



# Improving Percona Server performance with Flashcache on the Virident *tachION* Drive

A Percona White Paper

By Vadim Tkachenko and Baron Schwartz<sup>1</sup>

## Abstract

Using flash storage as a second-level cache for MySQL, between disk and main memory, reduces I/O latency while avoiding the size and cost limitations of storing the data on flash alone. The most popular implementation of this approach is Flashcache, a Linux kernel module created by Facebook. Flashcache creates a block device backed by both flash and traditional disk storage. However, many system architects are unsure whether this technique is production-ready, or how to choose and size the flash storage device. Another common question is how much performance improvement to expect, and how to characterize the performance of such a system as the data and cache sizes vary relative to each other. This white paper presents the results of our benchmarks with Percona Server and the Virident *tachION* PCI-E storage device. We explain when it is cost-effective to use Flashcache, we show the performance characteristics of the hybrid storage, and we develop guidelines for selecting and deploying the components.

## 1 Using Flash Storage with MySQL

Flash is now widely accepted as an enterprise-ready storage solution for MySQL, in both the SSD (SATA solid-state drive) and PCI-E form factors. Flash offers faster random access than traditional spinning-disk storage, because reading or writing to a random location does not require a disk seek. This is very important for database performance, because many types of database operations require random access to data.

Our previous research has shown that it is possible to achieve significant performance improvements with MySQL and flash storage. In particular, we used Percona Server, our own variant of the MySQL server, and the Virident *tachION* PCI-E solid-state storage device. We demonstrated that this combination enables significant server consolidation, avoiding the need to shard (partition) the database across multiple servers, and additionally making it possible to colocate many server instances on a single physical machine for better resource utilization.

The net effect is that flash storage gives system architects more choices for deploying MySQL

databases.<sup>2</sup> However, flash is still relatively costly, and storage capacities are not as large as traditional disks. As a result, it is not always possible to use flash storage for a database. The database's on-disk footprint can exceed the storage capacity of available devices, the cost can be too high, or both.

## 2 Using Flash as a Cache

The price and capacity limitations of flash storage have prompted the introduction of a hybrid approach: using a flash device as a cache in front of a larger storage device based on spinning disks. This is not a new idea, nor is it exclusive to MySQL or even to databases in general. For example, in 2008 Brendan Gregg explained how the ZFS filesystem's level 2 cache, known as the L2ARC, can be placed onto a flash device as an intermediate cache between DRAM and spinning disk.<sup>3</sup> Also in 2008, Adam Lenth foresaw<sup>4</sup> flash's broad usage as a new tier in the storage hierarchy.<sup>5</sup>

The operating principle of the hybrid flash cache is the same as all caches: access to a typical dataset is non-uniform, and only a subset of the data is accessed frequently. This is known as *locality of refer-*

<sup>1</sup>Thanks to Mark Callaghan (Facebook) for his comments on this paper.

<sup>2</sup>See *Scaling MySQL Deployments With Percona Server and Virident tachION Drives* and *Scaling MySQL With Virident Flash Drives and Multiple Instances of Percona Server*

<sup>3</sup>See <http://blogs.oracle.com/brendan/entry/test>

<sup>4</sup>See <http://mags.acm.org/communications/200807/?pg=49>

<sup>5</sup>See [http://en.wikipedia.org/wiki/Memory\\_hierarchy](http://en.wikipedia.org/wiki/Memory_hierarchy)

*ence*. This subset can be kept in a smaller, faster, and typically more expensive tier of storage. Flash storage fits perfectly in the memory hierarchy between DRAM and spinning disk, with sub-millisecond latency.

Facebook's Flashcache<sup>6</sup> uses flash storage as a non-volatile (persistent) storage device. It was specifically designed as a secondary cache for MySQL's InnoDB storage engine, but is application-agnostic and can be used for many types of applications. It is implemented with Linux's Device Mapper (dm), so it appears to be a regular block device. When the application reads from the block device, the kernel module searches for the block in the flash cache. If not found, it is a cache miss; the read must be serviced from the spinning disk, and the cache is populated with the block. Writes are performed to the flash device, and a background process eventually writes the changed blocks back to the disks, making Flashcache a *write-back* cache. Blocks are evicted from the flash storage as needed, and replaced. The replacement policy can be either FIFO (first-in, first-out) or LRU (least-recently used).

The result is that a system architect can create a large device based on spinning disks, and use a smaller flash device as a transparent cache in front of it. Both of these devices are abstracted and hidden within the block device created by the Flashcache kernel module. In principle, the resulting device will have the best characteristics of both types of storage: the large storage capacity of the spinning disks, and the low access latencies of the flash storage device. This directly addresses the cost and capacity limitations of flash storage, making it possible to have large, fast disk volumes at a lower cost than possible with flash storage alone.

### 3 Is Flashcache Ready for Production?

Flashcache is relatively new technology, and the implementation has some limitations, such as not protecting against torn (partially written) pages. However, this is an application-specific issue, because InnoDB has its own protection against partial page writes, the so-called *doublewrite buffer*. Therefore,

<sup>6</sup>See <https://github.com/facebook/flashcache>

<sup>7</sup>The buffer pool is InnoDB's in-memory data and index cache.

Flashcache may not be suitable for every use case. However, after evaluating it for over a year and contributing to its development, it is our opinion that it is production-ready for InnoDB data. In addition, Facebook's production deployment of Flashcache appears to be successful.

### 4 Benchmark Goals and Method

We were retained by Virident to evaluate Flashcache's performance, and develop guidelines for cost-effective deployment. Our goals were to determine under what conditions Flashcache offers the best price-to-performance ratio, derive guidelines for sizing, and offer insight into the performance of a Flashcache device as compared to disk alone.

We adopted the following approach to evaluate the combination of Flashcache and the Virident *tachION* drive. We benchmarked a database instance while varying the ratio of the data size to the InnoDB buffer pool size,<sup>7</sup> and the ratio of the data size to the Flashcache size. We compared this to benchmarks on the same disks without using the flash device as a cache, and to the flash device directly.

Our intuition was that the system would perform best when the *working set*, i.e. the actively accessed set of data, fit completely on the flash device. This should help prevent frequent cache misses, avoiding the additional latency caused by accessing the data on the traditional disk drives. However, in order to validate our assumption, we needed to define the working set precisely.

In real systems, the working set is not a simple concept. It can be considered to be a combination of an observation interval and a quantile. For example, if 95% of data accesses inside the database within one hour involve a given set of database pages, then the 95th percentile one-hour working set can be said to equal those pages. Note that the working set size is not influenced by cache sizes; the working set is the same set of pages, whether they reside in memory or on disk.

Real systems tend to have dynamically changing behavior, which we can expect to change the working

set over time. This makes realistic evaluation difficult. For our benchmarks, therefore, we chose a simpler and more static workload. This is the sysbench uniform OLTP benchmark, which accesses rows at random, with a uniform distribution; that is, each row is equally likely to be accessed. This distribution means that there will be no locality of reference, and all of the data pages will be accessed frequently. Thus, the working set of data is identical to the whole dataset over any significant quantile and period of time. Although these benchmarks will not match real-world systems, they make it possible to compare various Flashcache configurations.

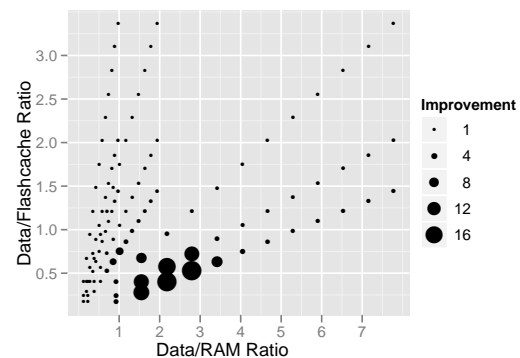
We must stress that the absolute performance measurements from these benchmarks are unrelated to any real-life system. You should use these results only as a means of comparing the various configurations and workloads to each other, as presented in this paper. You should not compare the results against any other benchmarks.

We ran benchmarks with twelve different datasets, varying from 60 million rows to 500 million rows, in increments of 40 million rows. In the sample dataset, 100 million rows is approximately 22GB of data. We tested three buffer pool sizes: 13GB, 52GB, and 104GB. This emulates roughly the buffer pool sizes of server hardware with 16GB, 64, and 128GB of DRAM, so it is a proxy for benchmarking on servers with those amounts of memory. We used the following Flashcache sizes: 30GB, 50GB, and 70GB. Finally, we ran both a read-only workload, and a mixed read-write workload that does not change any data permanently. This ensures that later benchmark runs are not influenced by earlier ones. Thus, we benchmarked a total of 216 different scenarios.

## 5 Read-Only Benchmark Results

The following chart displays the read-only benchmark results. The horizontal axis shows the ratio of the data size to the memory size, as configured through the buffer pool. For example, when the data-to-memory ratio is two, then the dataset size is twice as large as the buffer pool size, and when the data-to-memory ratio is less than one, the data fits completely into the buffer pool. The vertical axis shows the ratio of the data size to the Flashcache

size. The performance result is represented by the size of the point on the graph, as a ratio of performance relative to the RAID array with no Flashcache device. The bigger the point, the better the improvement relative to RAID-only.



The chart is different from benchmark results readers may be used to viewing. We formatted it in this way to avoid overwhelming readers with many charts. It is easiest to read the chart by beginning at the origin in the bottom left, where the data is smaller than both the buffer pool and the Flashcache device.

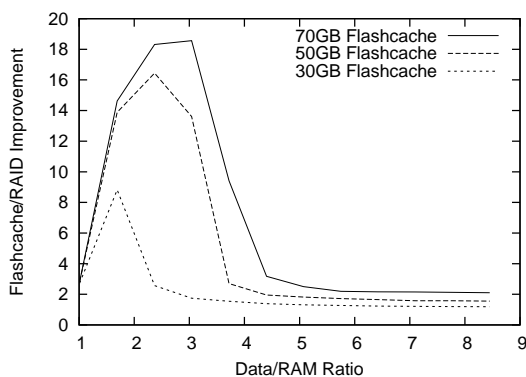
When the data fits into the buffer pool, it is easy to predict that there will be no buffer pool misses at all, and hence no benefit from Flashcache. This is indeed the case, which is why the points with a data-to-memory ratio less than one show no improvement. As we move horizontally to the right on the chart, the data grows larger than the buffer pool, but still fits inside the Flashcache. The large points on the chart show that Flashcache provides a very good performance improvement over RAID-only storage in these cases—up to an 18x boost, in fact.

In the other dimension, as we increased the data size relative to the Flashcache, we again saw the expected result: as long as the dataset fits within the flash device, access to it is much faster than access to the RAID device. However, as soon as the dataset no longer fits in the cache, the performance advantage relative to the RAID array disappears quickly.

How should we interpret these results? Consider them in the context of the storage hierarchy. When data fits in the buffer pool, access is very fast. When

it no longer fits in the buffer pool, buffer pool misses occur, and the data must be read from the next storage tier—the flash device. This is fast as well. But when that device is not large enough to hold the dataset, and cache misses occur, then the system begins to read from disk, which is slow. There is marginal improvement over disk alone even when the data is larger than the Flashcache, because some of the reads are still cache hits, but the improvement is not very large.

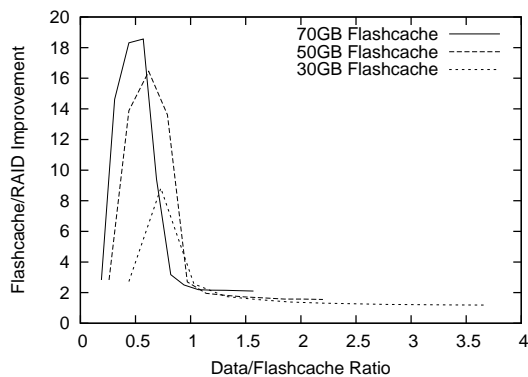
Another way of looking at the same data is to plot the improvement relative to RAID versus the fit in the buffer pool:



The chart shows results for the read-only workload with a 13GB buffer pool size. The vertical axis is performance improvement relative to the RAID device, and the horizontal axis is the ratio of data size to buffer pool size. Again, we can see that the Flashcache device extends the region of good performance significantly beyond the buffer pool size. Unsurprisingly, the larger the Flashcache device, the more performance we achieve from large datasets.

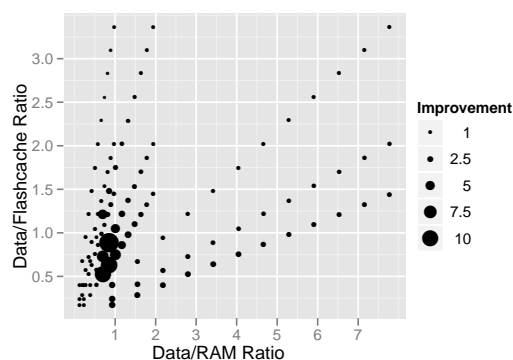
This chart shows only a small subset of the benchmarks we ran, which is why we presented a consolidated chart instead of individual ones. If we showed separate charts for all of the results, it would be too difficult to understand them.

If we change the horizontal axis and plot the performance improvement versus RAID relative to the Flashcache device's size, instead of the buffer pool size, we obtain the following chart. This shows that we can essentially use the flash device as a secondary buffer pool, giving very good performance as long as the working set fits into the flash device.



## 6 Read-Write Results

The following chart shows the results for the read-write benchmark. The axes have exactly the same meaning as in the read-only benchmarks:



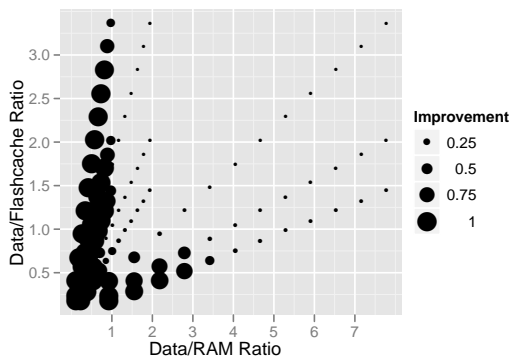
In contrast to the read-only case, the region where Flashcache provides a significant improvement over RAID alone in the read-write workload is smaller. Flashcache gives a big benefit over RAID when the working set fits in the flash card, and is almost the same size as the buffer pool. When the working set is smaller, the Flash and RAID devices are able to keep up with the workload equally well. When the working set is larger, Flashcache is not significantly faster than the RAID controller.

## 7 Performance Relative to Pure Flash

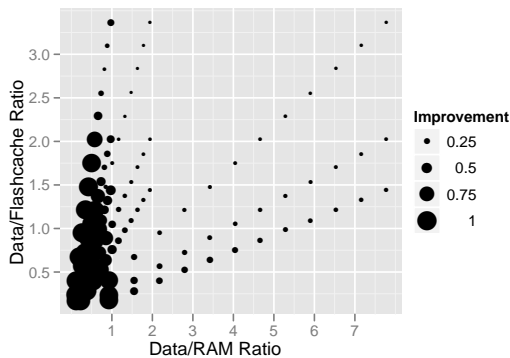
In addition to comparing Flashcache's performance to the RAID volume, we benchmarked with the data residing directly on the Virident *tachIO*n drive. We wanted to answer the following question: under

what circumstances does Flashcache perform nearly as well as the *tachION* drive alone? System architects might want to consider this separately, to help decide on the most economical way to deploy flash storage.

We again consolidated results for the read-only and read-write benchmarks onto a single chart each, with the same axes. However, we changed the meaning of the points. Each point represents Flashcache's performance relative to the *tachION* drive, instead of the RAID volume. Here are the results for the read-only benchmark:



And the results for the read-write benchmark:



From these plots we can see that in the cases where Flashcache improves performance over a RAID volume of spinning disks, it also delivers nearly as good performance as storing the data on pure flash. This supports our theory that when the working set fits into the flash device, Flashcache can deliver the speed of flash storage with the large capacity of traditional disks.

We need to perform more research on how Flashcache performs when the working set is larger than

both the buffer pool and the Flashcache device. As you can see, the lower right-hand corner of the charts is empty, and it would be interesting to know how Flashcache performs in that region. When we chose the results with the best improvement over RAID storage alone, where the working set is at least 50% larger than the buffer pool, we found that the improvement over RAID is not as dramatic as the read-only improvement:

Data/RAM	Data/Flashcache	Improvement
1.55	0.29	2.56
1.55	0.40	2.32
1.64	1.21	2.02
2.18	0.40	2.47
2.18	0.57	2.03
2.80	0.52	2.48
3.42	0.63	2.45
4.05	0.75	2.38
4.66	0.86	2.09

We have too few such data points to draw firm conclusions, and this is a topic for future research.

## 8 Conclusions

Based on the benchmarks we performed, and our analysis of the results, we offer the following guidelines for those considering Flashcache as storage for MySQL database servers:

- When the workload is read-only and the working set fits into the flash drive, Flashcache provides a significant improvement in performance over a RAID device—nearly the same performance as the Virident *tachION* drive alone. In our tests, we demonstrated that Flashcache can increase by approximately 2x-4x the working set size that the server can handle. When the working set exceeds the size of the flash card by more than 20% to 40%, performance decreases to only slightly faster than RAID alone, and Flashcache may not be economical.
- When the workload has a significant random write component, Flashcache provides a significant performance improvement over RAID when the working set fits into the buffer pool. However, we need more benchmarks to assess additional use cases.

### About Percona

Percona is the oldest and largest independent provider of commercial support, consulting, training, and engineering services for MySQL databases and the LAMP stack. You can contact us through our website at <http://www.percona.com/>, or to call us. In the USA, you can reach us during business hours in Pacific (California) Time, toll-free at 1-888-316-9775. Outside the USA, please dial +1-208-473-2904. You can reach us during business hours in the UK at +44-208-133-0309.



### About Virident Systems

Virident Systems builds enterprise-class solutions based on Flash and other storage-class memories (SCM). These disruptive technologies will revolutionize the data center and cloud computing by solving performance, reliability, and serviceability problems that further compound in large-scale deployment of SSDs in current environments. Visit <http://www.virident.com> for more information, or call us at (408) 503-0100 during business hours in Pacific (California) Time.



### About Percona Server

**Percona Server** is an enhanced, high-performance version of the world's most popular open-source database, MySQL. MySQL is used by many of the world's largest websites, including Facebook, Flickr, and YouTube. MySQL is also deployed widely in industries such as financial services, government, education, pharmaceuticals, and telecommunications. Its simplicity, reliability, and ease of use make it cost-effective to manage, and because it is open-source, it can be used without license fees. Percona Server is derived from the MySQL database, to which it adds features such as enhanced monitoring and configurability. Percona Server offers much faster and more consistent performance than the standard MySQL server. Percona also provides a free hot-backup program, **Percona XtraBackup**.

*Percona, XtraDB, and XtraBackup are trademarks of Percona Inc. InnoDB and MySQL are trademarks of Oracle Corp. Virident and tachION are trademarks of Virident Systems Inc.*