COMPUTACIÓ D'ALTES PRESTACIONS - PAC 4

Semestre setembre 2018

- Presentació i objectius
- Enunciat de la pràctica
- Criteris d'avaluació
- Format de lliurament
- Data de lliurament

Presentació

Objectius

Aquesta pràctica té com objectiu introduir els conceptes bàsics del model de programació CUDA. Per a la seva realització es posaran en pràctica varis dels conceptes presentats en aquesta assignatura.

Presentació de la pràctica

Cal presentar un document amb les respostes a les preguntes formulades. Per a la PAC es faran servir extensions associades CUDA.

Restriccions de l'entorn

En aquesta PAC utilitzarem l'entorn d'execució Ocelot, un emulador de GPU, que fa possible provar codi sense disposar d'una GPU física. En qualsevol cas, si es disposa d'una GPU podeu validar el vostre codi per a aquesta i podeu realitzar execucions reals pel vostre compte (opcional - no s'espera proporcionar suport).

Material per a la realització de la pràctica

No cal material específic per aquesta PAC més enllà dels materials de l'assignatura. Si decidiu realitzar la implementació per a una GPU real s'espera que feu una cerca per vosaltres mateixos sobre la instal·lació i configuració dels SDK requerit. Podeu desenvolupar un debat al fòrum de l'assignatura per a compartir qualsevol experiència relacionada que desitgeu.

Enunciat de la pràctica

1. "Hello World" en CUDA

A continuació es troba un codi d'exemple per CUDA, en concret una versió del "Hello World" en CUDA:

Codi (fitxer `hello.cu`):

```
#include <stdio.h>

const int N = 16;
const int blocksize = 16;

__global__
void hello(char *a, int *b)
{
   a[threadIdx.x] += b[threadIdx.x];
}

int main()
{
   char a[N] = "Hello \0\0\0\0\0\0";
   int b[N] = {15, 10, 6, 0, -11, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

   char *ad;
   int *bd;
   const int csize = N*sizeof(char);
   const int isize = N*sizeof(int);

   printf("%s", a);

   cudaMalloc( (void**)&ad, csize );
   cudaMalloc( (void**)&bd, isize );
   cudaMemcpy( ad, a, csize, cudaMemcpyHostToDevice );
   cudaMemcpy( bd, b, isize, cudaMemcpyHostToDevice );

   dim3 dimBlock( blocksize, 1 );
   dim3 dimGrid( 1, 1 );
   hello<<<dimGrid, dimBlock>>>(ad, bd);
   cudaMemcpy( a, ad, csize, cudaMemcpyDeviceToHost );
   cudaFree( ad );

   printf("%s\n", a);
   return EXIT_SUCCESS;
}
```

Preguntes (Sobre el programa "Hello World" en CUDA):

1. Descriu el funcionament del programa, és a dir, como s'aconsegueix escriure "Hello World" a partir dels vectors d'entrada i el kernel proporcionat.

2. Quina transferència de dades es produeix entre el host i device (GPU)?

2. Execució mitjançant emulador Ocelot (instruccions)

Ocelot un entorn que permet executar programes CUDA en GPUs o en processadors convencionals. Podeu trobar-hi informació a la següent adreça:

`http://code.google.com/p/gpuocelot/`

A continuació es detallen les instruccions bàsiques que heu de seguir per utilitzar Ocelot; per a la versió del "Hello World" per GPU (des del node de login):

1. Codi (fitxer `hello.cu`):

2. Modificar el LD_LIBRARY_PATH adequadament

```
export
LD_LIBRARY_PATH=/export/apps/gcc/4.8.2/lib:/export/apps/gcc/4.8.2/lib64:/e
xport/apps/ocelot/lib/:/export/apps/boost/lib/:$LD_LIBRARY_PATH
```

3. Compilació del codi CUDA

```
/export/apps/cuda/5.5/bin/nvcc -cuda hello.cu -I
/export/apps/ocelot/include/ocelot/api/interface/ -arch=sm_20
```

4. El pas anterior generarà un fitxer anomenat `hello.cu.cpp.ii` que s'haurà de compilar amb g++ per a obtindré un binari executable.

```
/export/apps/gcc/4.8.2/bin/g++ -o hello hello.cu.cpp.ii -I
/export/apps/ocelot/include/ocelot/api/interface/ -L
/export/apps/ocelot/lib/ -locelot
```

5. Executar el programa

```
./hello
```

Podeu afegir la modificació del LD_LIBRARY_PATH i el PATH en el vostre .bashrc per a simplificar aquestes instruccions.

L'execució per defecte retorna warnings:

```
==Ocelot== WARNING: Failed to find 'configure.ocelot' in current directory,
loading defaults.

==Ocelot== INFO: You may consider adding one if you need to change the
Ocelot target, or runtime options.
```

Podeu ignorar aquests warnings o bé crear un fitxer en el mateix directori "configure.ocelot" amb, per exemple:

```
{
   "ReRoutes": [],
   "GlobalConfiguration": {}
}
```

Instruccions per a GPUs físiques (opcional, sense suport)

Podeu trobar informació de com instal·lar i utilitzar els SDK corresponents a CUDA o OpenCL (per GPUs NVIDIA o ATI, respectivament) podeu utilitzar els links següents com a referència: http://developer.nvidia.com/cuda-downloads i http://www.khronos.org/

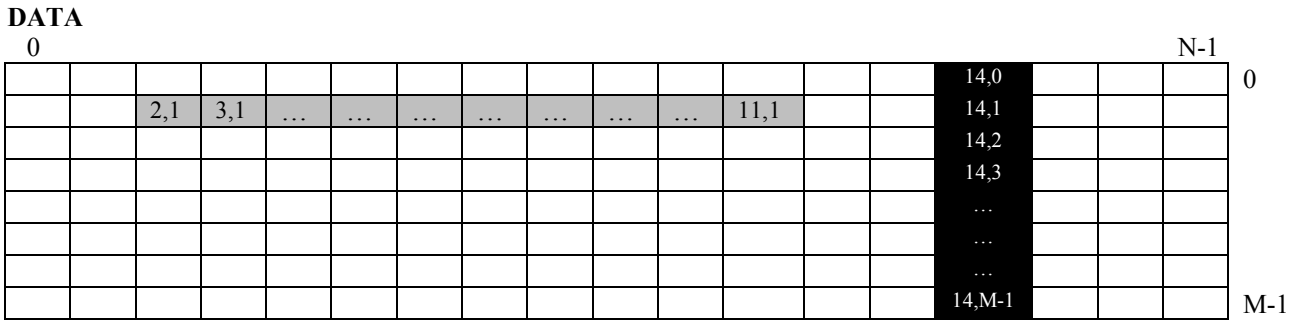
3. Exercici de programació

Es demana que programeu mitjançant CUDA el següent problema.

A partir d'una matriu **DATA**[N,M] de **N** columnes y **M** files, el programa ha de proporcionar:

1) Una matriu de sortida **MOV**, on cada punt MOV[i, j] es la mitjana de DATA[i-9, j], DATA[i-8, j], …, DATA[i, j] (representa una "moving average").
2) Un vector **AVG** amb la mitjana dels valores de totes les files per a cada columna, és a dir AVG[i] és la mitjana de DATA[i, 0], DATA[i, 1], …, DATA[i, M-1]

tal i com es mostra en la següent il·lustració:



Per exemple:

MOV[11,1] = mitjana dels valors marcats en gris
AVG[14] = mitjana dels valors marcats en negre

Preguntes:

3. Proporcioneu la teva implementació.

4. Proporcioneu un exemple d'ús del vostre codi. Per exemple podeu generar una matriu d'entrada d'una grandària petita (1000x10) amb valors aleatoris (o seguint una distribució determinada) de tal manera que pugueu generar un gràfic de les dades d'entrada (gràfic amb 10 sèries de 1000 punts), i la matriu i vector de sortida.

4. Preguntes addicionals (opcional)

En aquesta part de la PAC s'espera que programeu el problema de l'exercici de programació com un problema sintètic. En uns dies se us facilitarà informació per contextualitzar aquest problema aplicant-lo a un problema real relacionat amb cyber-infraestructura (part de la PAC1).

Es formularan unes preguntes addicionals (opcionals) que es tindran en compte de cara a l'avaluació.

Criteris d'avaluació

Es valorarà l'ús correcte de les interfícies/extensions de CUDA. També es tindrà en compte la correcta programació i estructura de l'aplicació i les preguntes addicionals.

Format de lliurament

Es crearà un fitxer `tar` amb tots els fitxers font de la pràctica.

Amb la comanda següent:

```
$ tar cvf tot.tar fitxer1 fitxer2 ...
```

es crearà el fitxer "tot.tar" on s'hauran emmagatzemat els fitxers "fitxer1", fitxer2" i . . .

Per llistar la informació d'un fitxer `tar` es pot utilitzar la comanda següent:

```
$ tar tvf tot.tar
```

Per extraure la informació d'un fitxer `tar` es pot utilitzar:

```
$ tar xvf tot.tar
```

El nom del fitxer tindrà el format següent: "Cognom1Cognom2PAC4.tar". Els cognoms s'escriuran sense accents. Per exemple, l'estudiant Marta Vallès i Marfany utilitzarà el nom de fitxer següent: `VallesMarfanyPAC4.tar`

Data de lliurament

Dijous 27 de Desembre de 2018.