POSTER: Accelerating Unmodified Databases using Persistent Memory and Flash Storage Tiers

Amit Golander
NetApp
Amit.Golander@netapp.com

Netanel Katzburg
NetApp

Omer Zilberberg
NetApp

ABSTRACT

Recent breakthroughs in Storage Class Memory (SCM) technologies have driven Persistent Memory (PM) devices to become commodity off-the-shelf components in 2018. PM devices are byte addressable, plug into the memory interconnect, and run at near memory speeds, densities and price points. PM availability is led by *Fast PM*, comprised from backed-DRAM devices such as NVDIMM-N, and will follow soon with *Slow PM*, comprised of new SCM materials, such as Intel 3D XPoint NVDIMM. Fast and Slow PM devices vary in speeds, densities and cost, but both are orders of magnitude faster than Flash devices and an order of magnitude more expensive per GB.

A PM-based file system was shown to accelerate unmodified transactional databases [2, 1], when the entire dataset was placed on NVDIMM-N cards. Most databases however are large and cannot fit entirely into the limited capacity provided by PM devices and even if they could - the high price per GB would prevent wide adoption.

This work explores accelerating unmodifed databases using software that supports both NVDIMM-N and Flash devices and can transparently tier data between them. Ideally, this would provide the performance benefits of PM, while maintaining the cost structure of Flash solutions. We run a transactional workload (DBT-2) on an unmodified Postgresql [3] database, and compare the default block-based file system running on Flash NVMe to a file system, which is the first to support auto-tiering between byte-addressable NVDIMM devices and block-addressable Flash. The rest of the server and the operating system version are identical for both configurations (refer to Table 1).

M1FS auto-tiering between PM pages and Flash blocks was implemented using the following architecture:

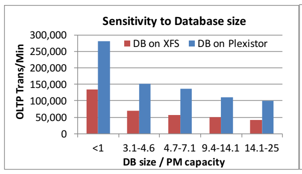
- Each 4KB of data can reside on a PM page, a Flash block or both at the same time.
- Data is speculatively copied to a Flash block ahead of needing to reuse the PM page, in order to hide the slower Flash access time.
- Unless data is modified, an existing Flash copy is maintained in order to reduce Flash wearout.
- PM pages are maintained in many queues in order to reduce the probability of lock contention when many cores are used concurrently
- Page allocations are preferably done from PM attached to the CPU socket (NUMA-aware FS)

SYSTOR '18, June 4–7, 2018, HAIFA, Israel
© 2018 ACM. ISBN 978-1-4503-5849-1/18/06...$15.00
DOI: https://doi.org/10.1145/3211890.3211902

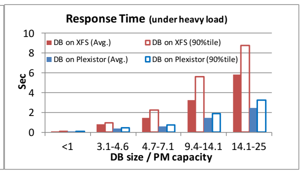
	Server w/ Traditional block-based FS	Server w/ byte-addressable FS
Database	Postgresql 9.5	
OS	Linux CentOS 7.2 (Kernel 4.5)	
CPU	Dual socket Intel Xeon E5-2650 v4 12 cores, 24 HW threads per socket; 2.20GHz	
Memory	64GB DDR4 at 2133MHz (Volatile)	
File System	XFS	M1FS v2.0.1
Storage	1600GB Flash NVMe	Tier 1: 64GB NVDIMM-N Tier 2: 1600GB Flash

Table 1: Hardware and software configuration

Figures 1a and 1b show that throughput and latency are much better than traditional block-based solutions. They show that performance degrades as the database size exceeds the PM tier, but at an acceptable rate that corresponds with how traditional storage degrades when exceeding main memory size. Using NVDIMMs in conjunction with NVMe achieved 1.8-3x improved response time, while serving 2.1-2.4x more transactions per minute, and overall 4.2 to 5.6x better. The overall speedup is relatively stable across all database sizes.



(a) Troughput improvement



(b) Latency improvement, lower is better

Figure 1: Performance enhancement. Response time is lower, throughput is higher

1. REFERENCES

[1] A. Golander, S. Manole, and Y. Korman. Persistent Memory over Fabric (PMoF). In *SYSTOR 2017 Poster*.
[2] N. Katzburg, A. Golander, and S. Weiss. Storage becomes first class Memory. In *ICSEE 2016*, pages 1–5.
[3] PostgreSQL. The postgresql global development group. www.postgresql.org/, 2016.