Parallel Computing Practice Exam

Instructions: Answer all questions to the best of your ability. Time allotted: 60 minutes.

Section 1: Multiple Choice (1 point each)

1. Which of the following is NOT a primary challenge in parallel computing?

a) Load balancing

b) Data dependency

c) Increased sequential execution speed

d) Communication overhead

Answer: _____

Section 2: Short Answer (3 points each)

2. Briefly explain the difference between shared memory and distributed memory parallel computing architectures. Include at least one advantage and one disadvantage of each.

Answer:

3. Describe the concept of Amdahl's Law and its implications for the scalability of parallel programs.

Answer:

Section 3: Problem Solving (5 points each)

4. Consider a program that takes 100 seconds to run sequentially. We parallelize it, resulting in 80% of the program being parallelizable. Assuming we have 4 processors, what is the theoretical speedup according to Amdahl's Law? Show your calculations.

Answer:

5. You are tasked with designing a parallel algorithm to sort a large array of numbers. Describe a suitable parallel sorting algorithm (e.g., merge sort, quicksort) and briefly explain how it would be implemented in a shared-memory environment. Consider how you would manage data partitioning and synchronization.

Answer:

Answer Key (For Instructor Use Only):

1. c) Increased sequential execution speed

2. Shared Memory: Advantages - Ease of data sharing, Disadvantages - Scalability limitations, potential for race conditions. Distributed Memory: Advantages - High scalability, Disadvantages - Complex data communication, higher communication overhead.

3. Amdahl's Law states that the overall speedup of a program is limited by the portion of the program that cannot be parallelized. It highlights the diminishing returns of adding more processors if a significant portion of the code remains sequential.

4. Speedup = 1 / [(1 - parallelizable portion) + (parallelizable portion / number of processors)] = 1 / [(1 - 0.8) + (0.8 / 4)] = 1 / (0.2 + 0.2) = 1 / 0.4 = 2.5

5. A suitable algorithm would be parallel merge sort. The array would be divided into sub-arrays for each processor. Each processor would sort its sub-array sequentially. Then, a merge step would be performed in parallel, merging the sorted sub-arrays. Synchronization would be needed during the merge step to ensure data consistency. Critical sections or mutexes could be used to protect shared data during the merge.