CINECA

Introduzione alle GPGPU e alla programmazione CUDA

www.cineca.it

User guides

http://www.hpc.cineca.it/content/ibm-plx-gpu-user-guide-0

Collegarsi alla macchina

host: login.plx.cineca.it

Creare una propria directory su SCRATCH in cui lavorare

cd $CINECA_SCRATCH
mkdir <nome directory>
cd <nome directory>

Caricare il modulo del compilatore

module avail fornisce la lista di moduli disponibili
(compilatori, tools, programmi, librerie)
module load cuda/4.0

Per compilare:

nvcc file.cu –o <nome eseguibile>

PLX job script - example

```
#!/bin/bash
#PBS -j oe -o exercise.out
#PBS -l select=1:ncpus=12:ngpus=1
#PBS -q debug

module load cuda/4.0
cd <work directory>

./executable.exe
```

Per sottomettere il job:
qsub <script>

Exercise 1
cuda memory model

Create an array of N integer elements on the device and inizialize it with 0.

Then copy the array back to the memory host and check the host elements.

NB.Check for errors.

Exercise 2
vector addition

Create 3 arrays a,b,c of N elements on the host and set a[i] = i and b[i]=-i.

Write a kernel function which compute the sum c=a+b.

Verify that all the elements of c are equal to 0.

Test the following cases:

- N=1024 and threads per block are 128. (add_simple.cu)
- N=1024*1024+3 and threads per block are 512. (add_simple2.cu)

Esercize 3
set matrix

Create a square matrix M of NxN elements.

Write a kernel that set the elements of M equal to the linear index of the current element: M[i,j] = i*N+j

Copy back the matrix to the host and print it.

Hint: submit the kernel with a grid of 2-dim blocks. Try with N=16 and blocks 4x4.

Exercize 4
sum reduction

Write a code which implements the sum reduction of an array of N elements:

$$S = a[0]+a[1]+\ldots+a[n]$$

Compare the result with the one computed by the host.

Try with N=1024 and N=1<<18.

Hint: compute it in two steps, using a kernel implementing a block-wide reduction.

- Step 1. compute partial sums of block-sized portion of the input
- Step 2. reduce the partial sums

Exercise 5
inclusive scan

Write a kernel that computes, per-block, a block-sized scan of the input array:

result[i] = input[0]+input[1]+…+input[i]
i=0,…,N

Assume that the block size evenly divides the input size

See the main file: block_scan.c