NVMeDirect 2.0: An Enhanced User-space I/O Framework for NVMe SSDs

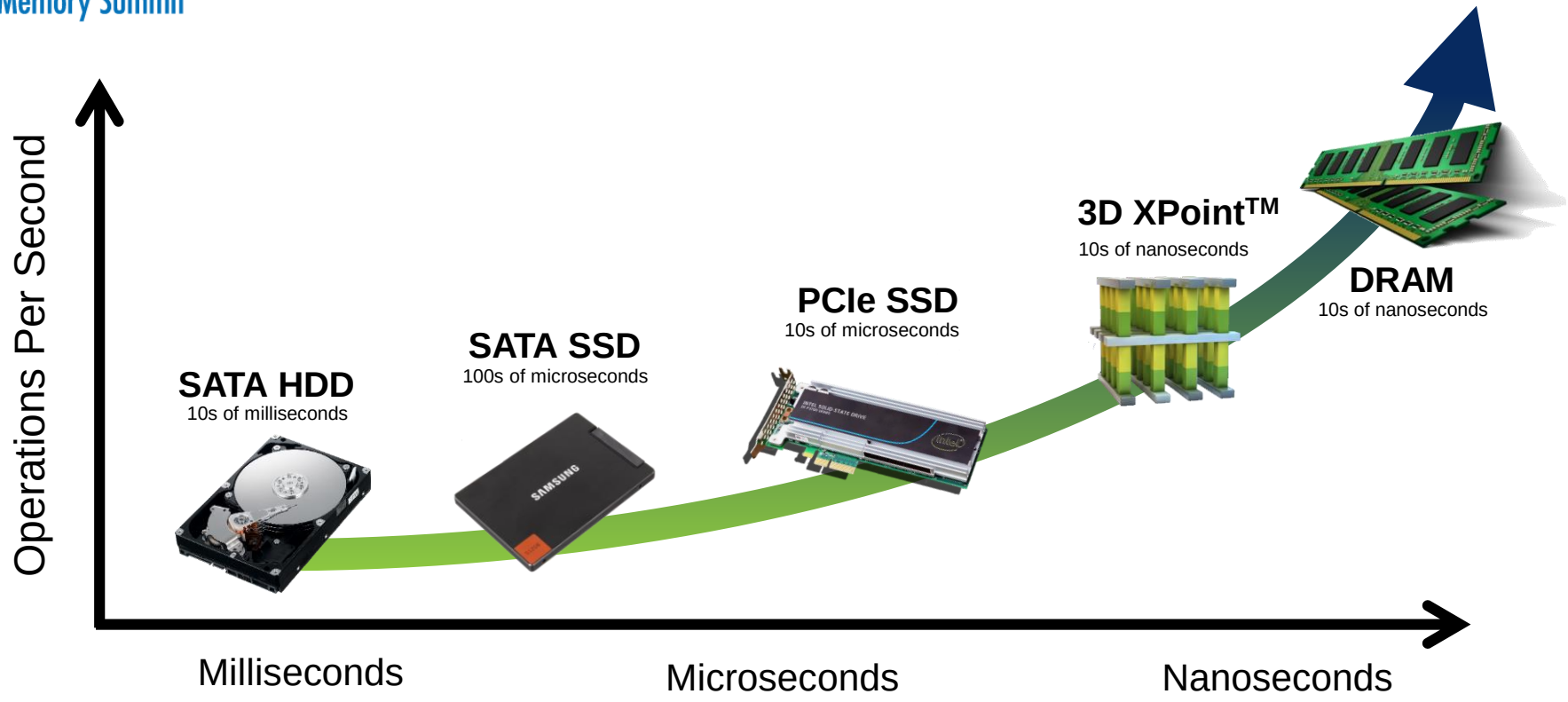
Hyeong-Jun Kim, Jin-Soo Kim

hyeongjun.kim@csl.skku.edu

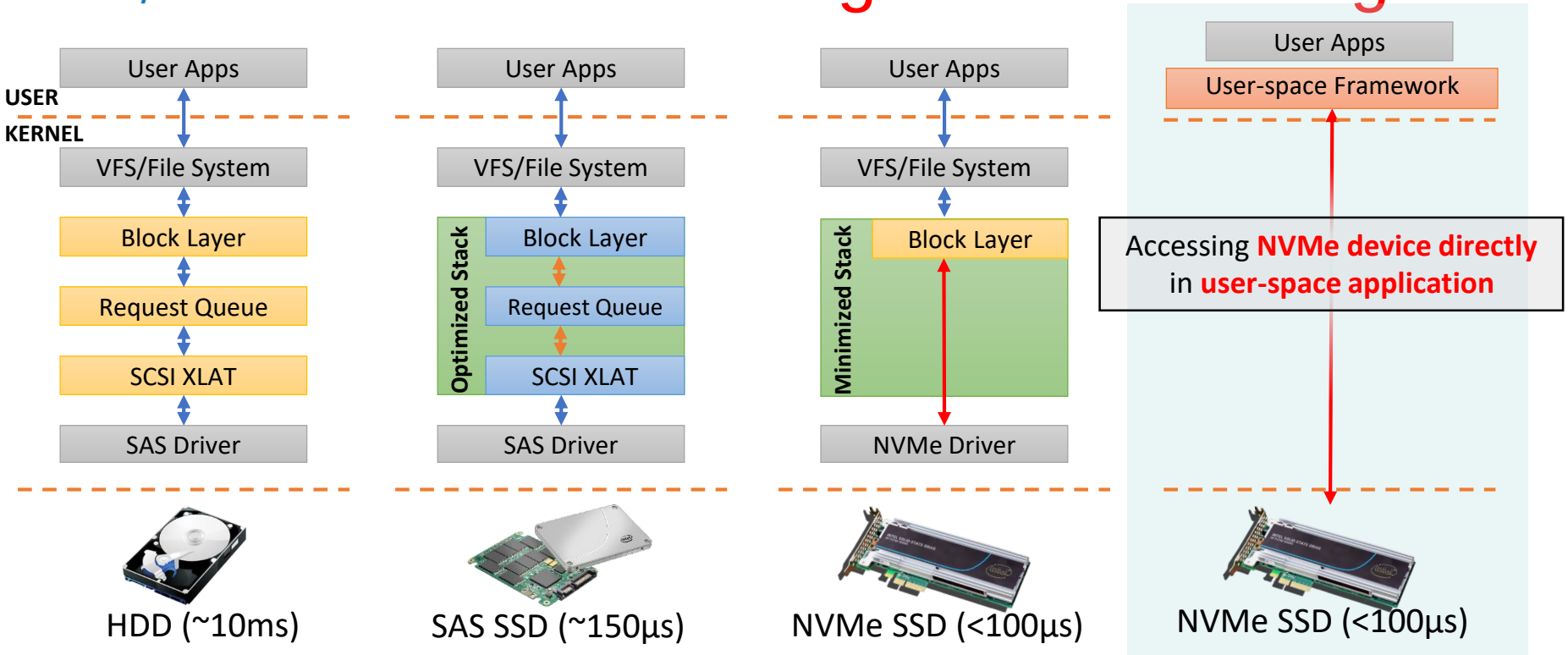
jinsookim@skku.edu

Sungkyunkwan University, South Korea

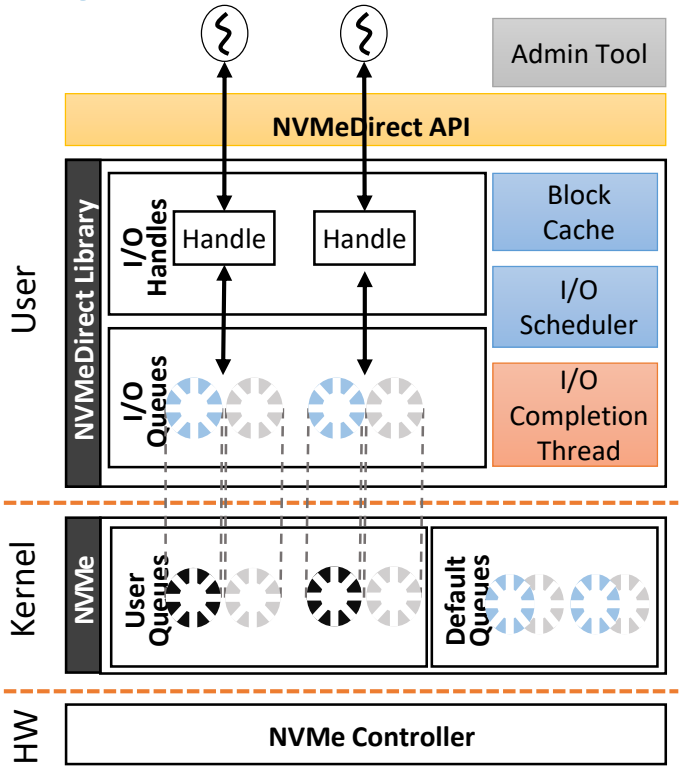
Storage Latency Close to DRAM



I/O Stack Optimization to Take Advantage of Fast Storage



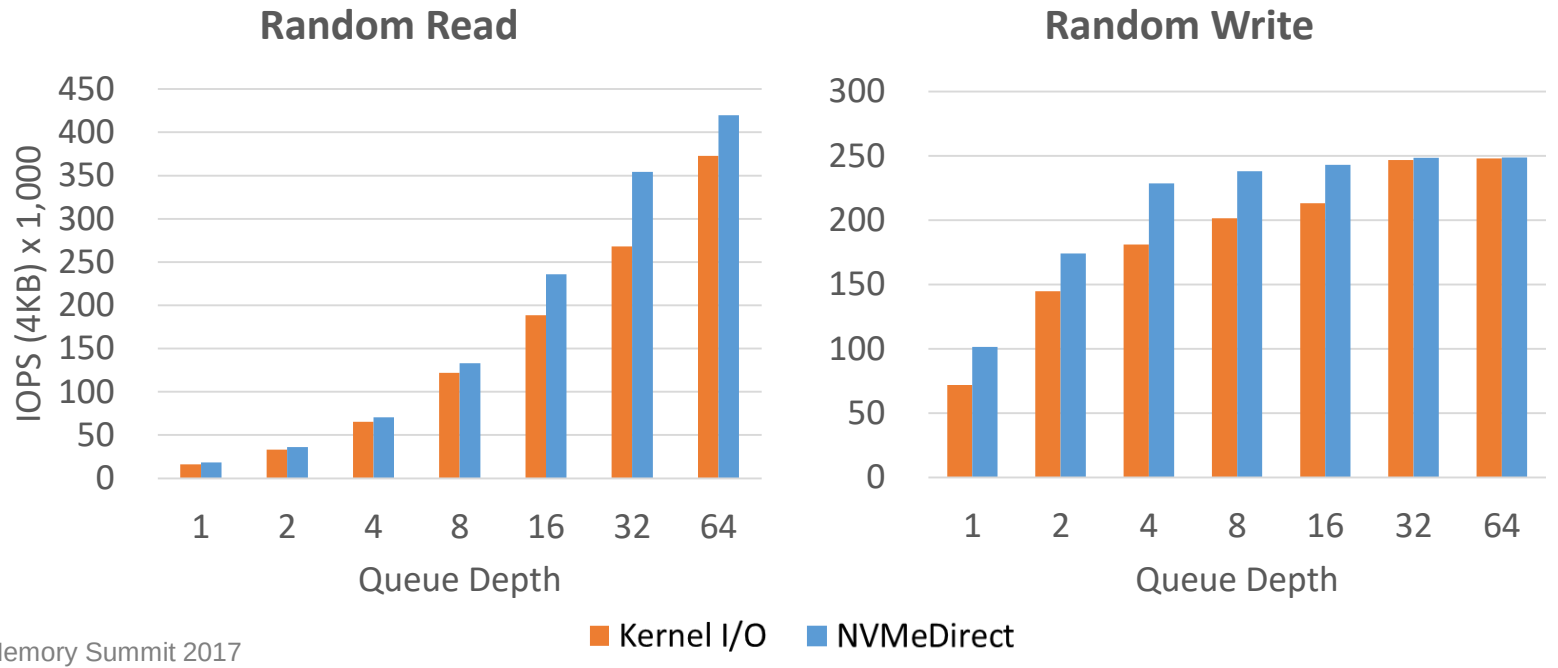
NVMeDirect Framework 1.0



- **User-space I/O Queues**
  - Memory-mapped address space for I/O Queues created in the kernel address space
  - Can co-exist with legacy kernel I/O queues
- **Various I/O policies**
  - Applications can be optimized according to their I/O characteristics
  - Async I/O, Sync I/O, Buffered I/O, Direct I/O
- **I/O Handles**
  - Used to send I/O requests to NVMe I/O Queue(s)
  - Each handle can be configured to use different features: caching, I/O scheduling, I/O completion, stream, etc.

Baseline Performance

- Asynchronous 4KB random I/O performance using FIO



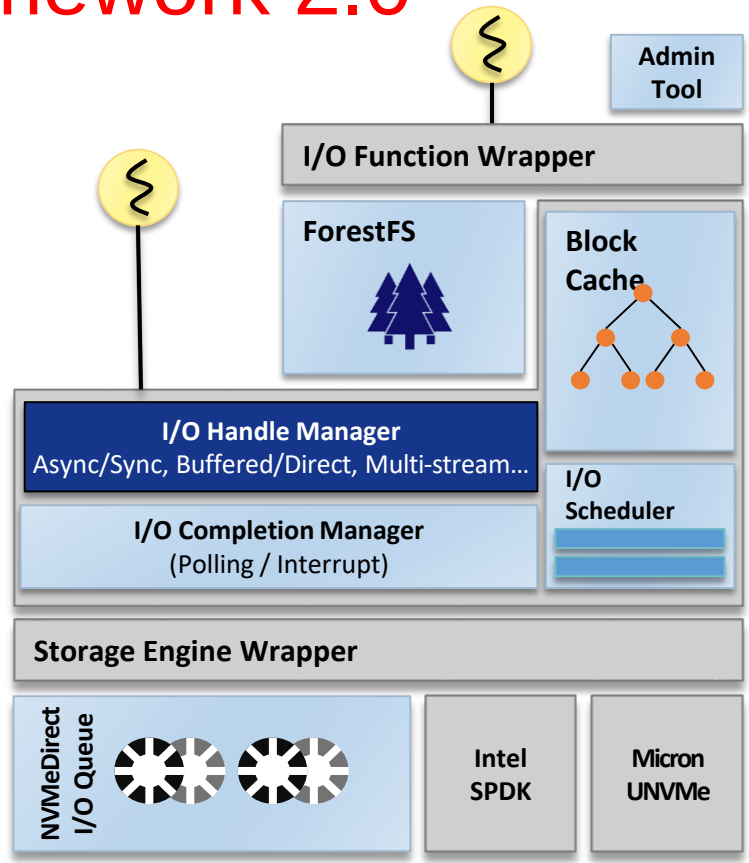
Challenges

- Many applications are developed using the POSIX interface
- Especially, most applications are relying on filesystem and page cache
  - Existing user-space I/O engines support only the raw block device
  - NVMeDirect 1.0 had a limited support for block-level cache
- Application code modification is inevitable
  - Source code should be modified to work without filesystem or page cache
  - Need to modify every single application to use user-space I/O engines

Provide higher-level, rich I/O framework for running existing applications on user-space I/O engines

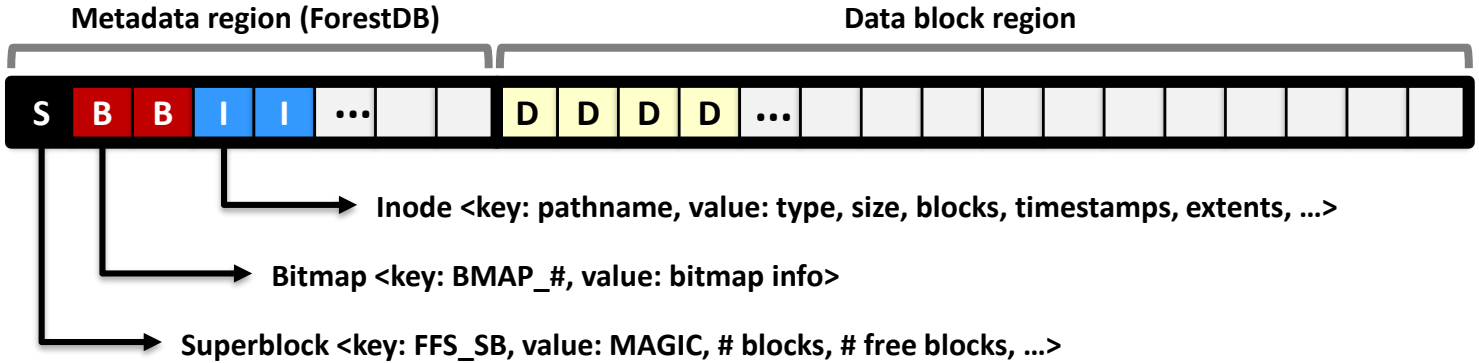
NVMeDirect Framework 2.0

- Application code modification is unnecessary
- I/O Function Wrapper makes applications work on NVMeDirect framework.
- Support for various user-space filesystems
  - Multithread-safe ForestFS
- Filesystem-aware block cache
- Selective I/O completion method
- Support for various user-space storage engines
  - NVMeDirect
  - Intel SPDK
  - Micron UNVMe
- Support for Open Channel SSD (planned)



ForestFS

- A simple user-level file system
- Filesystem metadata managed by ForestDB
  - Utilizing the circular block reuse mode for reducing compaction overhead
- Extent-based data block management
- Plan to support mmap()



NVMeDirect 2.0 vs. SPDK

	NVMeDirect 2.0	SPDK
Developed by	Computer System Labs @ SKKU	Intel
Key components	• Memory mapping • Ioctl()	• DPDK • libpciaccess
	• Access in user-space • **Partition-level management** • **Co-existence with kernel driver**	• Access in user-space • **Device-level management**
Framework features	• **User-level filesystem: ForestFS** - Supports directory structures - Supports SYNC/ASYN APIS - Supports writing to any offset of file • **User-level page cache** • **I/O function wrapper** • **Support for various I/O engines** - **Can be used with SPDK storage engine**	• **User-level filesystem: BlobFS** - Supports only a flat namespace - Supports SYNC API only - Do not support in-place update • **NVMe-oF support**

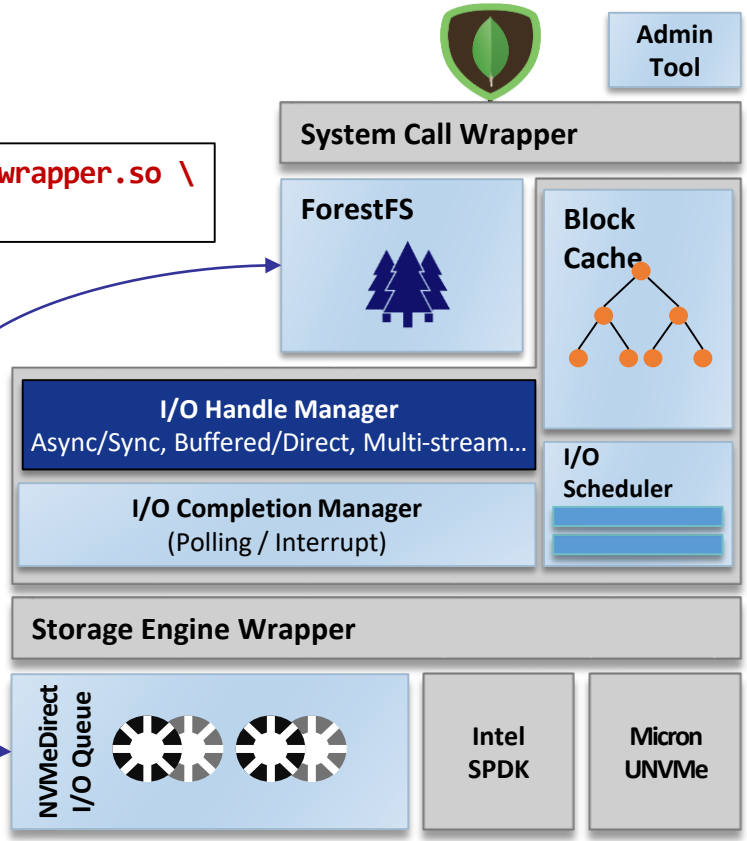
Usage Example

- Running mongodb

```
$ WRAPPER_CONFIG=./default.conf LD_PRELOAD=nvmed_wrapper.so \
  mongod –dbpath ./dbstore
```

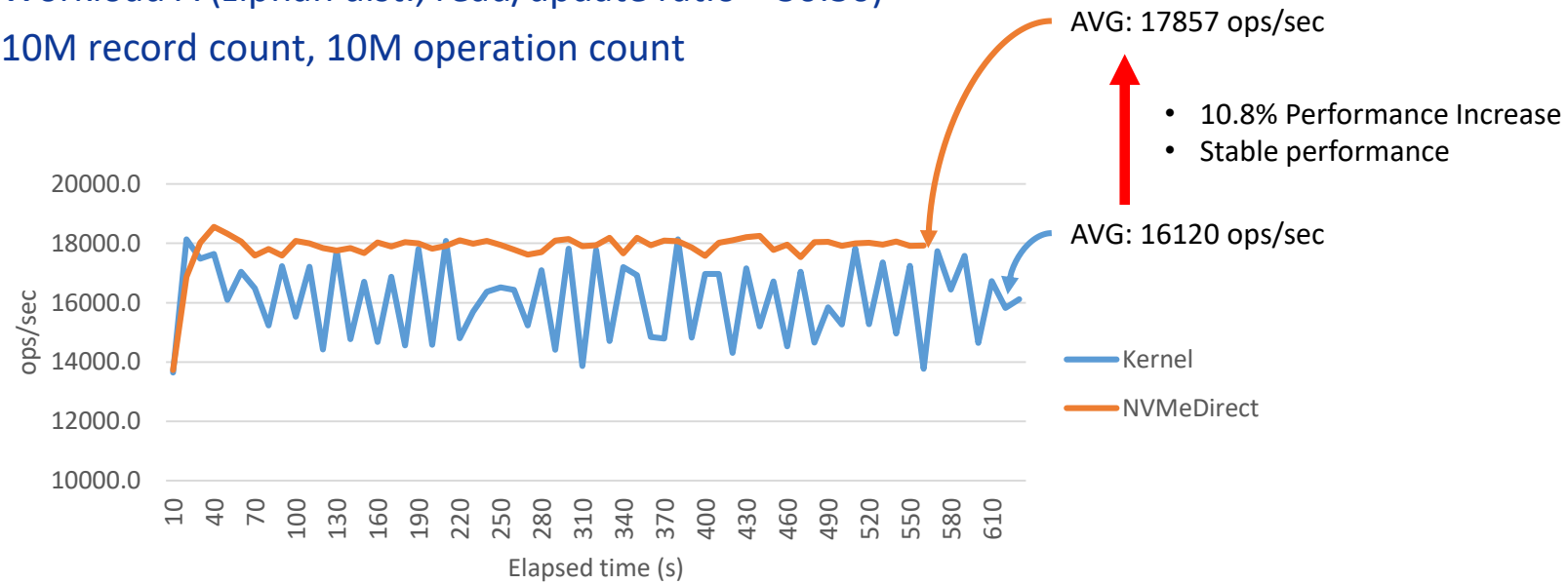
- Config file example : default.conf

```
wrap_dir = ./dbstore

filesystem = lib_ffs.so
ffs_prealloc = 128K
ffs_readahead = none
ffs_format = FALSE

storage_engine = lib_nvmed.so
storage_dev = /dev/nvme0n1
```



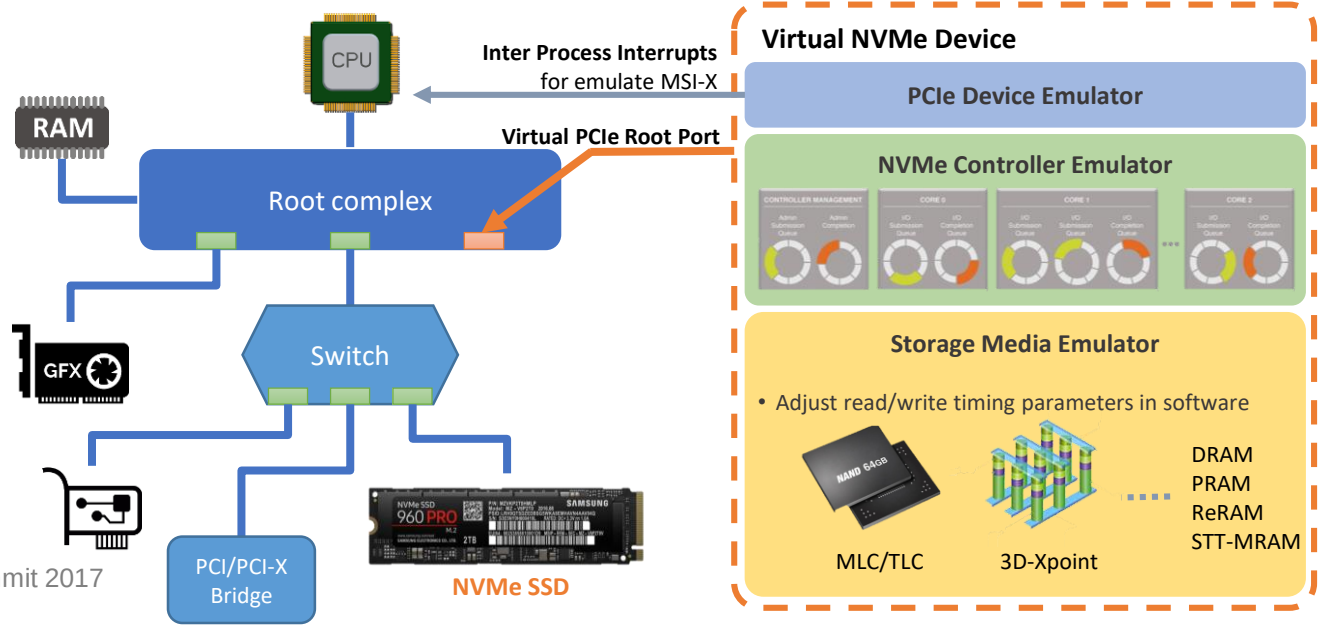
MongoDB Results

- YCSB benchmark on mongoDB
- Workload A (zipfian dist., read/update ratio = 50:50)
- 10M record count, 10M operation count



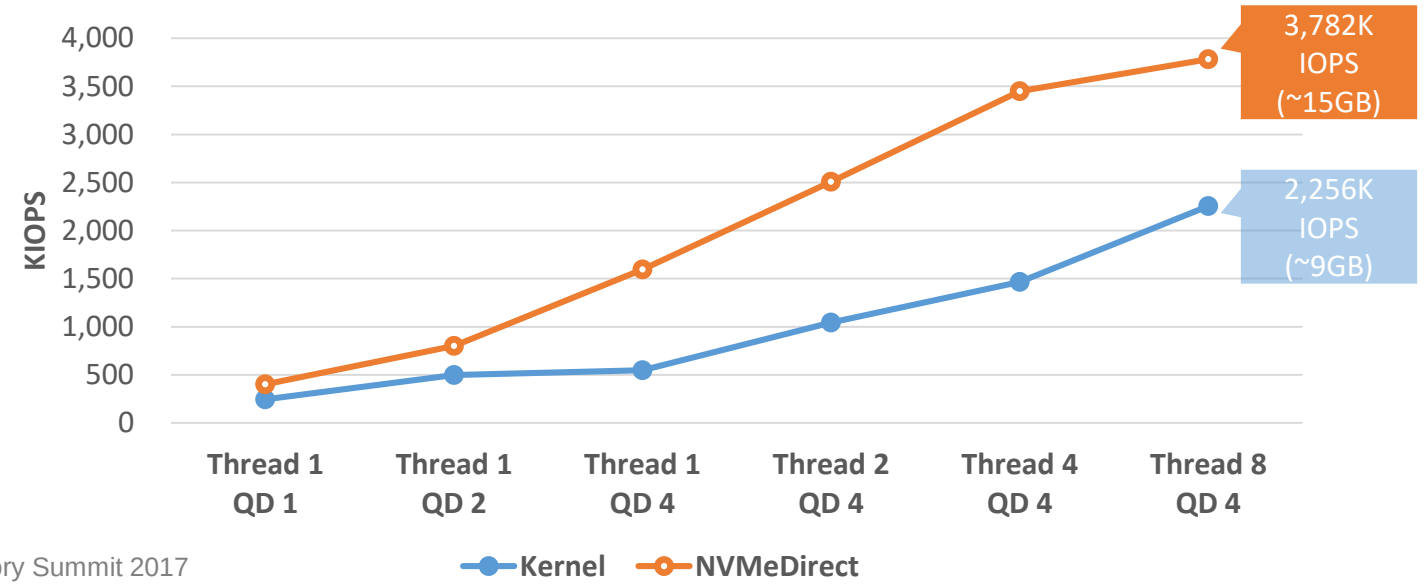
Virtual NVMe Device

- Virtual NVMe emulator for measuring the I/O stack overhead on fast storage device
- Developed as a kernel module and accessible using the existing kernel NVMe driver
- Easy to emulate various storage media (DRAM, NAND Flash, 3D-Xpoint, PRAM, etc)



Preliminary result : Virtual NVMe Device

- Asynchronous 4KB random read on Virtual NVMe (DRAM) using FIO
- Kernel I/O is not scalable on fast storage
- NVMeDirect achieves 3.7M IOPS, while existing kernel I/O achieves 2.2M IOPS (8 Thread, QD 4)



Conclusion

- We propose an enhanced version of user-level I/O framework, called NVMeDirect 2.0
  - Provides higher-level, rich I/O framework
  - Supports various user-space filesystems: ForestFS, etc.
  - Supports various user-space storage engines: NVMeDirect, SPDK, etc.
- NVMeDirect Framework 2.0 does not require application code modification
  - NVMeDirect Framework 2.0 can be used to port existing applications easily
- NVMeDirect Framework 2.0 makes applications exploit the fast NVMe storage devices
  - Performance improvement: 10.8% on MongoDB (without any code change)
- Will be available as open-source at https://github.com/nvmedirect
  - Any contributions are welcome to make it richer!

Reference

[1] H.-J. Kim, Y.-S. Lee, and J.-S. Kim, "NVMeDirect: A User-space I/O Framework for Application-specific Optimization on NVMe SSDs," Proc. HotStorage, 2016.

[2] J. Axboe, Flexible IO tester, http://git.kernel.dk/?p=fio.git;a=summary

[3] J. S. Ahn, C. Seo, R. Mayuram, R. Yaseen, J.-S. Kim, and S. Maeng “ForestDB: A Fast Key-Value Storage System for Variable-Length String Keys,” IEEE Transactions on Computers, vol. 65, no. 3, pp. 902-915. 2016.

[4] MongoDB, https://www.mongodb.com/

[5] Intel Corp., SPDK: Storage performance development kit. https://01.org/spdk

[6] Micron Technology, Inc., UNVMe, https://github.com/MicronSSD/unvme

[7] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking cloud serving systems with YCSB,” Proc. SOCC, pp. 143–154, 2010.

Thank you

hyeongjun.kim@csl.skku.edu

NVMeDirect Framework 2.0
Available as open-source at
https://github.com/nvmedirect