CSE 3033 PROJECT #3

Group Members:

150119711 Emre Cihan Varlı

150119567 Fatih Erkam Dinler

150119767 Furkan Bozkurt

Answers:

1. Method 1 (No Synchronization): No explicit synchronization mechanism is used. There is a potential for race conditions since multiple threads are updating the global variable global\_sqrt\_sum concurrently. The results may not be accurate due to lack of synchronization

Method 2 (Mutex Synchronization): A mutex is used to synchronize access to the critical section (global\_sqrt\_sum). Mutex ensures that only one thread can access the critical section at a time. This method is likely to provide correct results as it prevents race conditions.

Method (Mutex Synchronization and Alternative Implementation): Similar to Method 2, mutex is used for synchronization. However, the mutex is locked and unlocked in the same if block, making it equivalent to Method 2 in practice. This method is also likely to provide correct results.

1. We tested the 2nd and 3rd methods, believing that they provided accurate results. After running both methods three times and obtaining the results, we calculated the average. Based on these averages, Method 3 had a duration of 29,120, while Method 2 had a duration of 29,944. According to these results, we can conclude that Method 3 operates faster than Method 2.
2. The impact of increasing threads on total time is not guaranteed to be positive. It depends on factors such as the parallelizability of the computation, thread management overhead, and the number of available CPU cores. If a computation is easily parallelizable and there are sufficient CPU cores, increasing threads can reduce total time. However, factors like overhead and contention may counteract the benefits. Check the number of CPU cores using the ‘lscpu’ command. If threads exceed cores, contention may increase, affecting performance.
3. With more threads, user time may increase due to parallelization, but diminishing returns are possible. System time can rise due to kernel-level operations like synchronization, contention and context switching.

Possible Causes of Differences:

Contention: If threads contend for resources (e.g., shared variables, mutexes), the system time may increase due to the need for kernel-level coordination.

Synchronization Overhead: Methods 2 and 3 involve synchronization using a mutex. As the number of threads increases, contention for the mutex may lead to increased system time.

Context Switching Overhead: More threads may lead to increased context switching between threads, contributing to system time.

Cache Effects: Increasing the number of threads may introduce cache effects and cache contention, impacting overall performance.

Run Results:

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu