CUDA Programming Assignment

Dongwhan Kim
<dw83kim@dcslab.snu.ac.kr>
Table Of Contents

CUDA Programming Introduction

Assignment

Machine For Assignment
## Multi-Core Rules

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

- Specialized for
  - Compute Intensive, Highly Parallel Computation
GPU: Graphic Processing Unit

- By Lots Of Cores And Few Cache
  - Parallel Computations Does Not Share Data
GPU: Graphic Processing Unit

What If We Can Use GPU For General Purpose?
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
- Program Is Partitioned Into Blocks Of Threads

![Diagram showing the scalable programming model](image)
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

Initialize → Do Job1 → Do Job2 → Do Job3 → Show Results
CUDA Program: Kernel Function

Initialize → Do Job1 → Do Job2 → Do Job3 → Show Results

Kernel function

Normal function

Normal function

Normal function

Initialize → Do Job1 → Show Results → Do Job3
CUDA Program: Kernel Function

// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    ...
    // Kernel invocation with N threads
    VecAdd<<<1, N>>>(A, B, C);
    ...

CUDA Program: Thread Hierarchy

All the threads of a block are expected to reside on the same processor core.

# of threads per block: ~ 1024
CUDA Program: Thread Hierarchy

```c
int main()
{
    ...
    // Kernel invocation
    dim3 threadsPerBlock(16, 16);
    dim3 numBlocks(N / threadsPerBlock.x, N /
                    threadsPerBlock.y);
    MatAdd<<numBlocks, threadsPerBlock>>>(A, B, C);
    ...
}
```
CUDA Program: Thread Hierarchy

// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
float C[N][N])
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if (i < N && j < N)
        C[i][j] = A[i][j] + B[i][j];
}
CUDA Program: Memory Hierarchy
CUDA Program: Memory Hierarchy
CUDA Program: Memory Hierarchy

- cudaMalloc
  - Allocate Device Memory

- cudaMemcpy
  - Transfer Data Between Device And Host
CUDA Program: Build And Run

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

$ nvcc --help
Example Code

Matrix Multiplication Using CUDA
Reference

Machine For Assignment

Server Address: 147.46.242.21
ID: cudateam<1-8>
PW: welcomeCUDA<1-8>!

<1-8> is Your Team #
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: 10\textsuperscript{th} DEC.

Submit: dw83kim@dcslab.snu.ac.kr

Mail Subject Should Contain: [DIP2015_CUDA.TeamNo_Submit]
Question?