P3. Histograma en CUDA

UNIVERSIDAD DE GRANADA

Nicolás Calvo Cruz
Dpto de Arquitectura y Tecnología de los Computadores
@ncalvocruz
ncalvocruz@ugr.es

Índice

Bases

- Las prácticas se realizan en Linux nativo
- Se usarán al menos dos máquinas diferentes, la personal o del aula y genmagic.ugr.es
- Las prácticas se entregan en swad.ugr.es en la fecha indicada.
- Entregar tarde penaliza, pero es mejor que no hacerlo

Objetivo

Analizar y ejecutar nuestro primer programa en CUDA

Comenzamos

- Desarrollaremos un **histograma** que nos dará información de una imagen.
- El histograma de una imagen representa la frecuencia relativa de sus niveles de gris.
- Las técnicas de modificación del histograma sirven para aumentar el contraste de imágenes con histogramas muy concentrados. Más información: https://es.wikipedia.org/wiki/Histograma .

```
for(int i = 0; i < BIN_COUNT; i++)
    result[i] = 0;

for(int i = 0; i < dataN; i++)
    result[data[i]]++;
```

Histograma versión 1

Enfoque versión 1

1. Definir datos (buffer) y un solo histograma (histo) para dispositivo y otro para el host y acumular todos los hilos en la misma estructura con atomicAdd.

2. Usar un solo bloque de hilos que accedan a ese vector (1024 hilos máximo):

```
HistoKernel_v1 <<<1 , 1024>>> (buffer ,  histo);
```

- Los datos no podrían ser más de 1024 :-(

Enfoque versión 1

```
__global__ void
HistoKernel_v1 (const char *const buffer , int
*histo)
{
      int i = threadIdx.x ;
      atomicAdd (&histo[buffer[i]] , 1 );

}
```

Histograma versión 2

Enfoque versión 2

1. Definir variables y un solo histograma para dispositivo y otro para el host y acumular todos los hilos en la misma estructura con atomicAdd.

2. Usar varios bloques de hilos mapeando el trabajo entre todos

```
HistKernel_v2<<<N/1024 , 1024>>>(buffer ,  histo);

HistKernel_v2b<<<(N+1023)/1024 , 1024>>>(buffer ,  histo);
```

   - ❏ Repartimos en bloques de 1024 hilos.
   - ❏ Problema? Si n/1024 es mayor que **65535** no llegamos… :-(

Enfoque versión 2

```
__global__ void
HistoKernel_v2(const char *const buffer , int *histo){
    int id = blockidx.x * blockDim.x + threadIdx.x ;
    atomicAdd (&histo[buffer[id]] , 1 );
}

__global__ void
HistoKernel_v2b(const char *const buffer , int *histo){
    int id = blockidx.x * blockDim.x + threadIdx.x ;
    if( id < lenBuffer)
        atomicAdd (&histo[buffer[id]] , 1 );
}
```

Histograma versión 3

Enfoque versión 3

1. Definir variables y un solo histograma para dispositivo y otro para el host y acumular todos los hilos en la misma estructura con atomicAdd

2. Usar bloques en dos dimensiones (asumimos que todas las divisiones son exactas):

```
int mitadN = N/2 ;
dim3 blocks (16, 16);
dim3 grid(mitadN / 16 , mitadN/16) ;
HistoKernel_v3 <<<grid , blocks>>> (buffer , histo, mitadN ) ;
```

Enfoque versión 3

```
__global__ void
HistoKernel_v3(const char *const buffer , int *histo, int mitadN)
{
        int id_x = blockidx.x * blockDim.x + threadIdx.x ;
        int id_y = blockidx.y * blockDim.y + threadIdx.y ;
        int absoulteId = id_y*mitadN + id_x ;

        atomicAdd (&histo[buffer[absoulteId]] , 1 );

}
```

Histograma general

Contexto

- Se proporciona el archivo “pr3.cu”, donde se implementa un cálculo de histograma para GPU.

- Este trata de tener en cuenta los problemas previos y que incluye una medición de rendimiento.

- Estudia el código dado con detenimiento y asimila su funcionamiento.

¿Qué hago?

Tareas

- El código dado arrastra algunos problemas: El número de hilos por bloque no se tiene en cuenta correctamente, ya que asume un número adecuado de tamaño de bloque. **Empieza por generalizarlo.**
- Experimento 1: Realiza un estudio sobre cómo afecta la variación del tamaño de grid (número de bloques) con al menos 4 valores distintos.
- Experimento 2: Realiza el mismo tipo de estudio sobre el impacto del tamaño de bloque (número de hilos por bloque) fijando el tamaño del grid al mejor valor encontrado anteriormente.
- Experimento 3: Sobre la versión general y mejor configuración, calcula la aceleración sobre el cálculo en CPU (cálculo frente a cálculo). Posteriormente, te en cuenta las transferencias a GPU.

Entregables

- **Código** del ejemplo proporcionado generalizado para cualquier tamaño de bloque.
- **Memoria** en la que:
  1. **Se explique el código original**, **cómo se aborda la paralelización en la GPU**, y qué se le ha modificado para admitir un tamaño de bloque distinto a IMDEP.
  2. Se incluyan los resultados de los estudios de los tres experimentos (tabla y gráfica para 1 y 2), junto con su análisis (**ten en cuenta las características de la GPU usada**).

¡Vamos a trabajar!