Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Dév. GPGPU

TD n°2

Programmation GPGPU

Structures de données et GPGPU

1 – Soit le programme suivant :

```
 1 #include <stdio.h>
 2 #include <cuda.h>
 3
 4 #define TAILLE 32
 5
 6 float tree[2*TAILLE];
 7
 8 __global__ void mon_noyau(float *t, int r)
 9 {
10   int position = threadIdx.x + r;
11
12   if ((position%2) == 0)
13   {
14     t[position/2] = t[position] * t[position+1];
15   }
16 }
17
18 int main(void)
19 {
20   float *gpu_tree;
21   int t = TAILLE;
22
23   cudaMalloc((void **)&gpu_tree,TAILLE*2*sizeof(float));
24   for(int i=TAILLE;i<TAILLE*2;i++)
25     tree[i] = float(2);
26
27   for(int i=0;i<TAILLE*2;i++)
28     printf("%4.0f",tree[i]);
29
30   printf("\n\n");
31   cudaMemcpy(gpu_tree,tree,2*TAILLE*sizeof(float),cudaMemcpyHostToDevice);
32
33   while(t>1)
34   {
35     mon_noyau<<<1,TAILLE>>>(gpu_tree,t);
36     t=t/2;
37   }
38   cudaMemcpy(tree,gpu_tree,2*TAILLE*sizeof(float),cudaMemcpyDeviceToHost);
39
40   for(int i=0;i<TAILLE*2;i++)
41     printf("[%4.0f]",tree[i]);
42   printf("\n");
43 }
```

1. **Analysez** et **décrivez** son fonctionnement.
2. Soit la trace d'exécution :

```
xterm
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   2   2   2   2   2   2   2   2
   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
   2   2   2   2

[   0][4294967296][65536][65536][ 256][ 256][ 256][ 256][  16][  16][  16][
16][  16][  16][  16][  16][   4][   4][   4][   4][   4][   4][   4][   4][
 4][   4][   4][   4][   4][   4][   4][   4][   2][   2][   2][   2][   2][
 2][   2][   2][   2][   2][   2][   2][   2][   2][   2][   2][   2][   2][
 2][   2][   2][   2][   2][   2][   2][   2][   2][   2][   2][   2][   2]
```

   Vérifiez qu'elle est **correcte**.

3. Donnez une **version améliorée**.

Utilisation de `printf`

```
#include <stdio.h>
#include <cuda.h>
#include <cuda_runtime.h>

#if defined(__CUDA_ARCH__) && (__CUDA_ARCH__ < 200)
#define printf(f, ...) ((void)(f, __VA_ARGS__),0)
#endif

__global__ void helloCUDA(float f)
{
        printf("Hello thread %d, f=%f\n", threadIdx.x, f);
}

int main()
{
        helloCUDA<<<1, 5>>>(1.2345f);
        cudaDeviceReset();
}
```

Ce qui donne :

```
xterm
Hello thread 0, f=1.234500
Hello thread 1, f=1.234500
Hello thread 2, f=1.234500
Hello thread 3, f=1.234500
Hello thread 4, f=1.234500
```