Hamming Tree: The Case for Memory-Aware Bit Flipping Reduction for NVM Indexing

Saeed Kargar
skargar@ucsc.edu
UC Santa Cruz

Faisal Nawab
fnawab@ucsc.edu
UC Santa Cruz

ABSTRACT

Non-Volatile Memory (NVM) is improving the performance and cost-efficiency of data management systems. However, they introduce salient challenges that were not considered in traditional memory architectures. In particular, write endurance in NVM is significantly lower than other memory technologies which threatens the practicality and longevity of NVM devices. In this paper, we introduce *Hamming Tree*, an indexing structure that can be augmented with existing database indexing technologies to increase the write endurance of NVM. Hamming Tree proposes a new way of increasing write endurance by directing write operations to memory locations that would minimize the number of incurred bit flips. This method is capable of significantly increasing the endurance of NVM compared to existing write endurance methods that remain agnostic to the underlying memory such as local write optimizations and write amplification techniques. Our evaluations show that Hamming Tree can reduce bit flipping (hence increasing write endurance) by up to %93 on both traditional and optimized NVM indexing technologies with minimal changes to their indexing structure.

1 INTRODUCTION

To overcome the limited write endurance in NVM, two major approaches were developed: 1) the write optimization techniques developed by the storage community, which are based on a *Read-Before-Write* (RBW) pattern [1, 2]; and 2) the write reduction techniques, such as caching [6] and delayed merging [4], which are developed by data management community. However, these existing methods miss a crucial opportunity to increase write endurance significantly. This opportunity is to be memory-aware [5]. Prior methods pick the memory location for a write operation arbitrarily (new data items select an arbitrary location in memory and updates to data items overwrite the previous location.) This misses the opportunity to judiciously pick a memory location that is similar to the value to be written. When the new value and the value to be overwritten are similar, this means that the number of bit flips is going to be lower.

2 METHODOLOGY

In this work, we propose Hamming Tree, an auxiliary data structure that can be augmented with existing indexes. Hamming Tree is a data structure that organizes free memory locations based on their hamming distance. It can be built upon any existing tree-based data structure—whether they are designed for NVM or not—to improve their performance in terms of NVM write endurance. Hamming Tree is augmented with a data indexing structure. The data indexing structure handles the regular indexing of keys and values, and Hamming Tree handles the mapping of free memory locations for future writes and updates.

In this work, we assume a hybrid memory architecture consisting of a DRAM component (for volatile main-memory operations) and an NVM component (to persist data). In the hybrid memory architecture, both components are placed on the same memory bus, enabling mapping DRAM and NVM on a single physical address space [3]. The overall design consists of a Hamming Tree in DRAM, a pluggable data index that is used to index keys and values, and a K/V data zone to store the K/V pairs, which both are on the NVM device. Our implementation supports K/V operations such as `get()`, `put()`, and `delete()`.

Our evaluation shows that augmenting Hamming Tree to existing indexing structures, such as B+-tree, LSM-based persistent K/V store, cache optimized NVM index, and write-friendly hashing schemes, reduces bit flipping by up to 93% compared to both RBW and write amplification methods on both synthetic and real-world data sets.

3 CONCLUSION

In this paper, we introduce Hamming Tree, a simple tree-based data structure that can be augmented with existing database to improve the write endurance of NVMs. The main advantage of Hamming Tree is that it is a pluggable method that has the potential of bringing DRAM-optimized data structures, such as ones based on LSM-Tree and B+-Tree, into the NVM realm without having to worry about degrading the performance and write-endurance of NVMs.

REFERENCES

[1] Sangyeun Cho and Hyunjin Lee. 2009. Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance. In *MICRO 2009*. 347–357.
[2] David B Dgien et al. 2014. Compression architecture for bit-write reduction in non-volatile memory technologies. In *NANOARCH 2014*. IEEE, 51–56.
[3] Gaurav Dhiman, Raid Ayoub, and Tajana Rosing. 2009. PDRAM: A hybrid PRAM and DRAM main memory system. In *2009 46th ACM/IEEE Design Automation Conference*. IEEE, 664–669.
[4] Sudarsun Kannan et al. 2018. Redesigning LSMs for nonvolatile memory with NoveLSM. In *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*. 993–1005.
[5] Saeed Kargar, Heiner Litz, and Faisal Nawab. 2020. Predict and Write: Using K-Means Clustering to Extend the Lifetime of NVM Storage. *arXiv preprint arXiv:2011.02556* (2020).
[6] Ismail Oukid et al. 2016. FPTree: A hybrid SCM-DRAM persistent and concurrent B-tree for storage class memory. In *Proceedings of the 2016 International Conference on Management of Data*. 371–386.