Katana Graph Engine on Intel Optane DC Persistent Memory (PMM)

1 Introduction

Disruptive memory technologies are leading to major improvements in both the capacity and bandwidth of memories. When coupled with fast cores, these memories enable us to tackle large-scale data analytics problems at much lower cost and power than ever before. One such memory technology is the Intel® Optane™ DC persistent memory. Intel Optane DC persistent memory modules are like DRAM in form factor and can be configured as volatile main memory, persistent memory or as a combination.

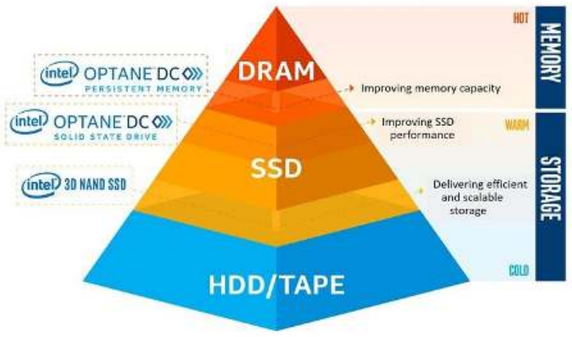
intel OPTANE DC PERSISTENT MEMORY
intel OPTANE DC SOLID STATE DRIVE
intel 3D NAND SSD
DRAM
SSD
HDD/TAPE
Improving memory capacity
Improving SSD performance
Delivering efficient and scalable storage
HOT
WARM
COLD
MEMORY
STORAGE

Figure1: Intel Optane DC Persistent Memory

One application area that benefits from this new memory technology is graph analytics. Graph analytics systems today must handle very large graphs such as the Facebook friends' graph, which has more than a billion nodes and 200 billion edges, and the indexable Web graph, which has roughly 100 billion nodes and trillions of edges. Parallel computing is essential for processing graphs of this size in reasonable time.

To process such graphs with billions of nodes and edges, two methods are commonly employed:

- use distributed-memory machines (clusters) that have sufficient main memory for in-memory processing of the graphs, or
- use secondary storage to store the graphs and use out-of-core algorithms to read a portion of the graph into DRAM at a time under software control and process it.

In both approaches, it is usually necessary to rethink algorithms and data structures from scratch. Intel Optane DC persistent memory presents an interesting alternative since shared-memory graph analytics can be used out-of-the-box to process very large graphs. Katana has been studying this approach on a 2-socket machine with 2nd-generation Intel® Xeon® Scalable processors with 6TB of Intel Optane DC persistent memory. Our results show that graph analytics on Intel Optane DC persistent memory can be competitive with performing analytics on production clusters.

2 Optane DC PMM Modes

Optane DC PMM has 2 modes of operation:

1. **Memory-Mode:** In memory mode, Optane PMM is treated as main memory, and DRAM acts as direct-mapped cache called near-memory. The granularity of caching from Optane PMM to DRAM is 4KB. This enables the system to deliver DRAM-like performance at substantially lower cost and power with no modifications to the application. Although the memory media is persistent, the software sees it as volatile memory. This enables systems to provide up to 6TB of randomly accessible storage, which is expensive if implemented in DRAM.
2. **App-Direct Mode:** In app-direct mode, Optane PMM modules are treated as byte-addressable persistent memory. One compelling case for app-direct mode is in large memory databases where indices can be stored in persistent memory to avoid rebuilding them on reboot, achieving a significant reduction in restart time.

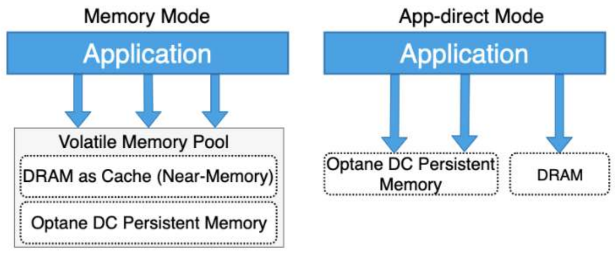
Memory Mode
Application
Volatile Memory Pool
DRAM as Cache (Near-Memory)
Optane DC Persistent Memory
App-direct Mode
Application
Optane DC Persistent Memory
DRAM

Figure 2: Modes in Optane PMM

In this article, we focus on the memory mode of Optane DC PMM.

3 Efficient Memory Hierarchy

In our study [1], we found the following system-level knobs to play a key role in efficiently using the Optane PMM memory hierarchy and getting good performance for graph analytics workloads.

1. **NUMA-Aware Allocation**: In an Optane PMM machine, applications must not only maximize local NUMA accesses, but must also use a NUMA policy that maximizes the near-memory used in order to reduce DRAM "conflict misses" since DRAM acts as a direct-mapped cache for Optane PMM in the memory mode. The limited DRAM size increases the probability of conflict misses arising from physical addresses that map to the same cache line. Katana provides application-level NUMA-allocation policies such as NUMA-Interleaved and NUMA-Blocked. The Katana Graph Engine also optimizes program execution to exploit NUMA locality; for example, it performs NUMA-aware dynamic load balancing.

2. **NUMA Migrations**: NUMA migration must be turned off because of its overheads: (a) it requires book-keeping to track accesses to the pages to select pages for migration, which is costlier on Optane as compared to DRAM, and (b) migration changes the virtual-to-physical address mapping, which makes the Page Table Entries (PTEs) cached in CPU's Translation Lookaside Buffers (TLBs) stale, causing TLB shootdown on each core to invalidate stale entries. TLB shootdowns add to DRAM access latency.

3. **Page Size Selection**: A page size of 2MB (termed huge pages) performs better than the default 4KB page size because for graph workloads, the time spent in handling TLB misses can be a performance bottleneck. Katana automatically allocates 2MB pages when available instead of relying on transparent huge pages (THP) provided by the operating system. We have observed this policy to perform better.

4 Efficient Graph Algorithms

In general, there are many algorithms that can solve a given graph problem; for example, the single-source (weighted) shortest-path (sssp) problem can be solved using Dijkstra's algorithm, the Bellman-Ford algorithm, chaotic relaxation, or delta-stepping. The Bellman-Ford algorithm is an example of a round-based algorithm: it is organized as a series of rounds, and in each round, the algorithm applies the well-known relaxation operator [10] to vertices to update their labels based on the labels of their immediate neighbors. On the other hand, Dijkstra's algorithm and the delta-stepping algorithm are asynchronous algorithms; instead of rounds, these algorithms use worklists of active nodes to which the operator must be applied, and they terminate when the worklist is empty. These algorithms may have different asymptotic complexities and different amounts of parallelism.
In addition, a given algorithm can usually be implemented in different ways and these differences can have a major effect on parallel performance; implementations with fine-grain locking, for example, usually perform better than those with coarse-grain locking. On machines with Optane PMM, it is advantageous to use algorithms with non-vertex operators [10] and asynchronous algorithms. The Katana Graph Engine provides a general non-vertex programming model [6] and highly scalable concurrent data structures such as concurrent graph representations and concurrent worklists to permit application programmers to implement efficient algorithms. The Katana core graph library provides graph analytics applications that use efficient algorithms on Optane PMM machines. The graph engine also provides a streaming graph partitioner and a communication runtime optimized for graph computations, so applications can also run on clusters with minimal effort by application programmers.

5 Performance Evaluation

The Optane DC PMM experiments reported in this note are conducted on a 2-socket machine with Intel's second-generation Xeon scalable processor ("Cascade Lake") with 48 cores (up to 96 threads with hyperthreading) with a clock rate of 2.2 Ghz. The machine has 6TB of Optane PMM and 384GB of DDR4 RAM.

To compare this Optane DC PMM solution with a cluster-based solution, we used the Stampede cluster at the Texas Advanced Computing Center using up to 256 Intel Xeon Platinum 8160 ("Skylake") 2 socket machines each with 48 cores with a clock rate of 2.1 Ghz and 192GB DDR4 RAM.

We study [1] widely used graph kernels such as betweenness centrality (bc), breadth first search (bfs), connected components (cc), pagerank (pr), single source shortest path (sssp), and triangle counting (tc). These kernels are run on both real-world and synthetically generated graphs. We use very large real-world graphs including social networks [2] such as twitter and friendster, protein networks like iso_m100, and web-crawls such as clueweb12, uk14, and wdc12 (the largest publicly available graph dataset).

Figure 3 shows execution times of the graph kernels for the clueweb12[3] graph, which is one of the largest publicly available web-crawls with 978 million nodes and 42.6 billion edges. Clueweb12 (~324GB in compressed binary format) fits in DRAM on our machine. For each kernel, we measured the execution times when (i) the graph is stored entirely in DRAM, and (ii) the graph is stored entirely in Optane PMM at the start of the computation and brought into DRAM as needed by the computation. The results show that Optane PMM can deliver performance comparable to DRAM for this graph.

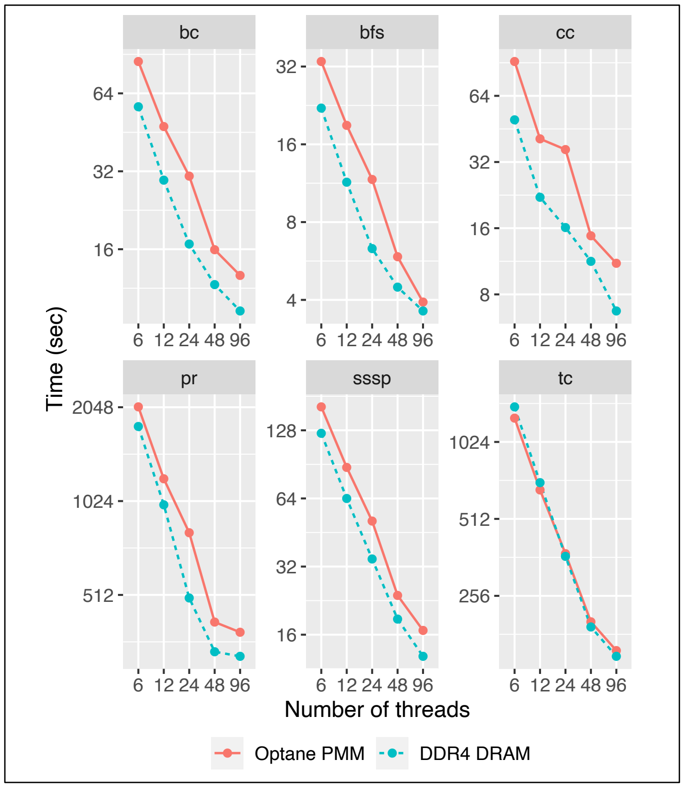

Figure 3: Strong scaling in execution time of benchmarks in Galois using DDR4 DRAM and Optane PMM.
Applications: bc: Betweenness-Centrality, bfs: Breadth First Search, cc: Weakly Connected Components, pr: PageRank, sssp: Single Source (weighted) Shortest Path, tc: Triangle Counting
Dataset: Clueweb12 (|N|: 0.98B, |E|: 42.6B)

Figure 4 shows the performance of the Katana Graph Engine [5] on different platform configurations, normalized to the performance on 32 machines of the Stampede cluster for single source (weighted) shortest path (sssp) on wdc12[4] which is the largest publicly available web-crawl. Observe that on the cluster, Katana scales well as the number of machines is increased; graph analytics on 128 Stampede machines is on average 2.7x faster than on 32 machines. Interestingly, graph analytics on a single Cascade Lake machine with Optane PMM outperforms even 64 Skylake machines of the Stampede production cluster! The

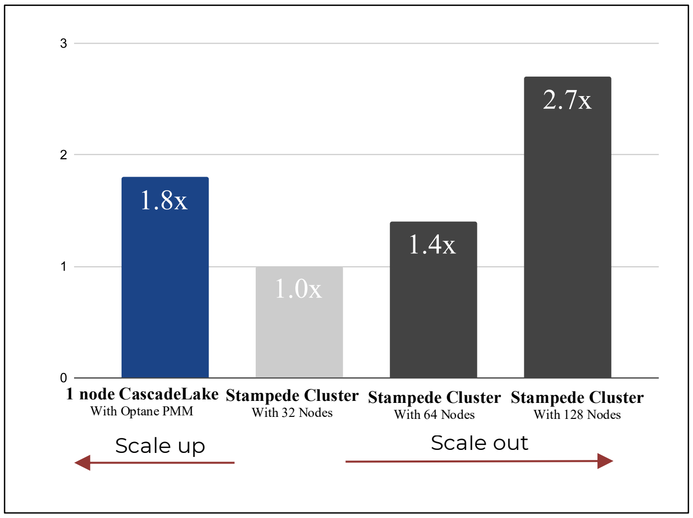

Figure 4: Optane PMM shared-memory machine vs. Stampede distributed cluster.
Application: sssp: Single source (weighted) Shortest Path
Dataset: wdc12 (|N|: 3.5B, |E|: 128B)

main reason is the Optane PMM solution avoids the overhead of cluster communication.

6 Conclusion

The Katana Graph Engine can be deployed on a single Xeon machine equipped with Optane DC PMM to provide a very high-performance platform for graph analytics on extremely large graphs that are several TB in size. These performance advantages arise from the generality of Katana's non-vertex programming model, and its efficient compute engine, support for asynchronous execution, highly scalable concurrent data structures, NUMA-aware allocation policies, and huge pages support. The graph engine also supports the use of clusters of such machines for graph analytics, providing a scale-out solution if needed for even larger graphs.

For more details, visit https://www.katanagraph.com.

References:

[1] G. Gill, R. Dathathri, L. Hoang, R. Peri, and K. Pingali. 2020. "Single machine graph analytics on massive datasets using Intel Optane DC persistent memory". VLDB, 2020.

[2] SNAP Datasets: https://snap.stanford.edu/data/com-Friendster.html

[3] T. L. Project. The ClueWeb12 Dataset, 2013.

[4] R. Meusel, S. Vigna, O. Lehmberg, and C. Bizer. Web data commons – hyperlink graphs, 2012.

[5] R. Dathathri, G. Gill, L. Hoang, H. Dang, A. Brooks, N. Dryden, M. Snir, and K. Pingali. 2018. Gluon: a communication-optimizing substrate for distributed heterogeneous graph analytics. PLDI'18

[6] D. Nguyen, A. Lenharth, and K. Pingali. 2013. A lightweight infrastructure for graph analytics. SOSP '13