**Report: Tradeoffs of Concurrency vs. Single-Threaded Programs**

**GitHub link:** [**https://github.com/CCT-Dublin/ca1-Pruthvi1-123.git**](https://github.com/CCT-Dublin/ca1-Pruthvi1-123.git)

**1. Tradeoffs of using concurrency for sorting a list of lists of integers:**

**Advantages:**

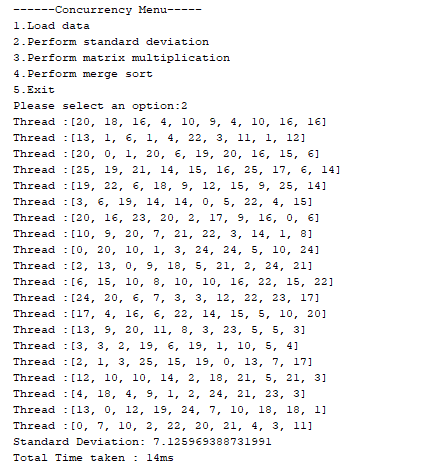
* Performance Improvement: Concurrency can improve performance by allowing the sorting tasks to be executed simultaneously on multiple threads. This can be especially helpful for large datasets, as the sorting tasks can be completed in parallel.
* Reduced Execution Time: Splitting the sorting process into smaller tasks and executing them concurrently can reduce the overall execution time compared to a single-threaded approach, where tasks are executed sequentially.
* Scalability: Concurrency can help a program scale to take advantage of multi-core processors. As the number of processor cores increases, the sorting tasks can be further parallelized, leading to improved scalability.

**Disadvantages:**

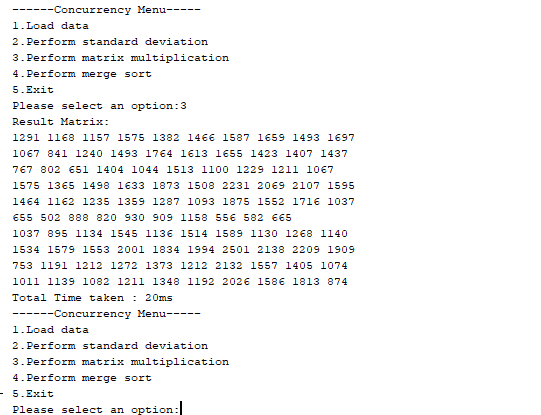
* Complexity and Overhead: Implementing concurrency introduces additional complexity and overhead, such as managing threads, synchronization, and potential race conditions. This complexity can make the code harder to understand, debug, and maintain.
* Resource Consumption: Concurrency can consume more system resources, such as CPU and memory, due to the overhead of managing multiple threads. This may lead to resource contention and increased system overhead, particularly on systems with limited resources.
* Potential for Deadlocks and Race Conditions: Concurrent execution introduces the risk of synchronization issues such as deadlocks and race conditions, where multiple threads access shared data concurrently, leading to unpredictable behavior and program crashes.

Execution Times :

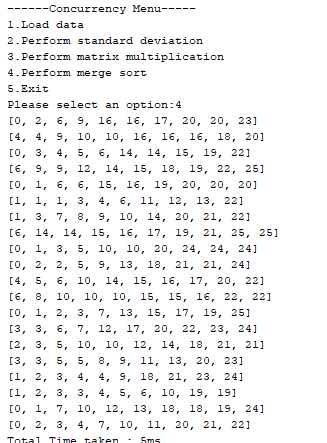
1. Standard deviation :

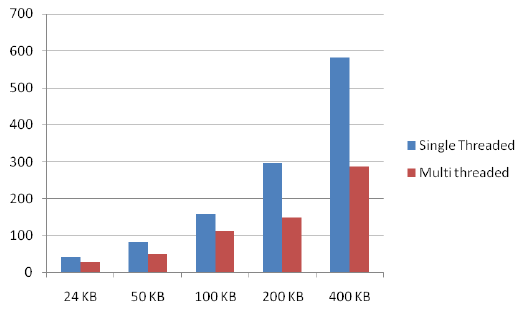


1. Matrix Multiplication :



1. Merge Sort :





**2. Tradeoffs of using concurrency for reading data from a CSV file:**

**Advantages:**

* Improved Throughput: Concurrency can improve throughput by allowing multiple threads to read data from the CSV file simultaneously. This can be helpful for large files or slow I/O operations, as it can lead to faster data retrieval and processing.
* Reduced I/O Wait Time: By overlapping I/O operations with computation, concurrency can help reduce the overall wait time for reading data from the CSV file. While one thread is waiting for I/O, other threads can perform useful computation, increasing overall efficiency.
* Asynchronous Processing: Concurrency enables asynchronous processing of data, where threads can start processing data as soon as it becomes available, without waiting for the entire file to be read. This can improve responsiveness and overall system performance.

**Disadvantages:**

* Complexity and Synchronization: Reading data concurrently from a CSV file requires proper synchronization mechanisms to ensure data consistency and avoid race conditions. Managing concurrent access to shared resources, such as the file pointer or buffers, adds complexity to the code and increases the risk of synchronization errors.
* Potential for I/O Contention: Concurrent access to the same file may lead to I/O contention, where multiple threads compete for access to the file, potentially degrading performance due to increased overhead and resource contention.
* File Fragmentation: Concurrent reading of a file can lead to file fragmentation, where different parts of the file are accessed by different threads simultaneously. This can result in suboptimal disk access patterns and reduced I/O performance, particularly for spinning disk drives.

**We can conclude that concurrency can offer significant advantages for improving performance and scalability. However, it is important to be aware of the potential tradeoffs, such as complexity, synchronization challenges, and potential resource contention, before deciding to use concurrency. The decision of whether or not to use concurrency should be based on the specific requirements of the application, the available hardware resources, and the tradeoffs between performance and complexity.**