A Write Efficient PCM-Aware Sort


School of Computing, National University of Singapore
meduri@cwi.nl, {suzhan,tankl}@comp.nus.edu.sg

Abstract. There is an increasing interest in developing Phase Change Memory (PCM) based main memory systems. In order to retain the latency benefits of DRAM, such systems typically have a small DRAM buffer as a part of the main memory. However, for these systems to be widely adopted, limitations of PCM such as low write endurance and expensive writes need to be addressed. In this paper, we propose PCM-aware sorting algorithms that can mitigate writes on PCM by efficient use of the small DRAM buffer. Our performance evaluation shows that the proposed schemes can significantly outperform existing schemes that are oblivious of the PCM.

1 Introduction

Design of database algorithms for modern hardware has always been a prominent research area. Phase Change Memory (PCM) is a latest addition to the list of modern hardware demanding the design of PCM-friendly database algorithms. With PCM being better than flash memory (SSD) in terms of write endurance, read and write latency, focus has shifted to exploring the possibilities of exploiting PCM for databases. Because of the high density and lower power consumption of PCM as compared to DRAM, it is evident that PCM might be an alternative choice for main memory [2,3]. However, as PCM has a low write endurance and high write latency as compared to DRAM, it is essential to design algorithms such that they do not incur too many writes on PCM and thus prevent the hardware from getting worn out soon.

In-memory sorting is a write-intensive operation as it involves huge movement of data in the process of ordering it. Quick sort is the most expensive in terms of data movement though it is the best in time complexity. Selection sort involves fewest data movement but it has quadratic time complexity and also incurs a lot of scans on the data. In this paper, we design write-aware in-memory sorting algorithms on PCM. Like [3], we also use a small DRAM buffer to alleviate PCM writes using efficient data structures. We assume that the data movement between DRAM and PCM can happen seamlessly. This can be achieved through a hardware driven page placement policy that migrates pages between DRAM and PCM [6].

* Please note that this work was done while the author was at the National University of Singapore.
Our first sort algorithm constructs a histogram that allows us to bucketize the in-memory data such that either the depth or width of each bucket is DRAM-size bound. This is an important heuristic that we introduce to make our algorithm write-efficient as well as run-time efficient. Quick sort is employed on buckets that are depth bound and counting sort is used to sort the buckets that are width bound such that minimum writes are incurred on PCM. We further minimize PCM writes aggressively with an improved version of our algorithm. In this variant, we construct the histogram even before the data is read into PCM by sampling data directly from the disk.

If the unsorted data on disk sorted doesn’t entirely fit into the PCM at one go, external sort is performed which involves getting the data in chunks to main memory one by one, sorting the chunk, creating the runs on disk and finally merging those runs. We show in the experiments that our algorithm performs well even in the scenario where the entire data is not memory resident.

There have been few works in adapting database algorithms for PCM. In [1], a B+ tree index structure that incurs fewer writes to PCM (which is used as main memory) is proposed. Though the idea to reduce memory writes is beneficial, the B+ tree nodes (at least the leaves) are kept unsorted to obtain fewer modifications during insertion and deletion of data to the index. This leads to more expensive reads. An algorithm to adapt hash join is also proposed in [1] which minimizes the writes from the partitioning phase on PCM by storing the record ids of the tuples (or the differences between consecutive record ids) in the hash partitions. The records are in-place accessed during the join phase using the record ids. The algorithm also aims at achieving fewer cache misses.

PCM-aware sorting is of significance in the context of databases as it is used in many query processing and indexing algorithms. Our work to produce a PCM aware and efficient sorting algorithm can help alleviate the heavy read exchange the existing B+ tree index algorithm in [1] does. It can also be extended to obtain a PCM-aware sort merge join algorithm. To our knowledge, this is the first report work on sorting algorithms in memory-based PCM.

We present our basic and advanced PCM sorting algorithms in Sections 2.3 and 2.4 respectively. We report results of an extensive performance study in Section 3.

2 PCM-Aware Sorting Algorithm

Our goal is to design efficient sorting algorithms that incur as few writes on PCM as possible. As context, we consider external multi-way merge sort that comprises two main phases: (a) generating sorted runs, and (b) merging the runs into a single sorted file. Our key contribution is in the first phase where PCM-aware in-memory sorting is proposed.

For our hybrid architecture, we divide the main memory into three sections. PCM is divided into two partitions - a large chunk to hold the incoming unsorted data and a small chunk for use by a histogram. We refer to the first partition as sort-chunk, and the second partition as hist-chunk. The DRAM forms the third partition.