National University of Engineering (UNI)
School of Artificial Intelligence
Syllabus 2024-I

1. COURSE

CS3P3. Internet of Things (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS3P3. Internet of Things
2.2 Semester	:	10^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	1 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 Prerrequisites	:	CS3P1. Parallel and Distributed Computing . (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The last decade has an explosive growth in multiprocessor computing, including multi-core processors and distributed data centers. As a result, parallel and distributed computing has evolved from a broadly elective subject to be one of the major components in mesh studies in undergraduate computer science. Both parallel computing and distribution involve the simultaneous execution of multiple processes on different devices that change position.

5. GOALS

- That the student is able to create parallel applications of medium complexity by efficiently taking advantage of different mobile devices.

6. COMPETENCES

1) Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones. (**Usar**)

2) Diseñar, implementar y evaluar una solución basada en computación para cumplir con un conjunto determinado de requisitos computacionales en el contexto de las disciplinas del programa. (**Usar**)

6) Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en computación. (**Usar**)

7. TOPICS

Unit 1: Fundamentos de paralelismo (18 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Procesamiento Simultáneo Múltiple. • Metas del Paralelismo (ej. rendimineto) frente a Concurrencia (ej. control de acceso a recursos compartidos) • Paralelismo, comunicación, y coordinación: – Paralelismo, comunicación, y coordinación – Necedidad de Sincronización • Errores de Programación ausentes en programación secuencial: – Tipos de Datos (lectura/escritura simultánea o escritura/escritura compartida) – Tipos de Nivél más alto (interleavings violating program intention, no determinismo no deseado) – Falta de vida/progreso (deadlock, starvation)	• Distinguir el uso de recursos computacionales para una respuesta mas rápida para administrar el acceso eficiente a un recurso compartido [Familiarizarse] • Distinguir múltiples estructuras de programación suficientes para la sincronización que pueden ser interimplementables pero tienen ventajas complementarias [Familiarizarse] • Distinguir datos de carrera (*data races*) a partir de carreras de mas alto nivel [Familiarizarse]
Readings : [Pac11], [Mat14], [Qui03]	

Unit 2: Arquitecturas paralelas (12 hours)

Competences Expected:

Topics	Learning Outcomes

• Procesadores mutlinúcleo.
• Memoria compartida vs memoria distribuida.
• Multiprocesamiento simétrico.
• SIMD, procesamiento de vectores.
• GPU, coprocesamiento.
• Taxonomia de Flynn.
• Soporte a nivel de instrucciones para programación paralela.
  – Instrucciones atómicas como Compare/Set (Comparar / Establecer)
• Problemas de Memoria:
  – Caches multiprocesador y coherencia de cache
  – Acceso a Memoria no uniforme (NUMA)
• Topologías.
  – Interconecciones
  – Clusters
  – Compartir recursos (p.e., buses e interconexiones)
	• Explicar las diferencias entre memoria distribuida y memoria compartida [Evaluar]
	• Describir la arquitectura SMP y observar sus principales caracteristicas [Evaluar]
	• Distinguir los tipos de tareas que son adecuadas para máquinas SIMD [Usar]
	• Describir las ventajas y limitaciones de GPUs vs CPUs [Usar]
	• Explicar las caracteristicas de cada clasificación en la taxonomía de Flynn [Usar]
	• Describir los desafíos para mantener la coherencia de la caché [Familiarizarse]
	• Describir los desafíos clave del desempeño en diferentes memorias y topologías de sistemas distribuidos [Familiarizarse]

Readings : [Pac11], [KH13], [SK10]

Unit 3: Descomposición en paralelo (18 hours)

Competences Expected:

Topics	Learning Outcomes

• Necesidad de Comunicación y coordinación/sincronización.
• Independencia y Particionamiento.
• Conocimiento Básico del Concepto de Descomposición Paralela.
• Decomposición basada en tareas:
  – Implementación de estrategias como hebras
• Descomposición de Información Paralela
  – Estrategias como SIMD y MapReduce
• Actores y Procesos Reactivos (solicitud de gestores)
	• Explicar por qué la sincronización es necesaria en un programa paralelo especifico [Usar]
	• Identificar oportunidades para particionar un programa serial en módulos paralelos independientes [Familiarizarse]
	• Escribir un algoritmo paralelo correcto y escalable [Usar]
	• Paralelizar un algoritmo mediante la aplicación de descomposición basada en tareas [Usar]
	• Paralelizar un algoritmo mediante la aplicación de descomposición datos en paralelo [Usar]
	• Escribir un programa usando actores y/o procesos reactivos [Usar]

Readings : [Pac11], [Mat14], [Qui03]

Unit 4: Comunicación y coordinación (18 hours)

Competences Expected:

Topics	Learning Outcomes
• Memoria Compartida. • La consistencia, y su papel en los lenguaje de programación garantias para los programas de carrera libre. • Pasos de Mensaje: – Mensajes Punto a Punto versus multicast (o basados en eventos) – Estilos para enviar y recibir mensajes Blocking vs non-blocking – Buffering de mensajes • Atomicidad: – Especificar y probar atomicidad y requerimientos de seguridad – Granularidad de accesos atómicos y actualizaciones, y uso de estructuras como secciones críticas o transacciones para describirlas – Exclusión mutua usando bloques, semáforos, monitores o estructuras relacionadas ∗ Potencial para fallas y bloqueos (*deadlock*) (causas, condiciones, prevención) – Composición ∗ Componiendo acciones atómicas granulares más grandes usando sincronización ∗ Transacciones, incluyendo enfoques optimistas y conservadores • Consensos: – (Ciclicos) barerras, contadores y estructuras relacionadas • Acciones condicionales: – Espera condicional (p.e., empleando variables de condición)	• Usar exclusión mútua para evitar una condición de carrera [Usar] • Dar un ejemplo de una ordenación de accesos entre actividades concurrentes (por ejemplo, un programa con condición de carrera) que no son secuencialmente consistentes [Familiarizarse] • Dar un ejemplo de un escenario en el que el bloqueo de mensajes enviados pueden dar *deadlock* [Usar] • Explicar cuándo y por qué mensajes de multidifusión (*multicast*) o basado en eventos puede ser preferible a otras alternativas [Familiarizarse] • Escribir un programa que termine correctamente cuando todo el conjunto de procesos concurrentes hayan sido completados [Usar] • Dar un ejemplo de un escenario en el que un intento optimista de actualización puede nunca completarse [Familiarizarse] • Usar semaforos o variables de condición para bloquear hebras hasta una necesaria precondición de mantenga [Usar]

Readings : [Pac11], [Mat14], [Qui03]

Unit 5: Análisis y programación de algoritmos paralelos (18 hours)

Competences Expected:

Topics	**Learning Outcomes**
• Caminos críticos, el trabajo y la duración y la relación con la ley de Amdahl. • Aceleración y escalabilidad. • Naturalmente (vergonzosamente) algoritmos paralelos. • Patrones Algoritmicos paralelos (divide-y-conquista, map/reduce, amos-trabajadores, otros) – Algortimos específicos (p.e., MergeSort paralelo) • Algoritmos de grafos paralelo (por ejemplo, la ruta más corta en paralelo, árbol de expansión paralela) • Cálculos de matriz paralelas. • Productor-consumidor y algoritmos paralelos segmentados. • Ejemplos de algoritmos paralelos no-escalables.	• Definir: camino crítico, trabajo y *span* [Familiarizarse] • Calcular el trabajo y el *span* y determinar el camino crítico con respecto a un diagrama de ejecución paralela. [Usar] • Definir *speed-up* y explicar la noción de escalabilidad de un algoritmo en este sentido [Familiarizarse] • Identificar tareas independientes en un programa que debe ser paralelizado [Usar] • Representar características de una carga de trabajo que permita o evite que sea naturalmente paralelizable [Familiarizarse] • Implementar un algoritmo dividir y conquistar paralelo (y/o algoritmo de un grafo) y medir empiricamente su desempeño relativo a su analogo secuencial [Usar] • Descomponer un problema (por ejemplo, contar el número de ocurrencias de una palabra en un documento) via operaciones *map* y *reduce* [Usar] • Proporcionar un ejemplo de un problema que se corresponda con el paradigma productor-consumidor [Usar] • Dar ejemplos de problemas donde el uso de *pipelining* sería un medio eficaz para la paralelización [Usar] • Implementar un algoritmo de matriz paralela [Usar] • Identificar los problemas que surgen en los algoritmos del tipo productor-consumidor y los mecanismos que pueden utilizarse para superar dichos problemas [Usar]
Readings : [Mat14], [Qui03]	

Unit 6: Desempeño en paralelo (18 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Equilibrio de carga. • La medición del desempeño. • Programación y contención. • Evaluación de la comunicación de arriba. • Gestión de datos: – Costos de comunicación no uniforme debidos a proximidad – Efectos de Cache (p.e., false sharing) – Manteniendo localidad espacial • Consumo de energía y gestión.	• Detectar y corregir un desbalanceo de carga [Usar] • Calcular las implicaciones de la ley de Amdahl para un algoritmo paralelo particular [Usar] • Describir como la distribuición/disposición de datos puede afectar a los costos de comunicación de un algoritmo [Familiarizarse] • Detectar y corregir una instancia de uso compartido falso (*false sharing*) [Usar] • Explicar el impacto de la planificación en el desempeño paralelo [Familiarizarse] • Explicar el impacto en el desempeño de la localidad de datos [Familiarizarse] • Explicar el impacto y los puntos de equilibrio relacionados al uso de energía en el desempeño paralelo [Familiarizarse]
Readings : [Pac11], [Mat14], [KH13], [SK10]	

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

[KH13] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. 2nd. Morgan Kaufmann, 2013. ISBN: 978-0-12-415992-1.

[Mat14] Norm Matloff. *Programming on Parallel Machines*. University of California, Davis, 2014. URL: `http://heather.cs.ucdavi`

[Pac11] Peter S. Pacheco. *An Introduction to Parallel Programming*. 1st. Morgan Kaufmann, 2011. ISBN: 978-0-12-374260-5.

[Qui03] Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP*. 1st. McGraw-Hill Education Group, 2003. ISBN: 0071232656.

[SK10] Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. 1st. Addison-Wesley Professional, 2010. ISBN: 0131387685, 9780131387683.