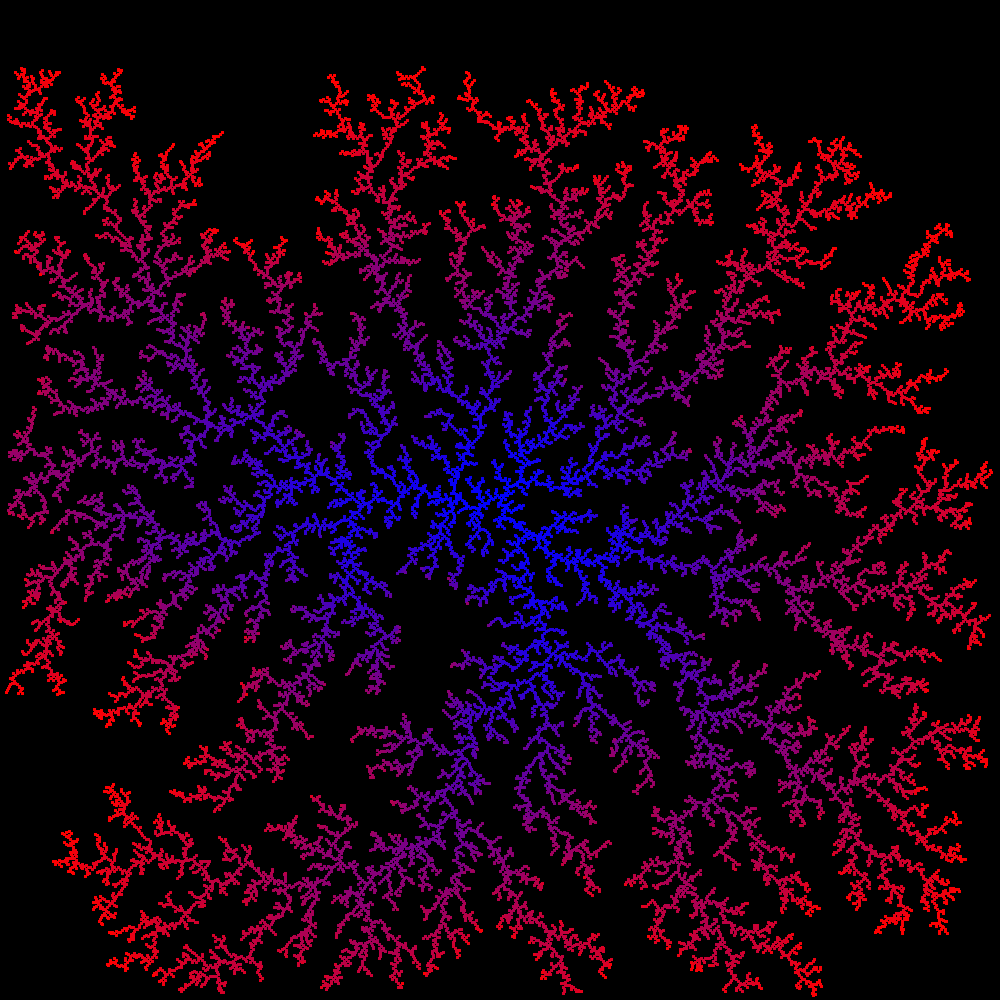
Relazione progetto DLA

Nicolo’ Madia 1960585



# Introduzione

In questa relazione, verrà fornita una descrizione dettagliata dell'implementazione di un'applicazione basata su CUDA (Compute Unified Device Architecture), in modalità single-threaded e su OpenMP (Open Multi-Processing) per la simulazione della crescita cristallina. Saranno esplorate le scelte progettuali effettuate, giustificate dalla natura del problema e dalle caratteristiche hardware coinvolte.

# Architettura dell'Applicazione

Implementazione in CUDA

L'implementazione in CUDA adotta un modello di programmazione parallela, sfruttando la potenza di elaborazione delle GPU. La simulazione coinvolge la crescita cristallina di particelle in un ambiente bidimensionale rappresentato da una griglia. Ogni particella è rappresentata come un thread CUDA, e il kernel CUDA si occupa del movimento e dell’ aggregazione delle particelle. La memoria globale della GPU è utilizzata per conservare la griglia e le informazioni sullo stato delle particelle.

Implementazione in OpenMP

L'implementazione in OpenMP adotta un modello di programmazione basato su thread, sfruttando i multi-core delle CPU. La parallelizzazione avviene distribuendo le particelle tra i thread disponibili. I costrutti OpenMP, come `parallel` e `for`, sono utilizzati per gestire la parallelizzazione e la distribuzione del lavoro tra i thread.

# Limitazioni Riscontrate

1. CUDA

- Accesso Globale: L'accesso globale può causare latenze di memoria, influenzando le prestazioni.

2. OpenMP

* Overhead: L'overhead associato alla gestione dei thread può diventare significativo su problemi computazionalmente leggeri.
* Il fatto che le particelle si muovano con moto browniano rende difficile la divisione dei thread sulla griglia creando problemi di false sharing.
* Le architetture NUMA moderne hanno la cosidetta “First Touch Policy”:  
  **Memory is mapped to the NUMA domain that first touches it**. Core 1 first touched this data element. It will be placed in the NUMA domain of core 1 Reads from this memory location issued from a core that is not in this NUMA domain are slower (higher latency). Per questo motivo, almeno sulla mia architettura, e per la natura poco ‘computazionale’ e piú ‘memory bound’ del problema, in quanto ogni particella deve controllare 8 celle attorno a se e, facendo quindi, almeno 3 accessi alla memoria, non mi é stato possibile – o non sono stato in grado - di ottenere uno speed up utilizzando piú di un core.

# Prestazioni e speed up

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Height | Width | duration | P count | Serial (s) | Parallel (s) | Speed-up | efficiency |
| 1000 | 1000 | 3000 | 50k | 2.080701 | 0.244668 | 8.49 | 0.85 |
| 1000 | 1000 | 3000 | 10k | 1.959175 | 0.275084 | 7.11 | 0.71 |
| 1000 | 1000 | 30000 | 10k | 18.100090 | 1.049694 | 17.26 | 0.57 |
| 1000 | 1000 | 30000 | 20k | 16.835194 | 1.145047 | 14.73 | 0.49 |

La tabella mostra i risultati delle prestazioni degli algoritmi con variazioni nel numero di particelle, iterazioni e dimensioni della griglia. I parametri "Speed-Up" ed "Efficiency" sono calcolati rispetto all'esecuzione seriale. Come si può notare, l'efficacia della parallelizzazione dipende dalla dimensione del problema, e in alcuni casi, aggiungere più risorse computazionali potrebbe non portare a un miglioramento lineare delle prestazioni.