Due by: **By 11:59 PM on Tue Mar 15 2016**

High Performance Computing

Homework #4 (Part B)

Due: Tuesday March 15 2016 by 11:59 PM

Email-based help Cutoff: 5:00 PM on Mon, March 14 2016

Maximum Points: 20

Submission Instructions

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

1. The source file `BlockMatrixOps.cpp`.
2. The report (duly filled-in) saved as a PDF document named with the convention `MUid_HW4Report_PartB.pdf`

Objective

The objective of this homework is to:

- Further understand the effective use of CPU caches by implementing a block matrix multiplication.
- Review reading and writing data to data files.
- Compare and contrast performance of block matrix multiplication.
- Gain familiarity with running programs via the PBS job scheduler on Miami's Red Hawk cluster

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 are shown further below.
- **-1 Points**: for each warning generated by `g++` 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:

```
$ /shared/raodm/bin/cpplint.py BlockMatrixOps.cpp
```

- **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. `MatrixOps.h`: This is a simple header file with prototype declarations for the various methods used by the `MatMul.cpp` tester class.
ii. `MatMul.cpp`: This is a very simple top-level tester class with a `main` method for testing correct operation of your block matrix implementation.
iii. `MatrixOps.cpp`: This is a reference implementation with a traditional matrix multiplication operation. This file also serves as a sample reference implementation for further developing block matrix multiplication.
iv. On Red Hawk, the following additional files are provided in the `/shared/raodm/csex43/homework4` directory for convenient testing.

Compiling:

In `NetBeans` place all the source and header files in project and compile them. If you are just compiling from the command-line you can just specify all the source files as part of the command-line as shown below:

```
$ g++ -O3 -std=c++11 -g -Wall MatMul.cpp BlockMatrixOps.cpp
MatrixOps.cpp -o MatMul
```

Homework Exercise

This task of the homework exercise involves developing the methods in the supplied `BlockMatrixOps.cpp` file. The functional description for each one of the methods in the BloackMatrixOps.cpp file is supplied below:

Methods (and descriptions)	Points
`Matrix blockMultiply(const Matrix& matrix1, const Matrix& matrix2, const size_t blockSize)` This method returns a Matrix that is the result of multiplying the two supplied matrices `matrix1` and `matrix2`. The method may assume matrix1 and matrix2 are of compatible sizes for multiplication. The block size to be used for matrix multiplication is supplied as the 3rd parameter. Note: In order to earn full points the block matrix version must be slightly faster than the traditional version (implemented in `MatrixOps.cpp`) with `-O3` optimization level with `g++` for a block size of 50 with a 1000×1000 matrix.	8
`Matrix load(const std::string& filePath)` This method returns a matrix whose data is loaded from a given file. The path to the file is supplied as the parameter to this method. See details on the input data format further below. The method may assume that `filePath` is valid.	4
`void write(const std::string& filePath, const Matrix& mat)` This method must write the supplied matrix `mat` to a given file. Path to the output file is specified by the first parameter `filePath`. See details on the output data format further below. The method may assume that `filePath` is valid.	3

Input data format (for reading matrix data):

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 writing matrix data):

The output format for the stream insertion operator:

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

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

Block matrix multiplication:

Section 8.2 of in the textbook for this course titled "Introduction to Parallel Computing" provides an overview of Block matrix multiplication. Review the textbook section for details. However, note that in this homework you are not expected to parallelize any of the operations. However, your program will need to appropriately handle cases where the matrix size is not an event multiple of block size.

Functional Testing

You can verify correct operation of your implementation in `BlockMatrixOps.cpp` using the following sample commands and outputs (your submission would also be tested using the following commands):

```
$ ./MatMul /shared/raodm/csex43/homework/homework4/test1/mat1x3.txt
/shared/raodm/csex43/homework/homework4/test1/mat3x1.txt test1_out.txt 4
Multiplying matrices from files:
/shared/raodm/csex43/homework/homework4/test1/mat1x3.txt and
/shared/raodm/csex43/homework/homework4/test1/mat3x1.txt using block
multiplication, block size: 4
Block matrix multiplication...
Diag sum: 14
$ diff test1_out.txt
/shared/raodm/csex43/homework/homework4/test1/test1_ref_out.txt
$
```

```
$ ./MatMul /shared/raodm/csex43/homework/homework4/test2/mat3x3.txt
/shared/raodm/csex43/homework/homework4/test2/mat3x3.txt test2_out.txt 2
Multiplying matrices from files:
/shared/raodm/csex43/homework/homework4/test2/mat3x3.txt and
/shared/raodm/csex43/homework/homework4/test2/mat3x3.txt using block
multiplication, block size: 2
Block matrix multiplication...
Diag sum: 82
$ diff test2_out.txt
/shared/raodm/csex43/homework/homework4/test2/test2_ref_out.txt
$
```

```
$ ./MatMul /shared/raodm/csex43/homework/homework4/test3/mat2x3.txt
/shared/raodm/csex43/homework/homework4/test3/mat3x2.txt test3_out.txt 2
Multiplying matrices from files:
/shared/raodm/csex43/homework/homework4/test3/mat2x3.txt and
/shared/raodm/csex43/homework/homework4/test3/mat3x2.txt using block
multiplication, block size: 2
Block matrix multiplication...
Diag sum: 212
$ diff test3_out.txt
/shared/raodm/csex43/homework/homework4/test3/test3_ref_out.txt
$
```

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

```
$ ./MatMul /shared/raodm/csex43/homework/homework4/test4/mat70x50.txt
/shared/raodm/csex43/homework/homework4/test4/mat50x70.txt test4_out.txt 19
Multiplying matrices from files:
/shared/raodm/csex43/homework/homework4/test4/mat70x50.txt and
/shared/raodm/csex43/homework/homework4/test4/mat50x70.txt using block
multiplication, block size: 19
Block matrix multiplication...
Diag sum: 1993668
$ diff test4_out.txt
/shared/raodm/csex43/homework/homework4/test4/test4_ref_out.txt
$
```

```
$ ./MatMul /shared/raodm/csex43/homework/homework4/test5/mat50x70.txt
/shared/raodm/csex43/homework/homework4/test4/mat70x50.txt test5_out.txt 7
Multiplying matrices from files:
/shared/raodm/csex43/homework/homework4/test5/mat50x70.txt and
/shared/raodm/csex43/homework/homework4/test4/mat70x50.txt using block
multiplication, block size: 7
Block matrix multiplication...
Diag sum: 1993668
$ diff test5_out.txt
/shared/raodm/csex43/homework/homework4/test5/test5_ref_out.txt
$
```

```
$ ./MatMul /shared/raodm/csex43/homework/homework4/test5/mat50x70.txt
/shared/raodm/csex43/homework/homework4/test4/mat70x50.txt test5_out.txt 27
Multiplying matrices from files:
/shared/raodm/csex43/homework/homework4/test5/mat50x70.txt and
/shared/raodm/csex43/homework/homework4/test4/mat70x50.txt using block
multiplication, block size: 27
Block matrix multiplication...
Diag sum: 1993668
$ diff test5_out.txt
/shared/raodm/csex43/homework/homework4/test5/test5_ref_out.txt
$
```

```
$ ./MatMul /shared/raodm/csex43/homework/homework4/test5/mat50x70.txt
/shared/raodm/csex43/homework/homework4/test4/mat70x50.txt test5_out.txt 1
Multiplying matrices from files:
/shared/raodm/csex43/homework/homework4/test5/mat50x70.txt and
/shared/raodm/csex43/homework/homework4/test4/mat70x50.txt using block
multiplication, block size: 1
Block matrix multiplication...
Diag sum: 1993668
$ diff test5_out.txt
/shared/raodm/csex43/homework/homework4/test5/test5_ref_out.txt$ diff
test5_out.txt /shared/raodm/csex43/homework/homework4/test5/test5_ref_out.txt
$
```

Note: The above sample outputs the `diff` command should not generate any output indicating outputs are consistent and indirectly verifying that method implementations in `BlockMatrixOps.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 (already implemented for you in `MatrixOps.cpp`) vs. block matrix multiplication implemented by you. The supplied `MatMul.cpp` already has an interface built-in to aid benchmarking. The three different versions of the program can be run by appropriately modifying the following PBS script:

```
#!/bin/bash

#PBS -N MatMul_MUid
#PBS -l walltime=1:00:00
#PBS -l mem=4GB
#PBS -l nodes=1:ppn=1

cd $PBS_O_WORKDIR
MAT_SIZE=1000
BLOCK_SIZE=50

for rep in `seq 1 5`
do
    /usr/bin/time -v ./MatMul $MAT_SIZE 0
    /usr/bin/time -v ./MatMul $MAT_SIZE $BLOCK_SIZE
done

# end of script
```

Using the runtime statistics supplied files complete the supplied report document `HW5Report_PartB.docx`. Ensure you include a chart with:

- 95% CI error bars
- Trendlines indicating the equations and regression values (R^2 values) in the report.

Tutorial on adding error bars and trend line to Excel charts are available at Microsoft Office Support Link and corresponding tutorial for LibreOffice is available at LibreOffice/OpenOffice Help Link. Record your inferences in the report document. Once you have completed your report save it as a PDF file using the convention `HW4Report_PartB.pdf`.

Turn-in:

This homework assignment must be turned-in electronically via Canvas. Ensure your C++ source code is named *`BlockMatrixOps`*`.cpp`, where you are expected to implement various methods in the file. Once you have implemented, tested, and benchmarked your implementation, upload the following onto Canvas:

1. The source file `BlockMatrixOps.cpp`.
2. The report (duly filled-in) saved as a PDF document named with the convention `MUid_HW4Report_PartB.pdf`

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