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 in the rectangle at the bottom of this page.
    - 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)

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? If yes explain how, in 1-2 lines. If not, explain why not.
d. [6] If we have a two-way superscalar core, how many fetch units do we need to get the best performance? Justify.

Problem 2

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

(Imagine a DAG here with nodes A, B, C, D, E, F, G, H. A has no dependencies. B depends on A. C depends on A. D depends on B. E depends on B and C. F depends on C. G depends on E and F. H depends on G.)

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. That is, one type of CPU is not always faster than the other type for all tasks. You can ignore communication overhead among tasks.

Task	CPU type X	CPU type Y
A	2	3
B	4	2
C	6	8
D	8	6
E	10	9
F	3	5
G	7	4
H	1	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] Using better algorithm/programming, we can enhance the DAG a bit by removing an arrow. Removing an arrow means less dependency and potential better performance. If you were allowed to remove one arrow from the above DAG, which one would you remove? 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 (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(&x, 1, MPI_INT, 3, 4, MPI_COMM_WORLD);
break;
case 3:
x=10; y=11; z=12;
MPI_Recv(&x, 1, MPI_INT, 2, 4, MPI_COMM_WORLD, &status);
MPI_Send(&z, 1, MPI_INT, 0, 3, MPI_COMM_WORLD);
```

```
break;
}
```

a. [9 points] What will be the values of x, y, and z for each of the 4 processes
after executing the above code?

	P0	P1	P2	P3
x				
y				
z				

b. [5] Is there a possibility that the communication among the 4 processes
executes out of order? If yes, explain the reason. If not, why not?
c. [5] What will happen if we execute the above code with: mpiexec -n 2
d. [5] What will happen if we remove the 'break' statement from 'case 0'?

Problem 4

a. [5] If we have an application with high communication overhead between
tasks. Does it have good scalability (i.e. as we keep increasing the number of
cores, do we see speedup)? Assume the problem size is big enough.
b. [5] If we have four threads that all have the same type and number of
computations. And we assign each thread to a core. Does this necessarily mean
we have perfect load balance? Explain.

Problem 5

Describe the differences between OpenMP and MPI in terms of: a. [5] Pro-
gramming model b. [5] Memory model c. [5] Scalability d. [5] Ease of use

Problem 6

Consider a parallel algorithm that sorts an array of n elements using p proces-
sors. Assume the algorithm uses a simple parallel merge sort. a. [10] Describe
the steps involved in the parallel merge sort algorithm. b. [10] Analyze the time
complexity of this parallel algorithm in terms of n and p. Consider both the
work and span.

Problem 7

a. [10] Explain the concept of Amdahl's Law and its implications for parallel
computing. Give an example. b. [10] Explain the concept of Gustafson's Law
and how it differs from Amdahl's Law. Give an example.