

Performance Evaluation of Disaggregated Storage Linux I/O Scheduler

Jaehyun Hwang, Qizhe Cai, Rachit Agarwal
Cornell University

August 17, 2020

1 Evaluation Setup

We use a testbed with two servers (host and target), each with 100Gbps links, directly connected without any intervening switches. Both servers have the same hardware/software configurations as shown in Table 1.

Table 1: Experimental setup used in our evaluation.

H/W configurations	
CPU	4-socket Intel Xeon Gold 6128 CPU @ 3.4GHz 6 cores per socket, NUMA enabled (4 nodes)
Memory	256GB of DRAM
NIC	Mellanox ConnectX-5 EX (100G) TSO/GRO=on, LRO=off, DIM disabled Jumbo frame enabled (9000B)
NVMe SSD	1.6TB of Samsung PM1725a
S/W configurations	
OS	Ubuntu 16.04 (Linux kernel 5.8.0)
Applied patches	Batching dispatch [1] nvme-tcp optimizations [2–4]
IRQ	irqbalance enabled
FIO	Block size=4KB, Direct I/O=on I/O engine=libaio, gtod_reduce=off CPU affinity enabled

2 Experimental Results

In all experiments, the host-side applications (FIO) access the target-side storage devices (NVMe SSD or RAM block device) over NVMe-over-TCP. We compare our Disaggregated Storage I/O Scheduler (DSS) with “Noop” I/O scheduler. (An early version of the DSS idea is described and evaluated in [5].)

Single core performance:

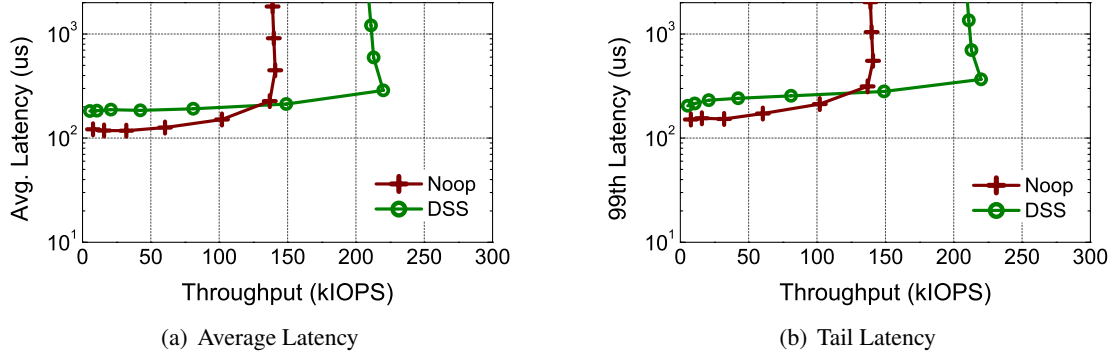


Figure 1: Target device: **NVMe SSD** (4KB random read)

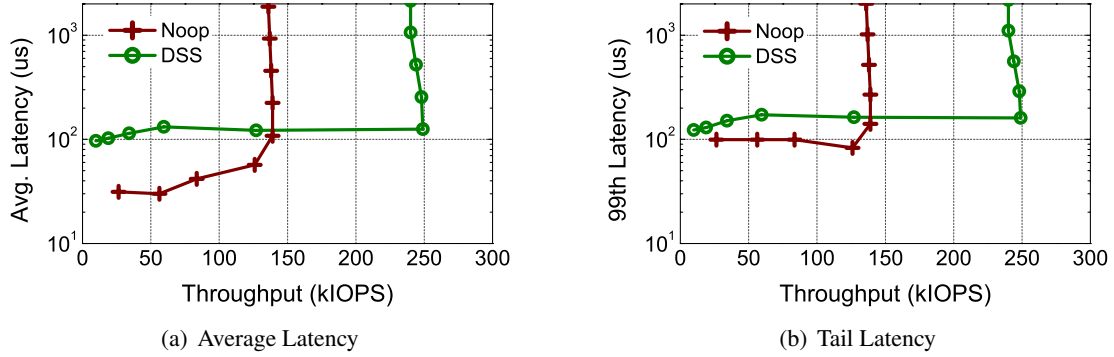
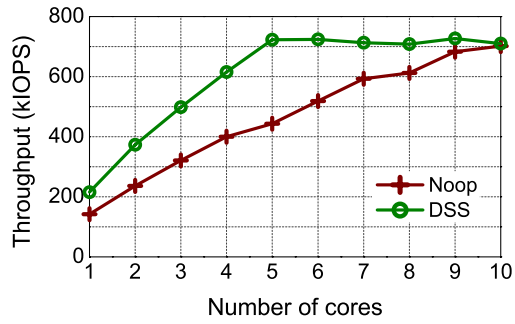
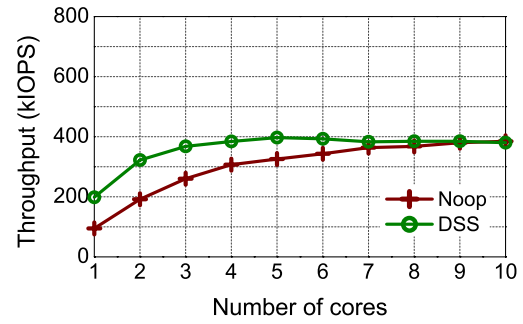


Figure 2: Target device: **RAM block device** (4KB random read)

Scalability with number of CPU cores:

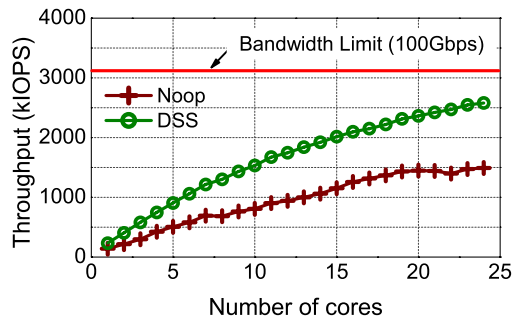


(a) Random read

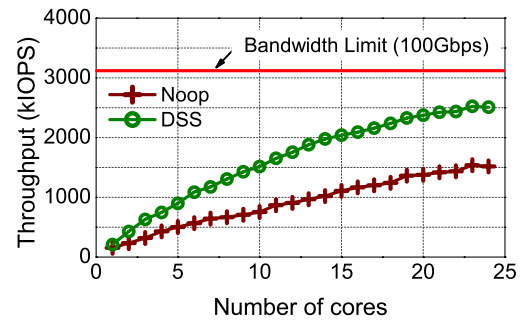


(b) Random write

Figure 3: Target device: **NVMe SSD** (4KB random read/write, I/O depth = 64)



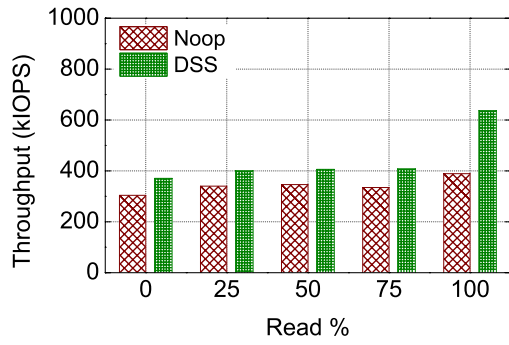
(a) Random read



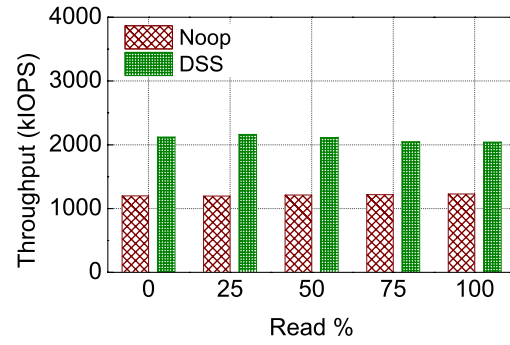
(b) Random write

Figure 4: Target device: **RAM block device** (4KB random read/write, I/O depth = 64)

Performance with varying read/write ratios:



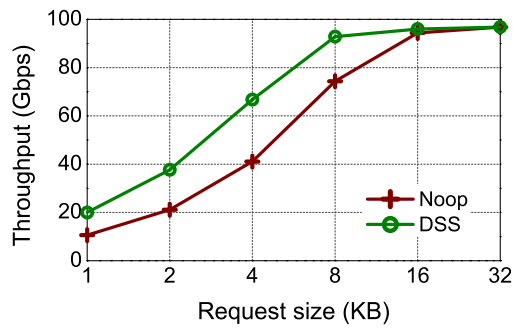
(a) **NVMe SSD** (4 cores)



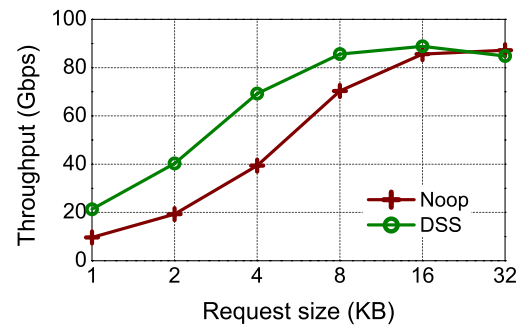
(b) **RAM block device** (16 cores)

Figure 5: 4KB mixed random read/write (I/O depth = 64)

Performance with varying request sizes:



(a) Random read



(b) Random write

Figure 6: Target device: RAM block device (I/O depth = 64, 16 cores)

References

- [1] <https://www.spinics.net/lists/linux-block/msg55860.html>.
- [2] <http://git.infradead.org/nvme.git/commit/122e5b9f3d370ae11e1502d14ff5c7ea9b144a76>.
- [3] <http://git.infradead.org/nvme.git/commit/86f0348ace1510d7ac25124b096fb88a6ab45270>.
- [4] <http://git.infradead.org/nvme.git/commit/15ec928a65e0528ef4999e2947b4802b772f0891>.
- [5] J. Hwang, Q. Cai, A. Tang, and R. Agarwal. TCP \approx RDMA: CPU-efficient Remote Storage Access with i10. In *USENIX NSDI*, 2020.