DLP Project

May 20, 2009

Introduction

- Use the NVIDIA Compute Unified Device Architecture (CUDA) Graphic Processor Unit (GPU) programming environment to explore
  - Data-parallel hardware
  - Programming environments
- Goals
  - Explore the space of parallel algorithms
  - Understand how the data-parallel hardware scales performance with more resources
  - Utilize the data-parallel programming model

ToolChain

- CUDA programming environment
  - Computer support
    - Snake machines in ece department
    - Your machine (NVIDIA G80 and CUDA environment)
- Gotchas
  - Make sure no one else is on console if remote

```
mamba<2044> who
Yliu pts/1         May 11 20:54 (d178-58-orchard-1.ucdavis.edu)
```

  - You will get device permission error, if someone else on console.

Setup

- Login to correct system:
  - Support has installed the NVIDIA driver, CUDA toolkit, and CUDA SDK onto :
    - adder mamba gopher crowned rattle cobra rat redbelly asp boa krait viper
  - The CUDA toolkit is installed under /opt/cuda and the CUDA SDK is installed under /opt/cuda-SDK.
- Setup environment variables:

```
setenv PATH ${PATH}:/opt/cuda/bin
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/opt/cuda/lib
```

- These can be added to the ~/.cshrc file or run manually.

Sample Code

- Flops (floating-point operations per second)
- Sum
- Branch
    - Provide the framework to analyze branch effect on performance

Using CUDA

- Code install
    - Flops code
    - Sum code
    - Branch code
- Evaluation mode setup
- Specific Tasks

Code Install

- Upload the branch.tar file into your NVIDIA_CUDA_SDK /projects directory on a snakes system
- Untar the directory tree
  - *tar –xf branch.tar*
- You should see the directory NVIDIA_CUDA_SDK
- Compile and run the program
  - *make clean*
  - *make*
  - *bin/linux/release/branch*

Task 1-Performance Scaling

- Goal
  - How much work you can get out of the GPU with various configurations of
    - Threads
    - Blocks
- Tasks:
  - Determine how performance in GFlops scales when changing threads per block
  - Determine how performance in GFlops scales with the number of blocks.

Task 2-Data Elements per Thread

- Note: If you have not installed and modified the sum project as specified in the optional section of the warmup, you must do that before continuing on in the project.
- Make sure your sum project adds up 1K (1024) elements in shared memory
- Tasks :
  - See how performance scales as you vary the amount of work per thread.
  - Does the performance track with your expectations, given the GPU architecture? Explain why or why not.

Task 3-Branch Effect on Performance

- One branch made in the kernel.
  - Call bigfunctiona or bigfunctionb
  - Extend the branch code and measure performance at all the other branch granularities: 256, 128, 64, 32, 16, 8, 4,…

```
testKernel() {
        if (threadIdx.x == 0) {
          bigfunctiona();
        } else if (threadIdx.x == 1) {
          bigfunctionb();
        } else if (threadIdx.x == 2) {
          bigfunctionc();
        } else …
      }
```

Task 3-Branch Effect on Performance

- Tasks:
  - See how performance scales with branch granularity, from 1 to 256 branches.
  - Save the source code for the version of your code that does 256 branches for handing in.

Warmup Problems (2)

- SUM 1000 elements
  - Modify the sum project to parallelize the code
  - Use 500 threads for the first addition, 250 for the next addition, and so on
  - Remove the print statements before you run

Submission

- Use the SmartSite to turn in
  - The working source code for the sum program, and the branch program (256-branch kernel),
  - The pdf report
  - A text file named "README" that describes what you made changes on the source code.
- **Due Date: Wed. 3 June at 5PM**