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] Explain the concept of Amdahl's Law in the context of parallel computing. Give a specific example illustrating its implications for achievable speedup.

b. [10] What are the main differences between OpenMP and MPI? Discuss their strengths and weaknesses in different parallel programming scenarios.

c. [10] Describe the concept of false sharing in shared-memory programming. Explain how it can degrade performance and suggest a technique to mitigate it.

d. [6] How does the number of cores affect the performance of a program with significant I/O operations? Justify your answer.

Problem 2

Suppose we have the following DAG that represents different tasks and their dependencies. The nodes represent tasks and the edges represent dependencies.

(Insert a DAG here. Example: A -¿ B, A -¿ C, B -¿ D, C -¿ D, D -¿ E)

The following table shows the execution time of each task on two different types of CPUs:

Task	CPU type A	CPU type B
A	2	3
B	5	2
C	3	4
D	4	6
E	1	2

a. [10] What is the minimum execution time if we use only CPU type A? Show your task scheduling.

b. [10] What is the minimum execution time if we use only CPU type B? Show your task scheduling.

c. [10] If we have one CPU of type A and one CPU of type B, what is the minimum execution time? Show your task scheduling and which CPU executes which task.

Problem 3

Consider the following MPI code snippet:

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int data = rank * 10;
    int recv_data;

    if (rank == 0) {
        for (int i = 1; i < size; i++) {
            MPI_Recv(&recv_data, 1, MPI_INT, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            printf("Process 0 received %d from process %d\n", recv_data, i);
        }
    } else {
        MPI_Send(&data, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return 0;
}
```

a. [9 points] What will be the output of this code if it's run with 'mpiexec -n 4'?

b. [5] Explain the purpose of 'MPI$_STATUS_IGNORE$'$in the$'MPI_Recv'$function.

c. [5] How would you modify the code to make it send and receive data in a ring topology (process 0 sends to 1, 1 sends to 2, ..., n-1 sends to 0)?

d. [5] What happens if you run this code with only one process?

Problem 4

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

b. [10] Explain the concept of load balancing in parallel programs. Discuss different techniques used to achieve good load balancing.

Problem 5

a. [12] Explain the concept of a critical section in parallel programming and the problems that can arise if it is not properly handled. Provide an example illustrating potential problems.

b. [8] Describe the role of semaphores in managing concurrent access to shared resources.

Problem 6

a. [10] Compare and contrast the performance characteristics of shared-memory and distributed-memory parallel systems. Discuss the tradeoffs involved in choosing between them.

b. [10] What are the key challenges in debugging parallel programs? Describe techniques that can be used to facilitate debugging.

Problem 7

Consider a parallel program that sorts an array of numbers. The array is divided into chunks, and each process sorts its chunk independently using quicksort. After the independent sorts, a merge step is needed to combine the sorted chunks.

a. [10] Describe a parallel merge algorithm to combine the sorted chunks efficiently.

b. [10] Analyze the scalability of this parallel sorting algorithm (considering both the sorting and merging phases). Discuss factors that might limit its scalability.