**LAB 2: WORKING WITH OPENMP PROGRAMS ON CLUSTER**

**GOAL**

In this lab, you will learn about programming and performance of OpenMP programs running on shared memory MIMD. Furthermore, you will compare the differences of algorithm designs, implementations, and performances obtained due to the choice of algorithms, languages, constructs used, data layout, and order of operations in parallel implementations.

Any questions/concerns, please send email to [pdslab@ucdenver.edu](mailto:pdslab@ucdenver.edu).

**SUBMISSION**

* Provide answers for each question in **REQUIREMENTS** section. Place your answers in a word document (.docx) and name it as “Lab2\_LastName\_FirstName.docx.”
* This lab requires you to develop codes. Name your code as **programName-firstName.cpp.**
* Submit your document and codes on Canvas (do not zip/compress your files).
* This lab is worth 40 and 5 extra credits.

**Gaussian Elimination**

In linear algebra, Gaussian Elimination is an algorithm for solving systems of linear equations, finding the rank of a matrix, and calculating the inverse of an invertible square matrix. For example, given matrix A4x4, the Gaussian Elimination converts matrix **A** into an upper triangle matrix **A4**.

* ***For example:***

|  |  |  |  |
| --- | --- | --- | --- |
| **A =** | 4 8 4 0  1 5 4 -3  1 4 7 2  1 3 0 -2 | 🡪 **A1 = …** 🡪 **A2** = … 🡪 **A3 =** | 4 8 4 0  0 3 3 -3  0 0 4 4  0 0 0 1 |

**REQUIREMENTS**

1. (4 pts) Read the source code of Version 1 of Gaussian Elimination program (choose either C++ version (**cge-omp1.cpp**) or Fortran version (**fge-omp1.f90)**) in Section 1 and apply it to compute matrix A1 and A2 in the above example. Note that to earn full credits, you must show how to calculate the intermediate matrices: A1 and A2, manually.
2. (6 pts) Read the source code of Version 1 of Gaussian Elimination program and answer the following questions:
3. What scheduling method is applied in the parallel loops used in the program?
4. What is the method used to synchronize the values of variables **gmax** (global max), **gindmax** (global max index)?
5. Explain why this synchronization is needed?

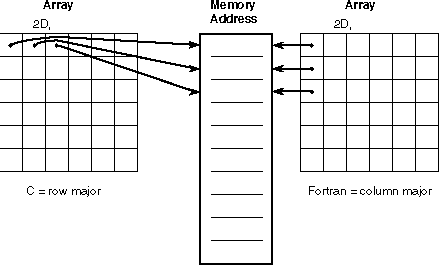
You may want to compile and run the codes: **cge-omp.cpp** or **fge-omp1.f90,** whichmayhelp answer the above questions. See Section 1,2,3 for compiling and running instructions.

1. (20 pts) Write Version 2 of cge-omp1.cpp (Parallel Gaussian Elimination C++ program) using OpenMP, name the new version source code file as **cge-omp2-FirstName.cpp**. In this version, you will use parallel region instead of parallel loops by applying the directive "**#pragma omp parallel**" outside the outer “For” loop of the **ComputeGaussianElimination** function. Also, you need to use the **critical section** construct for synchronization and use **static scheduling method** to schedule the workload of the “for” loops. Then, compile your code and generate a compiled code named **cge-omp2**. Test your program to be sure it produces correct outputs.

In order to get a full grade for this question you have to submit your code as attachment also explain how you designed your parallel code and input/output matrix samples and one screenshot that shows the correctness of your program.

**Hint:** Please refer to the Version 2 in Fortran and the **Openmp Tutorials at** <http://pds.ucdenver.edu/webclass/OpenMPDocumentation.html>(especially the OpenMP Application Program Interface) to write your program. However, please notice that the Fortran codes are using column wise data layout while your C++ program should use row wise data layout for optimal performance.

**Note**: 2D array of in the C++ and Fortran versions have different memory layout.



1. (5 pts) Compile/Run **cge-omp1.cpp, fge-omp1.cpp, fge-omp2.cpp,** and **cge-omp2-FirstName.cpp** on Heracles with different number of threads (See the Table below) and matrix size **n = 6432**. Collect the runtime results in a table and plot the runtime of all programs in a chart (x-axis for number of threads, y-axis for runtime). For each input of thread count, you should run the program at least 2 times and get their average runtime.

**Runtime in seconds for matrix size = 6432**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| threads -> | 1 | 2 | 4 | 8 | 16 | 24 | 48 |
| cge-omp1 |  |  |  |  |  |  |  |
| cge-omp2 |  |  |  |  |  |  |  |
| fge-omp1 |  |  |  |  |  |  |  |
| fge-omp2 |  |  |  |  |  |  |  |

1. (5 pts) Explain why the runtimes of C++ Version 2 and Fortran Version 1 are so different from the execution time of C++ Version 1 and Fortran Version 2?

**OpenMP AFFINITY (extra credits)**

In this section we are going to explore different affinities to run your cge-omp2. Read <https://www.ixpug.org/documents/1506981937ixpugfall2017_21_up2.pdf> to learn about affinities.

1. (2 pts) Change the Slurm script to export the appropriate affinity according to the table and run your code with matrix size = 6432. In order to perform the next experiments, make sure the right OMP\_PROC\_BIND parameter is set correctly inside the slurm script. For this, set the following parameter in the **cge-omp2\_slurm.sh** script:

// use this configuration in the slurm job to run the first experiment

export OMP\_PROC\_BIND='close'

// use this configuration in the slurm job to run the second experiment

export OMP\_PROC\_BIND=’spread’

**OMP\_PROC\_BIND=close**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| threads -> | 2 | 4 | 8 | 16 | 24 | 48 |
| cge-omp2 |  |  |  |  |  |  |

**OMP\_PROC\_BIND=spread**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| threads -> | 2 | 4 | 8 | 16 | 24 | 48 |
| cge-omp2 |  |  |  |  |  |  |

1. (3 pts) Explain the main differences among affinities. According to the table, which affinity reached a better performance? Why?

**SECTION 1: SOURCE CODES**

**OpenMP Gaussian Elimination** source codes are available in both C++ and Fortran languages. Please read documentation for all OpenMP directives and routines used in the programs. Please also get yourself familiar with some basic Fortran syntaxes.

* **cge-omp1.cpp** *(Gaussian Elimination – C++ OpenMP Version 1)*
* **fge-omp1.f90**  *(Gaussian Elimination – Fortran OpenMP Version 1)*
* **fge-omp2.f90** *(Gaussian Elimination – Fortran OpenMP Version 2)*

**SECTION 2: COMPILING**

Please follow instructions in Lab 1 and make sure the source codes are copied to your home directory

Compiling the given codes using following commands:

$ g++ -O -fopenmp cge-omp1.cpp -o cge-omp1

$ gfortran -O -fopenmp fge-omp1.f90 -o fge-omp1

$ gfortran -O -fopenmp fge-omp2.f90 -o fge-omp1

Compiling your code in **REQUIREMENTS:** problem 4 using following commands:

$ g++ -O -fopenmp **cge-omp2-FirstName.cpp** -o cge-omp2

For more information, please read:

* Compiling sequential programs on Heracles (C, C++ and Fortran)

<http://pds.ucdenver.edu/webclass/Compiling%20C_C++%20and%20Fortran%20programs.html>

* Compiling OpenMP programs on Heracles

<http://pds.ucdenver.edu/webclass/Compiling%20openMP%20programs.html>

**SECTION 3: RUNNING PROGRAMS ON HERACLES**

Use the following Slurm scripts to run each code

* **cge-omp1\_slurm.sh** (script to run cge-omp1)
* **cge-omp2\_slurm.sh** (script to run cge-omp2)
* **fge-omp1\_slurm.sh** (script to run fge-omp1)
* **fge-omp2\_slurm.sh** (script to run fge-omp1)

The OpenMP version of Gaussian Elimination program takes three arguments as its inputs.

1st argument: the size of the matrix

2nd argument: the number of threads to run the program with

3rd argument: an option for printing the matrix (1=print; 0= not print). Program only prints if the matrix size is less than 10.

From the master node run:

$ sbatch scriptName 1st argument 2nd argument 3rd argument

Please read instructions in Lab 1, if you want to customize your Slurm scripts.