CSCI-UA.0480-051: Parallel Computing
Midterm Exam (Mar 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 eight main problems, each one must be answered in 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, 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

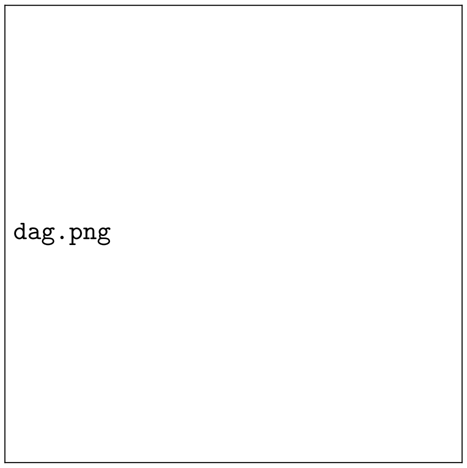
a. [10] Explain the difference between Amdahl's Law and Gustafson's Law in the context of parallel processing.

b. [10] Describe a scenario where a parallel algorithm might be slower than a sequential algorithm.

c. [10] What are the key challenges in designing and implementing parallel programs?

Problem 2

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



The following table shows the execution time of each task on different processors.

Task	Processor A	Processor B	Processor C
A	5	7	3
B	2	4	6
C	8	1	9
D	3	2	5
E	4	6	2

a. [10] Determine the minimum execution time using only Processor A.

b. [10] Determine the minimum execution time using only Processor B.

c. [10] Develop a schedule to minimize execution time using all three processors. What is the minimum execution time achievable with this schedule?

Problem 3

a. [10] Explain the concept of race conditions in parallel programming and provide an example.

b. [10] Describe different methods for synchronizing threads in a shared-memory environment.

Problem 4

Describe the differences between OpenMP and MPI, including their strengths and weaknesses. [20 points]

Problem 5

Suppose that MPI_COMM_WORLD consists of four processes (0, 1, 2, 3). The following code is executed after MPI initialization:

```
int my_rank, x, y;
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
if (my_rank == 0) {
  x = 10;
  MPI_Send(&x, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
} else if (my_rank == 1) {
  MPI_Recv(&x, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  y = x * 2;
  MPI_Send(&y, 1, MPI_INT, 2, 1, MPI_COMM_WORLD);
} else if (my_rank == 2) {
  MPI_Recv(&y, 1, MPI_INT, 1, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  x = y + 5;
  MPI_Send(&x, 1, MPI_INT, 3, 2, MPI_COMM_WORLD);
} else {
  MPI_Recv(&x, 1, MPI_INT, 2, 2, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  printf("Final value of x: %d\n", x);
}
MPI_Finalize();
```

a. [10] What will be the final value of x printed by process 3?

b. [5] What is the purpose of the tag in MPI_Send and MPI_Recv?

Problem 6

Explain the concept of load balancing in parallel computing and provide examples of techniques used to achieve it. [15 points]

Problem 7

Compare and contrast the following parallel programming models: data parallelism and task parallelism. Provide examples of applications suitable for each model. [15 points]

Problem 8

a. [5] What is a deadlock in parallel programming? Give an example scenario.

b. [5] Explain the concept of false sharing in a shared memory system and how it can affect performance.