Analysis of Lock-Free Scalable Vector Performance in High-Concurrency Environments

Your Name

February 24, 2024

1 Introduction

This document presents an analysis of a Lock-Free Scalable Vector (LFSV) designed for high-concurrency environments. The LFSV utilizes a MemoryBank for efficient vector instance recycling and a GarbageRemover for safe, delayed deletion of vectors, aiming to minimize traditional locking mechanisms' performance bottlenecks.

2 Implementation

The LFSV is implemented with two primary components to manage memory in a concurrent setting efficiently:

- **MemoryBank:** Recycles vector instances to reduce allocation and deallocation overhead.
- **GarbageRemover:** Safely deletes vectors after a delay, preventing use-after-free errors.

Together, these components allow for safe updates and access by multiple threads without serializing access, leveraging atomic operations for consistency.

3 Performance Evaluation

Performance was evaluated under different thread counts, with the following results observed:

4 Analysis

The results indicate improved performance with an increase in threads up to 4, after which performance plateaus and then degrades. This pattern suggests that the LFSV effectively utilizes CPU resources up to a point, beyond which the

Number of Threads	Operations per Thread	Execution Time (seconds)
1	25600	0.757726
2	12800	0.716873
4	6400	0.664578
8	3200	0.701546
16	1600	0.82039

Table 1: LFSV Performance Results

overhead associated with managing additional threads outweighs the benefits of parallelism. Factors such as lock contention, context switching, and cache effects likely contribute to this behavior.

Given the hardware configuration of 8 cores and 16 logical processors with enabled virtualization, optimal performance was expected near the core count. However, diminishing returns beyond 4 threads suggest a complex interplay between the LFSV's lock-free design and the underlying hardware's capabilities to manage concurrency efficiently.

5 Conclusion

The LFSV demonstrates the potential for lock-free data structures to improve performance in concurrent applications. Future work will focus on optimizing memory management components and further analyzing the impact of hardware characteristics on scalability and performance.