Campus: São José dos Campos

Curso (s): Engenharia de Computação e Ciência da Computação

Unidade Curricular (UC): Programação Concorrente e Distribuída

Unidade Curricular (UC):*Concurrent and Distributed Programming*

Código da UC: 3580

Docentes Responsáveis: Denise Stringhini e Álvaro Luiz Fazenda		Contato (e-mail): *[opcional]*
Docente (s) Colaborador/a (es/as):		Contato (e-mail): [opcional]
Ano letivo: 2022	Termo: 8	Turma (s): I e N
Nome do Grupo/Módulo/Eixo da UC (se houver):		Idioma predominante em que a UC será oferecida: (x) Português ( ) English ( ) Español ( ) Français ( ) Libras ( ) Outro:
UC: (x) Fixa ( ) Eletiva ( ) Optativa	Oferecida como: (x) Disciplina ( ) Módulo ( ) Estágio ( ) Outro:	Oferta da UC: (x) Semestral ( ) Anual

Ambiente Virtual de Aprendizagem:
( ) Moodle
(x) Classroom
( ) Outro:
( ) Não se aplica

Pré-Requisito (s) - Indicar Código e Nome (s) da (s) UC: 2612 - Sistemas Operacionais

Carga horária total (em horas): 72h

Carga horária teórica (em horas): 42h	Carga horária prática (em horas): 30h	Carga horária de extensão (em horas, se houver):

Se houver atividades de extensão, indicar código e nome do projeto ou programa vinculado na Pró-Reitoria de Extensão e Cultura (ProEC):

Ementa:
Introdução à programação concorrente; Arquitetura de máquinas paralelas e distribuídas; Análise de dependências; Técnicas e algoritmos clássicos em programação concorrente e distribuída (seções críticas, exclusão mútua, semáforos, monitores, sincronização de relógios, etc); Expressando concorrência em sistemas de memória compartilhada e distribuída; Medidas de desempenho de aplicações paralelas; Exploração de paralelismo; solução de problemas com concorrência; Introdução a programação para arquiteturas Multicore/Manycores e GPGPU. Técnica de Map-Reduce.

Conteúdo programático:

1. Introdução à programação concorrente. Arquitetura de máquinas paralelas e distribuídas (introdução), paralelismo de múltiplos níveis.
2. Concorrência em sistemas de memória compartilhada (introdução): processos Fork-Join e Threads (Posix-Threads e Java-Threads), OpenMP.
3. Medidas de desempenho de aplicações paralelas: Speedup, Eficiência, Escalabilidade e Lei de Amdahl.
4. Técnicas e algoritmos clássicos em programação concorrente e distribuída: Seções críticas; Exclusão mútua (MuteX); Semáforos; Monitores.
5. Problemas clássicos de sincronização.
6. Concorrência em sistemas de memória distribuída: Modelo de Troca de Mensagens (MPI - Message Passing Interface).
7. Análise de dependências.
8. Exploração de paralelismo: Paralelismo de dados (decomposição de domínio) e paralelismo de fluxo (decomposição funcional).
9. Algoritmos clássicos em Sistemas Distribuídos:
    a. Sincronização de relógios, Exclusão mútua distribuída
    b. Eleição do Líder
    c. Consenso

10. Introdução a programação para arquiteturas Multicore/Manycores e GPGPU (*General Purpose Graphics Processing Unit*).
11. Técnica de Map-Reduce

Objetivos:

Geral: Apresentar aos alunos os fundamentos da programação concorrente para arquiteturas paralelas e/ou distribuídas.
Específicos: Ao final do curso os alunos deverão ser capazes de compreender os princípios da programação concorrente para arquiteturas paralelas e distribuídas, bem como projetar algoritmos segundo estes princípios.

Metodologia de ensino:

- Aulas expositivas;
- Atividades monitoradas em grupos de trabalho;
- Laboratório de programação;
- Atividades complementares a distância;
- Quizzes online;
- Listas de exercícios.

Avaliação:

- Avaliação parcial 1 (**P1**);
- Avaliação parcial 2 (**P2**);
- **Atividades Prática**s de programação, as quais poderão ser realizadas individualmente ou em grupos de até 3 alunos.
- **Quizzes** semanais online com questões sobre a matéria da semana.

A nota final será dada por:

Nota_Exercícios: (MédiaQuiz + 3* Média das Atividades Práticas)/4

Nota final = (5*média(P1,P2) + 5*Nota_Exercícios)/10

O **Exame** consiste em uma prova que abrange todo o conteúdo do semestre.

Bibliografia:

Básica:

1. BEN-ARI, M. Principles of concurrent and distributed programming. 2nd ed. Harlow: Addison-Wesley, 2006. 361 p ISBN 9780321312839.
2. ANDREWS, G.R. Foundations of Multithreaded, Parallel, and Distributed Programming, Addison-Wesley, 1999.
3. GRAMA, A.; GUPTA, A.; KARYPIS, G.; KUMAR, V. Introduction to Parallel Computing, 2nd Edition. Addison Wesley. 2003.

Complementar:

1. HERLIHY, Maurice; SHAVIT, Nir. The art of multiprocessor programming. Burlington: Elsevier, c2008. 508 p. ISBN 97801237005914.
2. GHOSH, Sukumar. Distributed systems: an algorithmic approach. Boca Raton: Chapman & Hall/CRC, 2006. 402 p. ISBN 1584885645.
3. RAUBER, Thomas; RÜNGER, Gudula. Parallel programming: for multicore and cluster systems. New York: Springer, c2010. 455 p. ISBN 9783642048173.
4. KIRK, David B.; HWU, Wen-Mei W. Programando para processadores paralelos: uma abordagem prática à programação de GPU. Rio de Janeiro: Elsevier, 2011. 212 p. ISBN 9788535241884.
5. HEROUX, Michael A.; RAGHAVAN, Padma; SIMON, Horst D. (ed.). Parallel processing for scientific computing. Philadelphia: SIAM, 2006. 397 p. ISBN 9780898716191.
6. BRESHEARS, Clay. The art of concurrency: a thread monkey's guide to writing parallel applications. Sebastopol, CA: O'Reilly, 2009. 285 p. ISBN 9780596521530.