High-Performance Concurrent Sorted Vector Analysis

ByeongKyu Park

March 17, 2024

1 Introduction

This paper details the development and performance analysis of a high-performance concurrent sorted vector, focusing on its ability to efficiently handle concurrent write and read operations. The implementation leverages modern C++ features to achieve significant performance improvements, particularly in real-time data processing applications.

2 Performance Insights

2.1 Quick Sort Performance

The parallel Quick Sort algorithm demonstrates substantial speed improvements over the traditional ‘std::sort‘ method. The performance evaluation was conducted on a system with 16 logical cores, sorting an array of 100 ‘Ratio‘ objects.

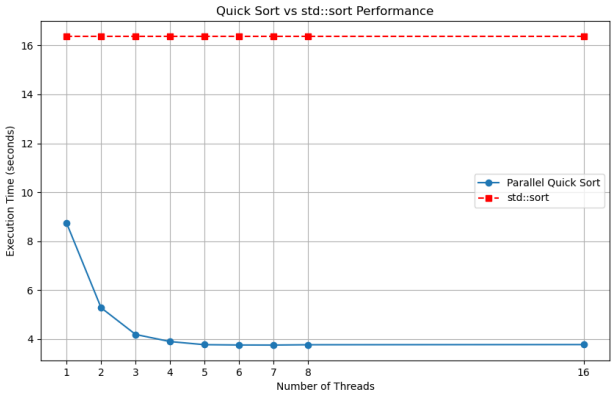

Figure 1: Comparison of Parallel Quick Sort and std::sort execution times.

2.2 Concurrent Read/Write Performance

The concurrent read/write performance was tested under varying numbers of writing threads while continuously reading from the container, showcasing the sorted vector's capability to manage high-volume concurrent modifications efficiently.

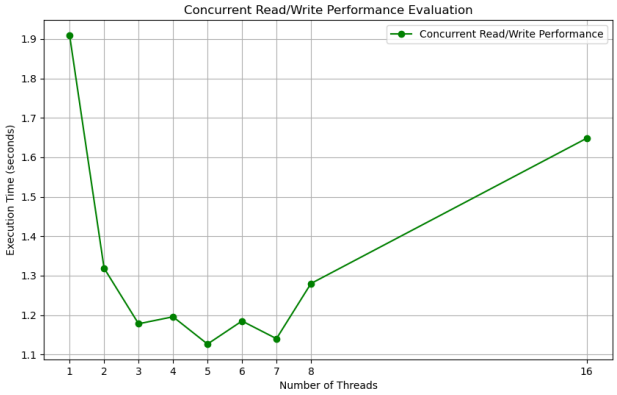

Figure 2: Execution times for concurrent read/write operations with varying numbers of writer threads.

3 Implementation Highlights

The implementation employs a sophisticated memory management system, thread safety measures through atomic operations, and a "bag of tasks" approach for the parallel Quick Sort algorithm to distribute work among threads efficiently.

4 The Role of the Ratio Class

The 'Ratio' class, with built-in delay for comparison operations, serves a crucial role in evaluating the time complexity and performance under simulated computational load. This approach ensures the benchmarks accurately reflect the vector's capabilities in handling complex operations.

5 Conclusion

The developed concurrent sorted vector represents a significant advancement in concurrent computing, offering a solution that not only excels in performance but also in efficiency and reliability for applications requiring real-time data processing.