Building an Elastic Block Storage over EBOFs Using Shadow Views

Sheng Jiang, Ming Liu[+]

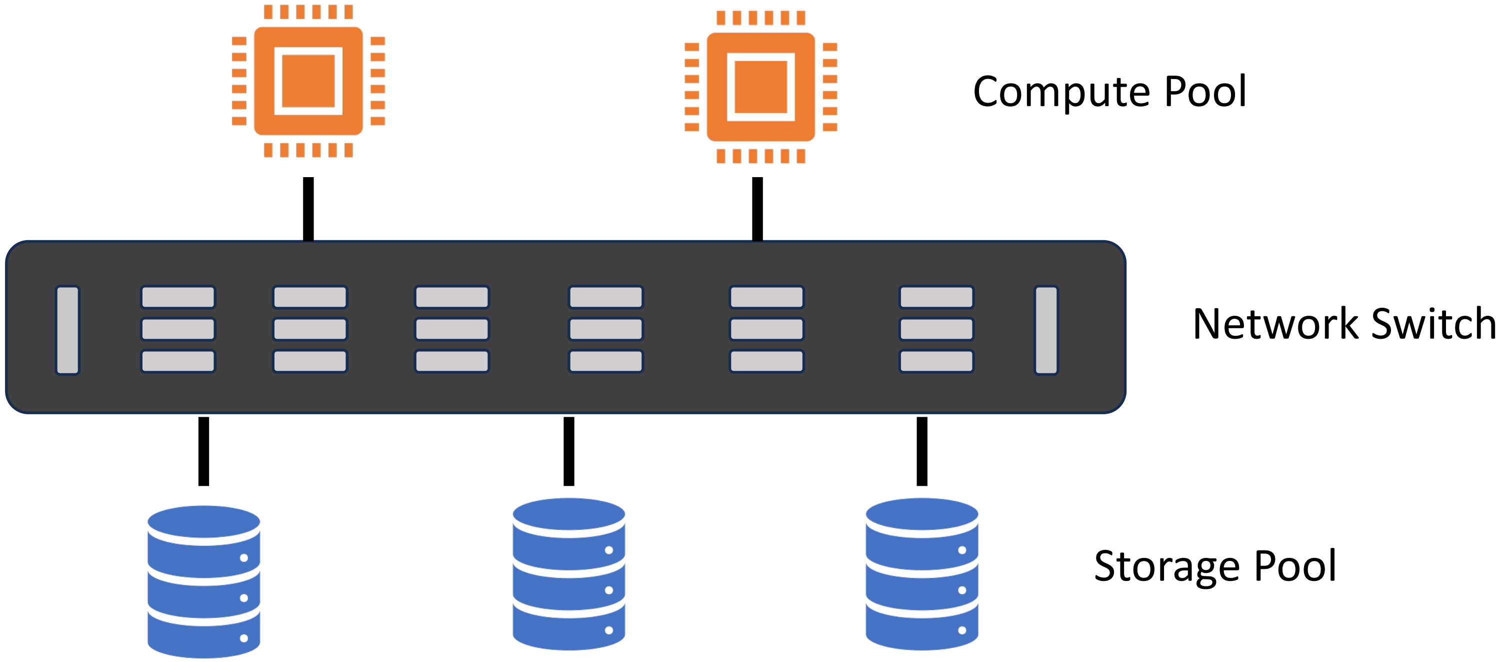
Carnegie Mellon University, University of Wisconsin – Madison[+]

Carnegie Mellon University

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Storage Disaggregation

- Storage disaggregation has been widely deployed
  - Independent scaling of compute/storage
  - Cost efficiency
  - High resource utilization

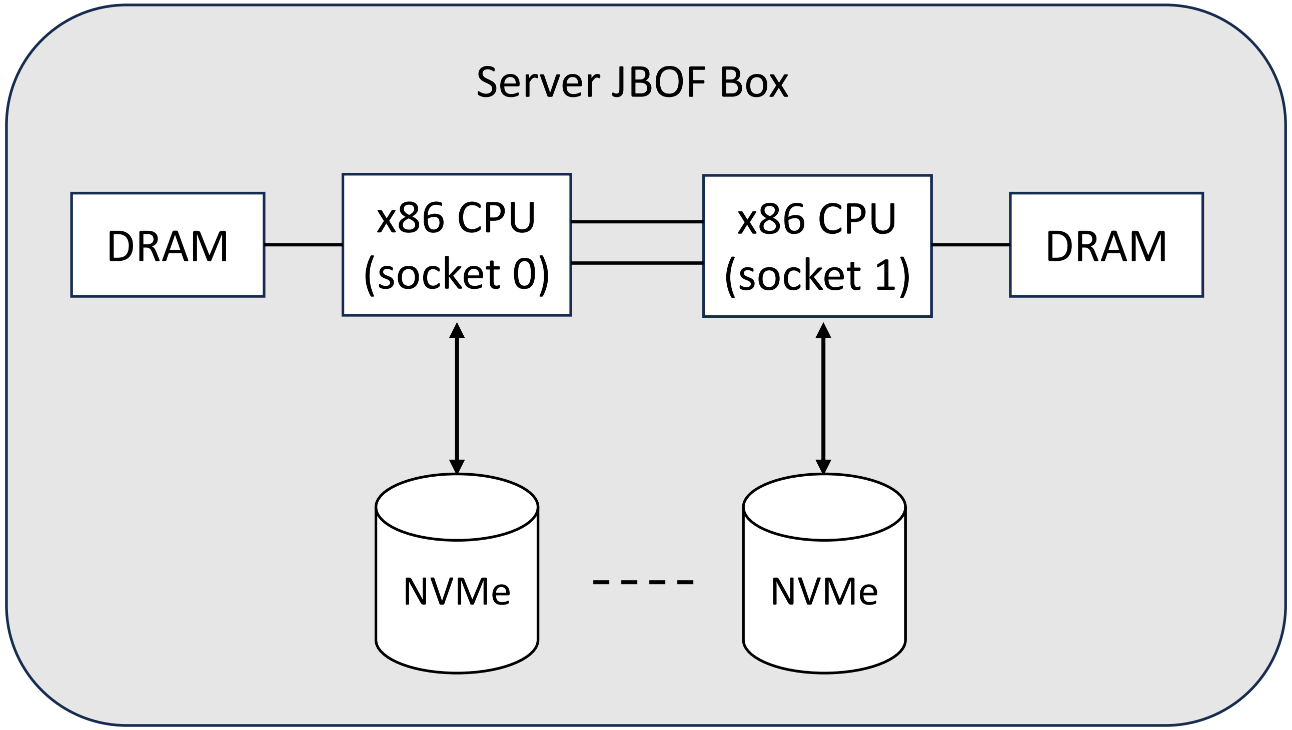


Storage Hardware Substrate – Server JBOF

- Server JBOFs enclose X86 processors and 8-24 NVMe drives

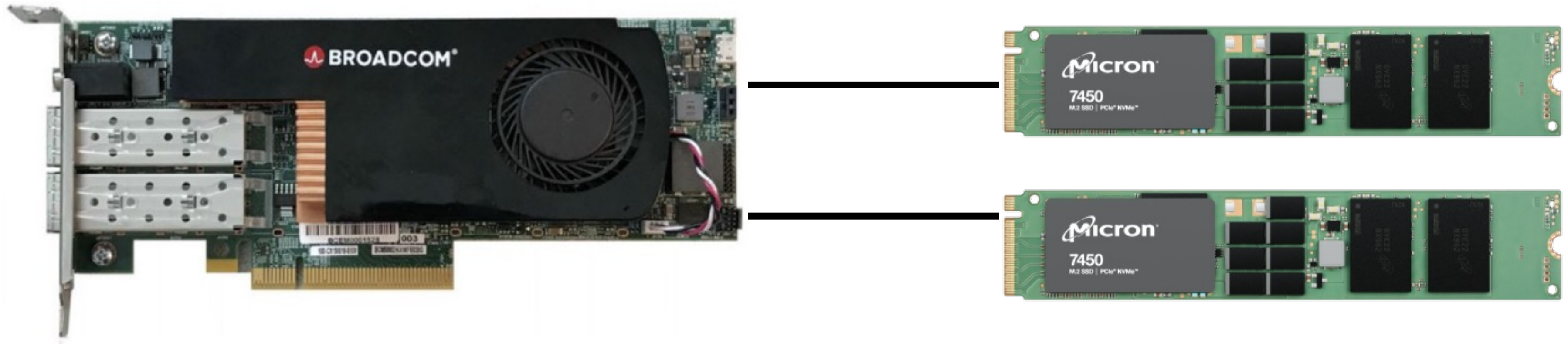
Storage Hardware Substrate – Server JBOF

- Server JBOFs enclose X86 processors and 8-24 NVMe drives



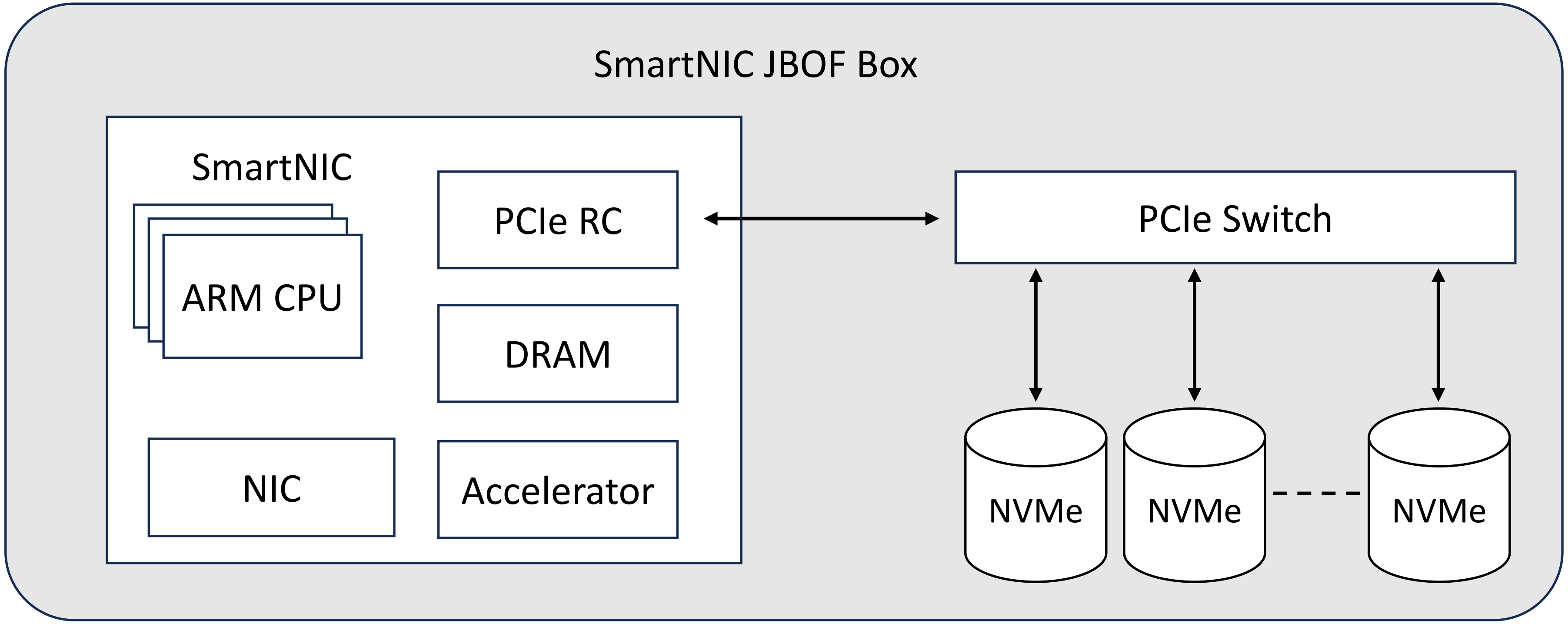
Storage Hardware Substrate – SmartNIC JBOF

- SmartNIC JBOFs: a low-power and high-performance storage appliance
  - SmartNICs and PCIe switch
  - 4-8 NVMe drives
  - Domain-specific accelerators



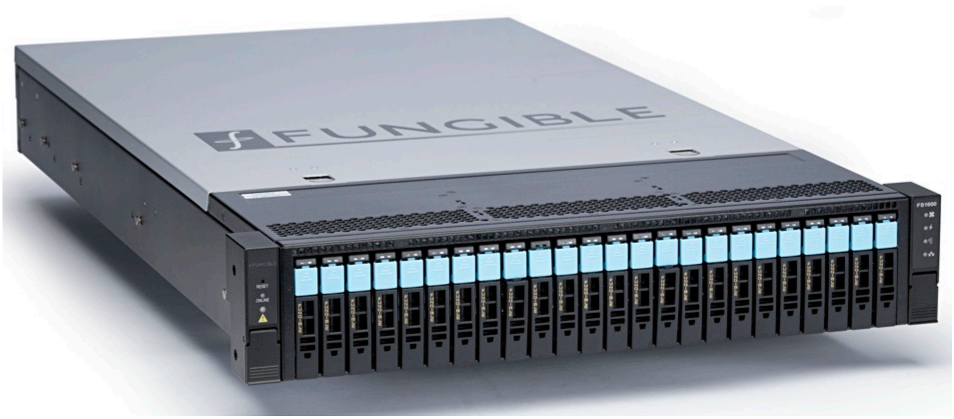
Storage Hardware Substrate – SmartNIC JBOF

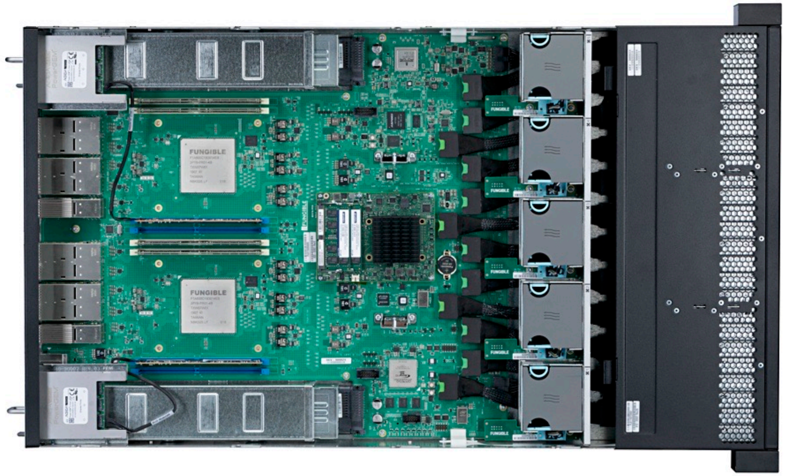
- SmartNIC JBOFs: a low-power and high-performance storage appliance
  - SmartNICs and PCIe switch
  - 4-8 NVMe drives
  - Domain-specific accelerators



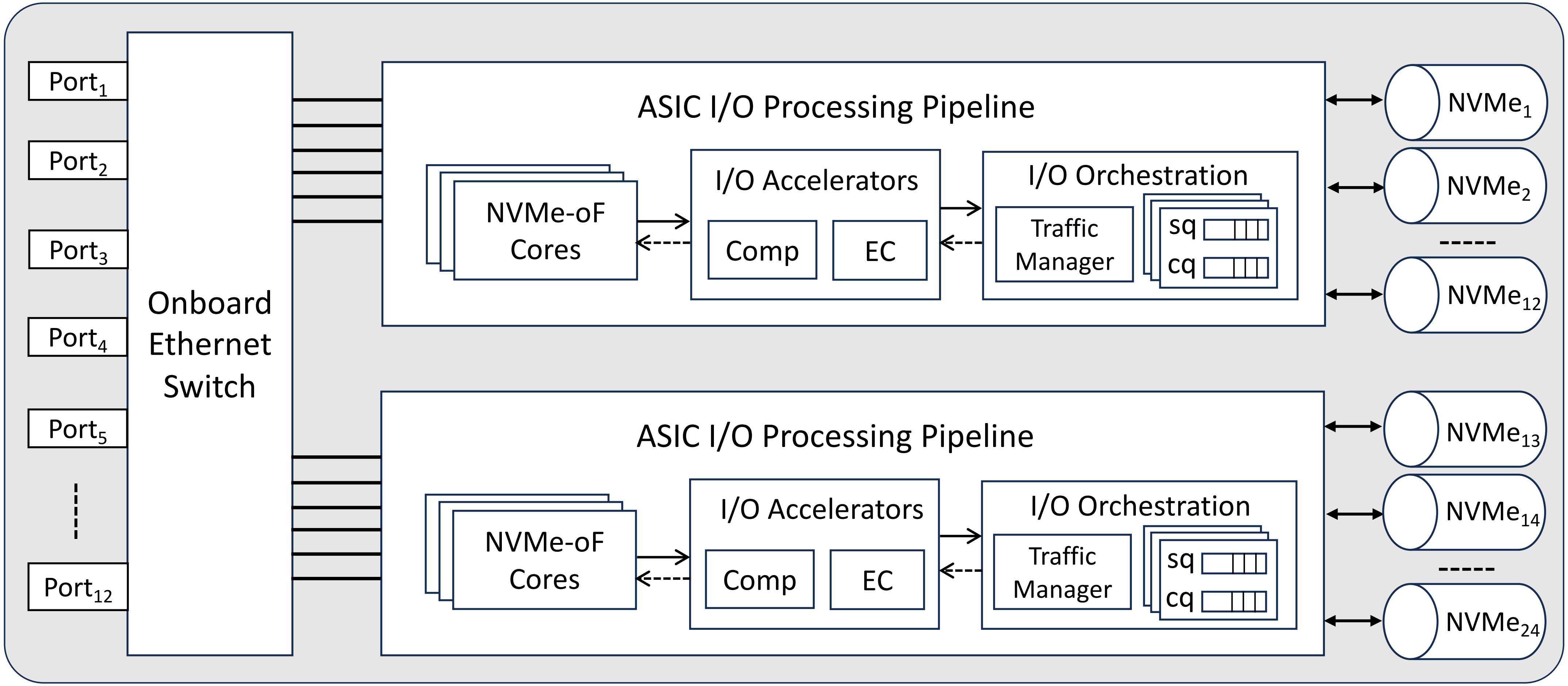
EBOF: An Emerging Storage Appliance

- Pack an Ethernet switch with NVMe drives into one SoC
  - Hardware-assisted remote I/O processing pipeline
  - Example model: Fungible/Microsoft FS1600





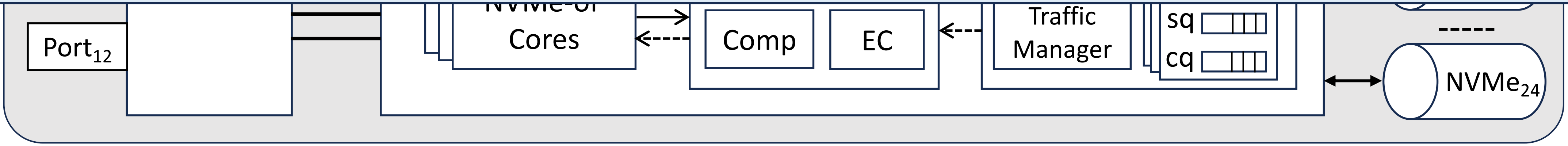
EBOF Hardware Architecture



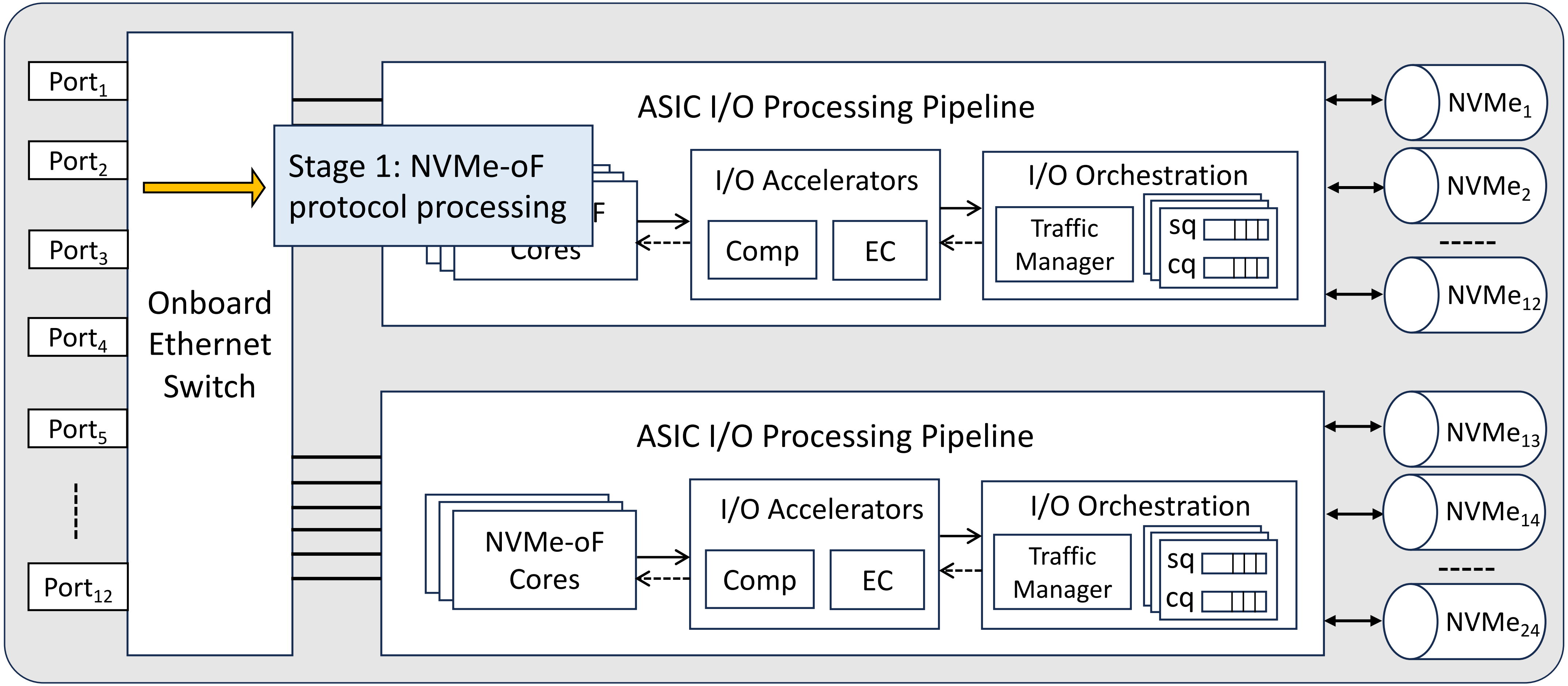
EBOF Hardware Architecture

Benefits

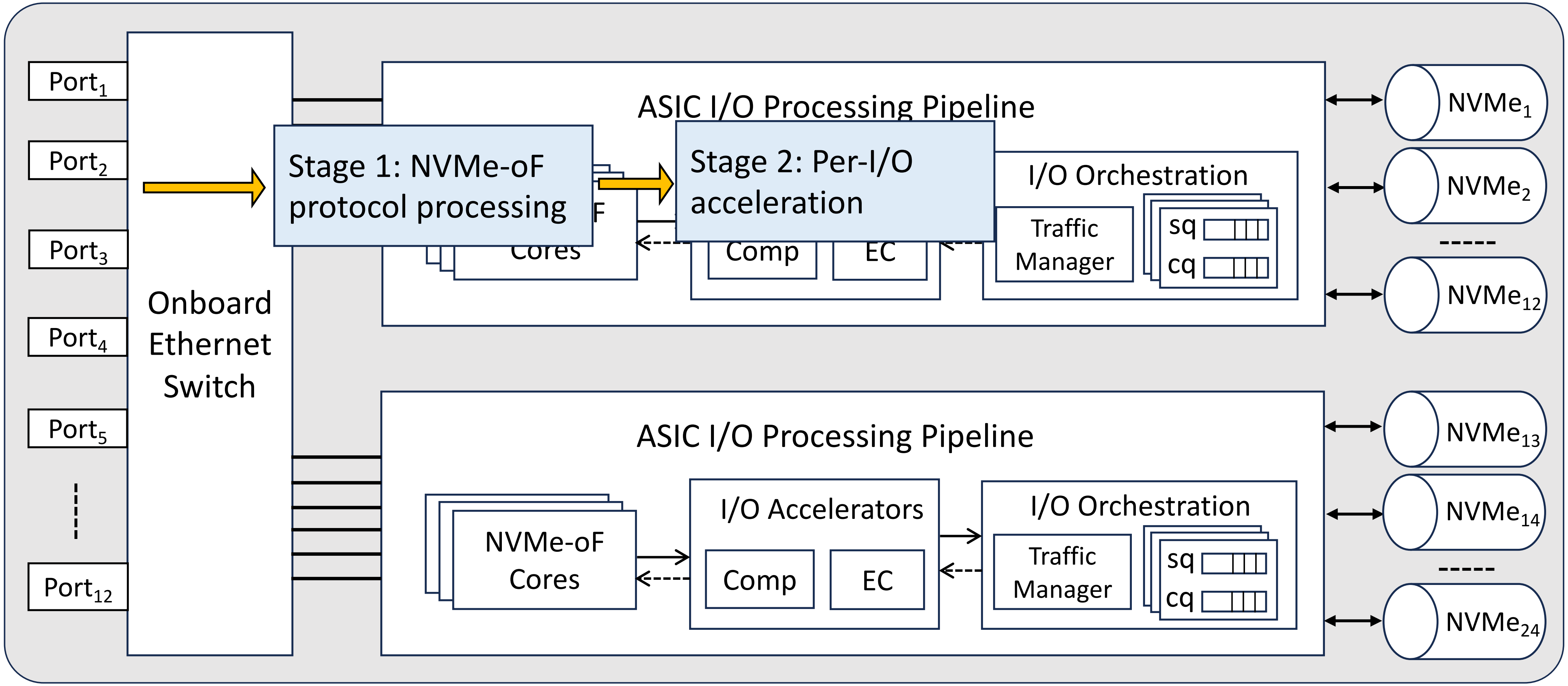
- Increase I/O scalability
- Improve overall resource utilization and reduce resource stranding
- Achieve higher energy/cost-efficiency



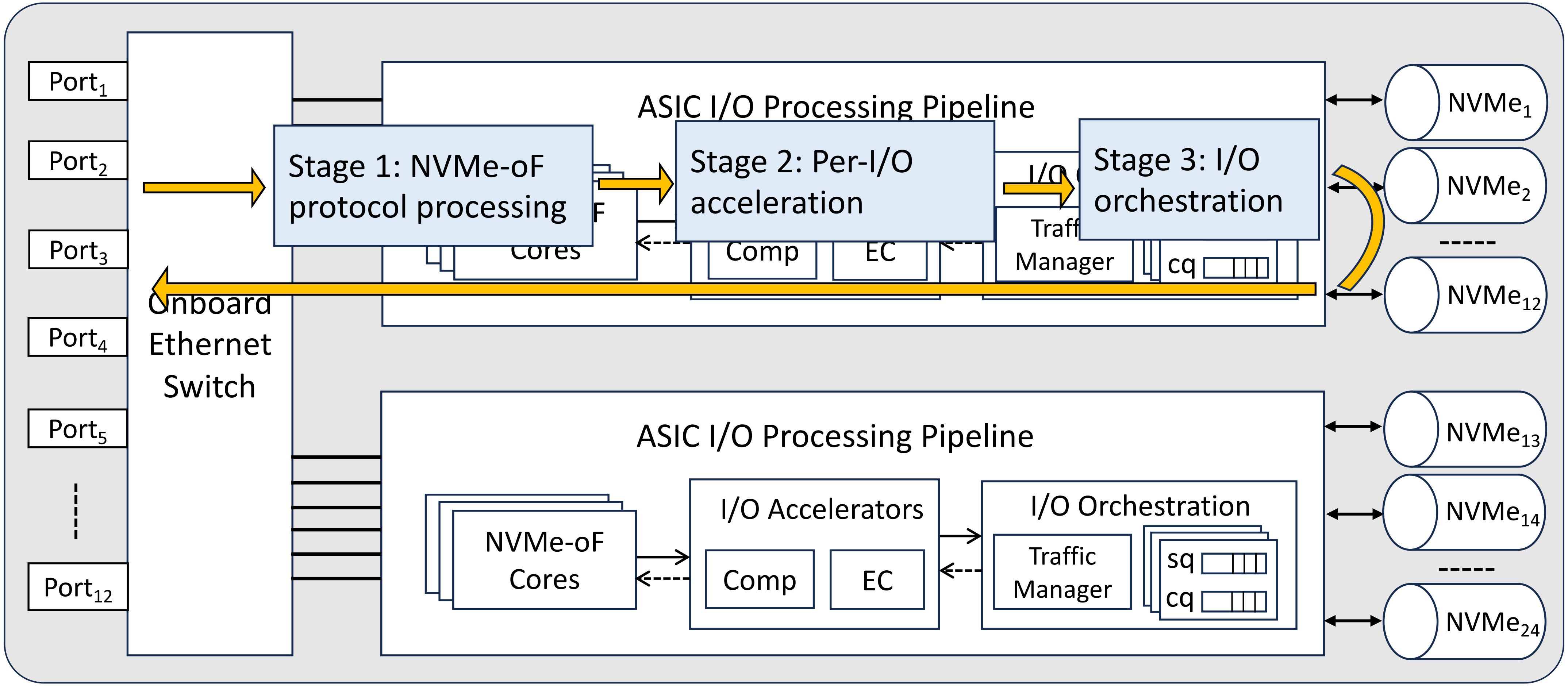
I/O Processing in EBOF



I/O Processing in EBOF



I/O Processing in EBOF



Outline

- EBOF characterization
- Our approach: shadow view
- Flint: elastic block storage over EBOFs
- Conclusion

EBOF Characterization – Experimental Setup

- Hardware testbed
  - Fungible/Microsoft FS1600
  - Dell R7525 server + Mellanox/Nvidia 100GbE CX6 NIC

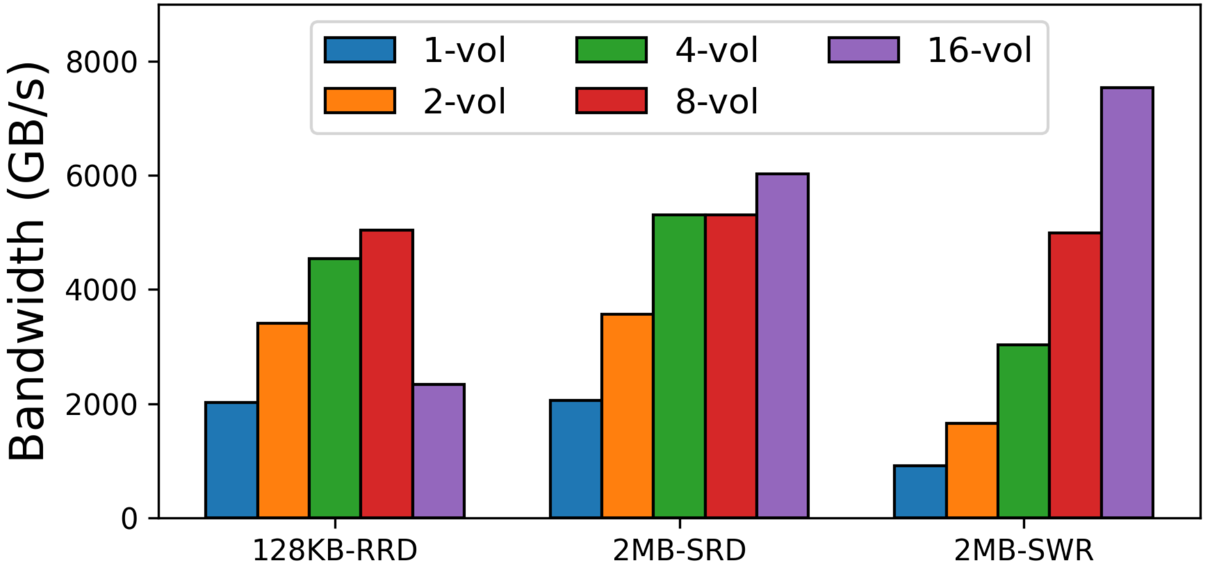
- Software system
  - NVMe over TCP and Linux kernel 5.15

- Application
  - Block-I/O based micro-benchmark
  - Metrics: latency and throughput

EBOF Issue #1: Location Oblivious Placement

- An EBOF volume can’t leverage the massive I/O bandwidth
  - Data volume is statically mapped to a single SSD upon creation
  - Volume placement is randomly chosen when there are multiple candidates

- Experiment results:



EBOF Issue #1: Location Oblivious Placement

- An EBOF volume can’t leverage the massive I/O bandwidth
  - Data volume is statically mapped to a single SSD upon creation
  - Volume placement is randomly chosen when there are multiple candidates

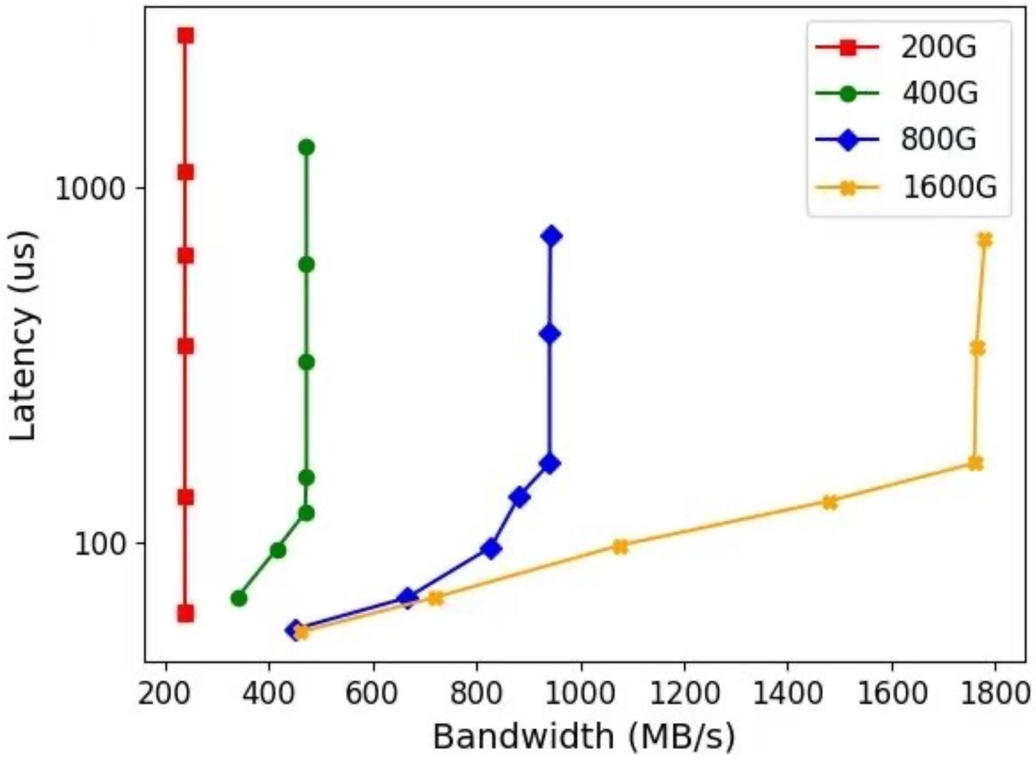
Takeaway: We need flexible and location-aware data placement to unleash EBOF’s bandwidth capacity.

EBOF Issue #2: Size-dependent Bandwidth Allocation

- An EBOF allocates bandwidth share of a volume proportional to its size:

$$QoS\ Upper\ Bound = \frac{EBOF\ Read\ /\ Write\ IOPS}{EBOF\ Capacity} * Volume\ Size$$

- Experiment results:

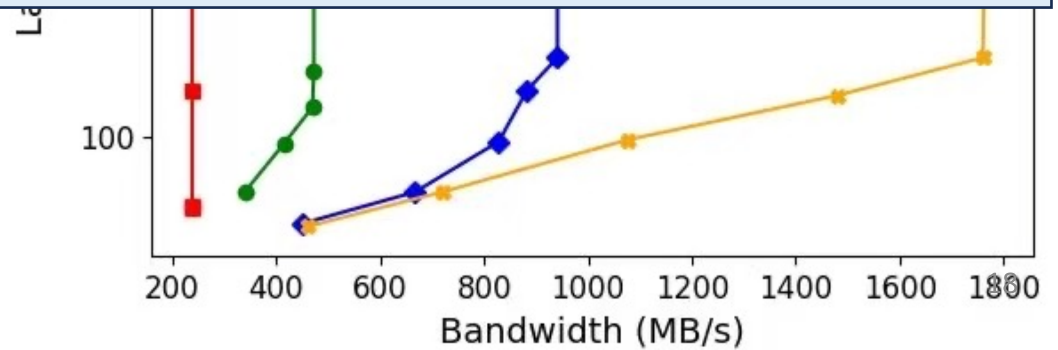


EBOF Issue #2: Size-dependent Bandwidth Allocation

- An EBOF allocates bandwidth share of a volume based on its size:

$$QoS\ Upper\ Bound = \frac{EBOF\ Read\ /\ Write\ IOPS}{EBOF\ Capacity} * Volume\ Size$$

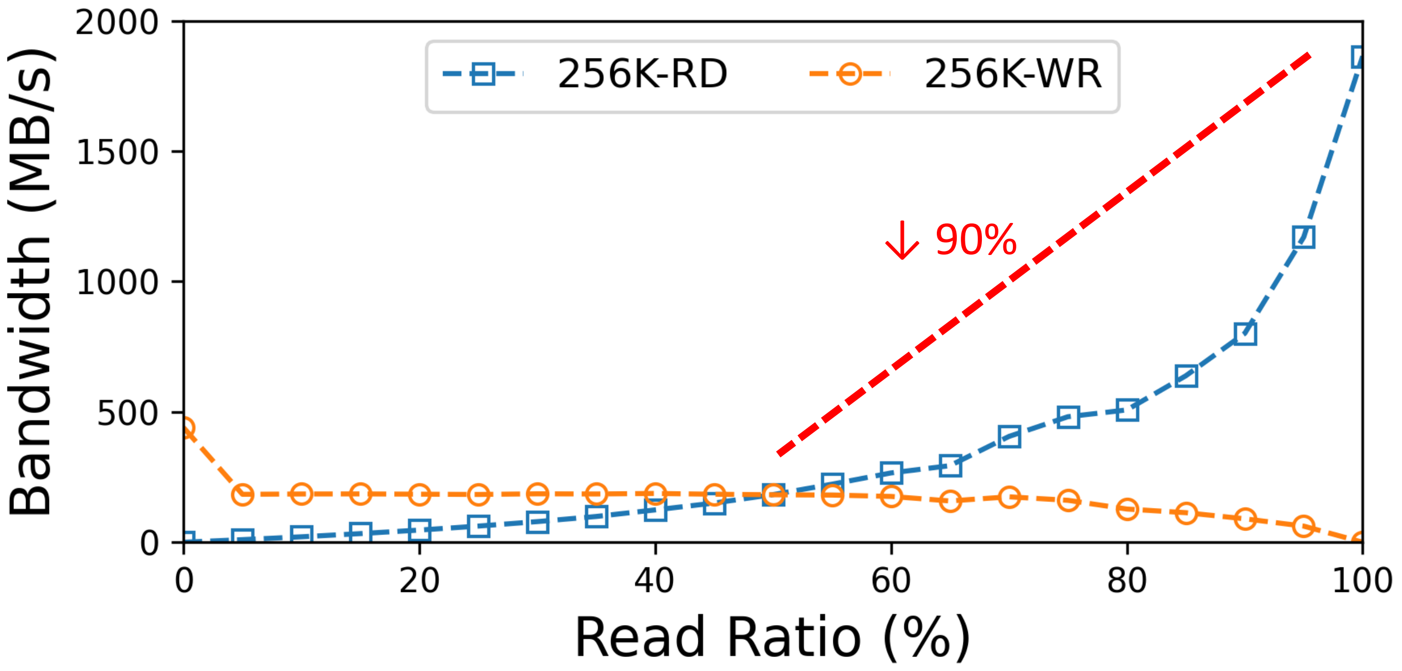
Takeaway: Bandwidth reservation should be decoupled from capacity allocation, and needs to be controlled in an on-demand manner.



EBOF Issue #3: Heavy I/O Interference

- An EBOF volume is tenant and device condition unaware
  - Existing EBOF volumes employ rate limiter to enforce performance isolation

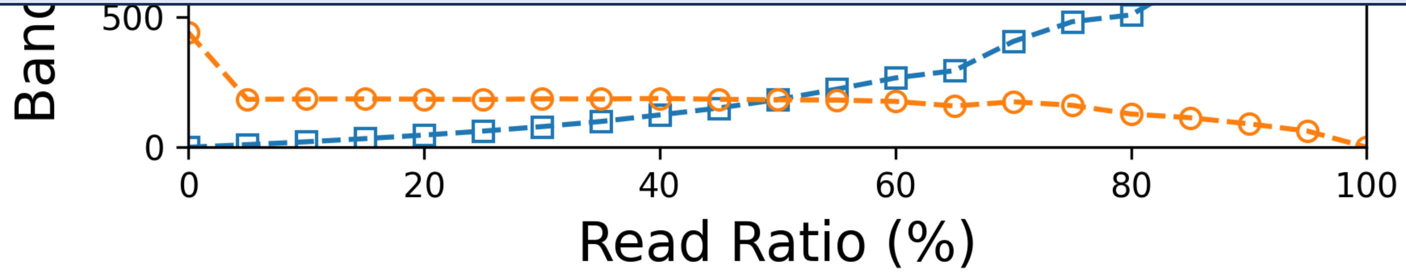
- Experiment results:



EBOF Issue #3: Heavy I/O Interference

- An EBOF volume is tenant and device condition unaware
  - Existing EBOF volumes employ rate limiter to enforce performance isolation

Takeaway: Mitigating I/O interference requires an EBOF to monitor its end-to-end bandwidth availability at the runtime.



Root Cause

- Problem of existing EBOFs
  - Location oblivious placement
  - Size-dependent bandwidth allocation
  - Heavy I/O interference

An EBOF system applies the *smart-sender dumb-receiver* design philosophy and provides backward-compatible volume-oriented storage functionalities.

Outline

- EBOF characterization
- Our approach: shadow view
- Flint: elastic block storage over EBOFs
- Conclusion

Our approach: a distributed software-based EBOF telemetry system called Shadow View

- Continuously monitors the EBOF running condition
- Exposes EBOF internal status to assist efficient I/O processing

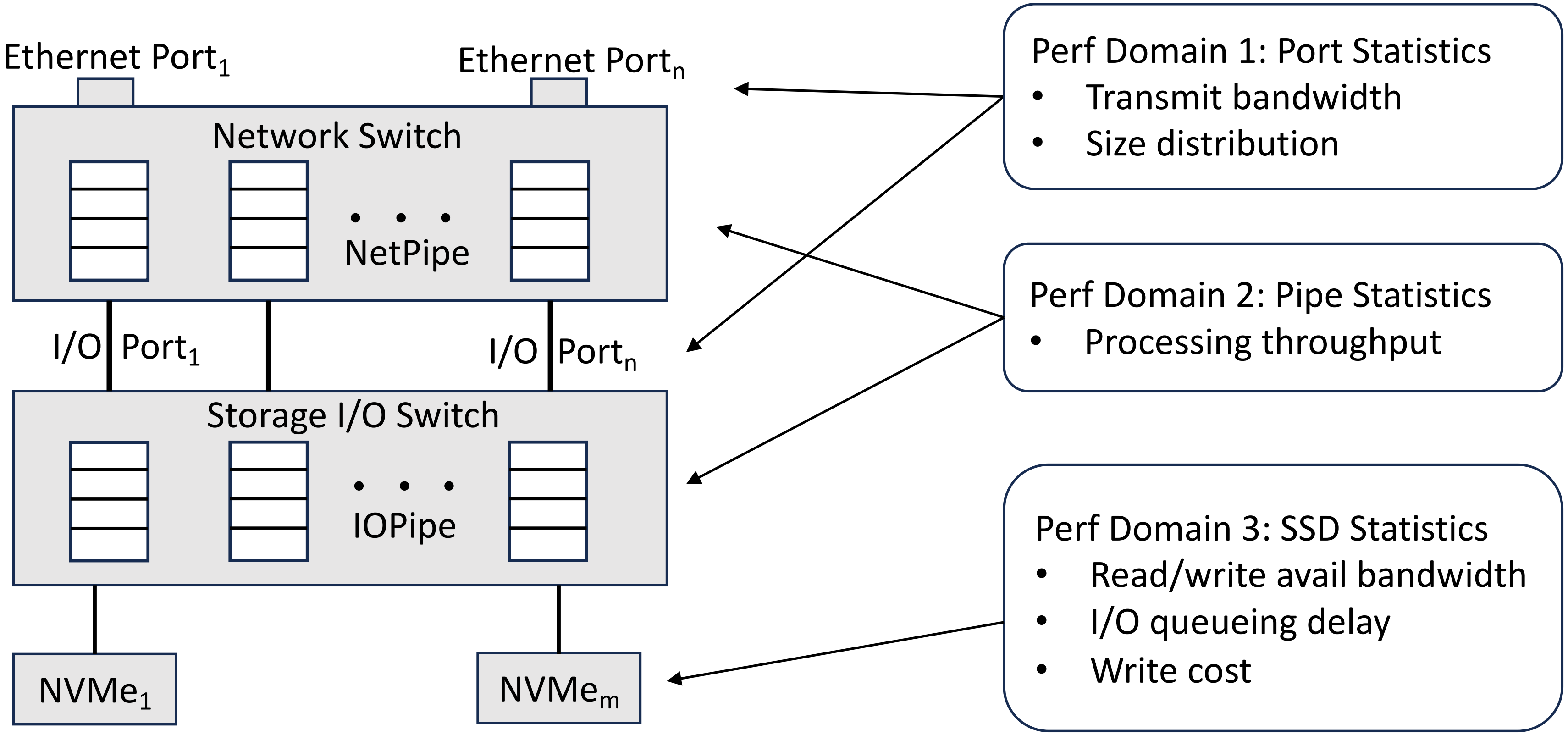
Shadow View: Overview

- Observation:
  - High-speed data center networks make fast data synchronization possible

- Capabilities:
  - Hardware model based running snapshots
  - Three performance monitor domains across the I/O data path

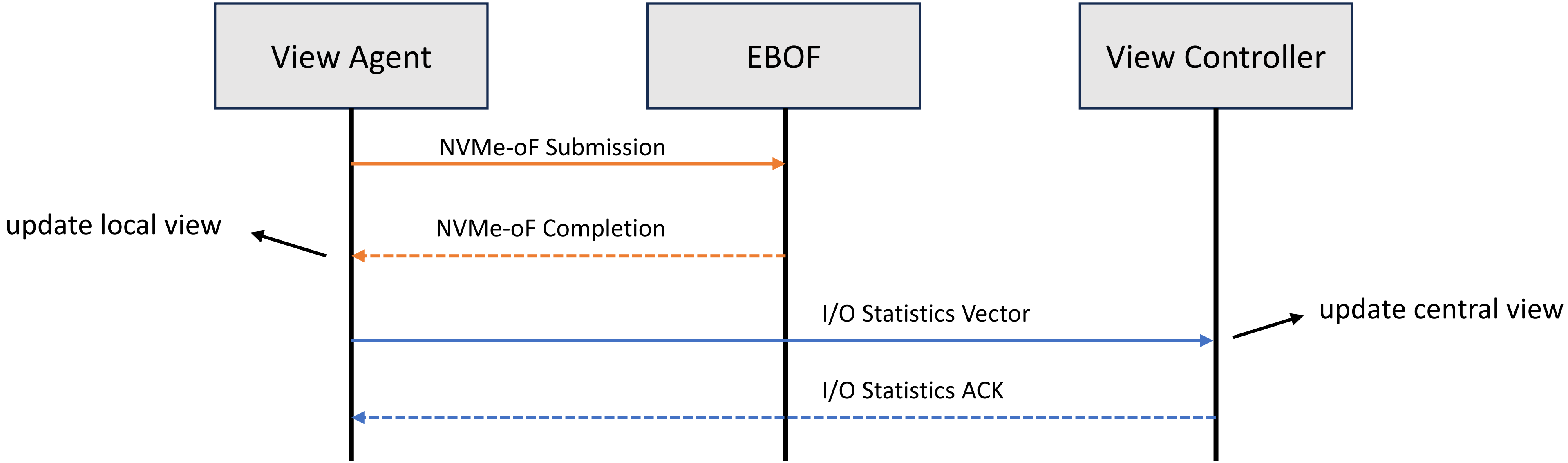
EBOF Hardware Model

- Upper half: network switch, consisting of N bi-directional NetPipes
- Bottom half: storage I/O switch, consisting of M bi-directional IOPipes



Shadow View Construction

- View agent
  - Updates local shadow view
  - Forwards I/O statistics vectors
- View controller
  - Update the central shadow view



Shadow View Synchronization

- Shadow view is collaboratively built by view agent and view controller
  - Use a monotonically increasing counter to represent view recency

- Synchronization modes
  - Push mode
  - Pull mode

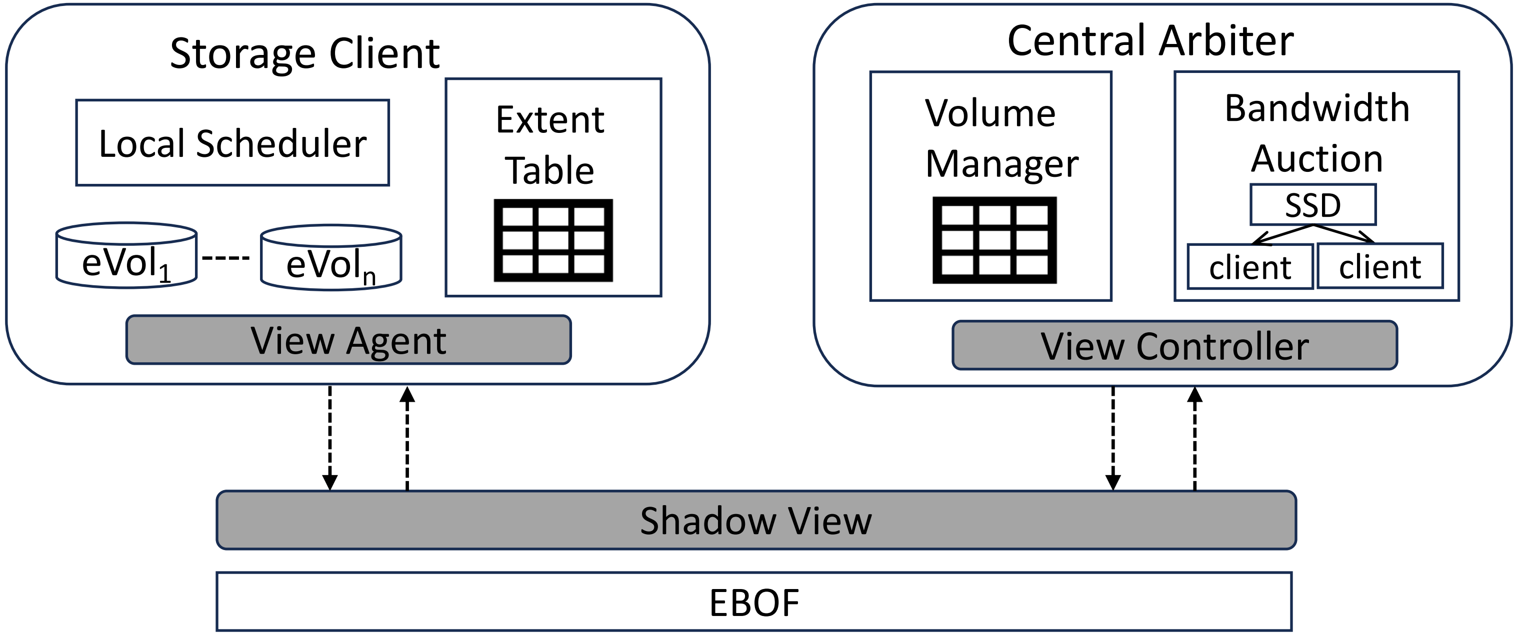
- Clients fetch view copies from the controller to obtain the latest EBOF condition

Outline

- EBOF characterization
- Our approach: shadow view
- Flint: elastic block storage over EBOFs
- Conclusion

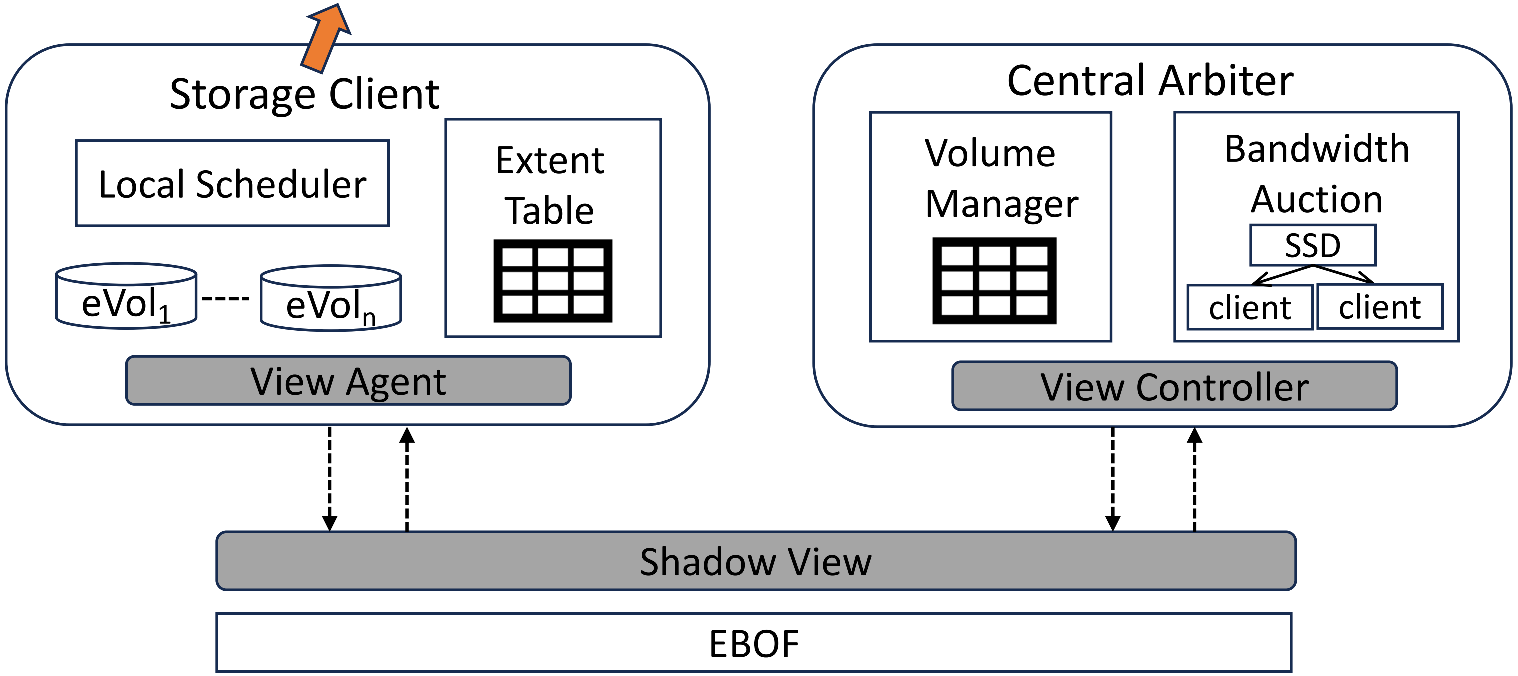
Flint: Elastic Block Storage over EBOFs

- Using shadow view, we build Flint, an elastic block storage system over EBOFs
  - Goals: High throughput, high utilization, and efficient multi-tenancy



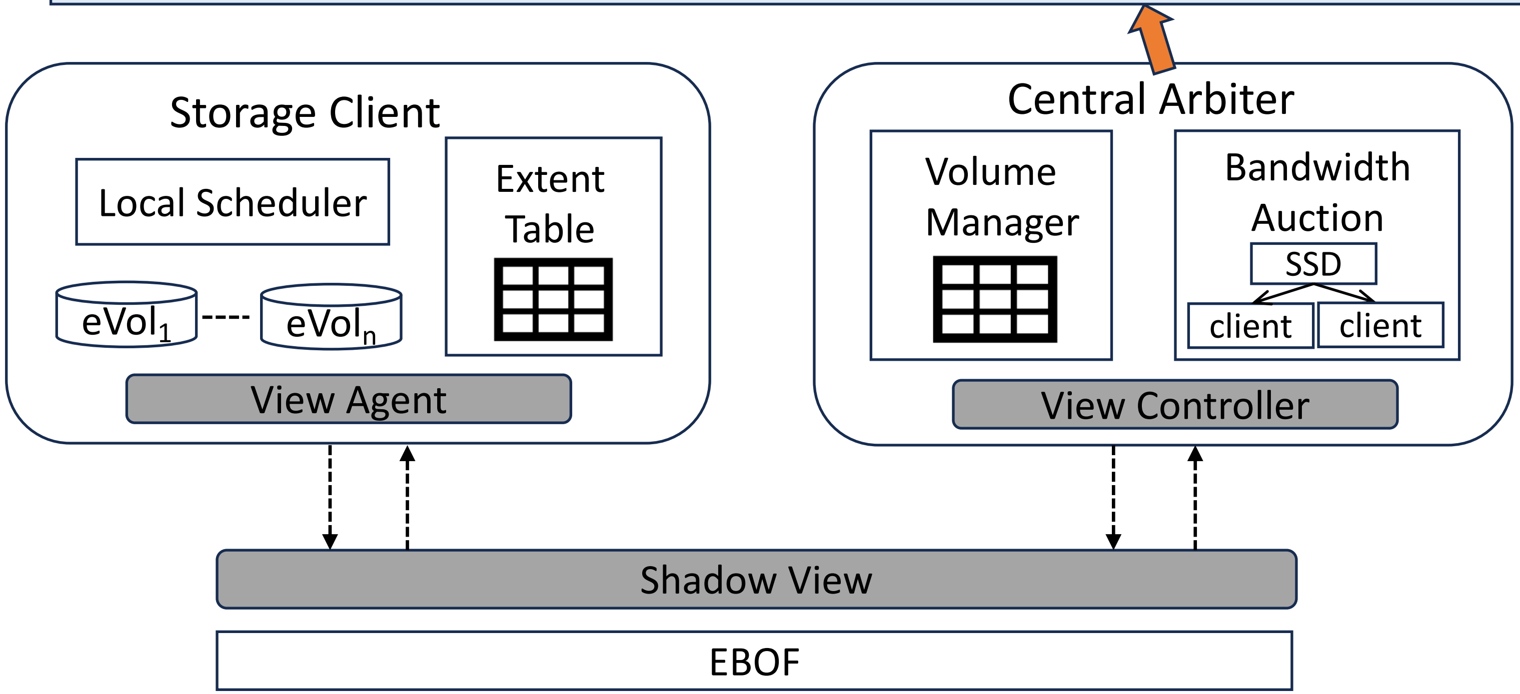
Flint Architecture: Storage Client

- Storage clients co-located with a view agent:
  - Create/delete/update eVols
  - Submit read/write I/Os via a local scheduler



Flint Architecture: Central Arbiter

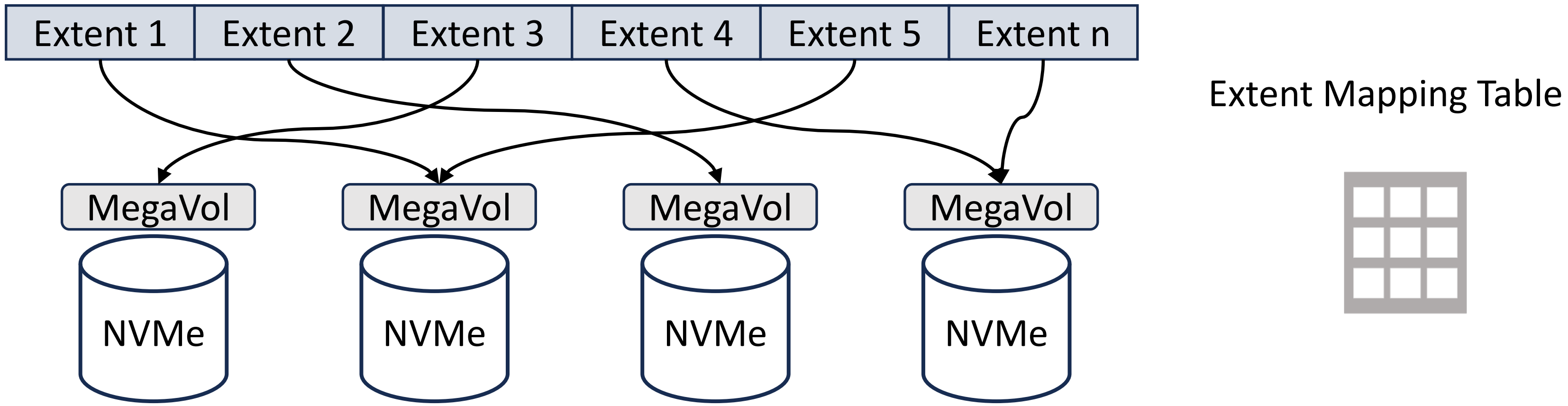
- An arbiter cooperates with a view controller:
  - Places data extents on EBOF and handles management requests
  - Partitions available bandwidth among clients



Flint Elastic Volume (eVol)

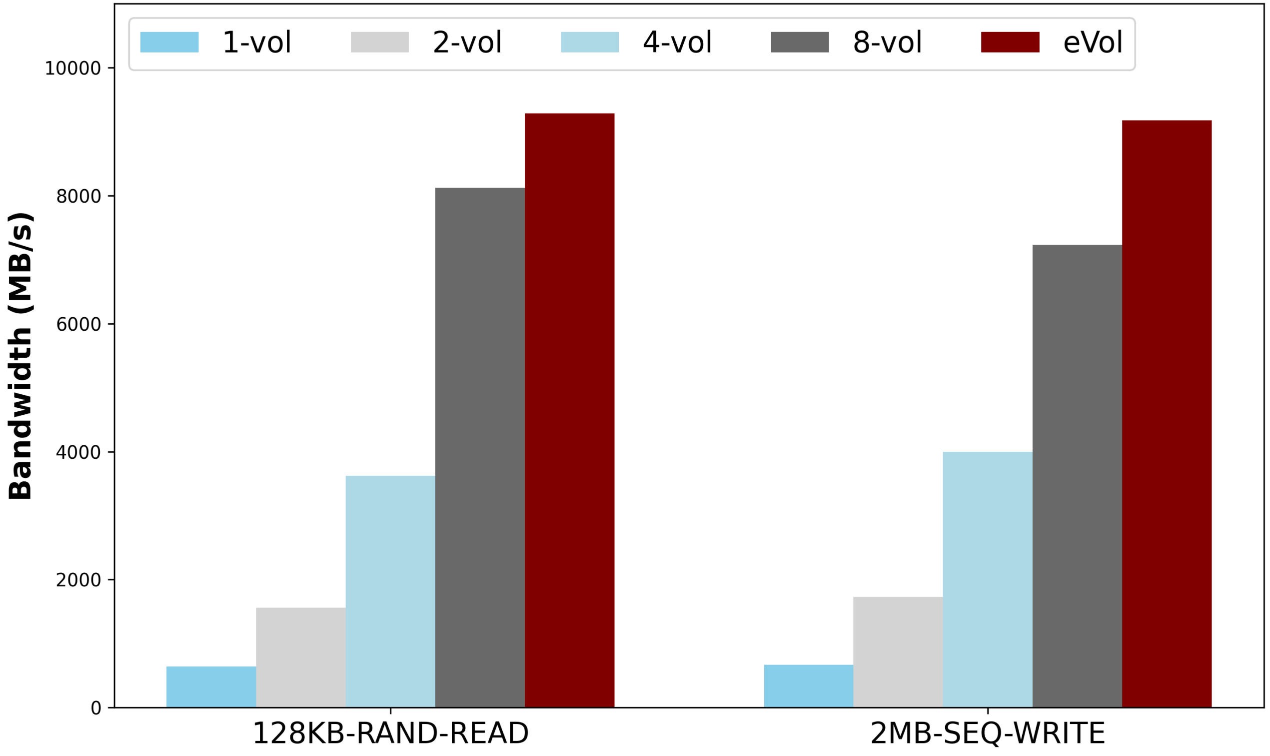
- An eVol consists of fixed-sized extents across multiple SSDs
  - Weighted-score placement function
  - Lazy allocation

- Extent mapping table: <eVol logical address, SSD physical address>



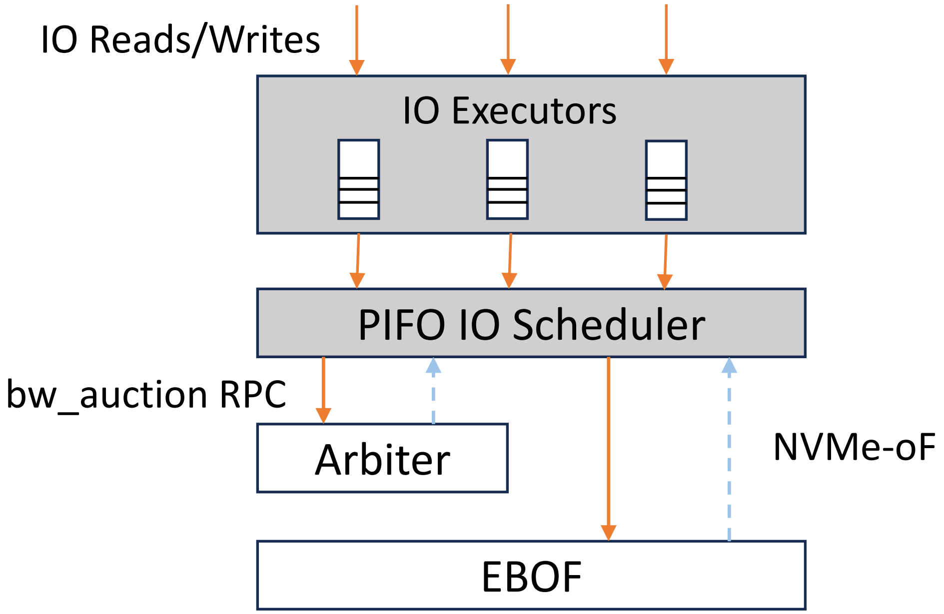
Elastic Volume Performance

- An eVol outperforms a vanilla EBOF volume by 14.5/13.6x regarding read/write
- An eVol consistently achieves higher bandwidth than a LVM volume



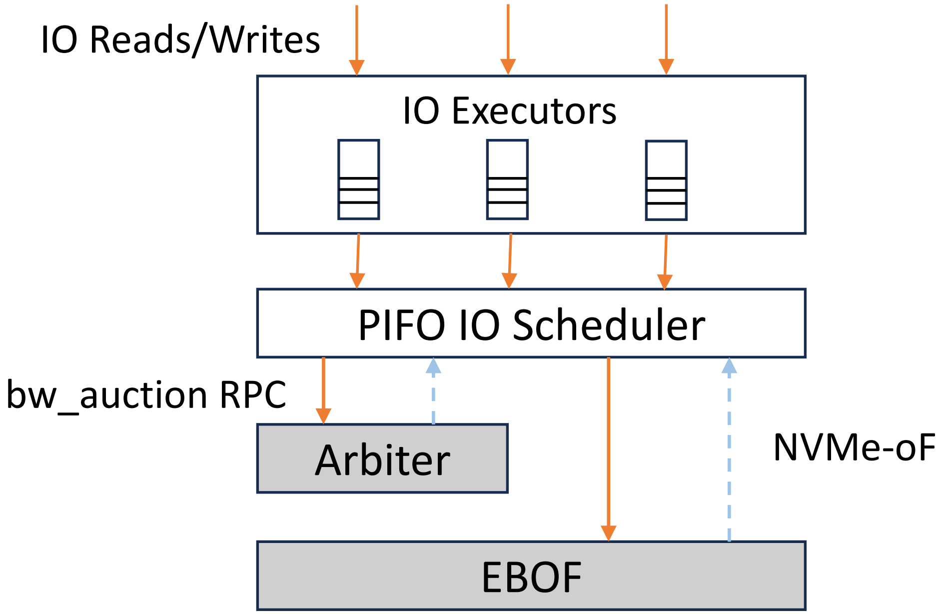
Client Side I/O Scheduling

- Submit read/write I/Os in a Push-In First-Out (PIFO) manner
  - Goal: mitigate head-of-line blocking
  - Rank calculation captures I/O and SSD states from shadow view



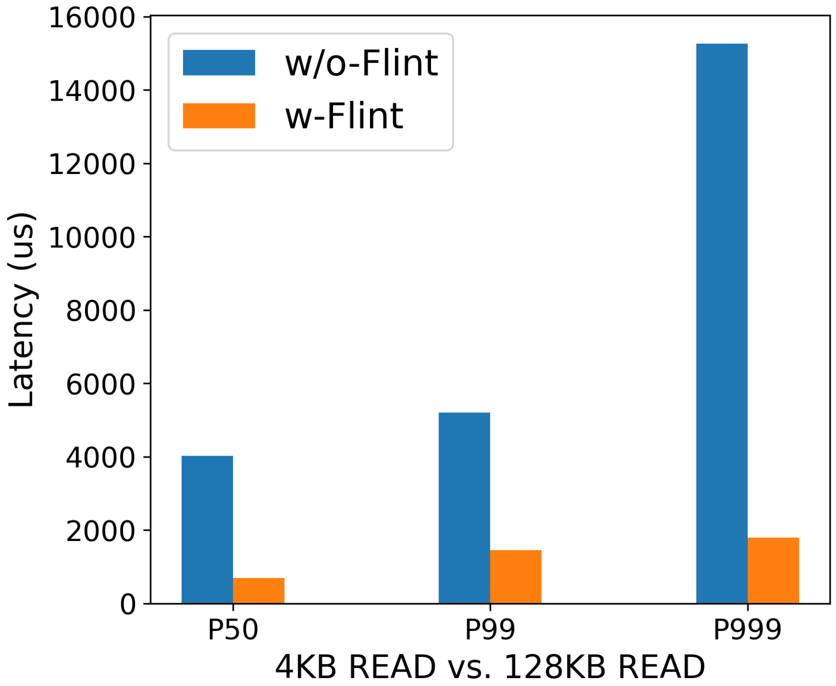
Arbiter Side Bandwidth Auction

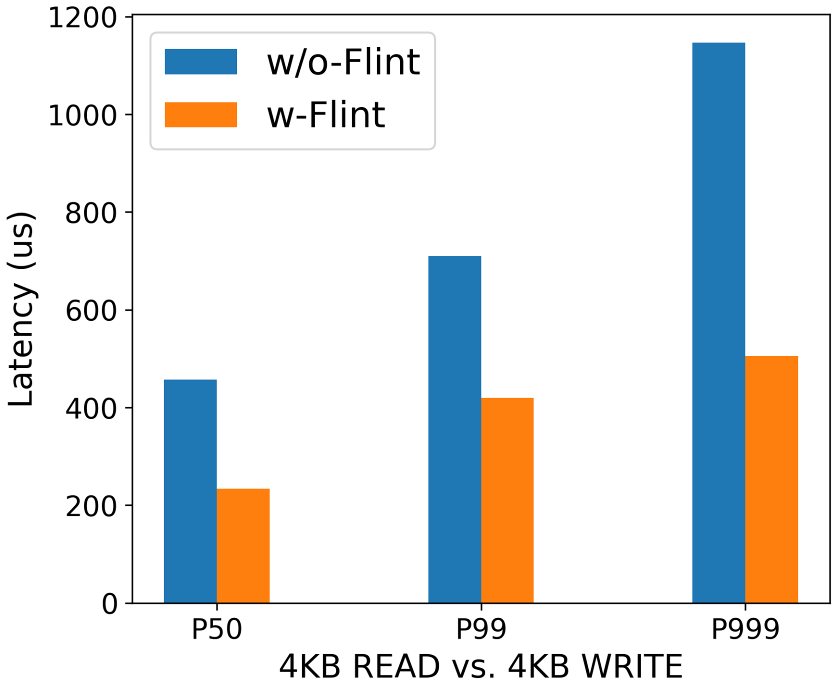
- Partition SSD available bandwidth among active clients
  - Adopt an RTS/CTS-based request and grant scheme
  - Employ Deficit-Round-Robin like algorithm
  - Allocate an I/O bandwidth slice in a batched manner



Flint Mitigates I/O interference

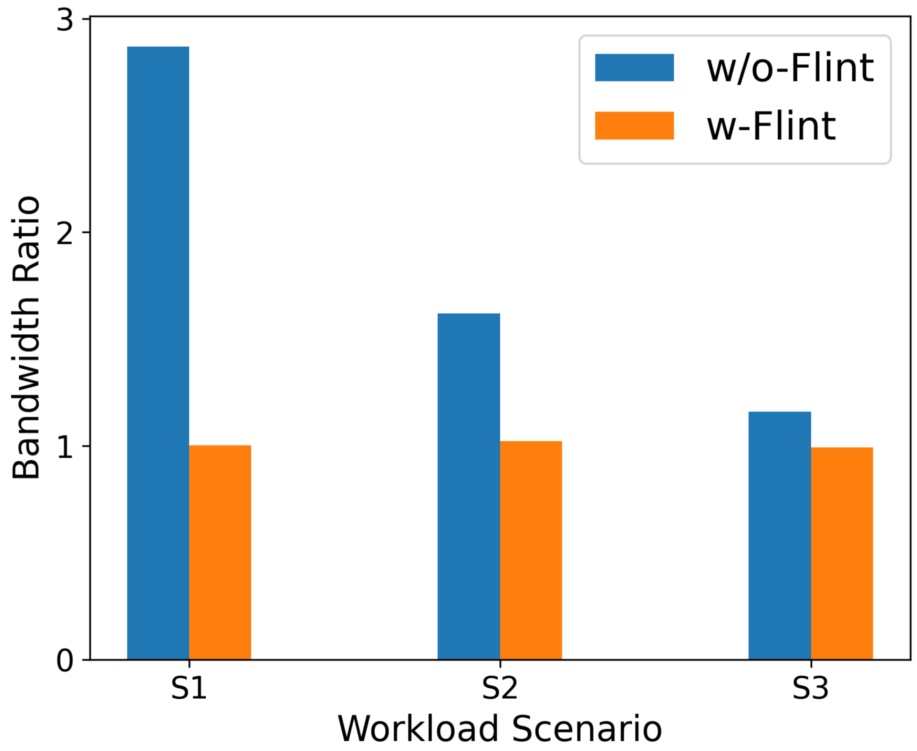
- Two types of co-located I/O streams
  - A victim stream issues 4KB I/Os with a small QD
  - Background streams sending different types of I/Os with a large QD
- Flint reduces 2.6x and 1.7x p99 latency for reads and writes





Flint Achieves Better Multi-tenancy

- Two types of co-located competing I/O streams over eVols
- Flint achieves nearly equal bandwidth share



S1=4KB read vs. 128KB read

S2=128KB read vs. 128KB write

S3=4KB 70% read vs. 4KB 70% write

Outline

- EBOF characterization
- Our approach: shadow view
- Flint: elastic block storage over EBOFs
- Conclusion

Conclusion

- An EBOF is an emerging disaggregated storage hardware substrate
- An EBOF applies the smart-sender dumb-receiver design philosophy
  - Lacks efficient resource allocation, I/O scheduling, and traffic orchestration

- Key Idea: a distributed software-based EBOF telemetry system (Shadow View)
- Flint: an elastic block storage for EBOFs over shadow view
  - Elastic volume, PIFO scheduler, and bandwidth auction