27 Lezione

martedì 7 maggio 2019 14:14

CUDA

M40 → nella macchina

NVIDIA → CUDA

AMD → openCL → g++ → -lopenCL

kernel code passed by CPU has to be provided as a string

different from openCL because openCL is a library and inside the program you can make calls and the library does all the things.
Cuda is a language (estension of C++) because introduces some features.

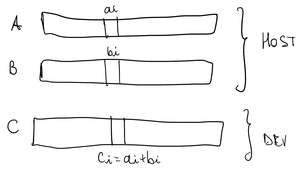
nvcc → compiler

-- global --

defined used the kernel keyword in front

<<< .... >>> → name of function( ) is used to tell the compiler to execute the kernel code on gpu.

Example of cuda code:



la funzione che somma i due vettori viene definita anteponendo davanti alla definizione la keyword __global__

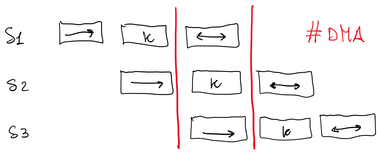
→ per allocare la memoria nella GPU Cuda Malloc

→ You need to manage events that happens in the GPU.
Example: we can create 2 events, start and stop, before starting the thread and after the end we register the time on this events.
these two events measures the execution time of the kernel.
the events are recorded on the GPU and sometimes we need to synchronize with CPU. [14:47]

CudaDeviceSynchronize() → wait for GPU to finish before accessing on host

MACCHINA per CUDA

131.114.136.61



Soluzione assignment 3

You need to consider the neighbours of the node.

Consider two matrix (we create) one matrice and we read from the other one.

ASSignment 4

Implement a mapReduce (tipo google) object:

std::function < std::pair<tout, tkey>, (Tin) >

f(x) → Map function

↓

<y, k>

std::function < tout (tout, tout) >

function g given <y',k'> @ <y'',k'> Compute <y''',k'>

reduce function

std::vector <tin>

↳ return std::vector <std::pair<tout, tkey>>

Implement this in parallel. You need to take into account that you are in shared memory.

V1: C++

V2: FastFlow