Persistent Memory k-Core Decomposition

Igor Jardim-Martins, Bin Guo

Department of Computing & Information Systems, Trent University, Ontario, Canada
ijardimmartins@trentu.ca; binguo@trentu.ca

Abstract

The k-core, as one of the most well-studied cohesive subgraph models, is widely used to identify graph nodes that are crucial in various applications, including biological, social, ecological, and financial networks. Existing in-memory methods do not scale due to limited DRAM capacity, disk-based methods suffer from I/O bottlenecks, and distributed methods have high communication and synchronization overheads. Persistent memory (PM) is a cost-effective, high-capacity, and non-volatile memory with speeds that are only $2-3$ times slower than DRAM. However, PM suffers from slow in-place writes and read/write amplification. We propose a PM-based k-core decomposition that utilizes DRAM as a buffer to reduce the number of PM accesses. We conduct extensive experiments to compare with existing k-core decomposition methods.

1 Introduction

The k-core is the maximal subgraph in which each vertex has a degree of at least k; and the core number of each vertex is the maximum value of k contained in the k-core. The k-core decomposition involves computing the core numbers of all vertices in the graph. The k-core has many applications, such as identifying critical users within a network and detecting influential spreaders in social networks . Several approaches exist for k-core decomposition, including in-memory, disk-based, and distributed methods. In-memory approaches cannot process large data graphs due to the limited DRAM capacity in a single machine [3]. Disk-based approaches suffer from an I/O performance bottleneck. Distributed approaches have a high cost on communication and synchronization, especially with vertices with high core numbers [1].

Persistent memory (PM) is a developing technology for non-volatile storage that is only $2-3$ times slower than DRAM [4]. This technology is expected to improve the performance of large-scale data graph processing and reduce the DRAM usage, thereby solving the problem of existing methods. However, PM experiences issues with in-place writes being $7-8$ times slower than sequential or random writes [2], as well as read/write amplification with small accesses.

Many graph algorithms, such as k-core decomposition, frequently perform small-sized in-place writes, which can deteriorate their performance on PM and reduce the hardware's lifespan more quickly. In this work, we are the first to propose the PM-based k-core decomposition. The whole graph is stored on PM, and the k-core decomposition is performed on PM. DRAM is used as a buffer to reduce the number of in-place write operations to achieve high performance. Furthermore, the methodology can be applied to other graph algorithms, such as k-truss decomposition, SCC decomposition, and Tarjan's algorithm.

2 Contributions

The contributions are summarized as follows.

- We will explore a PM-based k-core decomposition algorithm, which can handle large graphs on a single machine. We propose using DRAM as a buffer to reduce the total reads/writes on PM and to improve performance. Additionally, we develop several buffer strategies to support efficient buffer usage.
- We will perform comprehensive experimental analysis comparing our PM-based approach with state-of-the-art in-memory, disk-based, and distributed methods.

References

[1] Bin Guo and Runze Zhao. Experimental evaluation of distributed k-core decomposition. *arXiv preprint arXiv:2406.17580*, 2024.

[2] Abdullah Al Raqibul Islam and Dong Dai. Dgap: Efficient dynamic graph analysis on persistent memory. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, 2023.

[3] Rui Wang, Shuibing He, Weixu Zong, Yongkun Li, and Yinlong Xu. Xpgraph: Xpline-friendly persistent memory graph stores for large-scale evolving graphs. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1308–1325. IEEE, 2022.

[4] Jian Yang, Juno Kim, Morteza Hoseinzadeh, Joseph Izraelevitz, and Steve Swanson. An empirical guide to the behavior and use of scalable persistent memory. In *18th USENIX Conference on File and Storage Technologies (FAST 20)*, pages 169–182, 2020.