Utilizzo di OpenMP in problemi di render Parallel Programming for Machine Learning

Lorenzo Baiardi, Thomas Del Moro

GG MM AAAA

Indice

1 Introduzione

In questo elaborato verificheremo l'efficacia dell'utilizzo di metodi di parallelizzazione in problemi comuni, studiandone i risultati, la velocità e i tempi di esecuzione.

2 Analisi del problema

Il problema che abbiamo deciso di valutare è quello del compositing tra piani tramite l'utilizzo della libreria grafica OpenCV. Ogni piano avrà 4 canali di colore (RGBA), e per ogni piano verranno disegnati n cerchi. Durante la fase di sommatoria dei piani, viene applicato un effetto trasparenza in base alla posizione del piano all'interno della sommatoria. Una volta che vengono sommati tutti i piani, tramite operazione di compositing, verrà restituita una matrice risultante con l'immagine finale. I parametri utilizzati all'interno del progetto variano in base ai test effettuati.

3 Parallelizzazione

Qui riporteremo le operazioni di parallelizzazione che abbiamo eseguito all'interno del progetto.

3.1 OPENMP

Per la prima parallelizzazione abbiamo utilizzato la libreria OPENMP. L'idea di fondo è quella che ogni thread gestisca la sommatoria di un singolo pixel per ogni matrice, in modo da preservare l'ordinamento dei piani ma aumentandone la velocità di render. Di conseguenza ogni thread calcolerà il pixel risultante e successivamente, al termine di esso, passerà al successivo pixel. All'interno dell'esperimento varieremo il numero di thread che il calcolatore potrà utilizzare per la parallelizzazione per verificarne l'efficienza.

3.2 CUDA

Abbiamo poi analizzato il problema di parallelizzazione tramite l'utilizzo di schede grafiche NVDIA con il linguaggio di programmazione CUDA. In questo caso la parte di parallelizzazione si svilupperà principalmente nel calcolo della sommatoria dei principali canali delle matrici RGBA evitando fenomeni di burst della memoria, assegnando comunque un cuda-core a ogni pixel.

4 Tests

I test sono stati effettuati N volte variando il numero e le dimensioni dei piani, il numero di cerchi disegnati per piano e limitando il numero di thread da poter eseguire alla volta. I cerchi che dovranno essere disegnati sono generati precedentemente all'esecuzione del test, verificando così se i risultati delle varie operazioni riportino una soluzione identica.

4.1 Dati

5 Conclusioni

In conclusione, possiamo vedere come l'operazione di parallelizzazione CUDA risulti il più efficiente, a discapito però nel avere a disposizione una scheda grafica NVDIA. Nonostante ciò, anche la parallelizzazione tramite OPENMP risulta molto efficiente rispetto alla formulazione sequenziale.