CI-0117: Programación Paralela y Concurrente

Créditos: 4. **Horas**: 5. **Clasificación**: Curso propio, II ciclo, 2do año
Requisitos: Programación 2, Álgebra Lineal para Computación, Fundamentos de Arquitectura
Correquisitos: Análisis de Algoritmos y Estructuras de Datos

Grupos: Gr03 K 10-12, V9-12 104-IF, **Gr04** K 16-19, V 16-18 104-IF
Profesor Jeisson Hidalgo Céspedes <jeisson.hidalgo@ucr.ac.cr> Oficina 321-IF2. Casillero 04.
Consulta: K9-10,12-13 V15-16,18-19 104-IF
Asistente: José David Vargas Artavia <josed1608@gmail.com>

1 Descripción

La programación paralela y concurrente es una ampliación de las habilidades de programación serial desarrolladas en cursos previos. Esta ampliación es imprescindible por cuanto las plataformas de hardware actuales y futuras ofrecen características de computación paralela que no podrían ser aprovechadas por un(a) programador(a) que sólo domine la programación serial. Por lo anterior, este es un curso fundamental para los tres énfasis de la carrera.

2 Objetivos

El objetivo general del curso es resolver problemas de programación estudiando técnicas para la creación de software paralelo con el fin de sacar provecho a las nuevas arquitecturas de hardware por medio de la aplicación de principios básicos de diseño y "APIs" (*Application-Programming-Interfaces*) de uso extendido en la industria de software.

Durante el curso el estudiante desarrollará habilidades para:

1. Comprender y explicar la motivaciones y tendencias de la computación paralela para contextualizar el desarrollo de software paralelo en la actualidad mediante el estudio de las características generales de las tecnologías más relevantes.
2. Elaborar algoritmos paralelos y concurrentes correctos para construir programas paralelos y concurrentes efectivos mediante el uso del modelo de ejecución del lenguaje de programación (máquina nocional).
3. Construir programas paralelos y concurrentes eficientes para mejorar el desempeño con respecto a opciones seriales funcionalmente equivalentes, mediante la aplicación principios básicos de diseño de algoritmos paralelos.
4. Construir programas paralelos y concurrentes que aprovechen las arquitecturas paralelas de hardware vigentes mediante el uso de "APIs" ampliamente difundidas en la industria de software.
5. Depurar programas paralelos y concurrentes para asegurar su efectividad y calidad mediante el uso de herramientas adecuadas.
6. Evaluar el desempeño de programas paralelos y concurrentes para discernir si efectivamente representan una mejor opción con respecto a opciones seriales funcionalmente equivalentes mediante la aplicación de métricas básicas de uso común.

3 Contenidos

Objetivo	Eje temático	Desglose
1 Comprender motivaciones y tendencias	1.1 La necesidad de computación paralela	Las necesidades de desempeño crecientes que motiva el software paralelo.
	1.2 Hardware paralelo	Sinopsis de modelos de hardware paralelo.
	1.3 Software paralelo	Sinopsis de modelos de software paralelo.
2 Elaborar algoritmos paralelos y concurrentes	2.1 Jerarquía de Flynn	Descripción de las distintas arquitecturas paralelas.
	2.2 Técnicas de descomposición	Descomposición recursiva, de datos, exploratoria, especulativa y otras.
	2.3 Métricas de complejidad paralela	Diferentes métricas de complejidad para algoritmos paralelos.

3 Construir programas paralelos y concurrentes	3.1 Mapeo de tareas a procesos	Características de tareas e interacciones. Técnicas de mapeo para balanceo de carga. Técnicas para reducir la sobrecarga debida a la interacción de tareas.
	3.2 Modelos de programas paralelos	Paralelismo de datos, grafo de tareas, *work-pool* y otros.
4 Construir programas paralelos y concurrentes usando APIs	4.1 Concurrencia por hilos	Pthreads o primitivas básicas como *fork* en C, *mutex*, sincronización productor-consumidor, semáforos, bloqueos lectura-escritura, seguridad de hilos y otros temas afines.
	4.2 Programación por procesos	Aspectos básicos de la programación por procesos (como MPI).
	4.3 Entrada/Salida	Entrada y salida mediante procesos paralelos.
	4.4 Comunicación	Comunicación entre procesos paralelos.
	4.5 Tipos de datos	Tipos de datos específicos para programación por procesos.
	4.6 Desempeño	Evaluación del desempeño de programas basados en procesos paralelos.
	4.7 Ejemplos	Ejemplos clásicos de procesos paralelos.
	4.8 Programación paralela usando hilos	Programación paralela usando hilos (tal como OpenMP), la regla trapezoidal, pragmas (*parallel for, schedule* y otras), productores y consumidores, seguridad de hilos y otros temas.
5 Depurar	5.1 Depuración de procesos paralelos	Cómo depurar programas basados en procesos paralelos.
	5.2 Depuración de hilos paralelos	Cómo depurar programas basados en hilos paralelos.
6 Evaluar	6.1 Métricas y técnicas	Ley de Amdahl, ley de Gustafson y Barsis, métrica de Karp-Flatt, métrica de isoeficiencia, métricas de aceleración y eficiencia, métricas de escalabilidad.

4 Metodología y evaluación

40% Ejercicios
20% Proyectos (2)
40% Exámenes (2)

Se seguirá una metodología híbrida constructivista y tradicional. Durante las horas extraclase los alumnos estudiarán el material del curso y resolverán los ejercicios planteados. Las soluciones serán presentadas por los estudiantes en un repositorio de control de versiones. Las lecciones serán presenciales en un laboratorio de computadoras de la Escuela. Durante las lecciones se realizarán clases magistrales cortas y actividades de resolución de problemas con el fin de ayudar a los estudiantes a comprender las nociones y resolver los ejercicios planteados y proyectos. Los estudiantes aplicarán las nociones en dos proyectos a desarrollar en parejas. Los estudiantes realizarán dos exámenes parciales a responder en papel, en los que podrán consultar material estático. El profesor podría proponer actividades opcionales por crédito extra en la nota del curso en cualquiera de las evaluaciones.

5 Bibliografía

1. Pacheco, Peter S. “An introduction to parallel programming”. Morgan Kaufmann Pub, 2011.
2. McCool, Michael; Robison, Arch D.; Reinders, James. “Structured Parallel Programming: Patterns for efficient computation”. Elsevier, Inc. 2012.
3. Rauber, Thomas y Rünger, Gudula. “Parallel Programming: for multicore and cluster systems”. Springer-Verlag Berlin Heigelberg, 2010.
4. Grama, Ananth et.al. “Introduction to Parallel Computing”. Addison-Wesley, 2003.
5. Quinn, Michael J. “Parallel Programming in C with MPI and OpenMP”. McGraw-Hill Education, 2003.
6. Chandra, Rohit et.al. “Parallel Programming in OpenMP”. Morgan Kaufmann Pub, 2001.
7. Williams, Anthony. “C++ Concurrency in Action (Practical Multithreading)”, 1/ed. Manning Publications Co, 2012.