Programación Paralela y Concurrente

1. Características generales

Nombre:	Programación Paralela y Concurrente
Sigla:	CI-0117
Créditos:	4
Horas lectivas:	5
Requisitos:	CI-0113 Programación 2, MA-0292 Álgebra Lineal para Computación, CI-0114 Fundamentos de Arquitectura
Correquisitos:	CI-0116 Análisis de Algoritmos y Estructuras de Datos
Clasificación:	Curso propio
Ciclo: de carrera	II ciclo, 2do año
Docente(s):	Daniel Alvarado González
Datos de contacto:	daniel.alvarado_g@ucr.ac.cr | Oficina 245 edificio ECCI
Grupo(s):	**02, 04**
Semestre y año:	I-2025
Horario y lugar de clases:	Gr02: L 10-11, J 9-11 203-IF Gr04: K 18-20, V 18-19 301-IF
Horario y lugar de consulta:	L 15:00-18:00 / J 15:00-17:00 | Of. 245 o virtual con cita previa via telegram @YouPassButter o a través del grupo del curso: Gr02: enlace Gr04: enlace
Modalidad:	Presencial
Asistente(s):	Pendiente

2. Descripción

La programación concurrente y distribuida de datos y tareas 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 programadores que solo dominen la programación serial.

3. Objetivos

Objetivo general

El objetivo general del curso es que los estudiantes desarrollen habilidades para resolver problemas que requieren incremento del desempeño mediante la paralelización de datos y tareas a través de los paradigmas de programación concurrente y distribuido.

Objetivos específicos

Durante el curso el estudiante desarrollará habilidades para:

1. Comprender y explicar las 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. Diseñar soluciones concurrentes y distribuidas correctas.
3. Resolver problemas por paralelización de datos y paralelización de tareas.
4. Construir programas concurrentes para resolver problemas mediante hilos de ejecución y recursos compartidos.
5. Construir programas distribuidos para resolver problemas mediante procesos y recursos distribuidos.
6. Probar programas concurrentes y distribuidos para asegurar su efectividad y calidad mediante el uso de pruebas de software.
7. Evaluar y comparar el desempeño de programas concurrentes y distribuidos para determinar su incremento en el rendimiento respecto a versiones funcionalmente equivalentes mediante la aplicación de métricas básicas de uso común.
8. Explicar el modelo de ejecución concurrente y distribuido (máquina nocional) para implementar programas de forma correcta.

4. Contenidos

Objetivo	Eje temático	Desglose
1 Comprender motivaciones y	1.1 La necesidad de computación paralela	Las necesidades de separación de asuntos y desempeño que motiva el software paralelo.

tendencias	1.2 Hardware paralelo	Sinopsis de modelos de hardware paralelo. Jerarquía de Flynn
	1.3 Software paralelo	Concepto de proceso y de hilo de ejecución.
2 Diseñar soluciones concurrentes y distribuidas	2.1 Algoritmos paralelos	Diferencia con algoritmos seriales. Análisis espaciotemporal de algoritmos paralelos.
	2.2 Técnicas de descomposición	Descomposición recursiva, de datos, exploratoria, especulativa y otras.
	2.3 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.
	2.4 Modelos de programas paralelos	Paralelismo de datos, grafo de tareas, *work-pool* y otros.
3. Resolver problemas	3.1 Problemas de paralelización de datos y tareas	Resolver problemas como: consumidor-productor, filósofos comensales, problema de los fumadores, problema de la barbería, problema de Santa Claus, formar agua, y el problema de *Modus Hall*.
4, 8 Construir programas concurrentes con recursos compartidos	4.1 Concurrencia por hilos	Concepto de hilo de ejecución. Espacio de direcciones. Interfaces de programación por hilos (como Pthreads y OpenMP). Rastreo de memoria y procesamiento de hilos de ejecución.
	4.2 Integridad de hilos	Condiciones de carrera (regiones críticas). Código reentrante. Código *thread-safe*.
	4.3 Mecanismos de sincronización	Espera activa. Mecanismos de sincronización provistos por el API (como *mutex*, semáforos, candados de lectura-escritura, variables de condición, barreras, reducciones).
5, 8 Construir programas concurrentes con recursos distribuidos	5.1 Concurrencia por procesos	Concepto de proceso. Memoria distribuida. Interfaces de programación por procesos (como MPI). Rastreo de memoria y procesamiento de procesos.
	5.2 Entrada y salida	Entrada y salida mediante procesos paralelos.
	5.3 Comunicación	Comunicación punto a punto y comunicación colectiva entre procesos.
6	6.1 Pruebas de	Pruebas de correctitud en programas

Probar	software	concurrentes y distribuidos (como caja negra y caja blanca).
7 Evaluar y comparar	7.1 Métricas	Ley de Amdahl. Métricas de aceleración (*speedup*), eficiencia, y escalabilidad.
	7.2 Desempeño	Medición del tiempo de pared. Gráficos de desempeño.

5. Metodología

Se sigue una metodología híbrida tradicional-constructivista 100% presencial con aprendizaje basado en proyectos y aula invertida. En el caso de aula invertida, el alumnado estudia de forma independiente los materiales asignados por el personal docente y resolverán ejercicios evaluados antes de la lección. Las lecciones presenciales se dedicarán al acompañamiento de estudiantes en la resolución de ejercicios, tareas, y proyectos relacionados a dichos materiales.

Las horas de consulta requieren cita previa a través de telegram. Las horas de consulta nunca son para repetir lecciones ya impartidas. En caso de haberlo, toda interacción virtual será grabada.

6. Evaluación

%	Rubro	Descripción
25 %	Tareas	Tareas individuales a presentar en control de versiones u otros medios indicados en sus respectivos enunciados. Pueden ser de ponderaciones variables asignadas a criterio docente.
25 %	Ejercicios y quices	Se podrán realizar exámenes cortos (quices) en cualquiera de las lecciones del curso, en papel o Mediación Virtual. La duración de cada quiz será especificada en su enunciado. De acuerdo con el artículo 15 del Reglamento de Régimen Académico Estudiantil dichas reglas quedan establecidas después de este aviso. Los ejercicios cortos se plantearán tanto para trabajo en clase o como para extra-clase con plazos establecidos. Se entregan en control de versiones u otros medios indicados en su enunciado.

		Todos los ejercicios y quices pueden ser de diferente ponderación a criterio docente.
30 %	Proyecto 1	Proyecto orientado a concurrencia y distribución de tareas. Se evalúa con al menos tres entregables de igual ponderación cuyas fechas son indicadas en sus respectivos enunciados.
20 %	Proyecto 2	Proyecto orientado a concurrencia y distribución de datos. Se evalúa con al menos dos entregables de igual ponderación cuyas fechas son indicadas en sus respectivos enunciados.

Los estudiantes resolverán dos problemas con la metodología y herramientas de trabajo por **proyectos**, en equipos de tres o cuatro estudiantes que pueden variar entre proyectos. Equipos de menos integrantes requieren visto bueno previo de los docentes. Los equipos de estudiantes realizarán todas las fases del proceso de resolución de problemas que incluyen: el análisis (recabado de requerimientos), el diseño de la solución, implementación, y pruebas de software.

Los proyectos se evaluarán con evidencia en un repositorio de control de versiones y presentaciones orales de producto, organizadas en avances. Es obligatorio que todos los miembros del equipo participen en las presentaciones de los proyectos, de lo contrario se deberá justificar la ausencia y tener el visto bueno quienes presentarán. Es obligatoria la evidencia de aporte equilibrado de cada miembro en el historial de cambios del repositorio de control de versiones. El no cumplimiento de estas normas será calificado con cero. Las fechas de presentación de avances y rubros de evaluación serán especificados en sus respectivos enunciados.

7. Lineamientos

1. Una asignación asincrónica se considera tardía si se entrega después de las 6 a.m. posterior al día límite de entrega y será calificada con cero.

2. Además de los lineamientos estipulados en esta carta al estudiante, cada evaluación podrá indicar lineamientos específicos en sus respectivos enunciados.

3. Los quices se realizan durante las lecciones y en cualquier momento durante el transcurso de la lección, y solo se repondrán en los casos que establece el Reglamento de Régimen Académico Estudiantil en su artículo 24. Se recomienda a los estudiantes contar con medios adicionales que garanticen el acceso a Mediación Virtual en caso de anomalías como fallos eléctricos o de

telefonía fija, como disponer de la aplicación Moodle o un navegador para dispositivos móviles preconfigurados.

4. Todas las evaluaciones son estrictamente individuales excepto en aquellas en las que se especifique explícitamente el trabajo grupal en el enunciado.

5. Todo trabajo debe ser entregado de forma digital excepto los quices que sean realizados en papel.

6. En todos los trabajos y las evaluaciones de los estudiantes, se calificará la redacción, ortografía, estructura y contenido.

7. Toda solución programada digitalmente debe compilar y correr, de lo contrario será considerada como un producto inaceptable y calificado con cero. Recuerde que puede recurrir a mecanismos en el lenguaje de programación para desactivar código fuente que impida su ejecución. Tampoco se aceptan soluciones que usen paradigmas de programación distintos a los evaluados.

8. En toda asignación se evaluarán las buenas prácticas de programación, como identificadores significativos, indentación, apego a una convención de estilo, documentación de interfaces e implementaciones de subrutinas, modularización y reutilización de código. Serán castigadas malas prácticas de programación como vulnerabilidades, redundancia de código, fugas y accesos inválidos de memoria, condiciones de carrera, serialización innecesaria de la concurrencia, y otras prácticas estipuladas durante las lecciones del curso.

9. Si se detecta que una solución contiene una espera activa, será calificada con cero.

10. La persona docente podría ofrecer un incremento en la nota hasta un máximo de la mitad de la diferencia por correcciones tras una evaluación revisada. Este es un beneficio que se ofrece a discreción y disponibilidad de la persona docente. Este beneficio se pierde en evaluaciones grupales si la persona estudiante realiza un cambio de equipo de trabajo.

11. En cualquier evaluación se podrían ofrecer rubros opcionales por crédito extra a discreción de los docentes hasta un máximo de 10% de la evaluación.

12. El código de tareas y los avances de proyecto se entrega en un repositorio de control de versiones, cuyo historial es evidencia evaluable. No se aceptan entregas por correo electrónico ni otro medio. Tampoco un comprimido dentro del repositorio, o un único *commit* con todo el trabajo elaborado, o cualquier otra situación que impida reflejar la natural evolución de la solución o la colaboración de los miembros del equipo a lo largo del tiempo. Los mensajes en el historial de cambios deben significativos.

13. En caso de un desequilibrio significativo en la participación de los miembros de un equipo, deben reportarlo al docente. Los equipos en esta situación pueden acogerse a una coevaluación que podría otorgar un beneficio hasta un máximo de 10% de la evaluación a los miembros que trabajaron demás.

Es ilegal presentar como propio, código parcial o total escrito por otras personas u obtenido de fuentes de información, como por ejemplo de libros, Internet, o herramientas de inteligencia artificial generativa (ej.: ChatGPT), sin la debida referencia al autor de la propiedad intelectual. En cualquier asignación en que se sospeche de plagio se aplicará una calificación de cero y el debido proceso estipulado en el Reglamento de Orden y Disciplina de los Estudiantes de la Universidad de Costa Rica.

8. Cronograma

La tabla de abajo muestra las semanas (columna "S") y fechas tentativas de cobertura de los contenidos, que pueden ser reajustadas de acuerdo con el avance durante el ciclo lectivo. Los números en la columna "Ejes tem." corresponden a los ejes temáticos de los contenidos del curso (sección 4). Las fechas de entrega de los ejercicios y avances de proyecto estarán sujetas a esta cobertura temática, y se comunicará oportunamente. La columna "Eval." tiene las fechas tentativas de las evaluaciones para tareas (T1–T4), avances de proyecto 1 (P1.1–P1.3), y avances de proyecto 2 (P2.1–P2.2).

S	Fecha	Conceptos	Ejes tem.	Eval.
1	10mar	Paradigmas. Algoritmos concurrentes. Herramientas	1.1–1.3, 2.1	
2	17mar	Hilo de ejecución. Indeterminismo. Memoria privada/compartida	2.1, 4.1, 6.1	
3	24mar	Espera activa. Cond. carrera. Mutex Seguridad condicional. Semáforo.	4.2, 4.3	T1
4	31mar	Productor-consumidor de buffer acotado/no acotado	3.1, 4.3	
5	07abr	Patrón productor-consumidor. Sincronización: aviso	2.4, 3.1, 4.3	T2
6	21abr	Encuentro. Multiplex. Barrera	3.1, 4.3	
7	28abr	Paralelismo de datos. Descomp. Mapeo. Métricas. Amdahl. *Profiling*.	2.1–2.4, 7.1, 7.2	P1.1
8	05may	OpenMP: Regiones paralelas. Ciclos paralelizados	7.1, 7.2	
9	12may	OpenMP: Mapeo (*scheduling*). Reducciones.	2.1–2.4, 4.1–4.3	T3
10	19may	Distribución. Proceso. Rastreo. MPI. Clústers. Entrada y salida	1.3, 5.1–5.2	
11	26may	Comunicación punto a punto. Control de distribución.	4.3, 5.3	

				P1.2
12	02jun	Comunicación colectiva. Difusión. Reducciones.	2.3, 2.4, 5.3	
13	09jun	Variable de condición. Filósofos comensales, inanición.	3.1, 4.3	T4
14	16jun	Lectores-escritores. Candados de lectura-escritura.	3.1, 4.3	
15	23jun	Código thread-safe y reentrante. Falso compartido	4.2	P1.3
16	30jun	*(Seguimiento de proyectos)*		
17	07jul	*(Semana de exámenes)*		P2

9. Bibliografía

1. Pacheco, Peter S. "An introduction to parallel programming". Morgan Kaufmann Pub, 2011.
2. Rauber, Thomas y Rünger, Gudula. "Parallel Programming (for multicore and cluster systems)". Springer-Verlag Berlin Heigelberg, 2010.
3. Grama, Ananth et.al. "Introduction to Parallel Computing". Addison-Wesley, 2003.
4. Quinn, Michael J. "Parallel Programming in C with MPI and OpenMP". McGraw-Hill Education, 2003.
5. Chandra, Rohit et.al. "Parallel Programming in OpenMP". Morgan Kaufmann Pub, 2001.
6. Downey, Allen B. Little Book of Semaphores. Disponible en https://greenteapress.com/wp/semaphores
7. Zaconne, Giancarlo. "Python Parallel Programming" 2nd Ed., 2019
8. Trobec, Roman; Slivnik, Boštjan; Bulić, Patricio & Robič, Borut. "Introduction to Parallel Computing (From Algorithms to Programming on State-of-the-Art Platforms)". Springer, 2018
9. Peterson, James Lyle. "Petri Net Theory and the Modeling of Systems". Prentice Hall PTR, USA. 1981
10. Reisig, Wolfgang. Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Springer Publishing Company, Incorporated. 2013

9. Recursos estudiantiles

Para información sobre recursos estudiantiles disponibles en la UCR, incluyendo el Sistema de bibliotecas y la normativa universitaria vigente, favor visitar la página: https://www.ecci.ucr.ac.cr/vida-estudiantil/servicios-institucionales-para-estudiantes/guia-de-recursos-estudiantiles-de-la-ucr