High Performance Computing

Homework #5 (Part B)

Due: Tuesday March 17 2015 by 11:59 PM (Midnight)

Email-based help Cutoff: 5:00 PM on Mon, March 16 2015

Maximum Points: 20

Submission Instructions

This homework assignment must be turned-in electronically via Niihka. Ensure your C++ source code is named *MUid*`_Matrix.cpp`, where *MUid* is your Miami University unique ID. You are expected to implement various methods in the `Matrix` class. Once you have implemented, tested, and benchmarked your implementation, upload the following onto Niihka:

1. The source file `MUid_Matrix.cpp`.
2. The report (duly filled-in) saved as a PDF document named with the convention `MUid_HW5Report_PartB.pdf`

Objective

The objective of this homework is to:

- Further explore the effectiveness of CPU caches by implementing a block matrix multiplication.
- Use Cilk array notation for implementing the inner-loop of Matrix addition.

Grading Rubric:

This is an advanced course and consequently the expectations in this course are higher. Accordingly, the program submitted for this homework must pass necessary tests in order to qualify for earning a full score.

NOTE: Program that do not compile, have methods longer than 25 lines, or just some skeleton code will be assigned zero score.

Scoring for this assignment will be determined as follows assuming your program compiles (and is not skeleton code):

- The points allocated for each one of the methods to be implemented in this homework is shown further below.

- **-1 Points**: for each warning generated by `icpc` when compiling your C++ program with the `-Wall` option.

- **-1 Points**: for each warning generated by the CSE departments' C++ style checker (a slightly relaxed version from Google Inc). On Red Hawk you can run the C++ style checker as shown below:

- **NOTE:** Points will be deducted for violating stylistic qualities of the program such as: program follows formatting requirements (spacing, indentation, suitable variable names with appropriate upper/lowercase letters, etc). The program includes suitable comments at appropriate points in each method to elucidate flow of thought/logic in each method. Program strives to appropriately reuse as much code as possible.

Starter Code:

In order to streamline this homework the following file(s) are supplied. Do not modify any of these files and do not submit them (your instructor will use the version supplied with this homework for grading purposes):

i. `Matrix.h`: This is a simple `Matrix` class that defines the API for Matrix.
ii. `MatrixTester.cpp`: This is a very simple top-level tester class with a `main` method for testing correct operation of your `Matrix` implementation.
iii. `BlockMatMul.cpp`: A simple benchmarking program to compare performance of your block-matrix multiplication approach against the simple implementation in `MatMul.cpp`.
iv. On Red Hawk, the following additional files are provided in the `/shared/raodm/csex43/data` directory for convenient testing:
    i. `hw5_mat_data.txt` : Input matrix data file
    ii. `hw5_mat_ref_output.txt` : Expected output from your implementation

Homework Exercise

This homework exercise involves developing a simple `Matrix` class that provides a convenient interface to perform some basic matrix operations. Refer to the documentation in the `Matrix.h` header file to implement the various methods. Most of the methods are pretty standard and use regular C++ conventions to implement the methods. The scoring for the various methods is listed below:

- All constructors (total): 4 points
- Block matrix multiplication using value of `blockSize` instance variable: 6 points.
- Matrix addition using Cilk array notation for columns (use standard `for`-loop to iterate over rows): 4 points.
- Stream insertion and stream extraction operators: 5 points.
- Diagonal sum: 1 point.

Input data format (for stream extraction):

The input data to the stream extraction operator is supplied in the following format:

```
<rows> WS <cols> WS <num> WS <num> ....
```

Where WS indicates 1 or more white spaces. There are exactly `rows` × `cols` values indicated by `<num>`.

Output data format (for stream insertion):

The output format for the stream insertion operator:

```
<rows> SPC <cols> NL
<num> SPC <num> .... NL
<num> SPC <num> .... NL
...                                  } One line per row
<num> SPC <num> .... NL              } of the matrix
```

Where SPC indicates exactly one 1 blank space and NL indicates newline. There is a trailing blank space at the end of each row of numbers (before the NL).

Functional testing:

Your submission would be tested using the following commands:

```
$ icpc -g -Wall -std=c++11 -O2 rao_Matrix.cpp MatrixTester.cpp -o MatrixTester
$ ./MatrixTester /shared/raodm/csex43/data/hw5_mat_data.txt  > my_output.txt
$ diff my_output.txt /shared/raodm/csex43/data/hw5_mat_ref_output.txt
```

The above `diff` command should not generate any output indicating outputs are consistent and indirectly verifying that method implementations in `Matrix.cpp` are correct.

Performance Verification

Once you have verified correct functionality of your implementation, the next phase is to compare the performance of standard matrix multiplication (implemented by `MatMul.cpp`) vs. block matrix multiplication implemented by you. In order to aid benchmarking, a `BlockMatMul.cpp` file is supplied.

Using the supplied files complete the supplied report document `HW5Report_PartB.docx`. Ensure you include a chart with trendlines indicating the equations and regression values (R^2 values) in the report. Record your inferences in the report document. Once you have completed your report save it as a PDF file using the convention `MUid_HW5Report_PartB.pdf`.

Turn-in:

This homework assignment must be turned-in electronically via Niihka. Ensure your C++ source code is named *`MUid`*`_Matrix.cpp`, where *`MUid`* is your Miami University unique ID. You are expected to implement various methods in the `Matrix` class. Once you have implemented, tested, and benchmarked your implementation, upload the following onto Niihka:

1. The source file `MUid_Matrix.cpp`.
2. The report (duly filled-in) saved as a PDF document named with the convention `MUid_HW5Report_PartB.pdf`

Upload all the necessary C++ source files to onto Niihka. Do not submit zip/7zip/tar/gzip files. Upload each source file independently.