Lab 5

Introduction to Programming Laboratory

Goals

- Hades Cluster
- CUDA Introduction
- CUDA Techniques
- Task: Primes - Composites

Hades Cluster

Login Information

Address

hades.cs.nthu.edu.tw

Username

Same as apollo cluster

Password

Same as your changed password on apollo

Practice

Make sure you can log in to hades

CUDA Introduction

What is CUDA?

- **C**ompute **U**nified **D**evice **A**rchitecture
- A prognamming language / toolkit to write programs for **NVIDIA** GPUs. AMD GPUs are *not* supported.
- To write GPU programs that works in both NVIDIA and AMD GPUs, you may want to look into OpenCL (which is not covered by the course).

CUDA programming model

1. Allocate GPU memory
2. Copy data from CPU memory to GPU memory
3. Execute GPU kernel functions
4. Copy data back from GPU memory to CPU memory

CUDA functions for memory operations

- cudaMalloc
- cudaMemset
- cudaMemcpy
- cudaFree

Look them up at **https://docs.nvidia.com/cuda/cuda-runtime-api/**. **Don't** look at anywhere else!

CUDA kernel launch syntax

```
#include <stdio.h>
__global__ void kernelFunc() {
    printf("thread:%d/%d, block:%d/%d\n", threadIdx.x, blockDim.x, blockIdx.x, gridDim.x);
}
int main() {
    // ...
    kernelFunc<<<3, 4>>>();
    cudaDeviceSynchronize();
}
```

Run it!

Running CUDA programs

- Compile with `nvcc`, example:
  `nvcc -O3 -arch=sm_61 -std=c++11 code.cu`
- Run with
  `srun --gres=gpu:1 -pipl ./a.out argv1 argv2 ...`

In your own computer, you only need to use `./a.out argv1 argv2 ...` to run CUDA programs. `srun` is needed only because our GPU resources are managed by SLURM.

CUDA Techniques

nvprof

You can use `nvprof` to profile your program.

For example: `srun --gres=gpu:1 -pipl nvprof -o profile-output ./a.out argv1 argv2 ...`

Then, you can use `nvprof -i profile-output` to see the profile results.

Or you can omit `-o profile-output` to see the results immediately.

cuda-memcheck

Checks for memory errors in kernel functions. Think of AddressSanitizer, but for GPUs.

Run your program with `cuda-memcheck`.

For example `srun --gres=gpu:1 -pipl cuda-memcheck ./a.out argv1 argv2 ...`

printf

As mentioned in the example in the previous slides, you can actually use `printf` in kernel functions for debugging.

`std::cout` is not supported though.

Remember to remove `printf` after you finished debugging! It affects performance.

Task: Primes - Composites

Given a number N (`argv[1]`), find out the primes and composite numbers ≤ N

Calculate sum(primes) - sum(composites)

Requirements

- You can start from `/home/ipl19/y/lab5/lab5.cu`.
- Use CUDA to parallelize.
- Name your source code `lab5.cu`
- Demo to TA.