SCHEDA DELL'INSEGNAMENTO DI **High Performance Computing**
Prof. Livia Marcellino

OBIETTIVI FORMATIVI DELL'UNITA' DIDATTICA

Il corso approfondisce le metodologie, le tecniche e gli strumenti per lo sviluppo di algoritmi e software in differenti ambienti di calcolo ad alte prestazioni, affrontando le problematiche attuali degli strumenti di calcolo avanzato per gli ambienti multicore e GPU.

Per l'attività di laboratorio, il corso prevede l'impiego del linguaggio di programmazione C/C++, dell'ambiente CUDA di nVIDIA e del tolboox MATLAB per il calcolo parallelo, per lo sviluppo di software parallelo che sfrutta la potenza elaborativa dei moderni processori grafici.

Conoscenza e capacità di comprensione: lo studente deve dimostrare di conoscere i fondamenti del calcolo parallelo, l'organizzazione della gerarchia della memoria dell'ambiente CUDA sia hardware che software e le strategie di parallelizzazione per alcuni nuclei computazionali di base della programmazione con e senza l'uso della shared memory.

Capacità di applicare conoscenza e comprensione: lo studente deve dimostrare di saper impiegare le strategie studiate e le API di CUDA per sviluppare algoritmi in ambiente multicore/GPU, sfruttando le conoscenze relative alle problematiche della parallelizzazione in ambiente ibrido ad alte prestazioni.

Autonomia di giudizio: lo studente deve essere in grado di sapere valutare in maniera autonoma i risultati di un algoritmo parallelo mediante l'analisi delle performance, in termini di guadagno, Gflops e gestione della memoria.

Abilità comunicative: lo studente deve essere in grado di illustrare un algoritmo parallelo e di documentare la sua implementazione in ambiente multicore/GPU.

Capacità di apprendimento: lo studente deve essere in grado di aggiornarsi e approfondire in modo autonomo argomenti e applicazioni specifiche di calcolo parallelo, anche accedendo a banche dati, repository on-line di software parallelo ed altre modalità messe a disposizione dalla rete.

PREREQUISITI DELL'UNITA' DIDATTICA

È necessario avere acquisito le conoscenze e le competenze trasmesse dai corsi della laurea triennale, tra cui Calcolo Parallelo e Distribuito I ed Elaborazione delle Immagini. Inoltre deve aver acquisito le competenze trasmesse dai corsi della laurea magistrale: Algoritmi e Strutture Dati II, Applicazioni di Calcolo Scientifico e Laboratorio di ACS, Architettura e Programmazione di Reti Avanzate e laboratorio di Architettura e Programmazione di Reti Avanzate, Sistemi Operativi Distribuiti e Lab. di Sistemi Operativi Distribuiti

CONTENUTI DELL'UNITA' DIDATTICA (CORSO)

Il calcolo ad alte prestazioni: definizione e motivazioni - Misura delle prestazioni di un calcolatore e del tempo di esecuzione di un software - Storia dell'evoluzione dei supercalcolatori - Architetture distribuite e multicore - Legge di Moore e successive rimodulazioni - Processori grafici (GPU) e ambiente CUDA: il parallelismo delle GPU (motivazioni, vantaggi e svantaggi) - Il General Purpose GPU - Array di multiprocessori – Unità elaborative delle GPU: host e device - Il concetto di kernel -

Grid e blocchi - Il pitch ed il concetto di coalescenza - L'architettura CUDA - Organizzazione della memoria: i registri, la local memory e la shared memory; la constant, la textured e la global memory - Programmare in parallelo con CUDA e il linguaggio C - le principali routine CUDA: allocazione, de-allocazione e scambio dati - Invocazione e dichiarazione di un kernel - API per la gestione della memoria - Il processo di mapping - Gestione dei dati 2D: routine per il picth - Gestione degli errori - Il timing - Allocazione e gestione di strutture dati nella shared memory - Routine per la sincronizzazione dei thread - Il device menagement - Il toolkit CUDAprof e la sua versione grafica CUDAprofiler - La libreria CUBLAS di CUDA per le operazioni di base tra matrici e vettori - Breve introduzione alla librerie BLAS, PBLAS, PLASMA - I livelli di CUBLAS - Routine predefinite per il passaggio dei dati tra host e device - Creazione dell'ambiente CUBLAS - Gestione degli errori. Routine per le operazioni di base.

Modalità di verifica dell'apprendimento

L'obiettivo della procedura di verifica consiste nel quantificare il livello di raggiungimento degli obiettivi formativi precedentemente indicati.
La procedura di verifica è indicata precisamente nella piattaforma di e-learning del Dipartimento di Scienze e Tecnologie. In sintesi, l'esame consiste nello sviluppo di un progetto assegnato dal docente che consiste nell'implementazione di un software per la risoluzione di un problema reale in ambiente GPU (70% del voto), una prova orale per esaminare la capacità di analisi di un software parallelo in ambiente GPU (30% del voto).

Testi di riferimento

A. Grama, G. Karypis, V. Kumar, A. Gupta: "Introduction to Parallel Computing (2nd Edition)". Addison Wesley
J. Sanders, E. Kandrot. Foreword by J. Dongarra: "CUDA BY EXAMPLE: An introduction to General-Purpose GPU Programming". NVIDIA

Tutte le lezioni sono disponibili (in formato pdf) sulla piattaforma di e-learning del Dipartimento di Scienze e Tecnologie, insieme con esercizi per autovalutazione, il manuale d'uso della libreria, articoli recenti sugli argomenti più innovativi.

SCHEDA DELL'INSEGNAMENTO DI High Performance Computing **(eng)**

OBJECTIVES OF UNIT TEACHING

The course extends the methodologies, techniques and tools for developing algorithms and software in high performance computing environments by addressing the current issues of advanced computing tools for multi-core and GPU environments.

For laboratory activity, the course involves the use of C/C++ programming language, the CUDA environment of nVIDIA and the parallel toolbox of MATLAB for the development of parallel software that leverages the processing power of modern graphics processors.

Knowledge and understanding: the student must demonstrate knowledge of the fundamentals of parallel computing, the organization of the CUDA environment memory hierarchy, both hardware and software, and the parallelization strategies for some basic computational kernels of programming with and without the use of shared memory.

Ability to apply knowledge and understanding: the student must demonstrate how to use the strategies studied and the CUDA APIs to develop algorithms in a multicore/GPU environment by exploiting knowledge about parallelization issues related to high-performance and hybrid environment.

Autonomy of judgement: the student must be able to independently evaluate the results of a parallel algorithm by means of performance analysis in terms of gain, Gflops, and memory management.

Communication skills: the student should be able to illustrate a parallel algorithm and document its implementation in multicore/GPU environments.

Learning skills: the student must be able to update and deepen topics and specific applications of numerical computing, even accessing databases, on-line scientific software repositories and other tools available on the web.

DIDACTIC UNIT PREREQUISITES

The attendant student must have acquired knowledge and skills transmitted in the 1st level degree, including: Parallel and Distributed Computing and Image processing. Moreover, the student must have acquired knowledge and skills transmitted in the following course of 2nd level degree: Algorithms and Data Structures II and lab ASD II, Scientific computing Application and lab ACS.

COURSE CONTENTS

High Performance Computiong: Definition and Motivation - Measurement of a computer's performance and software execution time - Supercomputer evolution history - Distributed and multicore architectures - Moore's law and subsequent refinements - Graphics processors (GPUs) and CUDA environment: GPU parallelism (motivations, advantages and disadvantages)- The General Purpose GPU - Multiprocessor Array - GPU Computing Units: Host and Device - The Kernel Concept - Grid and Blocks - The pitch and the concept of coalescence - The CUDA architecture -

Memory organization: registers, local memory and the shared memory; The constant, the textured and the global memory - Parallel programming with CUDA and the C language - The main CUDA routines: allocation, de-allocation and data exchange host/device - Invocation and declaration of a kernel - Memory management API - Mapping process - 2D data management: Picth routines - Error handling - Timing - Allocation and management of data structures in the shared memory - Thread synchronization routines - Device management - The CUDAprof toolkit and its graphic versione CUDAprofiler - The CUBLAS library of CUDA for matrix and vector basic operations - Brief introduction to BLAS, PBLAS, PLASMA libraries - CUBLAS levels - Default routines for data transfer between host and device - Creation of the CUBLAS environment – Error Managing - Routines for basic operations.

Learning assessment mode

The goal of the verification procedure is to quantify, for each student, the degree of achievement of the learning objectives listed above. To be specific, the exam consists of the project development assigned by the teacher consisting in the implementation of a software to solve a real problem in the GPU environment (70% of the vote), an oral test to examine the capacity to analyze a parallel software In the GPU environment (30% of the vote).

Textbooks and other teaching material

A. Grama, G. Karypis, V. Kumar, A. Gupta: “Introduction to Parallel Computing (2nd Edition)”. Addison Wesley

J. Sanders, E. Kandrot. Foreword by J. Dongarra: “CUDA BY EXAMPLE: An introduction to General-Purpose GPU Programming”. NVIDIA

All lessons are available as slides (in pdf format) on the e-learning platform of the Department of Science and Technology, together with self-assessment exercises, libraries manuals, exams, recent papers on the most innovative parallel topics.