	Carrera/ Plan:
SISTEMAS PARALELOS	Licenciatura en Informática Plan 2021/Plan 2015/Plan 2012 Licenciatura en Sistemas Plan 2021/Plan 2015/Plan 2012
Año 2024	**Año:** 4to **Régimen de Cursada:** Semestral **Carácter (Obligatoria/Optativa):** Obligatoria (LI), Optativa (LS) **Correlativas**: Programación concurrente **Profesor/es:** Enzo Rucci **Hs. semanales teoría**: 2hs **Hs. semanales práctica**: 4hs

FUNDAMENTACIÓN

La evolución tecnológica de los procesadores ha impuesto 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 performance 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 métricas de performance asociadas al paralelismo, así como modelos de predicción de performance orientados a diferentes arquitecturas multiprocesador.
Plantear casos concretos de procesamiento paralelo, resolubles sobre distintas arquitecturas multiprocesador.
Conocer las tendencias tecnológicas en el área de procesamiento paralelo.

RESULTADOS DE APRENDIZAJE

1.1. Describir y explicar los conceptos, teorías y métodos matemáticos relativos a la informática, equipamiento informático, comunicaciones informáticas y aplicaciones informáticas de acuerdo con el plan de estudios (Básico).
1.2. Describir las características de los últimos avances en hardware y software y sus correspondientes aplicaciones prácticas (Adecuado).
1.3. Describir los avances informáticos actuales e históricos y demostrar cierta visión sobre tendencias y avances futuros (Adecuado).
2.5. Analizar la medida en la que un determinado sistema informático cumple con los criterios definidos para su uso actual y desarrollo futuro (Básico).
3.1. Definir y diseñar hardware/software informático/de red que cumpla con los requisitos establecidos (Adecuado).
3.5. Aplicar las correspondientes competencias prácticas y de programación en la creación de programas informáticos y/u otros dispositivos informáticos (Adecuado).
5.5. Diseñar y llevar a cabo investigaciones prácticas (por ejemplo, de rendimientos de sistemas) para interpretar datos y extraer conclusiones (Adecuado).

COMPETENCIAS

- CGS6- Capacidad para interpretar la evolución de la Informática con una visión de las tendencias tecnológicas futuras.
- CGT1- Identificar, formular y resolver problemas de Informática.
- CGT4- Conocer e interpretar los conceptos, teorías y métodos matemáticos relativos a la informática, para su aplicación en problemas concretos de la disciplina.
- CGT5- Utilizar de manera efectiva las técnicas y herramientas de aplicación de la Informática.
- CGT7- Contribuir a la generación de desarrollos tecnológicos y/o innovaciones tecnológicas.
- LI-CE1 – Planificar, dirigir, realizar y/o evaluar proyectos de especificación, diseño, implementación, verificación, validación, puesta a punto, mantenimiento y actualización para arquitecturas de sistemas de procesamiento de datos, con capacidad de incorporar aspectos emergentes del cambio tecnológico.
- LI-CE4 – Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real, especificación formal de los mismos, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software/sistemas de información que se ejecuten sobre equipos de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfases humano computador y computador-computador.
- LI-CE6 – Controlar las normas de calidad en el software o software integrado a otros componentes. Capacidad de evaluación de perfomance de sistemas de software y sistemas que integren hardware y software.
- LS-CE1- Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real. Especificación formal, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software que se ejecuten sobre sistemas de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfases humano computador y computador-computador.
- LS-CE5- Establecer métricas y normas de calidad y seguridad de software, contralando las mismas a fin de tener un producto industrial que respete las normas nacionales e internacionales. Control de la especificación formal del producto, del proceso de diseño, desarrollo, implementación y mantenimiento. Establecimiento de métricas de validación y certificación de calidad. Capacidad de evaluación de perfomance de sistemas de software y sistemas que integren hardware y software.
- LS-CE9- Analizar y evaluar proyectos de especificación, diseño, implementación, puesta a punto, mantenimiento y actualización de sistemas de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico.

CONTENIDOS MINIMOS (de acuerdo al Plan de Estudios)

- 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.
- Arquitecturas de almacenamiento.
- Green computing.

PROGRAMA ANALÍTICO

Unidad 1: Conceptos básicos
Paralelismo. Objetivos del procesamiento paralelo.
Proceso y Procesador. Interacción, comunicación y sincronización de procesos.
Concurrencia y Paralelismo. Modelos de Concurrencia.
Impacto del procesamiento paralelo sobre los sistemas operativos y lenguajes de programación.
Concepto de Sistema Paralelo.
Speedup y Eficiencia de algoritmos paralelos.
Concepto de asignación de tareas y balance de carga.
Balance de carga estático y dinámico.

Unidad 2: Arquitecturas orientadas a Procesamiento Paralelo
Paralelismo implícito: tendencias en las arquitecturas de microprocesadores.
Optimización de perfomance en los sistemas de memoria. Manejo de memoria cache.
Estructura de control y modelos de comunicaciones en plataformas de procesamiento paralelo.
Clasificación por mecanismo de control (SISD, SIMD, MISD, MIMD), por la organización del espacio de direcciones, por la granularidad de los procesadores y por la red de Interconexión.
Análisis del impacto del tiempo de comunicación en el speedup alcanzable.
Clusters de PCs. Multiclusters.
Evolución de los procesadores. Memory Wall. Power Wall. ILP Wall. Multicores.

Unidad 3: Principios de diseño de algoritmos paralelos
Metodología de diseño de algoritmos paralelos.
Técnicas de descomposición. Características de los procesos. Interacción.
Técnicas de mapeo de procesos/procesadores. Balance de carga.
Métodos para minimizar el overhead de la interacción entre procesos.
Modelos de algoritmos paralelos.
Problemas paralelizables y no paralelizables.
Paralelismo perfecto. Paralelismo de datos. Paralelismo funcional. Paralelismo mixto.

Unidad 4: Modelos y Paradigmas de Computación Paralela
Paradigma Master/Slave.
Paradigma Divide/Conquer.
Paradigma de Pipelining.
Paradigma SPMD.
Combinación de paradigmas.

Unidad 5: Métricas del paralelismo
Métricas de rendimiento tradicionales.
Fuentes de overhead en procesamiento paralelo.
Speedup. Rango de valores. Speedup superlineal.
Eficiencia. Rango de valores. Grado de paralelismo alcanzable. Efecto de la heterogeneidad.
Ley de Amdhal. Ley de Gustafson.
Efecto de la granularidad y el mapeo de datos sobre la performance.
Escalabilidad de sistemas paralelos.
Métricas relacionadas con el consumo energético.

Unidad 6: Programación de algoritmos paralelos sobre plataformas con memoria compartida.
Concepto de thread.
Estándar Pthreads. Modelo de ejecución. Primitivas de sincronización. Control de atributos en threads.
Estándar OpenMP. Modelo de ejecución. Directivas. Funciones.
Análisis de problemas.

Unidad 7: Programación de algoritmos paralelos sobre plataformas con memoria distribuida.
Principios de la comunicación/sincronización por pasaje de mensajes.
Primitivas Send y Receive. La interfaz MPI como modelo.
Cómputo y Comunicaciones.
Comunicaciones colectivas y operaciones de procesamiento.
Ejemplos sobre arquitecturas multiprocesador.
Combinación de memoria compartida y pasaje de mensajes. Modelo híbrido.

Unidad 8: Algoritmos paralelos clásicos.
Presentación de casos clásicos:
Sorting / Algoritmos sobre grafos /Procesamiento de matrices.
Algoritmos de búsqueda para optimización discreta.
Programación dinámica.
Análisis de soluciones sobre diferentes arquitecturas paralelas.

Unidad 9: Tendencias en procesamiento paralelo
Conceptos de Green computing.
Nuevas arquitecturas multiprocesador (GPUs, Xeon Phi, FPGAs) y sus modelos de programación.
Cloud computing para HPC. Lenguajes emergentes. Nuevas tecnologías de memoria.

BIBLIOGRAFÏA

Bibliografía básica

- “Introduction to Parallel Computing”. Grama, Gupta, Karypis, Kumar. Addison Wesley 2003
- “Parallel Programming”. Wilkinson, Allen. Prentice Hall 2005.
- “Sourcebook of Parallel Computing”. Dongarra, Foster, Fox, Gropp, Kennedy, Torczon, White. Morgan Kauffman 2003.
- “Introduction to High Performance Computing for Scientists and Engineers”. Georg Hager, Gerard Wellein. CRC Press (2011).
- “Parallel Programmng for Multicore and Cluster Systems”. Thomas Rauber, Gudulla Runger. Springer (2010).
- “An introduction to parallel programming”. Peter Pacheco. Elsevier (2011).

Bibliografía Complementaria

- “Using MPI-2”. Gropp, Lusk, Thakur. The MIT Press (1999).
- “Foundations of Multithreaded, Parallel and Distributed Programming”. Andrews. Addison Wesley 2000.
- “Programming Massively Parallel Processors”. Kirk, Hwu. Elsevier 2010.
- “Computer Architecture. A quantitative approach”. Henessy, Patterson.6ta edición. Elsevier (2017).
- “Using OpenMP. Poratble Shared Memory Parallell Programming”. Chapman, Jost, Van der Pas.MIT Press (2008).
- “Parallel Programming in OpenMP”. Chandra, Dagum, Kohr, Maydan, McDonald,Menon. Morgan Kauffman (2011).
- “Parallel Programming: Concepts and practice”. Schmidt, Bertil; Gonzalez-Domínguez, Jorge; Hundt, Christian ; Scharb, Moritz.. Morgan Kaufmann, 2018.
- IEEE, ACM Digital Library

METODOLOGÍA DE ENSEÑANZA

En la cátedra se pone énfasis en el proceso de identificación de problemas del mundo real (especialmente aquellos que presentan alta demanda computacional), en la especificación de los mismos como problemas resolubles desde la informática y en el desarrollo de las correspondientes soluciones verificables.
También se busca poner al alumno en el contexto de *aplicación* en el campo de la Informática, de los conceptos y métodos matemáticos que se enseñan en el programa de la asignatura. Esta contextualización es informativa y se discuten diferentes casos de aplicación para mostrar la utilidad de las teorías y herramientas matemáticas para resolver diferentes problemas "informáticos" conocidos por el alumno.
Adicionalmente, se trabaja en la capacidad del alumno para conocer técnicas y herramientas de aplicación en Informática (en lo posible siguiendo las tendencias marcadas por el cambio tecnológico) y en la aplicación efectiva de las mismas.
La cátedra acompaña el proceso con materiales para que el alumno estudie casos y valore la selección y empleo eficiente de herramientas y técnicas determinadas para cada problema.
Dentro de las actividades planificadas para los alumnos, se propone el estudio de la tecnología existente y prevista para un tipo de problema y se los "desafía" a presentar la posible evolución de la solución para ese tipo de problema y a analizar en qué podría mejorar al estado del arte. Para ello, el alumno debe buscar bibliografía relacionada con el cambio tecnológico y formar un criterio sobre las tendencias (por ejemplo, en los procesadores a utilizar, el tipo de topología de red o los leguajes y librerías para programación paralela).

La asignatura se estructura con clases teóricas, explicaciones de práctica y prácticas experimentales:

- Las clases teóricas son dictadas por el profesor de la asignatura y no son obligatorias.
- Las explicaciones de práctica son introductorias al trabajo en laboratorio, para facilitar la utilización del equipamiento y software por los alumnos. Si bien no son obligatorias, es recomendado que los alumnos asistan.
- Las clases prácticas consisten en el desarrollo de trabajos con diferentes arquitecturas paralelas y lenguajes de programación. De acuerdo con la cantidad de alumnos que cursen la materia, los docentes definen si los trabajos se realizan individualmente o en grupo.

Los trabajos realizados en Laboratorio son sobre diferentes arquitecturas paralelas que dispone la Facultad. Al igual que en años anteriores, se brindará acceso remoto a estos equipos.

Para la comunicación se utilizará el entorno virtual de enseñanza-aprendizaje (IDEAS), donde estarán disponibles material de clases, trabajos prácticos, avisos, resultados de exámenes, etc. En ese sentido, se hará disponible el material audiovisual generado para las cursadas virtuales durante la pandemia covid-19.

EVALUACIÓN

La cátedra acompaña el proceso del alumno, para contrastar las conclusiones del alumno y validar su habilidad para interpretar la evolución tecnológica y su visión de las tendencias futuras.
En la evaluación de las competencias en las pruebas parciales y finales se tienen en cuenta los siguientes aspectos:
- la capacidad para identificar, formular y resolver los problemas, reflejándolo en la corrección de las pruebas escritas y devoluciones a coloquios.
- la capacidad para conocer e interpretar los conceptos, teorías y métodos, aplicándolos a problemas concretos. Esto se evalúa a través de preguntas del tipo "donde cree Ud. que es aplicable este conocimiento o método", o la interpretación de código, o interpretación de resultados.
- en qué medida el alumno es capaz de utilizar de manera efectiva las técnicas y herramientas que son parte de la asignatura.
- los resultados del estudio bibliográfico y capacidad para formular las ventajas potenciales del cambio tecnológico en los problemas planteados, plasmando esto en una planilla.

Los alumnos deben aprobar los diferentes trabajos experimentales a realizar, con sus respectivos informes y coloquios. Cada trabajo cuenta con más de una fecha de defensa, donde se evalúa el desarrollo de los alumnos (asentado en una planilla) y se les realiza una devolución sobre el mismo (en el caso que corresponde, se explican los errores que se deben corregir).
La aprobación de estos trabajos experimentales otorga la aprobación de la cursada de la asignatura.

La aprobación final de la materia puede ser mediante:

- examen teórico/práctico en la mesa de final en que se inscriba el alumno.
- examen conceptual (al terminar la cursada) más (posible) proyecto específico que el alumno resuelve, presenta una monografía sobre el tema y defiende en un coloquio en una fecha de examen final.
- proyecto de desarrollo experimental, que involucra análisis de rendimiento de un sistema paralelo para un determinado problema a resolver.

CRONOGRAMA DE CLASES Y EVALUACIONES

El esquema de las clases teóricas es el siguiente:

Clase	Fecha	Contenidos/Actividades
1	Semana del 4/03	Introducción al Procesamiento Paralelo. Conceptos. Fundamentos Sistemas Paralelos. Aplicaciones. Sistemas Distribuidos y Paralelos.
2	Semana del 11/03	Plataformas de procesamiento para Sistemas Paralelos. Evolución de las arquitecturas y las comunicaciones. Relación con los Sistemas Distribuidos.
3	Semana del 18/03	Conceptos de programación distribuida y paralela. Aplicaciones de los sistemas paralelos. Programación en Memoria Compartida, modelos basados en threads (Pthreads).
4	Semana del 01/04	Programación en Memoria Compartida, modelos de programación basados en directivas – OpenMP.
5	Semana del 08/04	Principios de diseño de Algoritmos Paralelos. Descomposición en tareas, granularidad de aplicaciones, mapeo de tareas. Técnicas de descomposición de aplicaciones. Características de las tareas y las interacciones generadas.
6	Semana del 15/04	Programación de Algoritmos Paralelos utilizando el paradigma de Pasaje de Mensajes. MPI.
7	Semana del 22/04	Programación en arquitecturas híbridas. Modelo híbrido. Métricas de rendimiento. Granularidad. Escalabilidad. Grado de concurrencia.
8	Semana del 29/04	Tendencias en paralelismo: Cloud Computing; Arquitecturas many-core (Xeon Phi's y GPUs); FPGAs; Eficiencia Energética; Lenguajes y librerías de programación emergentes.
9	Semana del 24/06	Casos de estudio

Evaluaciones previstas	Fecha
Coloquio trabajo optimización secuencial	Semana del 08/04
Coloquio trabajo OpenMP y Pthreads	Semana del 06/05
Recuperatorio de coloquios de trabajos previos (optimización secuencial / OpenMP y Pthreads)	Semana del 20/5

Coloquio trabajo MPI e híbrido	Semana del 10/06
Recuperatorio de coloquio de trabajos previos (MPI e híbirdo)	Semana del 01/07

Contacto de la cátedra (mail, sitio WEB, plataforma virtual de gestión de cursos):

E-Mail: sparalelos@lidi.info.unlp.edu.ar
Web: http://weblidi.info.unlp.edu.ar/catedras/paralela/
Plataforma virtual: ideas.info.unlp.edu.ar

Firma del/los profesor/es