**LAB 3: 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 difference of algorithm design, implementation, and resulting performance obtained due to the choice of algorithms, languages, constructs used, data layout, and order of operations in parallel implementations.

**SUBMISSION**

1. You have to submit this document (.docx) with your answers.
   1. Comments and grades will be added to your documents **so do not submit your homework as a PDF.**
2. Rename this document to Lab1\_Last-FirstName.docx
3. If this lab requires you to develop any code, so name your code to progName\_firstlastname.cpp. Example: cge2\_ Linck-Iris.cpp
4. Use CANVAS to submit your lab answers/results as Word documents (.doc, .docx, .xls) attachments. If the lab assignment requires program implementation, submit the code (program and input) files on canvas as well as the commands for compiling and running them.
5. You are not allowed to run your code without using slurm job (use the scripts provided for you). Slurm guarantees that your code will run in an available node(s) without any other concurrent job.
6. Send your questions to [pdslab@ucdenver.edu](mailto:pdslab@ucdenver.edu) **with subject “CSC4551 Questions Lab 3” with copy to iris (**iris.linck@ucdenver.edu) **and Manh (**manh.huynh@ucdenver.edu).

**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**.

1. ***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. (2.0) Read Version 1 of Gaussian Elimination program (choose either C++ or Fortran version) in the Source Code section and apply it to compute matrix A1 and A2 in the above example.

**Note:** You have to show how to calculate manually the intermediate matrices, A1 and A2, and put here.

1. (3.0) Read the **Version 1 of OpenMP Gaussian Elimination** program in the **Source Code** section and identify following:
2. What scheduling method is applied in the parallel loops used in the program?
3. What is the method used to synchronize the values of variables **gmax** (global max), **gindmax** (global max index)?
4. Explain why this synchronization is needed?

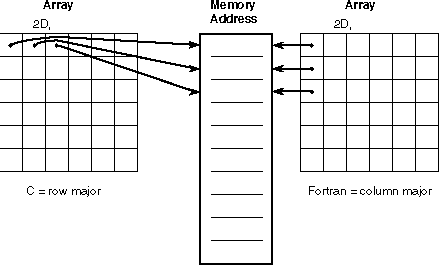
Compile source code files of three programs **cge-omp1, fge-omp1, fge-omp2**. (See Section 2)

1. (3.0) Write Version 2 of cge-omp1.cpp (Parallel Gaussian Elimination C++ program) using OpenMP, name the new version source code file as **cge-omp2\_Last-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. (2.0) Run **cge-omp1, cge-omp2, fge-omp1** and **fge-omp2** on Heracles with different number of threads (See the Table) 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 every input of thread count, you should run the program at least 2 times and get their average runtime.

**Note**: 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.

Before running your codes try to execute the following command in order to verify the cpu info on Heracles

ssh node[2-17] lscpu | grep -i 'core\|thread\|socket'

**\*\* choose any node on heracles from 2 to 15**

**Example:** ssh node2 lscpu | grep -i 'core\|thread\|socket'

**Run your job :**

**sbatch openmp\_slurm.sh matrixSize numofthreads**

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

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

1. (2.0) 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

In this section we are going to explore different affinities to run your cge-omp2 **(Read section 4 to learn about that).**

1. (1.0) Change the slurm job to export the appropriate affinity according to the table and run your code for matrix size = 6432

In order to perform the next experiments make sure the right OMP\_PROC\_BIND parameter was exported correctly inside the slurm job. For this, check the following parameter in your openmp\_slurm.sh job:

// 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’

**Run your job :**

**sbatch openmp\_slurm.sh matrixSize numofthreads**

**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. (2.0)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++ language and FORTRAN language. Please read documentation for all OpenMP directives and routines used in the programs.

* **cge-omp1.cpp** *(Gaussian Elimination – C++ OpenMP Version1)*
* **fge-omp1.f90**  *(Gaussian Elimination – Fortran OpenMP Version 1)*
* **fge-omp2.f90** *(Gaussian Elimination – Fortran OpenMP Version 2)*

# Section 2: Compiling programs on Heracles

**Step 1: Copy source code files to Heracles**

* Logon the cluster (see Lab 1)
* Make a folder named lab2 in your home directory using the **mkdir** command.

**mkdir /path/to/directory**

For example:

mkdir /home/john/lab2

* Copy the source code files to /lab2. If you are using MAC or Linux, you can copy these files using **scp** or **sftp** command. If you are using Windows, you can use Bitvise, WinSCP, or SSH Secure Shell to login and copy files from your PC to the cluster.
* Change the working directory to /lab2 using **cd** command

**cd /path/to/directory**

For example:

cd /home/john/lab2

* **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>

**Example how to compile the codes**

g++ -O -fopenmp cge-omp1.cpp -o cge1

gfortran -O -fopenmp fge-omp1.f90 -o fge1

gfortran -O -fopenmp fge-omp2.f90 -o fge2

# Section 3: SCRIPTS – Slurm jobs

## Use the following scripts to run each code

* **c\_slurm.sh** (script to run C/C++ sequential code)
* **openmp\_slurm.sh** (script to run openMP code in C/C++ or Fortram)

## How to run the scripts

* From the master node run:

sbatch scriptName argument1 argument 2

**Example:**

**sbatch c\_slurm.sh parameter 1 parameter 2**

**Note: you may change some parameters in your script, such as:**

#SBATCH --mail-user=myemailaddress ### put your email

#SBATCH --job-name=myjob ### Job Name

#SBATCH --output=slurm\_output.%j ### File in which to store job output

#SBATCH --error=slurm\_error.%j ### File in which to store job error messages

**In order to check the result of your job use:**

* squeue // check the job queue
* scancel jobID // cancel job execution
* cat slurm\_output.jobID // catches the runtime and diagnostic information.
* cat slurm\_error.jobID // catches errors and the profiling for cuda code (gpu)
* Check this link for more information about Slurm scripts:

<http://pds.ucdenver.edu/webclass/Heracles-RunningPrograms%20Slurm.html>

# Section 4- Learning about thread affinity on openMP in the following link

<https://www.ixpug.org/documents/1506981937ixpugfall2017_21_up2.pdf>

