Achieving Both Performance Isolation and Uniform Lifetime for Virtualized SSDs with FlashBlox*

Jian Huang† Anirudh Badam Laura Caulfield

Suman Nath Sudipta Sengupta Bikash Sharma* Moinuddin K. Qureshi§

†UIUC §Georgia Tech Microsoft *Facebook

Summary. A longstanding goal of SSD virtualization has been to provide performance isolation between multiple tenants sharing the device. Virtualizing SSDs, however, has traditionally been a challenge because of the fundamental tussle between resource isolation and the lifetime of the device – existing SSDs aim to uniformly age all the regions of flash and this hurts isolation. We propose utilizing flash parallelism to improve isolation between virtual SSDs by running them on dedicated channels and dies. Furthermore, we offer a complete solution by also managing the wear. We propose allowing the wear of different channels and dies to diverge at fine time granularities in favor of isolation and adjusting that imbalance at a coarse time granularity in a principled manner. Our experiments show that the new SSD wears uniformly while the 99th percentile latencies of storage operations in a variety of multi-tenant settings are reduced by up to 3.1x compared to software isolated virtual SSDs.

Motivation. SSDs have become indispensable for large-scale cloud services as their cost is fast approaching to that of HDDs. They out-perform HDDs by orders of magnitude, providing up to 5000x more IOPS, at 1% of the latency. The rapidly process shrinking and die stacking technologies have boosted the capacity and bandwidth of SSDs by increasing the number of chips. However, the limitations of SSDs' management algorithms have hindered these parallelism trends from efficiently supporting multiple tenants on the same SSD.

Tail latency of SSDs in multi-tenant settings is one such limitation. Cloud storage and database systems have started colocating multiple tenants on the same SSDs [6] which further exacerbates the already well known tail latency problem of SSDs [2]. The cause of tail latency is the set of complex flash management algorithms in the SSD's controller, called the Flash Translation Layer (FTL). The fundamental goals of these algorithms are decades-old and were meant for an age when SSDs had limited capacity and little parallelism. The goals were meant to hide the idiosyncrasies of flash behind a layer of indirection and expose a block interface. These algorithms, however, conflate wear leveling (to address flash's limited lifetime) and resource utilization (to exploit parallelism) which increases interference between tenants sharing an SSD.

While application-level flash-awareness [3] improves throughput by efficiently leveraging the device level parallelism, these optimizations do not directly help reduce the interference between multiple tenants sharing an SSD. These tenants cannot effectively leverage flash parallelism for isolation even when they are individually flash-friendly because FTLs hide the parallelism. Newer SSD interfaces [4, 5] that propose exposing raw parallelism directly to higher layers provide more flexibility in obtaining isolation for tenants but they complicate the implementation of wear-leveling mechanisms across the different units of parallelism.

FlashBlox Solution. In this work, we propose leveraging the inherent parallelism present in today's SSDs to increase isolation between multiple tenants sharing an SSD. We propose creating virtual SSDs that are pinned to a dedicated number of channels and dies depending on the capacity and performance needs of the tenant. The fact that the channels and dies can be more or less operated upon independently helps such virtual SSDs avoid adverse impacts on each other's performance. Based on these insights, we provide new abstraction for hardware isolation with virtualizing the physical SSD into different types of virtual SSDs that have different levels of isolation and allocation granularities of storage bandwidth and capacity. Such an abstraction fits for the existing pay-as-you-go model in cloud storage services, in which the application instances are configured with different levels of performance and capacity guarantees.

However, different workloads can write at different rates and in different patterns, this could age the channels and dies at different rates. For instance, a channel pinned to a TPC-C database instance wears out 12x faster than a channel pinned to a TPC-E database instance, reducing the SSD lifetime dramatically. This non-uniform aging creates an unpredictable SSD lifetime behavior that complicates both provisioning and load-balancing aspects of data center clusters. To address this problem, we propose a two-part wear-leveling model which balances wear within each virtual SSD and across virtual SSDs using separate strategies. Intra-virtual SSD wear is managed by leveraging existing SSD wear-balancing mechanisms while inter-virtual SSD wear is balanced at *coarse-time granularities* to reduce interference by using new mechanisms. We control the wear imbalance between virtual SSDs using a mathematical model and show that the new wear-leveling model ensures near-ideal lifetime for the SSD with negligible disruption to tenants. Flash-

*FlashBlox has been published at FAST'17 [1].

Blox uses a simple yet effective wear-leveling scheme:

Periodically, the channel that has incurred the maximum wear thus far is swapped with the channel that has the minimum rate of wear. We analytically derive the minimum necessary swapping frequency:

Let σ_i denote the wear (total erase count of all the blocks till date) of the i^{th} channel. $\xi = \sigma_{max}/\sigma_{avg}$ denotes the wear imbalance which must not exceed $1 + \delta$; where $\sigma_{max} = Max(\sigma_1, ..., \sigma_N)$, $\sigma_{avg} = Avg(\sigma_1, ..., \sigma_N)$, N is the total number of channels, and δ measures the imbalance.

SSDs are provisioned with a target erase workload and we analyze for the same – let's say M erases per week. We mathematically study the wear-imbalance vs. frequency of migration (f) tradeoff and show that manageable values of f can provide acceptable wear imbalance where ξ comes below $1 + \delta$ after αL weeks, where α is between 0 and 1. For an SSD with N channels, the wear imbalance of ideal wear-leveling is $\xi = 1$, while the worst case workload for FlashBlox gives a $\xi = N$: $\sigma_{max}/\sigma_{avg} = M * time/(M * time/N) = N$ before any swaps. A simple swap strategy of cycling the write workload through the N channels (write workload spends $1/f$ weeks per channel) is analyzed. Let's assume that after K rounds of cycling through all the channels, $KN/f \geq \alpha L$ holds true – that is αL weeks have elapsed and ξ has become less than $1 + \delta$ and continues to remain there. At that very instant ξ equals 1. Therefore, $\sigma_{max} = MK$ and $\sigma_{avg} = MK$, then after the next swap, $\sigma_{max} = MK + M$ and $\sigma_{avg} = MK + M/N$. In order to guarantee that the imbalance is always limited, we need:

$\xi = \sigma_{max}/\sigma_{avg} = (MK + M)/(MK + M/N) \leq (1 + \delta)$

This implies $K \geq (N - 1 - \delta)/(N\delta)$ which is upper bounded by $1/\delta$. Therefore, to guarantee that $\xi \leq (1 + \delta)$, it is enough to swap $NK = N/\delta$ times in the first αL weeks. This implies that, over a period of five years, if α were 0.9 then a swap must be performed once every 12 days ($= 1/f$) for a $\delta = 0.1$ ($N = 16$). This also implies that $\frac{2}{16}^{th}$ of the SSD is erased to perform the swap once every 12 days, which is negligible compared to the 3,000–10,000 cycles that typical SSDs have. For realistic workloads that do not have such a skewed write pattern with a constant bandwidth, swaps can be adaptively performed according to workload patterns to reduce the number of swaps needed while maintaining balanced wear (see details in our FAST'17 paper [1]).

Contributions. More specifically, this work makes the following contributions:

- We present the FlashBlox system using which tenants can share an SSD with minimal interference by working on dedicated channels and dies. FlashBlox exploits the internal flash parallelism to achieve the hardware isolation for multi-tenant applications.
- We present a new wear-leveling mechanism that allows measured amounts of wear imbalance to obtain better performance isolation between such tenants.
- We present an analytical model and a system that control the wear imbalance between channels and dies, so that they age uniformly with negligible interruption to the tenants.

We design and implement FlashBlox and its new wear-leveling mechanisms inside an open-channel SSD, and demonstrate benefits for the multi-tenant storage workloads in Microsoft datacenter: the new SSD delivers up to 1.6x better throughput and reduces the 99th percentile latency by up to 3.1x. Our wear leveling mechanism provides 95% of the ideal SSD lifetime even in the presence of adversarial write workloads that execute all the writes on a single channel while only reading on other channels.

Novelty and Impact. FlashBlox challenges the conventional wisdom for flash management in SSDs. Existing SSDs aim to uniformly age all the regions of flash, which hurts isolation. FlashBlox utilizes the increasing chip-level parallelism in flash for hardware isolation and adjusts the wear imbalance at a more coarse time granularity in a principled manner for guaranteed SSD lifetime. The proposed wear-leveling mechanism decouples the wear-leveling and resource utilization to meet the ever-increasing demand on storage performance while providing near-ideal device lifetime.

FlashBlox has had an impact on the storage system in Microsoft data centers, and it is being transferred into Microsoft Azure Storage. Our system prototype implemented with a real open-channel SSD has demonstrated its benefits for data center applications such as Bing search, database, and Azure storage workloads. Moreover, the scalability of FlashBlox matches with the virtual machine and container scalability in data centers. Taking a typical Microsoft server for example, it can support hundres of virtual SSDs.

References

[1] FlashBlox: Achieving Both Performance Isolation and Uniform Lifetime for virtualized SSDs (FAST'17). https://www.usenix.org/system/files/conference/fast17/fast17_huang.pdf.

[2] Mingzhe Hao, Gokul Soundararajan, Deepak Kenchammana-Hosekote, Andrew A. Chien, and Haryadi S. Gunawi. The Tail at Store: A Revelation from Millions of Hours of Disk and SSD Deployments. In *Proc. FAST'16*, Santa Clara, CA, February 2016.

[3] Changman Lee, Dongbo Sim, Joo-Young Hwang, and Sangyeun Cho. F2FS: A New File System for Flash Storage. In *Proc. FAST'15*, Santa Clara, CA, February 2015.

[4] Sungjin Lee, Ming Liu, Sangwoo Jun, Shuotao Xu, Jihong Kim, and Arvind. Application-Managed Flash. In *Proc. FAST'16*, Santa Clara, CA, February 2016.

[5] Jian Ouyang, Shiding Lin, Song Jiang, Yong Wang, Wei Qi, Jason Cong, and Yuanzheng Wang. SDF: Software-Defined Flash for Web-Scale Internet Storage Systems. In *Proc. ACM ASPLOS*, Salt Lake City, UT, March 2014.

[6] Ning Zhang, Junichi Tatemura, Jignesh M. Patel, and Hakan Hacigumus. Re-evaluating Designs for Multi-Tenant OLTP Workloads on SSD-based I/O Subsystems. In *Proc. SIGMOD'14*, Snowbird, UT, June 2014.