Universidad Católica San Pablo (UCSP)
Escuela Profesional de
Ciencia de la Computación
SILABO

CS3P1. Computación Paralela y Distribuída (Obligatorio)

2023-II

1. Información general		
1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS3P1. Computación Paralela y Distribuída
1.3 Semestre	:	8^{vo} Semestre.
1.4 Prerrequisitos	:	• CS212. Análisis y Diseño de Algoritmos. (5^{to} Sem) • CS231. Redes y Comunicación. (7^{mo} Sem)
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Presencial
1.7 horas	:	2 HT; 4 HP;
1.8 Créditos	:	4
1.9 Plan	:	Plan Curricular 2016

2. Profesores

Titular

- Alvaro Henry Mamani-Aliaga <ahmamani@ucsp.edu.pe>
  - Doctor en Ciencia de la Computación, UNSA, Perú, 2019.
  - Master en Ciencia de la Computación, IME-USP, Brasil, 2011.

3. Fundamentación del 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, cuyas operaciones tienen el potencial para intercalar de manera compleja. La computación paralela y distribuida construye sobre cimientos en muchas áreas, incluyendo la comprensión de los conceptos fundamentales de los sistemas, tales como: concurrencia y ejecución en paralelo, consistencia en el estado/manipulación de la memoria, y latencia. La comunicación y la coordinación entre los procesos tiene sus cimientos en el paso de mensajes y modelos de memoria compartida de la computación y conceptos algorítmicos como atomicidad, el consenso y espera condicional. El logro de aceleración en la práctica requiere una comprensión de algoritmos paralelos, estrategias para la descomposición problema, arquitectura de sistemas, estrategias de implementación y análisis de rendimiento. Los sistemas distribuidos destacan los problemas de la seguridad y tolerancia a fallos, hacen hincapié en el mantenimiento del estado replicado e introducen problemas adicionales en el campo de las redes de computadoras.

4. Resumen

1. Bases teóricas de la Computación Paralela y Distribuida 2. Sistemas distribuídos 3. Comunicación y coordinación 4. Programación para el Procesamiento Masivamente Paralelo

5. Objetivos Generales

- Que el alumno sea capaz de crear aplicaciones paralelas de mediana complejidad aprovechando eficientemente máquinas con múltiples núcleos.
- Que el alumno sea capaz de comparar aplicaciones secuenciales y paralelas.
- Que el alumno sea capaz de convertir, cuando la situación lo amerite, aplicaciones secuenciales a paralelas de forma eficiente.

6. Contribución a los resultados (*Outcomes*)

Esta disciplina contribuye al logro de los siguientes resultados de la carrera:

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

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

7. Contenido

UNIDAD 1: Bases teóricas de la Computación Paralela y Distribuida (24)

Resultados del estudiante: 1,6

Contenido	**Objetivos Generales**

• ¿Porqué y para qué Computación paralela y Distribuida?
• Procesadores mutlinúcleo.
• Memoria compartida vs memoria distribuida.
• Multiprocesamiento simétrico.
• SIMD, procesamiento de vectores.
• GPU, coprocesamiento.
• Taxonomia de Flynn.
• 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)
• La ley de Amdahl: la parte de la computación que no se puede acelerar limita el efecto de las partes que si pueden.
	• Distinguir el uso de recursos computacionales para una respuesta mas rápida para administrar el acceso eficiente a un recurso compartido [Familiarizarse]
	• 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]
	• Usar herramientas de software para perfilar y medir el desempeño de un programa

Lecturas: Pacheco (2011), **Schmidt**, Kirk and Hwu (2013)

UNIDAD 2: Sistemas distribuídos (20)	
Resultados del estudiante: 1,6	
Contenido	**Objetivos Generales**
• Fallos: – Fallos basados en red (incluyendo particiones) y fallos basados en nodos – Impacto en garantías a nivel de sistema (p.e., disponibilidad) • Envío de mensajes distribuido: – Conversión y transmisión de datos – Sockets – Secuenciamiento de mensajes – Almacenando *Buffering*, renviando y desechando mensajes • Compensaciones de diseño para Sistemas Distribuidos: – Latencia versus rendimiento – Consistencia, disponibilidad, tolerancia de particiones • Diseño de Servicio Distribuido: – Protocolos y servicios Stateful versus stateless – Diseños de Sesión (basados en la conexión) – Diseños reactivos (provocados por E/S) y diseños de múltiples hilos • Algoritmos de Distribución de Núcleos: – Elección, descubrimiento	• Distinguir las fallas de red de otros tipos de fallas [Familiarizarse] • Explicar por qué estructuras de sincronización como cerraduras simples (*locks*) no son útiles en la presencia de fallas distribuidas [Familiarizarse] • Escribir un programa que realiza cualquier proceso de *marshalling* requerido y la conversión en unidades de mensajes, tales como paquetes, para comunicar datos importantes entre dos *hosts* [Usar] • Medir el rendimiento observado y la latencia de la respuesta a través de los *hosts* en una red dada [Usar] • Explicar por qué un sistema distribuido no puede ser simultaneamente Consistente (*Consistent*), Disponible (*Available*) y Tolerante a fallas (*Partition tolerant*). [Familiarizarse]
Lecturas: Pacheco (2011), **Schmidt**	

UNIDAD 3: Comunicación y coordinación (24)

Resultados del estudiante: 1,6

Contenido	**Objetivos Generales**
• 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) • Caminos críticos, el trabajo y la duración y la relación con la ley de Amdahl. • Aceleración y escalabilidad. • Productor-consumidor y algoritmos paralelos segmentados. • Ejemplos de algoritmos paralelos no-escalables.	• 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] • 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] • 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]

Lecturas: Pacheco (2011), **Schmidt**

UNIDAD 4: Programación para el Procesamiento Masivamente Paralelo (20)	
Resultados del estudiante: 1,6	
Contenido	**Objetivos Generales**
• GPU, coprocesamiento. • SIMD, procesamiento de vectores. • Gestión dinámica de memoria, aproximaciones y técnicas: malloc/free, garbage collection (mark-sweep. copia, referencia), regiones (también conocidas como arenas o zonas) • Programación y contención. • Consumo de energía y gestión.	• Explicar las caracteristicas de cada clasificación en la taxonomía de Flynn [Usar] • Describir las ventajas y limitaciones de GPUs vs CPUs [Usar] • Comparar los beneficios de diferentes esquemas de administración de memoria, usando conceptos tales como, fragmentación, localidad, y sobrecarga de memoria [Usar] • Describir como la distribuición/disposición de datos puede afectar a los costos de comunicación de un algoritmo [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]
Lecturas: Schmidt, Kirk and Hwu (2013)	

8. Metodología

1. El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.
2. El profesor del curso presentará demostraciones para fundamentar clases teóricas.
3. El profesor y los alumnos realizarán prácticas
4. Los alumnos deberán asistir a clase habiendo leído lo que el profesor va a presentar. De esta manera se facilitará la comprensión y los estudiantes estarán en mejores condiciones de hacer consultas en clase.

9. Evaluar 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.

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.

Sistema de Evaluación:
La nota final se obtiene a través de:

EVALUACIONES PERMANENTES	EVALUACIONES
Evaluación Permanente 1 : 24 % **Evaluación Permanente 2** : 36 %	**Evaluación Parcial** : 20 % **Evaluación Final** : 20 %
60%	40%

Donde:

Evaluación Permanente: Comprende trabajos grupales, participación activa en clase, test de ejercicios.

- Permanente 1 (Semanas 1 - 9)

- Permanente 2 (Semanas 10 - 17)

Para aprobar el curso, el alumno debe obtener 11.5 o más en la nota final.

References

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

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