

# Pathological Behavior of SSDs and Application in HPC Storage

Youngjae Kim<sup>1</sup>, Junghee Lee<sup>2</sup>, and Galen M. Shipman<sup>1</sup>

<sup>1</sup>Oak Ridge National Laboratory, <sup>2</sup>Georgia Institute of Technology  
{kimy1, gshipman}@ornl.gov, jlee35@ece.gatech.edu

**Abstract**—Unlike hard drives, flash devices use out-of-update operations and require a garbage collection (GC) process to reclaim invalid pages to create free blocks. The GC process is a major cause of poor performance in SSDs. This GC process results in the I/O performance of flash devices being highly dependent on I/O workload characteristics, slowing down the SSDs in particular for bursty, write dominant workloads. From this work we not only empirically examine GC process using real commercial-off-the-shelf (COTS) SSDs but also define the pathological behavior of SSDs. In order to mitigate the SSD slowdown due to GC process, we propose a preemptive GC (PGC) scheme that gives a high priority to pending I/O requests in the queue by preempting on-going GC process.

## I. INTRODUCTION

Solid state disks (SSD), especially NAND Flash memory-based SSDs, are a leading media in storage systems. Recently several developments have been made to employ SSDs for enterprise-scale and HPC storage systems. NAND Flash memory technology offers a number of benefits over conventional hard disk drives (HDDs), such as lower power consumption, lighter weight, higher resilience to external shocks, ability to sustain hotter operating regimes, and lower I/O access times. Thus, SSDs are being seriously considered as a replacement of storage drive in back-end storage systems, however, there are challenges that SSDs should overcome.

SSD performance is highly dependent on I/O access patterns and SSD can suffer from high latency for bursty and write-dominant workloads. Due to SSD’s out-of-update operations, an SSD must clean stale data for providing free space. This cleaning process, known as garbage collection, causes unwanted delays in performing reads and writes that store the same target space. Fragmentation caused by small random writes increases the GC overhead. This overhead is directly related to the frequency of copy operations for non-stale data pages and block erase operations. In particular, HPC file systems are stressed with frequent writes, checkpointing and journal updates. In an effort to study the benefits of using SSDs for HPC systems, we characterized the workloads and modelled the I/O access patterns on an HDD based HPC storage platform - Spider.

Spider is a center-wide parallel file system that connects file and storage systems for Jaguar XT4, the Cray XT5 simulation platform at Oak Ridge National Laboratory (ORNL). The Spider storage system has been provisioned with 13,440 hard disk drives to support over 2 petaflops of computing infrastructure. Currently, Spider has employed a RAID (Redundant Array of Independent Disk) level 6 scheme in order to fulfill the need to provide a highly reliable and available storage system, as well as high I/O throughput.

In our study of I/O workload characterization of the Spider storage, we observed the peak read and write bandwidths can reach around 90GB/s and 65GB/s (from half of our total capacity) respectively [1]. Their bandwidth distributions are representative of a heavy long-tail distribution, and we saw these trends are observed across all 48 RAID controllers. Moreover, I/O requests to the RAID controllers are bursty and show heavy-tail distribution in their inter-arrival times.

SSDs are proven to have better throughput and access latency than HDDs, however, SSDs should be designed such that they can provide sustained bandwidth in spite of GCs. In this work, we (i) identify such a pathological behavior of SSDs from empirical experiments using commercial SSDs and (ii) provide a novel solution to mitigate the performance degradation of SSDs due to GCs.

## II. PATHOLOGICAL BEHAVIOR OF SSDS

One of the main shortcomings of SSDs is the slowdown during the garbage collection (GC) process that is hastened by small, random writes. This slowdown can even further impact future incoming requests, we term this “*pathological behavior*” of an SSD by delaying I/O request services [2]. In order to empirically observe the effect of GC, we performed a series of experiments using various commercially-off-the-shelf (COTS) SSDs. All experiments were performed on a single server with 24 GB of RAM and an Intel Xeon Quad Core 2.93GHz CPU. The operating system was Linux with a Lustre-patched 2.6.18-128 kernel. The *noop* I/O scheduler with FIFO queuing was used.

TABLE I  
CHARACTERISTICS OF SSDS USED IN OUR EXPERIMENTS.

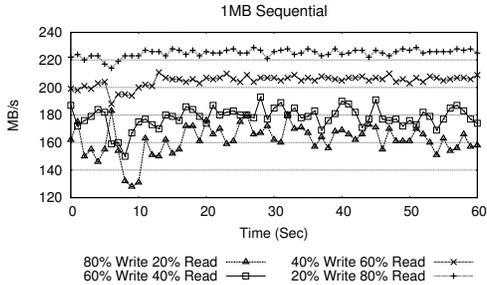
Label	SSD(A)	SSD(B)
Company	Super-Talent	Intel
Model	FTM28GX25H	SSDSA2SH064G101
Type	MLC	SLC
Capacity (GB)	120	64
Erase (#)	10-100K	100K-1M
Power (W)	1-2	1-2

We examined two representative SSDs that are detailed in Table I. We selected the Super Talent 128 GB SSD as a representative of multi-level cell (MLC) SSDs and the Intel 64 GB SSD as a representative of single-level cell (SLC) SSDs. We denote the SuperTalent MLC, and Intel SLC devices as SSD(A), and SSD(B) in the remainder of this study, respectively. We examined the I/O bandwidth of individual COTS SSDs for write-dominant workloads. To measure the I/O performance we used a benchmark that exploits the *libaio* asynchronous I/O library on Linux.

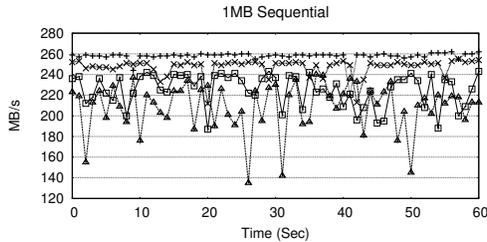
### A. Performance Anomaly on SSDs

In Figure 1(a)(b), we examine the large sequential I/O bandwidth responses of individual SSDs in time series. We varied the percentage of writes in workloads between 20% and 80% in increasing steps of 20%. We measured I/O bandwidth in one second intervals.

For write-dominant workloads, we observe that the bandwidth fluctuates widely due to excessive GCs. For example, the SSD(A) I/O throughput drops below 180MB/s at the 6<sup>th</sup> and 7<sup>th</sup> seconds under an 80% write workload. However, I/O throughput drops below 160MB/s for the 8<sup>th</sup> second and then drops further to 130MB/s in the next 3 seconds. Overall SSD(B) shows higher bandwidth than SSD(A). Also, SSD(B) has a higher variance than SSD(A). For instance, SSD B’s I/O throughput reached 240MB/s at the peak and dropped to 140MB/s (at 25<sup>th</sup> to 27<sup>th</sup> seconds). As we increased the amount



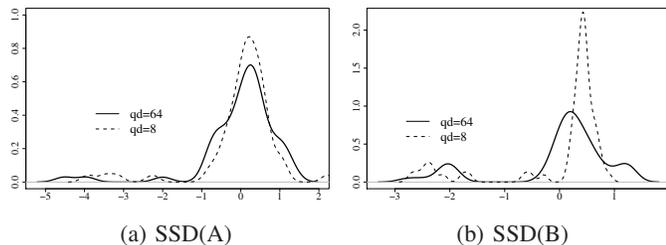
(a) SSD(A)



(b) SSD(B)

Fig. 1. Pathological behavior of single SSDs.

of reads in the workloads from 20% to 80%, we observed that SSD(A)'s and B's I/O throughput increased around 50% and 18%, respectively.



(a) SSD(A)

(b) SSD(B)

Fig. 2. Variability of bandwidth.  $qd$  denotes queue depth. High  $qd$  means requests are bursty and intense in their arrival rate.

### B. Performance Variability of SSDs:

Figure 2 illustrates the impact of GC on I/O bandwidth. In order to compare the bandwidth variability of individual SSDs for different arrival rates of requests, we measured I/O bandwidth for 512KB write requests by varying I/O queue depth (QD). We normalized the measured bandwidth with a Z-transform and plotted density functions with curve-fitting techniques. We observed that the performance variability increases with respect to the arrival rate of requests. This can be interpreted as for a workload with bursty arrival I/O request pattern, SSD is not able to guarantee bandwidth and it can be attributed to the GC process.

### III. MITIGATING SSD SLOW-DOWN DUE TO GC PROCESS

In order to not stop servicing incoming requests to SSDs, we allow preemption of GC, which incurs an extra unnecessary context overhead. Thus, we design and develop an efficient preemptive GC scheme, which can permit GC preemption at certain points.

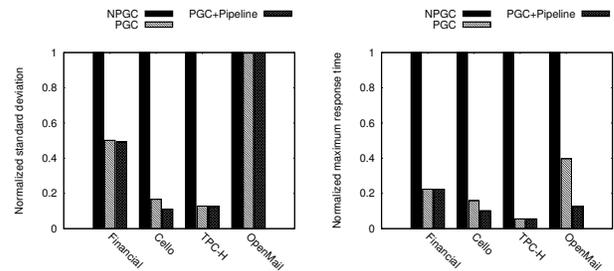
A typical garbage collection process is composed of a series of page read and write operations. Once a block is selected, victim block for GC, all the valid pages on the block should be moved into a free block and then, the victim block is erased. The moving operation of a valid page is divided into a page read, data transfer in the page, page write, and finally, meta-data update operations. If the free block is in the same plane to which the valid page moves, then, the data transfer operation

could be omitted. We identify two possible preemption points among the series of operations related to page movement during GC – within a page movement, and between a series of page movements. With this identification of preemption points, we insert the incoming requests (that arrive during GC process) comprised of page read and write operations similar to the operations during GC appropriately without page read-after-write (RAW) dependency violation.

We can further optimize the performance by merging incoming page read and write requests into GC process, effectively folding duplicate page operations into one. If incoming page requests matches a page that is touched by GC then, it is combined to one page requests. Moreover, we also discuss pipelining technique of incoming requests with GC. If the incoming request is on the page different from the page operation of GC then they can be pipelined, reducing the response time of the incoming request.

## IV. EXPERIMENTAL RESULTS

We developed our preemptive and GC optimization schemes on Microsoft SSD simulator and simulated a large block 32GB NAND flash based SSD. We used four commercial I/O traces – Financial, Cello99, TPC-H, and OpenMail.



(a) Variance of Response Times (b) Maximum Response Time

Fig. 3. Performance improvements by preemptive garbage collector for realistic server workloads. In legends, *NPGC*, *PGC*, and *PGC+Pipeline* respectively denote non-preemptive GC, preemptive GC, and preemptive GC with pipelining technique.

In Figure 3, we present the performance (in terms of system response time) and its variance. Figure 3(a) shows a big improvement for response time variance. PGC reduces the performance variability by 49.82% and 83.30% for each of workloads. In addition to the improvement of performance variance, from Figure 3(b) we observe that PGC can further reduce the maximum response time of NPGC by 77.59% and 84.09% for Financial and Cello traces.

## V. CONCLUSION

From this work, we argue that SSDs possess bandwidth variability problems due to non-preemptive ongoing GC process, which can severely slow down SSDs. The details of this study can be found in [2], [3].

## REFERENCES

- [1] Y. Kim, R. Gunasekaran, G. M. Shipman, D. Dillow, Z. Zhang, and B. W. Settlemyer, "Workload characterization of a leadership class storage," in *5th Petascale Data Storage Workshop Supercomputing '10 (PDSW'10)*, November 2011.
- [2] Y. Kim, S. Oral, D. A. Dillow, F. Wang, D. Fuller, S. Poole, and G. M. Shipman, "An empirical study of redundant array of independent solid-state drives (RAIS)," in *Technical Report, ORNL/TM-2010/61, Oak Ridge National Laboratory, National Center for Computational Sciences*, March 2010.
- [3] J. Lee, Y. Kim, G. M. Shipman, S. Oral, J. Kim, and F. Wang, "A semi-preemptive garbage collector for solid state drives," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (To Appear)*, April 2011.