Implementation of mirroring (RAID1) in SPDK with LIBAIO io-engine

Gaurav Chawda, Gauri Vijaykar, Yash Kalavadiya, Siddhesh Kotkar
Student
Dept. of Computer Technology
PICT, Pune

Dr. Girish Potdar
Associate Professor
Dept. of Computer Technology
PICT, Pune

Abstract—Data security is paramount for modern businesses, and data loss can be catastrophic. Mirroring offers a solution by automatically creating a redundant copy of data, while also distributing read loads. While NVMe drives boast low individual failure rates, the probability of simultaneous failure in mirrored configurations becomes significantly lower, enhancing data re-silience. This project presents a novel approach for implementing RAID 1 in SPDK modules using LIBAIO. RAID 1 also known as mirroring, offers data redundancy by replicating data across multiple NVMe Devices, thereby enhancing fault tolerance and data availability. SPDK provides a framework for building high-performance, user-space storage applications while LIBAIO of-fers asynchronous I/O handling capabilities enabling this process to be nonblocking. In our proposed solution, we leverage SPDK's BDev modules which are workers for performing IO operations, and build our virtual-Bdev module. By integrating these technolo-gies, we achieve a RAID 1 implementation that optimizes data mirroring across multiple drives while minimizing overhead. We discuss the implementation details and performance evaluation of our RAID 1 solution in terms of scalability. Overall, our work showcases the synergy between SPDK and LIBAIO in building high-performance RAID 1 systems, offering insights for future developments in storage technologies and data protection mechanisms.

Index Terms—IO_URING, LIBIO, IO-Engines, Storage Performance Development Kit.

I. Introduction

The advancement of Non-Volatile Memory(NVM) a Solid State Drive(SSD) technology has significantly enhanced the storage effectiveness and data storage capabilities [4,5]. De-spite the remarkable performance gains offered by Solid-State Drives (SSDs), fault tolerance is still a challenge [5,7], limiting their full potential. This paper investigates three key solutions to address storage performance limitations: RAID 1 for data redundancy, libaio for asynchronous I/O, and SPDK for high-performance storage access.

Data storage demands are continuously evolving, necessitat-ing solutions that optimize performance, power efficiency, and reliability. Redundant Arrays of Independent Disks (RAID) technology has emerged as a powerful approach, offering significant advantages over traditional single-disk storage. RAID configurations can concurrently achieve increased performance, reduced power consumption, and enhanced fault tolerance [3].

Redundant Array of Independent Disks (RAID) 1, also known as mirroring, utilizes a data storage configuration that prioritizes data security. This approach achieves fault tolerance by duplicating data entirely across multiple physical disks. In the event of a single disk failure, the remaining mirrored disk(s) maintain an identical copy of the data, guaranteeing uninterrupted system operation [4].

While write operations in RAID 1 may incur a slight performance penalty due to the simultaneous write process across all mirrored disks, read performance demonstrably improves. This enhancement stems from the ability to access data concurrently from multiple mirrored drives within the array [4].

The primary advantages of RAID 1 lie in its inherent simplicity and exceptional fault tolerance capabilities. The loss of a single mirrored disk does not cause system disruption, provided at least one functional disk remains operational. However, a notable drawback of RAID 1 is the reduction in usable storage capacity for applications [4].

The calculation of usable storage space within a RAID 1 configuration is uncomplicated: The total available storage capacity is determined by multiplying the number of mirrored arrays by the individual disk capacity.

One notable project that aims to transform how we interact with Non-Volatile Memory Express (NVMe) devices is the Storage Performance Development Kit (SPDK). It provides a high-performance, user-space driver designed exclusively for solid-state drives (SSDs) based on NVMe technology, enabling notable performance improvements and resolving is-

TABLE I
BENCHMARK ENVIRONMENT SETUP

CPU	Intel(R) Core(TM) i7-9750H CPU @2.60GHz, Hyper-threading disabled
Memory	16GB, DDR4
Storage	256GB NVMe SSD, SKHynix HFS256GD9TNG-L3A0B
OS	Arch Linux, x86 64, 6.7.0-arch3-1
SPDK	v24.01

sues frequently found in kernel-based solutions. Asynchronous operations, lockless NVMe driver, and zero-copy are supported by the user-space driver [6, 9]. It has been shown in nu-merous studies that user-space drivers based on the Storage Performance Development Kit (SPDK) outperform kernel-based alternatives such as libaio and io-uring, especially when dealing with a variety of system workloads [1,2]. Additionally, SPDK provides a flexible block device layer that abstracts underlying storage devices so that programs can effectively communicate with them and take advantage of their full capabilities.

Asynchronous I/O is supported by various mechanisms offered by the Linux kernel and its libraries. Libaio is one of the first and most renowned libraries. It provided one of the first storage device-specific asynchronous APIs when it was first introduced in kernel version 2.6 [8]. Two essential system calls provided by libaio are io submit and io getevents. By avoiding the system's I/O cache, these calls allow non-blocking, unbuffered I/O (O DIRECT flag), potentially yield-ing large performance gains [1].

II. PROBLEM STATEMENT

While software-defined storage solutions like SPDK with libaio offer enhanced storage performance and flexibility, they often lack inherent data redundancy mechanisms. This reliance on single disk storage creates a vulnerability: a hardware failure can lead to complete data loss, causing significant downtime and potential data recovery efforts. This is par-ticularly problematic for performance-sensitive applications where data integrity and continuous operation are paramount. Traditional RAID solutions, while offering data redundancy, may introduce additional overhead that can negate the per-formance. Therefore, there exists a need for a data mirroring solution within the SPDK framework that leverages libaio's capabilities to deliver both exceptional performance and robust fault tolerance for mission-critical storage deployments.

III. PERFORMANCE EVALUATION

The experiment tests how fast the RAID1 virtual block device can perform random reads and writes under different workload conditions (number of simultaneous requests). The chosen tool (bdevperf) [10] and fixed block size (4096) ensure consistent testing environment.

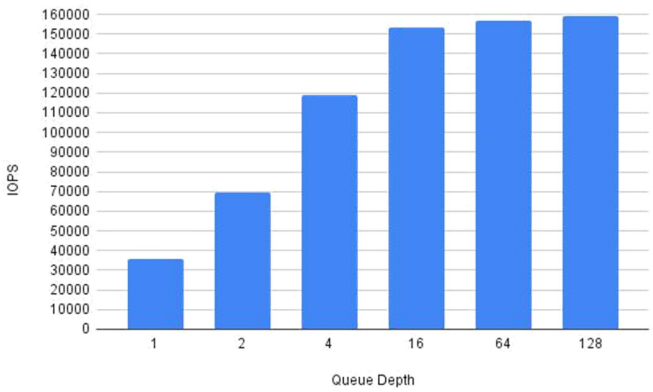

Fig. 1. Queue depth vs IOPS (randread)

Our investigation into the performance of a RAID1 vir-tual block device under a random read workload revealed a significant positive correlation between queue depth and IOPS (Input/Output Operations Per Second). Notably, a queue depth of 2 resulted in a remarkable 94% increase in IOPS, followed by a further 71% improvement at a queue depth of
4. Interestingly, IOPS gains became progressively smaller with further increases in queue depth.

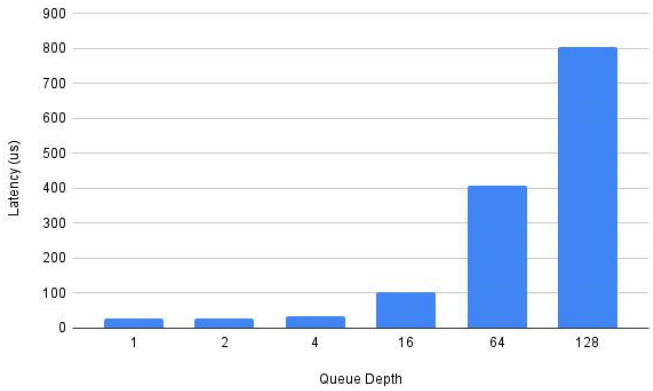

Fig. 2. Queue depth vs average latency (randread)

Figure 2 presents a bar graph illustrating the relationship between latency and queue depth. Lower latency values signify better performance. The graph reveals a critical threshold for queue depth's impact on latency.

While queue depths of 1, 2, and 4 exhibit minimal latency variations, averaging around 29 microseconds, a significant increase is observed at higher depths. Notably, a queue depth of 64 results in a latency increase of approximately 294% compared to the baseline.

Figure 3 depicts the IOPS performance for the randwrite I/O pattern, exhibiting a distinct contrast to the randread workload observed in Figure 1. Unlike randread, where increasing queue depth resulted in a notable IOPS increase, the randwrite workload displays a plateau effect. Notably, IOPS values

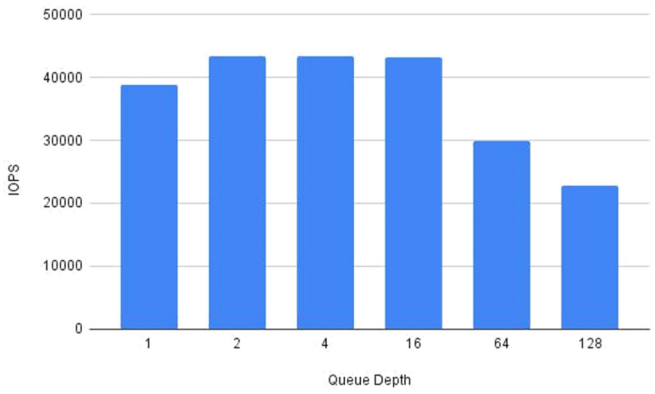

Fig. 3. Queue depth vs IOPS (randwrite)

remain relatively constant at approximately 40,000 across all queue depths. Conversely, deeper queues negatively impact performance in the randwrite scenario. The most significant degradation occurs at a queue depth of 64, with a 31% decrease in IOPS compared to the baseline. A further decrease of 24% is observed at a queue depth of 128. This phenomenon may be attributed to resource contention arising from the management of multiple queues. Additionally, the range of IOPS values for the randwrite workload is demonstrably narrower compared to the randread counterpart.

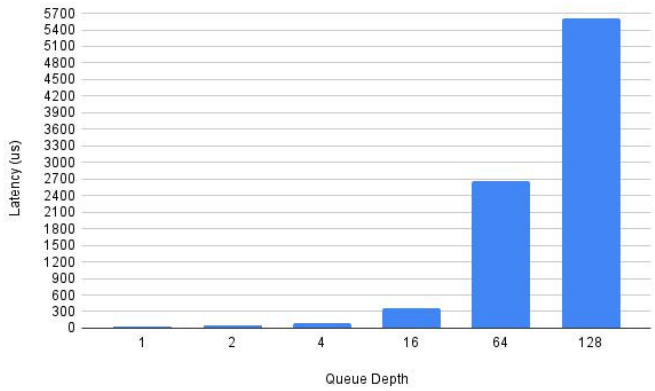

Fig. 4. Queue depth vs average latency (randwrite)

Figure 4 depicts the impact of queue depth on latency for random-write I/O operations. The data demonstrates a substantial increase in block device latency with increasing queue depth. At a queue depth of 1 latency is minimal, aver-aging approximately 25 microseconds. However, a significant rise is observed for deeper queues, with latency reaching 5611 microseconds at a queue depth of 128. This represents a notable increase of approximately 623% compared to the baseline value. Interestingly, queue depths of 1, 2, and 4 exhibit minimal variation in latency, averaging around 45 microseconds.

IV. Acknowledgements

We would like to extend my sincere gratitude to Dr. Girish P. Potdar, our esteemed project guide, for his invaluable mentorship, guidance, and unwavering support throughout the development of this project. His expertise and encouragement played a pivotal role in shaping our approach and achieving successful outcomes.

V. Conclusion

In conclusion, while traditional storage systems offer sim-plicity and affordability, they lack data redundancy. Our imple-mentation of RAID1 in SPDK using libaio offers a compelling solution for enhancing data reliability and performance in storage systems. Throughout this paper, we have explored the intricacies of integrating SPDK's efficient asynchronous I/O framework with libaio's kernel-level asynchronous I/O capa-bilities to achieve robust RAID1 setup. By leveraging SPDK's user-space design and libaio's efficient I/O handling, our implementation demonstrates significant improvements in data throughput and latency, essential for modern data-intensive applications. RAID1 configuration using AIO bdevs provides redundancy and fault tolerance, mitigating the risk of data loss due to hardware failures. Moreover, our study has highlighted the versatility and scalability of RAID1 implementation in SPDK using libaio, offering flexibility for deployment in vari-ous storage architectures and environments. Whether deployed in enterprise-grade storage systems or cloud infrastructures, our solution ensures data integrity while optimizing resource utilization. In essence, this paper underscore the significance of RAID1 implementation in SPDK using libaio as a crucial step towards achieving reliable and high-performance storage solutions, poised to address the challenges of modern data management effectively. Through our exploration and imple-mentation.

VI. Future Scope

The current focus on real-time mirroring can be comple-mented by snapshot functionality. This would allow capturing data states at specific points in time, enabling data rollback and version control. This additional layer of data protection would facilitate easier disaster recovery and cater to use cases requiring specific data versions. While RAID 1 offers excellent fault tolerance, its storage utilization is limited. Future research could investigate the feasibility of integrating RAID 10 within the SPDK framework. RAID 10 combines mirroring (RAID 1) with striping (RAID 0) across multiple disk sets. This configuration offers a balance between redun-dancy and improved storage efficiency compared to RAID 1
[11]. Evaluating the performance implications and potential benefits of incorporating RAID 10 would be a valuable area for future exploration.

These advancements would extend the applicability of the proposed solution to a wider range of storage scenarios de-manding both high performance and robust data protection with improved storage efficiency.

REFERENCES

[1] Diego Didona, Jonas Pfefferle, Nikolas Ioannou, Bernard Metzler, and Animesh Trivedi. 2022. Understanding modern storage APIs: a system-atic study of LIBAIO, SPDK, and IO-Uring. In Proceedings of the 15th ACM International Conference on Systems and Storage (SYSTOR '22). Association for Computing Machinery, New York, NY, USA, 120–127. doi: 10.1145/3534056.3534945

[2] Zebin Ren and Animesh Trivedi. 2023. Performance Characterization of Modern Storage Stacks: POSIX I/O, libaio, SPDK, and io_uring. In Proceedings of the 3rd Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems (CHEOPS '23). Association for Computing Machinery, New York, NY, USA, 35–45. doi: 10.1145/3578353.3589545

[3] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. 1994. RAID: high-performance, reliable secondary storage. ACM Comput. Surv. 26, 2 (June 1994), 145–185. https://doi.org/10.1145/176979.176981

[4] Chen, S. and Towsley, D., 1996. A performance evaluation of RAID architectures. IEEE Transactions on computers, 45(10), pp.1116-1130

[5] Gabriel Haas and Viktor Leis. 2023. What Modern NVMe Storage Can Do, and How to Exploit it: High-Performance I/O for High-Performance Storage Engines. Proc. VLDB Endow. 16, 9 (May 2023), 2090–2102. doi: 10.14778/3598581.3598584

[6] Z. Yang et al., "SPDK: A Development Kit to Build High Performance Storage Applications," 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, China, 2017, pp. 154-161, doi: 10.1109/CloudCom.2017.14.

[7] Youngjin Yu, Dongin Shin, Woong Shin, Nae Young Song, Jae-Woo Choi, Hyeong Seog Kim, Hyeonsang Eom, and Heon Young Yeom. 2014. Optimizing the Block I/O Subsystem for Fast Storage Devices. ACM Trans. Comput. Syst. 32, 2 (2014), 6:1–6:48. doi: 10. 1145/2619092

[8] Benjamin Block. "An Introduction to the Linux Kernel Block I/O Stack" https://chemnitzer.linux-tage.de/2021/media/programm/folien/165.pdf (accessed Feb 15, 2024).

[9] SPDK. "Block Device User Guide" https://spdk.io/doc/bdev.html (accessed Feb 15, 2024).

[10] Karol Latecki. "SPDK NVMe BDEV Performance Report Release 21.01" https://ci.spdk.io/download/performance-reports/SPDK_nvme_bdev_perf_report_2101.pdf (accessed Feb 15, 2024)

[11] Jin, H. and Hwang, K., 2000. Stripped mirroring RAID architecture. Journal of systems architecture, 46(6), pp.543-550.