Studio dello speedup del KMeans usando CUDA e OpenMP

Progetto Parallel Programming for Machine Learning

A.A. 2021/2022

Eduard Marchidan

1 KMeans

Il KMeans è un metodo di clustering non supervisionato che dato un numero di categorie K, trova i punti che più probabilmente appartengono a quella categoria.
L'algoritmo KMeans può essere riassunto in pochi passi:

- Input: N punti, K centroidi, E numero di epoche
- 1. Inizializza casualmente i centroidi
- 2. Assegna i punti al centroide più vicino
- 3. Calcola la media della posizione dei punti assegnati ad ogni centroide e aggiorna la posizione del centroide con la media calcolata
- 4. Ritorna al punto 2 fino al raggiungimento di una certa condizione di stop (in questo caso il numero di epoche E)

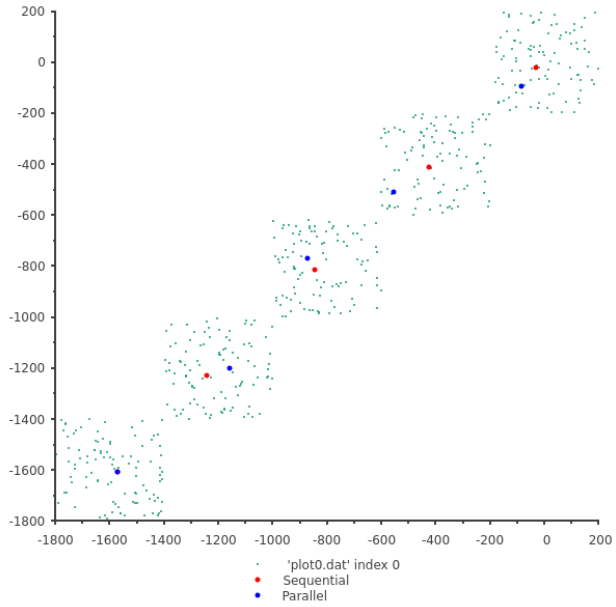
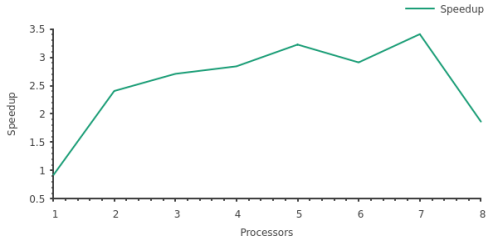
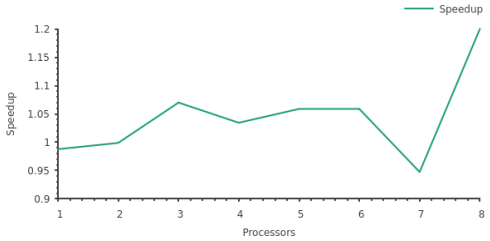


Figure 1: Esempio di esecuzione dell'algoritmo sul dataset sintetico



(a) Speedup all'aumentare del numero di cores usando AOS



(b) Speedup all'aumentare del numero di cores usando SOA

Figure 2

2 Implementazione e risultati

L'algoritmo è stato implementato sequenzialmente e parallelamente con OpenMP e CUDA. Inoltre è stato testato lo speedup utilizzando SOA invece che AOS per l'algoritmo sulla cpu (CUDA utilizza solo SOA). Il dataset di test è un insieme di 20 cluster generati da una distribuzione gaussiana. Viene testato il tempo di esecuzione a partire da 50k punti fino a 500k con incrementi di 50k, per poi fare una media. E' bene specificare che l'esecuzione dell'algoritmo anche nel peggiore dei casi con 500k punti richiede al massimo 20 secondi.

2.1 Speedup openmp vs. sequenziale

Com'è possibile vedere dalla figura 2 lo speedup in entrambi i casi non scala linearmente ma nel caso dell'AOS fino a 3 core è quasi-lineare per poi rimanere stabile, ciò sembra essere causato dalle parti sequenziali dell'algoritmo (in particolare lo step 3). Nel caso del SOA lo speedup è praticamente inesistente, la cpu lavora meglio su strutture soprattutto perchè può essere inserita l'intera struttura in cache permettendo di avere tutti i dati disponibili per il calcolo attuale immediatamente. Nel caso dell'AOS i vari array dei punti, dei centroidi, delle distanze minime etc... non possono stare in cache allo stesso tempo, richiedendo un costante recupero di nuovi dati dalla memoria, rallentando l'esecuzione.

2.2 Speedup openmp vs. cuda

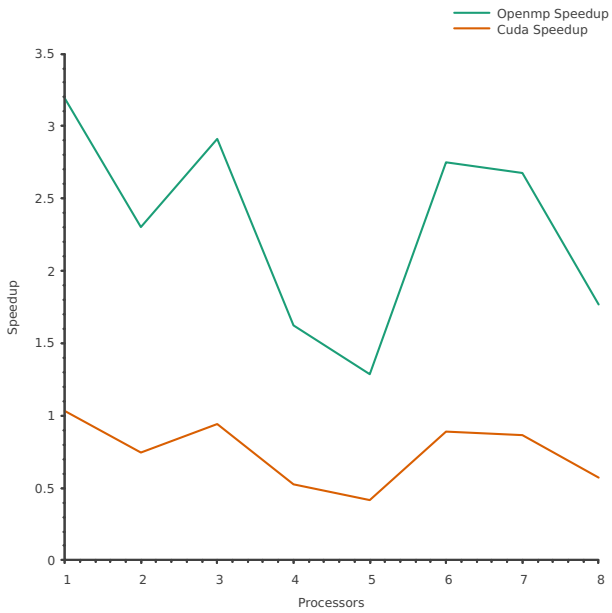


Figure 3: Speedup all'aumentare del numero di cores comparato openmp/sequenziale e openmp/cuda

Notiamo che nei picchi openmp è comparabile all'implementazione cuda, probabilmente causata dalla parte sequenziale che non può essere rimossa e lo spostamento dei dati tra cpu e gpu. Nonostante tutto cuda risulta essere fino a 2 volte più veloce.