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 in the rectangle at the bottom of this page.
  - In your answer sheet, answer one problem per page. The exam has eight main problems; each one must be answered on a separate page.
- This exam consists of 8 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] Explain the difference between cache coherence and memory consistency. Provide a concrete example illustrating a scenario where cache coherence is maintained but memory consistency is violated.

b. [10] What are the advantages and disadvantages of using a distributed shared memory (DSM) system compared to a traditional shared memory system?

c. [10] Describe Amdahl's Law and its implications for parallel program performance. Give a specific example illustrating how Amdahl's Law limits speedup even with a large number of processors.

d. [6] Suppose we have a two-way SMT core. How many instruction decoders would you recommend to maximize instruction-level parallelism? Justify your answer.

Problem 2

Suppose we have the following DAG that represents different tasks and their dependencies: (A DAG would be inserted here as a figure. Assume tasks A, B, C, D, E, F, G, H exist with various dependencies depicted in the DAG).

The following table shows the execution time of each task if we execute it on a core of type X and if we execute it on a core of type Y.

Task	CPU type X	CPU type Y
A	2	3
B	5	2
C	10	8
D	8	12
E	3	4
F	7	5
G	4	6
H	6	3

a. [10] What is the minimum number of CPUs of each type that we need to get the highest speedup over sequential execution on CPU of type X? Show which CPU will execute which task(s) and calculate the final speedup.

b. [10] Repeat the problem above but using CPU of type Y.

c. [10] If you could add one dependency (add an arrow to the DAG), which one would you add to potentially improve performance through better scheduling? Justify your choice.

Problem 3

Suppose that MPI_COMM_WORLD consists of four processes (0, 1, 2, 3), and the following code is executed:

```
int x, y;
MPI_Status status;
switch(my_rank) {
case 0:
x = 10;
MPI_Send(&x, 1, MPI_INT, 1, 10, MPI_COMM_WORLD);
MPI_Recv(&y, 1, MPI_INT, 2, 20, MPI_COMM_WORLD, &status);
break;
case 1:
MPI_Recv(&x, 1, MPI_INT, 0, 10, MPI_COMM_WORLD, &status);
MPI_Send(&x, 1, MPI_INT, 3, 30, MPI_COMM_WORLD);
break;
case 2:
y = 20;
MPI_Send(&y, 1, MPI_INT, 0, 20, MPI_COMM_WORLD);
break;
case 3:
MPI_Recv(&y, 1, MPI_INT, 1, 30, MPI_COMM_WORLD, &status);
break;
}
```

a. [9 points] What will be the final values of x and y for each process after the code executes?

	P0	P1	P2	P3
x				
y				

b. [5] Explain the concept of deadlock in MPI. Is there a possibility of deadlock in this code? If yes, explain why. If not, why not?

c. [5] What would happen if process 2 sends to process 1 instead of process 0?

d. [5] What would happen if we removed the `MPI_COMM_WORLD` argument from all MPI calls?

Problem 4

a. [10] Describe the concept of a race condition in the context of multithreaded programming. Provide a concrete example of a code snippet that demonstrates a race condition, and explain how to resolve it using mutexes.

b. [10] Explain the difference between threads and processes in terms of memory space, context switching overhead, and inter-process communication.

Problem 5

a. [10] Describe the difference between strong and weak scaling in parallel computing. Give examples of applications where each type of scaling is more relevant.

b. [10] Explain the concept of load balancing in parallel computing. Discuss techniques for achieving load balancing, including static and dynamic load balancing.

Problem 6

a. [12] Compare and contrast OpenMP and MPI. Discuss their strengths and weaknesses, and what types of parallel programming problems each is best suited for.

b. [8] Briefly describe the role of a compiler in generating parallel code from sequential code in the context of OpenMP.

Problem 7

Consider the following pseudocode:

```
for i = 1 to n do
  a[i] = a[i] + b[i]
end for
```

a. [10] Show how you would parallelize this code using OpenMP.

b. [10] Show how you would parallelize this code using MPI. Assume that the arrays a and b are distributed across the MPI processes.

Problem 8

a. [10] Explain the importance of profiling and debugging tools in the development of efficient parallel programs. Name at least three tools commonly used for profiling and debugging parallel code and briefly describe their capabilities.

b. [10] What are some common sources of performance bottlenecks in parallel programs? Discuss strategies for identifying and addressing these bottlenecks.