NV-BSP: A Burst I/O Storage Pool Based on NVMe SSDs

Qiong Li[1(✉)], Dengping Wei[1], Wenqiang Gao[2], and Xuchao Xie[1]

[1] College of Computer, National University of Defense Technology, Changsha, China
qiong_joan_li@aliyun.com
[2] Beijing Memblaze Technology Co., Ltd., Beijing, China

Abstract. The High-Performance Computing (HPC) systems built for future exascale computing, big data analytics, and artificial intelligence applications raise an ever-increasing demand for high-performance and highly reliable storage systems. In recent years, as Non-Volatile Memory express (NVMe) Solid-State Drives (SSDs) are deployed in HPC storage systems, the performance penalty paid for the legacy I/O software stack and storage network architecture turns out to be non-trivial. In this paper, we propose NV-BSP, an NVMe SSD-based Burst I/O Storage Pool, to leverage the performance benefits of NVMe SSD, NVMe over Fabrics (NVMeoF) Protocol, and Remote Direct Memory Access (RDMA) networks in HPC storage systems. NV-BSP disaggregates NVMe SSDs from HPC compute nodes to enhance the scalability of HPC storage systems, employs fine-grained chunks rather than physical NVMe SSD devices as the RAID-based data protection areas, and exploits high concurrent I/O processing model to alleviate the performance overhead from lock contentions and context switches in critical I/O path . We implement NV-BSP in Linux and evaluate it with synthetic FIO benchmarks. Our experimental results show that NV-BSP achieves scalable system performance as the number of NVMe SSD and CPU core increases and obtains much better system performance compared with the built-in MD-RAID in Linux. Compared with node-local SSDs in HPC, NV-BSP provides a full system solution of storage disaggregation, delivers comparable performance, and significantly improves system reliability.

Keywords: Burst I/O Storage Pool · NVMe SSD · NVMe over Fabrics · High-Performance Computing

1 Introduction

High-Performance Computing (HPC) has proven its great power in facilitating data-driven scientific discovery [10]. Future HPC systems will be not only built for large-scale scientific computing, but also for big data analytics and artificial intelligence applications, which raises an ever-increasing demand for high-performance and highly reliable storage systems [13,15].

D. Dong et al. (Eds.): ACA 2020, CCIS 1256, pp. 178–191, 2020.
https://doi.org/10.1007/978-981-15-8135-9_13

Conventional Hard Disk Drive (HDD)-based RAID architectures [2] have been used as a key component of HPC storage systems over the past 30 years. However, due to the inherent mechanical characteristics of the rotating media in HDDs, HDD-based storage arrays are unable to meet the high IOPS, high bandwidth, and low latency requirements of the HPC applications with data-intensive problems [7,20,24]. The new emerging NAND Flash-based Solid-State Drive (SSD) provides orders of magnitude lower latency and consumes less power than HDDs [17–19]. SSDs are considered to entirely replace HDDs in future HPC storage systems. SSDs are first designed as the drop-in replacements of traditional hard disks with interfaces like SATA and SAS. However, the interface protocols were designed for hard disks which significantly limit the output performance of SSDs. This promotes the design and development of the Non-Volatile Memory express (NVMe) protocol that is capable of leveraging the internal parallelism of SSDs and reducing the software overhead in the I/O path [14,16,22,23]. NVMe SSDs have been rapidly emerging on the storage market and will be widely deployed in both data center and HPC storage systems soon.

As HPC systems have an urgent need for high-performance and highly reliable large-scale storage systems, using NVMe SSDs to build All Flash Array (AFA) can effectively meet the requirements simultaneously. However, existing storage array architectures have critical limitations in terms of software overhead in I/O path and parallelism exploitation of NVMe SSDs [18,19,21,23]. The workload characteristics of a certain HPC application can be latency-sensitive or throughput-oriented or continuously changing during the application lifespan. As the aggregate bandwidth can be easily achieved in large-scale parallel storage systems, obtaining low latency is more challenging. As the scalability of the PCI express (PCIe) bus cannot satisfy the connections of a large amount of NVMe SSDs in large-scale storage systems, NVMe over Fabrics (NVMeoF) protocol is proposed to extend the advantages of NVMe protocol to shared storage architecture [4,6]. NVMeoF offers a solution that separates storage from HPC compute node (CN) and connects storage to CN through a network fabric. Currently, NVMeoF can adequately support fabric transports like Remote Direct Memory Access (RDMA), TCP, and Fibre Channel (FC), how to efficiently integrate NVMeoF with in-house interconnection networks of specific HPC systems remains stagnant.

In this work, we consolidate the storage array trend towards integrating NVMe SSDs and NVMeoF target in a single storage server. We propose NV-BSP, an NVMe SSD-based Burst I/O Storage Pool, to leverage the performance benefits of NVMe SSD, NVMeoF Protocol, and RDMA networks in HPC storage systems. Specifically, NVM-BSP disaggregates NVMe SSDs from HPC compute nodes, which improves the storage resource utilization and enhances the scalability of HPC storage systems. NV-BSP employs fine-grained chunks rather than physical NVMe SSD devices as the RAID-based data protection areas, which avoids an entire NVMe SSD participating data reconstruction and achieves load balance without data redirection or migration. NV-BSP exploits a high concurrent I/O processing model to alleviate the performance overhead from lock

contention and context switch in critical I/O path, which enables the performance of NV-BSP increasing linearly with the number of NVMe SSDs and CPU cores.

We implement NV-BSP in Linux and evaluate it with synthetic FIO benchmarks. Our experimental results show that NV-BSP achieves scalable system performance as the number of NVMe SSD and CPU core increases and obtains better system performance compared with the build-in MD-RAID in Linux. Compared with node-local SSDs in HPC, NV-BSP provides a full system solution of storage disaggregation, delivers comparable performance, and significantly improves system reliability.

The rest of this paper is organized as follows. Section 2 provides an overview of NV-BSP, Sect. 3 describes the high concurrent I/O processing mechanism in NV-BSP. We evaluate the performance of NV-BSP in Sect. 4 and summarize the related work in Sect. 5. Finally, we conclude the paper in Sect. 6.

2 NV-BSP Overview

In this section, we give an overview of the system architecture, storage management, and storage disaggregation designs of NV-BSP.

2.1 System Architecture

The system architecture of NV-BSP is shown in Fig. 1. The hardware of NV-BSP includes the NVMe SSDs connected to CPU via PCIe bus, NVMeoF network interface for NVMeoF purpose, and other common components of storage servers, i.e., CPU, RAM, etc. The software of NV-BSP mainly composes of the storage resource manager, I/O processing handlers, and NVMeoF target. NV-BSP manages the data to underlying NVMe SSDs in the block layer in the I/O path. The storage resource manager is responsible for managing all the storage resources in NV-BSP and exporting virtual disks (VDisk) to applications, which will be discussed in detail in Sect. 2.2. I/O processing handlers will produce and activate independent threads to serve I/O requests, reconstruction requests, and error events, etc. NVMeoF target provides NVMe over RDMA communication between HPC compute nodes (CN) and VDisks in NV-BSP, which will be further discussed in Sect. 2.3.

2.2 Resource Management

Figure 2 describes the storage resource management mechanism in NV-BSP. The storage resource management mechanism can be divided into four levels include storage pool management, resource allocation, data protection, and VDisk management. In NV-BSP, NVMe SSDs are first organized as a storage pool, in which the logical address space of all the NVMe SSDs is divided into fixed-size chunks. The storage pool manager maintains the states of all the fine-grained chunks. Different from traditional RAID, NV-BSP completely separates physical resources

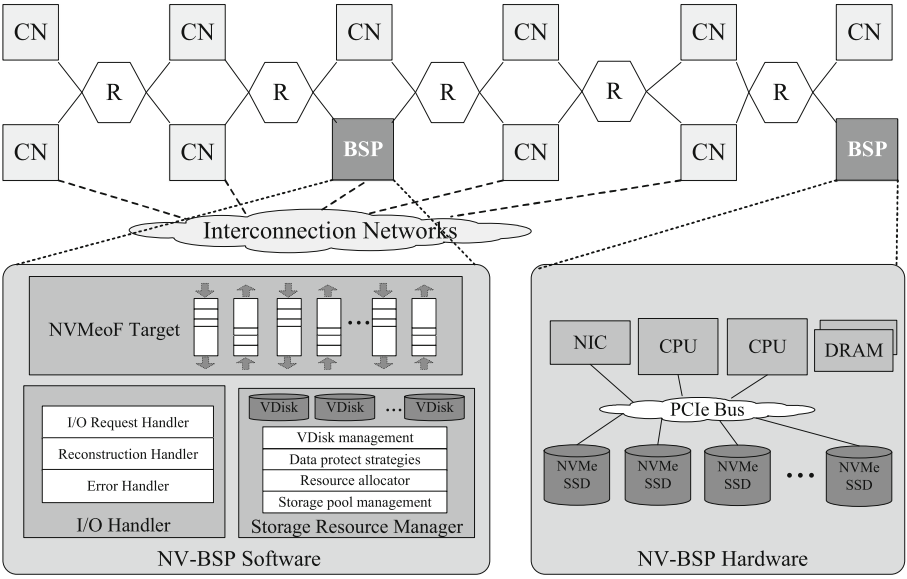

Fig. 1. NV-BSP system architecture.

and RAID-based data protection areas through chunk-level resource virtualization. Therefore, the data reconstruction operations in NV-BSP no longer relies on the entire SSD device to participate. Furthermore, through fine-grained division, the data written to the VDisks from the same NV-BSP will be evenly distributed on each NVMe SSD, which achieves load balancing among NVMe SSDs without data redirection and migration.

At the time of creating a VDisk from the storage pool, Resource allocator groups several chunks into a container which is divided into finer-stripes based on the configured data protection level (e.g.., RAID 0/1/5/6). Stripe is the basic granularity of read and write operations in NV-BSP. VDisk is a collection of containers and will be exported as a logical volume in the operating system. The corresponding logical volume of the VDisk will provide storage services for upper-layer applications through a standard block device interface.

2.3 Storage Disaggregation

In NV-BSP, a single storage pool can export several VDisks simultaneously and the VDisks can be carved into NVMe namespaces with each namespace allocated to a specific HPC compute node. The I/O processing handlers are responsible for serving I/O requests concurrently. In NV-BSP, the NVMeoF target dynamically and arbitrarily attaches virtual disks with needed capacity and performance via QoS management technology directly to the compute node where the application runs on. As NVMeoF initiators, computing nodes send NVMeoF read/write commands through the RDMA network to the destination NV-BSP. The NVMeoF

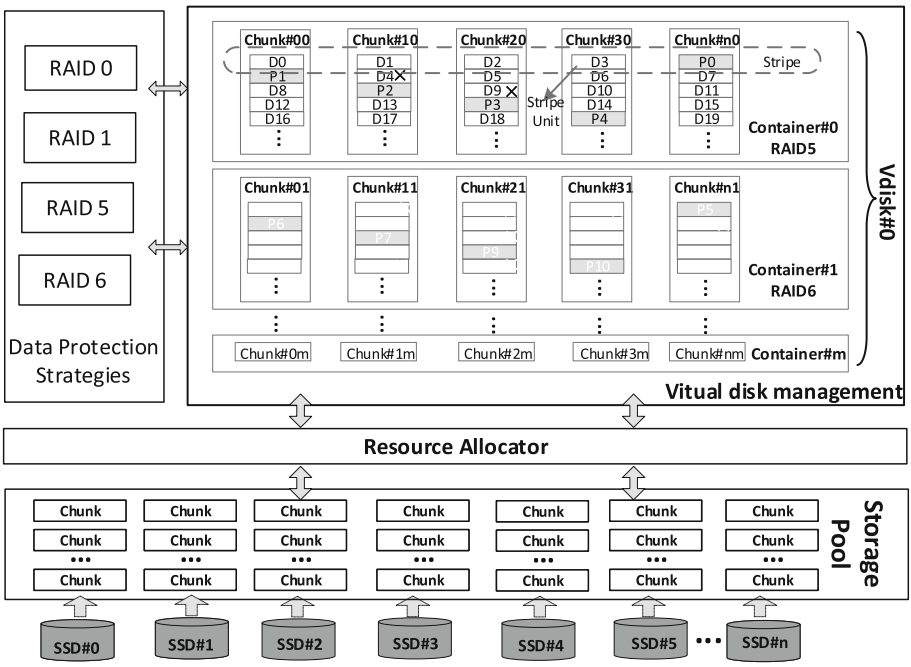

Fig. 2. Storage resource manager in NV-BSP.

target in NV-BSP parses the received NVMeoF commands and converts them into NVMe commands to a VDisk that exported from NV-BSP. The storage resource manager in NV-BSP manages all the VDisks, and finally transmits each NVMe command into multiple NVMe commands to the corresponding NVMe SSDs. Each NVMe SSD completes the subsequent read data from the NVMe SSD to the memory, or write data into the NVMe SSD, finally transmits the read/write completion message to NV-BSP. NV-BSP combines the arrived completion messages into a single NVMe response. Finally, the NVMeoF target in NV-BSP returns the NVMeoF response to corresponding compute nodes over the NVMeoF storage network.

3 High Concurrent I/O Processing

In this section, we describe the details of the high concurrent I/O processing model in NV-BSP.

3.1 Task Grouping

In NV-BSP, task grouping is designed to achieve high concurrent I/O processing. As NV-BSP serves as both NVMeoF target and RAID array simultaneously, significant CPU competitions between the two tasks will be introduced when

NVMeoF target and RAID array services are all enabled. In NV-BSP, CPU cores are divided into two different groups, i.e., NVMeoF Target Group (NT-Group) and RAID Array Group (RA-Group). The CPU cores in NT-Group will only be assigned for the NVMeoF target while that in RA-Group only assigned for the RAID array task. In this case, two different tasks will not run on the same CPU core anymore, which effectively alleviates CPU conflicts caused performance overhead and improves the concurrency of I/O processing.

In RA-Group, each CPU core occupies an independent data structure when performing an I/O handler. Thus, the CPU cores in RA-Group will no longer need to compete for a data structure and the lock contention overhead is eliminated. As shown in Fig. 3, for the I/O handlers that perform RAID tasks on N cores (indicated as CPU_1 to CPU_N), each I/O handler uses an independent data structure to avoid lock contentions among CPU cores, which enables the I/O performance increases linearly with the increase of the number of CPU cores in NV-BSP.

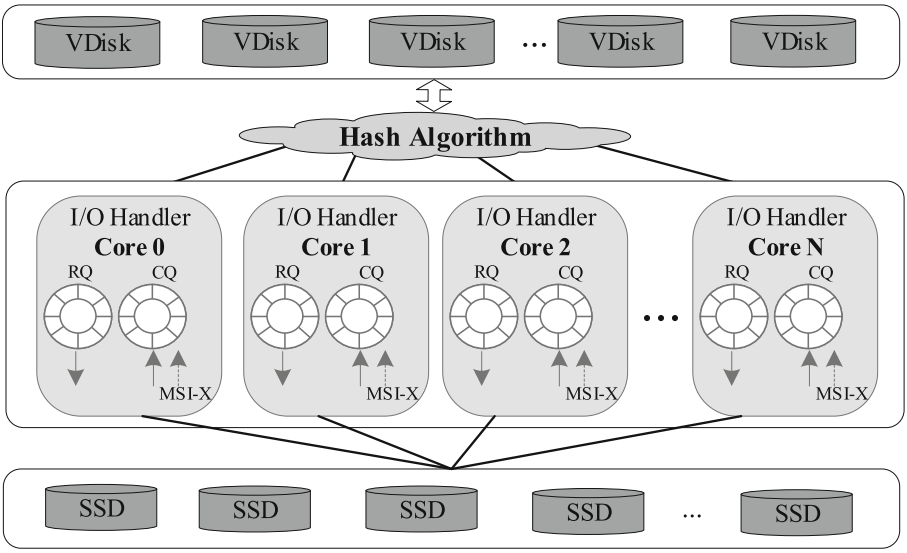

Fig. 3. High concurrent I/O processing model in NV-BSP.

3.2 I/O Handler Allocation

In NV-BSP, I/O handlers are responsible for handling I/O requests, including read and write requests from application and data reconstruction requests within NV-BSP. Each I/O handler thread processes I/O requests of different logical address areas in VDisk. The logical address space of a VDisk is divided into multiple regions that do not overlap with each other. I/O requests to a region are

processed by the corresponding I/O handler thread. As shown in Fig. 3, when an application accesses the VDisk in NV-BSP, several I/O handler threads will wake up according to the hash algorithm. Since there is no access correlation between the requests handled by different I/O handlers, these I/O handler threads can run on multiple CPU cores concurrently. Furthermore, each I/O Handler can be bounded to a dedicated CPU core to reduce the performance overhead caused by context switches.

3.3 I/O Request Processing

In NV-BSP, each I/O request is processed in two phases. In the first phase, I/O request is distributed into I/O handler command queue according to its target logical address region. In the second phase, the corresponding I/O handler processes the request based on the RAID request handling tree model, as shown in Fig. 4. The I/O request processing acts in accordance with the tree changes. The tree grows when an I/O request arrives at the corresponding VDisk and the tree shrinks as I/O requests being served successfully.

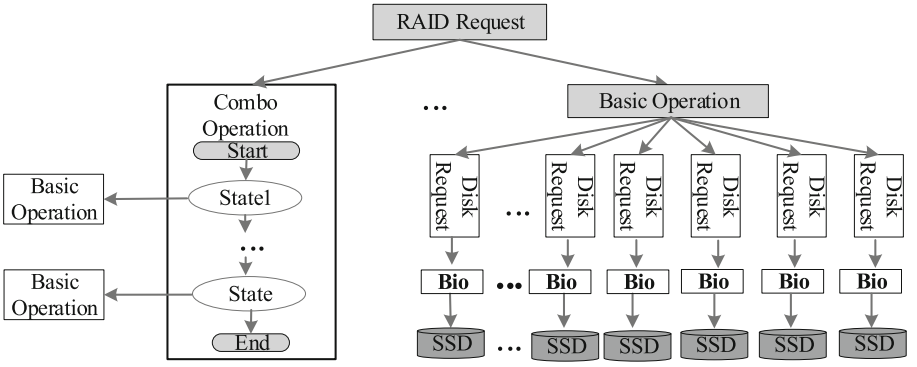

Fig. 4. I/O request processing in NV-BSP.

All the I/O operations to NV-BSP can be divided into two categories, i.e., basic operation and combo operation. A basic operation only involves in a single stripe and can be easily accomplished. A combo operation consists of one or more basic operations. For example, for the reconstruction combo operation, it consists of two basic operations, i.e., read a stripe to memory and write the reconstructed data from memory to the spare space. Each combo operation corresponds to a sequentially executed state machine. The combo operation is converted into multiple basic operations and executed recursively according to the state machine until all the basic operations are accomplished.

4 Performance Evaluation

In this section, we evaluate the detailed behavior of NV-BSP under synthetic FIO workloads.

4.1 Experimental Setup

The experimental setup consists of a server with two Intel Xeon Gold 6128 CPUs (each with 6 physical cores and 12 logical cores in Hyper-Threading mode), 192 GB of DDR4 RAM, 8 NVMe SSDs with 1.8 TB NAND Flash. The operating system is CentOS Linux 7.7 with the kernel version of 4.19.46. NV-BSP is implemented as a kernel module in Linux and rely on FIO of version 3.7 for performance evaluation. In the following experiments, all the workloads generated by FIO use Linux Asynchronous I/O (libaio) engine and enable direct IO.

4.2 Experiment Results

Performance Scalability Measurement. To understand the system performance of NV-BSP with a different number of NVMe SSDs, we measure the IOPS of the VDisks from storage arrays equipped with 3 to 8 NVMe SSDs. In this experiment, we configure FIO workloads to 4 KB read-intensive (30% write and 70% read) and write-intensive (70% write and 30% read) I/O, 8 threads with 64 queue depth per thread.

Figure 5 shows the comparison of the IOPS of VDisks from the NV-BSPs with a different number of NVMe SSDs. NV-BSP generally achieves obvious performance improvement for both read-intensive and write-intensive workloads as more NVMe SSDs are equipped in NV-BSP. Specifically, for the read-intensive workload, IOPS improves 86.94% when the number of NVMe SSDs increases from 3 to 6, and 97.82% when the number of NVMe SSDs increases from 4 to 8. For the write-intensive workload, IOPS improves 87.04% when the number of NVMe SSDs increases from 3 to 6, and 97.98% when the number of NVMe SSDs increases from 4 to 8.

To further evaluate the performance benefits from the high concurrent I/O processing design in NV-BSP, we configure a different number of I/O handlers in a VDisk from an NV-BSP epuipped with 8 NVMe SSDs and measure both the IOPS under 4 KB random read/write and the bandwidth under 128 KB sequential read/write workloads of the VDiks.

As shown in Fig. 6, both random read and random write performance of the VDisk improves linearly until the number of I/O handlers increases to 16, where the 4 KB random performance of the VDisk reaches the bottleneck and appears little improvement. Figure 7 depicts the sequential read and write bandwidth of the VDisks with a different number of I/O handlers. Similar to the random read and write performance, NV-BSP shows a good acceleration ratio with the increasing number of I/O handlers, especially for the sequential read performance. We can clearly see that the maximum read bandwidth of a single VDisk can be more than 20 GB/s, which indicates outstanding performance scalability of NV-BSP.

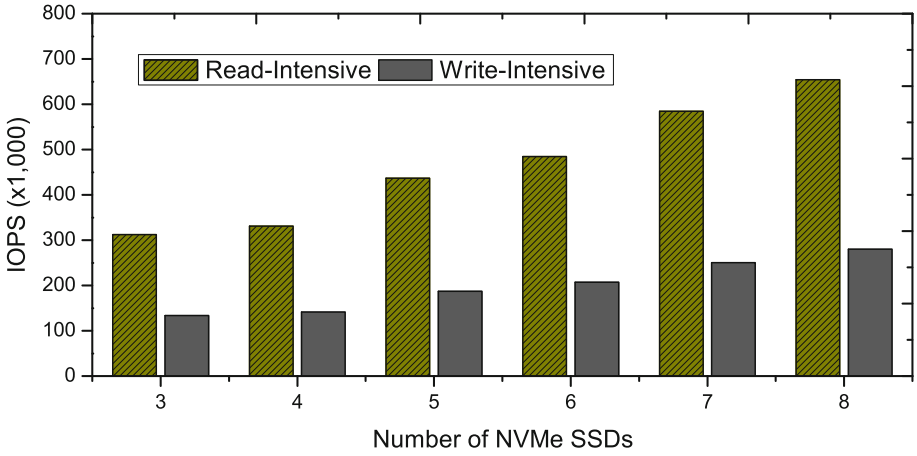

Fig. 5. IOPS comparison for different number of NVMe SSDs in NV-BSP.

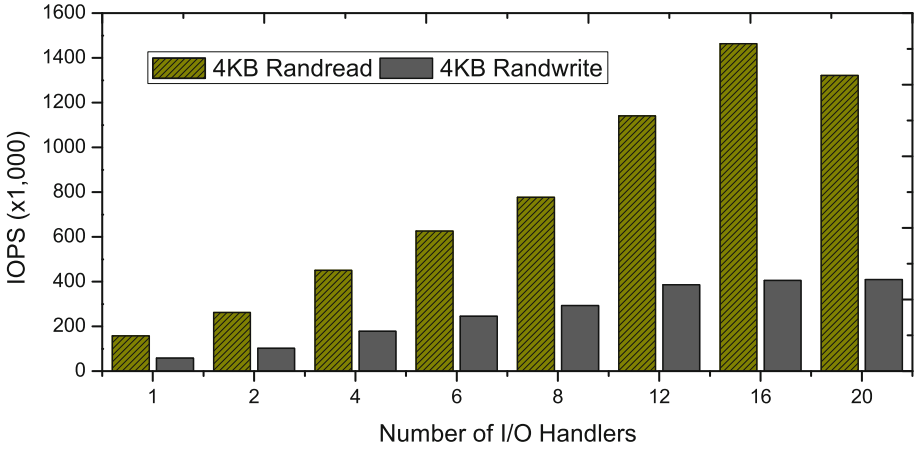

Fig. 6. IOPS comparison for different number of I/O Handlers in NV-BSP.

Performance Comparison with MD-RAID. We compare the performance of NV-BSP with the build-in MD-RAID in Linux. Two different VDisks created by NV-BSP and MD-RAID are evaluated in this experiment. Figure 8 and Fig. 9 show the average I/O latency and bandwidth comparisons for different I/O request sizes respectively. As shown in Fig. 8, the average I/O latency of NV-BSP is much lower than that of MD-RAID. As the I/O request size increases from 4 KB to 128 KB, the average I/O latency of NV-BSP is at least 4.75x lower than that of MD-RAID. Similarly, the bandwidth of the VDisk from NV-BSP is much higher than that of MD-RAID. As the I/O request size increases from 4 KB to 128 KB, the bandwidth of NV-BSP is 1.64x to 11.06x higher than that

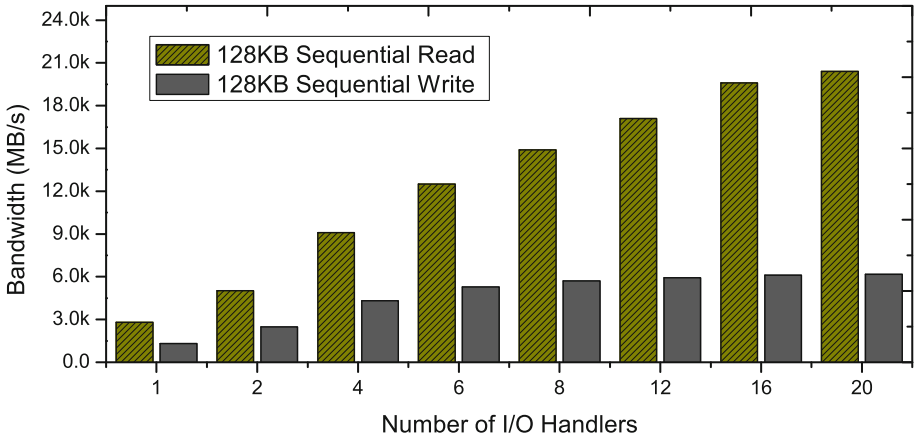

Fig. 7. Bandwidth comparison for different number of I/O Handlers in NV-BSP.

of MD-RAID. Apparently, both the I/O latency and bandwidth confirm that the performance of NV-BSP significantly outperforms that of MD-RAID.

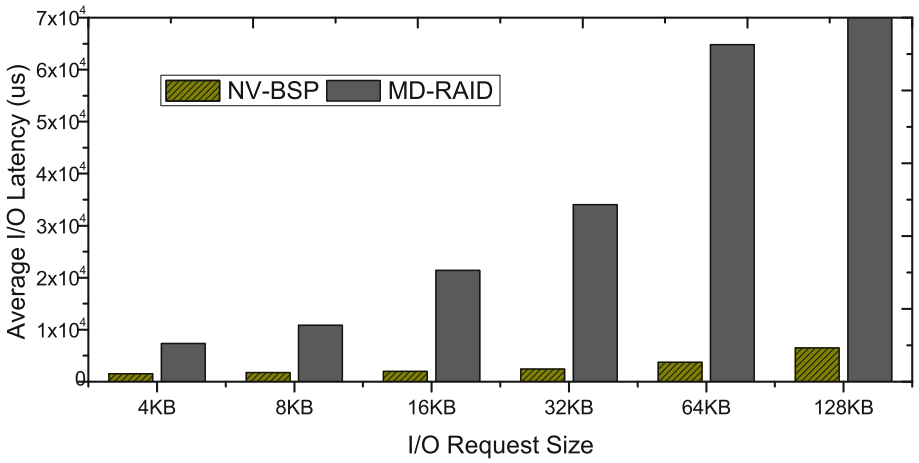

Fig. 8. Latency comparison for different I/O request sizes.

5 Related Work

The increasing number of CPU cores in modern storage servers enables a single storage server to host a large amount of high-performance NVMe SSDs [5,8]. However, the scalability issues of storage software stack significantly prevent

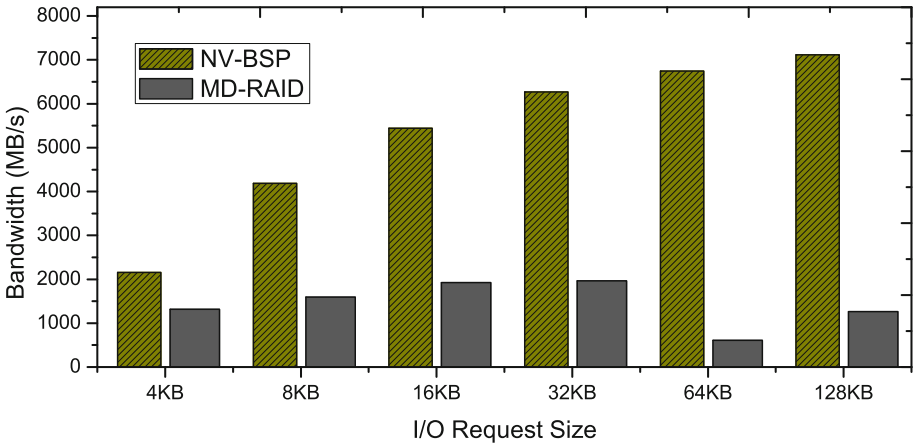

Fig. 9. Bandwidth comparison for different I/O request sizes.

the output performance of the NVMe SSDs on a server. Several studies have tried to sequentially write data into the storage array to improve system performance and achieve wear-leveling among NVMe SSDs [11]. SWAN [7] introduces a log-structured storage management logic at the host level to sequentialize the written data to RAID stripes. SWAN is specially designed for NVMeoF target by balancing the output performance of NVMe SSDs and network interface performance. Purity [3] developed by Pure Storage proposes to adopt an LSM-tree [12] based log-structured indexes and data layouts in storage array, thus data can be written in large sequential chunks for better performance. Besides, purity also integrates compression and deduplication to make better use of NVMe SSD capacity. Different from purity and SWAN, NV-BSP proposes to alleviate the overhead from lock contentions and context switches in the critical I/O path to achieve the high concurrent I/O processing.

Integrating flash storage disaggregation techniques (e.g.., iSCSI and NVMeoF) into storage array designs shows great benefits of improving storage resource utilization and aggregating the performance of a bunch of storage devices [1,9]. Storage array manufacturers like Huawei, Pure Storage, Apeiron Data Systems, Kaminario, Pavilion Data Systems have released their storage array products and solutions that integrate NVMeoF target logic respectively. For example, OceanStor Dorado V6 uses NVMe over FC and RDMA as the front end interfaces of RAID 2.0 storage pools. E8 Storage combines the high performance of NVMe drives, the high availability and reliability of centralized storage, and the high scalability of scale-out solutions in a single storage array. The E8-D24 and E8-S10 products of E8 Storage can deliver up to 10 million IOPS with 40 GBps throughput using 100 GbE or 100 Gbps InfiniBand connectivity.

6 Conclusion

This paper presents NV-BSP, an NVMe SSD-based Burst I/O Storage Pool that leverages the performance benefits of NVMe SSD, NVMeoF Protocol, and RDMA networks in HPC storage systems. NV-BSP disaggregates NVMe SSDs from HPC compute nodes to enhance the scalability of HPC storage systems, employs fine-grained chunks rather than physical NVMe SSD devices as the RAID-based data protection areas, and exploits high concurrent I/O processing model to alleviate the performance overhead from lock contentions and context switches in critical I/O path. We evaluated and analyzed the detailed behavior of NV-BSP. Compared with node-local SSDs in HPC, NV-BSP provides a full system solution of storage disaggregation, delivers comparable performance, and significantly improves system reliability. Compared with the built-in MD-RAID in Linux, NV-BSP achieves much better system performance. In future work, we will study global wear-leveling in NV-BSP to enhance the endurance of NVMe SSDs and QoS management scheme to create VDisk from NV-BSP with customized capacity and performance.

Acknowledgment. The authors would like to thank the anonymous reviewers. This work was supported in part by the Advanced Research Project of China under grant 31511010202 and the National Key Research and Development Program of China under Grant 2018YFB0204301.

References

1. Amvrosiadis, G., et al.: Data storage research vision 2025: report on NSF visioning workshop held May 30–June 1, 2018. Technical report, USA (2018)
2. Balakrishnan, M., Kadav, A., Prabhakaran, V., Malkhi, D.: Differential raid: rethinking raid for SSD reliability. ACM Trans. Storage **6**(2), 1–22 (2010). https://doi.org/10.1145/1807060.1807061
3. Colgrove, J., Davis, J.D., Hayes, J., Miller, E.L., Sandvig, C., Sears, R., et al.: Purity: building fast, highly-available enterprise flash storage from commodity components. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD 2015, pp. 1683–1694. Association for Computing Machinery, New York (2015). https://doi.org/10.1145/2723372.2742798
4. Guz, Z., Li, H.H., Shayesteh, A., Balakrishnan, V.: NVMe-over-fabrics performance characterization and the path to low-overhead flash disaggregation. In: Proceedings of the 10th ACM International Systems and Storage Conference, SYSTOR 2017. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3078468.3078483
5. Jackson, A., Turner, A., Weiland, M., Johnson, N., Perks, O., Parsons, M.: Evaluating the arm ecosystem for high performance computing. In: Proceedings of the Platform for Advanced Scientific Computing Conference, PASC 2019. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3324989.3325722
6. Jin, Y.T., Ahn, S., Lee, S.: Performance analysis of NVMe SSD-based all-flash array systems, pp. 12–21 (2018)

7. Kim, J., Lim, K., Jung, Y., Lee, S., Min, C., Noh, S.H.: Alleviating garbage collection interference through spatial separation in all flash arrays, pp. 799–812 (2019)
8. Kim, J., Ahn, S., La, K., Chang, W.: Improving I/O performance of NVMe SSD on virtual machines. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, SAC 2016, pp. 1852–1857. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2851613.2851739
9. Klimovic, A., Litz, H., Kozyrakis, C.: Reflex: remote flash local flash. In: Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2017, pp. 345–359. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3037697.3037732
10. Liao, X., Xiao, L., Yang, C., Lu, Y.: Milkyway-2 supercomputer: system and application. Front. Comput. Sci. **8**(3), 345–356 (2014)
11. Oh, Y., Choi, J., Lee, D., Noh, S.H.: Improving performance and lifetime of the SSD raid-based host cache through a log-structured approach. In: Proceedings of the 1st Workshop on Interactions of NVM/FLASH with Operating Systems and Workloads, INFLOW 2013. Association for Computing Machinery, New York (2013). https://doi.org/10.1145/2527792.2527795
12. Oneil, P., Cheng, E.Y.C., Gawlick, D., Oneil, E.: The log-structured merge-tree (LSM-tree). Acta Informatica **33**(4), 351–385 (1996)
13. Patel, T., Byna, S., Lockwood, G.K., Tiwari, D.: Revisiting I/O behavior in large-scale storage systems: the expected and the unexpected (2019)
14. Qian, J., Jiang, H., Srisa-An, W., Seth, S., Skelton, S., Moore, J.: Energy-efficient I/O thread schedulers for NVMe SSDs on NUMA. In: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2017, pp. 569–578. IEEE Press (2017). https://doi.org/10.1109/CCGRID.2017.24
15. Shi, X., Liu, W., He, L., Jin, H., Li, M., Chen, Y.: Optimizing the SSD burst buffer by traffic detection. ACM Trans. Archit. Code Optim. **17**(1), 1–26 (2020). https://doi.org/10.1145/3377705
16. Tavakkol, A., et al.: Flin: enabling fairness and enhancing performance in modern NVMe solid state drives. In: Proceedings of the 45th Annual International Symposium on Computer Architecture, ISCA 2018, pp. 397–410. IEEE Press (2018). https://doi.org/10.1109/ISCA.2018.00041
17. Xie, X., Li, Q., Wei, D., Song, Z., Xiao, L.: ECAM: an efficient cache management strategy for address mappings in flash translation layer. In: Wu, C., Cohen, A. (eds.) APPT 2013. LNCS, vol. 8299, pp. 146–159. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45293-2_11
18. Xie, X., Wei, D., Li, Q., Song, Z., Xiao, L.: CER-IOS: internal resource utilization optimized I/O scheduling for solid state drives. In: 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), pp. 336–343. IEEE (2015)
19. Xie, X., Xiao, L., Wei, D., Li, Q., Song, Z., Ge, X.: Pinpointing and scheduling access conflicts to improve internal resource utilization in solid-state drives. Front. Comput. Sci. Chin. **13**(1), 35–50 (2019)
20. Xie, X., Yang, T., Li, Q., Wei, D., Xiao, L.: Duchy: achieving both SSD durability and controllable SMR cleaning overhead in hybrid storage systems. In: Proceedings of the 47th International Conference on Parallel Processing, p. 81. ACM (2018)

21. Xu, G., et al.: RFPL: a recovery friendly parity logging scheme for reducing small write penalty of SSD raid. In: Proceedings of the 48th International Conference on Parallel Processing, ICPP 2019. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3337821.3337887
22. Xu, Q., et al.: Performance characterization of hyperscale applications on on NVMe SSDs. In: Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 2015, pp. 473–474. Association for Computing Machinery, New York (2015). https://doi.org/10.1145/2745844.2745901
23. Xu, Q., et al.: Performance analysis of NVMe SSDs and their implication on real world databases. In: Proceedings of the 8th ACM International Systems and Storage Conference, SYSTOR 2015. Association for Computing Machinery, New York (2015). https://doi.org/10.1145/2757667.2757684
24. Zhang, B., Yang, M., Xie, X., Du, D.H.C.: Idler: I/O workload controlling for better responsiveness on host-aware shingled magnetic recording drives. IEEE Trans. Comput. **69**(6), 777–788 (2020)