Reducing DRAM Footprint with NVM in Facebook (EuroSys'18)

Assaf Eisenman[1], Darryl Gardner[2], Islam AbdelRahman[2], Jens Axboe[2], Siying Dong[2], Kim Hazelwood[2], Chris Petersen[2], Asaf Cidon[1], Sachin Katti[1]

[1]Stanford University [2]Facebook, Inc.

Extended Abstract

Modern key-value stores like RocksDB and LevelDB are highly dependent on DRAM, even when using a persistent storage medium (e.g. flash) for storing data. Since DRAM is still about 1000× faster than flash, these systems typically leverage DRAM for caching hot indices and objects.

MySQL databases are often implemented on top of these key-value stores. For example, MyRocks, which is built on top of RocksDB, is the primary MySQL database in Facebook and is used extensively in data center environments. MyRocks stores petabytes of data and is used to serve real-time user activities across large-scale web applications [6]. In order to achieve high throughput and low latency, MyRocks consumes a large amount of DRAM. In our case, we leverage a MyRocks server that consists of 128 GB DRAM and 3 TB of flash. Many other key-value stores are also highly dependent on DRAM, including Tao [2] and Memcached [7].

As DRAM faces major scaling challenges [4, 5], its bit supply growth rate has experienced a historical low [3]. Together with the growing demand for DRAM, these trends have led to problems in global supply [3], increasing the total cost of ownership (TCO) for data center providers.

Our goal in this work is to reduce the DRAM footprint of MyRocks, while maintaining comparable mean latency, 99th percentile latency, and queries-per-second (QPS). The first obvious step would be to simply reduce the DRAM capacity on a MyRocks server configuration. Unfortunately, reducing the amount of DRAM degrades the performance of MyRocks. To demonstrate this point, Figure 1 shows that when reducing the DRAM cache in MyRocks from 96 GB to 16 GB, the mean latency increases by 2×, and P99 latency increases by 2.2×.

	DRAM	NVM	TLC Flash
Max Read Bandwidth	75 GB/s	2.2 GB/s	1.3 GB/s
Max Write Bandwidth	75 GB/s	2.1 GB/s	0.8 GB/s
Min Read Latency	75 ns	10 μs	100 μs
Min Write Latency	75 ns	10 μs	14 μs
Endurance	Very High	30 DWPD	5 DWPD
Capacity	128 GB	375 GB	1.6 TB

Table 1. An example of the key characteristics of DDR4 DRAM with 2400 MT/s, NVM block device, and 3D TLC flash device in a typical data center server. DWPD means Device Writes Per Day.

NVM offers the potential to reduce the dependence of systems like MyRocks on DRAM. NVM comes in two forms: a more expensive byte-addressable form, and a less expensive block device. Since our goal is to reduce total cost of ownership, we use NVM as a block device, which has a cost about an order of magnitude less per bit than DRAM. Table 1 provides a breakdown of key DRAM, NVM, and Triple-Level Cell (TLC) flash characteristcs in a typical data center server. An NVM block device offers 10× faster reads and has 5× better durability than flash [1]. Nevertheless, NVM is still more than 100× slower than DRAM. To make up for that, NVM can be used to reduce the number of accesses to the slower flash layer, by significantly increasing the overall cache size.

However, there are several unique challenges when replacing DRAM with an NVM device. First, NVM suffers from significantly lower read bandwidth than DRAM, and its bandwidth heavily depends on the block size. Therefore, simply substituting DRAM with NVM will not achieve the desired performance due to read bandwidth constraints. In addition, significantly reducing the DRAM footprint results in a smaller amount of data that can be cached for fast DRAM access. This requires a redesign of the indices used to lookup objects in RocksDB. Second, using smaller data blocks reduces their compression ratio, thus increasing the total database size on flash. Third, unlike DRAM, NVM has write durability constraints. If it is used as a DRAM replacement without modification, it will wear out too quickly. Fourth, because NVM has a very low latency relative to other block devices, the operating system interrupt overhead becomes significant, adding a 20% average latency overhead.

We present MyNVM, a system built on top of MyRocks, that significantly reduces the DRAM cache using a second-layer NVM cache. Our contributions include several novel

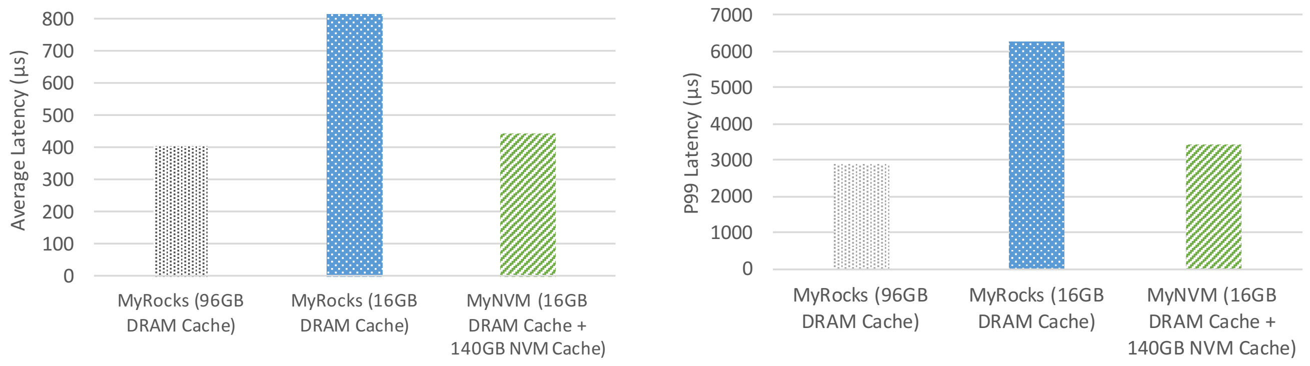

Figure 1. Average and P99 latencies for different cache sizes in MyRocks, compared with MyNVM, using real production workloads.

design choices that address the problems arising from adopting NVM in a data center setting:

1. NVM bandwidth and index footprint: Partititioning the database index enables the caching of fine grained index blocks. Thus, the amount of cache consumed by the index blocks is reduced, and more DRAM space is left for caching data blocks. Consequently, the read bandwidth of the NVM is also reduced. When writing to NVM, compressed blocks are grouped together in the same NVM page (when possible) and aligned with the device pages, to further reduce NVM read bandwidth.
2. Database size: When using smaller block sizes, compression becomes less efficient. To improve the compression ratio we utilize a dictionary compression.
3. Write durability: Admission control to NVM only permits writing objects to NVM that will have a high hit rate, thus reducing the total number of writes to NVM.
4. Interrupt latency: Hybrid polling (where the kernel sleeps for most of the polling cycle) can reduce the I/O latency overhead by 20% in certain server configurations, while minimizing CPU consumption.

We implemented MyNVM and deployed it in a MyRocks production environment in Facebook, and were able to reduce the DRAM cache by 6×. Our implementation achieves a mean latency and P99 latency of 443 μs and 3439 μs, respectively. As shown in Figure 1, both its mean and P99 latencies are 45% lower than the MyRocks server with a low amount of DRAM. Compared to the MyRocks with the high amount of DRAM, MyNVM's average latency is only 10% higher, and its P99 latency is 20% higher. Its QPS is 50% higher than the MyRocks server with a small amount of DRAM, and only 8% lower than MyRocks with the large DRAM cache.

Explicit Citation

Assaf Eisenman, Darryl Gardner, Islam AbdelRahman, Jens Axboe, Siying Dong, Kim Hazelwood, Chris Petersen, Asaf Cidon, and Sachin Katti. 2018. ”Reducing DRAM footprint with NVM in Facebook”. In Proceedings of the Thirteenth EuroSys Conference (EuroSys ’18). ACM, New York, NY, USA, Article 42, 13 pages

URL

`https://doi.org/10.1145/3190508.3190524`

References

[1] Intel Optane DC p4800x specifications. `https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/data-center-ssds/optane-dc-p4800x-series.html`.

[2] N. Bronson, Z. Amsden, G. Cabrera, P. Chakka, P. Dimov, H. Ding, J. Ferris, A. Giardullo, S. Kulkarni, H. Li, M. Marchukov, D. Petrov, L. Puzar, Y. J. Song, and V. Venkataramani. TAO: Facebook's Distributed Data Store for the Social Graph. In *Presented as part of the 2013 USENIX Annual Technical Conference (USENIX ATC 13)*, pages 49–60, San Jose, CA, 2013.

[3] D. Exchange. DRAM supply to remain tight with its annual bit growth for 2018 forecast at just 19.6`www.dramexchange.com`.

[4] U. Kang, H.-s. Yu, C. Park, H. Zheng, J. Halbert, K. Bains, S. Jang, and J. S. Choi. Co-architecting controllers and dram to enhance dram process scaling. In *The memory forum*, pages 1–4, 2014.

[5] S.-H. Lee. Technology scaling challenges and opportunities of memory devices. In *Electron Devices Meeting (IEDM), 2016 IEEE International*, pages 1–1. IEEE, 2016.

[6] Y. Matsunobu. Myrocks: A space and write-optimized MySQL database. `code.facebook.com/posts/190251048047090/`.

[7] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani. Scaling Memcache at Facebook. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 385–398, Lombard, IL, 2013.