April 26-28, 2022
DoubleTree by Hilton San Jose
SmartNICsSummit.com

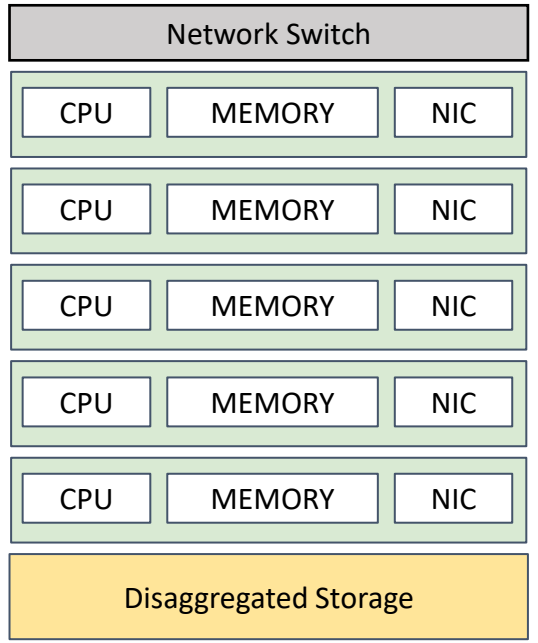
Gimbal: Enabling Multi-tenant Storage Disaggregation on SmartNIC JBOFs

Jaehong Min, Ming Liu, Tapan Chugh, Chenxingyu Zhao

Andrew Wei, In Hwan Doh, and Arvind Krishnamurthy

Storage Disaggregation

- Growth of network B/W enables disaggregated infrastructures
  - Storage and accelerator disaggregation are the most common today
- Benefits of the storage disaggregation
  - Independent resource scaling
  - High resource utilization
  - Negligible performance degradation over high-speed networks
- Accelerated by HW and SW innovations
  - SmartNIC, DPU, etc.
  - SPDK and OS supports



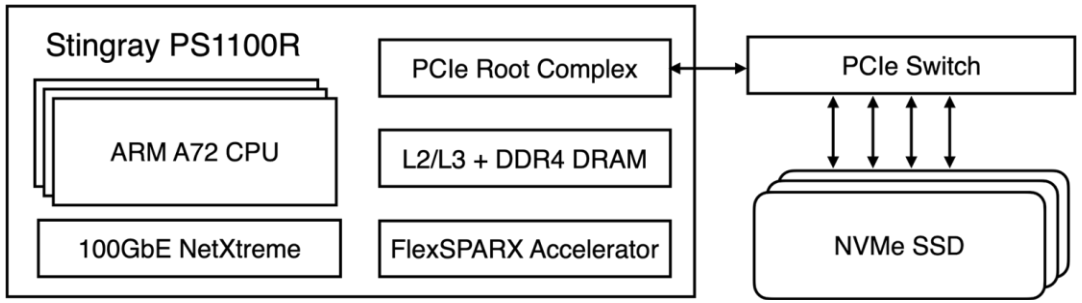
SmartNIC JBOF

Just-Bunch-Of-Flash

- SmartNIC JBOF implements cost-effective, power-efficient system

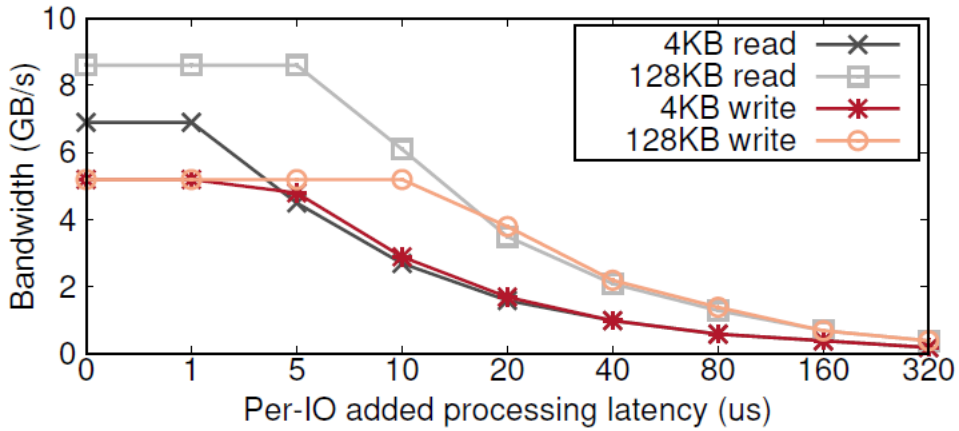
- Our testbed: Broadcom Stingray
  - Power-efficient design using SoC
  - Eight ARM A72 cores
  - Test platform supports up to four NVMe
  - Support NVMe-over-Fabrics (RDMA) using SPDK

Broadcom Stingray SmartNIC



SmartNIC JBOF Characterization

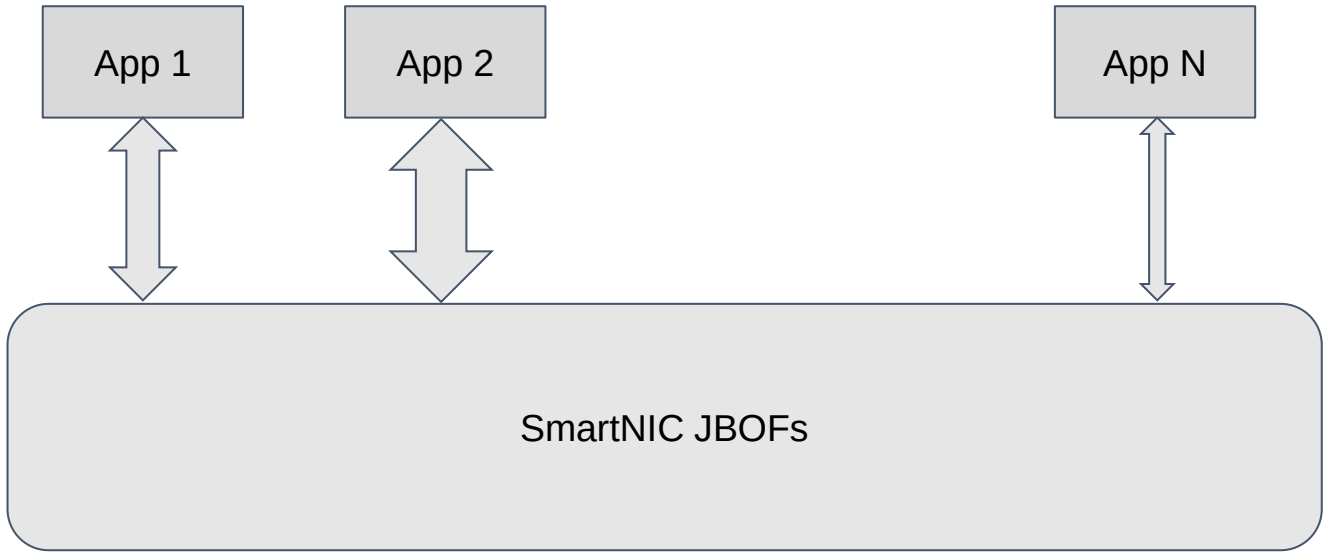
- Similar IO performance to server-based* JBOF
  - Less than 5% degradation for small IO, the same for large IO
- Significantly less power consumption than server-based JBOF
  - About half in the idle state (45.5W vs. 92W)
  - Only 27% under the maximum load (52.5W vs. 192W)
- Challenge
  - Small headroom for additional computation
  - 1us for a small IO, 5us for a large IO



* Xeon server with two Intel Xeon E5-2620 v4 CPU, All cores are active in the maximum load

Multi-tenant Disaggregated Storage

- Disaggregation inevitably creates a multi-tenancy environment
  - Current disaggregated storage lacks support for fair sharing of storage

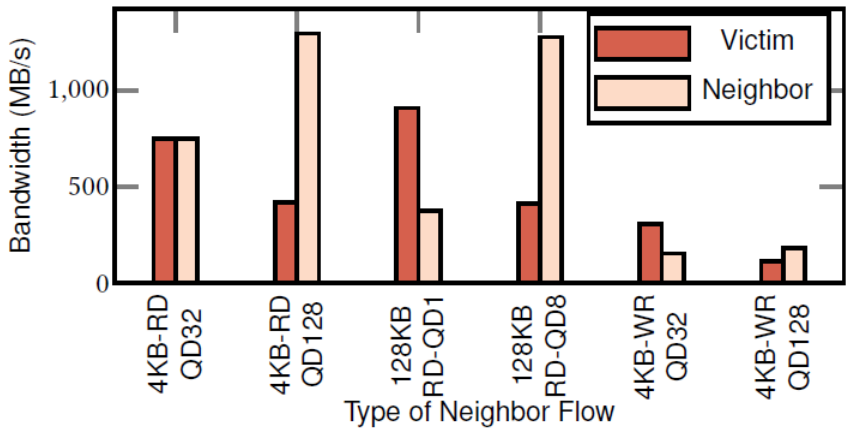


Challenges with SSD multi-tenancy

- IO Fairness
- Read and Write asymmetry and IO interference
- SSD condition and Non-deterministic device performance

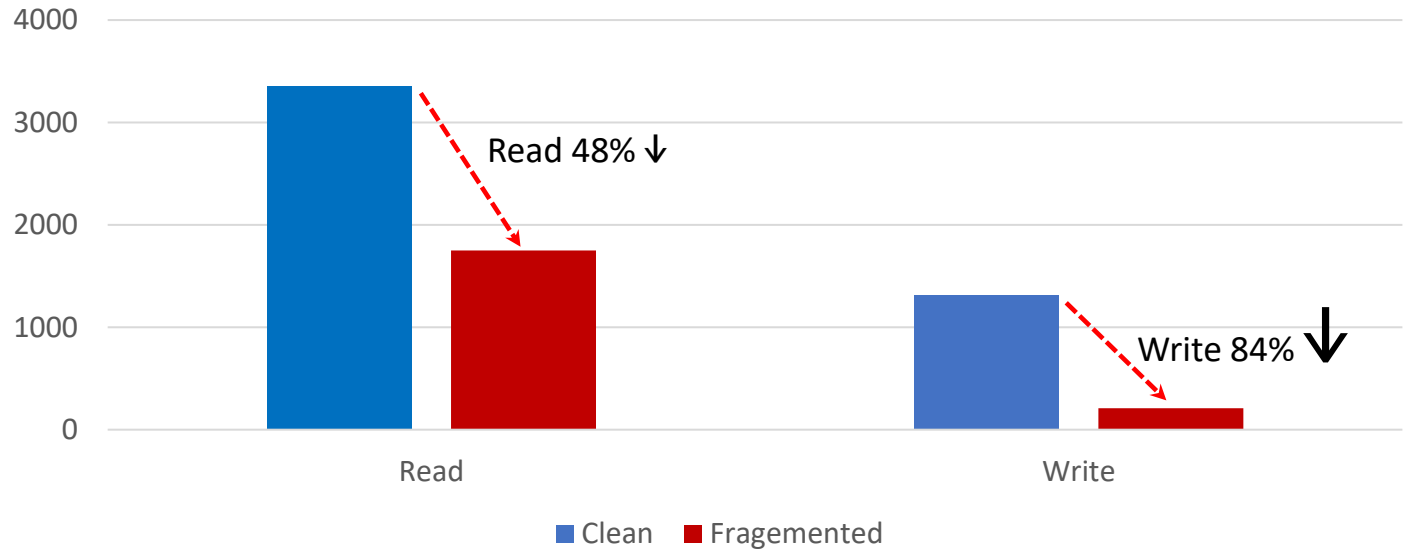
Read and Write Asymmetry

- NAND operations and Garbage Collection causes the asymmetry
  - NAND: Read (50~150us) vs. Write (2~2.5ms)
  - Garbage collection increases the cost of write IO
- Mixed IO significantly degrades the device performance
  - E.g., 4KB-Read and 4KB-Write mixed IO performs only 30% of 4KB-Read only
  - High tail latency of write causes read latency to deteriorate



SSD Conditions and Non-deterministic Performance

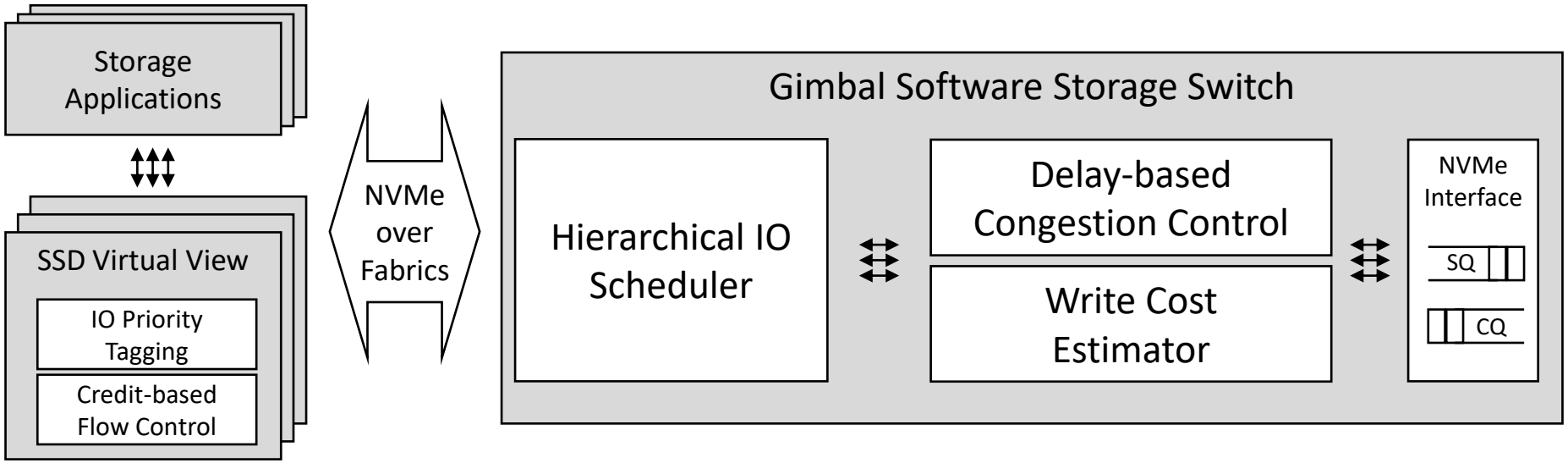
- SSD condition moves the performance curve
  - Internal fragmentation will trigger more GCs and cause fragmented reads
  - Not only significant, but also asymmetric



SSD conditions has significant impact on the performance capability

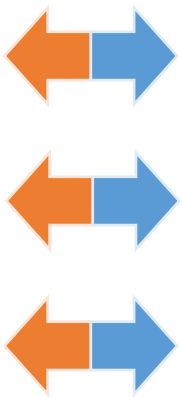
Gimbal: Architecture

- Pipelined architecture inspired by today's SAN switches
- Each pipeline is dedicated to a specific SSD



Key Components

- **Challenges**
  - ➢ Non-deterministic device performance
  - ➢ IO Fairness
  - ➢ Read and Write Asymmetry



- **Proposed Solutions**
  - ✓ Delay-based Congestion Control
  - ✓ Hierarchical IO Scheduler and Virtual Slot
  - ✓ Write Cost Estimator

Delay-based Congestion Control

Problems

- SSD internal parallelism is unknown to the host
- Housekeeping operations are unpredictable

Observations

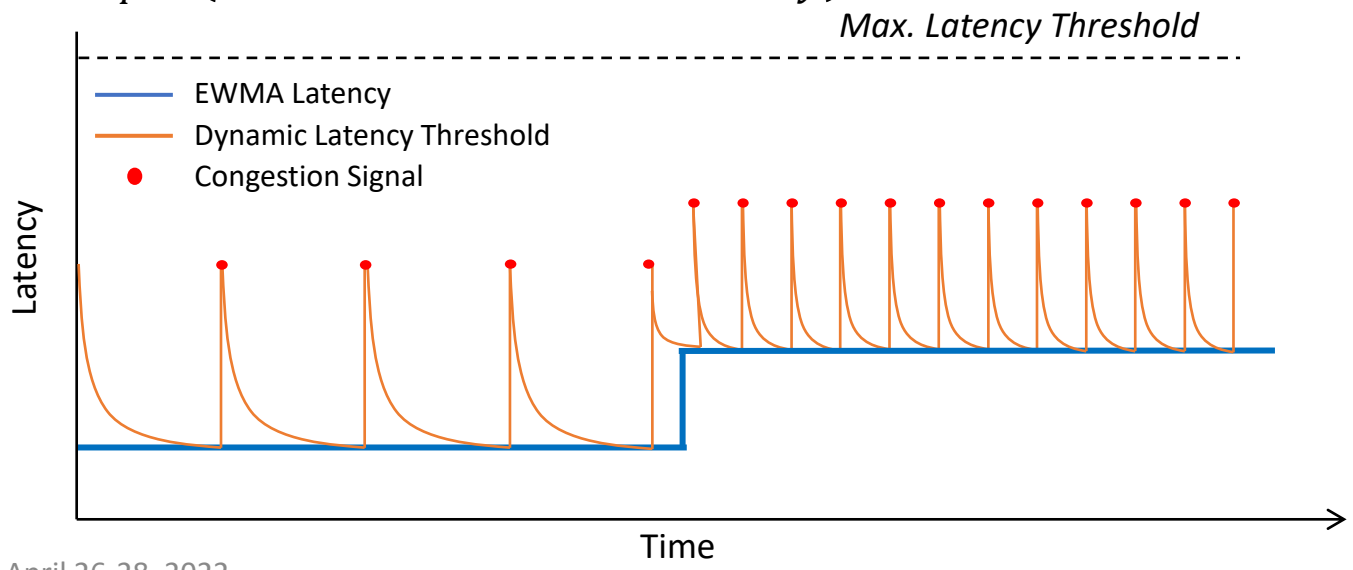
- Latency has an impulse response to the congestion
- Latency in congestion varies according to the IO size

Solution

- Delay-based congestion control
- SSD-specific optimization: Dynamic latency threshold

Dynamic Latency Threshold for Detecting Congestion

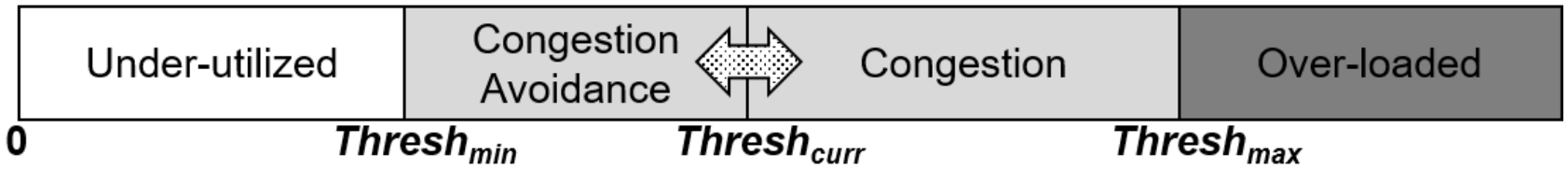
- Latency depends on IO sizes, and a static threshold is not effective
- Update the threshold with the latency on each IO completion
  - Latency exceeds the threshold: Signal congestion and increase the threshold
    → $(Threshold + Maximum\ Threshold) \div 2$
  - Latency under the threshold: Decay the threshold
    → $Threshold - \alpha_T \times (Threshold - EWMA\ Latency)$



The congestion level is the signal generation rate

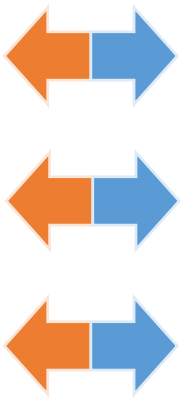
Congestion States and Rate Pacing

- Token bucket algorithm for the rate pacing
  - Avoid burst IO submission with separated buckets for read and write
- Four congestion states for adjusting the target rate
  - Congestion Avoidance : Increase the rate by the IO size
  - Congestion : Decrease the rate by the IO size
  - Under-utilized : Aggressive B/W probing (increase by $\beta \times IO\ Size$)
  - Over-loaded : Adjust to below the IO completion rate



Key Components

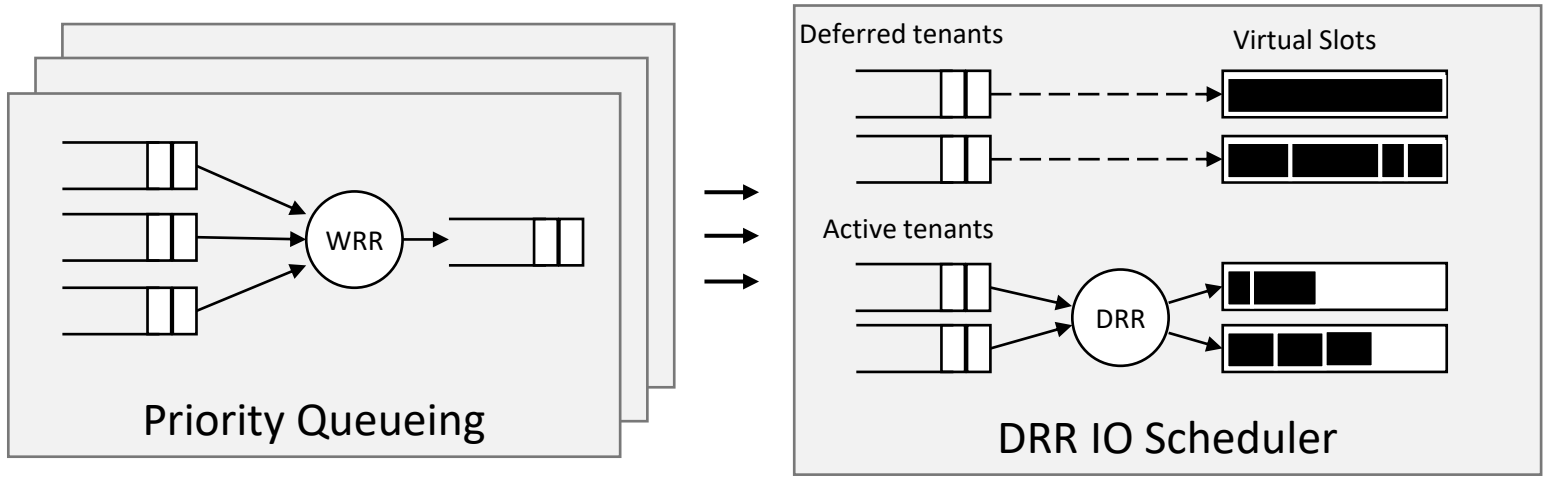
- **Challenges**
  - ➢ Non-deterministic device performance
  - ➢ IO Fairness
  - ➢ Read and Write Asymmetry



- **Proposed Solutions**
  - ✓ Delay-based Congestion Control
  - ✓ Hierarchical IO Scheduler and Virtual Slot
  - ✓ Write Cost Estimator

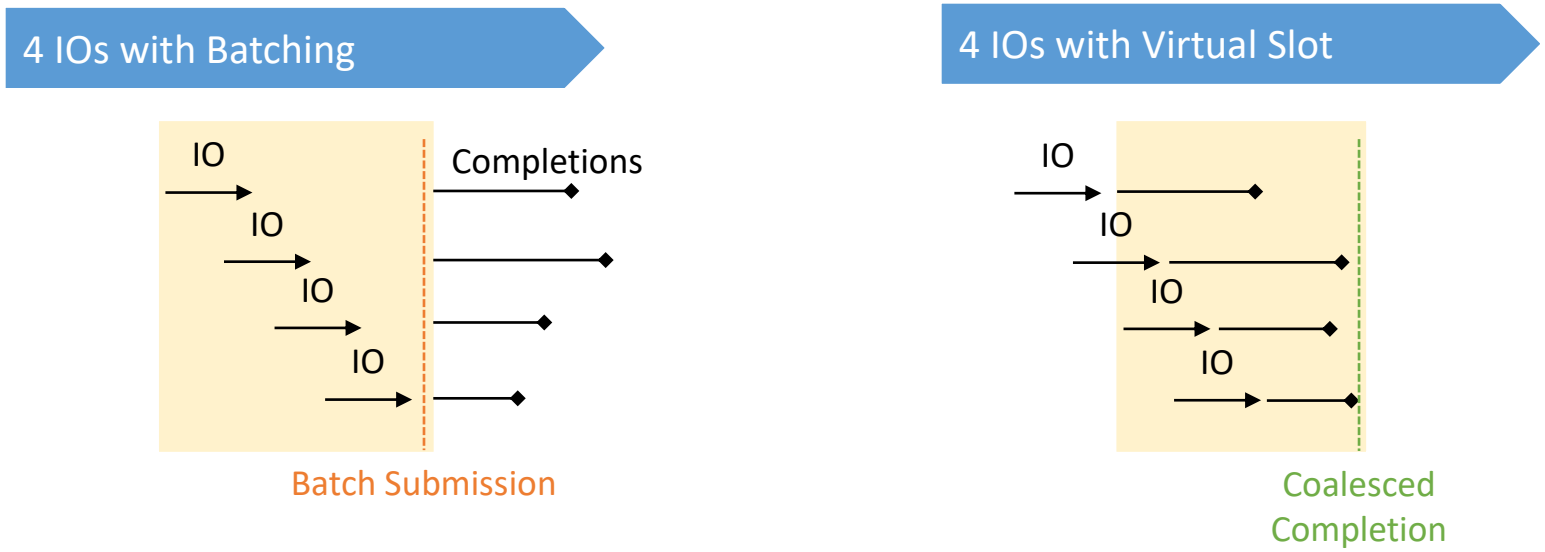
IO scheduler and Virtual Slot

- Two-level Hierarchical IO Scheduler
  - Per-tenant priority queue
  - DRR IO scheduler
- Virtual Slot
  - Normalized scheduling unit with a coalesced IO completion
  - Defer the queue when the tenant has no available virtual slots



Virtual Slot

- Manage the discrepancy of the completion for small and large IO
  - SSD generates the completion only when all chunks of the IO has processed
- Virtual slot coalesces IO completions
  - Batching IO submission may hurt the IO latency
  - The virtual slot completes only when all requests have processed



Evaluation Setup

- System Setup
  - Broadcom Stingray SmartNIC with Samsung DCT983 960GB NVMe SSD
  - Two SSD conditions: Clean and Fragmented
  - 100GbE RDMA Network
- Comparing schemes
  - Calibrated using the *Fragmented* condition

	ReFlex	Parda	FlashFQ	Gimbal
BW Estimation	Static	Dynamic	✗	Dynamic
Write Cost	Static	✗	Static	Dynamic
Fair Queueing	@Target	@Client	@Target	@Target
Flow Control	✗	✓	✗	✓

Microbenchmarks

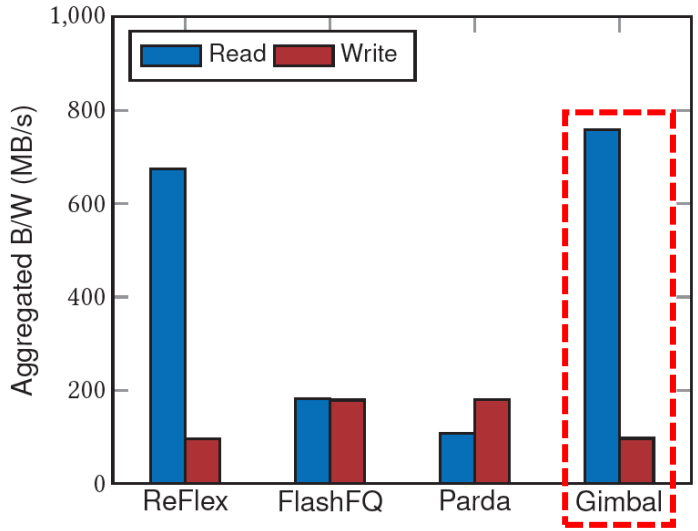
- Evaluations
  - Fairness in mixed workloads
  - Gimbal Overhead
  - Maximum Utilization
  - Flow Control and Latency
  - Performance over time with the dynamic workload
  - Generalization

Fairness in mixed workloads

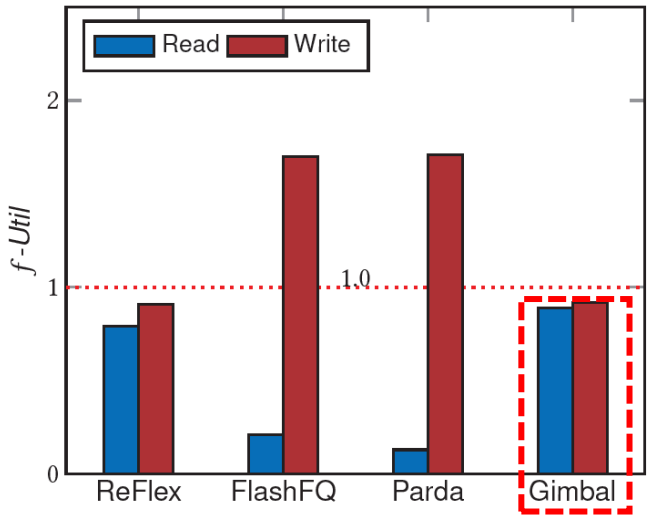
- Definition of Fair Utilization
  - Ratio to fair bandwidth share, 1.0 is ideal value

$$f - Util = \frac{Worker\ BW}{Standalone\ Max\ BW \div Number\ of\ workers}$$

- Mixed read and write tenants on the *Fragmented* SSD
  - Read/Write bandwidth capability ratio = 9 :1
  - 16 workers for each read and write
  - Gimbal performs the best for both utilization and fairness



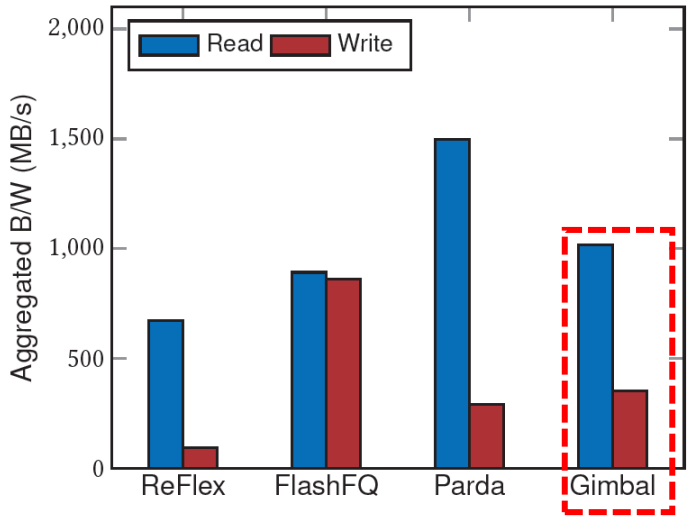
(a) Fragment-SSD: 4KB Read/Write (Bandwidth)



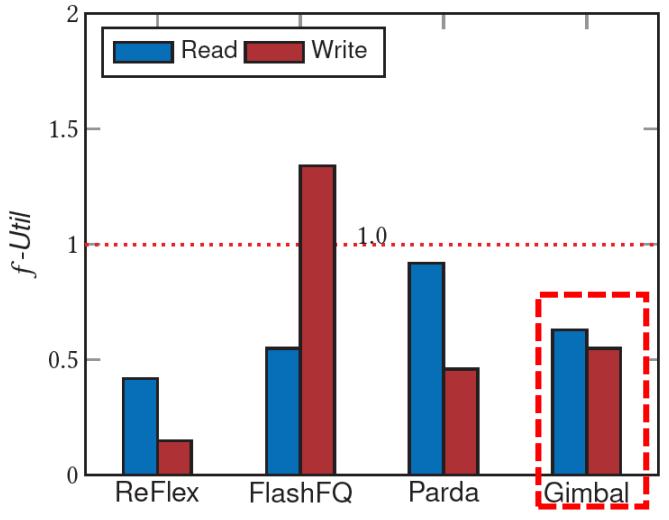
(b) Fragment-SSD: 4KB Read/Write (*f-Util*)

Fairness in mixed workloads (Cont.)

- Mixed read and write tenants on the *Clean* SSD
  - Read/Write bandwidth capability ratio = 2.7 :1
  - No re-calibration for all schemes
  - Gimbal shows the best fairness keeping the utilization high



(a) Clean-SSD: 128KB Read/Write (Bandwidth)



(b) Clean-SSD: 128KB Read/Write (*f*-*Util*)

Overhead

- Gimbal adds sub-microsecond overhead on each IO
  - Lightweight solution with a small performance penalty on SmartNIC

		Vanilla SPDK	Gimbal	
1 Tenant, 1 Outstanding IO	Submit	0.256 us	0.416 us	+62.5%
	Complete	0.128 us	0.176 us	+37.5%
16 Tenants, 32 Outstanding IOs	Submit	0.168 us	0.240 us	+42.9%
	Complete	0.126 us	0.200 us	+47.1%

- Max. performance is enough to handle an SSD with single core
  - Measured with NULL device (i.e., bypass a real device)

	Vanilla SPDK	Gimbal	
1 CPU Core, 1 Tenant	937 KIOPS	821 KIOPS	-12.4%
4 CPU Cores, 8 Tenants	2692 KIOPS	2446 KIOPS	-9.2%

Summary

- Disaggregated storage faces multi-tenancy challenges
  - IO Fairness
  - Read/Write asymmetry
  - Non-deterministic device performance
- Gimbal enables multi-tenant disaggregated storage
  - Optimizes SSD performance using techniques from networking domain
    - Delay-based Congestion Control
    - Write Cost Estimation
    - Hierarchical IO Scheduler with Virtual Slot Mechanism
  - Optimizes application performance via IO Flow Control
- Gimbal improves both fairness & QoS, keeping the utilization high

The source code is available at https://github.com/jaehongm/gimbal