Università di Bologna, Laurea in Ingegneria e Scienze Informatiche

Corso di High Performance Computing

Progetto d'esame 2025-2026

prof. Moreno Marzolla

Versione 1.0 del 3/12/2025

Prima versione di questo documento.

1. K-Means Clustering

K-Means Clustering [1, 2] è un algoritmo usato per suddividere insiemi di oggetti in gruppi (cluster) in base alla similarità. Gli algoritmi di clustering sono utili per ridurre la dimensione di un insieme di dati, rimpiazzandolo con un campione "rappresentativo": un esempio è ridurre il numero di colori in una immagine bitmap, selezionando una palette di dimensione limitata e rimpiazzando il colore di ogni pixel con quello più simile presente nella palette. Scopo di questo progetto è sviluppare due implementazioni parallele dell'algoritmo K-Means Clustering descritto di seguito, noto come *algoritmo di Lloyd* [3, 4]. Sebbene l'implementazione che consideriamo non sia la più efficiente, ha il vantaggio di essere abbastanza facile da parallelizzare.

L'input è costituito da un insieme $\mathbf{d} = \{\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots \boldsymbol{d}_{N-1}\}$ di N vettori di lunghezza D in cui ogni vettore rappresenta i valori dei D attributi di un oggetto, e un intero positivo K che indica il numero di gruppi (cluster) in cui suddividere i dati; si assume che K sia molto minore del numero N di oggetti ($K << N$). Al termine dell'esecuzione, l'algoritmo assegna ad ogni oggetto $\boldsymbol{d}_i$ un intero c_i in $\{0, \ldots K - 1\}$ che rappresenta il cluster a cui è stato assegnato.

Durante l'esecuzione, ogni cluster viene rappresentato da un vettore di lunghezza D detto *centroide* (i centroidi non devono necessariamente appartenere all'insieme **d**); quindi, i vettori di input e i centroidi si possono considerare come punti in uno spazio a D dimensioni. L'algoritmo opera iterativamente come segue:

1. Si definiscono K centroidi iniziali, ad esempio selezionando K punti casuali nell'insieme **d**.
2. Si assegna ogni punto in **d** al gruppo rappresentato dal centroide più vicino; a tale scopo si usa la distanza Euclidea tra i vettori che rappresentano le coordinate del punto e del centroide.
3. Si calcola un nuovo centroide per ogni gruppo; le nuove coordinate sono la media aritmetica di quelle dei punti assegnati a quel gruppo.
4. Si ripete dal punto 2 fino a quando le nuove posizioni dei centroidi (calcolate al punto 3) differiscono di poco da quelle precedenti, oppure fino a quando si raggiunge un numero massimo di iterazioni.

La Figura 1 mostra graficamente l'evoluzione dell'algoritmo su un insieme di punti in due dimensioni.

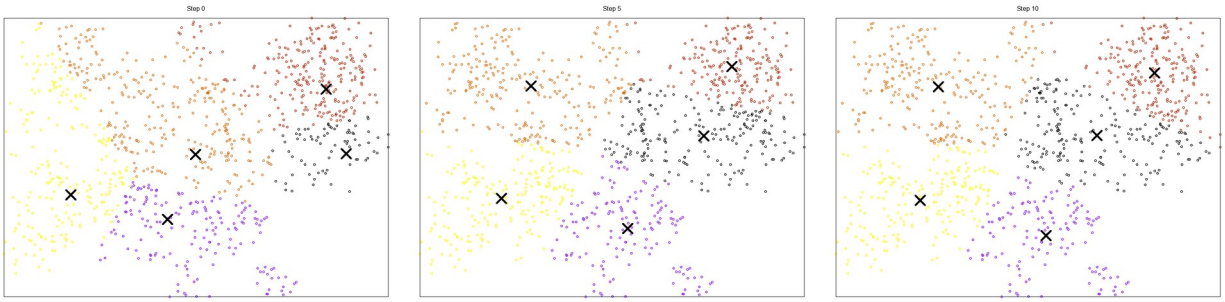

Figura 1: Esempio di esecuzione dell'algoritmo K-Means clustering su un input 2D. Le "x" rappresentano le posizioni dei centroidi, mentre i colori identificano il cluster a cui è associato ciascun punto.

Di solito l'algoritmo di Lloyd converge velocemente, anche se ci sono casi in cui il tempo di convergenza è esponenziale in N. Si noti che l'algoritmo non è deterministico perché i cluster dipendono dalla scelta dei centroidi iniziali. Il costo computazionale dell'algoritmo seriale è $O(N \times K \times D \times T)$ dove T è il numero di iterazioni necessarie per la convergenza.

2. Specifica del progetto

Scopo del progetto è la realizzazione di due implementazioni parallele dell'algoritmo di Lloyd. La prima deve essere realizzata usando OpenMP; la seconda usando a scelta uno tra MPI oppure CUDA. I nomi degli eseguibili devono essere `omp-k-means` (per la versione OpenMP), `mpi-k-means` (per la versione MPI) e `cuda-k-means` (per la versione CUDA).

Il programma viene eseguito dalla riga di comando nel modo seguente (si fa l'esempio della versione OpenMP; le altre versioni sono analoghe):

```
./omp-k-means K input_file output_file
```

Il programma legge l'input dal file il cui nome viene passato sulla riga di comando, e scrive il risultato su un altro file. Nel caso di MPI, il programma viene eseguito tramite `mpirun`; il processo 0 (il master) deve essere l'unico a leggere l'input e salvare l'output.

Il file di input è costituito da N righe, ciascuna contenente D numeri reali separati da spazi o tabulazioni; La riga i rappresenta le coordinate del punto $\boldsymbol{d}_i$ in D dimensioni. Al termine, il programma salve sul file di output le coordinate degli N punti, con in più un valore intero che indica il cluster assegnato a quel punto; quindi, ognuna delle N righe conterrà D coordinate (reali) e un intero compreso tra 0 e $K-1$ che indica il cluster. Il file di output contiene ulteriori righe di commento che iniziano con il carattere #. Il programma stampa a video alcune informazioni utili per monitorarne il progresso;

Viene fornita una versione seriale funzionante, che può essere usata come punto di partenza modificandola a piacimento, oppure si può iniziare da zero. Si tenga presente che, sebbene sia stata messa la massima cura nella stesura del codice, non se ne garantisce la correttezza, per cui ciascuno/a è responsabile di quanto consegnato.

3. Cosa consegnare

È necessario consegnare due versioni parallele del programma:

1. La prima, chiamata `omp-k-means.c`, deve utilizzare OpenMP;
2. La seconda deve utilizzare, a scelta, MPI oppure CUDA. Il file sorgente deve chiamarsi

rispettivamente `mpi-k-means.c` oppure `cuda-k-means.cu`

Oltre ai due programmi di cui sopra, è obbligatorio includere:

3. Una relazione in formato PDF, della lunghezza massima di 6 facciate, che descriva le strategie di parallelismo adottate e discuta la scalabilità ed efficienza dei programmi realizzati.

Ulteriori requisiti:

- I sorgenti devono essere adeguatamente commentati. All'inizio di ogni file deve essere indicato cognome, nome e numero di matricola dell'autore/autrice.
- Includere un file di testo chiamato README contenente le istruzioni per la compilazione e l'esecuzione dei programmi consegnati; chi lo desidera può usare un Makefile per la compilazione (non obbligatorio).
- Includere ogni altro file necessario alla compilazione (es., `hpc.h` per chi ne fa uso). È possibile scomporre il proprio codice in più sorgenti da compilare separatamente e riunire in fase di *linking*; in tal caso occorre documentare la procedura di compilazione nel file README di cui al punto precedente.
- I programmi verranno compilati ed eseguiti sul server `isi-raptor03.csr.unibo.it`. Ciò serve solo a verificare che i programmi compilino ed eseguano correttamente; le misure di prestazioni descritte nella relazione possono essere effettuate su proprio hardware.
- Tutti i programmi devono compilare sul server senza *warning*. Per le versioni OpenMP e MPI si usino i flag `-std=c99 -Wall -Wpedantic`.
- Verranno accettati programmi che impongono vincoli sulla dimensione del dominio (es., numero di punti multiplo di…), tenendo presente che questo comporterà una valutazione inferiore. La relazione deve riportare chiaramente l'esistenza di tali vincoli.
- La relazione non deve superare la lunghezza di **sei facciate in formato A4**, contando tutte le pagine (inclusi frontespizi, indici o altro; suggerisco di evitare frontespizi e indici, che sarebbero comunque poco utili su un documento così corto). Indicare il proprio cognome, nome e numero di matricola. Il formato della relazione è libero; viene fornito uno schema in formato LibreOffice per chi vuole uno spunto.
- La relazione non deve descrivere il codice riga per riga (per quello ci sono i sorgenti), ma le strategie di parallelizzazione adottate, eventualmente con l'ausilio di schemi o diagrammi. Inoltre, deve riportare e commentare le prestazioni delle versioni parallele realizzate utilizzando le metriche che si ritengono più appropriate.

Si invita a prestare la massima attenzione alla qualità della relazione. Vanno evitati errori grossolani come "~~ghost shell~~" invece di "ghost cell", "~~pull di thread~~" invece di "pool di thread", eccetera. Va inoltre evitato l'uso dei seguenti termini:

- "tempistica/tempistiche" (→ tempi di esecuzione)
- "prestante" o "performante" ("~~algoritmo più prestante~~" / "~~algoritmo più performante~~"→ algoritmo con prestazioni migliori)
- "randomico" (→ casuale)

Anche se la relazione non è una tesi di laurea, è utile prendere visione dei consigli per la stesura della tesi, limitatamente alla parte relativa ai consigli di scrittura. Può anche essere utile fare riferimento alla guida di stile di programmazione che viene adottata nel laboratorio di Algoritmi e Strutture Dati.

4. Suggerimenti

Il programma seriale fornito va inteso come un punto di partenza; è possibile modificarlo per adattarlo alle proprie esigenze, oppure partire da zero con una implementazione diversa. L'unica cosa che non può cambiare è il formato dell'input e dell'output, e il modo di invocare il programma.

Si suggerisce di **non modificare** la struttura dell'algoritmo, ad esempio usando strutture dati ad hoc per ottimizzare il calcolo delle distanze o simili. Ciò renderebbe il programma più complesso e difficile da parallelizzare.

Viene fornito un programma `inputgen.c` per generare ulteriori file di input, costituiti da punti distribuiti all'interno di rettangoli in D dimensioni in posizioni casuali. All'inizio del sorgente sono presenti alcune informazioni sulla compilazione e l'uso.

Il tempo di esecuzione del programma dipende dal contenuto dell'input, oltre che dalla sua dimensione, perché il numero di iterazioni può variare di molto in base alla scelta iniziale dei centroidi. Per studiare i tempi di esecuzione in funzione della dimensione dell'input **occorre modificare il codice per eseguire sempre lo stesso numero di iterazioni**, indipendentemente dal fatto che le posizioni dei centroidi si siano stabilizzate. Tuttavia, anche in questa versione modificata il programma deve calcolare lo spostamento massimo dei centroidi dopo ogni passo (in realtà, il quadrato dello spostamento massimo restituito dalla funzione `update_centroids()`).

Vengono forniti alcuni file di input, le cui caratteristiche sono riassunte nella tabella:

File	**Numero punti (*N*)**	**Dimensioni (*D*)**
avila.in	10430	10
iris.in	150	4
letter-recognition.in	20000	16
demo.in	1000	2
test-N50000-D80.in	50000	80
test-N100000-D30.in	100000	30
test-N150000-D20.in	150000	20

5. Modalità di svolgimento del progetto

- Il progetto deve essere frutto del lavoro individuale di chi consegna. Non è consentito condividere, in tutto o in parte, il codice o la relazione con altri fino al termine dell'anno accademico. In particolare, non è consentito rendere accessibile il codice o la relazione tramite repository pubblici come ad es. github (è però consentito usare repository privati).
- Non è consentito l'uso di AI generativa, né per la stesura del codice, né per la relazione. È ammesso l'uso di correttori ortografici, ad esempio quelli forniti con il programma di videoscrittura usato per la stesura della relazione.
- Il programma seriale fornito può essere modificato a piacimento.
- È ammesso l'uso di porzioni di codice pubblico (disponibile in rete o su libri e altre pubblicazioni), purché (*i*) la provenienza di codice scritto da terzi sia chiaramente indicata in un commento, e (*ii*) la licenza di tale codice ne consenta il riutilizzo. L'unica eccezione è il codice fornito dal docente a lezione o in laboratorio, che può essere usato liberamente senza necessità di indicarne la fonte.

- Le richieste di chiarimenti sulle specifiche (cioè su questo documento) vanno fatte tramite il forum apposito sulla piattaforma "Virtuale"; richieste inviate via mail non riceveranno risposta. Non verrà data risposta a domande relative al proprio codice: il progetto è una componente essenziale dell'esame, e deve essere svolto in totale autonomia.

6. Consegna e validità del progetto

Il progetto può essere consegnato in qualsiasi momento, prima o dopo aver sostenuto la prova scritta. Le eventuali valutazioni positive (del progetto e/o della prova scritta) restano valide fino alla sessione d'esame di settembre 2026 inclusa; dopo tale data i voti in sospeso saranno persi.

Non è possibile apportare modifiche o correzioni dopo la consegna: chi vuole migliorare la valutazione dovrà consegnare un nuovo progetto su nuove specifiche.

La consegna deve avvenire tramite la piattaforma https://virtuale.unibo.it/, mediante la quale è possibile caricare un unico archivio in formato `.zip` oppure `.tar.gz` contenente sia i sorgenti che la relazione. L'archivio dovrà essere denominato con il cognome e nome dell'autore (es., `MarzollaMoreno.zip` oppure `MarzollaMoreno.tar.gz`), e dovrà contenere una directory con lo stesso nome (es., `MarzollaMoreno/`) contenente a sua volta i file secondo il seguente layout:

`MarzollaMoreno/src/omp-k-means.c`

`MarzollaMoreno/src/mpi-k-means.c` (*se si svolge la versione MPI*)

`MarzollaMoreno/src/cuda-k-means.cu` (*se si svolge la versione CUDA*)

`MarzollaMoreno/src/...` (ogni altro file necessario)

`MarzollaMoreno/README` (*facoltativo*)

`MarzollaMoreno/Relazione.pdf`

Includere tutto quanto è necessario alla compilazione del codice, come ad esempio hpc.h o simili. La piattaforma Virtuale limita la dimensione dei file caricati a 20MB. **Non allegare file di input**: sono molto voluminosi e rischiano di eccedere la dimensione massima consentita.

7. Valutazione dei progetti

Un progetto verrà considerato **sufficiente** se soddisfa almeno i seguenti requisiti minimi:

- I programmi compilano ed eseguono correttamente su istanze di input anche soggette a vincoli (es., dimensione del dominio multipla di...) sul server `isi-raptor03.csr.unibo.it`.
- La relazione dimostra un livello sufficiente di padronanza degli argomenti trattati ed è scritta in modo adeguato (chiarezza, correttezza grammaticale, uso appropriato della punteggiatura, uso corretto della lingua). Chi non è madrelingua italiana può scrivere la relazione in inglese se lo ritiene utile.

Ulteriori aspetti che potranno comportare una valutazione superiore:

- Qualità del codice, in termini di generalità, chiarezza, ed efficienza (es., uso appropriato delle primitive e/o dei pattern di programmazione parallela appropriati ed efficienti).
- Qualità della relazione, in termini di correttezza, chiarezza, completezza e presentazione, tenuto conto della lunghezza massima consentita.

Ai fini di una valutazione elevata non è indispensabile che i programmi consegnati siano i più efficienti possibile; verrà piuttosto valutata la capacità di utilizzare i pattern di programmazione parallela più adeguati al problema e al paradigma di programmazione adottato, la chiarezza della relazione, nonché la correttezza delle metriche adottate per la misura delle prestazioni.

Chi desidera approfondire e applicare tecniche che non sono state viste a lezione è benvenuto/a, ma si fa presente che questo da solo non garantisce una valutazione migliore, soprattutto se sono presenti errori o si dimostrano carenze nelle competenze di base (es., l'analisi delle prestazioni non è corretta, il codice presenta dei bug, ecc.). Di conseguenza, prima di fare cose extra è importante assicurarsi che la parte obbligatoria sia stata svolta correttamente.

Sebbene il progetto possa essere consegnato in qualsiasi momento, effettuerò tre sessioni di valutazione al termine delle sessioni d'esame di gennaio/febbraio 2026, giugno/luglio 2026 e settembre 2026. Questo significa che alla fine di febbraio 2026, luglio 2026 e settembre 2026 valuterò tutti i progetti ricevuti fino a quel momento. Chi avesse delle scadenze, ad esempio per avere borse di studio o per laurearsi, è pregato/a di segnalarmelo alla consegna. È tuttavia obbligatorio consegnare il progetto almeno **10 giorni lavorativi prima della data entro la quale si richiede la correzione** per consentirmi di far fronte ad eventuali altri impegni accademici.

8. Checklist per la consegna

Prima di consegnare il progetto, assicurarsi che i requisiti fondamentali elencati nella lista seguente siano soddisfatti:

□	Vengono consegnate due versioni del programma, una che usa OpenMP, e una che usa (a scelta) MPI oppure CUDA.
□	I sorgenti compilano ed eseguono correttamente su `isi-raptor03.csr.unibo.it`.
□	La relazione è in formato PDF e ha lunghezza minore o uguale a 6 facciate.
□	I sorgenti e la relazione indicano chiaramente cognome, nome e numero di matricola dell'autore/dell'autrice.
□	Il progetto viene consegnato in un unico archivio .zip oppure .tar.gz, nominato con nome e cognome dell'autore, che include i sorgenti e la relazione in formato PDF.

9. Riferimenti bibliografici

[1] MacQueen, J. B. (1967). *Some Methods for classification and Analysis of Multivariate Observations*. Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability. Vol. 1. University of California Press. pp. 281–297.

[2] K-Means Clsutering su Wikipedia.

[3] Lloyd, Stuard (1982). *Least squares quantization in PCM*. IEEE Transactions on Information Theory, 28 (2), pp.129-137.

[4] Lloyd's Algorithm su Wikipedia.