Querying Persistent Graphs using Solid State Storage

Manos Athanassoulis[‡*] Bishwaranjan Bhattacharjee[◇] Mustafa Canim[◇] Kenneth A. Ross[◇§]

[‡]École Polytechnique Fédérale de Lausanne
manos.athanassoulis@epfl.ch

[◇]IBM Watson Research Labs
{*mustafa, bhatta, rossak*}*@us.ibm.com*

[§]Columbia University
kar@cs.columbia.edu

ABSTRACT

Solid State Drives (SSDs) are an important component of secondary storage systems. While Hard Disk Drives (HDDs) are cheaper per gigabyte, SSDs are cheaper per random I/O per second. A variety of of solid-state technologies are being developed with NAND flash being the most mature, and Phase Change Memory (PCM) beginning to enter the marketplace. Compared with flash, PCM has finer-grained addressability and higher write endurance. PCM is also expected to offer lower read and write response times.

In this work we study the use of solid-state storage in latency-bound applications, a type of workload that can benefit from the characteristics of flash and PCM technologies. We identify graph processing and Resource Description Framework (RDF) query processing as candidate applications. Using an early PCM prototype device, we demonstrate the benefits of PCM for this workload and compare with a flash device. Moreover, we describe *Pythia*, a prototype RDF repository designed for Solid State Storage. Using Pythia we investigate whether the predicted benefits for this type of workload can be achieved in a properly designed RDF repository.

1. INTRODUCTION

Flash devices have superior random read performance compared to magnetic hard-drives but suffer from several limitations. First, there is a significant asymmetry in read and write performance. Second, only a limited number of updates can be applied on a flash device before it becomes unusable; this number is decreasing with newer generations of flash [1]. Third, writing on flash not only is much slower than reading and destructive of the device, but it has proven to interfere with the redirection software layers, known as Flash Translation Layers (FTL) [6].

PCM addresses some of these challenges. The endurance of PCM cells is significantly higher than NAND flash [2], although still not close to that of DRAM. Unlike NAND flash, PCM does not require the bulk erasure of large memory units before it can be rewritten. Moreover, while cost is still uncertain, for our purposes, we assume normal cell size competitiveness and standard volume economics will apply to this technology as it ramps into high volume.

The most pronounced benefit of solid state storage over hard disks is the difference in response time for random accesses. Hence, we identify *dependent reads* as an access pattern that has the potential for significant performance gains. Latency-bound applications like path processing [7] in the context of graph processing, or RDF-data processing are typical examples of applications with such access patterns. The Resource Description Framework (RDF) [3] data model is widely adopted for several on-line, scientific or knowledge-based datasets because of its simplicity in modeling and the variety of information it can represent.

We find that PCM-based storage is an important step towards better latency guarantees with minor bandwidth penalties for random I/O, and we identify a concurrency trade-off between maximizing bandwidth and minimizing latency. In order to measure the performance benefit (decrease in response time) for long path queries we implement a simple benchmark and we compare the response times when using flash and PCM. We observe that PCM can yield 1.5x to 2.5x smaller response times for any bandwidth utilization without any graph-aware optimizations. We take this observation one step further and we design a new data layout suitable for RDF data optimized for a solid state storage layer. The proposed layout increases the locality of related information and decreases the cost of graph traversals by storing more linkage information (i.e., metadata about how to navigate faster when a graph is traversed).

In this short paper, which is based on recently published work [5], we show the benefits of path processing applications over data that is resident in solid state storage. First, we present a custom graph benchmark that is used to highlight the differences between two solid state technologies through two representative devices: a state-of-the-art flash device and an enterprise-level PCM prototype provided to us by Micron. Second, we develop a prototype RDF repository to show the benefits that RDF processing can have if it adopts PCM storage.

2. PATH PROCESSING

Current PCM prototypes behave as a "better" flash [9], in the sense that they have faster and more stable reads. We argue that the best way to make the most of this behavior is in the domain of applications with *dependent reads*. Hence, we create a simple benchmark that performs path traversals over randomly created graphs to showcase the potential benefits using PCM as secondary storage for such applications.

We create a benchmark that models path traversal queries by graph traversal over a custom built graph. The graph is stored in

[*]The majority of this work was completed while the author was an intern at IBM T. J. Watson Research Center, Hawthorne, NY.

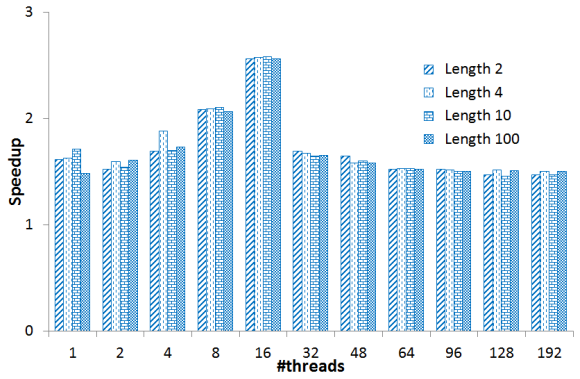

Figure 1: Custom path processing benchmark: Speedup of PCM over flash

fixed-size pages (each page has one node) and the total size of the graph is 5GB. Each node has an arbitrary number of edges (between 3 and 30). The path traversal queries are implemented as link following traversals of a random edge in each step. Each query starts from a randomly selected node of the graph and it follows at random one of the descendant nodes. When multiple queries are executed concurrently, because of the absence of buffering, locality will not yield any performance benefits. Each query keeps reading a descendant node as long as the maximum length is not reached.

We use a 74GB FusionIO ioDrive (SLC) and a 12GB Micron PCM prototype (SLC). The PCM device offers 800MB/s maximum read bandwidth, $20\mu s$ hardware read latency[1] (for 4K reads) and $250\mu s$ hardware write latency (for 4K writes), while the endurance is estimated to be 10^6 write cycles. While PCM chips can be byte-addressable the PCI-based PCM prototype available to us uses 4KB pages for reads and writes. The flash device offers 700MB/s read bandwidth (for 16K accesses) and hardware read latency as low as $50\mu s$ in the best case.

Figure 1 shows the speedup of the query response time for different values of path length and number of concurrently issued queries. The speedup varies between 1.5x and 2.5x having the maximum number of 16 threads. We observe as well that the length of the query does not play any important role. The sudden drop in speedup for 32 threads is attributed to the an observed sweet spot for flash for this level of concurrency.

3. PYTHIA

Next, we design an RDF-processing system which takes into account the graph-structure of the data and has the infrastructure needed to support any query over RDF data. Pythia is based on the notion of an *RDF-tuple* [5].

RDF data are often stored [8] as triples of <*Subject, Predicate, Object*> in a triple-store [10], as set of properties in property tables [11] or as vertical partitions grouped by predicate [4]. Vertical partitioning and property tables assume that the stored RDF data have some relational structure which is used to decompose the data in sub-tables.

An *RDF-tuple* is a hierarchical tree-like representation of a set of triples given an ordering of subject, predicate, and object. In Pythia, we store RDF-tuples in two complementary hierarchies: the subject/predicate/object (SPO) hierarchy and the object/predicate/subject (OPS) hierarchy. Each triple will thus be represented in RDF-tuples stored in two places, the SPO-store and the OPS-store. We envision RDF-tuples to be stored as a tuple with variable length in a database page containing many such tuples. Figure 2 shows how an RDF-tuple is laid out within a page.

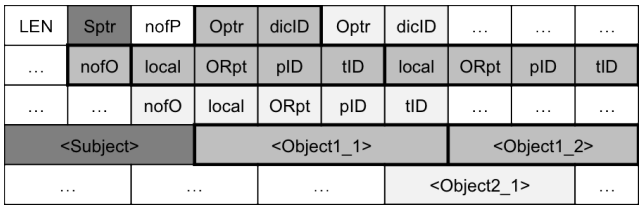

Figure 2: RDF-tuple layout

We experiment with Pythia and we corroborate the performance benefit achieved in the simple benchmark. Pythia for both devices (Flash and PCM) scales close to linearly until 4 threads (7KQ/s for PCM and 4.5KQ/s for flash leading to 1.56x speedup), while for higher number of threads the PCM device is more stable showing speedup from 1.8x to 2.6x. Last but not least, we compare the performance of Pythia repository against the state of the art RDF-3X repository in a limited set of tests and we find that Pythia is competitive with RDF-3X.

4. CONCLUSIONS

We have described our experiences and observations of a real PCM prototype. At least for the near future, PCM-based devices are going to be "better" flash devices, rather than a new incomparably fast and more efficient storage technology. The limits of interconnecting interfaces, and the overhead of the software stack used to access the devices, are significant determinants of overall system performance. As new generations of PCM become available, and as new software technologies are developed to minimize overheads, PCM device performance and access granularity constraints are likely to improve.

5. ACKNOWLEDGMENTS

The authors would like to thank Micheal Tsao of IBM T.J. Watson Research for his help in setting up the machine used in the experiments, and Micron Inc. for providing access to an early PCM prototype.

6. REFERENCES

[1] NAND Flash Trends for SSD/Enterprise. http://www.bswd.com/FMS10/FMS10-Abraham.pdf.
[2] Ovonic Unified Memory, page 29. http://ovonyx.com/technology/technology.pdf.
[3] Resource Description Framework (RDF). http://www.w3.org/RDF/.
[4] D. J. Abadi et al. Scalable semantic web data management using vertical partitioning. *VLDB*, 2007.
[5] M. Athanassoulis et al. Path processing using Solid State Storage. *ADMS*, 2012. http://www.adms-conf.org/athanassoulis_adms12.pdf.
[6] L. Bouganim et al. uFLIP: Understanding Flash IO Patterns. *CIDR*, 2009.
[7] A. Gubichev and T. Neumann. Path Query Processing on Very Large RDF Graphs. *WebDB*, 2011.
[8] O. Hassanzadeh et al. Publishing relational data in the semantic web. *ESWC*, 2011.
[9] J.-Y. Jung et al. Characterizing a Real PCM Storage Device (poster). *NVMW*, 2011.
[10] T. Neumann and G. Weikum. RDF-3X: a RISC-style engine for RDF. *PVLDB*, 2008.
[11] K. Wilkinson. Jena property table implementation. *SSWS*, 2006.

[1]Software read latency is about 16–17μs, which is negligible compared to magnetic disk I/O latency, but is close to 50% of the total latency for technologies like PCM.