TÍTULO: Computación paralela			
AÑO: 2020	**CUATRIMESTRE:** 1°	**N° DE CRÉDITOS:** 3	**VIGENCIA:** 3 años
CARGA HORARIA: 60 horas de teoría, 60 horas de práctica			
CARRERA/S: Doctorado en Astronomía, Doctorado en Física, Doctorado en Ciencias de la Computación			

FUNDAMENTOS

La tecnología de los microprocesadores modernos contiene un alto nivel de paralelismo. Para poder utilizar eficientemente estas arquitecturas se deben conocer los modelos de ejecución y las formas de sacar provecho a este paralelismo.

OBJETIVOS

Que el estudiante comprenda las tres dimensiones de paralelismo que actualmente posee una arquitectura de microprocesador: paralelismo de instrucciones (ILP), de datos (DLP) y de hilos (TLP), tanto en sus variantes de CPU como de GPU. Comprender las soluciones de compromiso de cada una de estas arquitecturas para obtener alto desempeño tanto en cálculo como en acceso a memoria. Saber discernir si un proceso está realizando un uso adecuado de todas las capacidades de la máquina. Al final de la materia los estudiantes deben ser capaces de adaptar programas a fin de utilizar estas tres dimensiones del paralelismo, tanto en CPU como en GPU.

PROGRAMA

Unidad 1: Introducción.

Escalado. Leyes de: Amdahl, Gustafson, Little. Eficiencia.

Factores que degradan el desempeño: inanición, latencia, sobrecarga, contención.

Paralelización: descomposición en tareas, orden y agrupamiento de tareas, descomposición de datos, datos compartidos.

Sincronización: condiciones de carrera, instrucciones atómicas. Primitivas de sincronización: mutexes, spinlocks, semáforos, barreras y fences.

Predicción de desempeño: modelo roofline. Medición de desempeño.

Unidad 2: CPU.

Paralelismo de instrucción (ILP): pipelining, procesadores superescalares, ejecución fuera de orden, SMT.

Memoria: jerarquía y asociatividad de cache, alineamiento de memoria, algoritmos cache-aware y cache-oblivious. Memoria virtual: efectos de la TLB en el desempeño. Memoria distribuida: NUMA, coherencia de cache. Afinidad de memoria y pinning de hilos a cores.

Vectorización: unidades SIMD, SSE intrinsics, técnicas de vectorización.

OpenMP: constructores work-sharing, atributos para compartir datos, planificadores, sincronización, entorno de ejecución, compilación.

Aplicaciones: extensiones ISA específicas para aplicaciones, bibliotecas para HPC.

Unidad 3: GPU.

Arquitectura interna.

Limitaciones de la GPGPU: serialización de saltos, ocultamiento de latencia, ocupación.

Jerarquía de memoria, cache de software vs. cache de hardware, unidades de textura.
CUDA: mapeo hilo-dato, lanzamiento de kernels, comunicación host-device, sincronización, contadores de desempeño y profiling, manejo de errores, compute capabilities, PTX ISA.
Optimización: aumento de la granularidad de los hilos, uso efectivo de la memoria compartida, código sin saltos, double buffering, reducción del uso de registros, aritmética de precisión mixta, cómo evitar instrucciones atómicas.
Ejemplos de algoritmos GPU: reducción, scan segmentado, compactación de streams, usos.
Bibliotecas: CUBLAS, CUFFT, CUSPARSE, Thrust, CUDPP, CUB.

PRÁCTICAS

Clases de laboratorio con exposición de temas puntuales y trabajo en grupo donde en los posible se intenta combinar diferentes perfiles de formación.

BIBLIOGRAFÍA

B. Chapman, G. Jost, R. van der Pas, Using OpenMP: Portable Shared Memory Parallel Programming, 2007.
D. B. Kirk, Wen-mei W. Hwu, Programming Massively Parallel Processors, 2nd edition, 2012.
NVIDIA Inc., CUDA C Programming Guide, versión CUDA 9.1, 2017.
NVIDIA Inc., CUDA C Best Practices, versión CUDA 9.1, 2017.
NVIDIA Inc., PTX ISA 6.1, versión CUDA 9.1, 2017.
J. Hennessy, D. Patterson, Computer Arquitecture: A Quantitative Approach, 5th edition, Morgan Kaufmann, 2011.
J. Hennessy, D. Patterson, Computer Organization and Design: The hardware / Software Interface, 5th edition, Morgan Kaufmann, 2013.

MODALIDAD DE EVALUACIÓN

El alumno deberá elegir un programa de computación numérica intensiva que será paralelizado de cuatro formas:
1. ILP, cache-aware.
2. SIMD (instrucciones vectoriales).
3. Multicore (típicamente OpenMP para CPU).
4. Manycore (típicamente CUDA para GPU).

Se deberá entregar un informe final donde se comparen las mejoras obtenidas. En el primer punto se deberá analizar la mejor utilización de las unidades de ejecución, y la mejora en las tasas de cache-hit. En el segundo caso, la mejora que se obtuvo al operar de manera vectorial sobre los datos y la estrategia de paralelización utilizada, así como un análisis de scaling respecto al ancho de la unidad vectorial (AVX vs AVX512). Para multicore CPU además de analizar el scaling con respecto a la cantidad de cores, se deberá informar sobre los efectos de la afinidad de memoria-cpu. Finalmente para manycore GPU se harán análisis de weak-scaling y de utilización del ancho de banda de memoria y potencia de cálculo respecto al pico teórico.

REQUERIMIENTOS PARA EL CURSADO

Conocimientos de programación numérica.