Simplified MCAS by Improving Memory Reclamation

Syed All Rahi Zarif, Bin Guo

Department of Computer Science, Trent University, Ontario, Canada
syedallrahizarif@trentu.ca, binguo@trentu.ca

1 Introduction

Compare-and-swap (CAS) is a widely used atomic primitive in concurrent programming. It works on a single memory word and writes a new value only when the current value matches the expected one. Multi-word compare-and-swap (MCAS) is an extension of CAS that atomically updates several memory locations. It guarantees that either all updates take place or none do. This is useful for building complex concurrent data structures. For example, MCAS simplifies the design of lock-free structures such as linked lists or Union-Find, where multiple pointers or values must be updated consistently. Without MCAS, programmers rely on complicated techniques or transactional memory, which increases both design complexity and execution cost.

Much work has been done in this area. The *CASN* algorithm of Harris et al. [2] and the volatile *PMwCAS* algorithm of Wang et al. [4] both require about $3k + 1$ CAS operations for a k-word update. They also depend on descriptors and garbage collection (GC), which adds extra cost. The *AOPT* algorithm of Guerraoui et al. [1] improved this by showing that only $k + 1$ CAS are needed in the uncontended case. *AOPT* achieved this by delaying the memory cleanup process. One major problem that remains inadequately solved is memory cleanup. Current approaches in *AOPT* delay cleanup and rely on epoch-based or lazy reclamation. This increases code complexity and slows down reads, because memory locations may continue to point to descriptors for some time. Sugiura and Ishikawa [3] showed that garbage collection itself is a major bottleneck. They built a GC-free design that gave higher throughput. However, they used spinlocks, and the algorithm is no longer lock-free.

In this work, we focus on this memory management problem in the MCAS operations. We propose a memory cleanup mechanism that can automatically reclaim the memory when the MCAS operations are finished, which prepares for the next round of MCAS operations. Our method ensures the use of $k + 1$ CAS operations, and provides efficient memory cleanup. Furthermore, we will apply our approach to many lock-free data structures, e.g., double-linked list, union-find, which achieve high performance. In addition, we will build open-source libraries for the MCAS.

2 Contributions

In this work, our contributions are summarized as follows.

1. We plan to design an optimized memory management mechanism that can efficiently release and recycle memory, thereby reducing the system overhead.
2. We plan to develop a user-friendly open source library to apply the improved method to existing data structures and algorithms.
3. We plan to extend the improved MCAS to more complex data structures and study its performance. As a case study, we aim to apply it to the Union-Find data structure, which is fundamental for many parallel graph algorithms, e.g., parallel SCC decomposition and Minimum Spanning Tree.
4. We plan to conduct experiments and evaluate whether the improved MCAS can achieve better performance compared to existing approaches.

References

[1] Rachid Guerraoui, Alex Kogan, Virendra J. Marathe, and Igor Zablotchi. Efficient multi-word compare-and-swap. In *Proceedings of the 34th International Symposium on Distributed Computing (DISC)*, pages 4:1–4:19, 2020. https://doi.org/10.4230/LIPIcs.DISC.2020.4 doi:10.4230/LIPIcs.DISC.2020.4.

[2] Timothy L. Harris, Keir Fraser, and Ian A. Pratt. A practical multi-word compare-and-swap operation. In *Proceedings of the 16th International Conference on Distributed Computing (DISC)*, pages 265–279, 2002.

[3] Kento Sugiura and Yoshiharu Ishikawa. Implementation of a multi-word compare-and-swap operation without garbage collection. *IEICE Transactions on Information and Systems*, E105-D(5):946–954, 2022. https://doi.org/10.1587/transinf.2021DAP0011 doi:10.1587/transinf.2021DAP0011.

[4] Tianzheng Wang, Justin Levandoski, and Per-Åke Larson. Easy lock-free indexing in non-volatile memory. In *Proceedings of the 34th IEEE International Conference on Data Engineering (ICDE)*, pages 461–472, 2018. https://doi.org/10.1109/ICDE.2018.00049 doi:10.1109/ICDE.2018.00049.