University of Illinois at Urbana-Champaign
Department of Electrical & Computer Engineering
ECE408 – Applied Parallel Programming

Programming Competition

1. Problem Description

The target application of the competition is a real-time x-ray imaging application. Please refer to the documentation to be release shortly for details about the theoretical background. We will also be having a presentation about it in class on Tuesday, November 13, 2012. Your objective in this competition is to accelerate the application using GPU.

2. Setup

Step 1: Update your local benchmarks repository to obtain the code needed for the assignment:

```
cd <path/to/parboil/directory>/benchmarks
hg pull
hg update
```

Step 2: The directory `competition/src/base/` contains the base C code for the application you are required to parallelize. Compile and test the base code.

```
./parboil compile competition base
./parboil run competition base <dataset>
```

You are provided with two datasets: `testquadcenter` and `armymannobad`. The first runs in a few seconds while the second is expected to take around 15 min.

Step 3: Create a copy of the `base` directory and call it `cuda`. This will be your working directory. Your task is to replace the existing C code with CUDA code that executes the application on GPU. The entry point to the code (main function) is located in `TomoSynth.cu`.

3. Judging Criteria

Contestants will be judged based on the speedup and execution time they are able to achieve. We are interested in the entire execution time of the program including allocation and copy time, not just the kernel time.

4. Presentations

The presentations will be held on Monday, December 10th, 2012 throughout the whole day. A sign-up sheet will be sent out closer to the date for students to reserve timeslots to present. In your presentation, you are expected to include:

- A very brief description of the problem that demonstrates your understanding of it
- An explanation of your overall parallelization strategy as well as interesting optimizations you have made to speed up your kernel(s)

- A summary of your experimental results
- The main challenges you faced while tackling the problem
- Other ideas you had but did not have time to try (if any)

5. Reports and Code

The report and code will be due on Friday, December 14, 2012 at 11:49 p.m. The report should cover the same points required for the presentation in more detail.

6. Submission

Submission of all material will be through the Sharepoint website. A submission folder will be provided closer to the deadline.

7. Tips and Guidelines

- It is up to you what functionality and data you would like to have on the host and what you would like to export to the device.
- The code emits information to standard output that informs that user about its progress. You are not required to preserve this feature as is in your ported code for functionality purposes. A minimal degree of progress information, such as number of frames processed so far, should be reported by your code to the user for you to receive a good score on code quality, but you should not worry about transferring data from the device to the host for the sake of providing detailed progress information.
- You do not need to maintain the code structure. Feel free to add/remove header files, inline functions, change when files are written, etc. Anything is fair game as long as the output files when the application terminates are the same.
- The tiff images generated by this code are 16bit tiff images which are not supported by most image viewers. A good viewer to use is called imagej which can be found at: http://rsb.info.nih.gov/ij/. One nice feature is that you can open a set of images and then step through them one at a time using: Ctrl+Shift+O.
- The header file `Reconstruct.h` contains macro definitions which can be used to set the parameters related to the location of the virtual image planes:
    - `D_SP` sets the distance from the sensor to the image plane
    - `DELTA_D_SP` set the spacing between planes
    - `NUM_PLANES` sets the total number of planes generated

The default values are: `D_SP = 35.4`, `D_DELTA_SP = 0.2`, and `NUM_PLANES = 96`. For `testquadcenter`, a good parameter is `D_SP = 38.764`. For `armmymannobad`, a good parameter is `D_SP = 46`.

For development and debugging purposes, you will probably want to compute a smaller number of planes and try different values for the other parameters. The test cases you are provided with were generated using the defaults. To test your own custom configurations, you may generate a reference using the base code (that is why it is important to retain a copy of the base code) then compare it to the output of your own code using the comparison script provided. Here is how to use the script to perform your own comparison:

- Run the base version to generate a reference output

  ```
  ./parboil run competition base <dataset>
  ```

- The output will be placed in `benchmarks/competition/run/<dataset>`. Rename it so that it doesn't get overwritten when you run your cuda version:

  ```
  mv <dataset> <dataset>_base
  ```

- Nor run your `cuda` implementation:

  ```
  ./parboil run competition cuda <dataset>
  ```

- Finally, compare the two datasets:

  ```
  benchmarks/competition/compare-datasets benchmarks/competition/run
     <dataset>_base <dataset>
  ```

- Since some jobs are expected to run for a long time on AC, you may find the following commands useful: `qdel`, `qpeek`

- You do not need to restrict yourself to the algorithm or implementation in the code. Feel free to think of alternative ways of doing things. For example, the current implementation of the reconstruction algorithm uses a "scatter" approach: it steps through the sensor space and then writes the weighted sensor data to the pixels it illuminates. Due to the geometry of the apparatus, neighboring sensor elements may write to the same pixel causing write contention, thereby the need for atomic writes. One optimization would be to step through the pixel space and pull in the appropriate sensor elements. Thus, instead of write contention, you will have multiple reads which will be cached.