Universidad del Desarrollo
Universidad de Excelencia

Programa de Asignatura
Computación Paralela y Distribuida

A. Antecedentes Generales

1. **Unidad Académica**	Facultad de Ingeniería							
2. **Carrera**	Ingeniería Civil Informática e Innovación Tecnológica							
3. **Código**	IIP412W							
4. **Ubicación en la malla**	IV año, I trimestre							
5. **Créditos**	UDD	8	SCT	5				
6. **Tipo de asignatura**	Obligatorio	X	Electivo		Optativo			
7. **Duración**	Bimestral		Semestral		Anual		Otro	X
8. **Módulos semanales**	Clases Teóricas	2	Clases Prácticas	1	Ayudantía			
9. **Horas académicas**	Clases	48	Ayudantía	24	Otras horas por periodo completo			
10. Pre-requisito	Arquitectura de Sistemas Redes de Computadores							

B. Aporte al Perfil de Egreso

Al concluir la asignatura, el estudiante será capaz de diseñar, desarrollar y optimizar programas utilizando paradigmas de computación paralela y distribuida, integrando técnicas específicas para arquitecturas de GPU y entornos distribuidos, con el fin de mejorar el rendimiento y la eficiencia en el procesamiento de datos complejos.

El estudiante demostrará competencias para:

- Comprender los fundamentos teóricos y arquitectónicos de la computación paralela y distribuida, incluyendo modelos de memoria, sincronización y comunicación entre procesos.
- Programar aplicaciones paralelas utilizando herramientas y librerías específicas, como OpenMP, MPI y CUDA, para arquitecturas CPU y GPU.
- Diseñar y optimizar algoritmos paralelos, identificando oportunidades de paralelismo y gestionando correctamente los recursos para maximizar el rendimiento y minimizar la latencia.
- Implementar soluciones distribuidas que aprovechen la escalabilidad y tolerancia a fallos, aplicando técnicas de particionamiento, balanceo de carga y comunicación eficiente entre nodos.
- Evaluar el desempeño de programas paralelos y distribuidos, mediante métricas de velocidad, escalabilidad y eficiencia, y realizar ajustes para mejorar la utilización del hardware.

- Integrar conocimientos de programación paralela y distribuida en proyectos reales, facilitando la solución de problemas computacionales de alta demanda en áreas como procesamiento científico, análisis de datos y simulaciones.

Este perfil prepara al estudiante para afrontar retos en ambientes de cómputo intensivo, optimizando recursos de hardware modernos, incluyendo GPUs, para desarrollar aplicaciones de alto rendimiento y escalabilidad.

C. Competencias y Resultados de Aprendizaje Generales que desarrolla la asignatura

<table>
<tr><th>Competencias Genéricas</th><th>Resultados de Aprendizaje Generales</th></tr>
<tr><td>Pensamiento Critico
Transformación Digital
Comunicación</td><td rowspan="3">RAG1: Analiza los principios y modelos fundamentales de la computación paralela y distribuida, diferenciando arquitecturas, modelos de memoria y mecanismos de sincronización.

RAG2: Diseña y construye programas que aprovechan eficientemente los recursos de hardware, aplicando técnicas de paralelización para arquitecturas CPU y GPU.

RAG3: Implementa soluciones escalables utilizando librerías especializadas (como OpenMP, MPI y CUDA/CuPy) para resolver problemas computacionales intensivos.

RAG4: Evalúa y optimiza el rendimiento de aplicaciones, justificando sus decisiones algorítmicas mediante el análisis de métricas de eficiencia, escalabilidad y profiling de recursos.</td></tr>
<tr><th>Competencias Específicas</th></tr>
<tr><td>Resolución de problemas bajo un enfoque sistémico
Trabajo en Equipo
Aprendizaje Continuo</td></tr>
</table>

D. Unidades de Contenidos y Resultados de Aprendizaje

Unidades de Contenidos	Competencia	Resultados de Aprendizaje
UNIDAD I: Fundamentos de Computación Paralela • Introducción a la computación paralela y distribuida • Motivación: problemas que requieren paralelismo • Caso de estudio: Password cracking y seguridad de contraseñas • Funciones hash criptográficas • Paradigmas de programación paralela • Tipos de paralelismo • Ley de Moore y el fin del "free lunch" • Arquitecturas paralelas • Modelos de memoria	*Pensamiento Crítico* *Aprendizaje Continuo*	**RA1**: Analiza los principios de la computación paralela y las arquitecturas de hardware modernas, identificando qué clases de problemas computacionales se benefician de la paralelización para evitar cuellos de botella. **RA2**: Compara los diferentes modelos teóricos de memoria (compartida vs. distribuida) y los paradigmas de programación concurrente, justificando analíticamente la elección de un modelo frente a un problema algorítmico específico.
UNIDAD II: Programación Paralela con OpenMP • Modelo de programación fork-join • Directivas básicas de OpenMP • Variables privadas vs compartidas • Cláusulas de alcance • Sincronización y control de concurrencia • Reducción de datos (reduction) • Scheduling de iteraciones • Granularidad y balance de carga	*Pensamiento Critico* *Transformación Digital* *Resolución de Problemas Bajo un Enfoque sistémico*	**RA1**: Implementa algoritmos paralelos de memoria compartida utilizando OpenMP, aplicando técnicas de sincronización y balance de carga que maximicen el rendimiento. **RA2**: Diagnostica y corrige anomalías de concurrencia (como deadlocks o race conditions) en código paralelo, gestionando de manera rigurosa el alcance de las variables (privadas vs. compartidas) y las dependencias de datos.

• Detección y prevención de condiciones de carrera (race conditions) • Análisis de dependencias de datos • Patrones de paralelización		
UNIDAD III: Programación Distribuida • Modelo de paso de mensajes • Conceptos básicos: procesos, ranks, comunicadores • Inicialización y finalización • Identificación de procesos • Comunicación punto a punto • Comunicaciones colectivas • Topologías de comunicación • Patrones de diseño distribuido • Balanceo de carga dinámico en entornos distribuidos	*Pensamiento Crítico* *Transformación Digital* *Resolución de Problemas Bajo un Enfoque Sistémico* *Trabajo en Equipo*	**RA1**: Diseña soluciones distribuidas basadas en el paso de mensajes utilizando MPI, orquestando la comunicación eficiente y el balanceo dinámico de carga en entornos multi-nodo. **RA2**: Desarrolla topologías de comunicación colectiva y punto a punto, minimizando la latencia, la sobrecarga de mensajes y los tiempos de inactividad entre los nodos de procesamiento.
UNIDAD IV: Evaluación y rendimiento • Métricas. • Ley de Amdahl y Ley de Gustafson. • Overhead de paralelización • Análisis de cuellos de botella • Herramientas de profiling y benchmarking • Metodología científica de benchmarking • Técnicas de optimización • Documentación y reporte de resultados de rendimiento	*Pensamiento Crítico* *Comunicación* *Aprendizaje Continuo*	**RA1**: Evalúa el rendimiento de aplicaciones de alto desempeño mediante el uso de herramientas de benchmarking y profiling, aplicando metodologías científicas para aislar las sobrecargas de paralelización (overhead). **RA2**: Argumenta cuantitativamente las mejoras logradas a través de reportes técnicos, aplicando la Ley de Amdahl y la Ley de Gustafson para calcular y justificar los límites teóricos de aceleración (speedup) de un sistema.

		RA3: Diagnostica cuellos de botella a nivel de procesador, memoria o red mediante el análisis detallado de trazas de ejecución, proponiendo refactorizaciones algorítmicas directas para mitigar la latencia. **RA4**: Compara de forma crítica múltiples estrategias de paralelización (ej. escalabilidad fuerte vs. débil) aplicadas a un mismo problema, defendiendo la solución más costo-eficiente en función de las restricciones del hardware.
Unidad V: Programación GPU • Arquitectura de GPU vs CPU • Modelo de programación CUDA • Jerarquía de memoria en CUDA • Transferencia de datos entre host y device • Programación de kernels • CUDA Python y CuPy • Optimización básica de kernel • Aplicaciones de CUDA en seguridad	*Pensamiento Crítico* *Transformación Digital* *Resolución de problemas Bajo un Enfoque Sistémico* *Trabajo en Equipo.*	**RA1**: Construye soluciones de paralelismo masivo utilizando arquitecturas GPU y la plataforma CUDA (Python/CuPy), mapeando problemas computacionalmente intensivos a grillas y bloques de hilos (threads). **RA2**: Orquesta la transferencia asíncrona de datos entre el host (CPU) y el device (GPU) para ocultar latencias de memoria, logrando el máximo aprovechamiento del bus de datos (PCIe). **RA3**: Diseña y optimiza kernels personalizados (custom kernels), administrando explícitamente la jerarquía de memoria de la GPU (global, compartida, constante y registros) para maximizar el throughput y evitar accesos redundantes.

		RA4: Integra paradigmas de computación heterogénea (CPU + GPU) para resolver un caso de estudio aplicado de alta complejidad (como password cracking o simulaciones), balanceando dinámicamente la carga de trabajo entre los distintos aceleradores del sistema.

E. Estrategias de Enseñanza

El curso será abordado mediante variadas estrategias metodológicas, cada una de ellas formulada sobre la base de los resultados de aprendizaje que se desea transferir y desarrollar en el estudiante, las cuales son:

1. Clases expositivas.
2. Talleres en sala.

F. Estrategias de Evaluación

Para las diferentes instancias evaluativas se contará con criterios claros y conocidos por los estudiantes, el curso será de carácter grupal donde se ejecutarán dos entregas parciales de un proyecto trimestral y una presentación final a ejecutarse el día del examen.

1. Rúbricas.
2. Aprendizaje basado en problemas (ABP)
3. Certámenes.
4. Exámenes.

G. Recursos de Aprendizaje

Bibliografía Obligatoria:

- Quinn, Michael J. Parallel Programming in C with MPI and OpenMP
- Sanders, Jason, y Kandrot, Edward., CUDA by Example: An Introduction to General-Purpose GPU Programming
- NVIDIA CUDA Toolkit Documentation, https://docs.nvidia.com/cuda/
- OpenMP API Specification, https://www.openmp.org/specifications/
- MPI Tutorial (LLNL), https://hpc-tutorials.llnl.gov/mpi/
- Cupy documentation, https://docs.cupy.dev/en/stable/
- Cuda Python, https://nvidia.github.io/cuda-python/latest/