Towards High Performance Disaggregated Flash Storage with Programmable NIC

Yuanwei Lu

Microsoft Research and USTC

Abstract

Existing storage disaggregation solutions all require remote host memory as a proxy to access storage devices. This creates a single bottleneck between CPU root complex and PCIe switch and induces too much DRAM footprint. Moreover, those software solutions introduce high tail latency due to commands batching, system interrupt and context switch. This latency overhead is non-trivial for fast NVMe flash storages. We propose S-Direct to direct access remote NVMe flash storage using a programmable NIC. This new design bypasses remote CPU and doesn't require host memory as a proxy, thus removing the single bottleneck. S-Direct enables peer-to-peer communication among flash devices which brings a new dimension of flexibility to create new storage access commands. Our preliminary results show that S-Direct achieves up to 8.2x tail latency reduction without throughput drop compared with previous approaches.

1. Introduction

Storage disaggregation separates computation and storage into standalone resource pools and uses network fabric to interconnect them. This disaggregated design addresses the challenge of imbalanced resource requirements and the resulting over-provisioning of datacenter resources, thus is widely used in datacenters [4–6].

Previous storage disaggregation mainly deals with hard disks which have high access latency (several ms) and low IOPS (100s). Recently, NVMe flash storage devices, such as PCIe-based solid state drives (SSDs) and next-generation technology like 3D XPoint storage are now significantly faster than disks in terms of both latency (10s μs) and throughput (100s KIOPS). Previous solutions [2, 3, 7] fall short when facing this new trend for storage disaggregation: 1) existing solutions access data from a remote flash pool using the remote host memory as a proxy. And many storage data operations like duplication and encryption among different devices are also done through remote host memory. This adds pressure to the link throughput between CPU root complex and PCIe switch [1], especially when accessing the high throughput NVMe flash storage. Moreover, the CPU DRAM footprint can be quite large to sustain line rate; 2) existing software solutions with millisecond level tail latency cannot meet the stringent latency requirement of fast flash storages, especially when application SLA is within sub-millisecond.

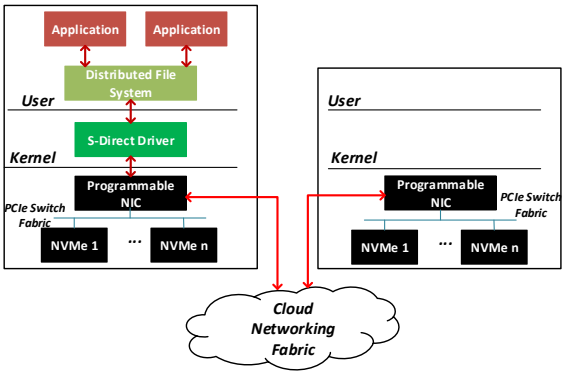

Figure 1: System overview of S-Direct.

In this paper, we propose a high performance disaggregated flash storage design called S-Direct to solve those problems using programmable NICs (*e.g.*, FPGA with ethernet ports). By connecting several programmable NICs to the PCIe switch, S-Direct enables direct access to storage devices which can bypass the remote host CPU. And S-Direct provides peer-to-peer communication among PCIe storage devices. This incurs zero CPU and DRAM on remote hosts and removes the single bottleneck between root complex and PCIe switch. S-Direct also demonstrates low and stable tail latency. Furthermore, S-Direct adds another dimension of flexibility to upper applications, *i.e.* besides conventional read/write commands, S-Direct can provide new commands like *copy between devices* and *multicast among devices*, which can potentially accelerate storage applications.

2. S-Direct

S-Direct assumes underlying flash storage devices supports NVMe protocol [1], which is a recent proposed efficient

[1] We assume all storage devices are connected to a PCIe switch which is directly connected with the CPU root complex.

standard for accessing PCIe SSDs. The system architecture of S-Direct is shown in Fig. 1. Applications access remote storage via a distributed file system which instructs the underlying programmable NIC via a S-Direct driver. S-Direct encapsulates certain NVMe commands into network packets and sends them to remote end. On receiving a command, the remote programmable NIC executes it by communicating directly with flash devices via the NVMe protocol. The responses of the commands are further encapsulated by the programmable NIC and sent back.

S-Direct driver. To enable programmable NIC to communicate with the NVMe PCIe flash devices, S-Direct driver creates a pair of IO queue on the programmable NIC for each NVMe device, *i.e.*, Submission Queue (SQ) and Completion Queue (CQ). Then NVMe devices communicate with the programmable NIC via those IO queue pairs. S-Direct driver is also responsible for translating and passing upper layer commands to local programmable NIC which will in turn transmit those commands to remote end.

NIC NVMe logic. The programmable NIC implements NVMe command encapsulation and decapsulation. Also, it submits commands to the submission queue and rings the NVMe controller doorbell without batching. Due to limited on-chip memory size (severl MBs), a cut-through solution is employed for data transmission, *i.e.* the programmable NIC will not buffer any data in its on-chip memory.

Flexible new commands. With the programmable hardware, S-Direct provides another dimension of flexibility to upper applications. A class of new commands can be enabled by peer-to-peer communication between NVMe devices. S-Direct creates a FIFO queue for each communication pair and employs credit-based flow control to ensure that the FIFO queue length is within certain range, thus minimizing on-chip memory footprint. With peer-to-peer communication, S-Direct can provide new commands like *Copy(ssd1_address1, ssd2_address2, length)* to do data duplication and *MultiCast(ssd1_address1, ssd2_address2, ..., ssd_k_address_k, length)* to do data multicast among multiple SSDs. Another class of new commands can be enabled by leveraging the computation ability of the programmable NIC. For example, S-Direct can provide command like *add_write(ssd1_address1, ssd1_address2)* to add the value at two locations and write back.

3. Preliminary Results

We have implemented part of S-Direct driver and NVMe logic in programmable NIC. With these implemented modules, programmable NIC can communicate with NVMe SSD devices (Intel DC P3700) via the NVMe protocol without CPU involvement. We have measured the IOPS and 90th tail latency of S-Direct, SPDK and Linux kernel when they issue random 4KB read/write commands to local NVMe devices. All three methods can read/write at peak throughput of the device, we avoid the detailed results due to space limitation. For latency, as shown in Fig. 2, S-Direct demonstrates low 90th percentile latency compared with Linux kernel and SPDK. Specifically, for 4KB random read operations, S-Direct is 2.8x and 3.7x lower than Linux kernel and SPDK respectively. For 4KB write operations, S-Direct is 3.7x and 8.2x lower than the other two approaches. By implementing the IO queues in hardware, S-Direct avoids system interrupt and context switching latency as well as software batching overhead, thus achieving relatively low latency. Notice that SPDK has larger tail latency than Linux kernel, this is because SPDK uses large command batching for issuing commands to underlying storage, thus incurs larger tail latency.

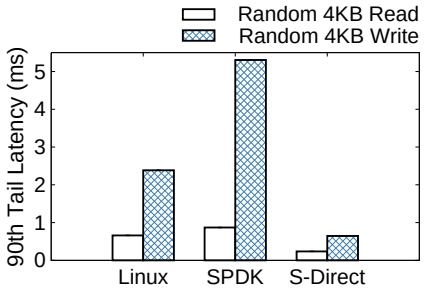

Figure 2: 90th percentile tail for 4KB random access.

4. Related work

iSCSI. Internet Small Computer Systems Interface (iSCSI) is a networking standard for carrying SCSI commands over IP networks to access remote storage. SCSI protocol is designed for SCSI devices, and works poorly on recently prevalent NVMe capable devices in terms of both latency and throughput because of its limited data parallelism.

NVMe over Fabrics. NVMe over fabrics is a standard for accessing remote NVMe devices by encapsulate NVMe commands and responses into data packets over certain fabrics. NVMe over fabrics still uses remote memory as a proxy to access remote NVMe devices, this incurs higher latency and higher CPU overhead than S-Direct.

SPDK. The Storage Performance Development Kit (SPDK) provides a set of tools and libraries for writing high performance user-mode storage applications. It avoids kernel context switches and eliminates interrupt handling overhead by moving all of the necessary drivers into userspace and operating in a polled mode. However, it requires batching of commands to achieve high throughput, which leads to high tail latency.

5. Conclusion

This paper presents S-Direct to access remote flash storage which bypasses remote CPU and doesn't require host DRAM as a proxy. S-Direct provides high throughput and low latency with zero remote CPU and DRAM overhead. Furthermore, S-Direct enables peer-to-peer communication among flash storages and provides a new dimension of flexibility to create new types of commands.

References

[1] Non volatile memory express, . URL http://www.nvmexpress.org/.

[2] Nvme over fabrics, . URL http://www.nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf.

[3] Spdk nvme over fabrics target. URL http://www.spdk.io/doc/nvmf.html.

[4] J. Hamilton. Internet scale storage. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 1047–1048. ACM, 2011.

[5] J. Heiliger. Building efficient data centers with the open compute project. *Facebook Engineering Notes*, 2011.

[6] A. Klimovic, C. Kozyrakis, E. Thereska, B. John, and S. Kumar. Flash storage disaggregation. In *Proceedings of the Eleventh European Conference on Computer Systems*, page 29. ACM, 2016.

[7] J. Satran and K. Meth. Internet small computer systems interface (iscsi). 2004.