Parallel Computing Practice Exam - Debugging Focus

Instructions: Answer all questions to the best of your ability. This exam focuses on debugging parallel programs.

Section 1: Multiple Choice (2 points each)

1. Which of the following is NOT a common source of errors in parallel programs?

a) Race conditions

b) Deadlocks

c) Compiler optimizations

d) Data dependencies

Answer: _____

2. What is a common tool used for debugging parallel programs that allows visualizing the execution flow of multiple threads?

a) gdb (GNU Debugger) only

b) Valgrind only

c) A debugger with threading support (e.g., gdb with pthreads support, TotalView)

d) A profiler only

Answer: _____

3. A race condition occurs when:

a) Multiple threads access and modify the same data without proper synchronization.

b) A thread gets stuck waiting for a resource that another thread holds.

c) A program terminates unexpectedly.

d) The compiler optimizes code in an unexpected way.

Answer: _____

4. Which debugging strategy is best suited to detect data races?

a) Using print statements for variable values

b) Running the program multiple times and observing different outputs

c) Using a race detector tool (e.g., ThreadSanitizer)

d) Examining the compiler output

Answer: _____

Section 2: Short Answer (3 points each)

5. Briefly explain the concept of a deadlock in the context of parallel programming. How might you detect it during debugging?

Answer:

6. Describe a scenario where false sharing could lead to performance degradation in a parallel program. How could this be mitigated?

Answer:

Section 3: Problem Solving (5 points each)

7. Consider the following pseudocode for a parallel program that sums elements of an array:

```
// Assume 'array' is a shared array of size N
```

```
// 'partialSums' is a shared array of size numThreads
for (int i = 0; i < numThreads; i++) {
partialSums[i] = 0;
}
#pragma omp parallel for
for (int i = 0; i < N; i++) {
partialSums[i % numThreads] += array[i];
}
int totalSum = 0;
for (int i = 0; i < numThreads; i++) {
totalSum += partialSums[i];
}
```

This code has a potential race condition. Identify the race condition and explain how to fix it using appropriate synchronization primitives.

Answer:

8. You are debugging a parallel program that consistently crashes with a segmentation fault. What are three possible causes of this error in a parallel context, and what debugging techniques would you employ to isolate the problem?

Answer:

9. You have a parallel program using threads that seems to be producing incorrect results intermittently. What are two debugging strategies you would employ to track down the source of the error?

Answer:

10. Explain how you would use a debugger to step through a parallel program, focusing on how you would manage the execution of multiple threads. What commands or features in a typical debugger would you use?

Answer:

Answer Key (For Instructor Use Only): This section will contain the correct answers for grading. It would be included separately from the student exam version.