PLANO DE ENSINO

Programa	Ciências Mecânicas (53001010053P0)
Nome	PRINCÍPIOS DE PROGRAMAÇÃO PARALELA
Sigla	PCMEC
Número	3937
Créditos	4
Período de Vigência	01/01/2012 -
Professor responsável	Eder Lima de Albuquerque
Disciplina obrigatória	Não

EMENTA

Objetivos:

Esta disciplina tem como objetivo discutir diferentes abordagens de programação paralela, fazer análise de desempenho de programas paralelizados e discutir a melhor técnica de paralelização, considerando o tamanho do problema e a infraestrutura de computação disponível para o programador.

Justificativa:

A disciplina tem o intuito de atender a demanda crescente de vários temas de trabalhos do Programa de Pós-Graduação em Ciências Mecânicas no qual a computação de alto desempenho está inserida. Como exemplos destes temas, podemos citar o escoamento de fluidos de alta complexidade, mecânica dos fluidos computacional, mecânica dos sólidos não linear, dentre outros.

Conteúdo:

1. Introdução: Evolução dos computadores. Programação sequencial e programação paralela. Diferentes arquiteturas dos processadores. Processadores com memória compartilhada. Processadores com memórias não compartilhadas. Placas gráficas. Processamento em placas gráficas. Taxonomia de Flym; 2**. Medidas de performance de computação paralela:** Eficiência e ganho de velocidade (speedup). Lei de Amdahl. Lei de Gustafson-Barsis. Métrica de Karp-Flatt. Métrica de isoeficiência. 3**. Paralelização com memória compartilhada:** O modelo de memória compartilhada. Programação paralela usando openmp: Um primeiro código openmp. Estruturas de repetição em paralelo. Condições de corrida. Seções críticas. Exemplos de códigos de programação paralela com openmp. **4. Programação paralela com memória distribuída:** O modelo de memória distribuída. Princípios do Message passing interface (MPI). A comunicação via mensagem entre os processadores. Distribuição dos dados nas memórias. Acesso aos dados. Exemplos de códigos de programação paralela com MPI. **5. Processamento em placas gráficas:** Os modelos de programação Cuda e opencl. Transferência de dados entre a CPU e a GPU. Exemplos de códigos de programação paralela com Cuda.

Forma de Avaliação	Development of a code using parallel programming (50% of the grade); Exercise lists (50% of the grade). **Serão atribuídas menções aos estudantes com base nas notas finais obtivas, de acordo com o critério de menções da UnB. Casos omissos serão resolvidos pelos professores da disciplina.**
Observação:	
Bibliografia:	1. M. J. Quinn. *Parallel programming in C with MPI and OpenMP*. Ed. MacGrawHill, 2004. 2. E. Bueler. *PETSc for Partial Differential Equations: Numerical Solutions in C and Python*. Ed. SIAM, 2020. 3. G. Ruetsch and M. Fatica. *CUDA Fortran for Scientists and Engineers: Best Practices for Efficient CUDA Fortran Programming*. Morgan Kaufmann, 2013. 4. H. P. Langtangen and A. Logg. *Solving PDEs in Python: The FeniCS: Tutorial I*. Ed. Springer, 2017 5. M. Curcic. *Modern Fortran: Building Efficient Parallel Applications.* Manning Publications Co., 2020.

Unit information

Program	Mechanical Science (53001010053P0)
Course unit	PRINCIPLES OF PARALLEL PROGRAMMING
Unit code	PPCMEC
Unit number	3937
Credit points	4
Period	01/01/2012 -
Professor	Éder Lima de Albuquerque
Prerequisites	No

Unit outline

Objective:

This unit aims to discuss different parallel programming approaches, perform performance analysis of parallel programs and discuss the best parallelization technique, considering the size of the problem and the computing infrastructure available to the programmer.

Purpose:

The discipline is intended to meet the growing demand for various work themes in the Graduate Program in Mechanical Sciences, in which high-performance computing is inserted. As examples of these themes, we can mention the flow of highly complex fluids, computational fluid mechanics, nonlinear solid mechanics, among others.

Contents:

1. Introduction: Evolution of computers. Sequential programming and parallel programming. Different processor architectures. Shared memory processors. Processors with non-shared memory. Graphics cards. Processing on graphics cards. Flym's Taxonomy; **2. Parallel computing performance measures:** Efficiency and speedup. Amdahl's Law. Gustafson-Barsis law. Karp-Flatt metric. Isoefficiency metrics. **3. Parallelization with shared memory:** The shared memory model. Parallel programming using openmp: A first openmp code. Repeating structures in parallel. Race conditions. Critical sections. Examples of parallel programming codes with openmp. **4. Parallel programming with distributed memory:** The distributed memory model. Message passing interface (MPI) principles. Message communication between processors. Distribution of data in memories. Data Access. Examples of parallel programming codes with MPI. **5. Processing on graphics cards:** The Cuda and opencl programming models. Data transfer between CPU and GPU. Examples of parallel programming codes with Cuda.

Assessment

A final exam (50% of the grade); Exercise lists (50% of the grade).

Obs:

Reference:	1. M. J. Quinn. *Parallel programming in C with MPI and OpenMP*. Ed. MacGrawHill, 2004. 2. E. Bueler. *PETSc for Partial Differential Equations: Numerical Solutions in C and Python*. Ed. SIAM, 2020. 3. G. Ruetsch and M. Fatica. *CUDA Fortran for Scientists and Engineers: Best Practices for Efficient CUDA Fortran Programming*. Morgan Kaufmann, 2013. 4. H. P. Langtangen and A. Logg. *Solving PDEs in Python: The FeniCS: Tutorial I*. Ed. Springer, 2017 5. M. Curcic. *Modern Fortran: Building Efficient Parallel Applications.* Manning Publications Co., 2020.