David Morillo Massagué

Niu: 1666540

# Exercicis opcionals OpenAcc VOPS

Responeu la següent pregunta fent servir el programa vops.c. Les vostres respostes han de figurar, respectant la numeració, en un document que lliurareu més tard sota el títol VOPS.

1. El programa vops.c inicialitza dos vectors i fa dues operacions matemàtiques amb ells. **Quines són les operacions matemàtiques?**
   En el primer bucle for, s'itera per cada valor de 0 a N i per a cada índex (i) dels vectors, se suma l'element vecA[i] amb el de vecB[i] i es guarda al vecC[i].

   Al segon bucle, s'itera un altre cop per a tots els valors del vector, però en aquest cas es multiplica el valor de l'element vecA[i] amb el de vecB[i] i se suma el producte a la variable "multp". S'obté el producte escalar dels dos vectors.

   **Quantes operacions punt flotant impliquen cada una d'elles?**
   Primer bucle:
   **N** operacions de punt flotant (una per iteració)

   Segon bucle:
   **2*N** operacions de punt flotant (una per la multiplicació i l'altra pel sumatori a la variable "multp", per cada iteració)

Responeu les següents preguntes fent servir el programa vops-acc-k.c. Aquest programa s'ha d'executar a la GPU, així que cal fer servir slurm i l'script job.sub on heu de descomentar la línia: nvc -acc=gpu -ta=tesla -Minfo=all -o executable $1 (deixant les altres dues comentades). A més, com a 2n paràmetre de l'script heu de passar -prof.

1. **Quants (i quins) kernels genera el compilador per aquest programa?**
   Comprovem a la taula de la sortida "CUDA Kernel Statistics":
   - Una instància de kernel amb en nom "main_27_gpu"
   - Una instància de kernel amb en nom "main_30_gpu"
   - Una instància de kernel amb en nom "main_30_gpu__red"

2. **Com és el grid (estructura de blocs i threads) que executa cadascun d'aquest kernels?**
   Comprovem a la taula de la sortida "CUDA Kernel & Memory Operations Trace":

| Nom de kernel | GrdX | GrdY | GrdZ | BlkX | BlkY | BlkZ |
|---|---|---|---|---|---|---|
| main_27_gpu | 8192 | 1 | 1 | 128 | 1 | 1 |
| main_30_gpu | 8192 | 1 | 1 | 128 | 1 | 1 |
| main_30_gpu__red | 1 | 1 | 1 | 256 | 1 | 1 |

3. **Quins moviments de dades s'han generat entre la CPU (Host) i la GPU (Device)?**
   Comprovem a la taula de la sortida "CUDA Memory Operation Statistics (by size in KiB)":

| Nom de la operació | Operacions | Total en KiB |
|---|---|---|
| [CUDA memcpy HtoD] (Host to Device) | 2 | 8192 |
| [CUDA memcpy DtoH] (Device to Host) | 2 | 4096 |

4. **Com es pot modificar aquest programa per fer que els blocs generats siguin de 256 threads?**
   Amb la clàusula **num_threads(256)** en la secció paral·lela

Responeu les següents preguntes fent servir el programa vops-acc-p.c. Aquest programa s'ha d'executar a la GPU, així que cal fer servir slurm i l'script job.sub on heu de descomentar la línia: nvc -acc=gpu -ta=tesla -Minfo=all -o executable $1 (deixant les altres dues descomentades). A més, com a 2n paràmetre de l'script heu de passar -prof.

1. **Quants (i quins) kernels genera el compilador per aquest programa?**
   Comprovem a la taula de la sortida "CUDA Kernel Statistics":
   - Una instància de kernel amb en nom "main_23_gpu"
   - Una instància de kernel amb en nom "main_27_gpu"
   - Una instància de kernel amb en nom "main_27_gpu__red"

2. **Com és el grid (estructura de blocs i threads) que executa cadascun d'aquest kernels?**
   Comprovem a la taula de la sortida "CUDA Kernel & Memory Operations Trace":

| Nom de kernel | GrdX | GrdY | GrdZ | BlkX | BlkY | BlkZ |
|---|---|---|---|---|---|---|
| main_23_gpu | 8192 | 1 | 1 | 128 | 1 | 1 |
| main_27_gpu | 8192 | 1 | 1 | 128 | 1 | 1 |
| main_27_gpu__red | 1 | 1 | 1 | 256 | 1 | 1 |

3. **Quins moviments de dades s'han generat entre la CPU (Host) i la GPU (Device)?**
   Comprovem a la taula de la sortida "CUDA Memory Operation Statistics (by size in KiB)":

| Nom de la operació | Operacions | Total en KiB |
|---|---|---|
| [CUDA memcpy DtoH] (Device to Host) | 2 | 4096 |
| [CUDA memcpy HtoD] (Host to Device) | 4 | 16384 |

4. **Feu servir la directiva data per optimitzar els moviments de dades**
   #pragma acc data copyin(vecA[0:N], vecB[0:N]) copyout(vecC[0:N])
   { bucles sense modificar }

5. **Com es pot modificar aquest programa per fer que els blocs generats siguin de 256 threads?**
   Amb la clàusula **num_threads(256)** en cadascuna de les seccions paral·leles.