Empowering Storage Systems Research & Development with **NVMeVirt**

Jaehoon Shim
(mattjs@snu.ac.kr)

Seoul National University

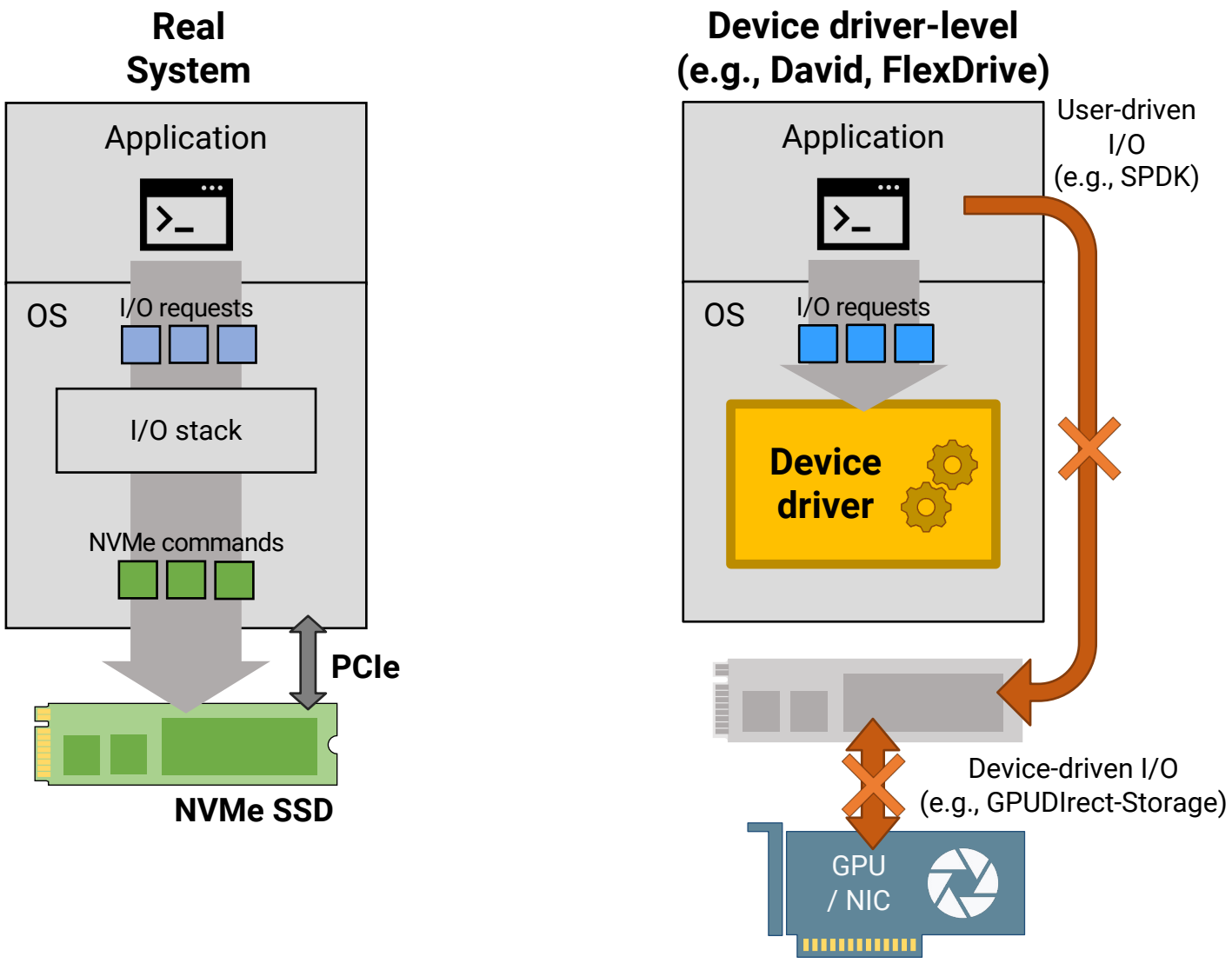
Dilemma of Emulator

- Emulators can facilitate advanced storage development by actualizing novel device concepts
  - NVM SSD, KV-SSD, ZNS SSD, FDP, computational storage, …
  - Advanced commands (e.g., batched read and write, transactional commands)
  - Can implement the concepts in software

- Cannot support some I/O models and modern storage configurations

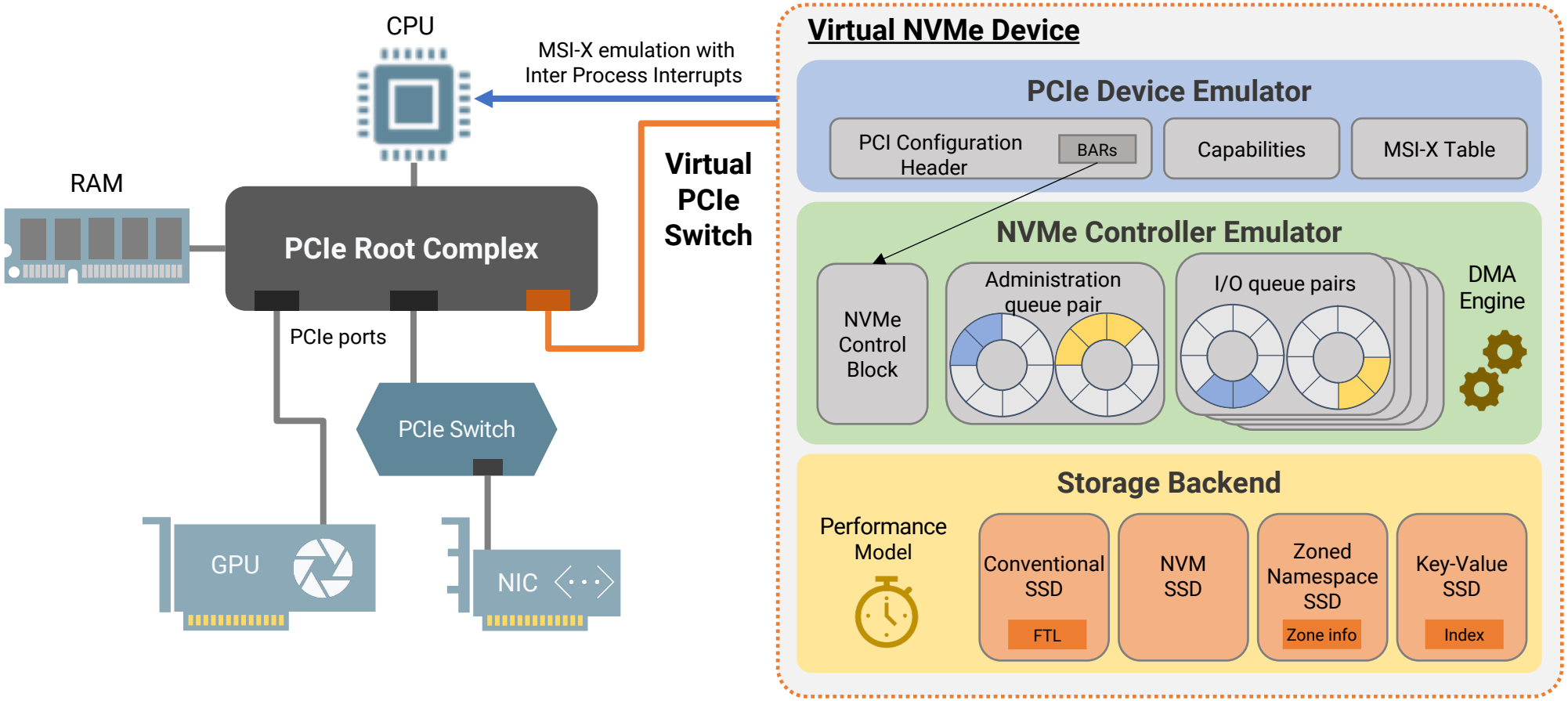
	Simulators		Emulators				
	Trace-driven [30, 36]	**Full-system [10, 22, 49]**	**VM-based [12, 32, 55]**	**Block-driver level [56]**	**NVMe-driver level [35]**	**HW platforms [21, 28]**	**NVMeVirt**
Deployable in real environments	No	Yes	Yes	Yes	Yes	Yes	**Yes**
Execution speed	Fast	Very slow	Slow	Fast	Fast	Real-time	**Real-time**
NVMe Multi-queue support	No	Yes	Yes	No	Yes	Yes	**Yes**
NVMe interface modification	Impossible	Easy	Easy	Impossible	Easy	Difficult	**Easy**
Low-latency device support	Possible	Possible	Difficult	Possible	Possible	Difficult	**Possible**
Kernel bypassing with SPDK	No	No	Yes	No	No	Yes	**Yes**
PCI peer-to-peer DMA support	No	No	No	No	No	Yes	**Yes**
NVMe-oF target offloading	No	No	No	No	No	Yes	**Yes**

Existing Approaches



NVMeVirt

- A virtual NVMe device in software
  - A light-weight kernel module that presents a native NVMe device to the system



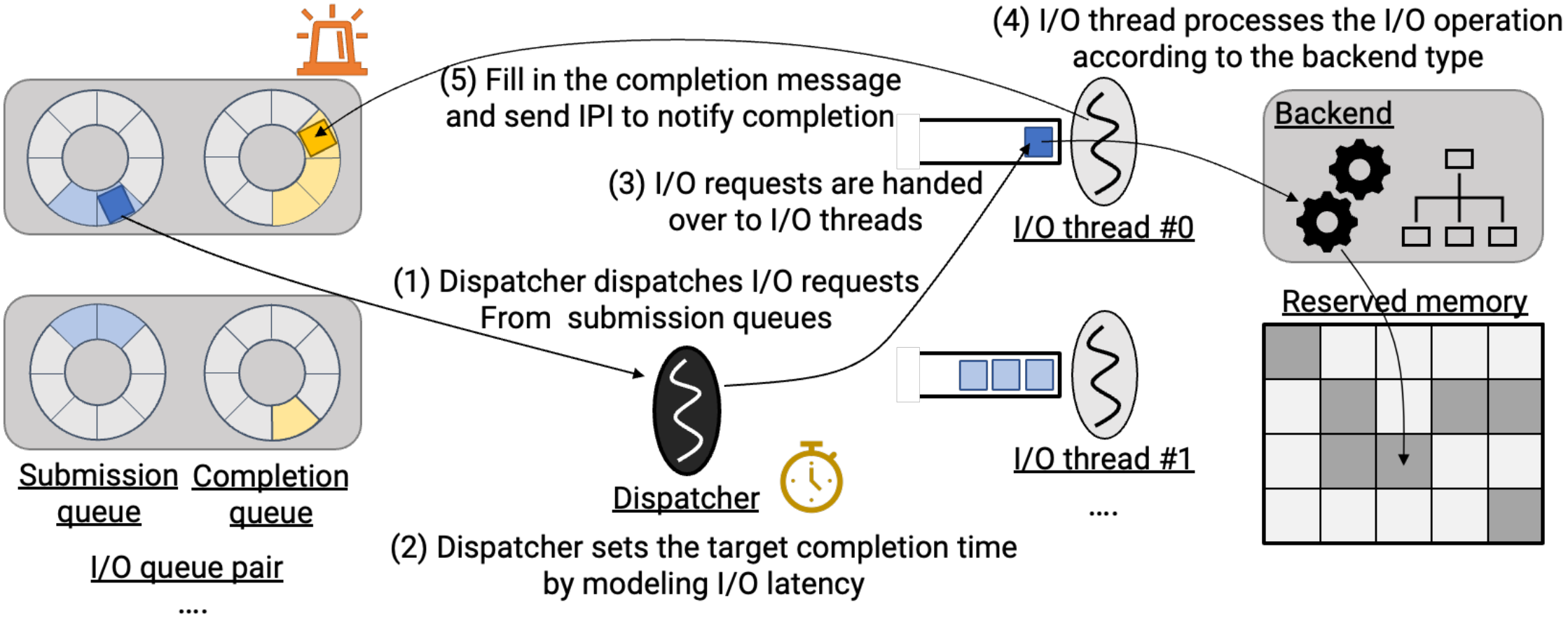
Design Issues

- How to create a virtual PCI device instance in software?
  - Make a PCI device instance indirectly through a virtual PCI bus
  - The virtual PCI bus presents the PCI configuration header to the PCI subsystem
  - No modification required in the Linux kernel

- How to emulate memory-mapped accesses to the device registers?
  - The host updates a memory-mapped region of the PCI BAR
  - Used to configure the device and ring the doorbells
  - A dedicated dispatcher thread polls the NVMe control block and doorbells

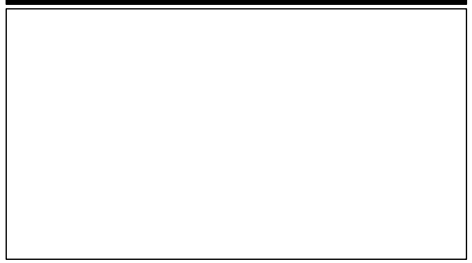
I/O Handling in NVMeVirt

- One dispatcher + multiple I/O threads (each pinned to a core)



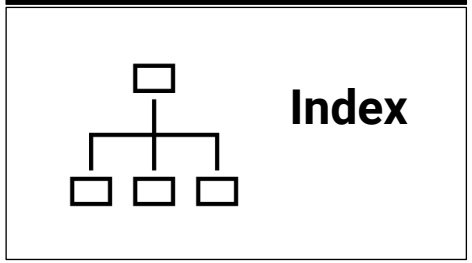
Available Backends

NVM SSD



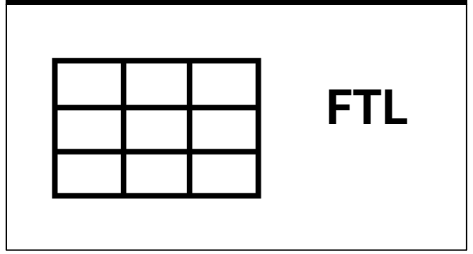
- In-place update
- No GC
- OptaneDC-like SSDs

KV-SSD



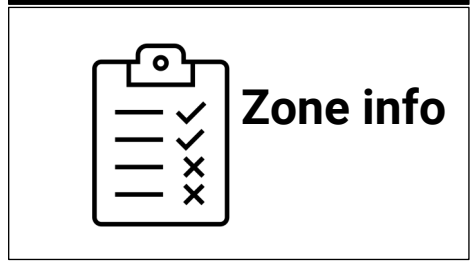
- SNIA KV APIs: store, retrieve, exist, delete, iterate
- OpenMPDK-compliant
- Hash-based index

Conventional SSD



- Page-mapping FTL
- GC
- Write buffer
- Multiple FTL instances

ZNS SSD



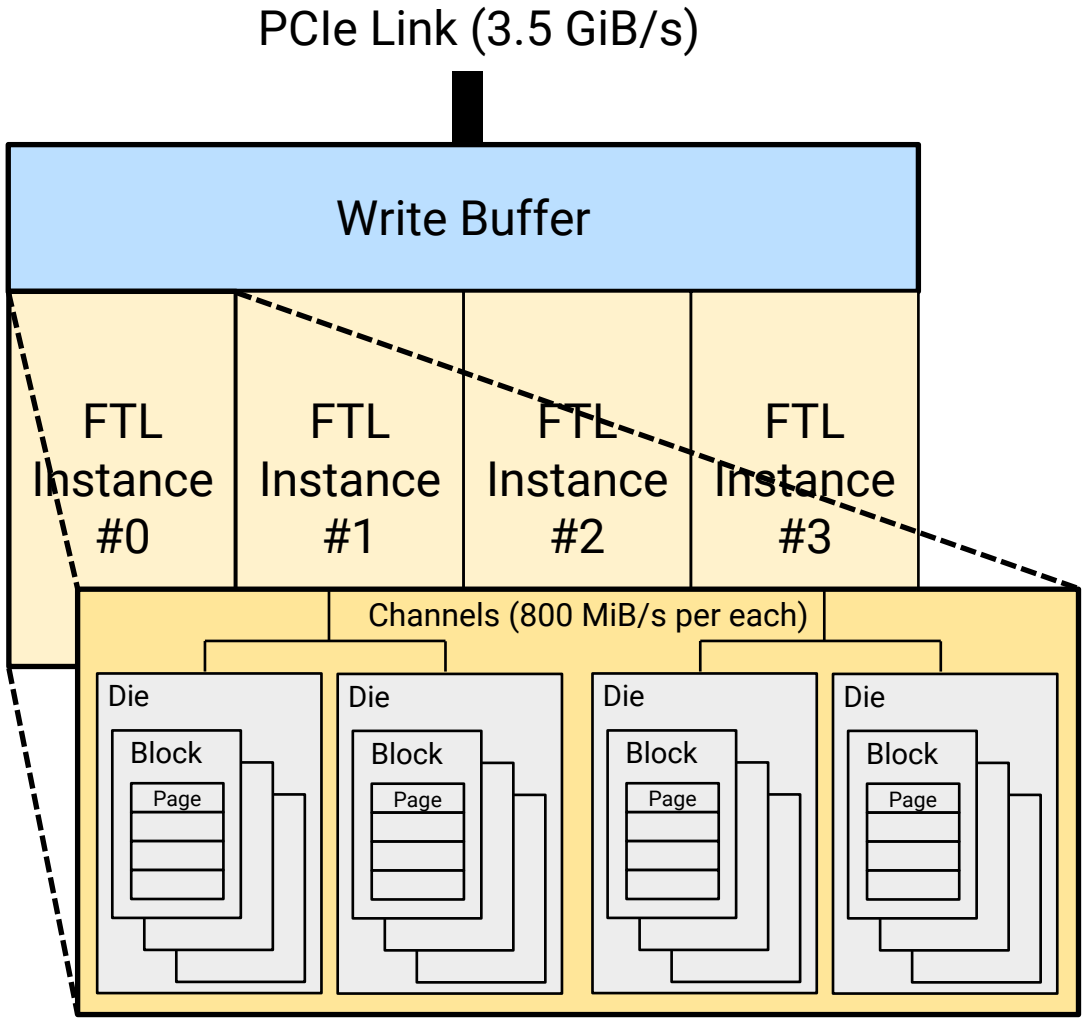
- Zone management
- Write buffer
- Multiple FTL instances

Simple performance model

Advanced performance model

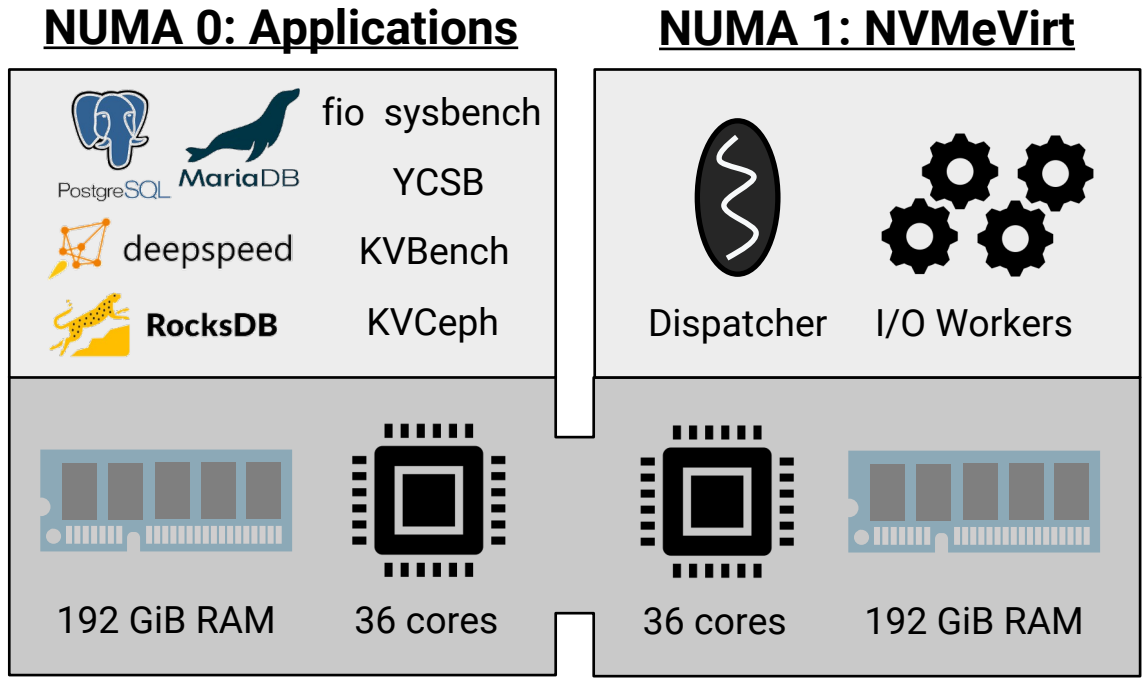
Advanced Performance Model

- Models complicated SSD internals
  - A full-scale page-mapped FTL with GC
  - Model the on-device write buffer
  - Model the parallel architectures in modern SSDs
    - Multiple FTL instances
    - Multiple dies and channels operating independently
    - Use superblock as the operation unit
    - PCIe link and channels with limited aggregate bandwidth
  - Reserved N% of space as OP area



Evaluation

- Intel Xeon Gold 6240 x 2
  - 18 cores or 36 threads per socket
- Total 384GiB RAM
  - 192GiB reserved for NVMeVirt
- Implemented as a kernel module (~ 9,000 LoC)
  - Tested using Linux kernel 5.15

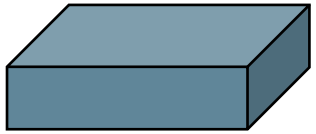


Samsung 970 PRO

- Conventional SSD
- 512 GB

Intel P4800X

- OptaneDC NVM SSD
- 350 GB



Samsung KVSSD

- Hash-based FTL
- 3.84 TB

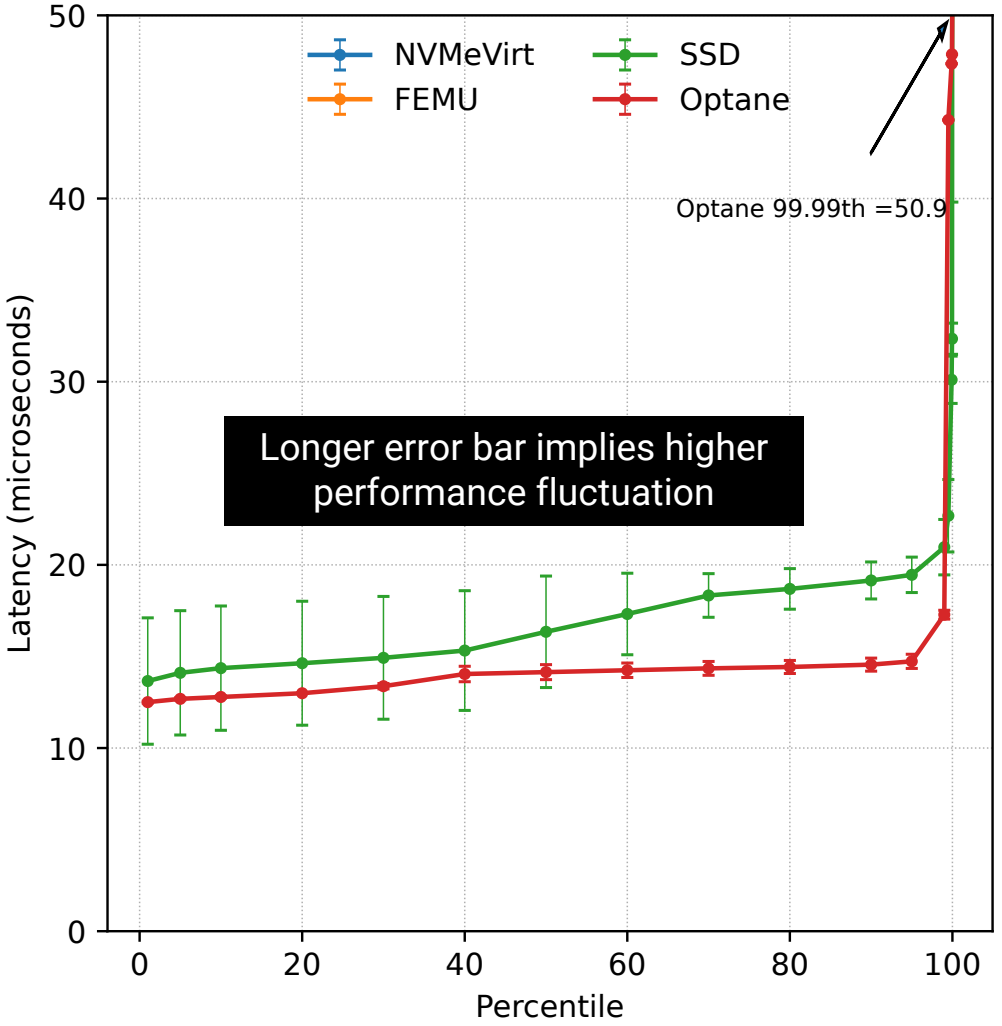
Prototype ZNS SSD

- 96 MiB zones
- 192 KiB write unit
- 32 TB

Ultrastar DC ZN540

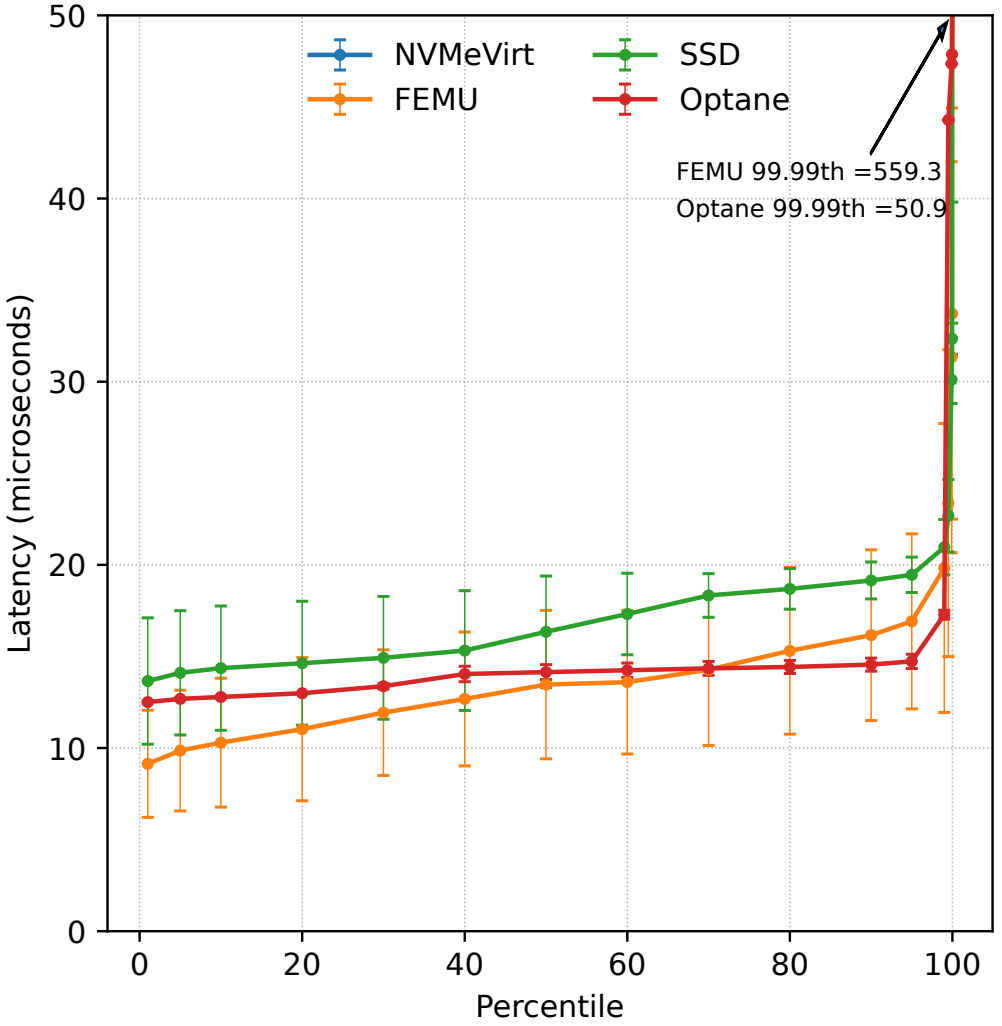
- 2048 MiB zones
- Support WB
- 4 TB

Emulation Quality: Performance Variance



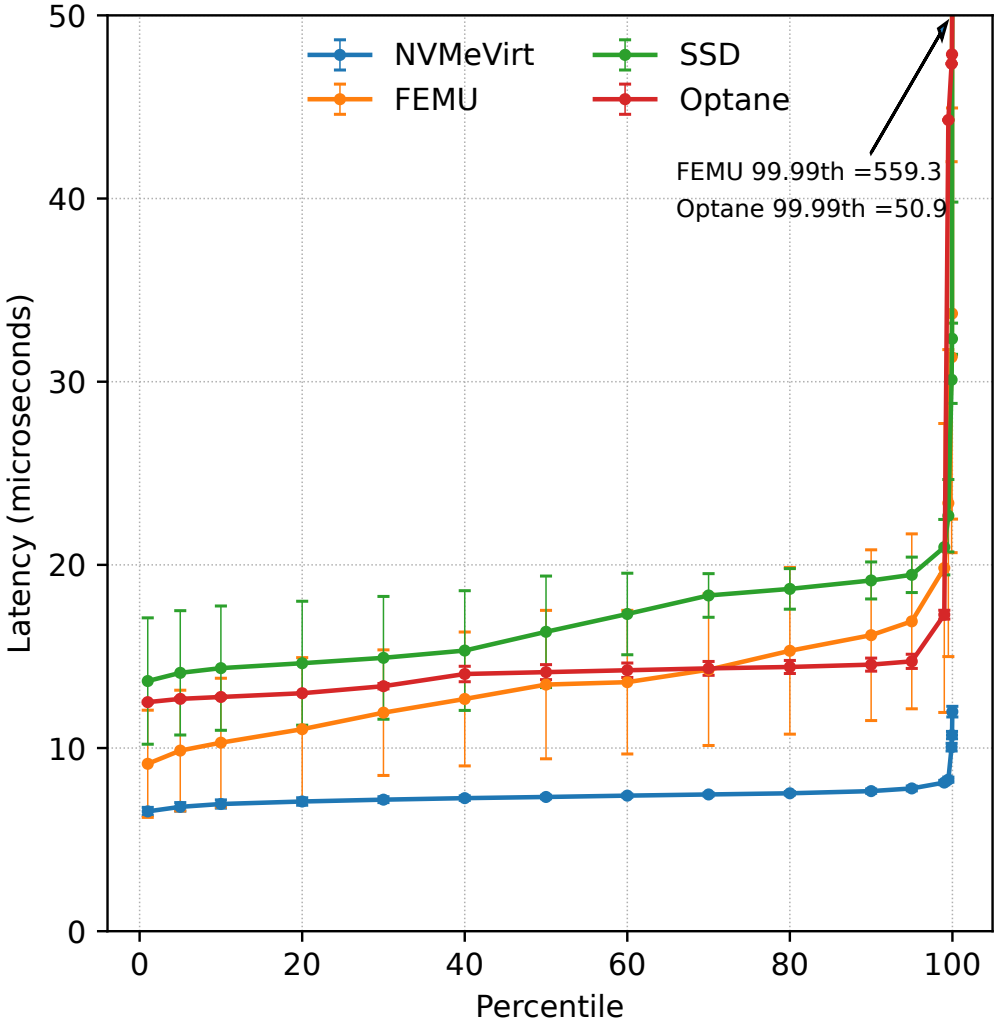
- Distribution of percentiles for 10 runs
  - Each run does 4 KiB random writes using fio
  - Error bar indicates the standard deviation for the percentile

Emulation Quality: Performance Variance



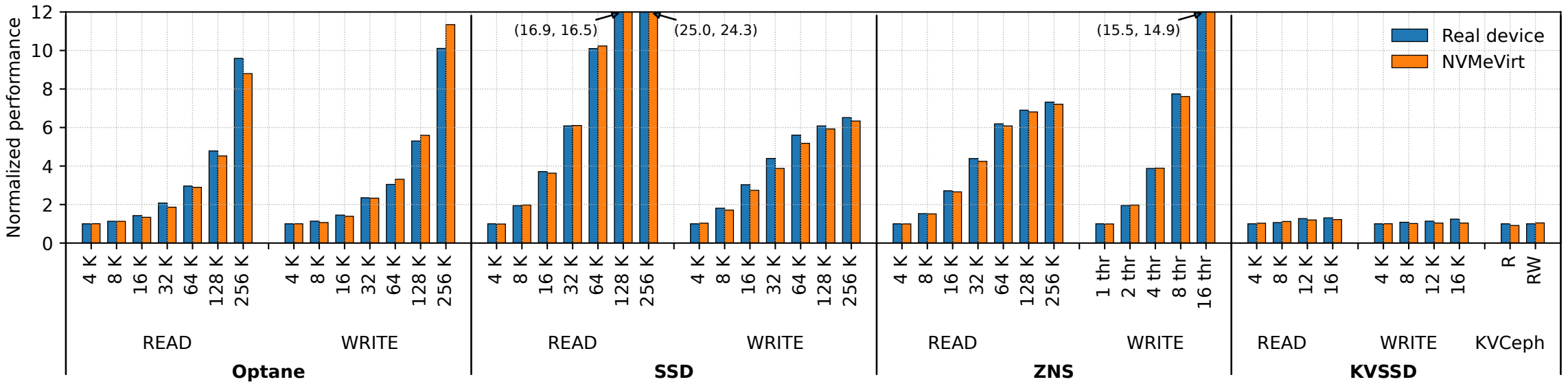
- Distribution of percentiles for 10 runs
  - Each run does 4 KiB random writes using fio
  - Error bar indicates the standard deviation for the percentile
- FEMU exhibits a long tail latency and high run-by-run performance fluctuation
- FEMU would not be able to consistently emulate high-performance NVM SSDs

Emulation Quality: Performance Variance



- Distribution of percentiles for 10 runs
  - Each run does 4 KiB random writes using fio
  - Error bar indicates the standard deviation for the percentile
- FEMU exhibits a long tail latency and high run-by-run performance fluctuation
- FEMU would not be able to consistently emulate high-performance NVM SSDs
- NVMeVirt provides low latency with little performance variation

Performance Comparison to Real Devices



fio random access latency

OpenMPDK KVBench agg. BW

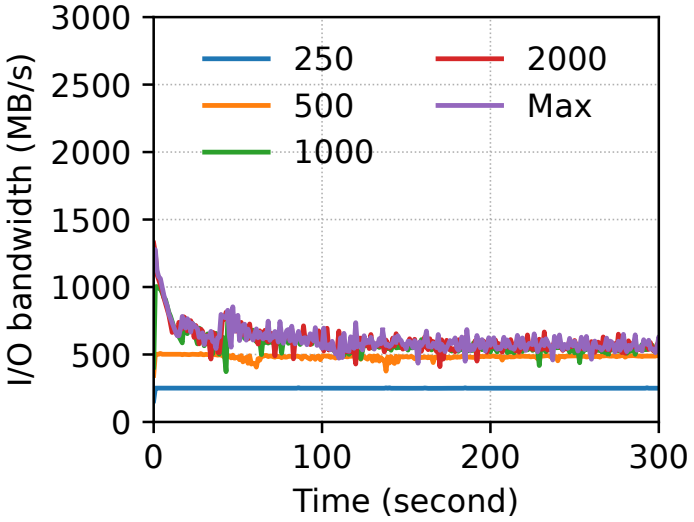
KVCeph agg. BW

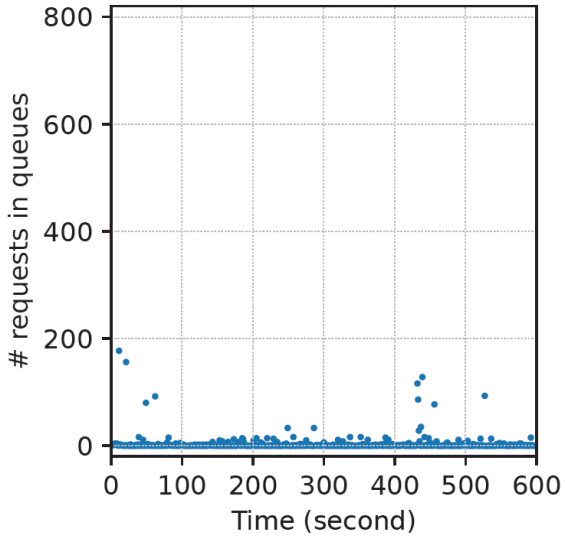
NVMeVirt can replicate the real devices' performance closely

Harmonic mean of performance differences = 1.17%

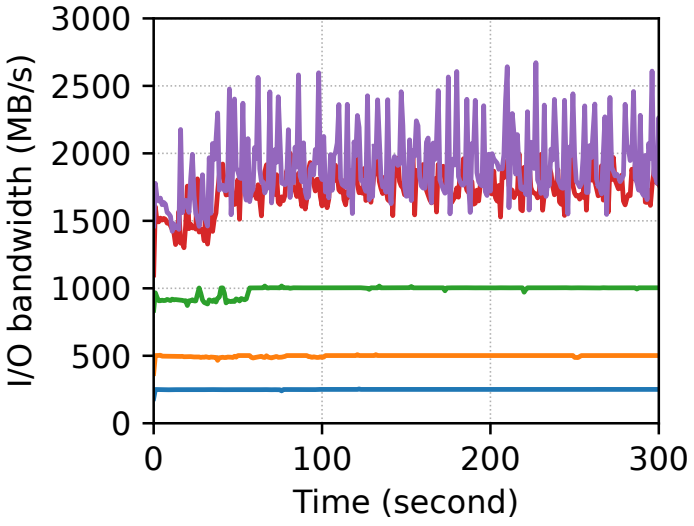
Case Study: Sysbench OLTP

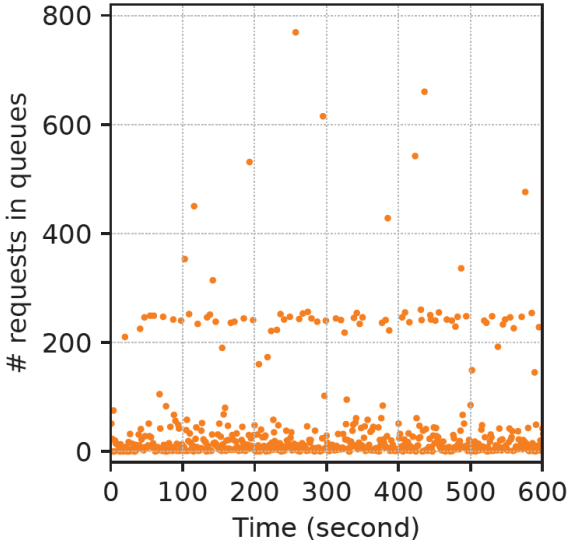
MariaDB



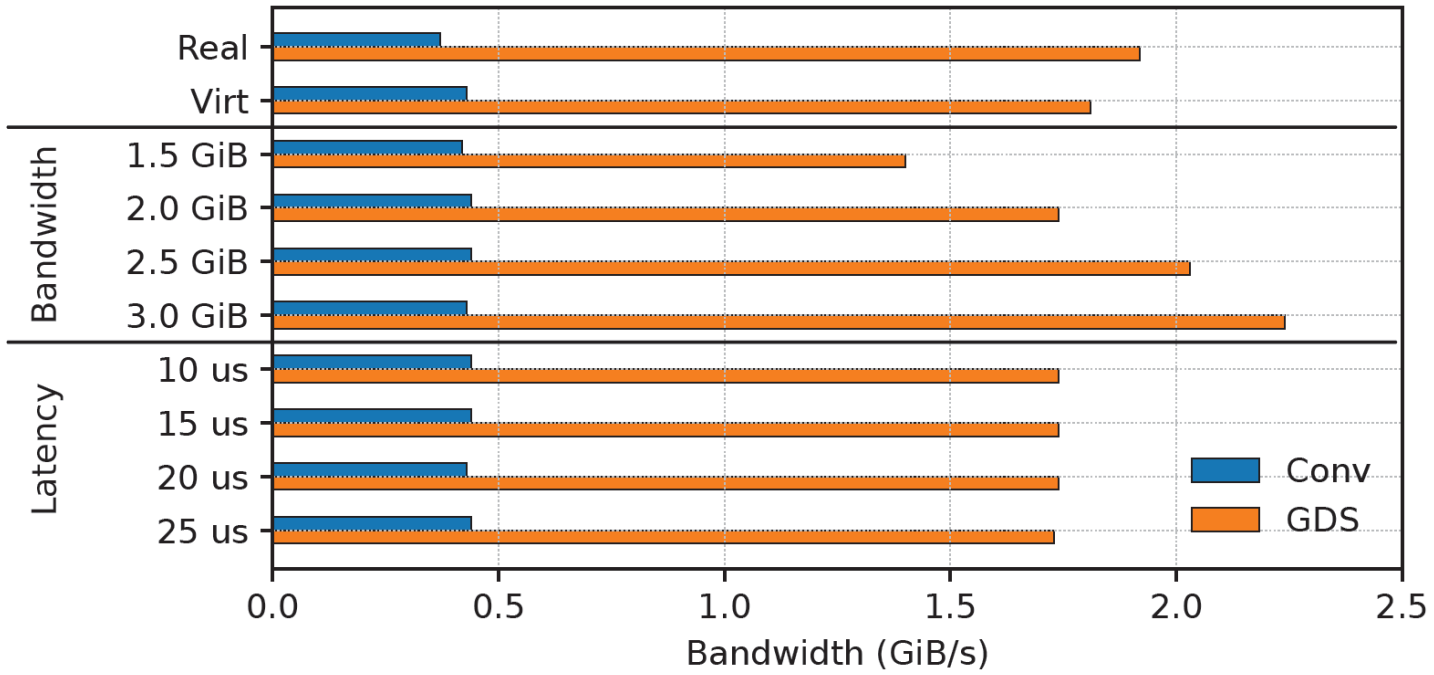


PostgreSQL

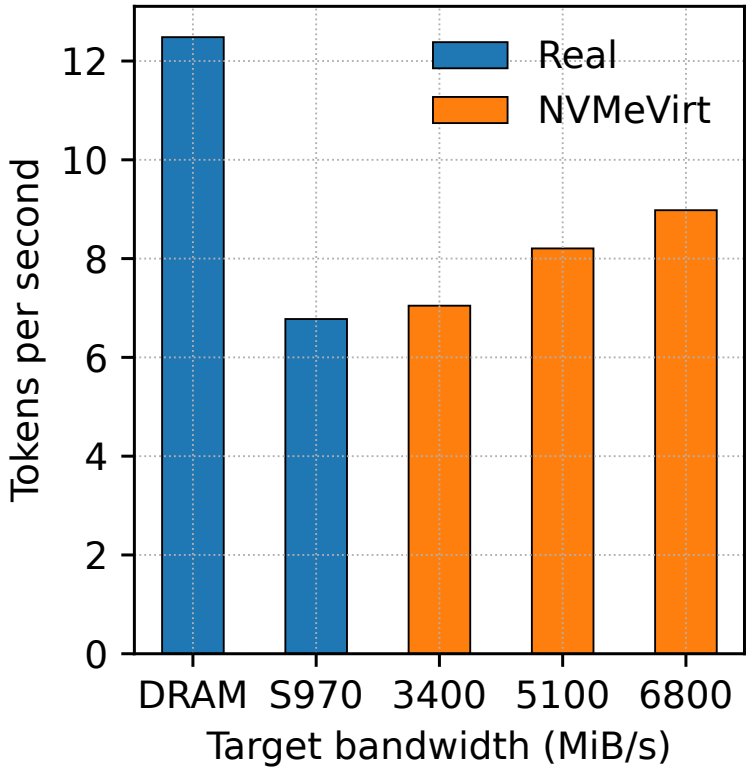




Case Study: AI Workloads



Checkpointing performance of Megatron DeepSpeed



Inference performance of OPT-30B

Use Cases

- Fast prototyping for new NVMe interface extensions
- Finding and improving software bottlenecks in the storage stack
- Analyzing application's scalability on future high-performance storage devices
- Investigating performance impact of hardware parameters (e.g., MDTS)
- Developing a new device-centric architecture
- Benchmarking and performance testing & monitoring

On-going Work

- A backend for computational storage (compliant to NVMe Spec) – *Done*
- CMB support – *Done*
- A backend for FDP – *In progress*
- Dynamic multiple SSD instances – *In progress*
- CXL support – *In progress*

Give It a Try!

- Please check our paper (FAST ‘23, ToS ‘24) for more details
- Code: https://github.com/snu-csl/nvmevirt
- Tutorial slides: http://csl.snu.ac.kr/nvmevirt/tutorial.pdf