JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL. 24, NO. 1, FEBRUARY, 2024
https://doi.org/10.5573/JSTS.2024.24.1.8
ISSN(Print) 1598-1657
ISSN(Online) 2233-4866

Provisioning CSD-based Storage Systems with Erasure-coding Offloaded to the CSD

Hongsu Byun[1], Safdar Jamil[1], Junghyun Ryu[1], Sungyong Park[1], Myungcheol Lee[2], Sung-Soon Park[3], and Youngjae Kim[1]

Abstract—**While commercially available Computational Storage Drives (CSD) have appeared, it is challenging to build a CSD array-based storage system due to the lack of storage provisioning tools determining the performance and cost-effectiveness of a storage system with CSDs. Therefore, CSDPLAN, a storage provisioning tool to find the number of performance-efficient CSDs when building a storage system with CSD, has been proposed. However, the effectiveness of CSDPLAN has only been evaluated using specific big data analysis workloads, which are not computationally intensive. In this work, we extend CSDPLAN to propose CSDPLAN-EC, a CSD provisioning tool for building storage systems with computationally intensive erasure coding offloaded to CSDs. Our evaluation shows that the optimal number of CSDs running erasure coding in a storage system is 5 and that it decreases to 1 when the computational power of the CSDs is improved by a factor of 5.**

Index Terms—**Storage system, computational storage drive, erasure coding, analytical modeling and simulation**

Manuscript received Jun. 12, 2023; reviewed Oct. 4, 2023; accepted Oct. 16, 2023
[1]Dept. of Computer Science and Engineering, Sogang University, Seoul, Korea, 04107
[2]Smart Data Research Section, ETRI, Daejeon, Korea, 34129
[3]GlueSys & Anyang University, Seoul, Korea, 34129
E-mail : youkim@sogang.ac.kr (Corresponding author)

I. INTRODUCTION

Computational storage drives (CSDs) [1-10] have gained attention due to their hardware and software characteristics, such as high computational power and near-data processing while significantly reducing data movement costs. The storage architects can design storage systems that take advantage of these characteristics of CSDs to offload various computational tasks and provide high-end storage systems. Therefore, with the advent of computational storage devices (CSDs), there have been attempts to build storage systems using multiple CSDs [11-13]. However, there has been a lack of research on storage provisioning tools that evaluate the effectiveness when a storage system is built with CSDs rather than when storage is built based on an existing HDD/SSD.

Recently, when building a storage system with CSDs, a storage provisioning planning tool, CSDPLAN [14], that can evaluate the effectiveness of CSDs according to the work-load type has been proposed. CSDPLAN finds the optimal number of CSDs (break-even point) that is more effective than building a system based on traditional HDD/SSD when building a storage system with CSDs through a combination of mathematical models and experimental evaluation. Storage architects or system administrators can use CSDPLAN to determine whether it is cost-effective to build a storage system with CSD for a given workload.

On the other hand, erasure coding has recently been widely applied in various storage systems as an effective technique for increasing the availability of storage systems [15, 16]. In general, erasure coding is a

compute-intensive task. Therefore, performing erasure coding using the powerful CPU of the host may be faster than performing erasure coding in a CSD, which has weak computing power. However, if multiple CSDs can cooperate to process multiple erasure coding tasks in parallel, utilizing CSDs can be more effective.

Therefore, in this work, we propose CSDPLAN-EC for CSD efficiency provisioning when building storage systems using devices where erasure coding is offloaded to CSD. CSDPLAN-EC extends CSDPLAN to take into account the kernel executing erasure coding. We evaluated the effectiveness of building a storage system with CSD by performing an evaluation to find the break-even point (BEP) of CSD for workloads that perform various erasure coding using the CSDPLAN-EC. Our extensive evaluation showed that when the number of encoding blocks increased from 12 to 48, the BEP increased by a maximum of $2\times$ to $5\times$. However, if the computational power of CSD is increased by $5\times$, the BEP drops from 5 to 1.

II. BACKGROUND

1. Scalable, Reliable Storage with Erasure Coding

Distributed storage systems have traditionally adopted data replication to ensure high availability and durability. Data replication has the disadvantage of requiring additional storage capacity equal to the number of replicas. As the amount of stored data increases exponentially [17, 18], erasure coding is being adopted to use storage capacity more efficiently. For erasure coding, Reed-Solomon (RS) [19] codes are widely used [20-23]. The (n, k) RS code, composed of two parameters, n and k, encodes one data block into n data blocks and k parity blocks, and even if up to k blocks are lost, the original data block can be restored. Therefore, data storage capacity can be drastically reduced compared to the data replication method.

2. Storage Capacity Provisioning Tool

In the past decade, several storage provisioning studies have been conducted that aimed to design cost-effective storage systems using SSD instead of HDDs [14, 24-27]. Narayanan et al. [24] conducted a study on the significance of Solid-State Drives (SSDs) within enterprise-level computing systems by analyzing actual data traces from data centers. They delved into the cost-efficiency comparisons across diverse arrangements of SSDs and Hard Disk Drives (HDDs). Kim et al. [25] explored the challenge of determining the most efficient storage configuration for computing systems that utilize both SSDs and HDDs, ensuring they fulfill performance criteria. Furthermore, they examined the strategy behind the fluid allocation of workloads in hybrid storage setups. On the other hand, Byun et al. [14] delved into the complexities of evaluating the cost-efficiency of establishing a storage system that incorporates Computational Storage Devices (CSDs). Given the intricacies of CSDs, which are significantly more advanced than SSDs, pinpointing the design prerequisites, including the number of CSDs necessary for a computational node based on a CSD array, poses a considerable challenge. They scrutinized the computational prowess and input/output processing capabilities of prevalent CSDs, finding that (i) CSDs exhibit a wide range of performance capabilities, distinct from SSDs, and (ii) there are noticeable disparities in performance trends even among CSDs themselves.

3. CSDPLAN: Capacity Planning

This section provides a brief introduction to CSDPLAN. More details can be found in the CSDPLAN paper [14].

1) System Modeling: CSDPLAN [14] is a software tool that uses a mathematical analysis model to guide system and storage designers in constructing storage systems. CSDPLAN's model requires the I/O and computation performance parameters of CSDs for a given application as input and produces the minimum number of CSDs that are cost-effective when compared to a conventional SSD-based storage system as output.

CSDPLAN considers the execution time of an analysis kernel (W) on two systems as follows.

- **SSD-array system (n)** ($T_{\text{SSD}(n)\text{-array}}$): A system with a host with a block-based SSD array. It executes the analytic kernel using the n cores and memory of the host.
- **CSD-array system** ($T_{\text{CSD-array}}$): A system with a CSD array on the host. It executes the analytic kernel using the memory and CPU of the CSDs.

The kernel's execution time in an SSD-array system is modeled as data transfer time ($T_{\text{SSD-tx}}$) and computation time ($T_{\text{SSD}(n)\text{-comp.}}$). If M SSDs are equipped, and the data required to run the kernel is uniformly distributed, the data transfer time is reduced to $1/M$. However, the computation is not reduced because it is performed by one host CPU. However, the computation is not reduced because it is performed by a single host CPU. Therefore, the model for an SSD-array system is as follows.

$$T_{\text{SSD}(n)\text{-array}} = \frac{1}{M} \cdot T_{\text{SSD}-\text{tx}} + T_{\text{SSD}(n)\text{-comp}} \quad (1)$$

In an SSD-array system with an array of M SSDs, the data transfer time can theoretically be reduced by $1/M$. However, if the bus connection becomes a bottleneck [28], the reduction is no longer possible. Therefore, CSDPLAN defines the number of SSDs (M) at which the bottleneck occurs in an SSD-array system as follows.

$$M = \begin{cases} M & \text{if}(M < k_{\text{limit}}) \\ k_{\text{limit}} & \text{else} \end{cases} \quad (2)$$

The model of CSD-array system is more efficient than the SSD-array system because each CSD is equipped with computing power, reducing both computation and data transfer time to $1/M$. Furthermore, without a shared bus between CSDs, there are no bottlenecks as the number of CSDs increases. To extend the model of CSD-array system, the following steps are required.

$$T_{\text{CSD-array}} = \frac{1}{M} \cdot T_{\text{CSD}-\text{tx}} + \frac{1}{M} \cdot T_{\text{CSD}-\text{comp}} \quad (3)$$

2) Solver: Finding the Break-Even Point: CSDPLAN deploys a solver to find the BEP for the number of CSDs in a CSD-array-based node. CSDPLAN's solve takes the CSD's performance characteristics as an input and returns an optimal number of CSDs as an output, which represents the following BEP where the CSD-array is more effective than the conventional compute node. Therefore, the following model represents the formula for $T_{\text{SSD}(n)\text{-array}}$ to exceed $T_{\text{CSD-array}}$.

$$T_{\text{SSD}(n)\text{-array}} > T_{\text{CSD-array}}$$

$$\Rightarrow \frac{1}{M} \cdot T_{\text{SSD}-\text{tx}} + T_{\text{SSD}(n)\text{-comp.}} > \frac{1}{M} \cdot T_{\text{CSD}-\text{tx}} + \frac{1}{M} \cdot T_{\text{CSD}-\text{comp}}$$

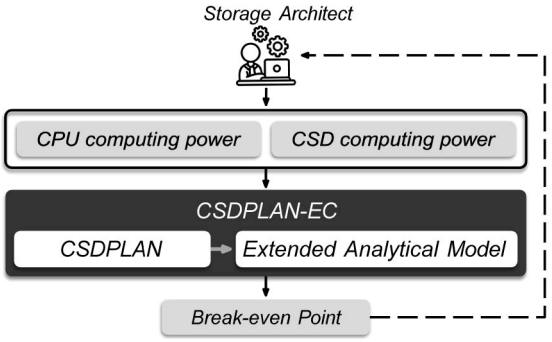


Fig. 1. An overview of CSDPLAN-EC.

Solve the above inequality for M. We omit the details of the theorem.

$$M > \frac{T_{\text{CSD-tx}} + T_{\text{CSD-comp.}} - T_{\text{SSD-tx}}}{T_{\text{SSD}(n)\text{-comp.}}}$$

Therefore, the BEP (M) function based on the internal I/O bandwidth and computational power of the CSD is as follows.

$$S\left(T_{\text{CSD-tx}}, T_{\text{CSD-comp.}}\right) = \left\lceil \frac{T_{\text{CSD-tx}} + T_{\text{CSD-comp.}} - T_{\text{SSD-tx}}}{T_{\text{SSD}(n)\text{-comp.}}} \right\rceil^{1} \quad (4)$$

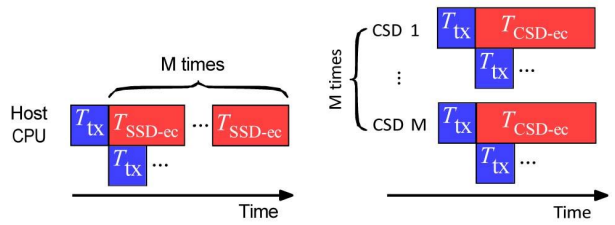
III. CSDPLAN-EC

CSDPLAN aims to provide CSD effectiveness guidelines based on workload-specific processing time (data transfer time and computation time). However, workloads performing continuous erasure coding are dominated by computation time with minimal data transfer time. Therefore, it is necessary to modify the mathematical analysis model to apply erasure coding into CSDPLAN. In this section, we propose CSDPLAN-EC, which provides CSD effectiveness guidelines when performing erasure coding on a storage server.

1. Overview of CSDPLAN-EC

Fig. 1 depicts an overview of CSDPLAN-EC. Like CSDPLAN, CSDPLAN-EC is a software tool that provides storage architects with a break-even point (BEP). CSDPLAN-EC is specifically extended to apply CSDPLAN

[1] ($\lceil\ \rceil$) is least integer function



(a) SSD system **(b)** CSD system

Fig. 2. Comparison of erasure coding execution time between SSD system and CSD system.

when building a storage server that runs erasure coding. The storage architect inputs into CSDPLAN-EC the computing power of the host CPU and the CSD. CSDPLAN-EC provides a BEP using a specialized analysis model based on CSDPLAN when executing erasure coding, which the storage architect can use as a guideline.

2. Extended of CSDPLAN for EC Offloading

We explain the CSDPLAN-EC, which is the methodology for using CSDPLAN when performing erasure coding. Fig. 2(a) and (b) depict the execution time phase when performing erasure coding for M data blocks in the SSD system and CSD system equipped with M number of devices, respectively. In each figure, erasure coding time ($T_{\text{SSD-ec}}$, $T_{\text{CSD-ec}}$) is (n, k) RS code execution time once. In other words, it means that erasure coding is executed M times in each system equipped with M number of devices. In Addition, it is assumed that block data transfer is possible asynchronously with erasure coding.

Therefore, in order to apply CSDPLAN modeling, as shown in Fig. 2(a), the execution time when erasure coding is performed for M blocks in an SSD system with M SSDs is as follows.

$$T_{\text{SSD}(n)\text{-array}} = \frac{1}{M} \cdot T_{\text{SSD-tx}} + M \cdot T_{\text{SSD}(n)\text{-ec}} \tag{5}$$

On the other hand, in the CSD system equipped with M number of CSDs, as shown in Fig. 2(b), parallel processing is performed in each CSD, as follows.

$$T_{\text{CSD-array}} = T_{\text{CSD-tx}} + T_{\text{CSD-ec}} \tag{6}$$

To find the BEP, we derive the mathematical model as follows:

$$T_{\text{SSD}(n)\text{-array}} > T_{\text{CSD-array}}$$
$$\Rightarrow \frac{1}{M} \cdot T_{\text{SSD-tx}} + M \cdot T_{\text{SSD}(n)\text{-ec}} > T_{\text{CSD-tx}} + T_{\text{CSD-comp}}$$

By assumption, data transfer time is only required once for the first time, so it is ignored for simplicity of the model. Then,

$$\Rightarrow M \cdot T_{\text{SSD}(n)\text{-ec}} > T_{\text{CSD-ec}}$$
$$M \geq \left\lceil \frac{T_{\text{CSD-ec}}}{T_{\text{SSD(n)-ec}}} \right\rceil \tag{7}$$

Therefore, BEP is affected by the computational power of the host CPU and CSD, and BEP changes according to this are as follows.

$$S_{\text{ec}}\left(T_{\text{CSD-ec}}, T_{\text{SSD(n)-ec}}\right) = \left\lceil \frac{T_{\text{CSD-ec}}}{T_{\text{SSD(n)-ec}}} \right\rceil \tag{8}$$

IV. EVALUATION

Our evaluation consists of two parts: one focuses on evaluating the performance of commercial CSDs, while the other concentrates on evaluating the effectiveness of CSDPLAN when erasure coding offloaded to CSD.

1. Evaluation of Commercial CSDs

We extensively evaluated the computing and I/O performance of two notable commercial CSDs, namely SmartSSD and Newport CSD. The evaluation was conducted on a host server that had dual AMD EPYC™ 7352 CPUs, 256GB DRAM and was running CentOS 7.9. The comprehensive specifications of the host server and the CSDs can be found in Tables 1 and 2.

We employed various methodologies to evaluate the performance of CSDs, depending on their unique hardware designs. To evaluate computing performance, we developed in-house big data analysis kernels[2] such as Count (4.8 GB), Vector Addition (4.8 GB), Array Merge

[2] The data size in parentheses is the workload size.

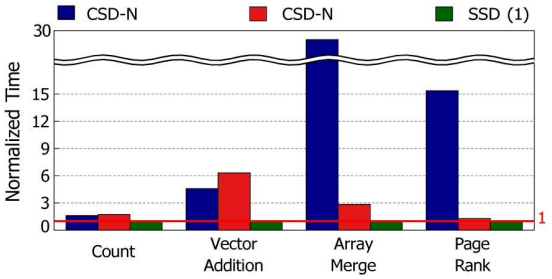


Fig. 3. Comparison of the computational power of SSD system and CSD system.

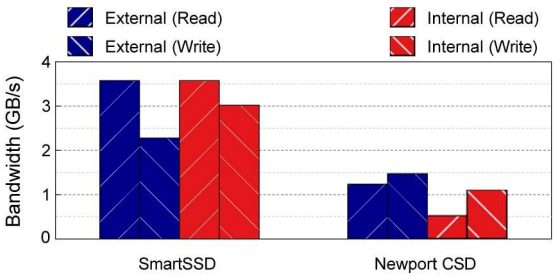


Fig. 4. Bandwidth evaluation of SmartSSD and Newport CSD.

(4.8 GB), and Page Rank (0.2 GB). In contrast, we used the FIO benchmark [31] for the host and Newport CSD to measure external and internal I/O bandwidth. For SmartSSD, we used a bandwidth measurement kernel program [32]. To set up the FIO benchmark, we chose LibAIO [3] and selected the direct option while configuring it for a 1MB request size, 64 queue depth, and sequential pattern.

1) I/O Bandwidth Capability: In Fig. 4, we compare the internal and external I/O bandwidth of the CSDs. The internal I/O bandwidth pertains to the rate at which the device's kernel accesses data in the NAND flash, whereas external I/O bandwidth refers to the speed at which the host's kernel accesses data in the NAND flash. For SmartSSD, we observed that the external bandwidth for reads is roughly 1.18× higher than the internal bandwidth. On the other hand, for writes, the internal bandwidth is about 1.36× higher than the external bandwidth. We attribute these constraints to the PCIe bus connecting the SSD and the FPGA, which limits the maximum internal bandwidth.

In contrast, when testing the Newport CSD, we made an unexpected observation: the external bandwidth for both read and write workloads was remarkably higher than the internal bandwidth by a factor of 2.28 and 1.33, respectively. This finding suggests that the internal bandwidth of commercial CSDs may be lower than their external I/O bandwidth. Therefore, kernels executed within the device cannot always rely on having higher I/O bandwidth.

2) Computational Capability: We evaluated the computing power of an SSD system (1) using a single core of the CPU in the host server and two CSD systems using SmartSSD and Newport CSD's processing engine when running the analysis kernel. Fig. 3 shows the execution times of the CSD systems for analysis kernel workloads normalized to the SSD system (1). For the convenience of description[4], the CSD system equipped with SmartSSD and Newport CSD and the SSD system (1) are expressed as CSD-S, CSD-N, and SSD(1), respectively. We conducted this experiment with the assumption that all data required for kernel execution was already loaded entirely in DRAM, thus eliminating any I/O time involvement. To maximize the performance, CSD-S used all FPGA optimization techniques [33-36], such as local memory buffer, loop pipelining, and multiple compute units, while CSD-N adopted multithreading.

Overall, both CSD-S and CSD-N are slower than the SSD (1) for compute-intensive workloads. In all analysis kernels, CSD-S was 1.6×, 4.6×, 28.1×, and 15.4× slower than SSD(1), respectively, and CSD-N was 1.7×, 6.3×, 2.8×, and 1.3× slower than SSD(1), respectively. Therefore, the representative commercial CSDs have low computing power, and naively offloading computation to the device might not be as effective as expected.

Table 1. Detailed specification of the host server

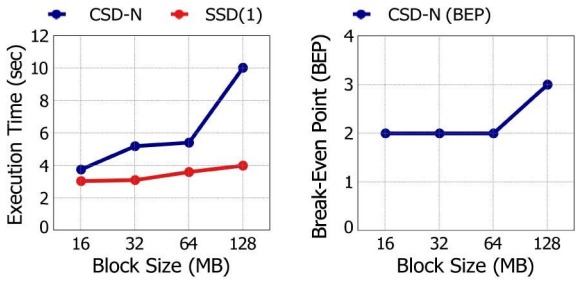
CPU	AMD EPYC™ 7352 24 Cores, 2.3 GHz (Up to 3.2 GHz)
Socket	2 NUMA Node
Memory	256 GB DRAM DDR4 3200 MHz
OS	Centos 7.92.2009 (Core) / Linux Kernel 4.14

Table 2. Detailed specification of the CSD

	SmartSSD [11]	Newport CSD [13]
In-Storage Processing Engine	Xilinx Kintex Ultrascale+ KU15P	ARM Cortex-A53 1.0 GHz, 4 Cores
	DRAM : 4 GB DDR4	DRAM : 8 GB DDR4
	Clock : 300 MHz	OS : Linux Kernel 4.14

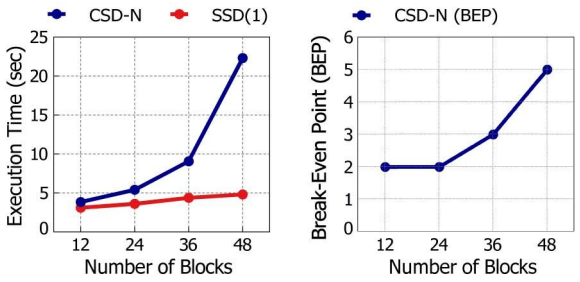
[3] Linux-native Asynchronous I/O Access Library

[4] Hereinafter, we use SSD system, CSD system, SSD(n), CSD-S, and CSD-N interchangeably.



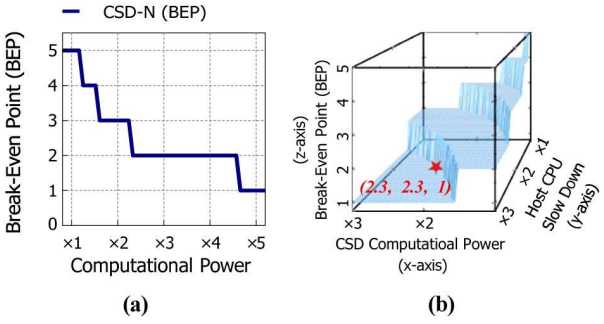
(a) Execution Time **(b)** Break-Even Point

Fig. 5. Changes of erasure coding execution time and break-even point according to block size.



(a) Execution Time **(b)** Break-Even Point

Fig. 6. Changes of erasure coding execution time and break-even point according to the number of blocks.



(a) **(b)**

Fig. 7. Change in break-even point with computational power: (a) Increase in computational power of CSD; (b) Increase in computational power of CSD and decrease in computational power of host CPU.

2. Evaluation of CSDPLAN for Erasure Coding

We evaluated CSDPLAN when erasure coding offloaded to CSD through the change in break-even point (BEP) according to the size and number of blocks and the computational power of CSD. The *block size* means the size of the data block to execute RS code, and the *number of blocks* is the number of blocks created after the completion of RS code execution. When (n, k) RS code is executed, $n + k$ blocks are created.

We evaluated erasure coding on both the SSD system and the CSD system with a single device. When running erasure coding, the SSD system uses the host CPU single-threaded, whereas the CSD system uses multi-threading. In addition, to use the same erasure coding library [37] in the Linux environment for both the SSD system and CSD system, the CSD system used Newport CSD running operating system.

1) Impact of Block Size: Fig. 5(a) shows the execution time of the SSD system and CSD system according to the block size when encoding 24 blocks with (16, 8) RS code. As the block size increases, SSD(1) has little change from 3 to 4 seconds, while CSD-N slows down significantly from 4 seconds to 10 seconds. This is because CSD's computational power is weaker than that of the host CPU, so it is more affected by the block size.

Fig. 5(b) shows the BEP related to Fig. 5(a). When the block size is 16 MB, the BEP is 2, and when the block size is 128 MB, the BEP increases to 3. Compared to the block size increase, the BEP increase is insignificant.

2) Impact of Number of Blocks: Fig. 6(a) shows the execution time for encoding 64 MB data blocks from 12 to 48 blocks. In (n, k) RS code, $n = 2k$, i.e., (8, 4) RS code when encoding with 12 blocks. When the number of blocks increases from 12 to 48, SSD(1) shows little change from 3 seconds to 4.8 seconds, while CSD-N increases significantly from 3.8 seconds to 22.3 seconds. Fig. 6(b) shows the BEP change of CSD-N related to Fig. 7. BEP more than doubles from 2 to 5. In erasure coding, it can be seen that BEP is more affected by the number of blocks than the size of blocks.

3) Impact of Computational Power: Fig. 7 shows the BEP as a function of computational power when executing the (32, 16) RS code with a 64 MB data block. Fig. 7(a) shows the BEP as the computational power of the CSD increases. It can be seen that the BEP decreases to 1 when the computational power is increased by a factor of 5. Through this, it is possible to know the required degree of computational power improvement of CSD to find an appropriate BEP. Fig. 7(b) shows the BEP with increasing computational power of the CSD and decreasing computational power of the host CPU. In Fig. 7(a), the BEP becomes 1 when the CSD computational power is increased by 5×, but the BEP becomes 1 when the CSD computational power is increased by 2.3×, and

the host CPU computational power is decreased by 2.3×. This shows that the BEP can be found by combining the host CPU and CSD.

V. CONCLUSION

In this paper, we proposed CSDPLAN-EC for evaluating the effectiveness of building a storage system that offloads erasure coding to CSD. CSDPLAN-EC finds the break-even point (BEP), which is the number of CSDs in the CSD system that outperforms the existing SSD-based storage system according to the number and size of blocks in erasure coding. In our extensive evaluation of CSDPLAN, when the number of encoding blocks increased from 12 to 48, the BEP increased by a maximum of 2× to 5×, and at this time, when the computational power of CSD increased by 5×, the BEP decreased from 5 to 1.

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. 2021-0-00136, Development of Big Blockchain Data Highly Scalable Distributed Storage Technology for Increased Applications in Various Industries) and the Korea government (MSIT) (No. 2020-0-00104, Development of Low-latency Storage Module for I/O Intensive Edge Data Processing).

REFERENCES

[1] Z. Ruan, T. He, and J. Cong, "INSIDER: Designing In-Storage computing system for emerging High-Performance drive," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. Renton, WA: USENIX Association, Jul. 2019, pp. 379-394. [Online]. Available: https://www.usenix.org/conference/atc19/presentation/ruan

[2] S. Watanabe, K. Fujimoto, Y. Saeki, Y. Fujikawa, and H. Yoshino, "Column-oriented Database Acceleration using FPGAs ," in *Proceedings of 2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 686-697.

[3] S. Xu, T. Bourgeat, T. Huang, H. Koim, S. Lee, and Arvind, "AQUOMAN: An Analytic-Query Offloading Machine," in *Proceedings of the 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture(MICRO)*, 2020, pp. 386-399.

[4] D. Kwon, D. Kim, J. Boo, W. Lee, and J. Kim, "A Fast and Flexible Hardware-based Virtualization Mechanism for Computational Storage Devices," in *Proceedings of the 2019 USENIX Annual Technical Conference (ATC)*, 2021, pp. 729-743.

[5] Y. Kang, Y.-s. Kee, E. L. Miller, and C. Park, "Enabling Cost-effective Data Processing with Smart SSD," in *Proceedings of the 29th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2013, pp. 1-12.

[6] B. Gu, A. S. Yoon, D.-H. Bae, I. Jo, J. Lee, J. Yoon, J.-U. Kang, M. Kwon, C. Yoon, S. Cho, J. Jeong, and D. Chang, "Biscuit: A Framework for near-Data Processing of Big Data Workloads," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16, 2016, p. 153-165.

[7] G. Koo, K. K. Matam, T. I, H. V. K. G. Narra, J. Li, H.-W. Tseng, S. Swanson, and M. Annavaram, "Summarizer: Trading Communication with Computing near Storage," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17, 2017, p. 219-231.

[8] W. Cao, Y. Liu, Z. Cheng, N. Zheng, W. Li, W. Wu, L. Ouyang, P. Wang, Y. Wang, R. Kuan, Z. Liu, F. Zhu, and T. Zhang, "Polardb meets computational storage: Efficiently support analytical workloads in cloudnative relational database," in *Proceedings of the 18th USENIX Conference on File and Storage Technologies*, ser. FAST'20. USA: USENIX Association, 2020, p. 29-42.

[9] S. Liang, Y. Wang, Y. Lu, Z. Yang, H. Li, and X. Li, "Cognitive SSD: A Deep Learning Engine for In-Storage Data Retrieval," in *Proceedings of the 2019 USENIX Annual Technical Conference (ATC)*. USENIX, 2019, p. 395-410.

[10] S.-W. Jun, M. Liu, S. Lee, J. Hicks, J. Ankcorn, M. King, S. Xu, and Arvind, "BlueDBM: An Appliance for Big Data Analytics," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)*. ACM, 2015, p. 1-13.

[11] Samsung Electronics, "SmartSSD." [Online]. Available: https://semiconductor.samsung.com/ssd/smart-ssd/

[12] (2022) Los alamos national laboratory and sk hynix to demonstrate first-of-a-kind ordered key-value store computational storage device. Last Accessed: November 28, 2022. [Online]. Available: https://discover.lanl.gov/news/0728-storage-device

[13] NGD Systems, "Newport CSD." [Online]. Available: https://www.ngdsystems.com/solutions#NewportSection

[14] H. Byun, S. Jamil, J. Han, S. Park, M. Lee, C. Kim, B. Choi, and Y. Kim, "An analytical model-based capacity planning approach for building csd-based storage systems," *ACM Trans. Embed. Comput. Syst.*, sep 2023, just Accepted. [Online]. Available: https://doi.org/10.1145/3623677

[15] S. B. Balaji, M. N. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan, and P. V. Kumar, "Erasure coding for distributed storage: An overview," 2018.

[16] A. A. Helal, A. A. Heddaya, and B. B. Bhargava, *Replication techniques in distributed systems*. Springer Science & Business Media, 2005, vol. 4.

[17] E. Brewer, L. Ying, L. Greenfield, R. Cypher, and T. T'so, "Disks for data centers," 2016.

[18] S. Ben-Yair, "Updating google photos storage policy to build for the future," *The Keyword Google Blog*, 2020.

[19] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300-304, 1960.

[20] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, V. Sivakumar, L. Tang *et al.*, "f4: Facebook's warm blob storage system," in *Proc. of USENIX OSDI*, 2014, pp. 383-398.

[21] D. Ford, F. Labelle, F. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," 2010.

[22] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster," in *Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems*, 2013.

[23] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th symposium on Operating systems design and implementation*, 2006, pp. 307-320.

[24] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, "Migrating server storage to ssds: analysis of tradeoffs," in *Proceedings of the fourth European Conference on Computer Systems*, ser. EuroSys '20, 2009.

[25] Y. Kim, A. Gupta, B. U. and. Piotr Berman, and A. Sivasubramaniam, "HybridStore: A Cost-Efficient, High-Performance Storage System Combining SSDs and HDDs," in *Proceedings of the 19th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2011, pp. 227-236.

[26] Y. Kim, A. Gupta, B. Urgaonkar, P. Berman, and A. Sivasubramaniam, "Hybridplan: a capacity planning technique for projecting storage requirements in hybrid storage systems," *The Journal of Supercomputing*, vol. 67, no. 1, pp. 277-303, 2014.

[27] D. Boukhelef, J. Boukhobza, K. Boukhalfa, H. Ouarnoughi, and L. Lemarchand, "Optimizing the cost of dbaas object placement in hybrid storage systems," *Future Generation Computer Systems*, vol. 93, pp. 176-187, 2019.

[28] K. Latecki and M. Wawryk, "SPDK NVMe BDEV Performance Report Release 22.01," pp. 11-12, February 2022. [Online]. Available: https://ci.spdk.io/download/performance-reports/SPDK_nvme_bdev_perf_report_2201.pdf

[29] Axboe, J, "Github—axboe/fio: Flexible i/o tester," 2021. [Online]. Available: https://github.com/axboe/fio

[30] ARM Xilinx, "P2P bandwidth Example," 2021. [Online]. Available: https://github.com/Xilinx/Vitis_Accel_Examples/tree/master/host/p2p_bandwidth

[31] ARM Xilinx, "Vitis Accel Examples," Mar. 30, 2023. [Online]. Available: https://github.com/Xilinx/Vitis_Accel_Examples

[32] ARM Xilinx. (2022) UG1416-Vitis-Documentation. [Online]. Available: https://docs.xilinx.com/v/u/en-

US/ ug1416-vitis-documentation
[33] ARM Xilinx, “UG1399-Vitis-HLS,” Mar. 30, 2023. [Online]. Available: https://docs.xilinx.com/r/en-US/ ug1399-vitis-hls
[34] ARM Xilinx, “Vitis Unified Software Platform,” Mar. 30, 2023. [Online]. Available: https://docs.xilinx.com/r/en-US/ug1393-vitis-application-acceleration
[35] Intel, “Intelligent Storage Acceleration Library.” [Online]. Available: https://github.com/intel/isa-l

Hongsu Byun received the BS degree in computer science from Sogang University, South Korea, in 2021. He is currently pursuing the M.S. degree leading to Ph.D. degree in integrated program with the Department of Computer Science and Engineering, Sogang University, Seoul. His research interests include operating systems, file and storage systems, and parallel and distributed systems.

Safdar Jamil received the BE degree in Computer Systems Engineering from Mehran University of Engineering and Technology (MUET), Jamshoro, Pakistan in 2017. He is working toward the MS leading to PhD integrated program degree in the Department of Computer Science and Engineering, Sogang University, Seoul, South Korea. His research interests include scalable indexing data structures and algorithms, NoSQL database, data deduplication and high performance computing.

Junghyun Ryu received the BS degree in computer science from Kookmin University, Seoul, South Korea, in 2021. He is currently pursuing the M.S. degree with the Department of Computer Science and Engineering, Sogang University, Seoul. His research interests include operating systems, file and storage systems.

Sungyong Park is a professor in the Department of Computer Science and Engineering at Sogang University, Seoul, Korea. He received his B.S. degree in computer science from Sogang University, and both the M.S. and Ph.D. degrees in computer science from Syracuse University. From 1987 to 1992, he worked for LG Electronics, Korea, as a research engineer. From 1998 to 1999, he was a research scientist at Telcordia Technologies (formerly Bellcore), where he developed network management software for optical switches. His research interests include cloud computing and systems, virtualization technologies, high performance I/O and storage systems, and embedded system software.

Myungcheol Lee (Member, IEEE) received the BS and MS degrees in computer engineering from Chungnam National University in 1999 and 2001, respectively. He has been working as a principal researcher at Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea since 2001. His research interests include database management systems, distributed storage and processing systems, distributed stream processing systems, cloud computing, big data storage and processing platforms, and blockchain storage systems.

Sung-Soon Park received the B.S. degree in computer science from Hongik University, in 1984, the master’s degree in computer science and statistics from Seoul National University, in 1987, and the Ph.D. degree in computer science from Korea University, in 1994. He worked as a Lecturer (full-time) at Korea Air Force Academy, from 1988 to 1990. He also worked as a Postdoctoral Researcher at Northwestern University, from 1997 to 1998. He is currently a Professor with the Department of Computer Science and Engineering, Anyang University, and also the CEO of Gluesys Company Ltd. His research interests include network storage systems and cloud computing.

Youngjae Kim (Member, IEEE) received the BS degree in computer science from Sogang University, South Korea in 2001, the MS degree in computer science from KAIST in 2003, and the PhD degree in computer science and engineering from Pennsylvania State University, University Park, Pennsylvania in 2009. He is currently an associate professor with the Department of Computer Science and Engineering, Sogang University, Seoul, South Korea. Before joining Sogang University, Seoul, South Korea, he was a R&D staff scientist at the US Department of Energy's Oak Ridge National Laboratory (2009-2015) and as an assistant professor at Ajou University, Suwon, South Korea (2015-2016). His research interests include operating systems, file and storage systems, database systems, parallel and distributed systems, and computer systems security.