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 (2pm EST).
- You have up to 24 hours to submit on Brightspace (i.e. till March 15th 2pm 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 in 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, E-mail, 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 different processes, be executed on a shared memory machine and get the same performance as when executed on a multicore with the same number of cores? If yes explain how, in 1-2 lines. If not, explain why not.

d. [6] If we have an eight-way superscalar core, how many register files might we need to get the best performance? Justify.

Problem 2

Suppose we have the following DAG that represents different tasks and their dependencies:

(Assume a DAG where task A depends on nothing, B and C depend on A, D depends on B, E depends on C and D, F depends on E)

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 core of type Y. Each CPU type is optimized for some type of operations. You can ignore communication overhead among tasks.

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

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 were allowed to add one dependency arrow to the above DAG, which one would you add? And why?

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:

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

a. [10 points] What will be the values of x and y for each of the 4 processes after executing the above code?

	P0	P1	P2	P3
x				
y				

b. [5] Is it guaranteed that the communication among the 4 processes will complete in the order specified by the code? Explain.

c. [5] What will happen if we execute the above code with: `mpiexec -n 2`

d. [5] What will happen if process 1 fails before it sends the message with tag 3?

Problem 4

a. [8] Explain the difference between Amdahl's Law and Gustafson's Law in the context of parallel program scalability.

b. [7] Describe a scenario where using OpenMP might be preferable to MPI, and vice-versa.

Problem 5

a. [10] What are the main differences between threads and processes in terms of memory management and scheduling?

b. [10] Explain the concept of race conditions in concurrent programming and give an example.

Problem 6

Consider a parallel algorithm for matrix multiplication. The matrices are of size N x N.

a. [10] Describe a simple parallel algorithm using a 2D decomposition to perform this multiplication.

b. [5] Analyze its computational complexity and scalability.

Problem 7

A. [10] Which of the following statements are true about cache coherence protocols? (Select all that apply)

1. Write-invalidate protocols are generally simpler to implement than write-update protocols.
2. Write-update protocols can lead to more write traffic than write-invalidate protocols.
3. Snooping cache coherence protocols require special hardware support.
4. Directory-based cache coherence protocols scale better to large numbers of processors.

B. [5] Briefly explain the concept of false sharing in shared-memory programming. How can it be mitigated?