DSCC 201/401 Homework Assignment #4
Due: **October 13, 2021 at 9 a.m. EDT**

Answers to these questions should be submitted via Blackboard. Questions 1-8 should be answered by all students (DSCC 201 and 401) and Questions 9 and 10 should be answered by students registered in DSCC 401. Make sure to answer all parts of the questions to receive credit. Please upload a file containing your answers and explanations to Blackboard (Homework #4: Software Parallelization Models and Techniques) as a Word document (docx) or PDF.

1. Create a new directory in your home directory called *parallel*. Copy the file */public/bmort/parallel/matrix_mult.c* to this directory. Examine the source code of *matrix_mult.c*. Which parallel computing library does this program use?

2. Compile the *matrix_mult.c* source code using the *gcc* compiler. Make sure to include the option for the correct parallel library when compiling your code. What command did you use to compile the source code?

3. Create a Slurm script to run a batch job for the binary produced from the compilation step in question 2. You should request exactly 4 cores and 2 GB of RAM to run your compiled binary. Set the partition to debug. Print out your complete Slurm script.

4. Submit your batch job to run in the debug partition on the BlueHive cluster. What command did you use to submit the job? How do you know the job has finished? Print the output from the job.

5. Notice that the wall time for the calculation is printed in the output. You can benchmark this code by varying the number of cores for the task. Start with 1 core and increase the values by powers of two up to the maximum value for a BlueHive node in the debug partition (i.e. 1, 2, 4, 8, 16). Include the output from each run of your benchmark and take note in the output of the hostname of the node where your benchmark runs. A) Would it be possible to get better performance of the binary compiled from the *matrix_mult.c* source code if you ran it on more than one node? Why or why not? B) Would the performance of *matrix_mult* program improve if you requested a GPU with the Slurm option --gres=gpu:1 and submitted the program to the gpu-debug partition? Why or why not?

6. Using a plotting program (e.g. Excel, Python, GNUplot, etc.), construct a graph showing the time it takes for the job to complete (wall time in seconds) vs. the number of cores. Calculate the speedup for each of the data points and create an additional plot that shows speedup (S) vs. cores (N). Embed your graph in the document (e.g. Word document or PDF) that you will submit for this homework assignment.

7. What fraction of the work (p) in the *matrix_mult* program is parallel? Show and explain how you derived your answer.

8. What is linear scaling? Does the *matrix_mult* program exhibit linear scaling? Why or why not?

9. **(DSCC 401 Only)** You are now going to run the *matrix_mult.c* program directly on one of the hosts in a **Default** FastX session on BlueHive. After logging into a Default desktop environment on BlueHive, open the terminal application. Using the *hostname* command, what is the host name of the node you are on? Now set an appropriate environment variable to control the number of threads for your *matrix_mult.c* program. Set the appropriate environment variable to use 1 thread. Run the matrix multiplication program directly on the command line in the terminal session. Take note of the wall time to execute. Change the number of threads to 2 using the environment variable and run the program again, noting the wall time. Repeat for 4 and 8 threads. Plot the timing data as you did in question 6 (wall time vs. number of threads and speedup vs. number of threads). Are there any significant differences between the graphs obtained in question 6 and this question? Why or why not?

10. **(DSCC 401 Only)** Read the paper, “Deep Learning at 15PF,” by Kurth, et al. A copy of this paper has been uploaded to the Blackboard site and is available under the instructions for this homework assignment (kurth.pdf). Provide detailed answers to the following questions:

    A. Examine Figure 6. Focus on the "synchronous" approach (blue line). If you were performing an analysis of HEP, would you recommend running the calculation on 1024 nodes? Why or why not?

    B. Based on the data in Figure 6, provide a rough estimate for the fraction of the synchronous algorithm that is parallelized.

    C. Again, looking at Figure 6, why would the speedup for 1024 nodes be less than the speedup for 512 nodes when using the synchronous approach?