1. Which of the following is NOT a commonly used method for program profiling in OpenMP programming?

a) Instrumentation-based profiling
b) Sampling-based profiling
c) Performance counters
d) Static code analysis

Answer: d) Static code analysis

Explanation: Program profiling in OpenMP typically involves instrumentation-based profiling, sampling-based profiling, and performance counters to measure various aspects of program execution such as runtime, memory usage, and thread behavior. Static code analysis, while useful for detecting certain types of issues, is not primarily used for profiling purposes.

2. What is a common performance pitfall in OpenMP programming?

a) Overutilizing synchronization
b) Underutilizing parallelism
c) Ignoring memory allocation
d) Neglecting loop unrolling

Answer: a) Overutilizing synchronization

Explanation: Overusing synchronization mechanisms like locks and barriers can lead to performance degradation due to excessive thread contention and overhead. Effective OpenMP programming involves minimizing unnecessary synchronization to achieve optimal performance.

3. How can the impact of OpenMP work-sharing constructs be improved?

a) By increasing the number of threads
b) By reducing the loop iteration count
c) By balancing workload among threads
d) By increasing the loop iteration count

Answer: c) By balancing workload among threads

Explanation: Balancing workload among threads ensures that each thread has a roughly equal amount of work to perform, maximizing efficiency and preventing idle threads. This can be achieved through techniques like loop scheduling and workload partitioning.

4. What is a potential overhead associated with short loops in OpenMP?

a) Memory fragmentation
b) Thread creation overhead
c) Task spawning overhead
d) Synchronization overhead

Answer: b) Thread creation overhead

Explanation: In short loops, the overhead of creating and managing threads may outweigh the benefits of parallel execution. This overhead includes thread creation, initialization, and termination, which can impact performance negatively.

5. What is serialization in the context of OpenMP programming?

a) Arranging data in a sequential order
b) Ensuring only one thread executes a critical section at a time
c) Converting parallel code into serial code
d) Optimizing loops for sequential execution

Answer: b) Ensuring only one thread executes a critical section at a time

Explanation: Serialization in OpenMP refers to the process of ensuring that only one thread can execute a critical section of code at a time, preventing data races and maintaining program correctness in parallel execution.

6. What is false sharing in OpenMP programming?

a) Sharing of incorrect data between threads
b) Sharing of data without proper synchronization
c) Overlapping memory regions causing unintended data access
d) Sharing of cache lines between threads leading to unnecessary synchronization

Answer: d) Sharing of cache lines between threads leading to unnecessary synchronization

Explanation: False sharing occurs when multiple threads inadvertently share the same cache lines, even though they are working on different data. This can lead to unnecessary cache coherence traffic and synchronization overhead, impacting performance.

7. Which of the following is NOT a recommended approach for reducing false sharing in OpenMP?

a) Increasing thread count

b) Padding data structures
c) Reordering data layout
d) Using thread-local storage

Answer: a) Increasing thread count

Explanation: Increasing the thread count does not directly address false sharing issues. Instead, techniques like padding data structures, reordering data layout, and using thread-local storage are recommended for mitigating false sharing overhead in OpenMP.

8. What does the term “work sharing” refer to in OpenMP?

a) Dividing work among threads
b) Sharing data among threads
c) Distributing memory access
d) Coordinating thread creation

Answer: a) Dividing work among threads

Explanation: Work sharing in OpenMP involves distributing computational tasks or loop iterations among multiple threads to achieve parallel execution and utilize available processor resources efficiently.

9. Which OpenMP directive is commonly used to identify critical sections of code?

a) #pragma omp atomic
b) #pragma omp parallel
c) #pragma omp critical

d) #pragma omp barrier

Answer: c) #pragma omp critical

Explanation: The #pragma omp critical directive is used to specify critical sections of code where only one thread is allowed to execute at a time, preventing data races and ensuring thread safety in parallel execution.

10. What is the purpose of loop scheduling in OpenMP?

a) Distributing loop iterations among threads
b) Controlling access to loop variables
c) Enforcing loop termination conditions
d) Specifying loop parallelism level

Answer: a) Distributing loop iterations among threads

Explanation: Loop scheduling in OpenMP determines how loop iterations are distributed among threads, ensuring workload balance and efficient utilization of parallel resources by assigning iterations dynamically or statically to threads.

Related posts:

1. Introduction to Information Security
2. Introduction to Information Security MCQ
3. Introduction to Information Security MCQ
4. Symmetric Key Cryptography MCQ
5. Asymmetric Key Cryptography MCQ

6. Authentication & Integrity MCQ
7. E-mail, IP and Web Security MCQ