CUDA Threads

Programación Paralela

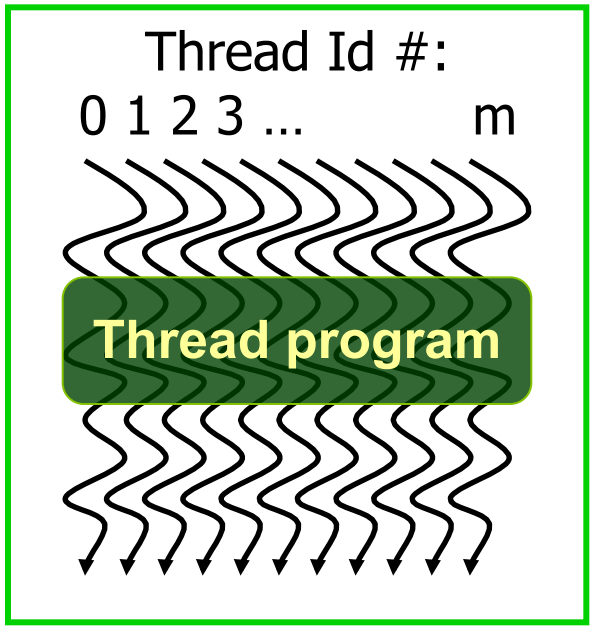
José María Cecilia

Grado en Ingeniería informática

CUDA Thread Block

- Todos los hilos de un bloque ejecutan el mismo kernel (SPMD)
- El programador declara bloques:
  - El tamaño de bloque puede ser de 1 a 1024 threads concurrentes (consultar device query)
  - El diseño del bloque puede ser 1D, 2D, or 3D
  - Las dimensiones de bloque son en hilos
- Los hilos tienen thread id dentro del bloque:
  - El programa para cada hilo usa thread id para seleccionar los datos con los que va a trabajar.
  - Hilos de un mismo bloque puede compartir datos eficazmente, y sincronizarse.
- Hilos en diferentes bloques no pueden cooperar:
  - Cada bloque puede ejecutar en cualquier orden relativo a otros bloques

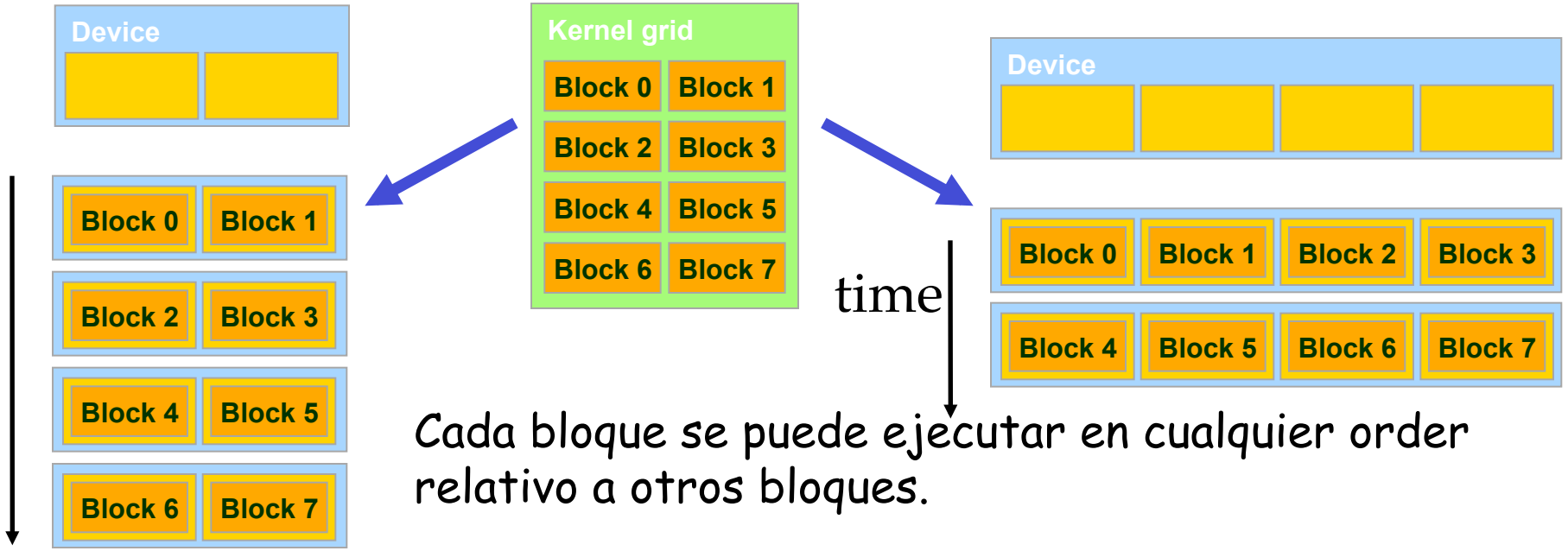
CUDA Thread Block



Courtesy: John Nickolls, NVIDIA

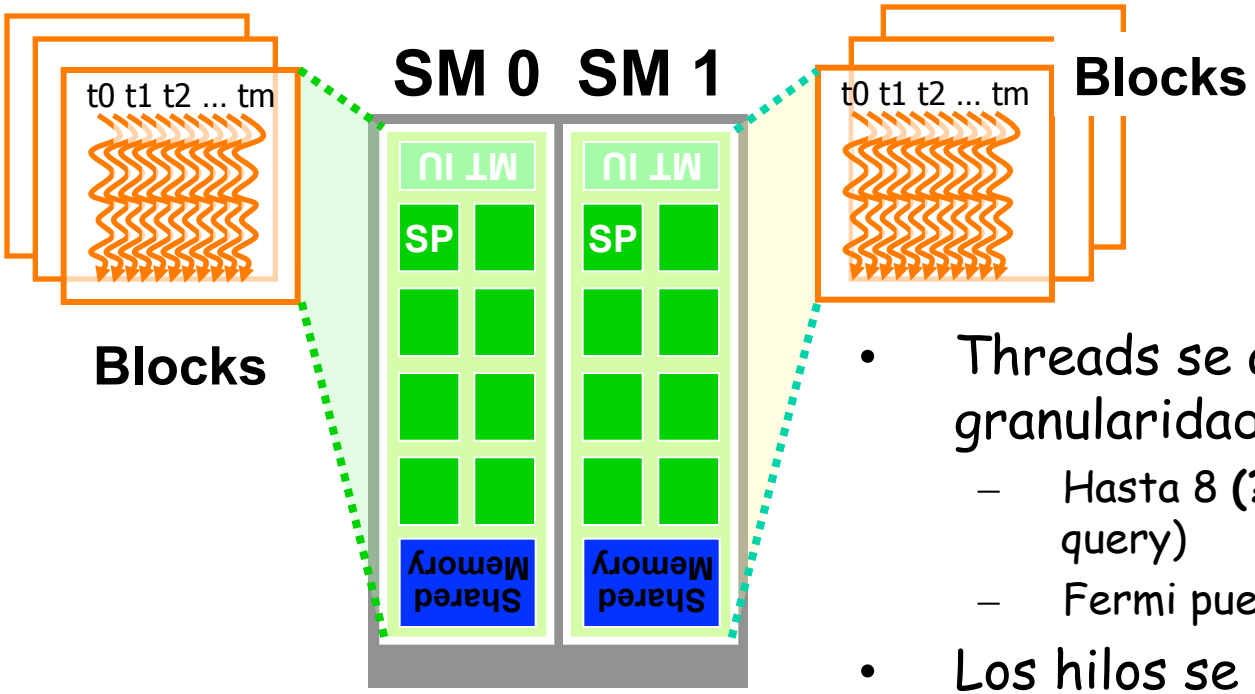
Escalabilidad Transparente

- Hardware es libre de asignar bloques a cualquier multiprocesador en cualquier momento.
- Un kernel escala junto con el número de SMs.



Cada bloque se puede ejecutar en cualquier order relativo a otros bloques.

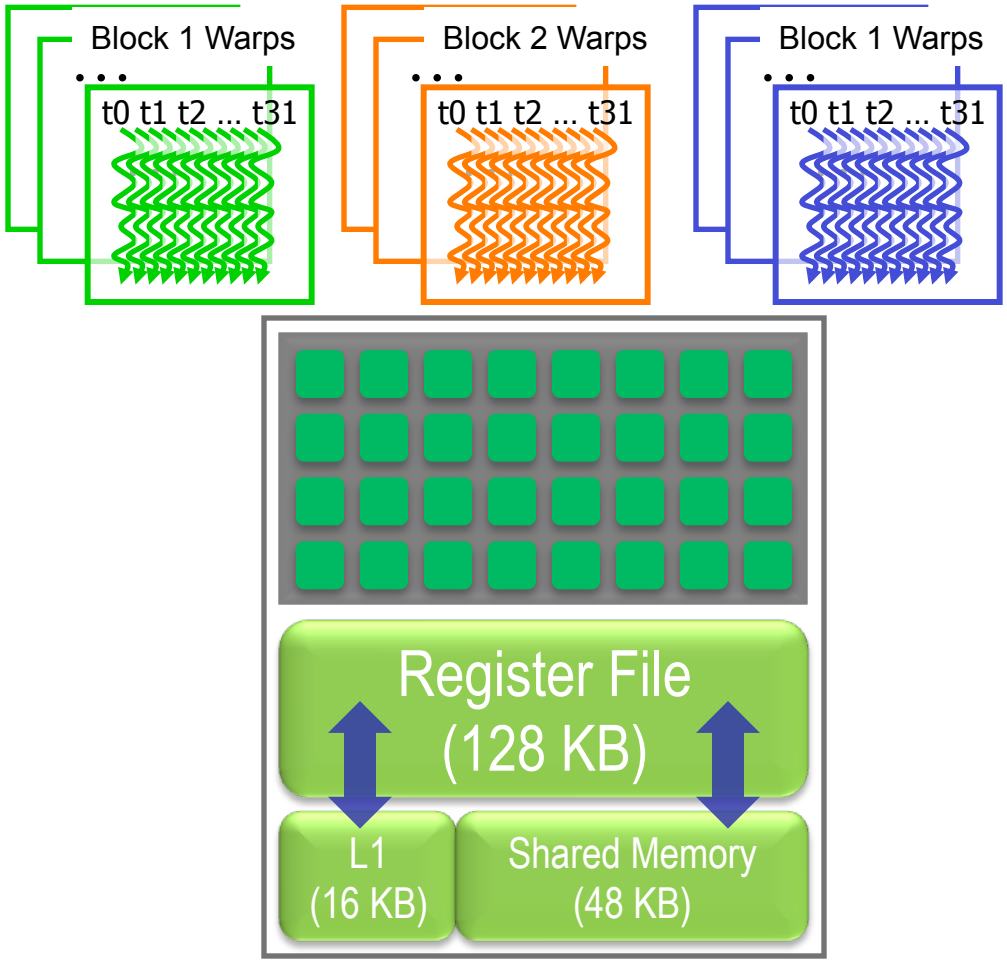
Ejemplo: Ejecutando Thread Blocks



- Threads se asignan a SMs con una granularidad de Blocks
  - Hasta 8 **(?) bloques** para cada SM (device query)
  - Fermi puede tener hasta **1536** threads
- Los hilos se ejecutan concurrentemente
  - SM mantiene thread/block id #s
  - SM maneja/planifica la ejecución de los thread

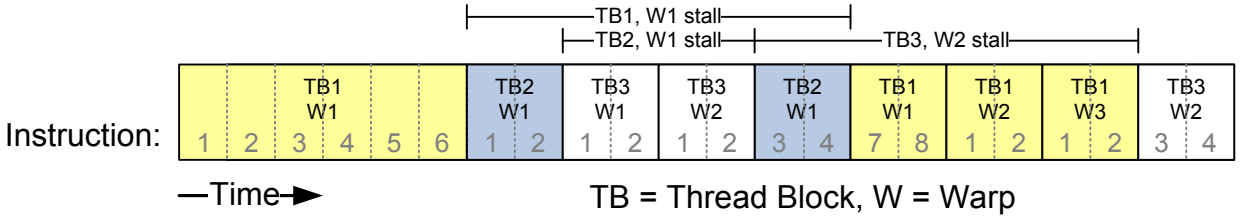
Ejemplo: Planificación de Threads

- Cada bloque es ejecutado en bloques de 32 llamados Warps
  - Decisión de implementación. No es parte de CUDA.
  - Es la unidad de planificación en cada SM.
- Si 3 blocks son asignados a cada SM, y cada bloque tiene 256 threads, ¿Cuántos Warps hay en un SM?
  - Cada bloque se divide en 256/32 = 8 Warps
  - Tenemos 8 * 3 = 24 Warps



Ejemplo: Planificación de Threads (cont)

- El scheduling entre Warps es gratis
  - Cada instante, 1 o 2 de los warps son ejecutados en cada SM.
  - Warps cuya próxima instrucción tiene sus operandos listos son seleccionados para la ejecución.
  - Los Warps elegibles son seleccionados en una planificación con políticas de prioridad.
  - Todos los hilos en un Warp ejecutan la misma instrucción cuando son seleccionados (SIMD).



Granularidad de bloques. Algunas consideraciones

- En la multiplicación de matrices con bloques, ¿Debería usar bloques de 8X8, 16X16 o 64X64 blocks?
- Bloques de 8X8, tenemos 64 threads por Bloque. Como cada SM puede reservar 1536 hilos (FERMI), podemos alojar hasta 24 bloques. Sin embargo cada SM, solo puede tener 8 bloques (limitación física), por tanto sólo 512 hilos irán a cada SM.
- Bloques de 16X16, tenemos 256 threads por Bloque. Como cada SM puede reservar 1536 hilos (FERMI), podemos alojar hasta 6 bloques, y por tanto conseguir un uso total de los recursos. A menos que haya otros factores que lo delimiten.
- Bloques de 64X64, tendríamos 4096 threads por Bloque. Cada bloque sólo puede tener hasta 1024 threads. Esta configuración no funcionaría.

Algunas Características del API Adicionales

Application Programming Interface (API)

EL API es una extensión del lenguaje C

- **Consiste en:**
  - **Extensiones de lenguaje**

Para indicar qué partes del código se ejecutan en la GPU.

  - **Librería de runtime se divide en:**
    - **Componentes comunes:** Brindan tipos de datos vectores y subconjuntos de la librería de runtime de C para usarlos en ambos: device and host
    - **Componentes host:** Para controlar el acceso a uno o más devices desde el host.
    - **Componentes device:** Funciones específicas del device

Extensiones del lenguaje: Built-in Variables

```
dim3 gridDim;
```

Dimensiones del grid en bloques (`gridDim.z`)

```
dim3 blockDim;
```

Dimensionds del bloque en threads

```
dim3 blockIdx;
```

Índice del bloque dentro del Grid

```
dim3 threadIdx;
```

Índice del thread dentro del bloque

Componentes del Runtime: Funciones Matemáticas

```
pow, sqrt, cbrt, hypot
exp, exp2, expm1
log, log2, log10, log1p
sin, cos, tan, asin, acos, atan, atan2
sinh, cosh, tanh, asinh, acosh, atanh
ceil, floor, trunc, round
```

Etc.

- Cuando se ejecutan en el host, la función usa el runtime C
- Estas funciones sólo soportan tipos escalares, no vectores.

Componentes del Runtime para el device: Funciones matemáticas

Algunas funciones matemáticas (e.j. `sin(x)`) tienen menos precisión pero son mas rápidas en su versión de GPU (e.j.`__sin(x)`)

`__pow`

`__log, __log2, __log10`

`__exp`

`__sin, __cos, __tan`

Componentes del Runtime para el Host

- Tiene funciones para tratar con:

Manejo de Devices (incluyendo sistemas multi-device)

Manejo de Memoria

Manejo de Errores

- Se inicializa la primera vez que una función del runtime es llamada.
- Un hilo host puede invocar un kernel en un sólo device.
- Múltiples hilos de CPU requieren varias GPUs.

Componente del Device Runtime: Función de sincronización

```
void __syncthreads();
```

Sincroniza todos los hilos de un bloque.

- Una vez que todos los hilos han alcanzado este punto, la ejecución sigue normalmente.
- Usado para evitar los problemas RAW / WAR / WAW cuando se accede a shared o global memory
- Esta permitido en condicionales sólo si el if es consistente en todo el bloque