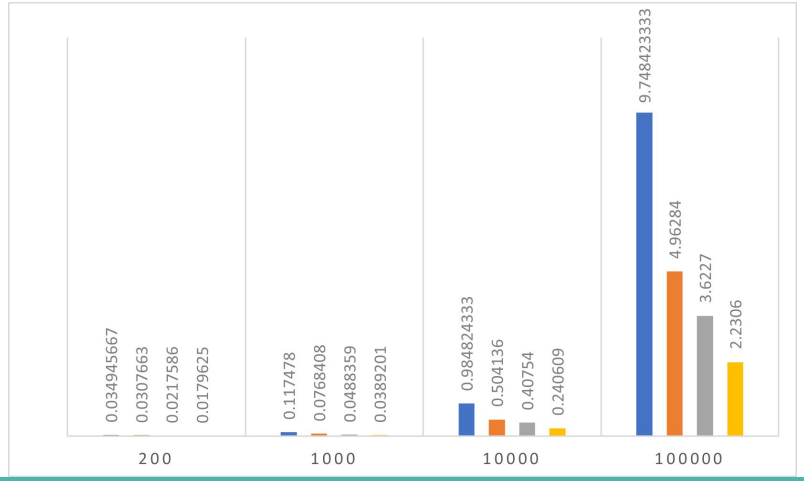
Parallelizzata, 4 processori

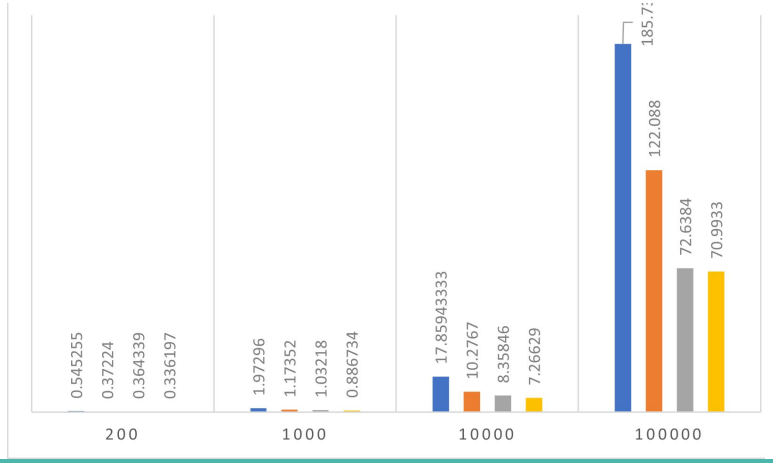
Parallelizzata, 8 processori

Risultati

Attraverso i grafici si può osservare che la versione parallela risulta conveniente quando il carico di lavoro diventa pesante.
Lo speed up massimo che si registra è 4.37.







Conclusioni

Dagli esperimenti svolti si può osservare che la parallelizzazione contribuisce a ridurre il tempo di esecuzione.

Lo speed up che si ottiene è massimo quando vengono utilizzati 8 core.