Universidad San Ignacio de Loyola (USIL)
Carrera Profesional de
Ingeniería de Software
Sílabo 2021-II

1. CURSO
SFW62062. Internet de las Cosas (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Créditos	:	2
2.2 Horas de teoría	:	-
2.3 Horas de práctica	:	2 (Semanal)
2.4 Horas autónomas	:	64 (horas)
2.5 Duración del periodo	:	16 semanas
2.6 Condición	:	Obligatorio
2.7 Modalidad	:	Presencial
2.8 Prerrequisitos	:	SFW62071. Computación Paralela y Distribuída. (8^{vo} Sem)

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO
La última década ha traído un crecimiento explosivo en computación con multiprocesadores, incluyendo los procesadores de varios núcleos y centros de datos distribuidos. Como resultado, la computación paralela y distribuida se ha convertido de ser un tema ampliamente electivo para ser uno de los principales componentes en la malla estudios en ciencia de la computación de pregrado. Tanto la computación paralela como la distribuida implica la ejecución simultánea de múltiples procesos en diferentes dispositivos que cambian de posición.

5. OBJETIVOS

- Que el alumno sea capaz de crear aplicaciones paralelas de mediana complejidad aprovechando eficientemente distintos dispositivos móviles.

6. COMPETENCIAS

3) Justifica un desempeño individual, como parte de equipos de trabajo o como líder de proyectos de grupos multidisciplinarios en entornos globales con el fin de asegurar la calidad de software, aplicando normas, configuraciones, regulaciones y métricas. (**Evaluar**)

4) Diseña soluciones de Software de acuerdo a los estándares y políticas de seguridad de la información en uno o varios dominios de aplicación siendo socialmente responsables y demostrando ética profesional. (**Evaluar**)

7. TEMAS

Unidad 1: Fundamentos de paralelismo (18 horas)	
Competencias esperadas:	
Temas	**Objetivos de Aprendizaje**
• 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]
Aprendizaje autónomo • Desarrollo de ejercicios prácticos	
Lecturas : [Pac11], [Mat14], [Qui03]	

Unidad 2: Arquitecturas paralelas (12 horas)

Competencias esperadas:

Temas	**Objetivos de Aprendizaje**

• 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]

Aprendizaje autónomo

- Desarrollo de ejercicios prácticos

Lecturas : [Pac11], [KH13], [SK10]

Unidad 3: Descomposición en paralelo (18 horas)	
Competencias esperadas:	
Temas	**Objetivos de Aprendizaje**

• 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]

Aprendizaje autónomo • Desarrollo de ejercicios prácticos	
Lecturas : [Pac11], [Mat14], [Qui03]	

Unidad 4: Comunicación y coordinación (18 horas)

Competencias esperadas:

Temas	**Objetivos de Aprendizaje**
• 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]

Aprendizaje autónomo

- Desarrollo de ejercicios prácticos

Lecturas : [Pac11], [Mat14], [Qui03]

Unidad 5: Análisis y programación de algoritmos paralelos (18 horas)

Competencias esperadas:

Temas	**Objetivos de Aprendizaje**
• 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]
Aprendizaje autónomo • Desarrollo de ejercicios prácticos	
Lecturas : [Mat14], [Qui03]	

Unidad 6: Desempeño en paralelo (18 horas)	
Competencias esperadas:	
Temas	**Objetivos de Aprendizaje**
• 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]
Aprendizaje autónomo • Desarrollo de ejercicios prácticos	
Lecturas : [Pac11], [Mat14], [KH13], [SK10]	

8. PLAN DE TRABAJO

8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

9. SISTEMA DE EVALUACIÓN

Cada uno de los rubros del esquema de evaluación y la nota final del curso son redondeados a números enteros. La nota final del curso es el promedio ponderado de los rubros correspondientes: evaluación permanente, examen parcial y examen final.

Los promedios calculados componentes del rubro 'Evaluación Permanente' mantendrán su cálculo con 2 decimales.

	%	**Observaciones**	**Semana**	**Rezagable**
Evaluación Continua	70%			
Práctica Calificada	70%			
Práctica Calificada$_1$		Se elimina la práctica con la menor nota	4	No
Práctica Calificada$_2$		Se elimina la práctica con la menor nota	8	No
Práctica Calificada$_3$		Se elimina la práctica con la menor nota	12	No
Proyecto	30%		15	
Examen final	30%			

10. BIBLIOGRAFÍA BÁSICA

[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.ucdavis.edu/~matloff/158/PLN/ParProcBook.pdf`.

[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.