Empirical Architectural Analysis on Performance Scalability of Petascale All-Flash Storage Systems

Mohammadamin Ajdari, Behrang Montazerzohour, Kimia Abdi, and Hossein Asadi

***Abstract*—In recent years, data-intensive applications have become dominant in data centers, demanding large-scale storage systems with hundreds of fast SSDs in multiple disk array enclosures. Although commercial products with large SSD arrays are available in the market, *no* performance analysis has been ever published on how the architectural choices affect the scalability of petascale storage systems. In this paper, we *first* analyze a real storage system consisting of 72 SSDs utilizing either *Hardware RAID* (HW-RAID) or *Software RAID* (SW-RAID), and show that SW-RAID is up to 7× faster. We then reveal that with increasing number of SSDs, the limited I/O parallelism in SAS controllers and multi-enclosure handshaking overheads cause significant performance drop, minimizing the total *I/O Per Second* (IOPS) of a 144-SSD system to less than a single SSD. *Second*, we disclose the most important architectural parameters that affect a large-scale storage system. *Third*, we propose a framework that models a large-scale storage system and estimates the system IOPS and system resource usage for various architectures. We verify our framework against a real system and show its high accuracy. *Lastly*, we analyze a use case of a 240-SSD system and reveal how our framework guides architects in storage system scaling.**

***Index Terms*—Solid-State Drives, RAID, Data Storage, Performance**

I. Introduction

In the current era of machine learning and the popularity of data-centric applications, data storage systems have become increasingly under pressure to provide very high capacity and performance. These systems, usually in the form of *Storage Area Network* (SAN) storage, consist of multi-hundred *Solid-State Drives* (SSDs), which normally do not fit in a single chassis, thus are placed in multiple or many inter-connected *Disk Array Enclosures* (DAEs). Examples of recent commercial products include Fujitsu Eternus AF250 S3 (with 264 SSDs) [1] and DELL Unity XT-380F (with 500 SSDs) [2].

The most common way to connect DAEs to increase the number of usable SSDs is to use daisy-chain topology (i.e., serializing DAEs) [3]. This topology provides the lowest design complexity and ease of adding DAEs with more emphasis on capacity expansion. Balancing the number of DAEs per expansion port on the main chassis (whenever enough ports exist) is also recommended by a few commercial products [4]. Unfortunately, the performance of these topologies has *never* been quantitatively studied before. Furthermore, the mutual impact of each topology with system configurations and resources (e.g., CPU utilization, RAID performance profile, and required number of PCIe slots on motherboard) has been also a mystery for system architects.

In this paper, we offer **four major contributions**. **First**, *we analyze a real enterprise-grade storage system with 72 SSDs and provide two major insights:* (a) *hardware RAID (HW-RAID) causes a major bottleneck on the RAID chip and limits IOPS even in a single chassis*, but *software RAID (SW-RAID) provides up to 7× higher IOPS* due to much higher scalability of CPU cores. (b) By increasing the number of SSDs and using *daisy-chain DAE connectivity, storage system IOPS significantly drops* due to *limited I/O parallelism in SAS controller chips and multi-enclosure communication overheads*. The expected performance drop in a 144-SSD system makes the overall system IOPS to be lower than that of a single SSD.

Mohammadamin Ajdari and Behrang Montazerzohour contributed equally.
Hossein Asadi is the corresponding author and is affiliated with Sharif University of Technology. All other authors are affiliated with both High Performance Data Storage and Processing (HPDS) Research and Sharif University of Technology in Tehran, Iran.
(e-mail: m.ajdari, beh.montazer, kimia.abdi, asadi@sharif.edu)

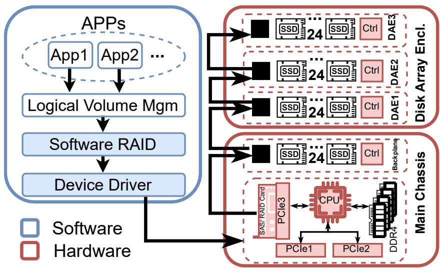


Fig. 1. Typical SAN storage system with a large number of SSDs

A naive approach to address the above limitation is hardware chip modification, e.g., redesigning SAS controller chips and DAE handshaking methods. However, redesigning such critical hardware components is not practical in the industry due to requiring a few years of time to mature before being comparable to current hardware in commercial environments. This adds to the excessive time and effort for general chip prototyping. Thus, an architect has to take another approach: increasing the number of existing SAS controller chips and providing more parallel accesses to DAEs. Nevertheless, to make this approach cost-effective, an architect needs to consider various real system limitations.

As our **second contribution**, we explore and reveal the most important architectural parameters for large-scale SSD-based storage systems. For example, we observe that DRAM capacity is not a key factor due to little DRAM usage for SSD array management; however, a) the number of available PCIe slots on the motherboard, b) the required *Quality of service* (QoS) for specific applications (on a specific set of SSD RAID arrays), and c) *performance and CPU utilization of the SW-RAID implementation for a single RAID array* highly affect the scalability of a large-scale-storage system.

As our **third contribution**, *we propose a framework that models a large-scale SSD-based storage system* and accurately estimates the system read/write IOPS and system resource usage for various possible architectures and DAE connectivity topologies. We verify our framework against a real enterprise-grade storage system with multiple configurations and up to 72 SSDs and show its *high accuracy with an average error of 13% for our framework predictions*.

As our **fourth contribution**, we analyze a use case with a 240-SSD system and reveal that *our framework provides optimal architectures that boost IOPS by an order of magnitude over the daisy-chain topology of DAEs while also meeting user requirements.*

II. Basics of SAN Storage and DAE Connectivity

When an enterprise application in SAN sends an I/O request, the request passes through the network and then traverses a number of steps in the storage stack of the SAN storage (Fig. 1). First, the request reaches a logical block device managed by a *Logical Volume Management* (LVM). Second, an LVM sends the request

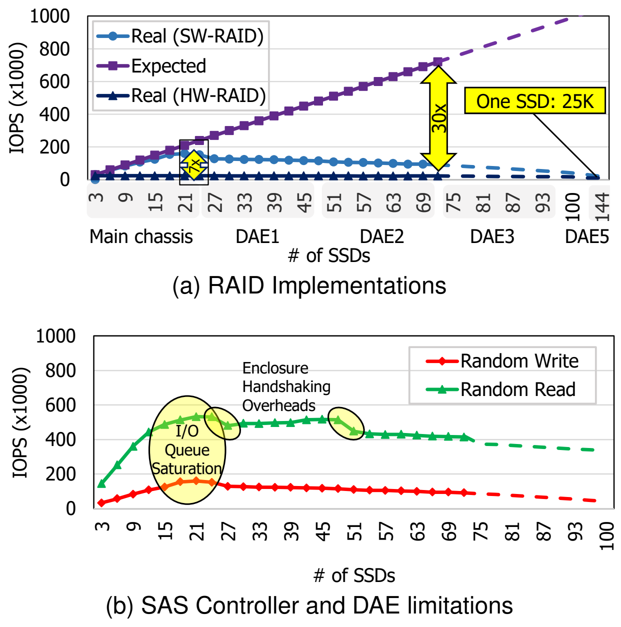


Fig. 2. Performance scalability limitation of large-scale all-flash storage systems (a) HW-RAID vs. SW-RAID, (b) SW-RAID scalability limitation

to the underlying RAID block device. The RAID block device consists of a number of SSDs grouped and exposed as a single block device. Distribution of data blocks among SSDs (i.e., RAID management) can be done completely by either HW-RAID or SW-RAID. Communicating with many SATA/SAS disks even with SW-RAID requires a SAS controller card. Therefore, SW-RAID uses a device driver and sends the I/O request to the SAS controller card through the motherboard and PCIe slot to which the SAS controller card is attached. Third, the SAS controller chip (on the card) prepares the SAS packets and sends them (on a bus or cables) to reach the main chassis backplane, on which the backplane controller chip routes them to the proper disk.

When a DAE is present, it is attached to the expansion ports on the main chassis backplane usually in the *daisy-chain topology* [3]. For more DAEs in this topology, the backplane of each DAE is serially connected to the previous one (Fig. 1). If an I/O request belongs to a disk on a DAE, the main chassis backplane (chip) re-routes the request to the first DAE, and the first DAE re-routes it to the next DAE, and it continues till the specific DAE and the disk is reached.

III. Our Insights Toward Performance Scalability

Here, we characterize a real enterprise-grade storage system with 72 SSDs and provide two novel insights and also the challenge to design a scalable storage system. First, we analyze the RAID implementation (HW-RAID vs. SW-RAID) and quantify the potential of SW-RAID over HW-RAID in large storage systems. Second, we reveal that even SW-RAID is not completely software-managed and the required SAS controller chip in data path limits the system performance to that of a few tens of SSDs. Next, for increasing number of SSDs and the need for more DAEs, we show that serialized DAEs and main chassis in the typical daisy-chain topology significantly degrade the performance, and quantify how this kills the scalability. Finally, we show the challenge of improving parallelism for SAS controllers and DAEs and motivate a practical solution.

A. RAID Implementation Problem

Observation 1. *HW-RAID has matured over many years in the storage industry. However, with SSD-based storage systems, the typical HW-RAID becomes the bottleneck with even a few number of SATA SSDs while it gets even 30× slower than expected when it is configured with 72 SSDs* (Fig. 2a).

We run a workload with uniform 8KB random writes on a real system with an industry-standard Broadcom 9361-8i HW-RAID and observe that on a RAID-5 array with three SSDs, the IOPS is around 25K. As the number of RAID-5 arrays increases (till 24 three-SSD arrays), the total system IOPS stays limited to around 25K due to limited on-chip resources (e.g., computational power and data distribution parallelism). Redesigning RAID-on-Chip with higher computational power and faster data distribution is excessively difficult. Adding multiple RAID chips on a single board, while slightly improves the performance, cannot resolve the huge performance gap either. Therefore, with many (and constantly increasing number of CPU cores), SW-RAID with minimal hardware support is the dominant industry trend for large SSD arrays. *The key question here is that whether SW-RAID (with many CPU cores and independent CPU threads per RAID array) provides linear scalability with the number of SSDs or not?*

B. SAS Controller and DAE Connectivity Problems

Observation 2: *We observe that SW-RAID is not completely software managed, despite being 7× faster than HW-RAID, it saturates IOPS with 24 SSDs, and significantly degrades with more SSDs due to other hardware components in data path* (Fig. 2).

First (**observation 2.a**), the SAS controller chip in the data path, which is required to aggregate and expose many SAS/SATA disks to the system, *has limited number of I/O paths, thus saturates the performance before a single chassis is filled with SSDs.* In our experimental setup with Broadcom 93 series-8i model, 8 internal ports, and thus 8 internal I/O queues are expected to exist, which is the maximum possible parallelism. Our tests show that with less than 8 arrays, IOPS gets saturated. Using the most expensive model with 24 internal ports increases the saturation threshold to at most 24 arrays (72 SSDs assuming 3-disk per array).

*The second scalability problem (**observation 2.b**) arises due to high communication overheads between controller chips of each backplane in the main chassis and DAEs.* As the number of DAEs in daisy-chain topology increases, we observe 10-20% IOPS degradation per added DAE, which is expected to bring down the total system IOPS below that of a single SSD, when the number of SSDs reaches 144 (with five DAEs). Such scalability limitations require increasing parallelism at both the SAS controller and DAE connectivity.

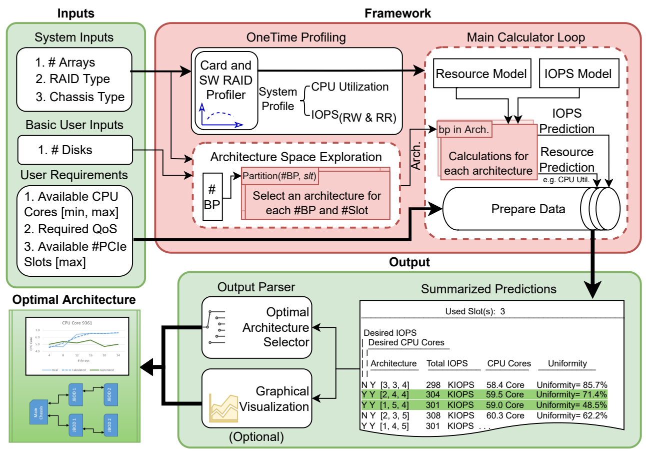
C. Practical Challenges

A straightforward approach to increase parallelism in SAS controller chips and reduce overheads in DAE communications is chip redesigning; however, this approach is impractical for the industry. Critical hardware components such as SAS controller chips and DAE controller chips have matured over a couple of years; and are globally designed by two or three major companies. Therefore, redesigning such chips would not only incur typical difficulties of hardware chip designs but also require a few years of maturity before becoming comparable to current hardware for the industry.

The second approach is reusing current matured hardware chips but increasing the instances of SAS controller cards and also providing more parallel connectivity to DAEs (instead of the default daisy-chain serialization topology). This approach is challenging because finding an optimal architecture requires considerations of various system limitations (e.g., the number of available PCIe slots on the motherboard for the SAS controller cards, the available CPU cores, a SAS controller performance profile) (Table I). In real multi-hundred all-flash storage systems, such considerations can easily lead to *over fifty different architectures*, which is hard to analyze by hand. Unfortunately, existing frameworks or simulators for storage systems focus on either modeling a single SSD (internal architecture) performance (e.g., [5], [6]), or reliability modeling (not performance modeling) of

TABLE I
SELECTED IMPORTANT ALL-FLASH STORAGE PARAMETERS

HW	Disks	# of Disks
	Chassis Type	# of Bays per Chassis
		Backplane Controller Chip Type
	SAS Controller Card	Performance Profile of the Card
	PCIe	# of PCIe Slots on Motherboard
	CPU	# of Cores
SW	SW-RAID	Performance & CPU Util. for One Array



```
Used Slot(s):  3

Desired IOPS
|  Desired CPU Cores
| |
| | Architecture   Total IOPS    CPU Cores    Uniformity
| |
N Y  [3, 3, 4]    298  KIOPS    58.4 Core    Uniformity= 85.7%
Y Y  [2, 4, 4]    304  KIOPS    59.5 Core    Uniformity= 71.4%
Y Y  [1, 5, 4]    301  KIOPS    59.0 Core    Uniformity= 48.5%
N Y  [2, 3, 5]    308  KIOPS    60.3 Core    Uniformity= 62.2%
Y Y  [1, 4, 5]    301  KIOPS   . . .
```

Fig. 3. Proposed framework: Overall flow

large storage systems (e.g., [7]). None of the existing studies have shown the scalability problems arising from SAS controller chips and DAE controller communications, thus no performance modeling for these problems has been done either. Such limitations point to the need for a framework to model a large-scale SSD-based system and automate the design-space exploration. Note that multiple main chassis (i.e., scale-out) may increase performance, but has different software stack overheads and challenges, thus our focus in this paper is single main chassis with many DAEs (scale-up approach).

IV. Our Proposed Framework

We offer three key ideas. **First**, we define critical parameters affecting performance scalability of large SSD-based systems and then propose *special considerations for the SAS controller and DAE serialization overheads* to enable our framework to provide highly accurate IOPS and system resource utilization predictions. **Second**, we propose *small-scale profiling then large-scale predictions* to easily update our HW/SW models for any new SW-RAID implementation or major hardware component changes. Such small-scale and one-time profiling usually requires a small number of SSDs and components of the target storage system; yet enables accurate, easy component model update. **Third**, we use *simplified IOPS and CPU utilization modeling* by considering fixed IOPS degradation rates depending on DAE location (for IOPS modeling) and linear regression (for CPU utilization). This approach, instead of complex models, enables easy implementation and fast predictions.

A. Simplified Operational Flow of Proposed Framework

Our framework consists of six major steps to disclose the optimal architecture for a large-scale storage system with a desired number of SSDs (Fig. 3). First, the framework receives the desired number of SSDs as the input. Second, it calculates all possible topologies that SAS controller cards, the main chassis, and DAEs can be connected together with the specified number of SSDs. Third, it applies our IOPS and resource utilization model to calculate the IOPS and resource utilization. Fourth, it receives additional user (or real system) requirements that the final architecture should meet. A storage system commonly runs various services and thus has (a) limited available CPU cores, (b) limited available PCIe slots on the motherboard, and (c) QoS requirements for some or all RAID arrays. We include these three as user requirements. Next, the framework outputs the summarized prediction results for each architecture and marks the architectures that have met the user requirements. Finally, a simple parser outputs the optimal or near-optimal architectures.

B. One-Time, Small-Scale Profiling

IOPS and CPU utilization prediction of a large-scale SSD-based system depend on the micro-architecture of a few major HW/SW components; thus a one-time, limited profiling of such components is required for accurate predictions (Table I). We propose to run a sample of experiments on a small-scale storage system with the desired components and workloads of interest to measure the IOPS and CPU utilization. The samples have two goals: (1) demonstrating the performance trend of components in the main chassis (mainly SAS controller card, SSDs, SW-RAID implementation), (2) disclosing the main chassis and DAE communication overhead for a specific chassis type. To achieve the first goal, a single chassis (full of SSDs) is usually enough. For example, with a popular card such as Broadcom 9400-8i, up to eight arrays (i.e., around 24 SSDs) would saturate the card. To achieve the second goal, sample tests on the system (*a chassis + a DAE*) must be conducted to generate the IOPS and CPU utilization model.

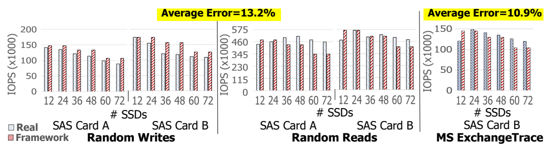
C. Details of System Modeling and Implementation

IOPS. We propose a simple yet effective chassis-wise IOPS modeling. In this approach, we consider an average fixed IOPS value for each chassis or DAE regardless of the number of arrays, and only the chassis or DAE location in the topology is considered. This is a suitable approach because the IOPS usually saturates with a few arrays and fluctuations after that are mainly due to DAE location.

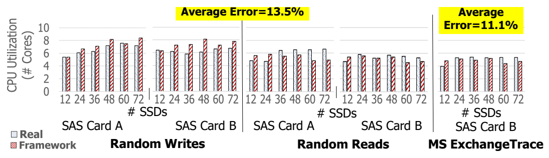
Following one-time profiling results, to ensure high accuracy of the framework, the framework sets the IOPS of the main chassis as the average of *maximum IOPS* and *IOPS at maximum number of arrays*. This approach is necessary to *soften IOPS fluctuations* that happen across different number of arrays. For example, with a 24-bay chassis and a 9400-8i card, a single array write IOPS is 45K and the maximum write IOPS achieved with six arrays is 193 KIOPS while the IOPS shrinks to 155 KIOPS with eight arrays. Thus, the framework reports the average of 193K and 155K (i.e., 174K) as the IOPS of the chassis.

To include the impact of DAEs and SAS card saturation on IOPS, the framework reduces the IOPS by a specified percentage for each DAE in the chain. Our profiling shows an average of 10% degradation till 1^{st} DAE and 20% from 1^{st} DAE to 2^{st} DAE. We conservatively assume a 20% value for farther DAEs, but real IOPS with many serialized DAEs may be even lower. Although the backplane chip and SAS card models are very limited, if they are significantly different from our tested hardware, one-time profiling with up to two DAEs is required to fine-tune these numbers.

CPU Utilization. SW-RAID handles each array on separate CPU processes (or threads), thus ideally the number of arrays (and IOPS) is proportional to CPU utilization. In a real system, the same IOPS does not result in the same CPU utilization because more arrays cause more CPU contention, thus additional CPU utilization. As our first choice for framework simplicity, we first define *CPU Factor* as the output of a linear regression model of *CPU util. over IOPS* with the dependent variable *number of SSD arrays (N)* (Eq. 1). We generate this regression model with four samples from the one-time profiling. Second, the framework multiplies the IOPS with the CPU Factor to calculate the CPU utilization for N arrays (Eq. 2).



(a) IOPS



(b) CPU Utilization

Fig. 4. Verification of our framework against a real system

$$CPUFactor_N = a.N + b \tag{1}$$

$$CPUUtil_N = IOPS_N * CPUFactor_N \tag{2}$$

Unimportant Parameters. We excluded two parameters from our model: *(a) DRAM capacity* due to little DRAM usage (i.e., MBs to few GBs) for fundamental array management of hundreds of SSDs. Additional storage services (e.g., caching) are not included in this analysis. (b) *per-CPU PCIe bandwidth,* because modern CPUs have a high number of PCIe lanes to provide enough (up to 10M) IOPS for high-end all-flash SATA/SAS storage systems.

Implementation. We implemented the core part of our framework in Python with *570 lines of code*.

V. Our Framework Verification

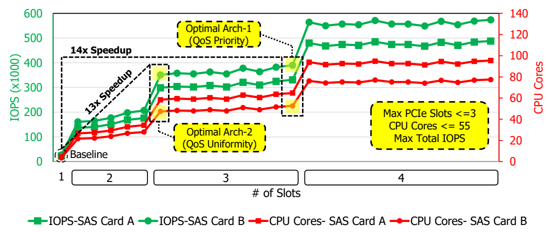
We validate our framework against a real system at different configurations. We use two types of SAS controller cards (i.e., Broadcom 9361-8i with HW-RAID feature disabled as Card A and Broadcom 9400-8i as Card B), different number of SSDs (from 12 SSDs up to 72 SSDs), Linux `mdadm` (for SW-RAID module) and run two major I/O request patterns (i.e., random 8KB reads and random 8KB writes), and a real trace (i.e., Microsoft Exchange mail trace [8]). The constant hardware components in all our tests are one X10DRL-i SuperMicro motherboard, dual-socket Intel E52620v4 CPUs, 128 GB of DDR4 DRAM, four Supermicro 24-bay chassis, and 72 Samsung SM863a 1.9TB SATA SSDs.

Our framework predictions show an average error of 10.9%-13.2% for IOPS and 11.1%-13.5% for CPU utilization compared to the real enterprise-grade storage system (Fig. 4). Our results reveal that chassis-wise IOPS modeling and linear modeling of CPU utilization are simple but highly effective in storage system scaling.

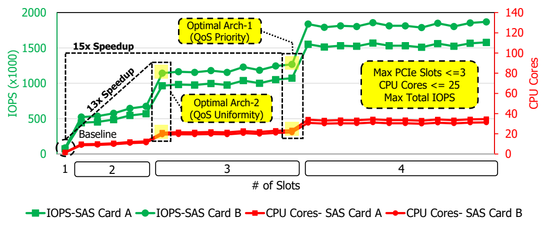
VI. Use case for a 240-SSD Storage System

We apply our framework to a use case of a 240-SSD system and show how our framework guides architects in scaling real storage systems. We assume the user requirements of (a) up to 55 CPU cores for write-intensive workloads and up to 25 CPU cores for read-intensive workloads, (b) up to three PCIe slots. We show two types of optimal architectures depending on QoS definition of user: (c1) maximizing total IOPS while providing similar IOPS for each RAID array regardless of its position in DAEs (*QoS-Uniformity*), (c2) higher IOPS priority for a small number of RAID arrays (*QoS-Priority*), which is similar to some real environments.

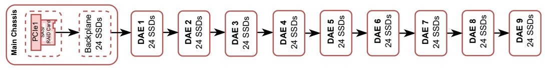
Our framework explores all *54 possible architectures* for a 240-SSD system and reveals two optimal architectures with *13×-15× speedup* for both reads and writes compared to the baseline daisy-chain topology (Fig. 5). Our results show the quantitative impact of SAS card/DAE parallelism through PCIe slots, SAS card model impact, and the QoS output of each topology. The framework predicts the baseline to provide only 27 KIOPS for writes and 86 KIOPS for



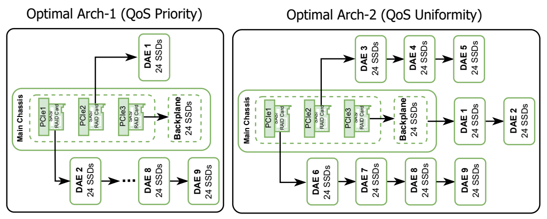
(a) Framework predictions for random writes



(b) Framework predictions for random reads



(c) Daisy-Chain Architecture



(d) Framework Result as Optimal Architecture

Fig. 5. Architectural analysis using our framework on a 240-SSD system

reads; however, by adding two SAS cards and distributing DAEs across all SAS controllers almost equally, IOPS reaches over 350 KIOPS for writes and one MIOPS for reads while QoS-uniformity is also achieved (*Optimal Arch-1*). By attaching eight DAEs to one SAS controller and one DAE per each of the two SAS controllers, higher total IOPS is achieved, and IOPS priority is given to 48 SSDs placed on two separate DAEs (*Optimal Arch-2*).

References

[1] Fujitsu. (2023) Fujitsu storage eternus af250 s3 all-flash array datasheet. [Online]. Available: https://sp.ts.fujitsu.com/dmsp/Publications/public/ds-eternus-af250-s3-ww-en.pdf

[2] DELL. (2023) Dell unity xt hfa and afa storage. [Online]. Available: https://www.delltechnologies.com/asset/en-us/products/storage/technical-support/h17713_dell_emc_unity_xt_series_ss.pdf

[3] D. EMC. (2019) Dell emc unity family installation guide. [Online]. Available: https://www.gotomojo.com/wp-content/uploads/2019/09/Dell-EMC-Unity-XT-380-installation-guide.pdf

[4] ——. (2021) Dell emc unity best practices guide. [Online]. Available: https://www.delltechnologies.com/asset/en-gb/products/storage/industry-market/h15093-dell_emc_unity-best_practices_guide.pdf

[5] M. Jung, J. Zhang, A. Abulila, M. Kwon, N. Shahidi, J. Shalf, N. S. Kim, and M. Kandemir, "Simplessd: Modeling solid state drives for holistic system simulation," *IEEE Computer Architecture Letters*, vol. 17, no. 1, pp. 37–41, 2017.

[6] H. G. Lee, M. Kim, J. Lee, E. Lee, B. S. Kim, S. Lee, Y. Kim, S. L. Min, and J.-S. Kim, "Learned performance model for ssd," *IEEE Computer Architecture Letters*, vol. 20, no. 2, pp. 154–157, 2021.

[7] Y. Qi, D. Feng, and B. Hou, "Towards building reliable and cost-efficient distributed storage systems," *IEEE Access*, vol. 8, pp. 157 862–157 877, 2020.

[8] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, "Migrating server storage to ssds: analysis of tradeoffs," in *Proceedings of the 4th ACM European conference on Computer systems*, 2009, pp. 145–158.