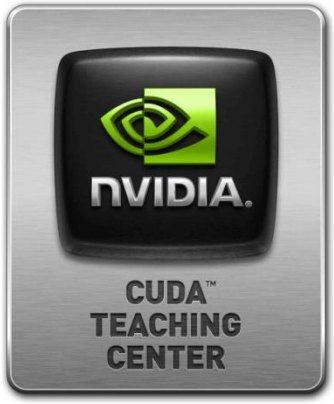
Programación masivamente paralela en procesadores gráficos (GPUs)

E. Dufrechou, P. Ezzatti y M. Pedemonte



Clase 8
OpenCL

OpenCL

Introducción

- OpenCL: Open Computing Language
- Es un estándar abierto que permite sacar partido de plataformas heterogéneas de cómputo:
  - CPU multicore
  - GPUs
  - Digital Signal Processors (DSPs)
  - Field-Programmable Gate Arrays (FPGAs)
  - Etc.

Introducción

- Desarrollado desde el 2009 por el grupo Khronos que nuclea a (entre otros):
  - Altera (fabricante de FPGAs adquirida por Intel)
  - AMD
  - Apple (en el último tiempo ha perdido interés)
  - ARM
  - Creative
  - IBM
  - Intel
  - Nvidia
  - Qualcomm
  - Samsung

Introducción

Del estándar destaca que:

- **Soporta tanto paralelismo de datos como de tareas.**
- **Está basado en el lenguaje C/C++.**
- **Define perfiles de configuración para manejar las diferentes unidades de procesamiento.**

Arquitectura

Framework para programación paralela que incluye:

- **Lenguaje**
- **API**
- **Bibliotecas**
- **Runtime system**

Se basa en una jerarquía de modelos:

- **De plataforma**
- **De memoria**
- **De ejecución**
- **De programación**

Modelo de plataforma

- El modelo consiste en un host conectado a uno o varios OpenCL devices.
- Cada OpenCL device se divide en una o más compute units (CUs) .
- Cada CUs se divide en processing elements (PEs)
- Los cómputos en los devices son realizados realmente por los PEs.

Modelo de plataforma

- Las aplicaciones mandan commands desde el host a los devices.
- Cada CU ejecuta un único stream de instrucciones (al estilo SIMP, SPMD o SPMT).
- Identificadores importantes
  - La versión de la plataforma
  - La versión del device
  - Versión del OpenCL C soportado

Modelo de ejecución

- **La ejecución de un programa OpenCL se divide en 2 partes:**

–**Kernels** que ejecutan en uno o más devices.

–**Host program** que ejecuta en el host.

- **Una instancia de un kernel se llama work-item:**

–Se identifica por su índice (1,2,3-dimensional)

- **Los work-items se agrupan en work-groups:**

–Se identifican por un índice único (1,2,3-dimensional)

Modelo de ejecución

- **El host define un contexto para la ejecución de los kernels que incluye:**

–**Devices**: la colección de los devices utilizados por el host.

–**Kernels**: las funciones OpenCL a ejecutar en los devices.

–**Program Objects:** el programa que implementa los kernels.

–**Memory Objects:** conjunto de espacios de memoria visibles por el host y los devices.

- **Para ejecutar los kernels utiliza una command-queue**

Modelo de ejecución

- **La ejecución es asincronica entre el host y los devices.**
- **Los comandos pueden ejecutar:**

–**En orden:** los comandos son lanzados en orden y esperan que se complete el comando anterior para ejecutar.

–**Sin orden:** los comandos son lanzados en orden pero no esperan por la finalizacion de los otros comandos.

Modelo de ejecución

Categoria de kernels

OpenCL soporta 2 categorias de kernels:

–OpenCL: escritos y compilados en OpenCL.

–Nativos: accedidos mediante un puntero a función en el host.

Modelo de memoria

Los work-items pueden acceder a 4 regiones de memoria:

- **Memoria Global**: permite lectura y escritura por todos los work-items. Puede tener caché, dependiendo de las capacidades del dispositivo.
- **Memoria Constante**: solamente de lectura.
- **Memoria Local**: memoria privada de un work-group.
- **Memoria Privada**: memoria privada de un work-item.

El host solo puede acceder a la memorias global y constante !!

Modelo de memoria

Las trasferencias de datos entre el host y los devices pueden ser:

- Bloqueantes
- No bloqueantes

Modelo de programación

El modelo soporta 2 paradigmas

–Paralelismo de datos

–Paralelismo de tareas

El modelo central es el paralelismo de datos !!

Modelo de programación

- Ofrece un modelo de paralelismo de datos relajado.
- Modelo de paralalelismo de datos jerárquico, con 2 niveles.
  - Explícito: se especifican la cantidad de work-groups y work-items por work-groups.
  - Implícito: se especifica únicamente la cantidad de work-items. Mapea automáticamente a unidades SIMD.

Modelo de programación

Sincronizaciones

–A nivel de work-item en un work-group.

–A nivel de comandos encolados en una command-queue en un contexto.

Componentes

- Capa de plataforma, permite descubrir los devices y sus capabilities.
- Runtime, permite manipular los contextos.
- Compilador, permite compilar programas OpenCL.

Buscando parecidos

CUDA	OpenCL
Hilo	work-item
bloque	work-group
Cuda core	processing elements (PEs)
multiprocessor	compute units (CUs)
Memoria compartida	Memoria local
Registros/Memoria local	Memoria privada
streams	Command-queue