COMPUTER ENGINEERING (LM55)

(Lecce - Università degli Studi)

Teaching PARALLEL ALGORITHMS

GenCod A003130

Owner professor Massimo CAFARO

Teaching in italian PARALLEL ALGORITHMS

Teaching PARALLEL ALGORITHMS

SSD code ING-INF/05

Reference course COMPUTER ENGINEERING

Course type Laurea Magistrale

Credits 9.0

Teaching hours Front activity hours: 81.0

For enrolled in 2016/2017

Taught in 2017/2018

Course year 2

Language ENGLISH

Curriculum PERCORSO COMUNE

Location Lecce

Semester First Semester

Exam type Oral

Assessment Final grade

Course timetable https://easyroom.unisalento.it/Orario

BRIEF COURSE DESCRIPTION

Algoritmi sequenziali

Introduzione. Ordini di grandezza. Analisi di algoritmi. Decrease and conquer. Divide and conquer. Ricorrenze. (7 ore) Algoritmi randomizzati. (5 ore) Transform and conquer. (2 ore) Lower bound for sorting. Linear time sorting algorithms: Counting sort, Radix sort and Bucket sort. (3 ore) Order statistics. Selection in expected and worst case linear time. (2 ore) Programmazione dinamica. (4 ore) Algoritmi greedy. (3 ore) Complessita' e calcolabilita'. NP-Completezza. (7 ore)

Algoritmi paralleli

1) Introduzione. Metodo scientifico moderno. Concorrenza e parallelismo. Equazioni di Bernstein. Evoluzione del supercalcolo. Calcolatori paralleli. Grafo delle dipendenze dei dati. Data parallelism. Functional Parallelism. Pipelining. Approcci per la programmazione di calcolatori paralleli. (2 ore)

2) Architetture parallele. Reti di interconnessione. Shared e switched media. Topologie di rete. 2D mesh. Binary tree. Hypertree. Fat tree. Butterfly. Ipercubo. Torus. Shuffle-exchange. Processor arrays. Multiprocessori centralizzati e distribuiti. Multicomputers asimmetrici e simmetrici. Clusters e reti di workstations. Tassonomia di Flinn. (5 ore)

3) Parallel algorithm design. Modello task-channel. Metodologia PCAM di Foster. Decomposizioni, tasks e grafo delle dipendenze dei tasks. Grain size di una decomposizione. Grado di concorrenza. Critical path length. Grafo delle interazioni dei tasks. Processi e mapping. Recursive decomposition, Data decomposition Exploratory decomposition, Speculative decomposition. Decomposizione di dati di input, intermedi e di output. Decomposizione ibride e gerarchiche. The Owner Computes Rule. Tasks static e dinamici. Comunicazione locale, globale, strutturata e non strutturata. Task interactions: read-only, read-write, one-way, two-way. Comunicazione static e dinamica, sincrona ed asincrona. Algoritmi red-black. Divide and conquer. Agglomerazione. Effetto surface to volume. Communication/computation ratio. Replicare la computazione per eliminare comunicazioni. Mapping. NP-completeness del mapping. Mapping static e dinamico. Mappings based on data partitioning. Mappings based on task graph partitioning. Hybrid mappings. Probabilistic mapping, cyclic and block-cyclic. Graph partitioning. Recursive bisection. Quad and Oct-trees. Space-filling curves. Scattered decomposition. Dynamic load-balancing. Centralized dynamic mapping, master-slave (manager-worker), chunck scheduling. Distributed dynamic mapping. Dynamic data driven mapping. Dynamic geometric decomposition: Adaptive Mesh Refinement (AMR). Esempi di progettazione di algoritmi paralleli: boundary value problem, reductions, n-body problem. Data input. (10 ore)

4) Message-Passing programming. Message-passing model. MPI. Circuit-SAT problem. Communicators. Point-to-point communications: vari tipi di send e receive. Collective communications: broadcast, reductions, scatter, gather, all-gather, all-to-all. Benchmark. (2 ore)

5) Crivello di Eratostene: progettazione, analisi, implementazione e benchmark. (2 ore)

6) Algoritmo di Floyd all-pairs shortest path: progettazione, analisi, implementazione e benchmark. (2 ore)

7) Performance analysis. Tempo di esecuzione sequenziale e parallelo. Communication overhead. Overhead totale. Speedup ed efficienza. Speedup relativo, reale, assoluto, relae asintotico e relativo asintotico. Cost-normalized speedup. Numero ottimale di processori. Speedup superlineare. Legge di Amdahl-Ware. Legge di Gustafson-Barsis. Metrica di Karp-Flatt. Legge di Lee (generalizzazione

della legge di Amdahl-Ware). Metrica di isoefficienza (Grama, Gupta and Kumar). Funzione di scalabilita' di Sun e Ni. Cost-optimality (work-efficiency). Cost-optimality ed isoefficienza. Grado di concorrenza ed isoefficienza. Impatto della non cost-optimality di un algoritmo parallelo. Minimum Parallel execution Time. Minimum Cost-Optimal Parallel Execution Time. (7 ore)

8) Moltiplicazione matrice-vettore. Algoritmo sequenziale. Rowwise block-striped decomposition. Columnwise block-striped decomposition. Checkerboard block decomposition. implementazione e benchmark. (3 ore)

9) Classificazione di documenti. Design. Manager-worker paradigm. Comunicazioni nonblocking. (2 ore)

10) Moltiplicazione di matrici. Algoritmo sequenziale iterativo row-oriented e ricorsivo block-oriented. Rowwise block-striped parallel algorithm. Algoritmo di Cannon. Algoritmo di Fox. Algoritmo DNS (dei tre indiani). (3 ore)

Esercitazioni. (21 ore)

REQUIREMENTS

Analisi I e II, Teoria della Probabilita'. Conoscenze pregresse del linguaggio di programmazione C.

COURSE AIMS

Il corso fornisce una moderna introduzione alla progettazione, analisi ed implementazione di algoritmi sequenziali e paralleli. In particolare, il corso e' basato su un approccio pratico alla programmazione parallela di algoritmi message-passing in linguaggio C con la libreria MPI.

Dopo aver seguito il corso, lo studente dovrebbe essere in grado di:

1) descrivere ed utilizzare le principali tecniche di progettazione di algoritmi sequenziali;
2) progettare, provare la correttezza, implementare ed analizzare la complessita' computazionale di algoritmi sequenziali;
3) comprendere le differenze tra algoritmi diversi che risolvono uno stesso problema e riconoscere quale algoritmo e' il migliore rispetto a condizioni diverse;
4) descrivere ed utilizzare algoritmi sequenziali di base;
5) descrivere ed utilizzare strutture dati di base; conoscere l'esistenza di strutture dati avanzate;
6) comprendere le differenze tra algoritmi sequenziali e paralleli;
7) progettare, provare la correttezza, implementare ed analizzare la complessita' computazionale di algoritmi paralleli basati su message-passing in C utilizzando la libreria MPI;
8) descrivere ed utilizzare algoritmi paralleli di base.

ASSESSMENT TYPE

L'esame e' orale. Opzionalmente, allo studente puo' essere assegnato un piccolo progetto. Duramte l'esame, allo studente viene chiesto di illustrare argomenti teorici per verificare la sua conoscenza e comprensione degli argomenti scelti. Allo studente puo' essere chiesto di progettare un semplice algoritmo per verificare la sua capacita' di identificare ed utilizzare le tecniche di progettazione piu' appropriate. Alternativamente, allo studente puo' essere chiesto di analizzare la complessita' computazionale di un breve frammento di codice.