RESOLUCIÓN 25/18/126-00
ACTA 1226/25/08/2025

"POR LA CUAL SE APRUEBA EL PROGRAMA DE ESTUDIO DE LA ASIGNATURA *PROGRAMACIÓN PARALELA*, DE LA CARRERA INGENIERÍA EN INFORMÁTICA – PLAN 2026, SEDE SAN LORENZO"

VISTO: El Memorando DA/1707/2025 del Director Académico de la FP-UNA, Prof. MSc. Felipe Santiago Uzabal Escurra, con el cual remite el Memorando CCPTCC/030/2025 de la Comisión Coordinadora del Proyecto de Transformación Curricular de Carreras de Grado de la FP-UNA, en el que presenta la propuesta de Programas de Estudio de las Asignaturas de la Carrera Ingeniería en Informática.

CONSIDERANDO: La Ley 4995/2013 de Educación Superior, el Estatuto de la Universidad Nacional de Asunción y las deliberaciones sobre el tema.

Que la Comisión Coordinadora del Proyecto de Transformación Curricular de Carreras de Grado, solicita la aprobación del Programa de Estudio de la asignatura **"Programación Paralela"**, de la carrera Ingeniería en Informática – Plan 2026.

Que los programas fueron elaborados conforme a las disposiciones establecidas por el Consejo Nacional de Educación Superior (CONES) en materia de **créditos académicos**, según lo dispuesto en la Resolución CONES N.° 221/2024, que regula el *Sistema de Créditos Académicos – Paraguay* y los criterios para su publicación en las carreras de grado.

EL CONSEJO DIRECTIVO DE LA FACULTAD POLITÉCNICA
RESUELVE:

25/18/126-01 APROBAR el Programa de Estudio de la Asignatura **"Programación Paralela"** de la carrera Ingeniería en Informática – Plan 2026, Sede San Lorenzo, detallado en el ANEXO 88 de la presente Acta.

25/18/126-02 COMUNICAR, copiar y archivar.

Prof. Abg. Joel Arsenio Benítez Santacruz
Secretario

Prof. Ing. Silvia Teresa Leiva León, MSc.
Presidenta

TEL/FAX: +595 21 5887000 – C.C. 1130 (Asunción) – 2111 (San Lorenzo)
www.pol.una.py

DEPARTAMENTO DE ENSEÑANZA DE INFORMÁTICA
PROGRAMA DE ESTUDIO

I. IDENTIFICACIÓN

Nivel	Grado					
Asignatura	Programación Paralela					
Carrera	**Plan**	**Sede/Filial**	**Carácter**	**Semestre**	**Prerrequisitos**	
Ingeniería en Informática	2026	Sede San Lorenzo	Obligatoria	Noveno	Redes de Computadoras, Estructura de Lenguajes de Programación.	

Semanal					**Periodo**				
HT	**HP**	**HTD**	**HTI**	**HS**	**PA**	**THTD**	**THTI**	**THA**	**CA-PY**
2	2	4	4	8	18	72	72	144	5

***HT: Horas Teóricas semanales.**
***HP: Horas Prácticas semanales.**
***HTD: Horas semanales de Trabajo académico con acompañamiento Docente.**
***HTI: Horas semanales de Trabajo académico Independiente del estudiante.**
***HS: Horas Semanales (HTD+HTI).**
***PA: Periodo Académico en semanas.**
*** THTD: Total de Horas de Trabajo académico con acompañamiento Docente (HTD*PA).**
*** THTI: Total de Horas de Trabajo académico Independiente del estudiante (HTI*PA).**
*** THA: Total de Horas de trabajo Académico (THTD+THTI).**
*** CA-PY: Créditos académicos de la asignatura.**

II. FUNDAMENTACIÓN

La asignatura Programación Paralela se incluye en la malla curricular de la carrera para responder a la creciente demanda de profesionales capaces de diseñar, analizar e implementar soluciones computacionales que aprovechen eficientemente los recursos de hardware modernos. En un contexto donde la cantidad de datos a procesar es masiva y las necesidades de cálculo son intensivas, es imprescindible formar ingenieros con competencias sólidas en paradigmas de programación paralela, capaces de realizar cómputos utilizando las capacidades de procesamiento paralelo de diversos tipos de procesadores actuales.

Esta asignatura contribuye al desarrollo del perfil profesional del ingeniero en informática, permitiéndole adquirir competencias clave para:

- Diseñar e implementar algoritmos eficientes que se beneficien de arquitecturas de hardware paralelas y distribuidas.
- Analizar y resolver problemas complejos utilizando técnicas de programación concurrente y distribuida.
- Evaluar críticamente las ventajas, desafíos y sobrecostes asociados a diferentes paradigmas y modelos de programación paralela.
- Adaptar algoritmos a arquitecturas de hardware específicas, optimizando el rendimiento y gestionando eficientemente el equilibrio de carga.
- Utilizar librerías y herramientas de programación paralela para el desarrollo de aplicaciones modernas.

De naturaleza teórico-práctica, esta asignatura busca equilibrar el entendimiento conceptual con la aplicación práctica, proporcionando a los estudiantes la oportunidad de desarrollar experiencias directas mediante la implementación de algoritmos y el uso de plataformas actuales para la programación paralela y distribuida.

La estructura del curso se organiza en torno a cinco unidades principales, las cuales se alinean con los siguientes ejes temáticos:

- Concurrencia, paralelismo y sistemas distribuidos: Fundamentado en las Unidades 1 y 4, se introducen los conceptos clave del paralelismo y la programación en sistemas distribuidos.
- Limitaciones del paralelismo, incluidos los sobrecostes: Abordado en las Unidades 2, 3 y 4, se analizan los desafíos y estrategias para mitigar los sobrecostes asociados a la comunicación y sincronización en sistemas paralelos.
- Algoritmos paralelos y su adaptación a arquitecturas específicas: Desarrollado en la Unidad 5, enfocándose en la optimización de algoritmos y la gestión del equilibrio de carga.
- Paradigmas típicos de programación paralela: Presentado en las Unidades 3 y 5, con énfasis en técnicas como MapReduce y su aplicación práctica.
- Complejidad de los algoritmos paralelos y concurrentes: Cubierto en la Unidad 5, enseñando a los estudiantes a analizar y optimizar algoritmos para entornos paralelos.

III. COMPETENCIAS DEL PERFIL DE EGRESO ASOCIADAS

1. Aplicar en la práctica profesional los valores humanos, la ética y los mecanismos de seguridad laboral.
2. Actuar proactivamente frente a los problemas sociales y ambientales.
3. Adaptarse respetuosamente a contextos nuevos o adversos, así como a diversidades personales, disciplinares y culturales.
4. Actualizarse permanentemente mediante la obtención y gestión autónoma de información de calidad, utilizando tecnología de la información y comunicación.
5. Evaluar el comportamiento de diversos fenómenos disciplinares e interdisciplinares relacionados con la ingeniería informática con una visión de sistema, mediante modelos teóricos validados y actualizados, capaces de abarcarlos integralmente en un contexto de incertidumbre.
6. Aplicar, producir y difundir conocimientos técnicos y científicos en el área de la ingeniería informática.
7. Planificar, proyectar, diseñar y ejecutar proyectos sostenibles e integrales para la resolución de problemas, la mejora y la innovación en el ámbito de la ingeniería informática.
8. Interpretar, modelar y comunicar información referida a la ingeniería informática en forma gráfica.

IV. ORGANIZACIÓN DE LA ASIGNATURA

Unidades	Contenidos	Resultados de Aprendizaje
1. Introducción a la programación paralela.	1.1. Definición y motivación. 1.2. Historia y evolución. 1.3. Conceptos clave. 1.4. Beneficios y desafíos.	1 Explica los fundamentos del paralelismo en la informática moderna, destacando su importancia y aplicaciones. 2 Describe la necesidad y las ventajas del procesamiento

Unidades	Contenidos	Resultados de Aprendizaje
		paralelo en diferentes contextos computacionales. 3 Identifica y analiza la evolución histórica y tecnológica del paralelismo, vinculándola con las tendencias actuales. 4 Distingue entre términos clave como concurrencia, paralelismo y distribución, explicando sus diferencias y aplicaciones. 5 Evalúa los desafíos y oportunidades en la programación paralela, proponiendo enfoques para abordar las limitaciones.
2. Arquitecturas paralelas y redes.	2.1 Arquitecturas y tecnologías en computación paralela y distribuida. 2.2 Clasificación de sistemas basados en el tipo de paralelismo y ejecución. 2.3 Ventajas y desafíos de los procesadores multi-core. 2.4 Paradigmas de ejecución SIMD y MIMD y sus aplicaciones. 2.5 Visión general de las GPUs en el cómputo paralelo. 2.6 Principales topologías y características de las redes de computadoras. 2.7 Sistemas de computación de alto rendimiento.	1 Describe las arquitecturas y tecnologías que permiten la computación paralela y distribuida, explicando sus características principales. 2 Clasifica sistemas computacionales según el tipo de paralelismo y ejecución, destacando sus diferencias y aplicaciones. 3 Analiza las ventajas y desafíos asociados con los procesadores multi-core en entornos modernos. 4 Explica los paradigmas de ejecución SIMD y MIMD, ilustrando sus aplicaciones prácticas en diferentes contextos. 5 Identifica el papel de las GPUs en el cómputo paralelo, describiendo sus principales características y beneficios. 6 Distingue las principales topologías y características de las redes de computadoras, evaluando su impacto en la computación distribuida. 7 Describe la importancia y la estructura de los sistemas de computación de alto rendimiento, destacando su rol en el procesamiento intensivo de datos.
3. Programación con memoria compartida y GPU.	3.1 Concepto de memoria compartida. 3.2 Threads. 3.3 Mecanismos de sincronización. 3.4 Problemas clásicos de sincronización. 3.5 Herramientas y bibliotecas. 3.6 Introducción a la programación en GPU.	
4. Programación con memoria distribuida.	4.1 Concepto de memoria distribuida. 4.2 Modelos de programación	1. Desarrolla habilidades prácticas en programación distribuida utilizando el paso de mensajes

Unidades	Contenidos	Resultados de Aprendizaje
	distribuida. 4.3 Comunicación punto a punto. 4.4 Introducción a MPI. 4.5 Desafíos.	como modelo principal. 2. Distingue las características clave entre la programación con memoria distribuida y memoria compartida, evaluando sus ventajas y limitaciones. 3. Diseña programas que implementen los modelos SPMD (Single ProgramMultiple Data) y MPMD (MultipleProgramMultiple Data). 4. Implementa programas que empleen comunicaciones punto a punto y colectivas, garantizando eficiencia y precisión. 5. Utiliza MPI (MessagePassing Interface) para desarrollar soluciones de procesamiento distribuido en aplicaciones prácticas. 6. Identifica y resuelve desafíos comunes en la programación distribuida, optimizando el rendimiento y la robustez de los sistemas.
5. Diseño de algoritmos paralelos.	5.1 Principios básicos. 5.2 Técnicas de descomposición. 5.3 Patrones de programación paralela. 5.4 Balanceo de carga. 5.5 Casos prácticos.	1 Diseña y optimiza algoritmos que aprovechen el procesamiento paralelo para mejorar el rendimiento en diversos contextos. 2 Aplica principios fundamentales para la creación de algoritmos paralelos, asegurando su eficiencia y escalabilidad. 3 Utiliza técnicas de descomposición para desarrollar algoritmos adaptados al paralelismo, dividiendo problemas complejos en subproblemas manejables. 4 Implementa patrones comunes de programación paralela, como MapReduce y Divide-and-Conquer, en escenarios prácticos. 5 Optimiza programas paralelos mediante técnicas de balanceo de carga, maximizando el uso eficiente de los recursos disponibles.

Unidades	Contenidos	Resultados de Aprendizaje
		6 Analiza y resuelve problemas complejos utilizando algoritmos paralelos, evaluando su desempeño y adecuación al hardware.

V. ESTRATEGIAS DIDÁCTICAS

En el desarrollo del curso se aplicarán estrategias didácticas diseñadas para combinar la comprensión teórica con la práctica intensiva, fomentando el aprendizaje activo, el trabajo en equipo y la resolución de problemas complejos. Estas estrategias incluyen:

- **Clases magistrales:** El docente presentará los fundamentos de la programación paralela, utilizando materiales como diapositivas, ejemplos prácticos y demostraciones en vivo. Se incluirán explicaciones detalladas sobre arquitecturas de hardware, técnicas de sincronización y paradigmas de programación paralela. Las clases serán interactivas, promoviendo preguntas y debates para clarificar conceptos clave.

- **Aula invertida:** Los estudiantes tendrán acceso a materiales previos, como videos, tutoriales y lecturas sobre herramientas como OpenMP, MPI y CUDA. Esto permitirá que las sesiones presenciales se centren en la resolución de problemas prácticos, la discusión de casos y la optimización de algoritmos paralelos.

- **Aprendizaje Basado en Problemas (ABP):** Los estudiantes resolverán problemas reales que requieran diseñar e implementar soluciones utilizando programación paralela y distribuida. Esto incluirá la creación de algoritmos optimizados para hardware específico y la resolución de desafíos relacionados con la sincronización y el balanceo de carga.

- **Trabajos colaborativos:** Los estudiantes trabajarán en equipos para desarrollar proyectos complejos que integren múltiples aspectos de la programación paralela, como el uso de memoria compartida, sistemas distribuidos y algoritmos paralelos. Se fomentará la asignación de roles y el uso de herramientas colaborativas para la gestión de proyectos.

- **Talleres prácticos:** Las sesiones de laboratorio estarán dedicadas a la implementación de programas paralelos y distribuidos. Los estudiantes trabajarán con herramientas y bibliotecas como OpenMP, pthreads y MPI, abordando tareas específicas bajo la supervisión y guía del docente.

- **Simulaciones y análisis crítico:** Los estudiantes realizarán simulaciones para analizar el rendimiento de algoritmos paralelos en diferentes entornos y arquitecturas. Esto incluirá la evaluación de factores como el balanceo de carga, los sobrecostes y la escalabilidad. También asumirán roles de evaluadores para comparar enfoques de programación paralela en distintos contextos.

- **Proyecto Final Integrador:** Los estudiantes diseñarán, implementarán y presentarán un proyecto que combine herramientas y técnicas de programación paralela, justificando sus decisiones técnicas. Este proyecto reflejará la aplicación práctica de los conceptos aprendidos durante el curso.

La elección particular de la estrategia didáctica aplicada será explícita en el Planeamiento de la Asignatura, de acuerdo con el perfil de los estudiantes, los recursos disponibles y el contexto educativo.

VI. ESTRATEGIAS EVALUATIVAS

Procesos de producción grupales e individuales: Desarrollo de proyectos prácticos que incluyan el diseño, implementación y optimización de algoritmos paralelos utilizando memoria compartida y distribuida. Pruebas individuales orales y/o escritas: Evaluaciones conceptuales y de resolución de problemas para medir el conocimiento teórico sobre concurrencia, paralelismo, sincronización y arquitecturas paralelas. Diseño y programación de soluciones que utilicen herramientas y bibliotecas como OpenMP, MPI y CUDA, resolviendo problemas computacionales específicos. Elaboración de informes técnicos donde se evalúen ventajas, desafíos y limitaciones de diferentes paradigmas y modelos de programación paralela en contextos reales. Actividades prácticas que incluyan la simulación de entornos paralelos, resolución de problemas clásicos de sincronización y experimentación con patrones de programación como MapReduce o Divide-and-Conquer. Desarrollo de un proyecto integrador que aborde la implementación y optimización de un sistema paralelo o distribuido, acompañado de un informe técnico que analice el rendimiento, los desafíos enfrentados y las soluciones adoptadas.

Con fines de calificación y promoción se aplicará el Reglamento Académico vigente en la institución que prevé valoraciones de proceso y final.

VII. MEDIOS AUXILIARES

Aula virtual, pizarrón, proyector, marcadores, equipo de audio, wifi, plataformas para videoconferencias, aplicaciones, software, etc.

VIII. BIBLIOGRAFÍA

- Pacheco, P. (2021). AnIntroduction to ParallelProgramming. 2nd Edition. Elsevier.
- Grama, A., Gupta, A., Karypis, G., &Kumar, V. (2003). Introduction to Parallel Computing. Addison Wesley.
- OpenMPArchitectureReviewBoard. (s.f.). Tutorials and Articles. https://www.openmp.org/resources/tutorials-articles/
- NVIDIA. (s.f.). Developer Blog. https://developer.nvidia.com/blog/
- The Open Group. (s.f.). POSIX ThreadsProgramming. Recuperado [19 de octubre 2023]
- https://hpc-tutorials.llnl.gov/posix/
- Barney, B. (2009). POSIX threadsprogramming. Recuperado [19 de octubre de 2023]. Lawrence LivermoreNationalLaboratory, https://computing.llnl.gov/tutorials/pthreads/
- Barney, B. (2018). MessagePassing Interface (MPI)(Tutorial). Recuperado [19 de octubre de 2023]. Lawrence LivermoreNationalLaboratory, https://computing.llnl.gov/tutorials/mpi
- Hwu, W. M. W. (Ed.). (2019). GPU Computing Gems Jade Edition. Elsevier.
- Matloff, N. (2014). ProgrammingonParallel Machines; GPU, Multicore, Clusters and More. https://heather.cs.ucdavis.edu/~matloff/158/PLN/ParProcBook.pdf
- Dean, J., &Ghemawat, S. (2008). MapReduce: simplified data processingonlargeclusters. Communications of the ACM, 51(1), 107-113.
- Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., ... y Sen, K. (2009). A view of theparallelcomputinglandscape. Communications of the ACM, 52(10), 56-67.
- Lee, V. W., Kim, C.,Chhugani, J., Deisher, M., Kim, D., ... &Dubey, P. (2010). Debunkingthe 100X GPU vs. CPU myth: anevaluation of throughputcomputingon CPU and GPU. ACM SIGARCH ComputerArchitecture News, 38(3), 451-460