

**UNIVERSITÀ DEGLI STUDI DI MESSINA**

**DIPARTIMENTO DI INGEGNERIA**

CORSO DI LAUREA MAGISTRALE IN

INFORMATICA(LM-18)

**PROGETTO DI HIGH PERFORMANCE COMPUTING:**

**Federated Parallel Classifier KNN**

Relazione a cura di:

**Giorgio Aveni Dario Miligi**

Matricola n° 556126 Matricola n° 552033

**ANNO ACCADEMICO 2023/2024**

**INDICE**

**CAPITOLO 1 – INTRODUZIONE**

**CAPITOLO 2 – ARCHITETTURA E TOOLS USATI**

**2.1 ARCHITETTURA DEL PROGRAMMA**

**2.2 MPI**

* 1. **PTHREAD**

**CAPITOLO 3 – CONTESTO FEDERATO**

**CAPITOLO 4 – ALGORITMO KNN**

**4.1 ESEMPIO**

**CAPITOLO 5 – IMPLEMENTAZIONE**

**5.1 IMPLEMENTAZIONE SERIALE**

* 1. **IMPLEMENTAZIONE PARALLELA FEDERATA**

**CAPITOLO 6 – CASO DI STUDIO**

**CAPITOLO 7 – RISULTATI SPERIMENTALI**

**INTRODUZIONE**

Il progetto in esame costituisce un approfondito esplorare nell'ambito dell'High-Performance Computing (HPC), con l'obiettivo di sviluppare un sistema avanzato di classificazione basato sull'algoritmo k-Nearest Neighbors (KNN). L'aspetto distintivo della nostra implementazione risiede nell'integrazione di un contesto federato nell'esecuzione parallela, una metodologia innovativa che ridefinisce il modo in cui il calcolo distribuito può essere concepito e implementato per ottenere prestazioni ottimali su hardware eterogeneo.

L'aspetto federato di questo sistema si manifesta nella decentralizzazione della gestione del dataset, con ogni nodo del calcolo addebitato dell'indipendente gestione del proprio "chunk" di dati. Questo paradigma federato non solo promuove l'autonomia e l'ottimizzazione delle risorse locali, ma rivoluziona anche il tradizionale modello di distribuzione centralizzata del carico di lavoro, introducendo una nuova era di calcolo distribuito avanzato e altamente efficiente.

L'implementazione sinergica dei paradigmi di programmazione parallela, MPI (Message Passing Interface) e pthreads (POSIX Threads), è stata studiata attentamente per massimizzare l'efficienza computazionale. MPI facilita la comunicazione tra processi, consentendo uno scambio strategico di informazioni, mentre pthreads gestisce la parallelizzazione fine dei thread, ottimizzando le operazioni di calcolo locale. Questa convergenza di tecnologie è stata progettata per ridurre il costo computazionale intrinseco all'algoritmo KNN, consentendo un'elaborazione più rapida e dinamica delle distanze, specialmente in contesti di grandi dimensioni.

Il nostro obiettivo è non solo implementare un sistema di classificazione avanzato ma anche introdurre un paradigma federato che potrebbe rivoluzionare il modo in cui affrontiamo i problemi computazionali su larga scala. Sfide significative emergono durante lo sviluppo, richiedendo soluzioni innovative per affrontare la gestione distribuita dei dati e garantire coerenza e precisione nell'esecuzione parallela. La relazione esaminerà come l'integrazione di MPI e pthreads offre soluzioni avanzate a tali sfide.

Attraverso test approfonditi e misurazioni delle prestazioni, analizzeremo criticamente l'efficacia delle scelte implementative e valuteremo l'impatto dell'aspetto federato sulla scalabilità complessiva del sistema. L'obiettivo è dimostrare come la decentralizzazione e la gestione distribuita del dataset possano portare a un miglioramento significativo delle prestazioni, soprattutto quando si lavora con grandi dataset.

In conclusione, il nostro progetto non solo si propone di implementare un algoritmo di classificazione avanzato ma introduce un paradigma federato rivoluzionario nell'ambito dell'High-Performance Computing. L'aspetto federato potrebbe delineare nuovi standard di efficienza e scalabilità, non solo per il KNN ma anche per una vasta gamma di applicazioni. La relazione fornirà una visione prospettica sulle potenzialità future di questa proposta, delineando percorsi di miglioramento e raffinamento del sistema alla luce delle sfide uniche dell'HPC.

**ARCHITETTURA E TOOL USATI**

Il programma è organizzato in diversi file per migliorare la modularità e la chiarezza del codice:

* **Header files**: I file di intestazione (header file) sono file al cui interno vengono posti prototipi di funzioni, definizione di tipi e di costanti simboliche. Il nostro programma prevede la presenza di due diversi file di intestazione: **header\_processes** ed **header\_threads**. Entrambi i file possiedono i prototipi di funzioni e la dichiarazione delle costanti utili ai processi (header\_processes) ed ai thread (header\_thread) con la definizione della struttura dati per gli stessi. Entrambi i file avranno l’estensione **.h** che indica proprio la tipologia del file di header.
* **Functions files**: I file funzioni contengono l’implementazione delle funzioni cui prototipi sono stati dichiarati nei file di header. Il programma prevede anche in questo caso la presenza di due file di funzioni: **processes\_functions** e **threads\_functions**. Ciascuno dei due file contiene dunque il codice relativo a tutte le funzioni che saranno utilizzate dai processi (processes\_functions) e dai threads (threads\_functions).
* Il file **mergeSort.h** implementa il codice relativo all’algoritmo di ordinamento mergeSort, che verrà utilizzato dai threads.
* **Knn.c**: il file in questione implementa il codice relativo all’algoritmo knn che verrà eseguito dai thread.
* **Hierarchical\_structure.c**: tale file è il main file del programma. In esso è implementata la struttura gerarchica dei processi nodi del cluster MPI.

Per poter sfruttare la modularità offerta dalla suddivisione del codice in diversi file è necessario includere la dipendenza tra i vari file con il comando **#include “nome\_file”**. Di seguito vengono riportate le dipendenze:

* I functions files (processes\_functions e threads\_functions) includono i file di intestazione header\_processes ed header\_thread, poiché al loro interno sono dichiarati i prototipi delle funzioni che sono implementate nei functions files.