Progetto 1: Implementazione di algoritmi paralleli per string matching

Dato un insieme di stringhe **S** ed uno stream di pacchetti di dati (pacchetti contenenti traffico di rete tipo TCP/UDP) **T** controllare se all'interno dei pacchetti è presente una delle stringhe di **S.** Utilizzare almeno uno tra gli algoritmi Knuth-Morris-Pratt ed Aho-Corasick.

Realizzare sia un'implementazione che utilizza un sistema a memoria distribuita MPI e sia un'implementazione che utilizza OpenMP.

Oltre a verificare la correttezza degli algoritmi implementati (ad esempio confrontando i risultati con quelli ottenuti da una versione single-thread), valutare le prestazioni degli algoritmi sviluppati in termini di Speed-up ed efficienza al variare del numero di thread e delle dimensioni del problema (numero stringhe di **S** o numero di pacchetti di **T**).

Il progetto è dimensionato per essere realizzato da un gruppo composto da due studenti.

La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;

Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Progetto 2: Implementazione di algoritmi paralleli per string matching

Dato un insieme di stringhe **S** ed uno stream di pacchetti di dati (pacchetti contenenti traffico di rete tipo TCP/UDP) **T** controllare se all'interno dei pacchetti è presente una delle stringhe di **S.** Utilizzare almeno uno tra gli algoritmi Knuth-Morris-Pratt ed Aho-Corasick.

Realizzare sia un'implementazione che utilizza i PThread, sia un'implementazione che utilizza una GPGPU.

Oltre a verificare la correttezza degli algoritmi implementati (ad esempio confrontando i risultati con quelli ottenuti da una versione single-thread), valutare le prestazioni degli algoritmi sviluppati in termini di Speed-up ed efficienza al variare del numero di thread e delle dimensioni del problema (numero stringhe di **S** o numero di pacchetti di **T**).

Il progetto è dimensionato per essere realizzato da un gruppo composto da due studenti.

La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;

Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Progetto 3: Implementazione di algoritmi paralleli per il simulare la crescita cristallina

Diffusion-limited aggregation (DLA) è un processo di formazione di cristalli nel quale le particelle si muovono in uno spazio 2D con moto browniano (cioè in modo casuale) e si combinano tra loro quando si toccano. DLA può essere simulato utilizzando una griglia 2D in cui ogni cella può essere occupata da uno o più particelle in movimento. Una particella diventa parte di un cristallo (e si ferma) quando si trova in prossimità di un cristallo già formato. I parametri di base della simulazione sono la dimensione della griglia 2D, il numero iniziale di particelle, il numero di iterazioni e il "seme" cristallino iniziale. Realizzare sia un'implementazione che utilizza MPI, sia un'implementazione che utilizza una GPGPU.

Oltre a verificare la correttezza degli algoritmi implementati (ad esempio confrontando i risultati con quelli ottenuti da una versione single-thread), valutare le prestazioni degli algoritmi sviluppati in termini di speed-up ed efficienza al variare del numero di processi/thread e delle dimensioni del problema (numero di particelle, numero di iterazioni e dimensioni della griglia).

Il progetto è dimensionato per essere realizzato da un gruppo composto da due studenti.

La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;

Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Progetto 4: Implementazione di algoritmi paralleli per il simulare la crescita cristallina

Diffusion-limited aggregation (DLA) è un processo di formazione di cristalli nel quale le particelle si muovono in uno spazio 2D con moto browniano (cioè in modo casuale) e si combinano tra loro quando si toccano. DLA può essere simulato utilizzando una griglia 2D in cui ogni cella può essere occupata da uno o più particelle in movimento. Una particella diventa parte di un cristallo (e si ferma) quando si trova in prossimità di un cristallo già formato. I parametri di base della simulazione sono la dimensione della griglia 2D, il numero iniziale di particelle, il numero di iterazioni e il "seme" cristallino iniziale. Realizzare sia un'implementazione che utilizza OpenMP, sia un'implementazione che utilizza una Pthread.

Oltre a verificare la correttezza degli algoritmi implementati (ad esempio confrontando i risultati con quelli ottenuti da una versione single-thread), valutare le prestazioni degli algoritmi sviluppati in termini di speed-up ed efficienza al variare del numero di processi/thread e delle dimensioni del problema (numero di particelle, numero di iterazioni e dimensioni della griglia).

Il progetto è dimensionato per essere realizzato da un gruppo composto da due studenti.
La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;
Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Progetto 5: Implementazione di algoritmi paralleli per il calcolo della similarità tra documenti

Dato un insieme di documenti **D** sviluppare un algoritmo parallelo per controllare la similarità dei documenti appartenenti a **D**. Sviluppare l'algoritmo utilizzando MinHash. Realizzare sia un'implementazione che utilizza MPI, sia un'implementazione che utilizza una GPGPU.

Oltre a verificare la correttezza degli algoritmi implementati (ad esempio confrontando i risultati con quelli ottenuti da una versione single-thread), valutare le prestazioni degli algoritmi sviluppati in termini di speed-up ed efficienza al variare del numero di processi/thread e delle dimensioni del problema (numero di documenti **D**).

Il progetto è dimensionato per essere realizzato da un gruppo composto da due studenti.

La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;
Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Riferimenti bibliografici:
[1] https://en.wikipedia.org/wiki/MinHash
[2] A. Broder, "Identifying and Filtering Near-Duplicate Documents", http://cs.brown.edu/courses/cs253/papers/nearduplicate.pdf
[3] http://matthewcasperson.blogspot.com/2013/11/minhash-for-dummies.html

Progetto 6: Implementazione di algoritmi paralleli per il calcolo della similarità tra documenti

Dato un insieme di documenti **D** sviluppare un algoritmo parallelo per controllare la similarità dei documenti appartenenti a **D**. Sviluppare l'algoritmo utilizzando MinHash. Realizzare sia un'implementazione che utilizza OpenMP, sia un'implementazione che utilizza una PThreads.

Oltre a verificare la correttezza degli algoritmi implementati (ad esempio confrontando i risultati con quelli ottenuti da una versione single-thread), valutare le prestazioni degli algoritmi sviluppati in termini di speed-up ed efficienza al variare del numero di processi/thread e delle dimensioni del problema (numero di documenti **D**).

Il progetto è dimensionato per essere realizzato da un gruppo composto da due studenti.
La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;

Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Riferimenti bibliografici:

[1] https://en.wikipedia.org/wiki/MinHash
[2] A. Broder, "Identifying and Filtering Near-Duplicate Documents", http://cs.brown.edu/courses/cs253/papers/nearduplicate.pdf
[3] http://matthewcasperson.blogspot.com/2013/11/minhash-for-dummies.html

Progetto 7: Implementazione di algoritmi paralleli per la soluzione di Sudoku

Esistono diversi approcci algoritmici per risolvere un sudoku. Selezionare almeno 2 degli approcci descritti in [1][2] (Backtracking, Simulated Annealing, Exact Cover (Dancing Links), Algoritmo di Crook) e implementarne una versione parallela utilizzando 2 tra i seguenti approcci: MPI, PThread, OpenMP, CUDA.
Oltre a verificare la correttezza degli algoritmi implementati (ad esempio confrontando i risultati con quelli ottenuti da una versione single-thread), valutare le prestazioni degli algoritmi sviluppati in termini di speed-up ed efficienza al variare del numero di processi/thread.

Il progetto è dimensionato per essere realizzato da un gruppo composto da due studenti.

La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;
Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Riferimenti bibliografici:
[1] https://en.wikipedia.org/wiki/Sudoku_solving_algorithms
[2] http://www.ams.org/notices/200904/tx090400460p.pdf

Progetto 8: Implementazione dell'algoritmo di clustering K-means in parallelo

Dato un insieme **O** di N oggetti, ognuno con *i* attributi, implementare l'algoritmo di clustering di k-means [1] utilizzando 2 tra i seguenti approcci: MPI, PThread, OpenMP, CUDA.
Oltre a verificare la correttezza degli algoritmi implementati (ad esempio confrontando i risultati con quelli ottenuti da una versione single-thread), valutare le prestazioni degli algoritmi sviluppati in termini di speed-up ed efficienza al variare del numero di processi/thread, del numero di oggetti N e del numero di attributi *i*.

Il progetto è dimensionato per essere realizzato da un gruppo composto da due studenti.
La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;
Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Riferimenti bibliografici:
[1] https://it.wikipedia.org/wiki/K-means

Progetto 9: Implementazione di un algoritmo per la soluzione del problema degli n-corpi

Dato un insieme **O** di N oggetti, di cui siano noti massa, posizione e velocità iniziale, implementare un algoritmo che simula l'evoluzione temporale del sistema soggetto alla forza di gravitazione universale (problema degli n-corpi [1]). Utilizzare 2 tra i seguenti approcci: MPI, PThread, OpenMP, CUDA. Oltre al metodo esaustivo [2, ch.6], implementare il metodo di Barnes-Hut [3]. Verificare la correttezza degli algoritmi implementati (ad esempio confrontando i risultati con quelli ottenuti da una versione single-thread), valutare le prestazioni degli algoritmi sviluppati in termini di speed-up ed efficienza al variare del numero di processi/thread e del numero di oggetti N.

Il progetto è dimensionato per essere realizzato da un gruppo composto da due/tre studenti.
La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;
Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Riferimenti bibliografici:
[1] https://it.wikipedia.org/wiki/Problema_degli_n-corpi
[2] Pacheco, Peter. An introduction to parallel programming. Elsevier, 2011.
[3] https://it.wikipedia.org/wiki/Algoritmo_di_Barnes-Hut

Progetto 10: Implementazione di un framework master/worker multithreading usando MPI/Pthreads

Il progetto richiede di sviluppare un framework basato sul paradigma master/worker. Il framework deve essere in grado di:

- mandare in esecuzione sui worker una serie di task, anche indicando il numero di threads da utilizzare per il workload
- ricevere i risultati ottenuti
- controllare lo stato dei workers (i.e. richiedere il numero di threads attualmente attivi)
- allocare in workload bilanciando il carico tra i workers,
- sviluppare un meccanismo di tolleranza ai guasti che permette di recuperare/riavviare un worker se viene rilevato un malfunzionamento

Il progetto è dimensionato per essere realizzato da un gruppo composto da uno/due studenti.

La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;
Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Progetto 11: Implementazione di un framework MapReduce usando MPI/OpenMP

MapReduce [1] è un framework molto utilizzato per realizzare applicazioni parallele in un sistema a memoria distribuita. Il JobTracker si occupa di gestire l'allocazione del workload tra i vari nodi del sistema, evitando al programmatore di doversi occupare dell'allocazione e sincronizzazione delle risorse. Il progetto prevede di sviluppare un Framework simile a MapReduce utilizzando MPI e OpenMP con le seguenti funzionalità:

- Funzioni per File System Distribuito (Scatter/Gather di un file su diversi nodi)
- InputReader/OutputWriter
- Task coordination
- Map
- Partitioning/Reduce

Il framework deve essere in grado di eseguire workload tipo *wordcount* su un insieme di file di teso e *find possible friends*, in cui ad ogni utente del social network, vogliamo suggerire i possibili amici da inserire nella propria lista guardando la sua lista attuale e quella degli altri utenti.

Il progetto è dimensionato per essere realizzato da un gruppo composto da due studenti.
La consegna del progetto (almeno una settimana prima dell'orale) consiste in:
(a) tutti i sorgenti (opportunamente commentati) necessari per il funzionamento;
(b) Una relazione contenente:
- La descrizione dettagliata dell'architettura dell'applicazione e delle scelte progettuali effettuate, opportunamente motivate.
- La descrizione delle eventuali limitazioni riscontrate;
- I risultati in termini di prestazioni opportunamente commentati;

Il giorno dell'orale è necessario preparare una presentazione powerpoint ed una demo del progetto.

Riferimenti bibliografici:

[1] https://en.wikipedia.org/wiki/MapReduce