CUDA Programming Assignment

Hara Kang <hara0115@gmail.com>
Table Of Contents

GPU Introduction

CUDA Programming Introduction

Assignment

Machine For Assignment
Graphic Processing

Extremely Computing Intensive Job

Also, Parallel Friendly
Why GPU?

- High resolution
- Fast graphic processing
- Movies, animations, games
- GPU and CPU specialization
GPU: Graphic Processing Unit

- Specialized for
  - Compute Intensive, Highly Parallel Computation
  - Vector coordinate calculation and pixel processing
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?
GPGPU (General Purpose)

- Uses GPU for normal signal processing, physics simulation, financial forecasts, biological calculations, and so on.
- Possible with larger DRAM bandwidth and generic API for normal developers.
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
Cuda stands for

- Compute Unified Device Architecture
  - Programming model
  - Programming language
  - Compiler
  - Libraries
  - Debugger
  - Profiler
Advantages

- High operation processing capability
- Scalability
- Low cost
- Applicable in many fields
  - Entertainment, industry, design, architecture, medicine, education, and finance
Scalable Programming Model

- GPU Is Built Around An Array Of SMs
- Program Is Partitioned Into Blocks Of Threads
- Automatic scalability
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
```

- **Normal function**
  - Initialize
  - Do Job1
  - Do Job2
  - Do Job3
  - Show Results

- **Kernel function**
  - Initialize
  - Do Job1
  - Do Job2
  - 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
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: Memory Hierarchy

- Allocate memory space in graphic card with cudaMalloc
- Transfer host data to device global memory with cudaMemcpy (cudaMemcpyHostToDevice)
- Do parallel processing using GPU
- Transfer processed data from device memory back to host memory using cudaMemcpy (cudaMemcpyDeviceToHost)
CUDA Program: Build And Run

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

$ nvcc --help
Example Code

Matrix Multiplication Using CUDA
Reference


Cuda 병렬 프로그래밍 – 정영훈
Assignment

Implement DES Algorithm Using C and CUDA

Due Date: 12th, Dec, 2014

Submit: hara0115@gmail.com

Mail Subject Should Contain:
[DIP2014_CUDA_TeamNo_Submit]
Machine For Assignment

Server Address: 147.46.242.21
ID: cudateam<1-5>
PW: teamcuda<1-5>

<1-5> 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> [output] [command]

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