ReFlex: Remote Flash ≈ Local Flash

Ana Klimovic  Heiner Litz
Christos Kozyrakis

NVMW’18
Memorable Paper Award Finalist
Flash in Datacenters

• Flash provides 1000× higher throughput and 100× lower latency than disk

• Flash is often underutilized due to imbalanced resource requirements

PCle Flash:
– 1,000,000 IOPS
– 70 μs read latency

Solution: share SSD between remote tenants
Existing Approaches

• Remote access to disk (e.g. iSCSI)
• Remote access to DRAM or NVMe over RDMA

• There are 2 main issues:
  1. Performance overhead
  2. Interference on shared remote flash device
Issue 1: Performance Overhead

- Traditional network storage protocols and Linux I/O libraries (e.g. libaio, libevent) have high overhead
Issue 2: Performance Interference

To share Flash, we need to enforce performance isolation.
How does ReFlex achieve high performance?

**Linux** vs. **ReFlex**

- **User Space**
  - Remote Storage Application

- **Software**
  - Filesystem
  - Block I/O
  - Device Driver

- **Hardware**
  - Network Interface
  - Flash Storage

- **Control Plane**
  - Data Plane

- **User Space**
  - Remote Storage Application

- **Data Plane**
  - Network Interface
  - Flash Storage
How does ReFlex achieve high performance?

Linux vs. ReFlex

- **User Space**
  - Remote Storage Application

- **Linux**
  - Filesystem
  - Block I/O
  - Device Driver

- **Hardware**
  - Network Interface
  - Flash Storage

- **ReFlex**
  - Remote Storage Application
  - Control Plane
  - Data Plane

- **Hardware**
  - Network Interface
  - Flash Storage

- **Remove SW bloat by separating control & data plane**
How does ReFlex achieve high performance?

Linux vs. ReFlex

- **Linux**
  - User Space
  - Remote Storage Application
  - Linux
    - Filesystem
    - Block I/O
    - Device Driver
  - Hardware
    - Network Interface
    - Flash Storage

- **ReFlex**
  - User Space
  - Remote Storage Application
  - ReFlex
    - Control Plane
    - Data Plane
      - DPDK
      - SPDK
  - Hardware
    - Network Interface
    - Flash Storage

Direct access to hardware

1 data plane per CPU core
How does ReFlex achieve high performance?

Linux vs. ReFlex

User Space

Remote Storage Application

Linux

Filesystem

Block I/O

Device Driver

Hardware

IRQ

Network Interface

Flash Storage

Polling vs. interrupts

Data Plane

Hardware

Network Interface

Flash Storage
How does ReFlex achieve high performance?

Linux vs. ReFlex

Polling vs. interrupts

Run to completion
How does ReFlex achieve high performance?

Linux vs. ReFlex

- Polling vs. interrupts
- Adaptive batching
How does ReFlex achieve high performance?

1. How does ReFlex achieve high performance?

2. Zero-copy device-to-device

3. Data Plane

4. External Storage

Linux vs. ReFlex

User Space

Remote Storage Application

Filesystem

Block I/O

Device Driver

Hardware

Network Interface

Flash Storage

User Space

Remote Storage Application

Hardware

Network Interface

Flash Storage

Remote Storage Application

Hardware

Network Interface

Flash Storage

User Space

Remote Storage Application

Hardware

Network Interface

Flash Storage

Remote Storage Application

Hardware

Network Interface

Flash Storage
How does ReFlex enable performance isolation?

• Request cost based scheduling

• Determine the impact of tenant A on the tail latency and IOPS of tenant B

• Control plane assigns tenants with a quota

• Data plane enforces quotas through throttling
Request Cost Modeling

Compensate for read-write asymmetry

For this device:
Write == 10x Read
Request Cost Based Scheduling
Request Cost Based Scheduling

1ms tail latency SLO
Request Cost Based Scheduling

![Graph showing p95 read latency (us) vs. Weighted IOPS]

1ms tail latency SLO

Device max IOPS: 510K

Weighted IOPS

p95 read latency (us)
Request Cost Based Scheduling

- **1ms tail latency SLO**
- **Device max IOPS:** 510K
- **200K IOPS SLO**
Request Cost Based Scheduling

- **Device max IOPS:** 510K
- **200K IOPS SLO:** 250K
- **310K Slack SLO:** 500K
- **1ms tail latency SLO**
Results: Local ≈ Remote Latency

- Linux: 75K IOPS/core
- ReFlex: 850K IOPS/core
Results: Local ≈ Remote Latency

Latency:
- Local Flash: 78 µs
- ReFlex: 99 µs
- Linux: 200 µs
Results: Local $\approx$ Remote Latency

ReFlex: saturates Flash
Results: Performance Isolation

- **Tenants A & B:** latency-critical; **Tenant C + D:** best effort
Results: Performance Isolation

- Tenants A & B: latency-critical; Tenant C + D: best effort
- Without scheduler: latency and bandwidth QoS for A/B are violated
Results: Performance Isolation

- Tenants A & B: latency-critical; Tenant C + D: best effort
- Without scheduler: latency and bandwidth QoS for A/B are violated
- Scheduler rate limits best-effort tenants to enforce SLOs
ReFlex Summary

1. Enables Flash disaggregation → improve utilization
   – Performance: remote ≈ local
   – Commodity networking, low CPU overhead

2. Guarantees QoS in shared resource deployments
   – Quality of Service aware request scheduling
Impact of ReFlex

- Open source: [https://github.com/stanford-mast/reflex](https://github.com/stanford-mast/reflex)
  - Works on AWS i3 cloud instances with NVMe Flash
  - Integrated as a remote Flash dataplane in the Apache Crail distributed storage system (collaboration with IBM Research)
  - Broadcom is porting ReFlex to ARM-based SoC
Thank You!

Download the source code at: https://github.com/stanford-mast/reflex

Original paper presented at ASPLOS’17.