UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

Taller de programación sobre GPUs

Carrera: Ingeniería en Computación
Profesor Responsable: Pousa, Adrián
Año: Optativa
Duración: Semestral
Carga Horaria Semanal: 6hs
Carga Horaria Total: 96hs

OBJETIVOS GENERALES

Profundizar el conocimiento de las arquitecturas tipo GPU y su programación.
Comparar su performance con los multicores convencionales.
Analizar los modelos de resolución de problemas específicos.
Introducir conceptos de consumo y green computing a partir de la utilización de GPUs.

PROGRAMA

Unidad 1: GPU: Introducción a HPC y GPGPU

- Introducción al cómputo de altas prestaciones (HPC).
- Paradigmas de Computación Paralela en GPUs: Modelo de Memoria Compartida, Modelo de Memoria Distribuida. Paralelismo de Datos y Paralelismo Funcional.
- Taxonomía de Flynn.
- Arquitecturas y herramientas para HPC.
- Introducción a la arquitectura GPU y su uso en HPC.
- GPGPU: Computación de Propósito General en GPU.

Unidad 2: Arquitecturas GPU - Modelo GPU-CPU

- Evolución de las GPUs.
- Arquitecturas Nvidia.
- Arquitecturas ATI-AMD.
- Arquitecturas Xeon-Phi.
- Modelo de interacción GPU-CPU.
- Introducción a la planificación de hilos en GPU Nvidia - Concepto de Grid, Bloque, Thread y Warp.
- Rendimiento y consumo de las arquitecturas GPU según Top500 y Green500.

Unidad 3: Modelo de Programación GPU - Resolución de aplicaciones

- Modelo de programación en GPU.
- Relación con SIMD, modelo SIMT.
- Modelo de programación CUDA.
    - Concepto de Host y Device. Identificadores.

  - Tipos de datos. o Definición de Constantes.
  - Variables: alcance y tiempo de vida.
  - Gestión de memoria, copia explicita CPU-GPU y GPU-CPU, Síncrona y Asíncrona.
  - Gestión de Hilos: Grid, Bloques, Threads. Dimensiones: 1D, 2D y 3D.
  - Kernel, llamados Síncronos y Asíncronos.
  - Funciones.
  - Identificadores de Threads y Bloques.
  - Planificación de Threads.
  - Sincronización de Threads.
- Modelo de programación OpenCL.
  - Arquitecturas con soporte OpenCL. Conceptos básicos de OpenCL Context, WorkQueue, WorkItems, Kernels.
  - Equivalencias OpenCL – CUDA.
- Diseño de programas en GPU.
- Estudio experimental de casos.
- Métricas de rendimiento: speedup y eficiencia.
- Métricas de consumo y eficiencia energética: Watt/flop.
- Análisis de perfomance. Aceleración en GPU con respecto a CPU.

Unidad 4: Modelo y jerarquía de Memoria de GPU

- Modelo de Memoria de GPU.
- Jerarquía de Memoria: Registros, Memoria Compartida, Memoria de constantes, Memoria de Texturas, Memoria Global. Memorias Cache: Constantes, Texturas, Nivel 1, Nivel 2.
- Patrones de Acceso a Memoria Global, relación entre segmentos y cantidad de transacciones.
- Patrones de Acceso a Memoria Compartida, bancos de memoria, conflicto de bancos, accesos sin conflictos.
- Concepto de Acceso Coalescente.
- El problema de la latencia. Unidad 5: Optimizaciones
- Divergencia.
- Coalescencia y prefetching.
- Mezcla y granularidad de instrucciones.
- Asignación de recursos. Unidad 6: Multi-GPUs y Arquitecturas Híbridas.
- Maquinas con más de una GPU.
- Arquitectura Híbrida Multicore-GPU: Integración de herramientas CUDA – OpenMP/Pthreads.
- Arquitectura Híbrida Cluster-GPU: Integración de herramientas CUDA - MPI.
- Arquitectura Híbrida Cluster-Multicore-GPU: Integracion de herramientas CUDA - OpenMP/Pthreads – MPI.

- Heterogeneidad – Balance de carga.
- Análisis de perfomance. Aceleración en GPU con respecto a Arquitecturas Multicores y Clusters.
- Casos de Estudio. Programación de aplicaciones

BIBLIOGRAFIA

Por las características de la asignatura se utiliza como único material de consulta los contenidos disponible en el sitio https://developer.nvidia.com de Nvidia Corporation.

DESCRIPCIÓN DE LAS ACTIVIDADES TEÓRICAS Y PRÁCTICAS

Las clases teórico-prácticas son dictadas por los Profesores de la asignatura y son obligatorias para la promoción.

Las explicaciones de práctica son introductorias al trabajo en Laboratorio, para facilitar la utilización del equipamiento y software por los alumnos. Se desarrollan en las clases teórico-prácticas.

METODOLOGÍA DE ENSEÑANZA Y EVALUACIÓN

METODOLOGÍA DE ENSEÑANZA Modalidad presencial

La asignatura se estructura con clases teórico-prácticas y prácticas experimentales.

- Las clases teórico-prácticas son dictadas por los Profesores de la asignatura y son obligatorias para la promoción.
- Las explicaciones de práctica son introductorias al trabajo en Laboratorio, para facilitar la utilización del equipamiento y software por los alumnos. Se desarrollan en las clases teórico-prácticas.
- El Taller propone el desarrollo de trabajos concretos con arquitecturas GPU y combinaciones de multicores y GPUs. Las actividades de Taller se hacen en máquina, en el contexto de las clases teórico-prácticas.
- Las consultas y correcciones son realizadas en forma presencial.
- En principio se utilizará la Sala de Cómputo de Postgrado (por la disponibilidad de placas GPU) y equipamiento especial del III-LIDI (Instituto de Investigación de la Facultad)

METODOLOGÍA DE ENSEÑANZA Modalidad semipresencial

Se hace notar que por la característica de las tareas experimentales, el alumno deberá tener acceso a algún modelo de arquitectura paralela y contar con alguna GPU para poder realizar los trabajos que se solicitan en el curso.

El alumno puede seguir los temas por el entorno WEB-UNLP y asistir a las consultas que se fijen para los alumnos presenciales.

EVALUACIÓN Modalidad presencial

Para obtener la aprobación de cursada de la asignatura los alumnos deben aprobar todas las entregas de los diferentes trabajos experimentales, estas entregas pueden ser en grupos de 2 personas. Los trabajos no tienen reentregas. Además de las entregas los alumnos deben aprobar un examen parcial para el que se dispone de una fecha y dos recuperatorios.
Para la aprobación final de la asignatura se les propondrá un trabajo final experimental que deberán defender en un coloquio en una fecha de examen final.

EVALUACIÓN Modalidad semipresencial

Deben cumplir con los mismos requisitos que los alumnos en modalidad presencial.