Description | Code | Questions | Attempts | History

Lab Tour with Device Query

The coding deadline is 22 days from now (2015-07-01T23:00:00Z) .

Objective

The purpose of this lab is to get you familiar with using the submission system for this course and the hardware used.

Instructions

Click on the code tab and then read the code written. Do not worry if you do not understand all the details of the code (the purpose is to get you familiar with the submission system). Once done reading, click the "Compile & Run" button.

The submission system will automatically switch to the compile-and-run results that will also be available through the **Attempts** tab. There, you will be able to see a summary of your attempt.

The `Timer` section has 3 columns:

- *Kind* corresponds with the first argument to `wbTimer_start` ,
- *Location* describes the `file::line_number` of the `wbTimer` call, and
- *Time* in millisecond that it took to execute the code in between the `wbTime_start` and `wbTime_stop` , and
- *Message* the string you passed into the second argument to the timer

Similarly, you will see the following information under the `Logger` section.

The `Logger` section has 3 columns:

- *Level* is the level specified when calling the `wbLog` function (indicating the severity of the event),
- *Location* describes the `function::line_number` of the `wbLog` call, and
- *Message* which is the message specified for the `wbLog` function

The `Timer` or `Logger` seconds are hidden, if no timing or logging statements occur in your program.

We log the hardware information used for this course — the details which will be explained in the first few lectures.

- GPU card's name
- GPU computation capabilities
- Maximum number of block dimensions

- Maximum number of grid dimensions
- Maximum size of GPU memory
- Amount of constant and share memory
- Warp size

All results from previous attempts can be found in the Attempts tab. You can choose any of these attempts for submission for grading. Note that even though you can submit multiple times, only your last submission will be reflected in the Coursera database.

After completing this lab, and before proceeding to the next one, you will find it helpful to read the tutorial (/help) document

Suggestions

- The system's autosave feature is not an excuse to not backup your code and answers to your questions regularly.
- If you have not done so already, read the tutorial (/help)
- Do not modify the template code provided – only insert code where the `//@@` demarcation is placed
- Develop your solution incrementally and test each version thoroughly before moving on to the next version
- Do not wait until the last minute to attempt the lab.
- If you get stuck with boundary conditions, grab a pen and paper. It is much easier to figure out the boundary conditions there.
- Implement the serial CPU version first, this will give you an understanding of the loops
- Get the first dataset working first. The datasets are ordered so the first one is the easiest to handle
- Make sure that your algorithm handles non-regular dimensional inputs (not square or multiples of 2). The slides may present the algorithm with nice inputs, since it minimizes the conditions. The datasets reflect different sizes of input that you are expected to handle
- Make sure that you test your program using all the datasets provided (the datasets can be selected using the dropdown next to the submission button)
- Check for errors: for example, when developing CUDA code, one can check for if the function call succeeded and print an error if not via the following macro:

```
#define wbCheck(stmt) do {                                                    \
        cudaError_t err = stmt;                                               \
        if (err != cudaSuccess) {                                             \
            wbLog(ERROR, "Failed to run stmt ", #stmt);                       \
            wbLog(ERROR, "Got CUDA error ...  ", cudaGetErrorString(err));    \
            return -1;                                                        \
        }                                                                     \
    } while(0)
```

An example usage is `wbCheck(cudaMalloc(...))`. A similar macro can be developed while programming OpenCL code.

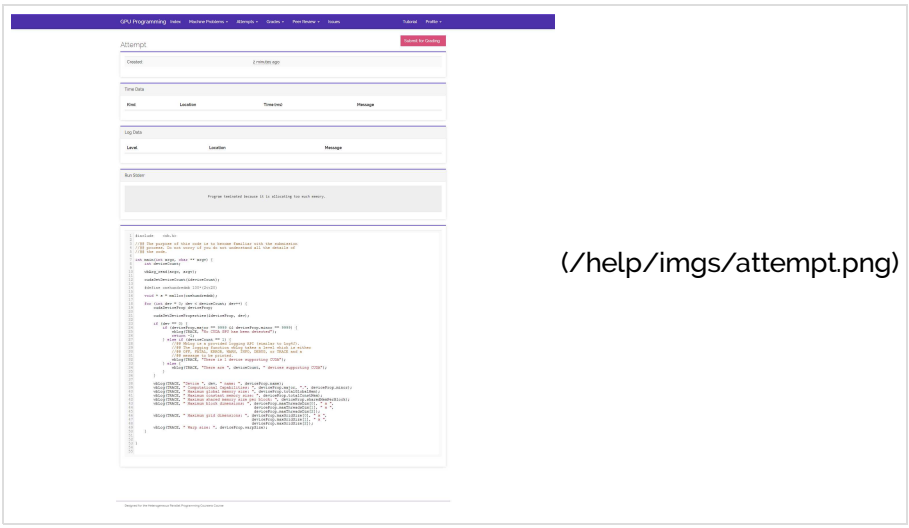
Plagiarism

Plagiarism will not be tolerated. The first offense will result in the two parties getting a 0 for the machine problem. Second offense results in a 0 for the course.

Grading

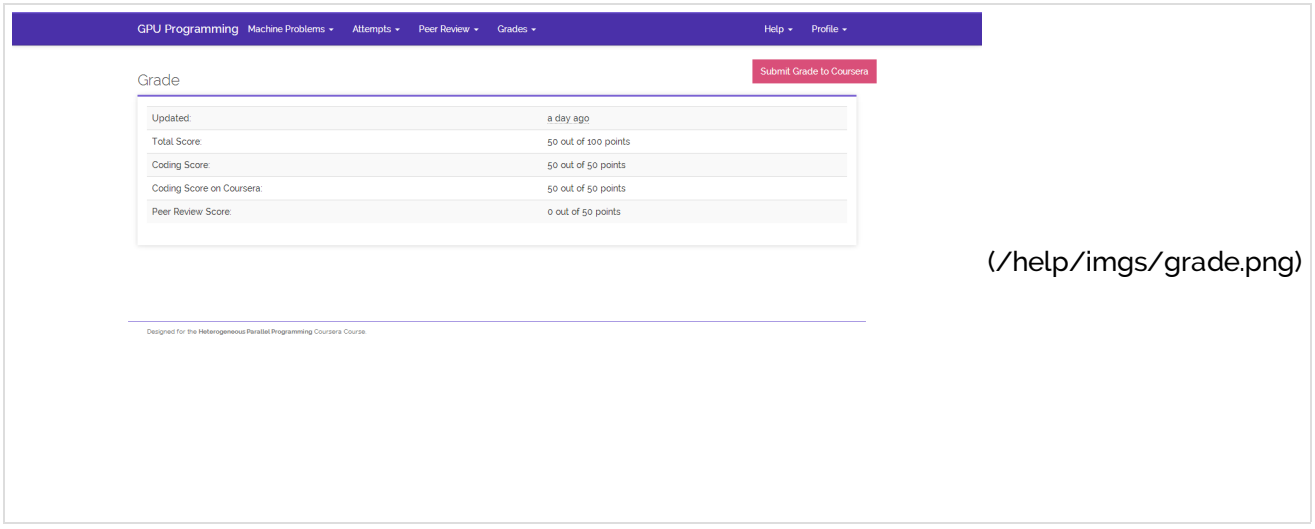
Grading is performed based on criteria that are specific for each lab.

For each attempt in the attempts tab, you'll see a *submit for grading* button.



(/help/imgs/attempt.png)

Clicking on that would tell the system to grade that attempt and redirect you to the grade report page.



(/help/imgs/grade.png)

The grades are automatically sent to Coursera. You will have to submit your grade back to Coursera by clicking the button submit to Coursera button.

Note: you cannot submit code for grading after the corresponding deadlines have passed.

Local Development

While not required, the library used throughout the course can be downloaded from Github (https://github.com/abduld/libwb). The library does not depend on any external library (and should be cross platform), you can use `make` to generate the shared object file (further instructions are found on the

Github page). Linking against the library would allow you to get similar behavior to the web interface (minus the imposed limitations). Once linked against the library, you can launch your program as follows:

```
./program -e <expected_output_file> -i <input_file_1>,<input_file_2> -o <output_file> -t <type>
```

The `<expected_output_file>` and `<input_file_n>` are the input and output files provided in the dataset. The `<output_file>` is the location you'd like to place the output from your program. The `<type>` is the output file type: `vector`, `matrix`, or `image`. If an MP does not expect an input or output, then pass `none` as the parameter.

In case of issues or suggestions with the library, please report them through the issue tracker (https://github.com/abduld/libwb/issues) on Github.

Issues/Questions/Suggestions

In case of questions, please post them to the forums. For issues or suggestions, please use the issue tracker (https://github.com/abduld/wb/issues).

Deadline

The coding deadline is 22 days from now (2015-07-01T23:00:00Z) .

Note: you cannot submit code and answer questions for grading nor perform a peer review after the corresponding deadlines have passed.

Grading Rubric

100 out of the 100 possible points is reserved for code development.

Designed for the Heterogeneous Parallel Programming (https://class.coursera.org/hetero-004) Coursera Course.