

IMT2112 - Algoritmos Paralelos en Computación Científica

Programar en tarjetas gráficas

Elwin van 't Wout

26 de noviembre de 2019



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Facultad de Matemáticas • Escuela de Ingeniería

imc.uc.cl

Clase previa

- Sistemas heterogéneos de computadores

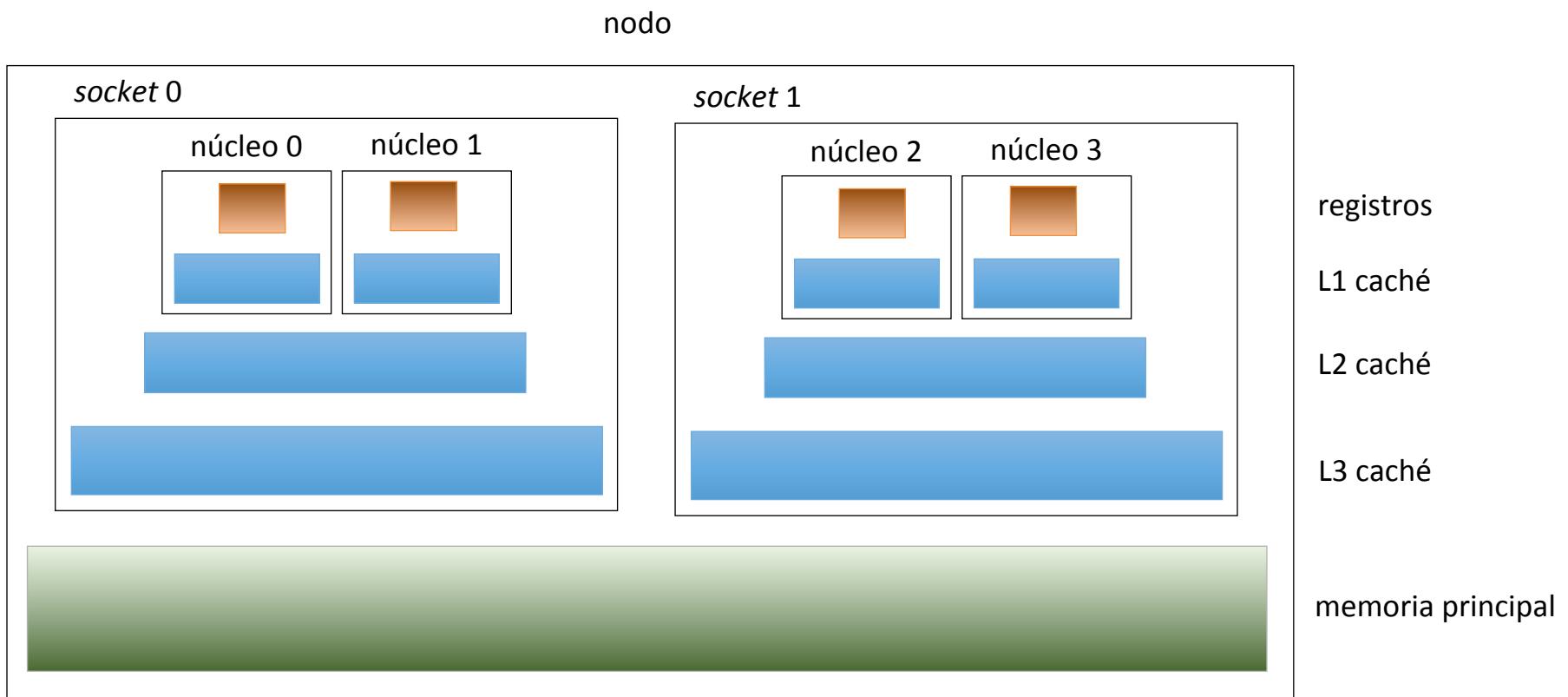
Agenda

- ¿Como programar en computadores con arquitectura heterogénea?

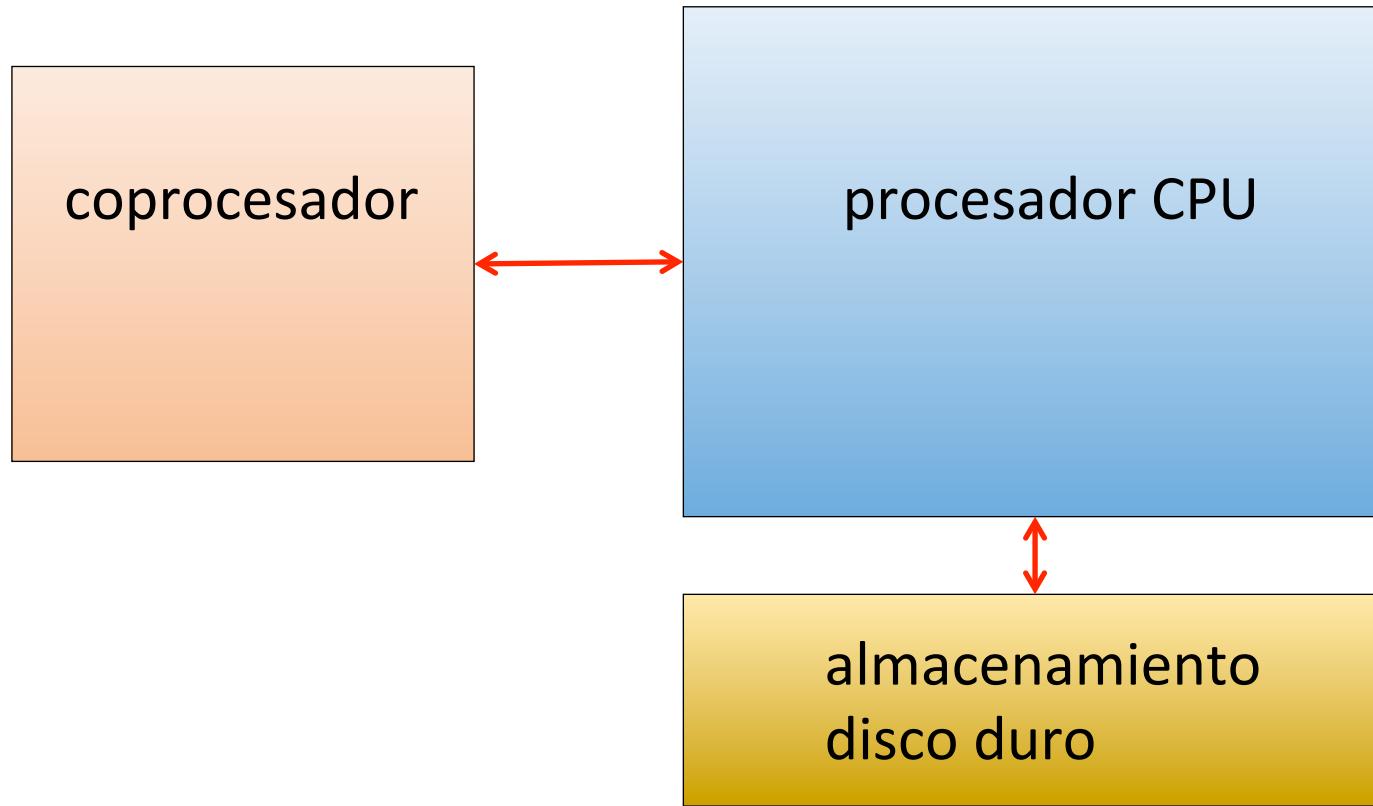
Arquitectura heterogénea

- Una arquitectura heterogénea es un sistema computacional con distintos tipos de dispositivos de cálculo
- Es cada día mas usado en supercomputadores y ya tiene una historia larga de desarrollo

Los procesadores *multi-core*



Sistemas con coprocesadores



Sistemas heterogéneos de computación

- Debido al diseño del hardware, los procesadores pueden tener diferentes características
 - el CPU es bueno para realizar instrucciones difíciles y de alto nivel, incluidas el I/O (*input* y *output*)
 - los coprocesadores son buenos para realizar cálculos de bajo nivel en paralelo
- Ejemplos de coprocesadores
 - tarjetas gráficas (Nvidia, AMD, etc.)
 - Intel Xeon Phi

Tarjetas gráficas para la computación científica

- Tarjetas gráficas fueron diseñados para la visualización
 - calcular y mostrar el valor RGB (red-green-blue) de cada pixel en la pantalla
- Científicos comenzaron de usar GPUs para la computación científica
 - hacer cálculo de métodos que son conveniente paralelo
- Fabricantes comenzaron de diseñar GPUs para la computación de alto rendimiento
 - p.ej. la línea de NVIDIA Tesla
 - a menudo vendido para la inteligencia artificial y *deep learning*

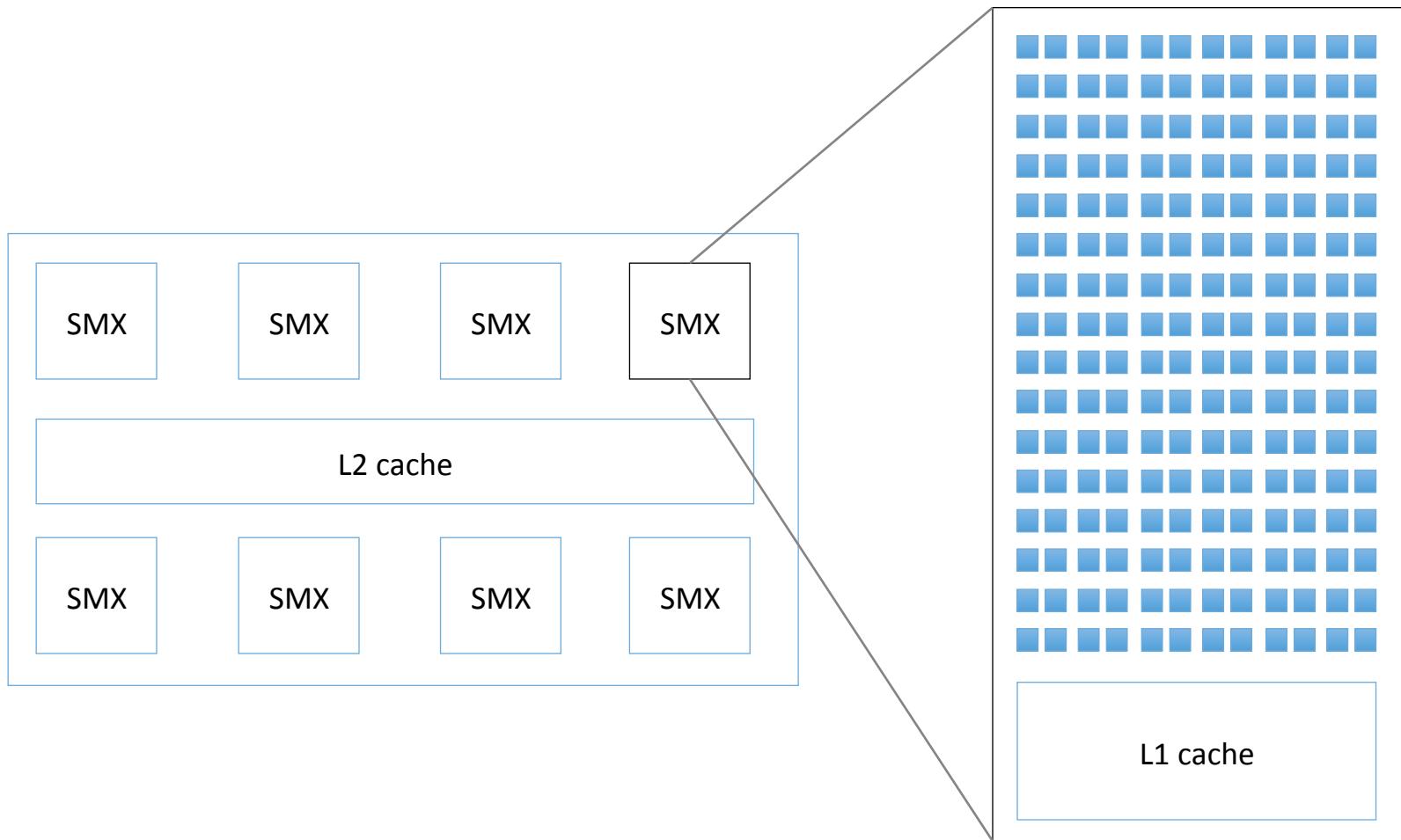
Tarjetas gráficas para la computación científica

- Tarjetas gráficas normalmente son usados como coprocesador
 - los GPGPUs (*General-Purpose GPU*) no requieren un *host*
- Ejemplos de tarjetas gráficas
 - Nvidia GeForce para procesamiento de gráficos
 - \$50.000 - \$1.000.000 CLP
 - Nvidia Tesla para HPC
 - Tesla K20: 2496 núcleos, 5 GB (\$300 USD)
 - Tesla V100: 5120 núcleos, 16 GB (\$6.000 USD)
 - Requieren un computador de alto rendimiento

Tarjetas gráficas para la computación científica

- Las características principales de las tarjetas gráficas
 - un gran cantidad de núcleos
 - ejecución simultánea de hilos (SIMD)
 - registros privados por cada hilo
 - un tamaño limitado de memoria
 - requiere comunicación con un *host* CPU
- Implicancias para computación en GPU
 - require un alto grado de paralelización
 - una granularidad en el nivel de operaciones
 - intensidad aritmética alta

Arquitectura de tarjetas GPU (NVIDIA Kepler)



Arquitectura de tarjetas GPU

- Los componentes básicos de las GPUs son multiprocesadores de transmisión (SMs - *streaming multiprocessors*)
 - procesadores con muchos núcleos, p.ej. 196 núcleos
 - caché L1 compartida, p.ej. 64 KB
 - hasta 2000 hilos
- Los hilos se ordenan en bloques de hilos
 - todos los hilos en el mismo bloque realizan la misma instrucción
 - diferentes bloques de hilos pueden realizar diferentes instrucciones
 - todos los hilos tienen datos privados y compartidos

Programar en tarjetas gráficas

Literatura:

- B.R. Gaster, L. Howes, D. Kaeli, P. Mistry, and D. Schaa. “Heterogeneous Computing with OpenCL,” Morgan Kaufmann, Waltham, MA, 2012.
- A. Munshi, B.R. Gaster, T.G. Mattson, J. Fung, and D. Ginsburg. “OpenCL Programming Guide,” Addison-Wesley, Boston, MA, 2012.
- NVIDIA. “CUDA C Programming Guide”, 2018.

Programación de GPU

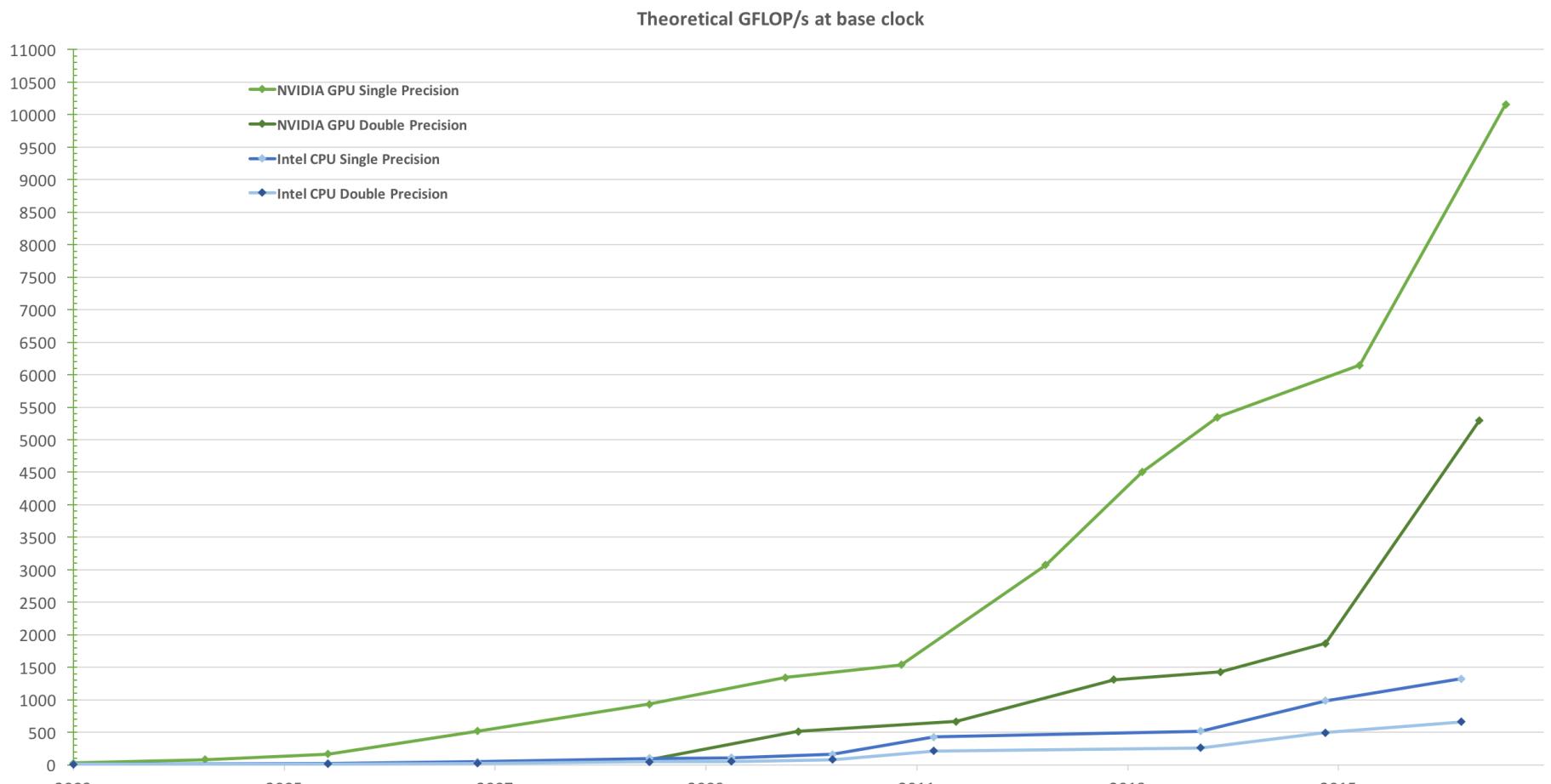
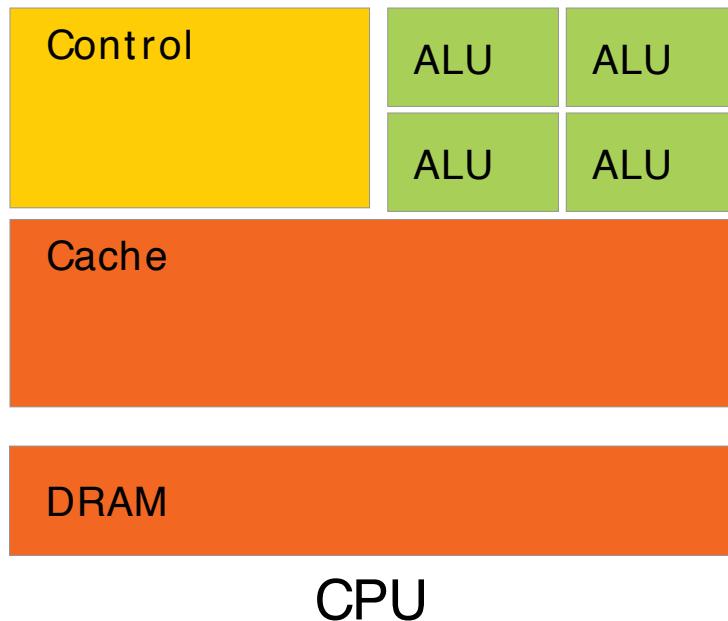


Figura de “CUDA programming guide.”

Arquitectura de tarjetas GPU

- El número de transistores es el factor limitante para ambas la CPU y GPU
- Las GPUs priorizan los transistores para cálculo y procesamiento de datos en lugar de almacenamiento en caché de datos y control de flujo
- Ejemplo de Nvidia Tesla V100 (Volta)
 - 815 mm² dimensión del chip
 - 21.1 mil millones de transistores
 - 16 GB de memoria
 - 1530 MHz velocidad de reloj *boost*
 - 80 *streaming multiprocessors*
 - 5120 núcleos de cálculo

Programación de GPU



ALU - *Arithmetic Logit Unit*

Figura de “CUDA programming guide.”

Programación de GPU

- El dispositivo GPU está conectado a un *host* de CPU
 - para GPGPU, generalmente hay una pequeña CPU integrada en la tarjeta GPU
- El hilo maestro en la CPU realiza lo siguiente:
 1. Inicializar dispositivo de cómputo
 2. Definir dominio del problema
 3. Asignar memoria en *host* y dispositivo
 4. Copiar datos del *host* al dispositivo
 5. Iniciar el *kernel* de ejecución en el dispositivo
 6. Copiar datos del dispositivo al *host*
 7. Desasignar toda la memoria

Programación de GPU

- Las llamadas a los coprocesadores suelen estar bloqueando
 - la CPU espera hasta que la GPU haya terminado los cálculos
- Se puede usar llamadas sin bloqueo a la GPU
 - recuerde que no tiene control sobre la velocidad de cálculo
 - la superposición de cómputo de CPU y GPU puede mejorar la eficiencia

Programación de GPU

- Interfaces de programación especiales son necesarias para la computación GPU
 - CUDA (*Compute Unified Device Architecture*) es una extensión C / C++ para GPUs de Nvidia
 - OpenCL (*Open Computing Language*) es un estándar de programación abierto para arquitectura heterogénea
 - OpenACC (*Open Accelerators*) es un estándar de programación para computación CPU / GPU con directivas de compilación

Programación de GPU

- Para grandes equipos heterogéneos, normalmente se necesita múltiples interfaces de programación
 - MPI para la paralelización de memoria distribuida
 - OpenMP / CUDA / OpenCL para cada nodo de proceso
- Hay bibliotecas de código abierto para extender OpenCL a la memoria distribuida
 - desarrollado y mantenido por grupos de investigación
 - sin apoyo oficial de una organización comunitaria

Programación de GPU

- La interfaz de programación CUDA
 - código de propiedad de la empresa Nvidia, sin cargo
 - solo funciona para GPUs fabricadas por Nvidia
 - interfaz de programación de bajo nivel que es estable, rápida y relativamente fácil de usar para desarrolladores de C++

Programación de GPU

- La interfaz de programación OpenCL
 - estándar de software libre de uso y código abierto
 - coordinado por *The Khronos Group*, que es una organización sin fines de lucro para los estándares de software abiertos
 - estas normas son apoyadas por muchas universidades y empresas de todo el mundo
 - los proveedores incluyen NVIDIA, AMD, Intel y Apple
 - basado en software C, con una extensión C ++
 - apto para computación paralela de propósito general en CPU, GPU y otros procesadores
 - diseñado para sistemas heterogéneos

Programación de GPU

- Hay bibliotecas en Python que proveen una interfaz a CUDA y OpenCL
 - PyCUDA
 - PyOpenCL
- Son bibliotecas desarrollados por grupos de usuarios
 - no tiene soporte oficial de organizaciones o fabricantes

Programación de GPU

- El hilo maestro en la CPU realiza lo siguiente:
 1. Inicializar dispositivo de cómputo
 2. Definir dominio problemático
 3. Asignar memoria en host y dispositivo
 4. Copie datos del host al dispositivo
 5. Inicie el kernel de ejecución en el dispositivo
 6. Copiar datos del dispositivo al host
 7. Desasignar toda la memoria

Programación de GPU

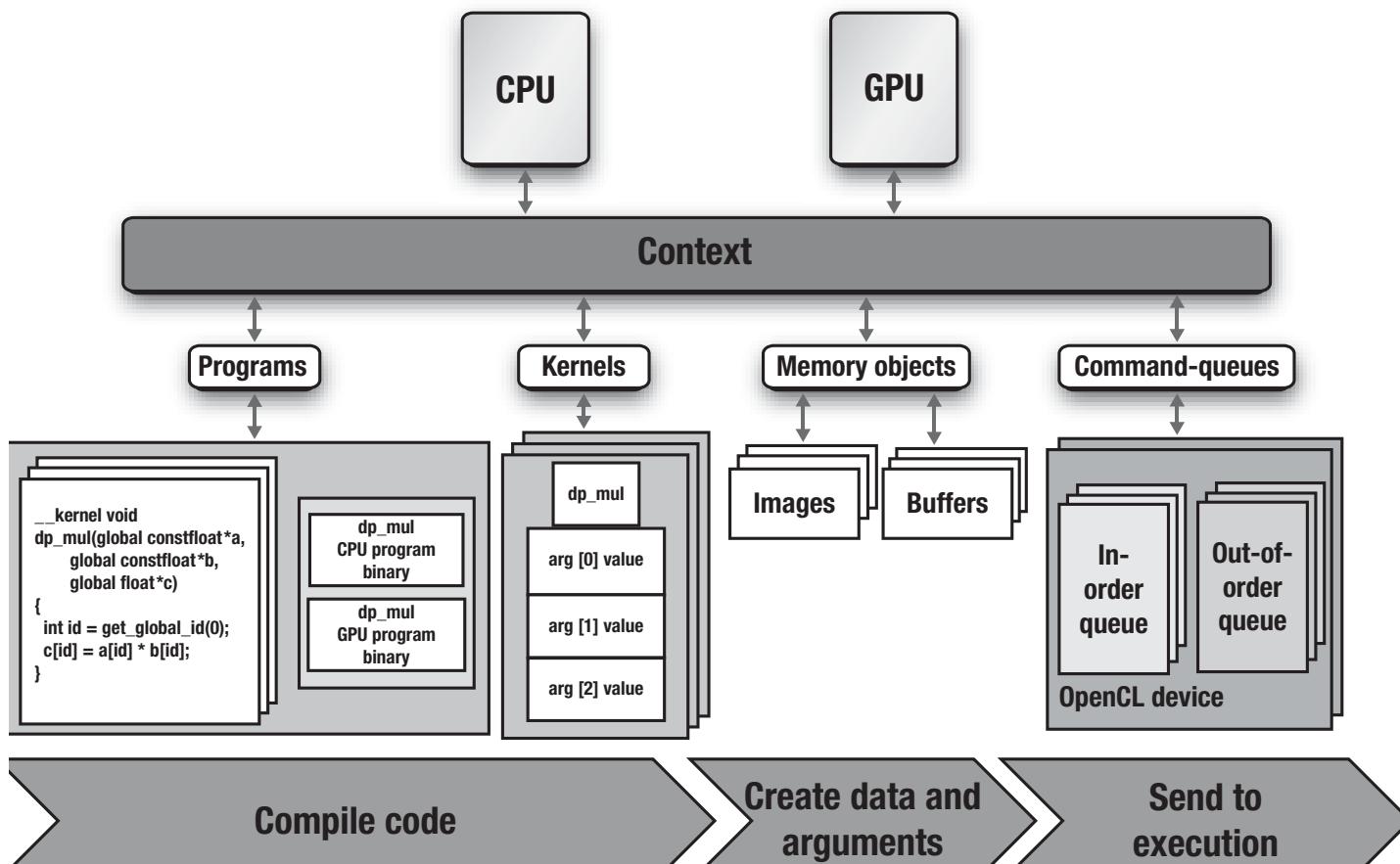


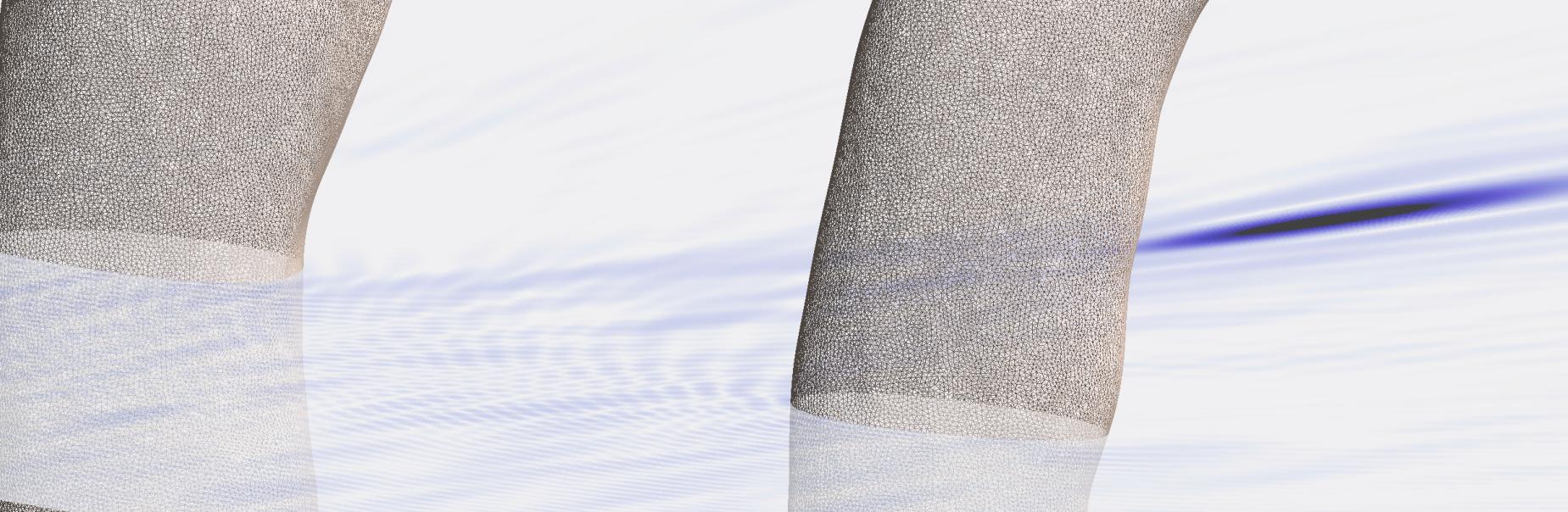
Figura de “OpenCL programming guide.”

Resumen

- Programar en tarjetas gráficas
- Programar con OpenCL

Clase siguiente

- Programar algoritmos en OpenCL



IMT2112 - Algoritmos Paralelos en Computación Científica

Programar en tarjetas gráficas

Elwin van 't Wout

26 de noviembre de 2019



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Facultad de Matemáticas • Escuela de Ingeniería

imc.uc.cl