CSCI-UA.0480-051: Parallel Computing
Midterm Exam (March 14th, 2024)

Total: 100 points
Important Notes – READ BEFORE SOLVING THE EXAM

- If you perceive any ambiguity in any of the questions, state your assumptions clearly and solve the problem based on your assumptions. We will grade both your solutions and your assumptions.
- This exam is take-home.
- The exam is posted on Brightspace at the beginning of the March 14th lecture (2 pm EST).
- You have up to 24 hours to submit on Brightspace (i.e., until March 15th, 2 pm EST), in the same way as you submit an assignment. However, unlike assignments, you can only submit once.
- Your answers must be very focused. You may be penalized for giving wrong answers and for putting irrelevant information in your answers.
- Your answer sheet must be organized as follows:
  - The very first page of your answer must contain only:
    - Your Last Name
    - Your First Name
    - Your NetID
    - Copy and paste the honor code shown below.
  - In your answer sheet, answer one problem per page. The exam has seven main problems, each one must be answered on a separate page.
- This exam consists of 7 problems, with a total of 100 points.
- Your answers can be typed or written by hand (but with clear handwriting). It is up to you. But you must upload one pdf file containing all your answers.

Honor code (copy and paste to the first page of your exam)

- You may use the textbook, slides, the class recorded lectures, the information in the discussion forums of the class on Brightspace, and any notes you have. But you may not use the internet.
- You may NOT use communication tools to collaborate with other humans. This includes but is not limited to Google-Chat, Messenger, Email, etc.
- You cannot use LLMs such as ChatGPT, Gemini, Bard, etc.
- Do not try to search for answers on the internet; it will show in your answer, and you will earn an immediate grade of 0.
- Anyone found sharing answers, communicating with another student, searching the internet, or using prohibited tools (as mentioned above) during the exam period will earn an immediate grade of 0.
- "I understand the ground rules and agree to abide by them. I will not share answers or assist another student during this exam, nor will I seek assistance from another student or attempt to view their answers."

Problem 1

a. [10] Suppose we have a core with only superscalar execution (i.e., no pipelining or hyperthreading). Will this core benefit from having a larger instruction cache? Justify your answer in 1-2 lines.
b. [10] Can a single process be executed on a distributed memory machine? If yes, explain how, in 1-2 lines. If not, explain why not.
c. [10] Can several threads, belonging to the same process, be executed on a shared memory machine and get the same performance as when executed on a single-core machine with hyperthreading? If yes, explain how, in 1-2 lines. If not, explain why not.
d. [6] If we have an eight-way superscalar core, what is the minimum number of ALUs (Arithmetic Logic Units) needed to avoid performance bottlenecks due to ALU limitations? Justify.

Problem 2

Consider a parallel program that calculates the sum of elements in a large array. The array is divided into chunks, and each process calculates the sum of its assigned chunk. The final sum is obtained by summing the partial sums from each process.

a. [10] Describe a parallel algorithm using MPI to accomplish this task. Include relevant MPI calls.

b. [10] Analyze the runtime complexity of your algorithm, considering both computation and communication time. Assume the array has N elements and P processes.

Problem 3

Suppose that MPI_COMM_WORLD consists of the four processes 0, 1, 2, and 3, and suppose the following code is executed after MPI has been initialized (`my_rank` contains the rank of the executing process):

```
int x, y, z;
switch(my_rank) {
case 0:
x=1; y=2; z=3;
MPI_Send(&x, 1, MPI_INT, 1, 1, MPI_COMM_WORLD);
MPI_Recv(&z, 1, MPI_INT, 3, 3, MPI_COMM_WORLD, &status);
break;
case 1:
x=4; y=5; z=6;
MPI_Recv(&x, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, &status);
MPI_Send(&y, 1, MPI_INT, 2, 2, MPI_COMM_WORLD);
break;
case 2:
x=7; y=8; z=9;
MPI_Recv(&y, 1, MPI_INT, 1, 2, MPI_COMM_WORLD, &status);
MPI_Send(&z, 1, MPI_INT, 3, 3, MPI_COMM_WORLD);
break;
case 3:
x=10; y=11; z=12;
MPI_Recv(&z, 1, MPI_INT, 2, 3, MPI_COMM_WORLD, &status);
break;
}
```

a. [12 points] What will be the values of x, y, and z for each of the 4 processes after executing the above code?

	x	y	z
P0			
P1			
P2			
P3			

b. [5] Explain how MPI handles blocking sends and receives in this code.
c. [8] What will happen if process 1 fails before sending its message?

Problem 4

Consider a parallel algorithm for matrix multiplication. We have two NxN matrices, A and B, and we want to compute their product C.

a. [10] Describe a parallel algorithm using threads to perform this matrix multiplication. Assume the matrices are divided into blocks, and each thread handles a block of the computation.

b. [10] Analyze the scalability of your algorithm. How does the runtime change as the number of threads increases? Consider both computation and synchronization overhead.

Problem 5

a. [10] Explain Amdahl's Law and its implications for parallel programming. Give an example.

b. [10] Explain Gustafson's Law and how it differs from Amdahl's Law. Give an example.

Problem 6

a. [10] Describe the concept of a critical section and how it relates to race conditions in parallel programming.

b. [10] Explain different mechanisms for handling critical sections, such as mutexes and semaphores. Discuss their advantages and disadvantages.

Problem 7

a. [10] Describe the difference between strong and weak scaling in parallel computing.

b. [10] Give examples of applications that would benefit more from strong scaling and applications that would benefit more from weak scaling. Justify your choices.