Sistemas Distribuidos y Paralelos

Año 2026

Carrera/ Plan:

Ingeniería en Computación Plan 2024

Año: 5
Régimen de Cursada: *Semestral*
Carácter (Obligatoria/Optativa): Obligatorio
Correlativas: Concurrencia y Paralelismo
Profesor/es: Dr. Adrian Pousa
Hs. semanales: 6

FUNDAMENTACIÓN

La evolución tecnológica de los procesadores ha motivado el procesamiento paralelo. Formar al alumno (que ya tiene conocimientos previos de Concurrencia y sus aplicaciones) en los fundamentos de los sistemas paralelos, los paradigmas de programación paralela y las métricas de rendimiento asociadas resulta un aporte fundamental para el futuro profesional. Esta tarea de formación se combina con trabajo experimental sobre sistemas paralelos concretos, disponibles en la Facultad.

OBJETIVOS GENERALES

Caracterizar los problemas de procesamiento paralelo desde dos puntos de vista: la arquitectura física y los lenguajes de programación, poniendo énfasis en la transformación de algoritmos secuenciales en paralelos.
Describir los modelos de cómputo paralelo y los paradigmas de programación paralela.
Estudiar las medidas de performance asociadas al paralelismo.
Plantear casos concretos de procesamiento paralelo, resolubles sobre distintas arquitecturas multiprocesador.

EJES TRANSVERSALES

01. Diseño e implementación de diversas Arquitecturas de Computadoras y todos los subsistemas relacionados. (2-Medio)
03. Especificación, proyecto y Desarrollo de Software y Sistemas Conjuntos de Hardware y Software haciendo uso de conceptos, métodos y herramientas de gestión de proyectos, ingeniería de software, base de datos, experiencia del usuario, elicitación, análisis, especificación y validación de requerimientos. (2-Medio)
04. Desarrollo de Redes de Computadoras y de Redes de Computadoras de área amplia,locales, inalámbricas y móviles. (1-Bajo)
05. Sistemas de Gestión de Recursos de Hardware y Software a sistemas generales, de tiempo real, distribuidos, para dispositivos fijos y móviles. (2-Medio)
06. Proyecto, desarrollo, dirección, control, construcción, operación y mantenimiento de Sistemas de Procesamiento de Señales, Sistemas Embebidos y sus periféricos incluido en software de soporte, Sistemas Computarizados de automatización y control y Sistemas Conjuntos de Hardware y Software. (2-Medio)
07. Certificación del funcionamiento, condición de uso o estados de Sistemas de Procesamiento de Señales, Sistemas Embebidos, Sistemas Computarizados de automatización y control, Sistemas Conjuntos de Hardware y Software. (1-Bajo)
08. Proyecto, Dirección y Aseguramiento de la calidad en lo referido a Seguridad Informática. (1-Bajo)
09. Identificación, formulación y resolución de problemas de ingeniería en computación. (2-Medio)
10. Concepción, diseño y desarrollo de proyectos de ingeniería en computación. (1-Bajo)

12. Utilización de técnicas y herramientas de aplicación en la ingeniería en computación. (2-Medio)
13. Generación de desarrollos tecnológicos y/o innovaciones tecnológicas. (2-Medio)
14. Fundamentos para el desempeño en equipos de trabajo. (1-Bajo)

CONTENIDOS MINIMOS

Arquitecturas de procesamiento paralelo.
Modelos de comunicación. Métricas de performance.
Memoria compartida, Memoria distribuida, esquemas mixtos.
Lenguajes y sistemas operativos para procesamiento paralelo.
Paradigmas de resolución de sistemas paralelos.
Adaptación entre arquitectura y software.
Aplicaciones.

PROGRAMA ANALÍTICO

Unidad 1: Conceptos básicos

Concurrencia y Paralelismo. Objetivos del procesamiento paralelo. Concepto de Sistema Paralelo. Software de un Sistema Paralelo. Hardware de un Sistema Paralelo. Motivación de los Sistemas Paralelos. Áreas de aplicación. Limitaciones actuales de hardware. Ley de Moore. Conceptos de paralelismo implícito y explícito. Paralelismo implícito. Conceptos de Segmentación/Pipelining y división funcional. Transparencia. Procesador Escalar y Superescalar. Paralelismo explícito. Comportamiento de las aplicaciones: CPU-Bound, Memory-Bound, I/O-Bound, Híbridos por fases. Etapas en el diseño de algoritmos paralelos: Descomposición, Comunicación, Aglomeración y Mapeo. Herramientas de desarrollo. Herramientas basadas en el modelo de memoria compartida, Herramientas basadas en el modelo de memoria distribuida, Modelos híbridos. Características del procesamiento paralelo en los lenguajes de programación y los sistemas operativos. Características del desarrollo de aplicaciones paralelas. Concepto de High Performance Computing (HPC).

Unidad 2: Sistema de memoria

Sistema de memoria. Limitaciones. La importancia de la memoria caché. Conceptos asociados a la memoria cache (Estados, Niveles, Contención y Coherencia) Principio de localidad: Localidad Temporal y Espacial.
Principio de localidad aplicado a la multiplicación de matrices
Minimizando el espacio de memoria. Caso de estudio matrices triangulares.
Conceptos de contadores hardware para evaluar el comportamiento del sistema de memoria.

Unidad 3: Arquitecturas Paralelas

Clasificación de arquitecturas paralelas. Por Organización lógica: mecanismo de control (Taxonomía de Flynn: SISD, SIMD, MISD y MIMD) y modelos de comunicación, (arquitecturas de memoria compartida UMA-NUMA y de memoria distribuida). Por Organización física: organización del espacio de direcciones, redes de interconexión y granularidad de los procesadores.
Evolución y Tendencias en las arquitecturas paralelas. Multiprocesadores, Clusters, Multiclusters, Grid, Multicores (homogéneos, heterogéneos y asimétricos), Manycores (GPUs y otros coprocesadores), Cloud para HPC. Computación cuántica.
Top 500 y Green 500.
Redes de comunicación en sistemas paralelos y distribuidos orientadas a HPC: Ethernet, Infiniband, Myrinet.

Unidad 4: Principios de diseño de algoritmos paralelos

Paralelismo explícito. Etapas de diseño de algoritmos paralelos: Descomposición, Comunicación, Aglomeración y Mapeo.
Etapa de Descomposición. Descomposición en tareas: Descomposición de Datos o de Dominio y Descomposición Funcional. Dependencias de tareas, Grafo de dependencias. Embarrassingly Parallel Problems. Características de las tareas: Tareas estáticas y dinámicas, Comportamiento uniforme y no uniforme, Volumen de tareas y Granularidad. Técnicas de descomposición de datos: recursiva, basada en datos (de salida, de entrada - problema de la frontera, de entrada/salida y de datos intermedios), exploratoria y especulativa. Técnica de descomposición híbrida.
Etapa de asignación/aglomeración. Reducción del volumen de tareas. Asignación, balance de carga. Aglomeración, localidad.
Etapa de Comunicación. Patrones de comunicación: Global/Local, Estructurado/No Estructurado, Estático/Dinámico, Síncrono/Asíncrono. Overhead de las comunicaciones.
Etapa de Aglomeración. Reducción de overhead y optimización de la localidad. Conservación de la escalabilidad. Costos asociados a la Ingeniería de Software.
Etapa de Mapeo. Mapeo estático. Estrategias de mapeo estático: basado en datos (arrays y grafos), en particiones de tareas y jerárquico. Mapeo dinámico. Estrategias de mapeo dinámico: centralizados y distribuidos. Mapeo dinámico y balance de carga.

Diseño sobre arquitecturas modernas. Heterogeneidad e integración: arquitecturas, sistemas operativos, herramientas y compiladores. Afinidad.

Unidad 5: Programación de algoritmos paralelos. Modelo de memoria compartida

Conceptos del Modelo de Programación sobre Memoria Compartida. Concepto de thread. Herramientas de desarrollo Posix Threads (Pthreads) y OpenMP.
Pthreads: Gestión de hilos, Sincronización, Afinidad.
OpenMP: modelo Fork-Join, Estructura de control paralela, Distribución de trabajo entre hilos, Gestión de entorno de datos, Sincronización, Funciones y variables de ambiente, Afinidad.

Resolución de aplicaciones específicas.

Unidad 6: Programación de algoritmos paralelos. Modelo de memoria distribuida

Conceptos del Modelo de Programación sobre Memoria Distribuida. Antecedentes: Herramienta de desarrollo PVM. Herramientas de desarrollo MPI: Funcionamiento y Estructura de programa, Comunicación Punto a Punto y Colectiva. Ocultamiento de la latencia.

Resolución de aplicaciones específicas.

Unidad 7: Programación de algoritmos paralelos. Modelo híbrido

Arquitecturas Paralelas Híbridas y desafíos en la programación paralela. Modelos de Programación Híbridos: MPI-Pthreads y MPI-OpenMP. Compilación y Modularidad.

Resolución de aplicaciones específicas.

Unidad 8: Métricas

Importancia de las métricas en los sistemas paralelos. Métricas de rendimiento, Speedup, Speedup Absoluto y Relativo. Speedup lineal, Speedup perfecto, Speedup superlineal. Eficiencia. Overhead. Balance de carga. Escalabilidad en los sistemas paralelos: Ley de Amdahl y Ley de Gustafson, Definición de Escalabilidad, Modelo de Isoeficiencia.
Métricas en arquitecturas heterogéneas. Concepto de potencia de cómputo relativa (PCR). Limitaciones del speedup. Ejecución Paramétrica.

Unidad 9: Paradigmas

Paradigmas de programación paralela: Paralelismo de Datos SIMD/SPMD, Divide y Vencerás, Pipelines, Algoritmos sistólicos, Master-Worker, Task pools. Modelos híbridos.

Ejemplos de resolución de aplicaciones específicas.

Unidad 10: Introducción a la programación sobre GPUs

Era Manycore - Introducción al concepto de GPGPU. Características paralelas de las GPUs.
Arquitectura Nvidia: Sistema de procesamiento y Sistema de memoria. Modelo de programación Nvidia CUDA: Declaración de variables, Gestión de la memoria global, Gestión de hilos, Kernel, Modularidad, Variables Built-in y Thread ID, Planificación, Multithreading por Hardware.
CUDA Capability.
Jerarquía de memoria Nvidia. Memorias On-Chip vs Memorias Off-Chip. Capacidades, Velocidades y Ancho de Banda. Conceptos de Latencia, Alcance y Tiempo de Vida. Memoria Global: patrón de acceso, unificación de transacciones y minimización de volumen de lectura. Memoria de Constantes y Memoria de Texturas. Memoria compartida: Bancos y Conflictos de Bancos. Registros y Memoria Local. Limitaciones.
Optimizaciones Nvidia CUDA: Coalescencia y prefetching. Divergencia. Mezcla y granularidad de instrucciones. Ocupación y asignación de recursos.
CUDA Streams: Impacto de las transferencias H2D y D2H. Ocultamiento de la latencia mediante CUDA Streams. Concurrencia a nivel Kernel y a nivel de memoria. Copia de memoria asíncrona.

Ejemplos de resolución de aplicaciones específicas.

BIBLIOGRAFIA

- "Sistemas Paralelos - Material de estudio de la Cátedra Sistemas Distribuidos y Paralelos" Adrian Pousa, 2025.
- "An Introduction to Parallel Computing. Design and Analysis of Algorithms" Grama, Gupta, Karypis, Kumar, 2003.
- "Foundations of Multithreaded, Parallel and Distributed Programming" Andrews, 2000.
- "Introduction to HPC for Scientists and Engineers" Hager, Wellein, 2011.
- "An introduction to parallel programming" Peter Pacheco, 2011.
- "Parallel Programming for Multicore and Cluster Systems" Rauber, 2010.
- "Parallel programming in C with MPI and OpenMP" Quinn, 2003.
- "Sourcebook of Parallel Computing" Dongarra, Foster, Fox, 2003.
- Jason Sanders, Edward Kandrot "CUDA by Example An Introduction to General Purpose GPU Programming". NVIDIA Corporation, Addison Wesley, 2011.
- M. F. Piccoli, "Computación de Alto Desempeño utilizando GPU". XV Escuela Internacional de Informática. Editorial Edulp, 2011.
- John Cheng, Max Grossman, Ty McKercher, "Professional CUDA C Programming", John Wiley & Sons, 2014.
- David B. Kirk, Wen-mei W. Hwu, "Programming Massively Parallel Processors - A Hands-on Approach" 3ed, NVIDIA Corporation and Wen-mei W. Hwu, Elsevier, 2017.

METODOLOGÍA DE ENSEÑANZA

La asignatura se organiza en clases teóricas y clases prácticas experimentales, ambas de carácter presencial.

Las clases teóricas introducen los conceptos teóricos de la materia, que luego se aplican en las clases prácticas.

Las clases prácticas experimentales se llevan a cabo en una de las Salas de Cómputo de la Facultad y en equipamiento especial del III-LIDI, accedido de forma remota. En las clases prácticas se trabajará sobre un conjunto de problemas que deberán resolverse utilizando las arquitecturas disponibles. Además, podrán incluirse explicaciones relacionadas con el trabajo en la sala y con el uso del equipamiento.
Las consultas se realizarán durante los días de clases prácticas. Eventualmente, se responderán consultas mediante alguna plataforma virtual de forma asincrónica (entorno IDEAS) o sincrónica (Webex o similar) cuando se trate de dudas puntuales de menor complejidad.

Redictado

La cátedra no considera necesario realizar un redictado.

Alcance de los ejes transversales

Los temas desarrollados a lo largo de la asignatura abarcan distintos aspectos de hardware y software. A partir del estudio del hardware de bajo nivel, se presenta el concepto de arquitectura paralela. Se clasifican y analizan en profundidad las principales arquitecturas paralelas actuales, considerando su diseño e implementación. Se estudia su interacción con el sistema operativo, el sistema de comunicaciones y los mecanismos de seguridad, el sistema de entrada/salida, la jerarquía de memoria y los lenguajes y herramientas de programación paralela. También se incluye la gestión de estos recursos, considerando su implementación, optimización y administración.

Se desarrolla el proceso completo de diseño, implementación y análisis de aplicaciones paralelas. A partir de algoritmos secuenciales, se aplican técnicas para identificar posibilidades de paralelización. Luego, se estudian las distintas etapas de diseño hasta obtener una o más implementaciones de un algoritmo paralelo, en función de las herramientas utilizadas y de la arquitectura en la que se ejecutará.

Finalmente, se adquieren los conocimientos necesarios para definir adecuadamente los distintos escenarios de prueba y realizar un análisis experimental mediante el uso de métricas, a fin de analizar el rendimiento y otros atributos del algoritmo paralelo. A partir de un análisis exhaustivo, se podrán determinar las soluciones y combinaciones de hardware y software más adecuadas para la resolución de un problema específico.

Evaluación

La materia se aprueba mediante la evaluación satisfactoria de dos aspectos: los conceptos teóricos y los conceptos prácticos.

Durante la cursada se evalúan los conceptos prácticos, mientras que los conceptos teóricos se evalúan a través de un examen final.
Los conceptos prácticos, o simplemente cursada, se aprueba mediante la realización de un trabajo experimental integrador y su correspondiente defensa.
El trabajo es individual y consiste en resolver uno o más problemas específicos sobre arquitecturas paralelas, utilizando las herramientas vistas en clase y realizando un análisis de escalabilidad de las soluciones propuestas.
La entrega del trabajo es por los mecanismos establecidos por la cátedra, no se aceptan trabajos publicados en repositorios, ni enviados fuera de término, ni reenviados por error.
En caso de que el trabajo presentado no cumpla con los requisitos mínimos establecidos por la cátedra, se otorgará una única instancia de reentrega. Una vez que el trabajo sea considerado satisfactorio, se habilitará la instancia de defensa.
La defensa del trabajo experimental consiste en responder preguntas relacionadas con el trabajo realizado y con los contenidos estudiados durante las prácticas experimentales. La cátedra determinará, en cada caso, si esta instancia se desarrollará en modalidad oral (coloquio) o escrita. La defensa no cuenta con una instancia de evaluación de recuperatorio.
En caso de que, durante la defensa, se demuestre que el trabajo no fue realizado por el alumno, que se desconoce el funcionamiento del código o su estructura, o que este fue copiado total o parcialmente, automática o manualmente, se considerará la pérdida de la cursada y la obligación de recursar la materia.

Una vez aprobada la cursada, quienes se encuentren en esa condición podrán presentarse a rendir el examen final en cualquiera de las mesas establecidas en el calendario académico de la Facultad, previa inscripción.
El examen final es escrito y comprende los temas del programa analítico desarrollados en clase.
La calificación final de la materia es la nota del examen final.

Quedarán eximidos de rendir el examen final quienes aprueben el *régimen de promoción*. El *régimen de promoción* es un mecanismo de evaluación de los conceptos teóricos durante el semestre en curso; su aprobación implica la eximición del examen final.

Régimen de promoción de examen final

Para acceder al régimen de promoción de examen final deben cumplirse los siguientes requisitos:

- Estar inscripto en modalidad *Promoción*.
- Cumplir con 80% de las condiciones de asistencia a las clases teóricas. Específicamente, la asistencia se considerará válida de acuerdo con el método aplicado en cada clase:

  Asistencia a clase: en algunas clases se pasará lista.

  Participación escrita: en otras clases se planteará una pregunta relacionada con el tema, y se dará un tiempo breve para desarrollarla por escrito y entregarla. Si la respuesta es correcta o regular, se considerará que la participación fue activa y la asistencia será válida. En caso de respuestas insuficientes, se registrará como ausente.

En cada clase se evaluará por uno de estos métodos; no se aplican ambos simultáneamente.

Quienes cumplan con todo lo anterior, podrán realizar el trabajo experimental integrador en grupo de a lo sumo dos personas.

- Aprobar la cursada sin haber reentregado el trabajo experimental integrador.
- Obtener una calificación de 6 o más en el examen de promoción, que comprende los temas del programa analítico desarrollados en clase y no cuenta con instancia de examen de recuperatorio.
- Tener aprobadas las materias correlativas antes de diciembre del corriente año.

La calificación final de la materia es la nota del examen de promoción.

No se evaluará a ningún alumno que no esté inscripto en actas, ya sea de cursada, promoción o examen final, o que no cumpla con los requisitos establecidos previamente.

Evaluación de los ejes transversales

Durante las prácticas experimentales es posible evaluar el desempeño de los grupos de trabajo, pudiendo conformarse grupos de hasta dos personas para resolver los distintos problemas planteados.
El trabajo experimental y su posterior defensa permiten evaluar la capacidad de idear, diseñar e implementar algoritmos paralelos utilizando diferentes herramientas de software. Además, se evalúa la comprensión de las arquitecturas paralelas y su aplicación adecuada para la correcta ejecución de los algoritmos implementados.
Se considera también la capacidad de generar escenarios de prueba, aplicar métricas, analizar resultados y extraer conclusiones.
Los conceptos teóricos se evalúan mediante un examen individual.

Uso adecuado de los recursos provistos por la cátedra

La ejecución de los experimentos correspondientes al trabajo experimental integrador se realizará utilizando los recursos provistos por la cátedra, en particular el Cluster de Cómputo del III-LIDI, compartido por todos los alumnos. La gestión de estos recursos compartidos se realiza mediante un *Resource Manager*, lo que permite organizar las distintas ejecuciones sin interrupciones ni conflictos.

Los alumnos deberán comprender y cumplir con las normas mínimas de uso del cluster, las cuales estarán debidamente documentadas y explicadas por la cátedra.

Serán sancionados los alumnos que:

- Hagan un uso indebido de los recursos, afectando a otros alumnos, bloqueando accesos, interrumpiendo el servicio o causando daños físicos o lógicos.
- Incurran en plagio o copiado de trabajos de otros alumnos u otras fuentes, y no demuestren haberlo resuelto por sus propios medios.
- Accesos indebidos y violaciones a la seguridad y privacidad.

Entre las posibles sanciones se incluyen, por ejemplo, la pérdida de la cursada y la obligación de recursar la materia.

Cronograma de clases y evaluaciones

Cronograma de clases teóricas:

Clase	Fecha	Contenidos/Actividades
1	11/3	Unidad 1 (Conceptos básicos) Unidad 2 (Sistema de memoria)
2	18/3	Unidad 3 (Arquitecturas Paralelas. Clasificación. Evolución y Tendencias)
3	25/3	Unidad 4 (Principios de Diseño de algoritmos paralelos)
4	1/4	Unidad 5 (Programación de algoritmos paralelos. Modelo de memoria compartida) Pthreads y OpenMP
5	8/4	Unidad 6 (Programación de algoritmos paralelos. Modelo de memoria distribuida MPI) Unidad 7 (Programación de algoritmos paralelos. Modelo híbrido)
6	15/4	Unidad 8 (Métricas)
7	22/4	Unidad 9 (Paradigmas)
8	29/4	Unidad 10 (Programación sobre GPUs. Modelo de programación CUDA. Jerarquía de memoria Nvidia)
9	6/5	Unidad 10 (Programación sobre GPUs. Optimizaciones. CUDA Streams. Modelos Híbridos)

Cronograma de prácticas experimentales:

Clase	Fecha	Contenidos/Actividades
1	18/3	Práctica 1 – Optimización de algoritmos secuenciales
2	25/3	Práctica 1 – Optimización de algoritmos secuenciales
3	1/4	Práctica 2 y 3 – Modelo de memoria compartida – Pthreads y OpenMP
4	8/4	Práctica 2 y 3 – Modelo de memoria compartida – Pthreads y OpenMP
5	15/4	Práctica 4 y Práctica 5 – Modelo de memoria distribuida – MPI e Híbridos
6	22/4	Práctica 4 y Práctica 5 – Modelo de memoria distribuida – MPI e Híbridos
7	29/4	Práctica 5 –Introducción a la programación sobre GPUs
8	6/5	Práctica 5 –Introducción a la programación sobre GPUs

Cronograma de entregas y defensas:

Fecha tentativa	Contenidos/Actividades
13/5	Presentación del trabajo a entregar
27/5	Entrega
10/6	Resultados de Entrega
17/6, 24/6	Defensa del trabajo a quienes cumplieron satisfactoriamente con la Entrega
24/6	Re-entrega
1/7	Resultados de Re-entrega
8/7	Defensa del trabajo a quienes cumplieron satisfactoriamente con la Re-Entrega

Evaluaciones:

Fecha tentativa	Evaluación previstas
8/7	Examen de Promoción

Contactos de la cátedra:

Mail: apousa@lidi.info.unlp.edu.ar
Plataforma virtual: IDEAS (https://ideas.info.unlp.edu.ar)

Firma del profesor responsable