UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Tarjetas Gráficas y Aceleradores

CUDA – Sesion06 - Streams

Agustín Fernández

Departament d'Arquitectura de Computadors

Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya

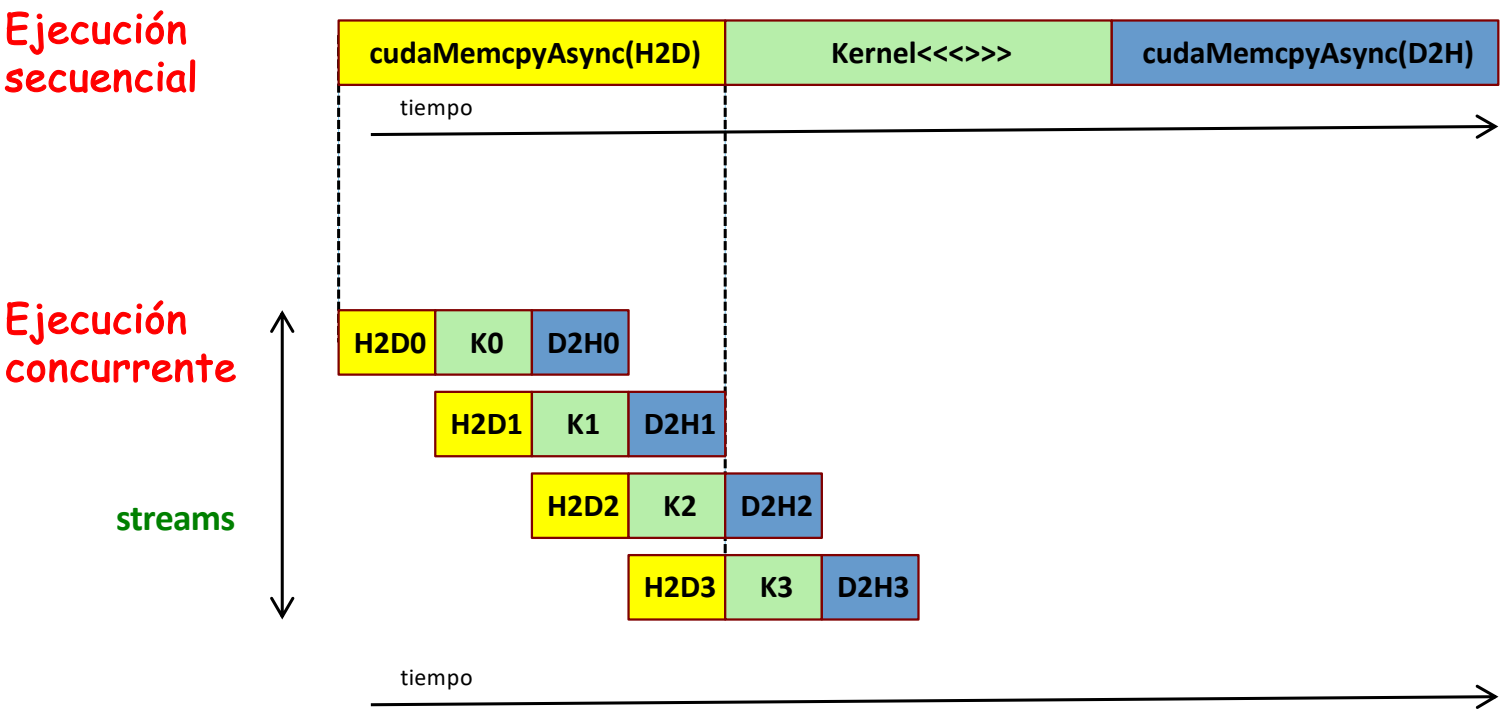
CUDA Streams

- **Objetivo: ejecutar varias operaciones CUDA simultáneamente:**
  - CUDA `kernel<<<>>>`
  - `cudaMemcpyAsync(HostToDevice)`
  - `cudaMemcpyAsync(DeviceToHost)`
  - Cálculos en la CPU

- **Stream**
  - Secuencia de operaciones que se ejecutan en la GPU en el orden en que se lanzan.

- **Las operaciones CUDA de diferentes streams pueden ejecutarse concurrentemente.**
- **Las operaciones CUDA de diferentes streams pueden entrelazarse.**

CUDA Streams



Punto de partida: stream00

```
__global__ void Kernel00(int N, float a, float b, float *A, *B, *C) {

  int i = blockIdx.x * blockDim.x + threadIdx.x;
  int j;
  if (i<N) {
      C[i] = a*A[i] + b*B[i];
      for (j=0; j<DUMMY; j++)
        C[i] += a*A[i] + b*B[i];
  }
}
```

No hacemos ningún cálculo útil.

DUMMY sirve para incrementar el peso del cálculo.

DUMMY == 30

Punto de partida: stream00

```
// Obtiene Memoria [pinned] en el host
// Inicializa las matrices
// Obtener Memoria en el device

// Copiar datos desde el host en el device
cudaMemcpy(d_A, h_A, numBytes, cudaMemcpyHostToDevice);
cudaMemcpy(d_B, h_B, numBytes, cudaMemcpyHostToDevice);

// Ejecutar el kernel
Kernel00<<<nBlocks, nThreads>>>(N, a, b, d_A, d_B, d_C);

// Obtener el resultado desde el host
cudaMemcpy(h_C, d_C, numBytes, cudaMemcpyDeviceToHost);

// Liberar Memoria del device
```

Código a Evaluar

Ejecución:

```
nvprof –unified-memory-profiling off ./kernel00.exe 10000 Y
```

Punto de partida: stream00

```
KERNEL 00: NO - Streams
Dimension Problema: 10240000
Invocacion Kernel <<<nBlocks, nKernels>>> (N): <<<10000, 1024>>> (10240000)
nKernels: 1
Usando Pinned Memory
Tiempo Global (00): 19.390848 milseg
Rendimiento Global (00): 64.95 GFLOPS
TEST PASS
```

```
==231238== NVPROF is profiling process 231238, command: ./kernel00.exe 50000 Y

Time(%)      Time     Calls       Avg       Min       Max  Name
 40.50%  7.7632ms         2  3.8816ms  3.8812ms  3.8820ms  [CUDA memcpy HtoD]
 39.25%  7.5239ms         1  7.5239ms  7.5239ms  7.5239ms  Kernel00(…)
 20.25%  3.8817ms         1  3.8817ms  3.8817ms  3.8817ms  [CUDA memcpy DtoH]
```

Tiempos de cálculo y comunicación "equivalentes".

1ª Versión: stream01

```
numB4 = numBytes/4; N4 = N/4; nB = (N4+nThr-1)/nThr;

cudaMemcpyAsync(&d_A[0], &h_A[0], numB4, cudaMemcpyHostToDevice, str1);
cudaMemcpyAsync(&d_B[0], &h_B[0], numB4, cudaMemcpyHostToDevice, str1);
Kernel00<<<nB,nThr,0,str1>>>(N4, a, b, &d_A[0], &d_B[0], &d_C[0]);
cudaMemcpyAsync(&h_C[0], &d_C[0], numB4, cudaMemcpyDeviceToHost, str1);

cudaMemcpyAsync(&d_A[N4], &h_A[N4], numB4, cudaMemcpyHostToDevice, str2);
cudaMemcpyAsync(&d_B[N4], &h_B[N4], numB4, cudaMemcpyHostToDevice, str2);
Kernel00<<<nB,nThr,0,str2>>>(N4, a, b, &d_A[N4], &d_B[N4], &d_C[N4]);
cudaMemcpyAsync(&h_C[N4], &d_C[N4], numB4, cudaMemcpyDeviceToHost, str2);

cudaMemcpyAsync(&d_A[2*N4], &h_A[2*N4], numB4, cudaMemcpyHostToDevice, str3);
cudaMemcpyAsync(&d_B[2*N4], &h_B[2*N4], numB4, cudaMemcpyHostToDevice, str3);
Kernel00<<<nB,nThr,0,str3>>>(N4, a, b, &d_A[2*N4], &d_B[2*N4], &d_C[2*N4]);
cudaMemcpyAsync(&h_C[2*N4], &d_C[2*N4], numB4, cudaMemcpyDeviceToHost, str3);

cudaMemcpyAsync(&d_A[3*N4], &h_A[3*N4], numB4, cudaMemcpyHostToDevice, str4);
cudaMemcpyAsync(&d_B[3*N4], &h_B[3*N4], numB4, cudaMemcpyHostToDevice, str4);
Kernel00<<<nB,nThr,0,str4>>>(N4, a, b, &d_A[3*N4], &d_B[3*N4], &d_C[3*N4]);
cudaMemcpyAsync(&h_C[3*N4], &d_C[3*N4], numB4, cudaMemcpyDeviceToHost, str4);
```

1ª Versión: stream01

```
KERNEL 01: 4 Streams
Dimension Problema: 10240000
Invocacion Kernel <<<nBlocks, nKernels>>> (N): <<<2500, 1024>>> (2560000)
nKernels: 4
Usando Pinned Memory
Tiempo Global (01): 11.884384 milseg
Rendimiento Global (01): 105.98 GFLOPS
TEST PASS
```

```
KERNEL 00: NO - Streams
. . .
Tiempo Global (00): 19.390848 milseg
Rendimiento Global (00): 64.95 GFLOPS
. . .
```

Speedup = 1,63

2ª Versión: stream01 Reordenado

```
numB4 = numBytes/4; N4 = N/4; nB = (N4+nThr-1)/nThr;

cudaMemcpyAsync(&d_A[0], &h_A[0], numB4, cudaMemcpyHostToDevice, str1);
cudaMemcpyAsync(&d_A[N4], &h_A[N4], numB4, cudaMemcpyHostToDevice, str2);
cudaMemcpyAsync(&d_A[2*N4], &h_A[2*N4], numB4, cudaMemcpyHostToDevice, st
cudaMemcpyAsync(&d_A[3*N4], &h_A[3*N4], numB4, cudaMemcpyHostToDevice

cudaMemcpyAsync(&d_B[0], &h_B[0], numB4, cudaMemcpyHostToDevice
cudaMemcpyAsync(&d_B[N4], &h_B[N4], numB4, cudaMemcpyHostTo          2);
cudaMemcpyAsync(&d_B[2*N4], &h_B[2*N4], numB4, cudaMemcp          ice, str3);
cudaMemcpyAsync(&d_B[3*N4], &h_B[3*N4], numB4, cudaMe          Device, str4);

Kernel00<<<nB,nThr,0,str1>>>(N4, a, b, &d_A[0]          , &d_C[0]);
Kernel00<<<nB,nThr,0,str2>>>(N4, a, b, &d          B[N4], &d_C[N4]);
Kernel00<<<nB,nThr,0,str3>>>(N4, a, b,          , &d_B[2*N4], &d_C[2*N4]);
Kernel00<<<nB,nThr,0,str4>>>(N4, a, b, &          N4], &d_B[3*N4], &d_C[3*N4]);

cudaMemcpyAsync(&h_C[0], &d_C[0], numB4, cudaMemcpyDeviceToHost, str1);
cudaMemcpyAsync(&h_C[N4], &d_C[N4], numB4, cudaMemcpyDeviceToHost, str2);
cudaMemcpyAsync(&h_C[2*N4], &d_C[2*N4], numB4, cudaMemcpyDeviceToHost, str3);
cudaMemcpyAsync(&h_C[3*N4], &d_C[3*N4], numB4, cudaMemcpyDeviceToHost, str4);
```

¡Rendimiento EQUIVALENTE!

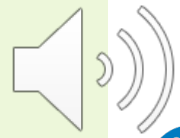
3ª Versión: stream02

```
nK = 2500*1024; nB = nK * sizeof(float);
nThreads = 1024;            // numero de Threads
nBlocks = nK / nThreads;    // numero de Blocks en cada dimension

for (s=0, p=0; p < N; s = (s+1)%nStreams, p += nK) {
  cudaMemcpyAsync(&dA[p], &hA[p], nB, cudaMemcpyHostToDevice, str[s]);
  cudaMemcpyAsync(&dB[p], &hB[p], nB, cudaMemcpyHostToDevice, str[s]);
  Kernel00<<<nBlocks, nThreads, 0, str[s]>>>(nK, a, b, &dA[p], &dB[p], &dC[p]);
  cudaMemcpyAsync(&hC[p], &dC[p], nB, cudaMemcpyDeviceToHost, str[s]);
}
```

```
KERNEL 02: Streams 2
Dimension Problema: 10240000
Invocacion Kernel <<<nBlocks, nKernels>>> (N): <<<2500, 1024>>> (2560000)
nKernels: 4
Usando Pinned Memory
Tiempo Global (02): 11.528608 milseg
Rendimiento Global (02): 109.25 GFLOPS
TEST PASS
```

Speedup = 1,68



Comparativa para N = 300000

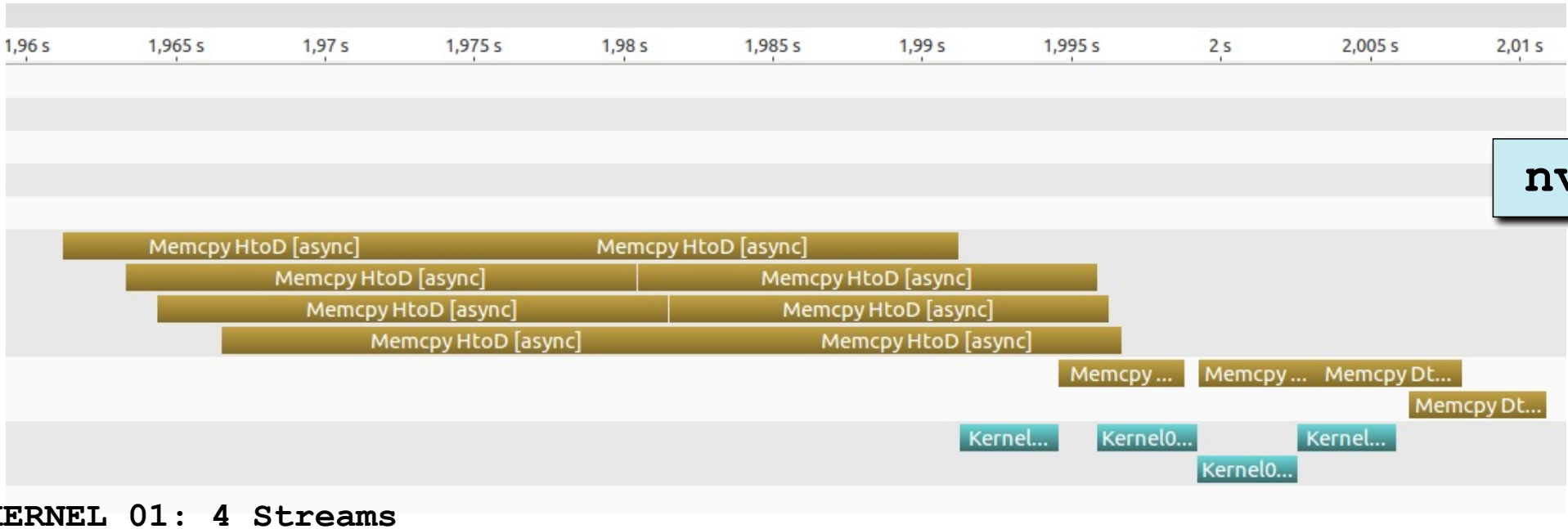
	STREAMS00	STREAMS01	STREAMS02
N	300000	300000	300000
#kernels ejecutados	1	4	120
nBlocks por kernel	300000	75000	2500
nThreads por Block	1024	1024	1024
Rendimiento	65,94 GFLOPS	106,58 GFLOPS	**139,66 GFLOPS**
Speedup	-	x1,61	**x2,12**

Sesion 06 en una GTX1080ti


```
KERNEL 00: NO - Streams
Dimension Problema: 51200000
Invocacion Kernel <<<nBlocks, nKernels>>> (N): <<<50000, 1024>>> (51200000)
nKernels: 1
Usando Pinned Memory
Tiempo Global (00): 62.918270 milseg
Rendimiento Global (00): 100.09 GFLOPS
TEST PASS
```

Rendimientos CONSISTENTES

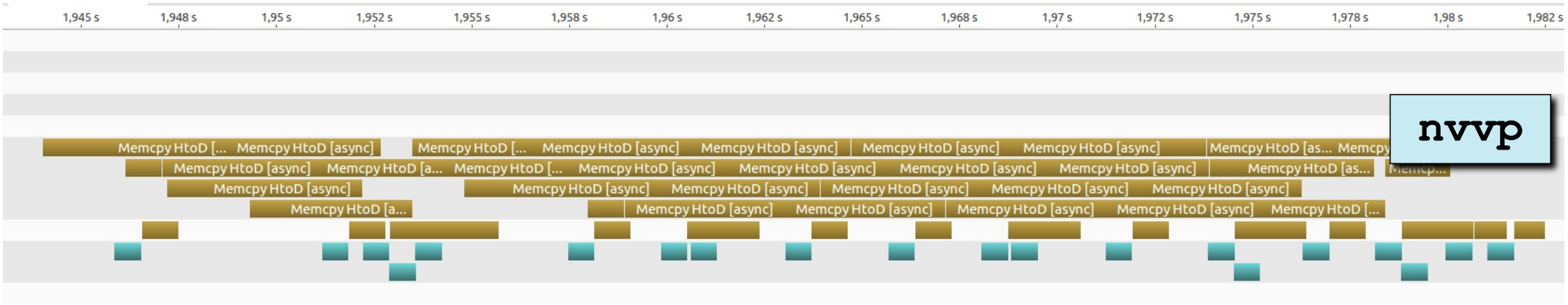
Sesion 06 en una GTX1080ti




```
KERNEL 01: 4 Streams
Dimension Problema: 51200000
Invocacion Kernel <<<nBlocks, nKernels>>> (N): <<<12500, 1024>>> (12800000)
nKernels: 4
Usando Pinned Memory
Tiempo Global (01): 52.053600 milseg
Rendimiento Global (01): 120.98 GFLOPS
TEST PASS
```

Rendimientos CONSISTENTES

Sesion 06 en una GTX1080ti




```
KERNEL 02: Streams 2
Dimension Problema: 51200000
Invocacion Kernel <<<nBlocks, nKernels>>> (N): <<<2500, 1024>>> (2560000)
nKernels: 20
Usando Pinned Memory
Tiempo Global (02): 38.490849 milseg
Rendimiento Global (02): 163.61 GFLOPS
TEST PASS
```

Rendimientos CONSISTENTES

Proyecto TGA / Resto del curso

- Para realizar el proyecto tenemos 4 sesiones de Laboratorio.
- Para acabar el curso nos quedan 4 sesiones de teoría
- Distribución del examen: 20 de mayo.
- Fecha límite para entregar examen y proyecto: lunes 18 de junio a las 12:00.

Proyecto TGA / Resto del curso

- El proyecto os ha de costar unas 30 horas de trabajo (incluyendo las horas de laboratorio).
- La entrega la haremos a través del racó . Además del informe, en la entrega del proyecto hay que incluir las versiones implementadas, los makefiles y los job.sh. **[TODO EN 1 ZIP.]**
- El informe [en general] debe contener los siguientes apartados:

1. Portada (1 pág.). Nombre y apellidos de los integrantes del grupo y título del proyecto.
2. Definición del problema a resolver (1-4 pág.). Documentación (referencias bibliográficas) utilizada para entender el problema y realizar el código secuencial. Si el código secuencial no lo habéis implementado vosotros indicad la fuente.
3. Implementaciones CUDA realizadas (versiones del código) y descripción de los aspectos más relevantes de cada una (principales diferencias respecto al resto de versiones implementadas). Indicad para cada versión el nombre del fichero que tiene el fichero fuente (1-4 págs.)
4. Analizad el rendimiento obtenido para cada implementación CUDA realizada (1-2 págs.). Incluid alguna medida de rendimiento para evaluar las mejoras introducidas (GFLOPS, ancho de banda, tiempo, …). Si se observa alguna inconsistencia en los resultados indicad posible causas. Se recomienda realizar varias ejecuciones por cada prueba y obtener una media. Recordad que el código secuencial debe ejecutarse en el mismo servidor que el cód CUDA.

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Tarjetas Gráficas y Aceleradores

CUDA – Sesion06 - Streams

Agustín Fernández

Departament d'Arquitectura de Computadors

Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya