ReFlex: Remote Flash ≈ Local Flash

Ana Klimovic * Heiner Litz* Christos Kozyrakis
Stanford University
{anakli, hlitz, kozyraki}@stanford.edu

Summary

Remote access to NVMe Flash enables flexible scaling and high utilization of Flash capacity and IOPS within a datacenter. However, existing systems for remote Flash access either introduce significant performance overheads or fail to isolate the multiple remote clients sharing each Flash device. We present ReFlex, a software-based system for remote Flash access that provides nearly identical performance to accessing local Flash. ReFlex uses a dataplane kernel to closely integrate networking and storage processing to achieve low latency and high throughput at low resource requirements. Specifically, ReFlex can serve up to 850K IOPS per core over TCP/IP networking, while adding 21μs over direct access to local Flash. ReFlex uses a QoS scheduler that can enforce tail latency and throughput service-level objectives (SLOs) for thousands of remote clients. We show that ReFlex allows applications to use remote Flash while maintaining their original performance with local Flash.

ReFlex was published in the proceedings of *ASPLOS* 2017 [5]. ReFlex is open-source software. The code is available at `https://github.com/stanford-mast/reflex`.

Motivation

NVMe Flash devices deliver up to 1 million I/O operations per second (IOPS) at sub 100μs latencies, making them the preferred storage medium for many data-intensive, online services. However, Flash devices deployed in datacenters are often underutilized in terms of capacity and throughput due to the imbalanced requirements across applications and over time [4, 6]. In general, it is difficult to design machines with the perfect balance between CPU, memory, and Flash resources for all workloads, which leads to over-provisioning and higher total cost of ownership (TCO). Similarly to sharing disks within a datacenter, remote access to Flash over the network can greatly improve utilization by allowing access to Flash on either any machine that has spare capacity and bandwidth or on servers dedicated to serving a large number of NVMe devices.

There are significant challenges in implementing remote access to Flash. Achieving *low latency* requires minimal processing overheads at the network and storage layers in both the server and client machines. In addition to low latency, each server must achieve *high throughput at minimum cost*, saturating one or more NVMe Flash devices with a small number of CPU cores. Moreover, managing interference between multiple tenants sharing a Flash device and the uneven read/write behavior of Flash devices requires *isolation mechanisms* that can guarantee predictable performance for all tenants. Read/write interference can be managed when a single application uses a local device, but becomes a big challenge with multiple tenants sharing the same remote Flash device, unaware of each other. Finally, it is useful to have *flexibility* in the degree of sharing, the deployment scale, and the network protocol used for remote connections. Existing, software-only options for remote Flash access, like iSCSI [7] or event-based servers, cannot meet performance expectations. Recently proposed, hardware-accelerated options, like NVMe over RDMA fabrics [2], lack performance isolation.

ReFlex Design

ReFlex is a software-based Flash storage server that implements remote Flash access over commodity Ethernet networks at comparable performance to local Flash access. The system serves remote read/write requests for logical blocks of any size over general networking protocols like TCP and UDP. ReFlex achieves high performance with limited compute requirements using a novel dataplane kernel that tightly integrates networking and storage. ReFlex also uses a novel I/O scheduler to enforce latency and throughput SLOs for multiple tenants sharing a storage device.

Dataplane execution model: ReFlex leverages hardware virtualization capabilities in NICs and NVMe Flash devices to operate directly on hardware queues to efficiently forward data between NICs and Flash devices without copying. The polling-based execution model allows requests to be processed without interruptions, improving cache locality and reducing unpredictability. ReFlex employs adaptive batching of requests to amortize overheads while improving prefetching and instruction cache efficiency [1]. Under low load, incoming requests are processed immediately without any delay while at high load, multiple incoming packets or completed NVMe Flash requests are processed in a batch. ReFlex scales to multiple threads, each using a dedicated core and separate hardware queue pairs. The system can support thousands of tenants and connections.

* The first two authors contributed equally to this work.

QoS scheduling and isolation: ReFlex uses a QoS-aware scheduler to provide performance guarantees for multiple tenants sharing remote Flash. Each tenant registers with ReFlex as either a latency-critical tenant (which has a tail latency and throughput SLO) or a best-effort tenant (which opportunistically uses any unallocated or unused Flash bandwidth and tolerates higher latency). ReFlex employs a request cost model to account for the impact of each individual I/O on the tail latency of concurrent I/O requests. The model assigns to each I/O request a cost, expressed in terms of tokens, depending on the type (read vs. write) and size of the I/O. Incoming I/O requests are placed in per-tenant software queues. In each scheduling round, which occurs once per polling loop, the ReFlex scheduler generates and assigns tokens for each tenant based on the tenant's SLO. In each round, the scheduler also spends tenants' tokens by submitting to Flash the requests for which tenants have accumulated sufficient tokens.

Result Highlights

The unloaded round-trip latency of the ReFlex server is only 21μs higher than direct, userspace access to local Flash through NVMe queues. Figure 1 plots tail latency as a function of throughput (IOPS) for a 1KB read workload. For local Flash access with SPDK [3], a single core can support up to 870K IOPS and it takes two cores to saturate the 1M IOPS device. ReFlex achieves up to 850K IOPS with a single core for network and storage processing. With two cores, ReFlex saturates 1M IOPS on Flash, introducing negligible latency overhead compared to local access. In contrast, the Linux baseline, which uses `libaio` and `libevent`, achieves only 75K IOPS/core and at higher latency due to higher compute intensity, requiring over 10× more CPU cores to achieve the throughput of ReFlex.

The ReFlex server can support thousands of remote tenants. Its QoS scheduler can enforce the tail latency and throughput requirements of tenants with SLOs, while allowing best-effort tenants to consume all remaining throughput of the NVMe device. Finally, using legacy applications, we show that even with heavy-weight clients, ReFlex allows for performance levels nearly identical to those with local Flash.

Impact

ReFlex has immediate and long-term impact on the design and deployment of high performance, cost effective datacenter storage systems.

Flash storage disaggregation: ReFlex's ability to serve millions of IOPS on commodity Ethernet networks with a small number of cores and without impacting tail latency is important for making remote Flash practical and cost effective in datacenters. Disaggregating Flash allows customizing the ratio of resources allocated to an application based on its unique and evolving requirements. The flexibility to scale storage independently from compute resources improves utilization in the datacenter [4]. ReFlex enables this flexibility while providing the same performance applications expect from local Flash.

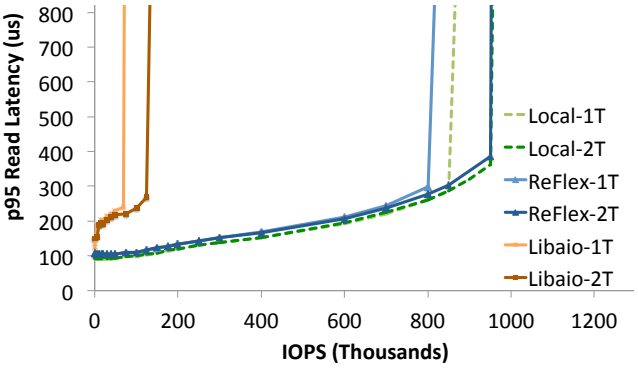

Figure 1: Tail latency vs. throughput for 1KB read workload

Isolation and QoS: Whether Flash is distributed across the datacenter or pooled in specialized storage servers, remote access enables sharing Flash among multiple tenants for greater resource efficiency. Our work on ReFlex demonstrates the importance of managing interference on shared Flash using isolation mechanisms to provide *predictable* performance. ReFlex uses a request cost model to account for the uneven behavior of read/write operations and a novel QoS scheduler to submit requests to a Flash device based on tenant SLOs and device capabilities.

References

[1] Adam Belay, George Prekas, Ana Klimovic, Samuel Grossman, Christos Kozyrakis, and Edouard Bugnion. IX: A protected dataplane operating system for high throughput and low latency. In *Proc. of USENIX Operating Systems Design and Implementation*, OSDI'14, pages 49–65, October 2014.

[2] Chelsio Communications. NVM Express over Fabrics. `http://www.chelsio.com/wp-content/uploads/resources/NVM_Express_Over_Fabrics.pdf`, 2014.

[3] Intel Corporation. Storage Performance Development Kit. `https://01.org/spdk`, 2016.

[4] Ana Klimovic, Christos Kozyrakis, Eno Thereska, Binu John, and Sanjeev Kumar. Flash storage disaggregation. In *Proc. of European Conference on Computer Systems*, EuroSys '16, pages 29:1–29:15, 2016.

[5] Ana Klimovic, Heiner Litz, and Christos Kozyrakis. ReFlex: Remote Flash == Local Flash. In *Proc. of Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, pages 345–359, 2017.

[6] Jian Ouyang, Shiding Lin, Jiang Song, Zhenyu Hou, Yong Wang, and Yuanzheng Wang. SDF: software-defined flash for web-scale internet storage systems. In *Proc. of Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, pages 471–484, 2014.

[7] Satran, J., Meth K., Sapuntzakis, C., Chadalapaka, M., Zeidner, E. Internet Small Computer Systems Interface (iSCSI). `https://www.ietf.org/rfc/rfc3720.txt`, 2004.