



National University of Engineering (UNI)
School of Computer Science
Syllabus 2026-I

1. COURSE

CS3P1. Parallel and Distributed Computing (Mandatory)

2. GENERAL INFORMATION

- | | | |
|-----------------------------------|---|--|
| 2.1 Course | : | CS3P1. Parallel and Distributed Computing |
| 2.2 Semester | : | 8 th Semester. |
| 2.3 Credits | : | 4 |
| 2.4 Hours | : | 2 HT; 4 HP; |
| 2.5 Duration of the period | : | 16 weeks |
| 2.6 Type of course | : | Mandatory |
| 2.7 Learning modality | : | Face to face |
| 2.8 Prerequisites | : | <ul style="list-style-type: none">• CS212. Analysis and Design of Algorithms. (5th Sem)• CS231. Networking and Communication. (6th Sem) |

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

With the end of frequency scaling in single-core processors, parallel and distributed computing has become essential for high-performance software development. This course introduces students to the programming models, communication patterns, and coordination mechanisms required to exploit multi-core systems and distributed clusters. Students will learn to design scalable algorithms and evaluate their performance under different computational constraints.

5. GOALS

- Design and implement parallel programs using shared and distributed memory.
- Master communication and synchronization protocols in distributed systems.
- Evaluate performance and scalability through formal metrics.
- Apply parallel algorithms to solve complex computational problems.

6. COMPETENCES

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Programas: Paralelismo Declarativo (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Paralelismo – Paralelismo declarativo: Determinar qué acciones pueden, o no deben, realizarse en paralelo, a nivel de instrucciones, funciones, clausuras, acciones compuestas, sesiones, tareas y servicios es la idea principal que subyace a los algoritmos PDC; no hacerlo es la principal fuente de errores. Ver también: Computación Paralela y Distribuida (PDC)-Algorithms – Definir orden: por ejemplo, usando relaciones de "sucede antes" o grafos acíclicos dirigidos serie/paralelo que representan programas. – Independencia: determinar cuándo el orden no importa, en términos de conmutatividad, dependencias, precondiciones. – Garantizar el orden entre acciones que de otro modo serían paralelas cuando sea necesario, incluyendo bloqueos, publicación segura; e imponer comunicación - enviar un mensaje sucede antes de recibirlo; y relajarlo cuando no sea necesario.	• Mostrar gráficamente (como un Grafo Acíclico Dirigido - <i>Directed Acyclic Graph</i> (DAG)) cómo paralelizar una expresión numérica compuesta; por ejemplo, $a = (b + c) * (d + e)$ [Diseñar] • Explicar por qué los conceptos de consistencia y tolerancia a fallos no surgen en programas puramente secuenciales [Explicar]
Readings : [PM21]	

Unit 2: Programas: Inicio de Actividades (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Iniciar actividades – Las opciones que permiten que las acciones se realicen (eventualmente) en lugares van desde el cableado directo hasta scripts de configuración; también establecer comunicación y gestión de recursos; estos se expresan de manera diferente en los lenguajes y contextos, generalmente confiando en el aprovisionamiento y gestión automatizados por las plataformas. Ver también: Fundamentos de Sistemas (SF)-Resource – Procedimental: Permitir que múltiples acciones comiencen en un punto de programa dado; por ejemplo, iniciar nuevos hilos, posiblemente delimitando su alcance u organizándolos en grupos jerárquicos. – Reactiva: Habilitar al ocurrir un evento instalando un manejador de eventos, con menos control de cuándo comienzan o terminan las acciones, y puede aplicarse incluso en monoprocesadores. – Dependiente: Habilitar al completarse otras; por ejemplo, secuenciando conjuntos de acciones paralelas. Ver también: Computación Paralela y Distribuida (PDC)-Coordination. – Granularidad: El costo de ejecución de los cuerpos de las acciones debe superar la sobrecarga de organizarlos.	• Escribir un servicio que cree un hilo (u otra forma de activación procedimental) para devolver una página web solicitada a cada nuevo cliente [Escribir]
Readings : [PM21]	

Unit 3: Programas: Propiedades de Ejecución (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Propiedades de Ejecución – Ejecución no determinista de acciones sin orden. – Consistencia: Garantizar el acuerdo entre partes sobre valores y predicados cuando sea necesario para evitar carreras, mantener seguridad y atomicidad, o llegar a consenso. – Tolerancia a fallos: Manejar fallos en las partes o la comunicación, incluyendo (Bizantino) mal comportamiento debido a partes y protocolos no confiables, cuando sea necesario para mantener el progreso o la disponibilidad. Ver también: Fundamentos de Sistemas (SF)-Reliability. – Los compromisos son un foco de evaluación. Ver también: Computación Paralela y Distribuida (PDC)-Evaluation.	• Escribir una función que cuente eventos, como recepciones de paquetes de red, de manera eficiente [Escribir]
Readings : [ST23]	

Unit 4: Programas: Distribución (2 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Distribución – Definir lugares, como dispositivos que ejecutan acciones, incluyendo componentes de hardware, hosts remotos; también pueden incluir dispositivos, hosts y usuarios externos no controlados. Ver también: Arquitectura y Organización (AR)-InterfacingCommunication – Un dispositivo puede dividir el tiempo o emular múltiples acciones paralelas mediante menos procesadores mediante planificación y virtualización. Ver también: Sistemas Operativos (OS)-Scheduling – Nombrar o identificar lugares (por ejemplo, IDs de dispositivo) y acciones como partes (por ejemplo, IDs de hilo). – Las actividades a través de lugares pueden comunicarse a través de medios. Ver también: Computación Paralela y Distribuida (PDC)-Communication	• Escribir un programa de filtro/mapeo/reducción en múltiples estilos [Escribir]
Readings : [ST23]	

Unit 5: Programas: Mapeos de Implementación (2 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Una o más de las siguientes asignaciones y mecanismos a través de sistemas en capas: – Paralelismo a nivel de instrucción y datos en la CPU. Ver también: Arquitectura y Organización (AR)-FunctionalOrganization. – SIMD y paralelismo de datos heterogéneo. Ver también: Arquitectura y Organización (AR)-HeterogeneousArchitectures. – Concurrencia planificada en multinúcleo, tareas, actores. Ver también: Sistemas Operativos (OS)-Scheduling. – Clústeres, nubes; aprovisionamiento elástico. Ver también: Desarrollo de Plataformas Especializadas (SPD)-CommonAspectsSharedConcerns. – Sistemas distribuidos en red. Ver también: Redes y Comunicaciones (NC)-Applications. – Tecnologías emergentes como la computación cuántica y la computación molecular.	• Explicar las compensaciones entre diferentes estrategias de mapeo en términos de rendimiento, costo y complejidad de implementación [Explicar]
Readings : [PM21], [HKH22]	

Unit 6: Programación GPU (4 hours)	
Competences Expected: AG-C09	
Topics	Learning Outcomes
• Computación de Propósito General en GPU (GPGPU) – Arquitectura de GPU: multiprocesadores de flujo (SMs), núcleos CUDA y modelo de ejecución por warp. – Modelo de ejecución SIMT (Single Instruction, Multiple Threads) y sus diferencias respecto al SIMD de CPU. – Interacción host-dispositivo: transferencia de datos entre memoria de CPU (host) y GPU (dispositivo) vía PCIe. – Casos de uso: computación científica, aprendizaje automático, procesamiento de imágenes y simulación a gran escala. • Modelo de Programación CUDA – Jerarquía de ejecución CUDA: grids, bloques de hilos e hilos individuales; mapeo al hardware GPU. – Definición y lanzamiento de kernels; parámetros de configuración (dimensiones de grid y bloque). – Indexación de hilos: threadIdx, blockIdx, blockDim, gridDim; cálculo de índices globales en 1D, 2D y 3D. – Divergencia de warp: implicaciones de las bifurcaciones dentro de un warp en el rendimiento. – Primitiva de sincronización intra-bloque: __syncthreads(). • Jerarquía de Memoria CUDA – Memoria global: grande, alta latencia, accesible por todos los hilos; patrones de acceso coalescente para el rendimiento. – Memoria compartida: memoria en chip de baja latencia compartida dentro de un bloque de hilos; conflictos de banco. – Registros y memoria local: almacenamiento privado por hilo. – Memoria constante y de textura: cachés de solo lectura optimizadas para patrones de acceso específicos. – Memoria unificada: modelo de programación simplificado con migración automática de datos entre host y dispositivo.	• Escribir un kernel CUDA que realice una operación data-paralela (p. ej., suma de vectores, multiplicación de matrices) y configurar correctamente las dimensiones de grid y bloque [Escribir] • Analizar el impacto de los patrones de acceso a memoria global (acceso coalescente vs. acceso entrelazado) y el uso de memoria compartida en el rendimiento de un kernel GPU [Analizar] • Diseñar un algoritmo paralelo usando el modelo de programación CUDA, incluyendo la jerarquía de hilos, estrategia de asignación de memoria y transferencias de datos entre host y dispositivo [Diseñar]
Readings : [KH16], [Cor24]	

Unit 7: Comunicación (8 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
- Medios - Variedades: canales (paso de mensajes o E/S), memoria compartida, heterogéneos, almacenes de datos. - Dependencia de la disponibilidad y naturaleza del hardware subyacente, conectividad y protocolos; soporte del lenguaje, emulación. Ver también: Arquitectura y Organización (AR)-InterfacingCommunication. - Canales - Medios de comunicación de parte a parte explícitos (generalmente nombrados). - APIs: Sockets, constructos arquitectónicos, basados en lenguaje y de kits de herramientas, como Message Passing Interface (MPI), y constructos en capas como Llamada a Procedimiento Remoto (RPC). Ver también: Redes y Comunicaciones (NC)-Fundamentals. - APIs de canales de E/S. - Memoria - Arquitecturas de memoria compartida en las que las partes se comunican directamente solo con la memoria en direcciones dadas, con extensiones a memoria heterogénea que admite múltiples almacenes de memoria con transferencia de datos explícita entre ellos; por ejemplo, memoria local y compartida de GPU, Acceso Directo a Memoria (DMA). - Jerarquías de memoria: Múltiples capas de dominios, alcances y cachés compartidos; localidad: latencia, falso uso compartido. - Propiedades de consistencia: Límites de atomicidad a nivel de bits, coherencia, orden local. - Almacenes de Datos - Estructuras de datos mantenidas cooperativamente que implementan mapas y TADs relacionados. - Variedades: Propiedad exclusiva, compartida, fragmentada, replicada, inmutable, versionada.	- Explicar las similitudes y diferencias entre: (1) La parte A envía un mensaje en el canal X con contenido 1 recibido por la parte B (2) A establece la variable compartida X a 1, leída por B (3) A establece "X=1" en un mapa compartido distribuido al que accede B [Explicar] - Escribir un programa que distribuya diferentes segmentos de un conjunto de datos a múltiples trabajadores y recoja los resultados (por ejemplo, sumar segmentos de un arreglo) [Escribir] - Escribir un programa paralelo que solicite datos de múltiples sitios y los resuma usando alguna forma de reducción [Escribir] - Comparar el rendimiento de versiones con y sin búfer de un programa productor-consumidor [Comparar]
Readings : [ST23], [Kle17]	

Unit 8: Comunicación: Propiedades y Extensiones (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Una o más de las siguientes propiedades y extensiones: – Topologías: Unicast, Multicast, Buzones, Conmutadores; Enrutamiento a través de redes de interconexión de hardware y software. – Propiedades de concurrencia de medios: Orden, consistencia, idempotencia, superposición de comunicación con computación. – Rendimiento del medio: Latencia, ancho de banda (rendimiento), contención (congestión), capacidad de respuesta (vivacidad), confiabilidad (tasas de error y pérdida), progreso basado en protocolo (acks, tiempos de espera, mediación). – Propiedades de seguridad del medio: integridad, privacidad, autenticación, autorización. Ver también: Seguridad (SEC)-Coding. – Formatos de datos: Serialización, validación, cifrado, compresión. – Políticas de canal: Puntos finales, sesiones, almacenamiento en búfer, respuesta a saturación (esperar vs. descartar), control de velocidad. – Multiplexación y demultiplexación de muchos dispositivos o partes de E/S relativamente lentos; técnicas basadas en finalización y basadas en planificador; APIs async-await, select y polling. – Formalización y análisis de comunicación por canales; por ejemplo, CSP. – Aplicaciones de la teoría de colas para modelar y predecir el rendimiento.	• Determinar si un esquema de comunicación dado proporciona propiedades de seguridad suficientes para un uso dado [Determinar] • Dar un ejemplo de un escenario en el que los envíos de mensajes bloqueantes pueden causar un punto muerto (deadlock) [Crear] • Describir al menos una técnica de diseño para evitar fallos de vivacidad en programas que usan múltiples bloqueos [Describir]
Readings : [ST23], [Kle17]	

Unit 9: Memoria y Consistencia (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
- Modelos de memoria, consistencia de datos y tolerancia a fallos: - Modelos de memoria: consistencia secuencial y de liberación/adquisición. - Gestión de memoria; incluyendo la reclamación de datos compartidos; conteo de referencias y alternativas. - Colocación y transferencia masiva de datos; reducción del tráfico de mensajes y mejora de la localidad; superposición de transferencia de datos y computación; impacto del diseño de datos como array de estructuras vs. estructura de arrays. - Emular memoria compartida: memoria compartida distribuida, Acceso Directo a Memoria Remota (RDMA). - Consistencia de almacenes de datos: Atomicidad, linealizabilidad, transaccionalidad, coherencia, orden causal, resolución de conflictos, consistencia eventual, cadenas de bloques. - Fallos, particionamiento y fallos parciales; votación; protocolos como Paxos y Raft. - Compromisos de diseño entre consistencia, disponibilidad, tolerancia a partición (fallos); imposibilidad de cumplir con todos a la vez. - Seguridad y confianza: Fallos bizantinos, prueba de trabajo y alternativas.	- Dar un ejemplo de un orden de accesos entre actividades concurrentes (por ejemplo, un programa con una carrera de datos) que no sea secuencialmente consistente [Crear] - Escribir un programa que ilustre el reordenamiento de acceso a memoria o de mensajes [Escribir] - Describir los méritos relativos del control de concurrencia optimista versus conservador bajo diferentes tasas de contención entre actualizaciones [Describir] - Dar un ejemplo de un escenario en el que un intento de actualización optimista podría nunca completarse [Crear] - Modificar un sistema concurrente para usar un almacén de datos más escalable, confiable o disponible [Crear] - Usando una plataforma existente que admita almacenes de datos replicados, escribir un programa que mantenga un mapeo clave-valor incluso cuando uno o más hosts fallen [Escribir]
Readings : [ST23], [Kle17]	

Unit 10: Coordinación (7 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Dependencias – La iniciación o progreso de una actividad puede depender de otras actividades, para evitar condiciones de carrera, garantizar la terminación o cumplir otros requisitos. – Garantizar el progreso evitando ciclos de dependencia, usando condiciones monótonas, eliminando dependencias no esenciales. • Constructos de control y patrones de diseño: – Basados en finalización: Barreras, reuniones (joins), incluido el control de terminación. – Habilitados por datos: Colas, diseños productor-consumidor. – Basados en condición: Polling, reintentos, retrocesos, ayuda, suspensión, señalización, tiempos de espera. – Reactivos: Habilitar y activar continuaciones.	• Mostrar cómo garantizar que un programa termine correctamente cuando todas las tareas concurrentes de un conjunto hayan finalizado [Diseñar] • Escribir una función que cuente eventos, como entradas de sensores o recepciones de paquetes de red, de manera eficiente [Escribir] • Escribir un programa de filtro/mapeo/reducción en múltiples estilos [Escribir] • Escribir un programa en el que la terminación de un conjunto de acciones paralelas sea seguida por otra [Escribir] • Escribir un servicio que cree un hilo (u otra forma de activación procedimental) para devolver una página web solicitada a cada nuevo cliente [Escribir]
Readings : [Her+20], [ST23]	

Unit 11: Coordinación: Sincronización y Atomicidad (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Atomicidad – Instrucciones atómicas, ordenamientos de acceso local impuestos. – Bloqueos y exclusión mutua; granularidad de bloqueo. – Uso de bloqueos en un lenguaje específico; mantener la vivacidad sin introducir carreras. – Evitar punto muerto (<i>deadlock</i>): Ordenamiento, mayor granularidad, reintentos aleatorios; retrocesos, encapsulación mediante administradores de bloqueos. – Errores comunes: No bloquear o desbloquear cuando es necesario, mantener bloqueos mientras se invocan operaciones desconocidas. – Evitar bloqueos: replicación, solo lectura, propiedad, y construcciones no bloqueantes.	• Mostrar cómo evitar o reparar un error de carrera en un programa dado [Analizar]
Readings : [Her+20]	

Unit 12: Coordinación: Propiedades Avanzadas (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Una o más de las siguientes propiedades y extensiones: – Propiedades de progreso incluyendo libre de bloqueo (<i>lock-free</i>), libre de espera (<i>wait-free</i>), equidad (<i>fairness</i>), planificación por prioridad, interacciones con consistencia, confiabilidad. – Rendimiento con respecto a contención, granularidad, convoy, escalabilidad. – Estructuras de datos y algoritmos no bloqueantes. – Propiedad y control de recursos. – Variantes y alternativas de bloqueos: bloqueos de secuencia, bloqueos de lectura-escritura; Actualizar-Copiando-Lectura (RCU), reentrada; tickets; control del ciclo activo (<i>spinning</i>) versus bloqueo. – Control basado en transacciones: Optimista y conservador. – Bloqueo distribuido: confiabilidad. – Alternativas a barreras: Relojes; contadores, relojes virtuales; flujo de datos y continuaciones; futuros y RPC; basadas en consenso, recolección de resultados con reductores y colectores. – Especulación, selección, cancelación; consecuencias en observabilidad y seguridad. – Control de recursos usando semáforos y variables de condición. – Flujo de control: Planificación de computaciones, bucles serie-paralelo con líderes (posiblemente elegidos), tuberías y flujos, paralelismo anidado. – Excepciones y fallos. Manejadores, detección, tiempos de espera, tolerancia a fallos, votación.	• Escribir un programa que busque especulativamente una solución mediante múltiples actividades, terminando las demás cuando se encuentre una [Escribir] • Escribir un programa en el que una excepción numérica (como división por cero) en una actividad cause la terminación de las demás [Escribir] • Escribir un programa para que múltiples partes acuerden la hora actual del día; discutir sus limitaciones en comparación con protocolos como el protocolo de transferencia de red (NTP) [Escribir]
Readings : [Her+20]	

Unit 13: Evaluación (7 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Requisitos de seguridad (<i>safety</i>) y vivacidad (<i>liveness</i>) en términos de constructos de lógica temporal para expresar "siempre" y "eventualmente" Ver también: Fundamentos de los Lenguajes de Programación (FPL)-ParallelDistributedComputing. • Identificar, probar y reparar violaciones, incluyendo formas comunes de errores como no garantizar el orden necesario (errores de carrera), atomicidad (incluyendo errores de "verificar luego actuar") y terminación (bloqueo activo). • Métricas de requisitos de rendimiento para rendimiento (<i>throughput</i>), capacidad de respuesta, latencia, disponibilidad, consumo de energía, escalabilidad, uso de recursos, costos de comunicación, espera y control de velocidad, equidad; acuerdos de nivel de servicio. Ver también: Fundamentos de Sistemas (SF)-Performance. • Impacto en el rendimiento de las opciones de diseño e implementación, incluyendo granularidad, sobrecarga, costos de consenso y consumo de energía. Ver también: Sociedad, ética y la Profesión (SEP)-Sustainability. • Estimar limitaciones de escalabilidad, por ejemplo usando la Ley de Amdahl o la Ley de Escalabilidad Universal. Ver también: Fundamentos de Sistemas (SF)-Evaluation.	• Revisar una especificación para habilitar el paralelismo y la distribución sin violar otras propiedades o características esenciales [Rediseñar] • Explicar cómo las nociones concurrentes de seguridad (<i>safety</i>) y vivacidad (<i>liveness</i>) extienden sus contrapartes secuenciales [Explicar] • Especificar un conjunto de invariantes que deben mantenerse en cada paso de cómputo de paralelismo masivo [Analizar] • Escribir un programa de prueba que pueda revelar un error de carrera de datos; por ejemplo, perder una actualización cuando dos actividades intentan incrementar una variable [Escribir] • En un contexto dado, explicar hasta qué punto se esperaría que introducir paralelismo en un programa por lo demás secuencial mejoraría el rendimiento (<i>throughput</i>) y/o reduciría la latencia, y cómo podría afectar la eficiencia energética [Explicar] • Mostrar cómo cambian la escalabilidad y la eficiencia para problemas de muestra con y sin el supuesto de que el tamaño del problema cambia con el número de procesadores; además, explicar si y cómo cambiaría la escalabilidad bajo relajaciones de dependencias secuenciales [Diseñar]
Readings : [SBA24], [PM21]	

Unit 14: Evaluación: Métodos Formales (3 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Métodos formales de verificación y análisis: – Extensiones a requisitos formales secuenciales como la linealizabilidad. – Especificaciones de protocolo, sesión y transacciones. – Uso de herramientas como Lenguaje Unificado de Modelado (UML), Lógica Temporal de Acciones (TLA), lógicas de programa. – Análisis de seguridad: seguridad y vivacidad en presencia de comportamientos hostiles o con errores de otras partes; propiedades requeridas de mecanismos de comunicación (por ejemplo, ausencia de fugas entre capas), filtrado de entrada, limitación de velocidad. Ver también: Seguridad (SEC)-Foundations. – Análisis estático aplicado a corrección, rendimiento (<i>throughput</i>), latencia, recursos, energía. Ver también: Sociedad, ética y la Profesión (SEP)-Sustainability. – Análisis del modelo de Grafo Acíclico Dirigido (DAG) de eficiencia algorítmica (trabajo, span, caminos críticos).	• Especificar y medir el comportamiento cuando un servicio es solicitado por un número inesperadamente grande de clientes [Analizar] • Identificar y reparar un problema de rendimiento debido a cuellos de botella secuenciales [Analizar] • Comparar empíricamente el rendimiento (<i>throughput</i>) de dos implementaciones de un diseño común (quizás usando un marco de pruebas existente) [Comparar]
Readings : [SBA24], [PM21]	

Unit 15: Evaluación: Pruebas y Medición (2 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Herramientas de prueba y técnicas de medición: – Pruebas y depuración; herramientas como detectores de carreras, fuzzers, verificadores de dependencia de bloqueos, pruebas de unidad/esfuerzo/tortura, visualizaciones, integración continua (CI), despliegue continuo (CD), y generadores de pruebas. – Medir y comparar rendimiento (<i>throughput</i>), sobrecarga, espera, contención, comunicación, movimiento de datos, localidad, uso de recursos, comportamiento en presencia de números excesivos de eventos, clientes o hilos. Ver también: Fundamentos de Sistemas (SF)-Evaluation. – Análisis específicos del dominio de aplicación y técnicas de evaluación.	• Identificar y reparar un problema de rendimiento debido a latencia de comunicación o datos [Analizar] • Identificar y reparar un problema de rendimiento debido a sobrecarga en la gestión de recursos [Analizar] • Identificar y reparar un problema de confiabilidad o disponibilidad [Analizar]
Readings : [SBA24], [PM21]	

Unit 16: Algoritmos (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Expresar e implementar algoritmos en lenguajes y marcos de trabajo dados, para iniciar actividades (por ejemplo hilos), usar constructos de memoria compartida, y APIs de canales, sockets y/o llamada a procedimiento remoto (RPC). Ver también: Fundamentos de los Lenguajes de Programación (FPL)-ParallelDistributedComputing. – Ejemplos de datos paralelos incluyendo mapeo/reducción. – Uso de APIs de canal, socket y/o RPC en un lenguaje dado, con control del programa para enviar (generalmente procedimental) vs recibir (generalmente reactivo o basado en RPC). – Uso de bloqueos, barreras y/o sincronizadores para mantener la vivacidad sin introducir carreras.	• Implementar un componente paralelo/distribuido basado en un algoritmo conocido [Implementar] • Escribir un programa de datos paralelos que, por ejemplo, calcule el promedio de un arreglo de números [Escribir] • Escribir un programa productor-consumidor en el que un componente genere números y otro calcule su promedio. Medir las aceleraciones cuando los números son escalares pequeños versus valores de multiprecisión grandes [Escribir]
Readings : [PM21], [Her+20]	

Unit 17: Algoritmos: Dominios de Aplicación (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Panorama de dominios de aplicación comunes en multinúcleo, reactivo, datos paralelos, clúster, nube, sistemas distribuidos abiertos y marcos de trabajo (con referencia a la siguiente tabla). – Multinúcleo: Agentes de ejecución típicos: Hilos. Mecanismos de comunicación típicos: Memoria compartida, Atómicos, bloqueos. Dominios algorítmicos típicos: Gestión de recursos, procesamiento de datos. Objetivos de ingeniería típicos: Rendimiento (<i>throughput</i>), latencia, energía. – Reactivo: Agentes de ejecución típicos: Manejadores, hilos. Mecanismos de comunicación típicos: Canales de E/S. Dominios algorítmicos típicos: Servicios, tiempo real. Objetivos de ingeniería típicos: Latencia. – Datos paralelos: Agentes de ejecución típicos: GPU, SIMD, aceleradores, híbridos. Mecanismos de comunicación típicos: Memoria heterogénea. Dominios algorítmicos típicos: álgebra lineal, gráficos, análisis de datos. Objetivos de ingeniería típicos: Rendimiento (<i>throughput</i>), energía. – Clúster: Agentes de ejecución típicos: Hosts gestionados. Mecanismos de comunicación típicos: Sockets, canales. Dominios algorítmicos típicos: Simulación, análisis de datos. Objetivos de ingeniería típicos: Rendimiento (<i>throughput</i>). – Nube: Agentes de ejecución típicos: Hosts aprovisionados. Mecanismos de comunicación típicos: APIs de servicio. Dominios algorítmicos típicos: Aplicaciones web. Objetivos de ingeniería típicos: Escalabilidad. – Distribuido abierto: Agentes de ejecución típicos: Hosts autónomos. Mecanismos de comunicación típicos: Sockets, Almacenes de datos. Dominios algorítmicos típicos: Almacenes de datos y servicios tolerantes a fallos. Objetivos de ingeniería típicos: Confiabilidad.	• Extender un programa secuencial dirigido por eventos estableciendo una nueva actividad en un manejador de eventos (por ejemplo, un nuevo hilo en un manejador de acciones de GUI) [Diseñar] • Mejorar el rendimiento de un componente secuencial introduciendo paralelismo y/o distribución [Crear] • Elegir entre diferentes diseños paralelos/distribuidos para componentes de un sistema dado [Evaluar (valorar)]
Readings : [PM21], [Her+20]	

Unit 18: Algoritmos: Dominios Algorítmicos (4 hours)	
Competences Expected: AG-C09,AG-C11	
Topics	Learning Outcomes
• Uno o más de los siguientes dominios algorítmicos. Ver también: Fundamentos Algorítmicos (AL)-AlgorithmicStrategies: – álgebra lineal: Operaciones con vectores y matrices, precisión/estabilidad numérica, aplicaciones en análisis de datos y aprendizaje automático. – Procesamiento de datos: ordenación, búsqueda y recuperación, estructuras de datos concurrentes. – Grafos, búsqueda y combinatoria: Marcado, paralelización de aristas, acotamiento, especulación, análisis basado en redes. – Modelado y simulación: ecuaciones diferenciales; aleatorización, problemas de N-cuerpos, algoritmos genéticos. – Lógica computacional: satisfactibilidad (SAT), programación lógica concurrente. – Gráficos y geometría computacional: Transformaciones, renderizado, trazado de rayos. – Gestión de recursos: Asignar, colocar, reciclar y planificar procesadores, memoria, canales y hosts; recursos exclusivos vs compartidos; algoritmos estáticos, dinámicos y elásticos; Restricciones de tiempo real; Lotes, priorización, partición; descentralización mediante robo de trabajo y técnicas relacionadas. – Servicios: Implementar APIs web, moneda electrónica, sistemas de transacción, juegos multi-jugador.	• Diseñar, implementar, analizar y evaluar un componente o aplicación para X que opere en un contexto dado, donde X esté en uno de los dominios listados, por ejemplo, un algoritmo genético para el diseño de una planta de fábrica [Diseñar] • Criticar el diseño e implementación de un componente o aplicación existente, o uno desarrollado por compañeros de clase [Criticar (análisis crítico)] • Comparar el rendimiento y la eficiencia energética de múltiples implementaciones de un diseño similar, por ejemplo, multinúcleo versus clúster versus GPU [Comparar]
Readings : [PM21], [Her+20]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

***** EVALUATION MISSING *****

10. BASIC BIBLIOGRAPHY

[KH16] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. 3rd. Cambridge, MA: Morgan Kaufmann, 2016.

- [Kle17] Martin Kleppmann. *Designing Data-Intensive Applications*. 1st. O'Reilly Media, 2017.
- [Her+20] Maurice Herlihy et al. *The Art of Multiprocessor Programming*. 2nd. Morgan Kaufmann, 2020.
- [PM21] Peter S. Pacheco and Matthew Malensek. *An Introduction to Parallel Programming*. 2nd. Morgan Kaufmann, 2021.
- [HKH22] Wen-mei W. Hwu, David B. Kirk, and Izzat El Hajj. *Programming Massively Parallel Processors: A Hands-on Approach*. 4th. Morgan Kaufmann, 2022.
- [ST23] Maarten van Steen and Andrew S. Tanenbaum. *Distributed Systems*. 4th. Maarten van Steen, 2023.
- [Cor24] NVIDIA Corporation. *CUDA C++ Programming Guide*. Documentación oficial de NVIDIA. 2024. URL: <https://docs.nvidia.com/cuda/cuda-c-programming-guide/>.
- [SBA24] Thomas Sterling, Maciej Brodowicz, and Matthew Anderson. *High Performance Computing: Modern Systems and Practices*. 2nd. Morgan Kaufmann, 2024.