Performance Characterization of Hyperscale Applications on NVMe SSDs

Qiumin Xu*, Huzefa Siyamwala‡, Mrinmoy Ghosh†, Manu Awasthi†,
Tameesh Suri†, Zvika Guz†, Anahita Shayesteh†, Vijay Balakrishnan†
*Univeristy of Southern California, ‡ San Jose State University, †Samsung Semiconductor Inc., Milpitas, CA
qiumin@usc.edu, mrinmoy.g@ssi.samsung.com

ABSTRACT

The storage subsystem has undergone tremendous innovation in order to keep up with the ever-increasing demand for throughput. NVMe based SSDs are the latest development in this domain, delivering unprecedented performance in terms of both latency and peak bandwidth. Given their superior performance, NVMe drives are expected to be particularly beneficial for I/O intensive applications in datacenter installations. In this paper we identify and analyze the different factors leading to the better performance of NVMe SSDs. Then, using databases as the prominent use-case, we show how these would translate into real-world benefits. We evaluate both a relational database (MySQL) and a NoSQL database (Cassandra) and demonstrate significant performance gains over best-in-class enterprise SATA SSDs: from 3.5× for TPC-C and up to 8.5× for Cassandra.

CATEGORIES AND KEYWORDS

Primary Classification: B. Hardware B.4 INPUT/OUTPUT AND DATA COMMUNICATIONS B.4.4 Performance Analysis and Design Aids** Subjects: Worst-case analysis** Keywords: SSDs;NVMe;Hyperscale Applications;NoSQL Databases;Performance Characterization

1. INTRODUCTION

The vast majority of today's web-based services require a datacenter installation in the background. There has been an explosion of the amount of data being stored, and an increasing demand for processing this volumes of data faster. To meet the ever-increasing performance demands, storage subsystems and data-storage devices have been evolving. Solid State Drives (SSDs) were the first technology leap: using traditional interfaces like Serial-ATA (SATA) and SAS, they provide significantly better performance than hard disk drives (HDDs). In the next technology leap, SSDs were connected to the host via PCIe interconnect to provide a higher bandwidth and lower latency interface. PCIe-based solutions provided higher performance, but their adoption was hindered by the lack of standardization and inter-vendor compatibility [3].

The NVMe standard was designed to enable fast and widespread adoption of PCIe SSDs. NVMe was architected from the ground up for non-volatile memory devices over PCIe, focusing on latency, performance, and request parallelism. Being a widely used standard which driver authors conform to, NVMe has contributed towards eliminating dependencies on a single vendor, allowing standard drivers to be written across different vendors, and increasing interoperability between cross-vendor PCIe based SSD storage solutions. The next section provides an overview of the benefits of the NVMe standard.

SIGMETRICS'15, June 15-19, 2015, Portland, OR, USA.
ACM 978-1-4503-3486-0/15/06.
http://dx.doi.org/10.1145/2745844.2745901.

2. BENEFITS OF NVME

NVMe devices are able to leverage scalable bandwidth advantages of the fast PCIe interconnect, while being able to build additional features like multipath IO and shared namespaces. The performance benefit of NVMe stems from three main factors:

Interface capabilities: PCIe supports much higher throughput compared to SATA: while SATA supports up to 600 MB/s, a single PCIe 3.0 lane allows transfers of up to 1GB/s. Typical PCIe based SSDs are ×4 PCIe generation 3 devices and support up to 4 GB/s.

Hardware data path: SATA drives connect to the system through the host bus: the data path traverses the AHCI driver, the host bus, and an AHCI host bus adapter (HBA). NVMe SSDs, on the other hand, connect directly through the PCIe root complex port. Therefore, an NVMe I/O access has a much shorter data path, resulting in a reduced overall access latency.

Simplified Software stack: In the conventional SATA I/O path, an I/O request arriving at the block layer is first inserted into a request queue. For most modern SSDs, the block layer applies the *noop* scheduler to the request queue. The *noop* scheduler is the simplest scheduler in the kernel that performs request merging whenever possible.

The NVMe standard introduced many changes to the I/O software stack. Figure 1 provides a high level representation of the differences between the I/O stacks of SATA and NVMe. An NVMe based request bypasses the conventional block layer request queue and instead implements a paired *Submission* and *Completion* queue mechanism [1]. The standard supports up to 64K I/O queues and up to 64K commands per queue. These queues are saved in host memory and are managed by the NVMe driver and the NVMe controller cooperatively: new commands are placed by the host software into the submission queue (SQ) and completions are placed by the NVMe controller into an associated completion queue (CQ). The controller fetches commands from the front of the queue, processes the commands, and rings the completion queue doorbell to notify the host. This asynchronous handshake reduces CPU time spent going through many different kernel modules and prevents resource sharing with other block devices.

[1]The 3.17 linux kernel has a re-architected block layer for NVMe drives. NVMe accesses no longer bypass the kernel. The results shown in this paper are for the 3.14 kernel.

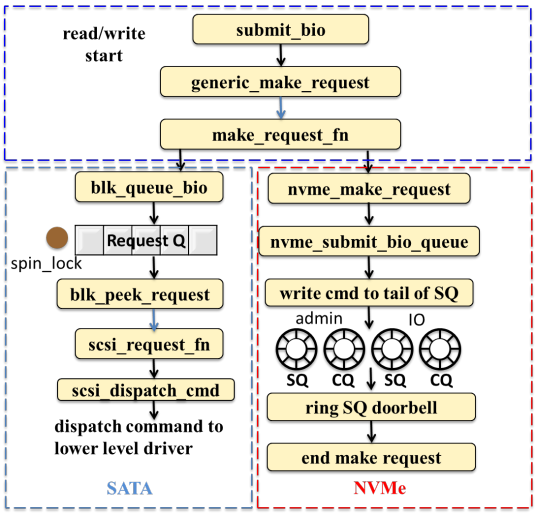

Figure 1: An illustration of the I/O software stacks of SATA SSD and NVMe SSD.

3. NVME DRIVES CHARACTERIZATION

To analyze the performance of NVMe drives as compared to SATA SSDs, we first perform a detailed latency characterization of the raw devices. We then demonstrate the benefits of the raw performance for different database applications. The `blktrace` tool is adequate for doing this analysis for SATA devices. In order to extract a detailed latency analysis for NVMe SSDs, we instrument the NVMe driver to add tracing events, and use the `blktrace` tool to capture timestamps of events during an NVMe I/O access. Using this tool in conjunction with `fio` for generating requests, we are able to estimate the time spent at different layers of the software stack during an I/O access.

The measured random read latency for the SATA SSD (Samsung 843T) we tested at low loads is 500μs. We found that system software accounts for a whopping 28% of the overall access latency, and that the block layer alone accounts for 5% of the total latency. In contrast, the measured latency for the tested NVMe SSD (Samsung XS1715) is only 110μs, and the total system software overhead in NVMe drives is significantly reduced to only 7.3%. Considering absolute latencies, our analysis indicates that NVMe SSD consumes 4.5× less time for random reads compared to the SATA SSD. By reducing system software inefficiencies, breaking interface bottlenecks, innovation in FTL design, and better NAND device characteristics, NVMe devices under test were able to sustain a maximum throughput of 750 K IOPS of 4 KB random read accesses , compared to 100K IOPS for SATA SSDs.

Next, we show how real world applications can benefit from the improved raw performance of NVMe SSDs. We use two example database workloads: (i) TPC-C [1] with MySQL, and (ii) YCSB with Cassandra [6]. TPC-C is an extensively used online transaction processing (OLTP) workload, composed of several operations that best represent a complex web based application environment. Cassandra is an open-source, production quality NoSQL data store. Cassandra provides a number of features similar to Google's BigTable [4] and is used in a number of production datacenters [2].

Figure 2 shows the relative client-side throughput gains when using Samsung XS1715 NVMe drives compared to a system that uses enterprise class Samsung 843T SATA SSDs. The metric of interest in case of databases is client side performance, measured in transactions per second. We show that NVMe delivers speedup of 3.5× for TPC-C (tpmC) using MySQL, and up to 8.5× for YCSB [5] using the Cassandra database.

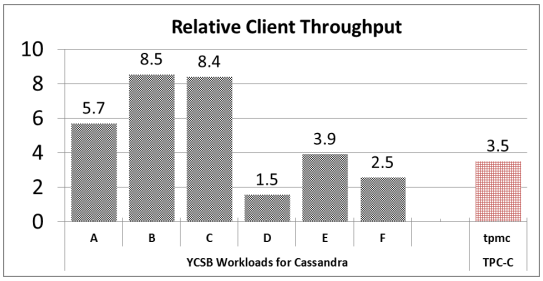

Figure 2: Relative performance improvement for different databases when using one NVMe compared to one SATA SSD.

In summary, the reduction of access latency, the asynchronism between request and completion queues, and the high bandwidth of the PCIe interface, all result in tremendous performance gains for real-world applications, especially for disk bound database applications.

4. CONCLUSIONS

Scale-out systems are driving the need for high performance storage solutions with high available bandwidth and low access latencies. To address this need, newer standards are being developed exclusively for non-volatile storage devices like SSDs. In this paper, we present a characterization of SSDs based on the NVMe standard for storage subsystems, and make a number of important contributions.

Firstly, we instrument the Linux NVMe driver and the popular `blktrace` tool and quantify the benefits of a leaner I/O stack. Secondly, using this instrumentation tool, we show that the NVMe access stack allows the I/O requests to bypass most of the legacy I/O layers, resulting in a 4.5× decrease in access latencies. Lastly, using databases as an example class of applications, we show that NVMe's hardware and software redesign of the storage subsystem translates into *real world* benefit for relational and scale-out database applications. In particular, we show that, as compared to a single SATA SSD, NVMe based SSDs can provide performance benefits of up to 8.5×.

5. REFERENCES

[1] "TPC-C Benchmark Standard Specification, Revision 5.11," http://www.tpc.org/tpcc/spec/tpcc_current.pdf, 2010.

[2] "Benchmarking Cassandra Scalability on AWS - Over a million writes per second," http://techblog.netflix.com/2011/11/benchmarking-cassandra-scalability-on.html, 2011.

[3] "NVM Express Explained," http://nvmexpress.org/wp-content/uploads/2013/04/NVMe_whitepaper.pdf, 2013.

[4] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," in *Proceedings of OSDI*, 2006.

[5] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," in *SoCC*, 2010.

[6] A. Lakshman and P. Malik, "Cassandra: A Decentralized Structured Storage System," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, Apr. 2010.