UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática

Unidade Universitária: FACULDADE DE COMPUTAÇÃO E INFORMÁTICA		
Curso: Ciência da Computação		
Disciplina: Computação Paralela		Código da Disciplina: ENEX50148
Professor(es): Calebe de Paula Bianchini Mário Olímpio Menezes Marcio Porto Feitosa	DRT: 113066-4 114606-6 1153419	Etapa: 05
Carga horária: 04 h/a – 02 Sala de aula – 02 Laboratório		Semestre Letivo: 1/2020

Ementa:

Estudo de modelos e arquiteturas de sistemas paralelos. Análise de algoritmos paralelos. Implementação de algoritmos paralelos em arquiteturas multicore. Implementação de algoritmos paralelos em arquiteturas manycore. Implementação de algoritmos paralelos em *clusters*

Objetivos:

Fatos e Conceitos	Procedimentos e Habilidades	Atitudes, Normas e Valores
- Estudar os princípios de computação paralela, arquiteturas, modelos e desafios, tais como, concorrência e sincronização. - Aprender a desenvolver algoritmos paralelos eficientes para resolver um problema. - Aprender a analisar a complexidade de um algoritmo paralelo, ponderando o ganho de desempenho (speedup) e sua eficiência. - Aprender a implementar um algoritmo paralelo utilizando OpenMP, pthreads ou uma combinação destas tecnologias. - Aprender a analisar um código paralelo para determinar os gargalos computacionais e otimizar o desempenho do código, bem como depurar um código paralelo e consertar seus erros. - Aprender os princípios de programação de aceleradores de computação	- Capacidade para definir a terminologia comumente utilizada em programação paralela, tal como eficiência e speedup. - Capacidade para descrever diferentes arquitetura paralelas, redes de interconexão, modelos de programação e algoritmos para operações comuns tais como multiplicação matriz-vetor. - Capacidade para desenvolver algoritmos paralelos eficientes para problemas dados, bem como analisar sua complexidade como função do tamanho do problema e do numero de processos/processadores. - Capacidade de mostrar os passos realizados por um algoritmo paralelo para determinados dados de entradas e numero de processadores. - Capacidade de implementar um algoritmo paralelo utilizando OpenMP, pthreads	Valorizar o ambiente computacional cooperativo para resolver problemas de eficiência e melhoria de desempenho da computação. - Trabalhar em equipe como forma de resolver problemas em sistemas paralelos envolvendo diversos setores da empresa. - Considerar que um programa paralelo envolve requisitos nas áreas de interconexão de computadores, hierarquia de memórias, desempenho e eficiência de algoritmos.

	ou uma combinação destas tecnologias. - Capacidade de analisar um código paralelo em termos de sua performance, determinar os gargalos computacionais e otimizar o desempenho do código. - Capacidade de, dado um código paralelo, depurá-lo e consertar seus erros. - Capacidade de desenvolver algoritmos apropriados para o uso de aceleradores.	

Conteúdo Programático:

1. Visão Geral da Computação Paralela - Introdução
2. Hardware Paralelo e Software Paralelo
2.1. Recapitulação de Organização - Arquitetura de von Neumann
2.2. Hardware Paralelo - Taxonomia de Flynn
2.3. Redes de Interconexão
2.4. Software Paralelo
2.5. Desempenho (Performance)
2.5.1. Lei de Amdahl
2.5.2. Lei de Gustafson
2.6. Medindo tempos
2.7. Projeto de Programas Paralelos
2.7.1. Metodologia de Foster
3. Programação de Memória Compartilhada com Pthreads
3.1. Posix Threads
3.1.2. Variáveis Globais
3.1.3. Inicializando Threads
3.1.4. Executando as Threads
3.1.5. Parando as Threads
3.2. Multiplicação Matriz-Vetor com Pthreads
3.3. Seções Criticas
3.3.1. Busy-Waiting
3.3.2. Mutex
3.3.3. Semáforos
3.3.4. Barreiras e Variáveis de Condição
3.4. Locks de escrita-leitura
3.4.1. Estudo de Caso - lista ligada ordenada compartilhada.
3.5. Thread-Safety
4. Programação de Memória Compartilhada com OpenMP
4.1. OpenMP - conceitos introdutórios
4.2. Paralelizando laços seriais com OpenMP
4.3. Paralelizando tarefas com OpenMP
4.4. Sincronização explicita de threads
4.5. Problemas típicos em programação para memórias compartilhadas
5. Projeto de Programas Paralelos
5.1. Resolvendo problemas não triviais
5.2. Problema dos n-corpos
5.3. Problema do caixeiro viajante
5.4. Algoritmos paralelos sem análogo serial.

6. Programação de GPUs
6.1. Aceleradores e Unidades de Processamento Gráfico de Proposito Geral
6.2. CUDA - Compute Unified Device Architecture
6.3. Exemplos de aceleradores - Nvidia GPUs
6.4. Arquitetura de software com CUDA/OpenACC
6.5. Exemplos de Programação CUDA/OpenACC

Metodologia:

- ✓ Aulas expositivas.
- ✓ Exercícios individuais e em grupos.
- ✓ Trabalhos/pesquisas extraclasse.
- ✓ Exercícios em laboratório.
- ✓ Prova escrita sobre os conteúdos da disciplina.
- ✓ Utilização do ambiente Mackenzie Virtual.

Critério de Avaliação:

--- N1 ---

PP1 – Prova Parcial 1: prova individual – 70%
Exercícios Propostos – Média dos exercícios propostos no **Lab** - 30%
Sendo que:
NI1 = 0,7 *PP1 + 0,3 * Media Exercícios Propostos

--- N2 ---

PP2 – Prova Parcial 2: prova individual – 30%
PROJETO: Projeto da disciplina a ser desenvolvido em grupo de três alunos peso de 70% da nota intermediária - Aula de Laboratório

Sendo que:
NI2 = 0,3 *PP2 + 0,7 * PROJETO

--- Média intermediária (MI) ---

MI = (N1 + N2)/2 + NP

--- Nota de Participação (NP) ---

NP = 1,0 (máximo 1,0 – de acordo com as entregas de exercícios de fixação, pesquisa e de laboratório e outras atividades de participação)

CRITÉRIOS DE APROVAÇÃO
se MI >= 7.5 e FREQUENCIA >= 75%, **APROVADO**.
se MI >= 8.5 e 65% <= FREQUENCIA < 75%, **APROVADO**.
se FREQUENCIA >= 75% e (MI+PROVA FINAL)/2 >= 6.0, **APROVADO**.

OBS: o aluno tem o direito de fazer uma PROVA SUBSTITUTIVA para substituir uma nota de uma avaliação que tenha se ausentado. A PROVA SUBSTITUTIVA contém todo o conteúdo do semestre. Caso o aluno tenha se ausentado em mais de uma avaliação, utilizar-se-á a nota de MAIOR PESO.

Bibliografia Básica:

- McCOOL, M., REINDERS, J., ROBINSON, A. **Structured Parallel Programming: Patterns for Efficient Computation**. New York: Morgan Kaufmann, 2012.

- PACHECO, P. **An Introduction to Parallel Programming**. New York: Elsevier, 2011.
- RAUBER, T.; RUNGER, G. **Parallel Programming for Multicore and Cluster Systems**. New York:
Springer Verlag, 2010.

Bibliografia Complementar:

- GEBALI, F. **Algorithms and Parallel Computing**. New York: Wiley, 2011.
HELIHY, M., SHAVIT, N. **The Art of Multiprocessor Programming**. New York: Morgan Kaufmann, 2012.
- KAMINSKY, A. **Building Parallel Programs: SMPs, Clusters, and Java**. Course Technology, CENGAGE Learning, 2010.
- KIRK, D.; HWU, W. **Programming Massively Parallel Processors: A Hands-On Approach**.2.ed. New York: Morgan Kaufmann, 2012.
- RAYNAL, M. **Concurrent Programming: Algorithms, Principles and Foundations**. New York: Springer, 2012.