Abstract

Phase Change Memory (PCM) is a recently developed non-volatile memory technology that is expected to provide an attractive combination of the best features of conventional disks (persistence, capacity) and of DRAM (access speed). For instance, it is about 2 to 4 times denser than DRAM, while providing a DRAM-comparable read latency. On the other hand, it consumes much less energy than magnetic hard disks while providing substantively smaller write latency. Due to this suite of desirable features, PCM technology is expected to play a prominent role in the next generation of computing systems, either augmenting or replacing current components in the memory hierarchy. A limitation of PCM, however, is that there is a significant difference between the read and write behaviors in terms of energy, latency and bandwidth. A PCM write, for example, consumes 6 times more energy than a read. Further, PCM has limited write endurance since a memory cell becomes unusable after the number of writes to the cell exceeds a threshold determined by the underlying glass material.

Database systems, by virtue of dealing with enormous amounts of data, are expected to be a prime beneficiary of this new technology. Accordingly, recent research has investigated how database engines may be redesigned to suit DBMS deployments on PCM, covering areas such as indexing techniques, logging mechanisms and query processing algorithms. Prior database research has primarily focused on computing architectures wherein either a) PCM completely replaces the DRAM memory ; or b) PCM and DRAM co-exist side-by-side and are independently controlled by the software. However, a third option that is gaining favor in the architecture community is where the PCM is augmented with a small hardware-managed DRAM buffer. In this model, which we refer to as **DRAM_HARD**, the address space of the

application maps to PCM, and the DRAM buffer can simply be visualized as yet another level of the existing cache hierarchy. With most of the query processing research being preoccupied with the first two models, this third model has remained largely ignored. Moreover, even in this limited literature, the emphasis has been restricted to exploring execution-time strategies; the compile-time plan selection process itself being left unaltered.

In this thesis, we propose minimalist reworkings of current implementations of database operators, that are tuned to the DRAM_HARD model, to make them PCM-conscious. We also propose novel algorithms for compile-time query plan selection, thereby taking a holistic approach to introducing PCM-compliance in present-day database systems. Specifically, our contributions are two-fold, as outlined below.

First, we address the pragmatic goal of minimally altering current implementations of database operators to make them PCM-conscious, the objective being to facilitate an easy transition to the new technology. Specifically, we target the implementations of the "workhorse" database operators: sort, hash join and group-by. Our customized algorithms and techniques for each of these operators are designed to significantly reduce the number of writes while simultaneously saving on execution times. For instance, in the case of sort operator, we perform an in-place partitioning of input data into DRAM-sized chunks so that the subsequent sorting of these chunks can finish inside the DRAM, consequently avoiding both intermediate writes and their associated latency overheads.

Second, we redesign the query optimizer to suit the new environment of PCM. Each of the new operator implementations is accompanied by simple but effective write estimators that make these implementations suitable for incorporation in the optimizer. Current optimizers typically choose plans using a latency-based costing mechanism assigning equal costs to both read and write memory operations. The asymmetric read-write nature of PCM implies that these models are no longer accurate. We therefore revise the cost models to make them cognizant of this asymmetry by accounting for the additional latency during writes. Moreover, since the number of writes is critical to the lifespan of a PCM device, a new metric of write cost is introduced in the optimizer plan selection process, with its value being determined using the

above estimators.

Consequently, the query optimizer needs to select plans that simultaneously minimize query writes and response times. We propose two solutions for handling this dual-objective optimization problem. The first approach is a heuristic propagation algorithm that extends the widely used dynamic programming plan propagation procedure to drastically reduce the exponential search space of candidate plans. The algorithm uses the write costs of sub-plans at each of the operator nodes to decide which of them can be selectively pruned from further consideration. The second approach maps this optimization problem to the linear multiple-choice knapsack problem, and uses its greedy solution to return the final plan for execution. This plan is known to be optimal within the set of non interesting-order plans in a single join order search space. Moreover, it may contain a weighted execution of two algorithms for one of the operator nodes in the plan tree. Therefore overall, while the greedy algorithm comes with optimality guarantees, the heuristic approach is advantageous in terms of easier implementation.

The experimentation for our proposed techniques is conducted on Multi2sim, a state-of-the-art cycle-accurate simulator. Since it does not have native support for PCM, we made a major extension to its existing memory module to model PCM device. Specifically, we added separate data tracking functionality for the DRAM and PCM resident data, to implement the commonly used read-before-write technique for PCM writes reduction. Similarly, modifications were made to Multi2sim's timing subsystem to account for the asymmetric read-write latencies of PCM. A new DRAM replacement policy called N-Chance, that has been shown to work well for PCM-based hardware, was also introduced.

Our new techniques are evaluated on end-to-end TPC-H benchmark queries with regard to the following metrics: number of writes, response times and wear distribution. The experimental results indicate that, in comparison to their PCM-oblivious counterparts, the PCM-conscious operators collectively reduce the number of writes by a factor of 2 to 3, while concurrently improving the query response times by about 20% to 30%. When combined with the appropriate plan choices, the improvements are even higher. In the case of Query 19, for instance, we obtained a 64% savings in writes, while the response time came down to two-thirds of the original.

In essence, our algorithms provide both short-term and long-term benefits. These outcomes augur well for database engines that wish to leverage the impending transition to PCM-based computing.