ANNOUNCEMENT

Paper number [5] and [7]
will be presented by one person rather than a group of two

Send e-mail to jsmin@dcslab.snu.ac.kr
about your preference to the papers
IF you have sent email about your preference and group, just ignore this announcement.

16 15 8 2 ~ all 16 of them ~ 1.

CUDA Programming Assignment

Sunggon Kim
<skim@dcslab.snu.ac.kr>

Table Of Contents

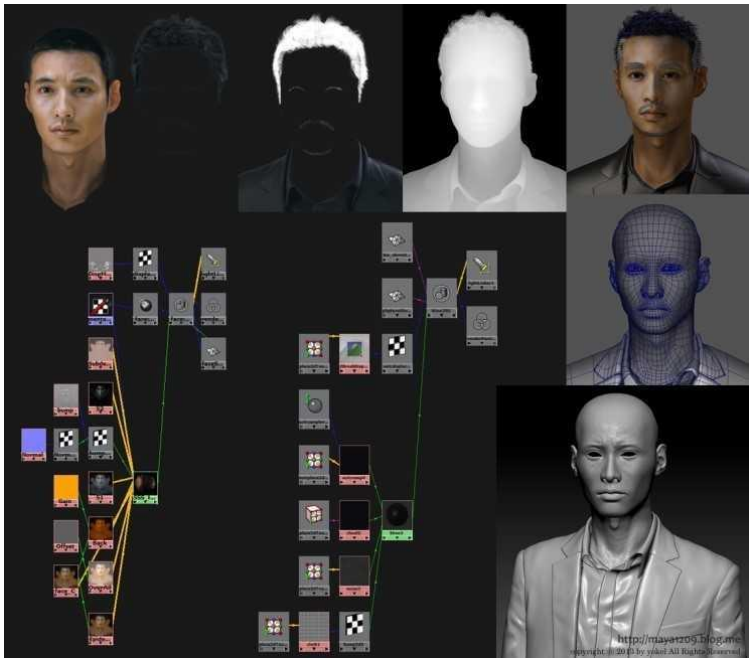
Multi-Core Rules

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National University of Defense Technology China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462,462	5,168.1	8,520.1	4,510

Graphic Processing

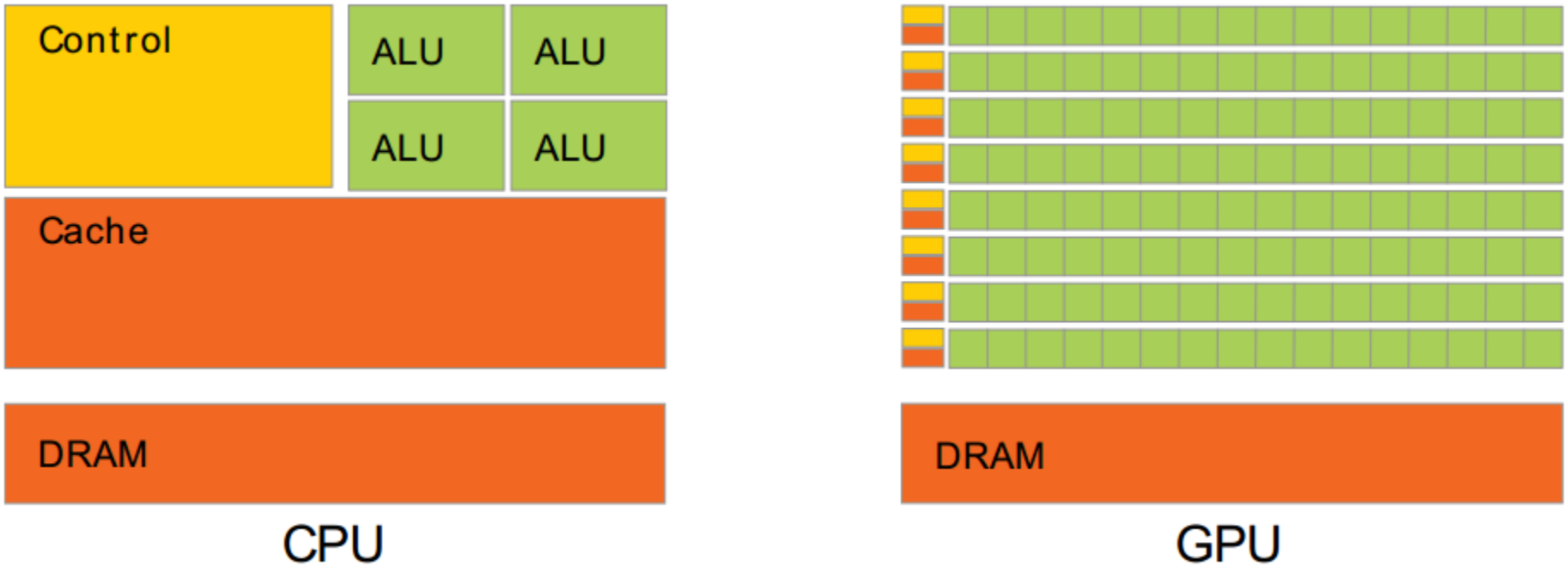
Extremely Computing Intensive Job

Also, Parallel Friendly



GPU: Graphic Processing Unit

- Specialized for
  - Compute Intensive, Highly Parallel Computation



GPU: Graphic Processing Unit

What If We Can Use GPU For General Purpos e?

CUDA™

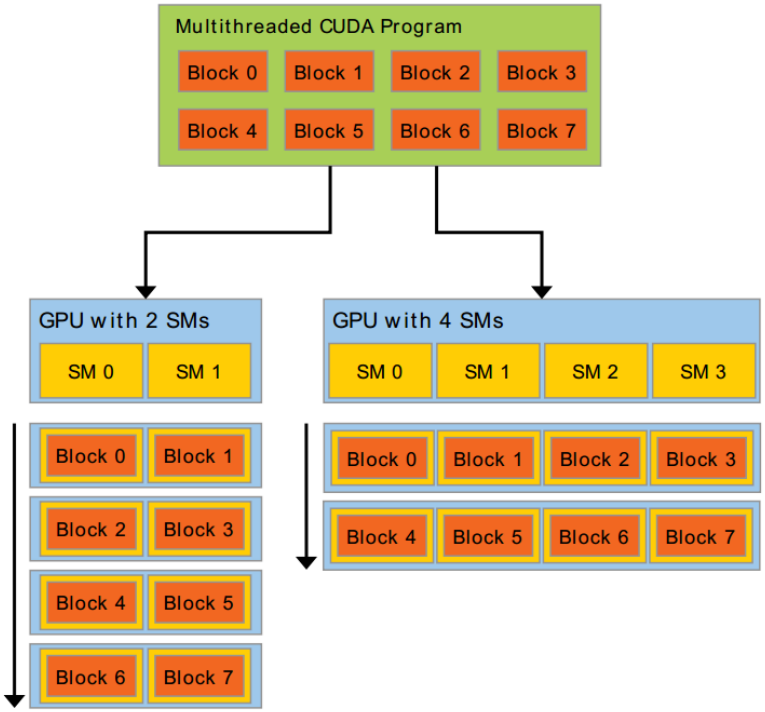
A General-Purpose
Parallel Computing Platform and
Programming Model

Introduced By NVIDIA, Nov 2006

Can Program GPU with standard programming languages such as C

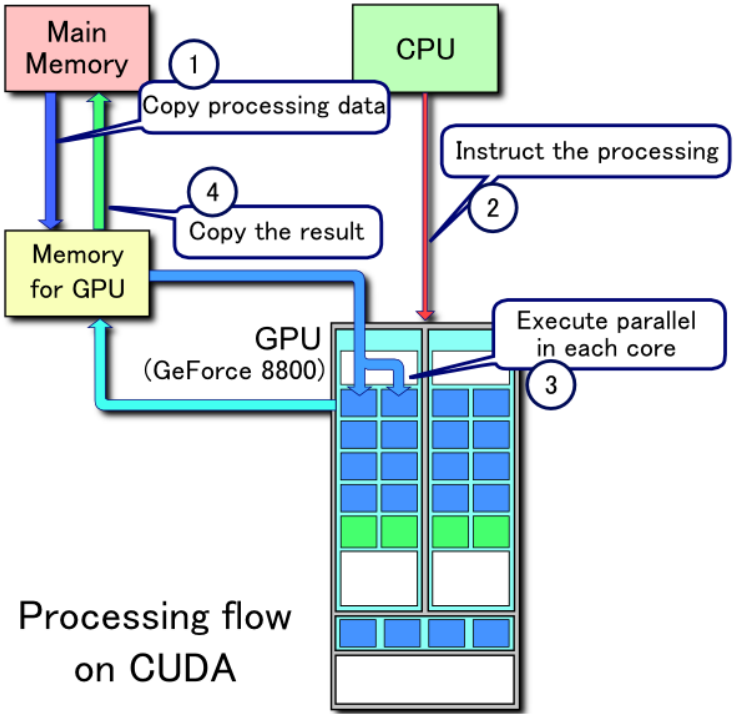
Scalable Programming Model

- GPU Is Built Around An Array Of SMs
  - Streaming multiprocessors
- Program Is Partitioned Into Blocks Of Threads



CUDA: The basic

- Terminology
  - Host – The CPU and its memory
  - Device – The GPU and its memory

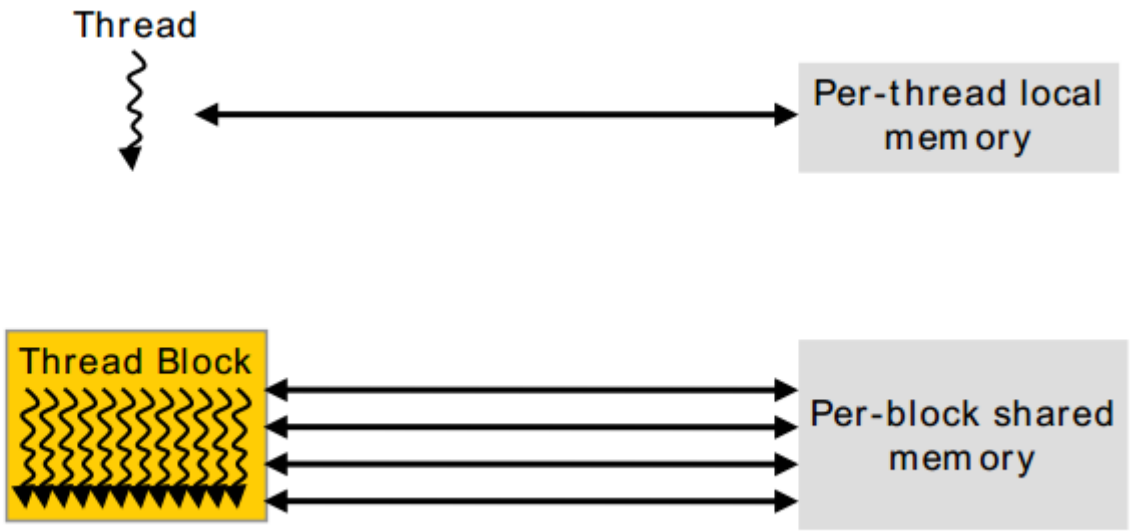


Hello, World

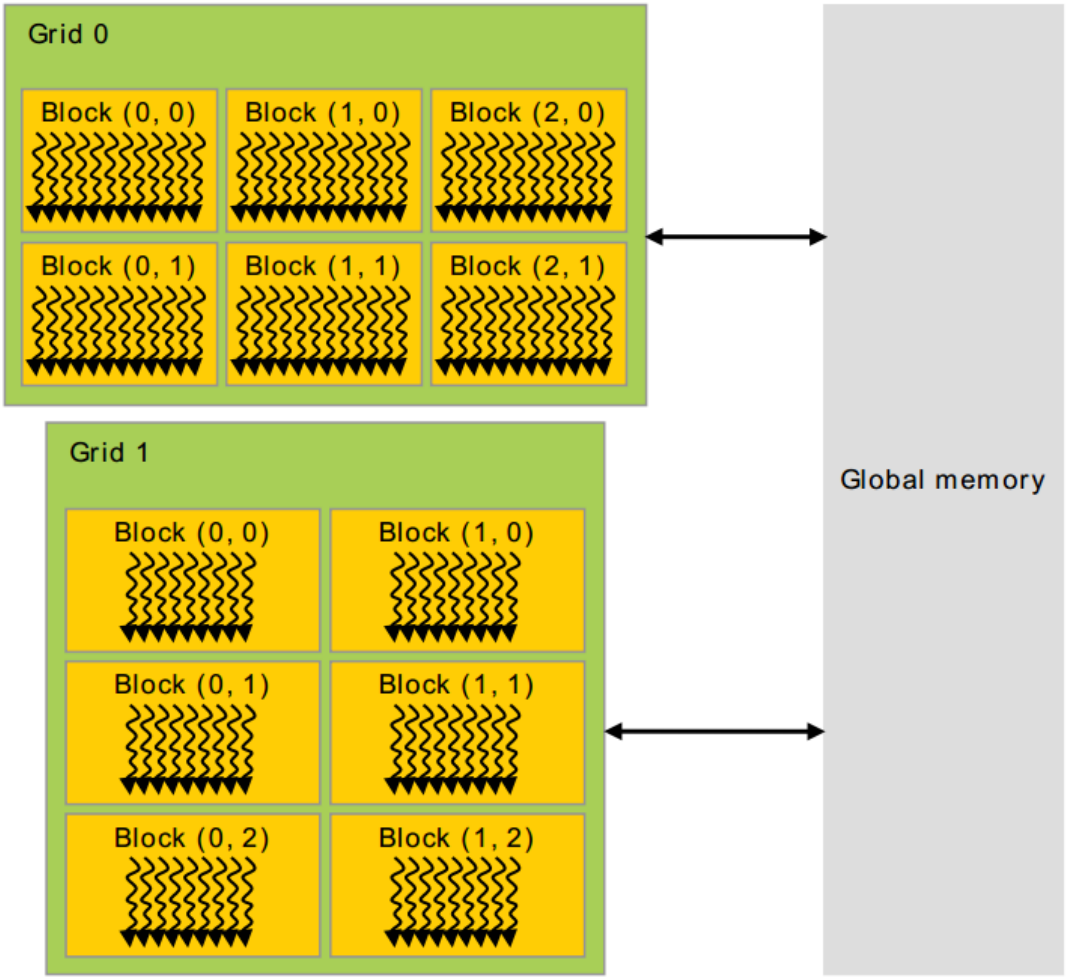
- __global__ void kernel(void) {
- } (__global__ = runs on the device)

- int main(void) {
  - kernel<<<1,1>>>(); ← go run kernel function up there
  - printf("Hello, World!\n");
  - return 0;
- }

CUDA Program: Memory Hierarchy



CUDA Program: Memory Hierarchy



CUDA Program: Memory Hierarchy

- cudaMalloc
  - Allocate Device Memory
- cudaMemcpy
  - Transfer Data Between Device And Host
  - cudaMemcpyDeviceToHost
  - cudaMemcpyHostToDevice
  - …and MORE
    - AsyncMemcpy, using streams
- cudaFree
  - Fairly simple. Just frees the memory

CUDA Program: Kernel Function

```
// Kernel definition
__global__ void add(int* A, int* B, int* C)
{
        *c = *a + *b;
}
```

REMEMBER to use pointer as the arguments.

A, B, C must point to device memory

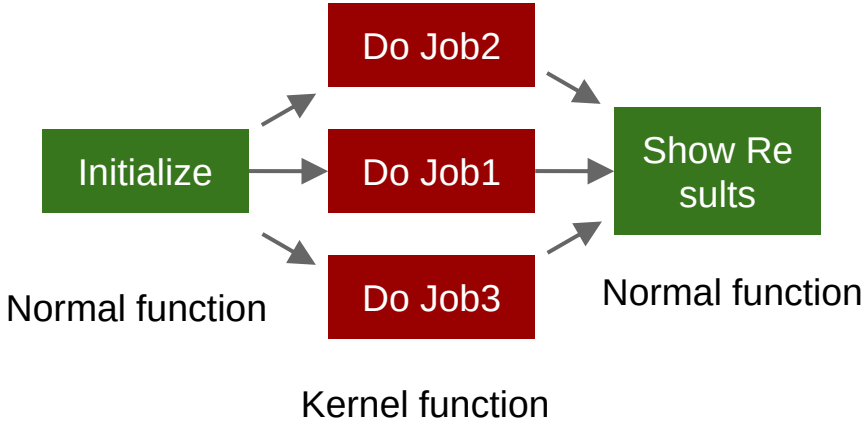
```
int main()
{    ...
    int hostA, hostB, hostC;
    int *a,*b,*c;
    cudaMalloc((void**)&a, sizeof(int); //allocate device mem
    cudaMemcpy(a, &hostA, sizeof(int), cudaMemcpyHostToDevice);
   // Kernel invocation with N threads
    VecAdd<<<1, 1>>>(a, b, c);
    cudaMemcpy(&hostC, c, sizeof(int), cudaMemcpyDeviceToHost);
    cudaFree(a,b,c);
 }
```

Cuda Program: Kernel Function

- Normal Functions
  - Runs on CPU
  - Same With Standard Program
- Kernel Functions
  - Runs on GPU
  - Similar With Standard Program

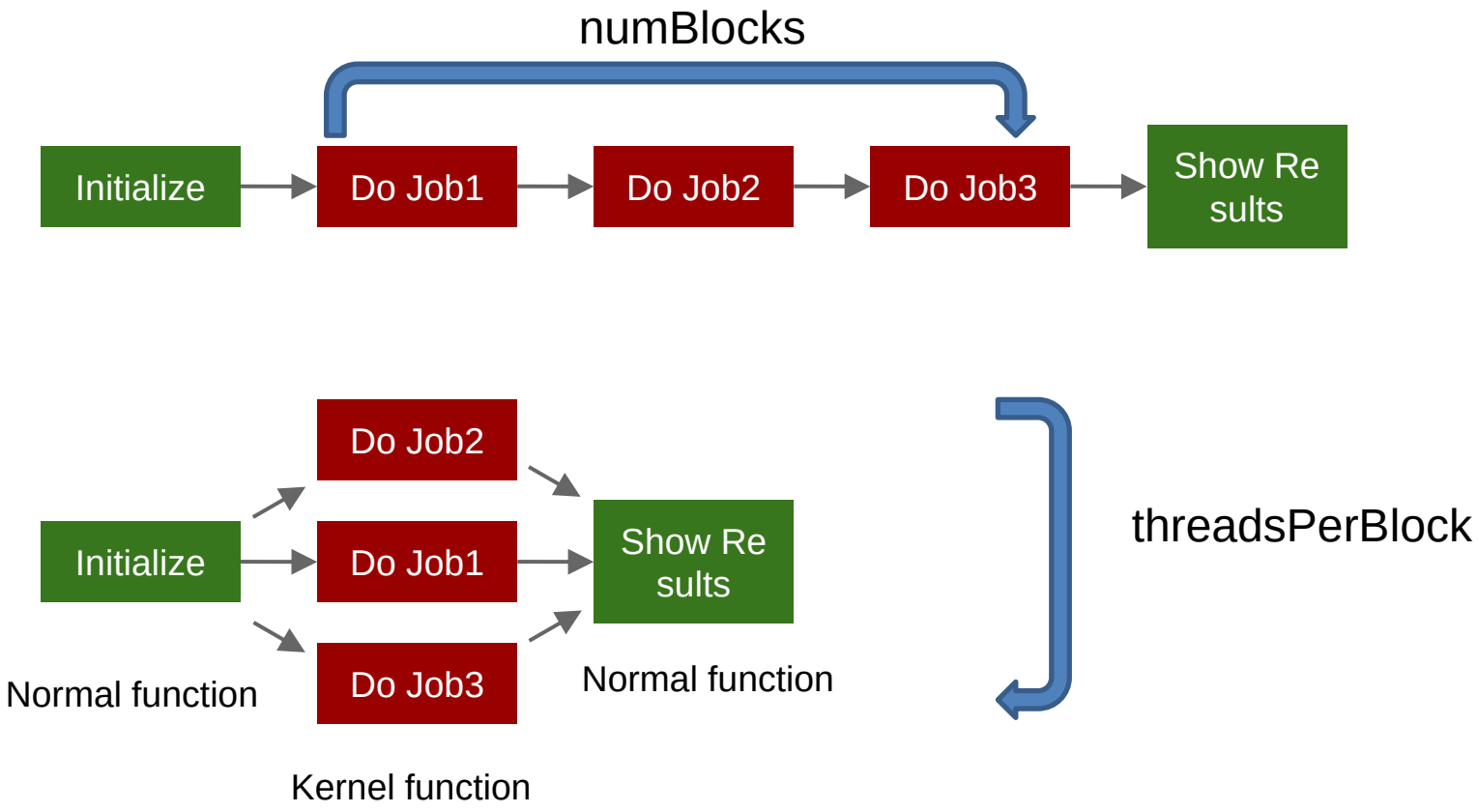
CUDA Program: Kernel Function

CUDA Program: Kernel Function



Parallel Programming

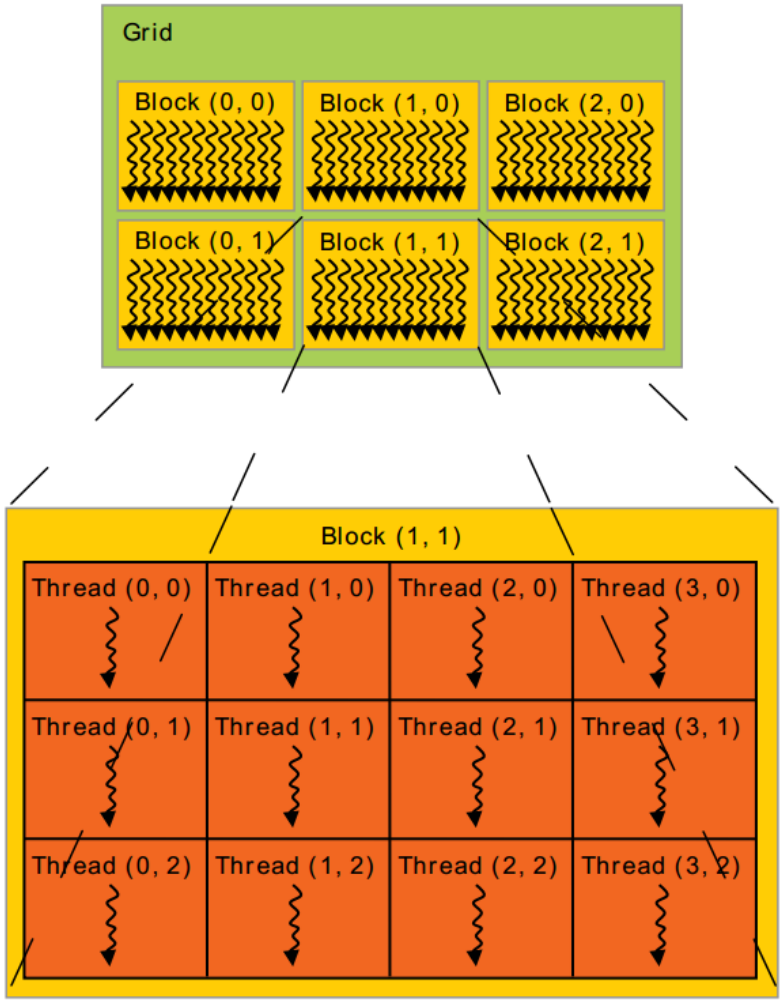
- Add<<numBlocks, threadsPerBlock



CUDA Program: Thread Hierarchy

All the threads of a block are expected to reside o n the same processor co re

# of threads per block:
~ 2048



Block hierarchy

- __global__ void add(int *a, int *b, int *c) {
- c[blockIdx.x] = a[blockIdx.x] + b[blockIdx.x];
- }

CUDA Program: Thread Hierarchy

```
__global__ void add( int *a, int *b, int *c ) {

 c[threadIdx.x] = a[threadIdx.x] + b[threadIdx.x];

}
```

CUDA Program: Build And Run

```
$ nvcc --output-file output sourcecode.cu
$ ./output

$ nvcc --help
```

Reference

http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf

http://www.nvidia.com/content/gtc-2010/pdfs/2131_gtc2010.pdf

Machine For Assignment

Server Address: 147.46.242.21
ID: cudateam<1-10>
PW: cuda

Email-me for your team name

Time Sharing

- Do Aggressive, Dynamic Time Sharing
  - Use Task Queueing Program(tasq)
  - Just Execute Program Using tasq
- Policy Can Be Changed Later

tasq: Task Queueing Program

Basic Usage:

$ tasq <enq | list> [command] [output]

Example:

$ tasq enq nvcc --output-file matmul matmul.cu output
$ tasq enq ./matmul output2
$ tasq list

Assignment

Implement DES Algorithm Using C and CUDA

Due Date: ??

Submit: skim@dcslab.snu.ac.kr

Mail Subject Should Contain:
[DIP2016_CUDA_TeamNo_Submit]

Question?