A Systematic Configuration Space Exploration of the Linux Kyber I/O Scheduler

Zebin Ren, Krijn Doekemeijer, and Animesh Trivedi

VU Amsterdam

@Large Research
Massivizing Computer Systems
https://atlarge-research.com/

GRAPH MASSIVIZER

FUTURE 6G NETWORK SERVICES

CLOUD STARS

NWO

VU VRIJE UNIVERSITEIT AMSTERDAM

Background

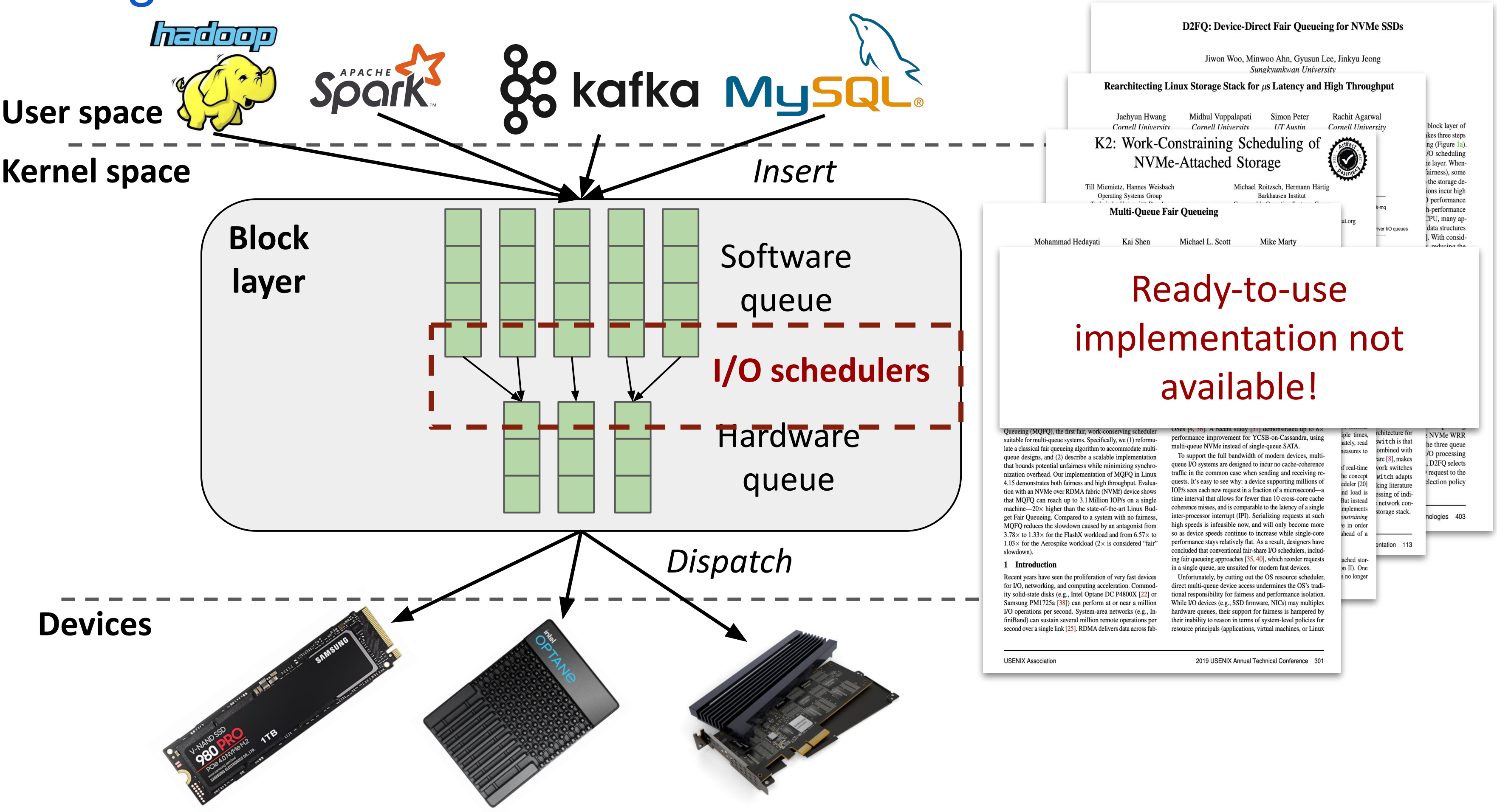
QoS guarantees

- Latency
- Throughput

Up to millions of IOPS

< 10 µs latency

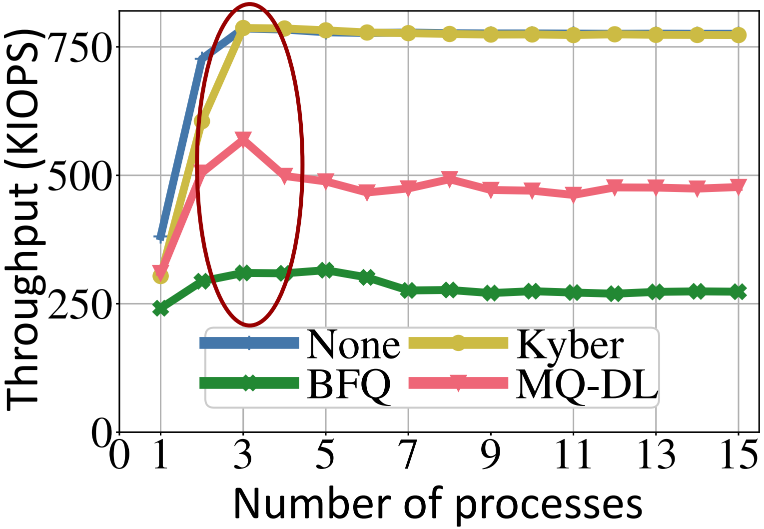
Background: I/O Schedulers



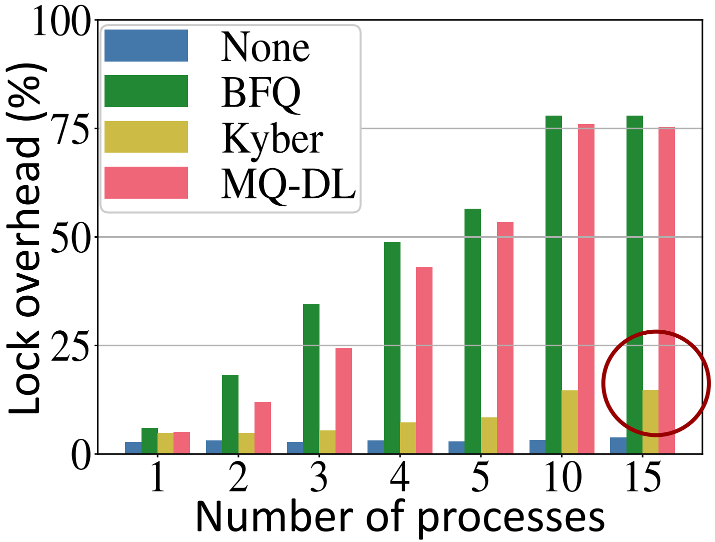
Why Kyber?

In our previous paper: *BFQ, Multiqueue-Deadline, or Kyber? Performance Characterization of Linux Storage Schedulers in the NVMe Era (ICPE'24)*

Kyber has

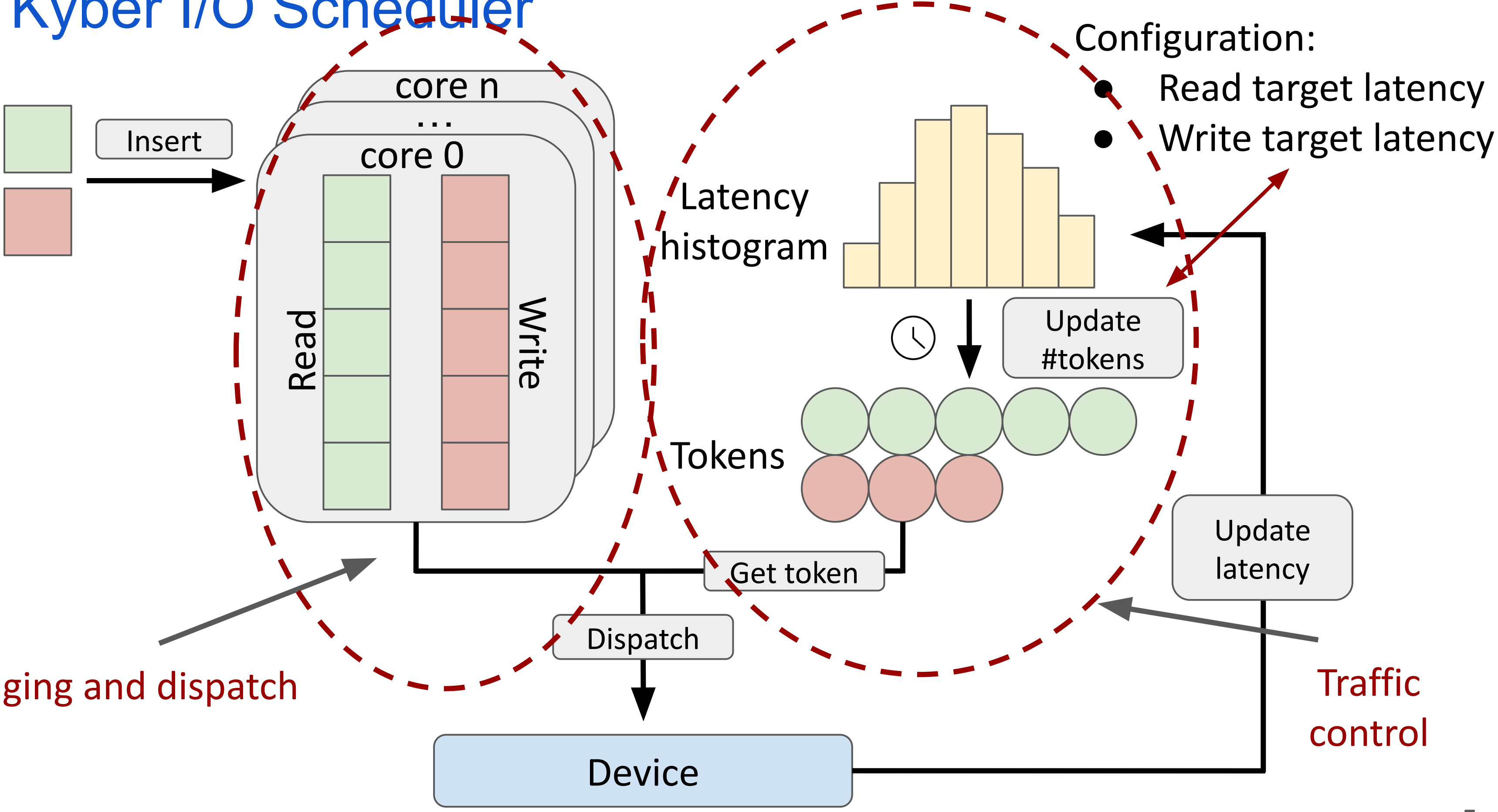


Less overhead, better scalability.

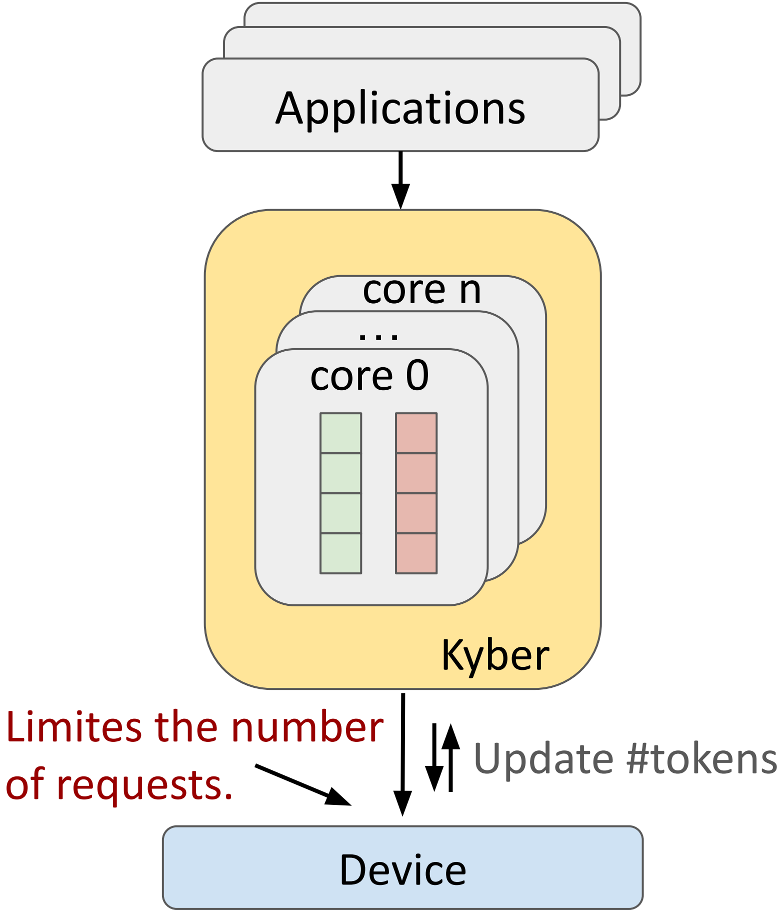


Less lock contention.

The Kyber I/O Scheduler

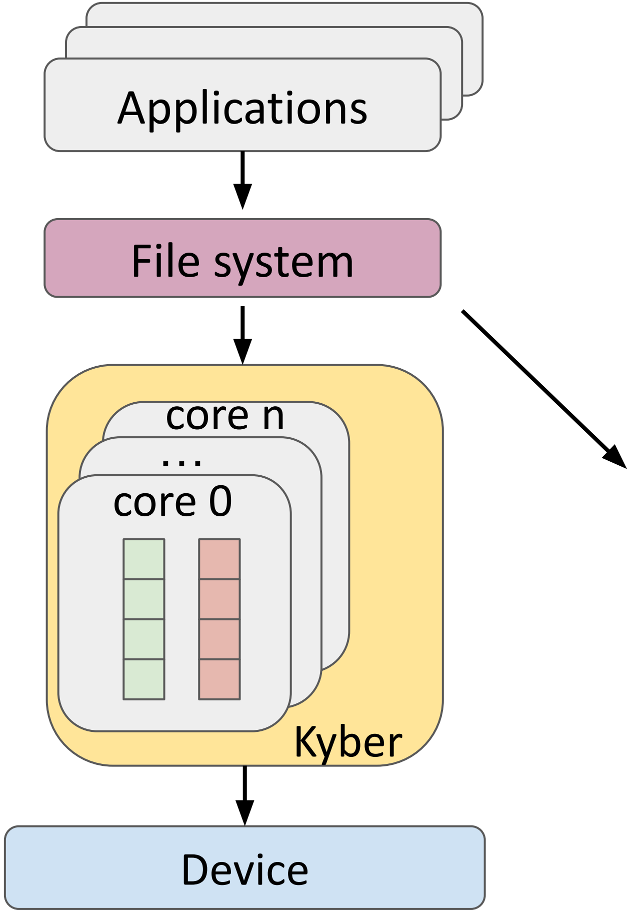


Research Questions



RQ1: Effects of the configurations on performance?

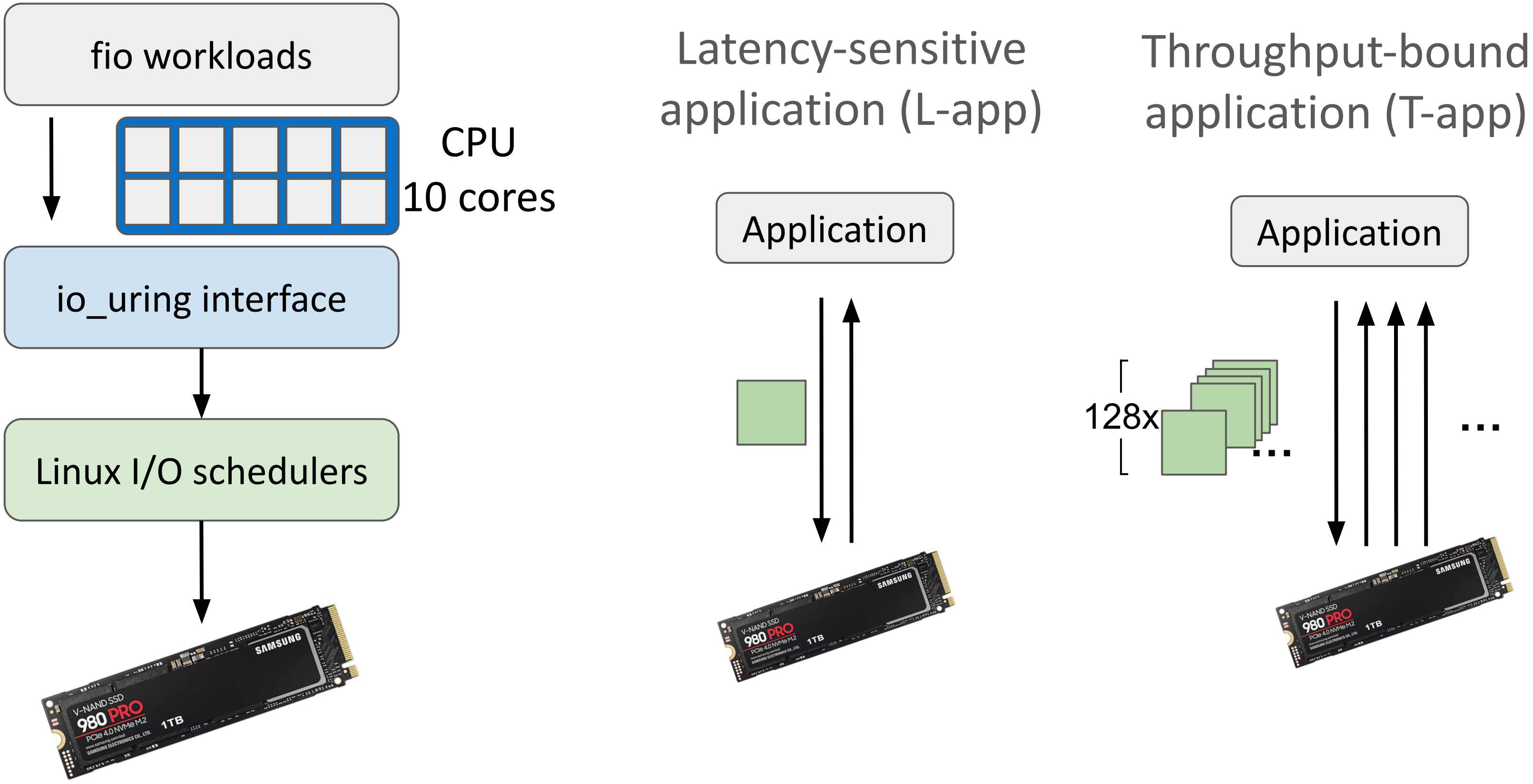
Research Questions



RQ1: Effects of the configurations on performance?

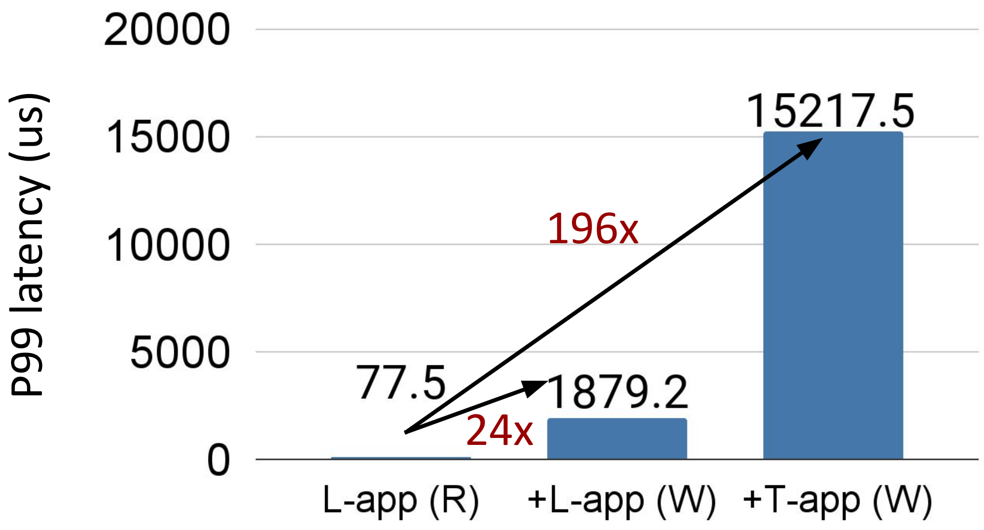
RQ2: Effect of the configurations with different file systems?

Setup



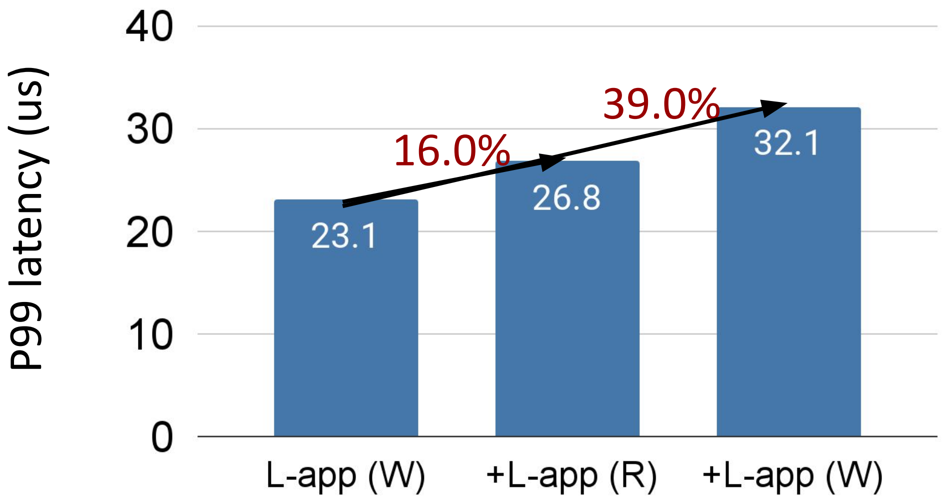
fio workloads
CPU
10 cores
io_uring interface
Linux I/O schedulers
Latency-sensitive application (L-app)
Application
Throughput-bound application (T-app)
Application
128x
...
...

SSD Performance: Interference



Write has significant effect on read performance.

SSD Performance: Interference



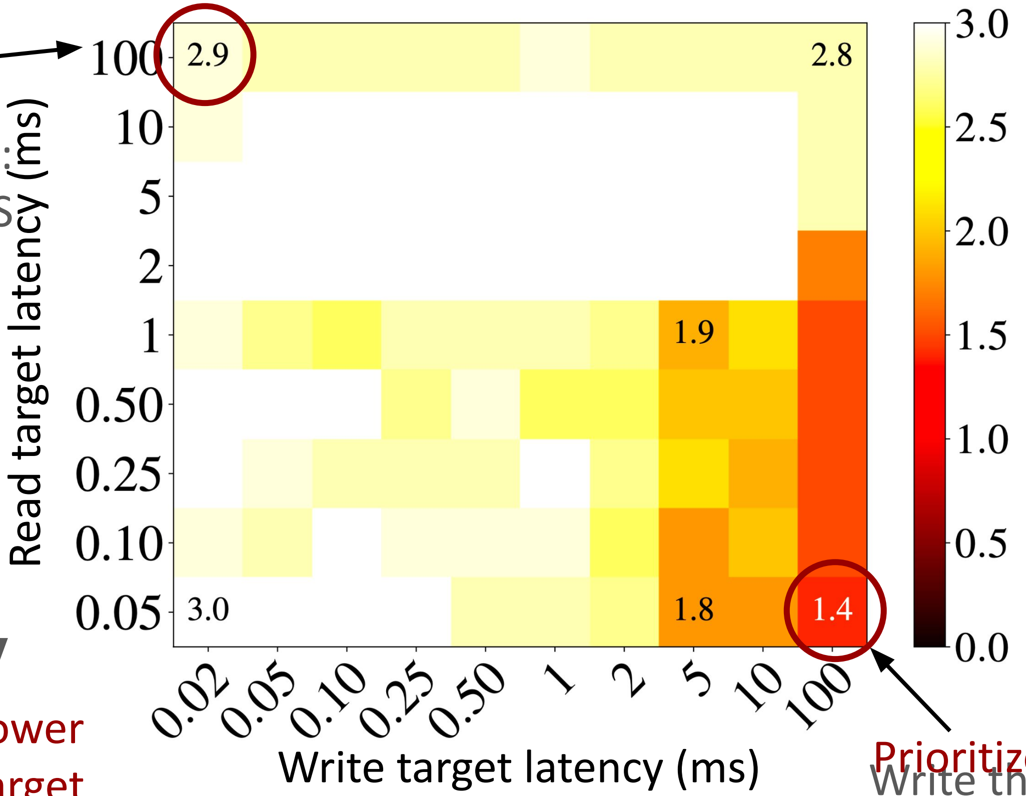
Read has less significant effect on write.

RQ1:
Effects of the configurations on performance

Read L-app + Write T-app (Read Latency)

Prioritize write

Write throughput: 131.3 KIOPS



Lower Target Latency

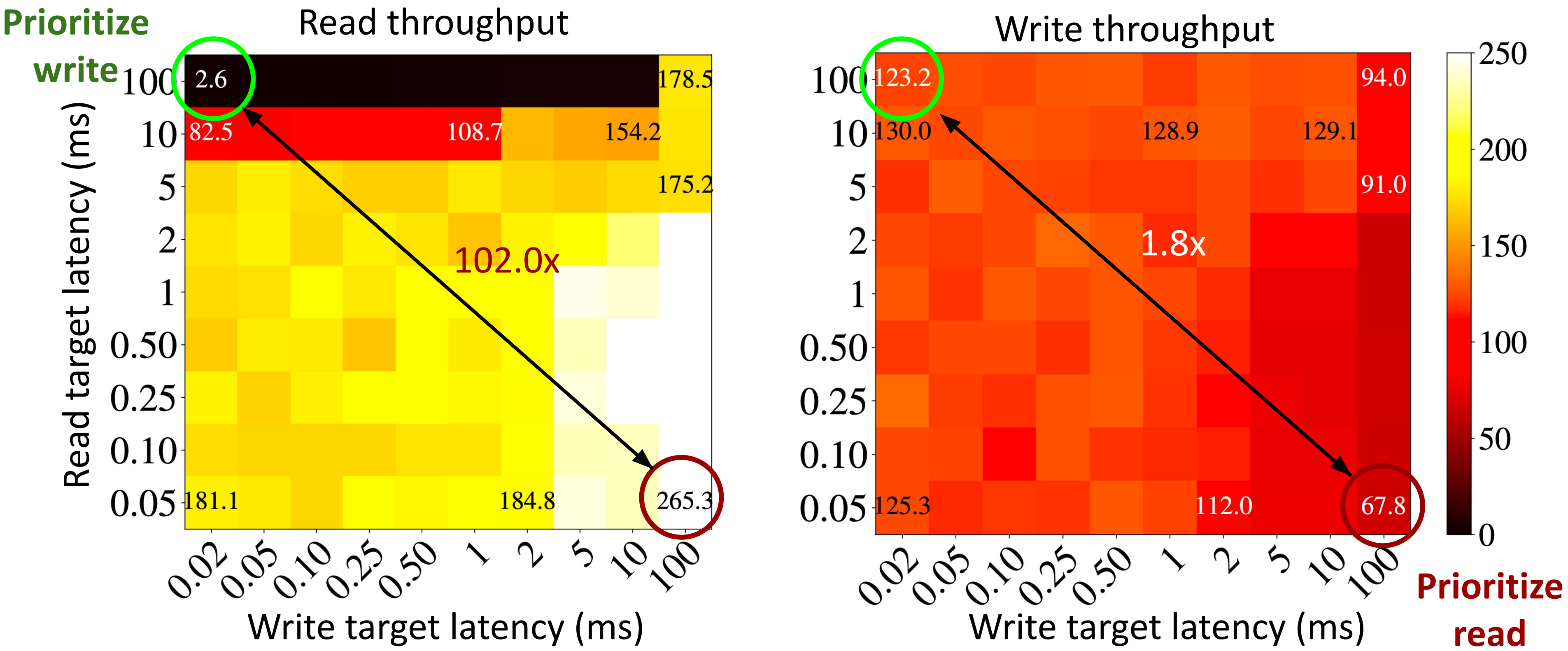
Prioritize read

Write throughput: 74.6 KIOPS

Prioritize read ⟶ lower read latency.

Cost ⟶ lower write throughput.

Read T-app + Write T-app (Read Throughput)

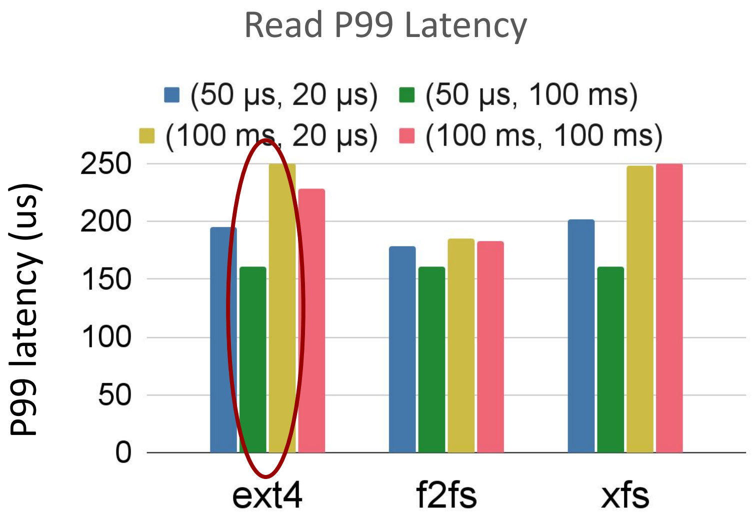


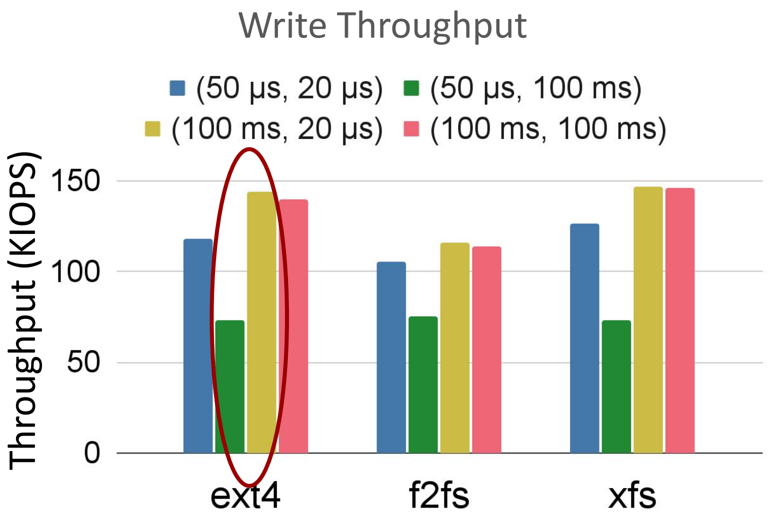
Lower target latency ⟶ higher throughput.

Read and write have different sensitivity to Kyber configurations.

RQ2:
Effect of the configurations with different file systems

Read L-app + Write T-app, with File System

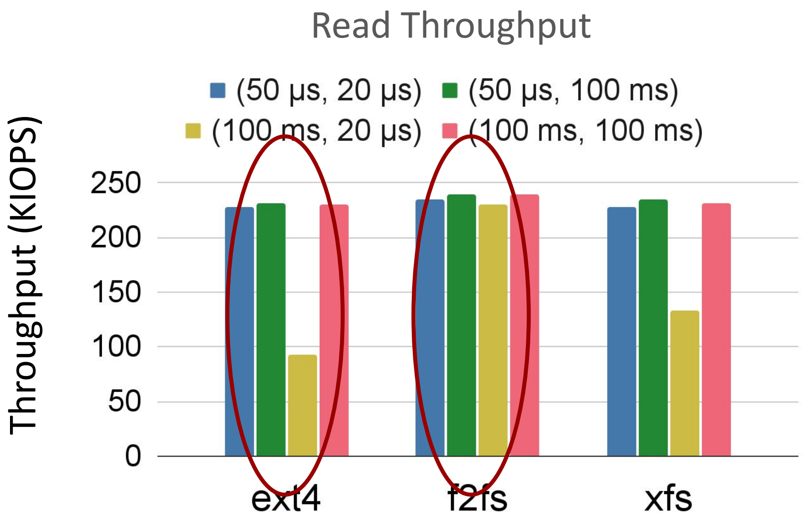


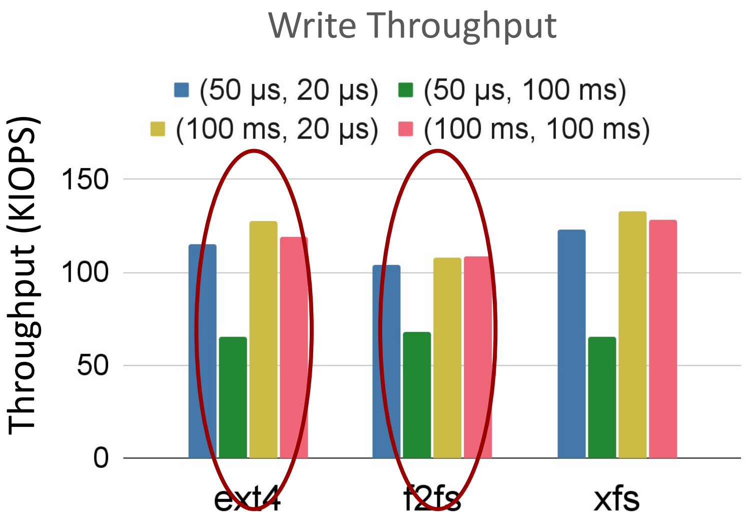


All the three file systems can provide lower read latency than the block layer.

Prioritize read → low read latency at the cost of write throughput.

Read L-app + Write T-app, with File System





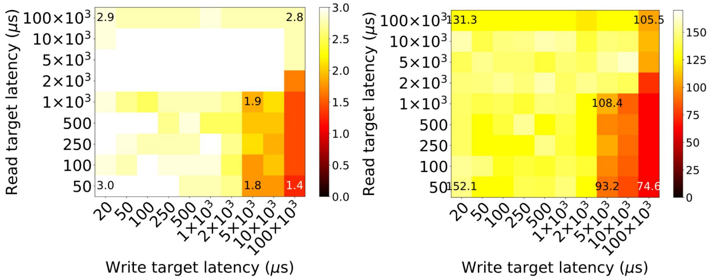
ext4 and xfs: prioritizing read/write →high read/write throughput.

f2fs: prioritizing read → slightly higher read throughput but much lower write throughput.

Conclusion

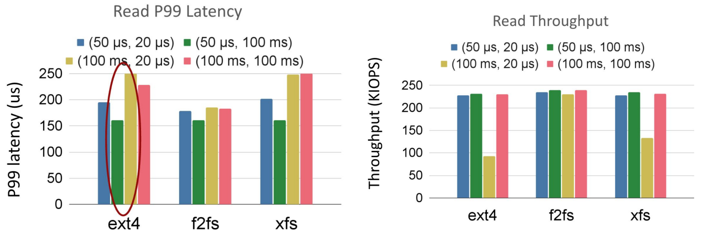
1. What are effects of the Kyber configurations on performance?

- Relative lower target latency ⟶ lower latency and higher throughput.
- Read performance is more sensitive than write.



2. What are the effects of the configurations with different file systems?

- ext4 and xfs ⟶ similar to using the block layer directory.
- f2fs ⟶ prioritizing reads lead to comparable read throughput than other configurations.



Take-home Messages

1. Kyber’s configurations, read/write target latency, can be **treated as priority**.

2. How much that Kyber’s configuration affect the performance depends the ***sensitivity of the requests on concurrency***.

3. Kyber’s configuration has **different effect** on the I/O performance with different file systems.

GitHub

Paper: https://atlarge-research.com/pdfs/hotcloudperf24-kyber.pdf
Source code: https://github.com/ZebinRen/hotcloudperf24-kyber-artifact-public

PDF

Thank you!
Questions?

Paper: https://atlarge-research.com/pdfs/hotcloudperf24-kyber.pdf
Source code: https://github.com/ZebinRen/hotcloudperf24-kyber-artifact-public

Resources

Images used:

https://www.samsung.com/nl/memory-storage/nvme-ssd/980-pro-pcle-4-0-nvme-m-2-ssd-1tb-mz-v8p1t0bw/
https://www.intel.com/content/www/us/en/products/details/memory-storage/data-center-ssds/optane-dc-ssd-series.html
https://www.anandtech.com/show/12376/samsung-launches-zssd-sz985-up-to-800gb-of-znand

References

[1] Till Miemietz, Hannes Weisbach, Michael Roitzsch, Hermann Härtig: K2: Work-Constraining Scheduling of NVMe-Attached Storage. RTSS 2019: 56-68
[2] Mohammad Hedayati, Kai Shen, Michael L. Scott, Mike Marty: Multi-Queue Fair Queuing. USENIX Annual Technical Conference 2019: 301-314 2018
[3] Jaehyun Hwang, Midhul Vuppalapati, Simon Peter, Rachit Agarwal: Rearchitecting Linux Storage Stack for µs Latency and High Throughput. OSDI 2021: 113-128
[4] Jiwon Woo, Minwoo Ahn, Gyusun Lee, Jinkyu Jeong: D2FQ: Device-Direct Fair Queueing for NVMe SSDs. FAST 2021: 403-415

Further Reading

Linux I/O schedulers

1. BFQ (Budget Fair Queueing) https://www.kernel.org/doc/html/latest/block/bfq-iosched.html
2. Two new block I/O schedulers for 4.12 https://lwn.net/Articles/720675/
3. Deadline IO scheduler tunables https://docs.kernel.org/block/deadline-iosched.html#:~:text=The%20goal%20of%20the%20deadline,value%20in%20units%20of%20milliseconds.
4. BFQ I/O Scheduler For Linux Sees Big Scalability Improvement https://www.phoronix.com/news/BFQ-IO-Better-Scalability
5. MQ-Deadline Scheduler Optimized For Much Better Scalability

New I/O schedulers

1. Myoungsoo Jung, Wonil Choi, Shekhar Srikantaiah, Joonhyuk Yoo, and Mahmut T. Kandemir. HIOS: A Host Interface I/O Scheduler for Solid State Disks. ISCA 2014.
2. Mingyang Wang and Yiming Hu. An I/O Scheduler Based on Fine-Grained Access Patterns to Improve SSD Performance and Lifespan. In Symposium on Applied Computing, SAC 2014.
3. Hui Lu, Brendan Saltaformaggio, Ramana Rao Kompella, and Dongyan Xu. vFair: Latency-Aware Fair Storage Scheduling via per-IO Cost-Based Differentiation. SoCC 2015.
4. Jiayang Guo, Yiming Hu, Bo Mao, and Suzhen Wu. Parallelism and Garbage Collection Aware I/O Scheduler with Improved SSD Performance. IPDPS 2017.
5. Minhoon Yi, Minho Lee, and Young Ik Eom. 2017. CFFQ: I/O Scheduler for Providing Fairness and High Performance in SSD Devices. IMCOM 2017.
6. Mohammad Hedayati, Kai Shen, Michael L. Scott, and Mike Marty. Multi- Queue Fair Queuing. In 2019 USENIX Annual Technical Conference, USENIX ATC 2019.
7. Till Miemietz, Hannes Weisbach, Michael Roitzsch, and Hermann Härtig. K2: Work-Constraining Scheduling of NVMe-Attached Storage. RTSS 2019.
8. Jaehyun Hwang, Midhul Vuppalapati, Simon Peter, and Rachit Agarwal. Rearchitecting Linux Storage Stack for μs Latency and High Throughput. OSDI 2021.
9. Jiwon Woo, Minwoo Ahn, Gyusun Lee, and Jinkyu Jeong. D2FQ: Device- Direct Fair Queueing for NVMe SSDs. FAST 2021.
10. Jieun Kim, Dohyun Kim, and Youjip Won Fair I/O Scheduler for Alleviating Read/Write Interference by Forced Unit Access in Flash Memory. HotStorage 2022.
11. Caeden Whitaker, Sidharth Sundar, Bryan Harris, and Nihat Altiparmak. Do We Still Need I/O Schedulers for Low-Latency Disks?. HotStorage 2023.

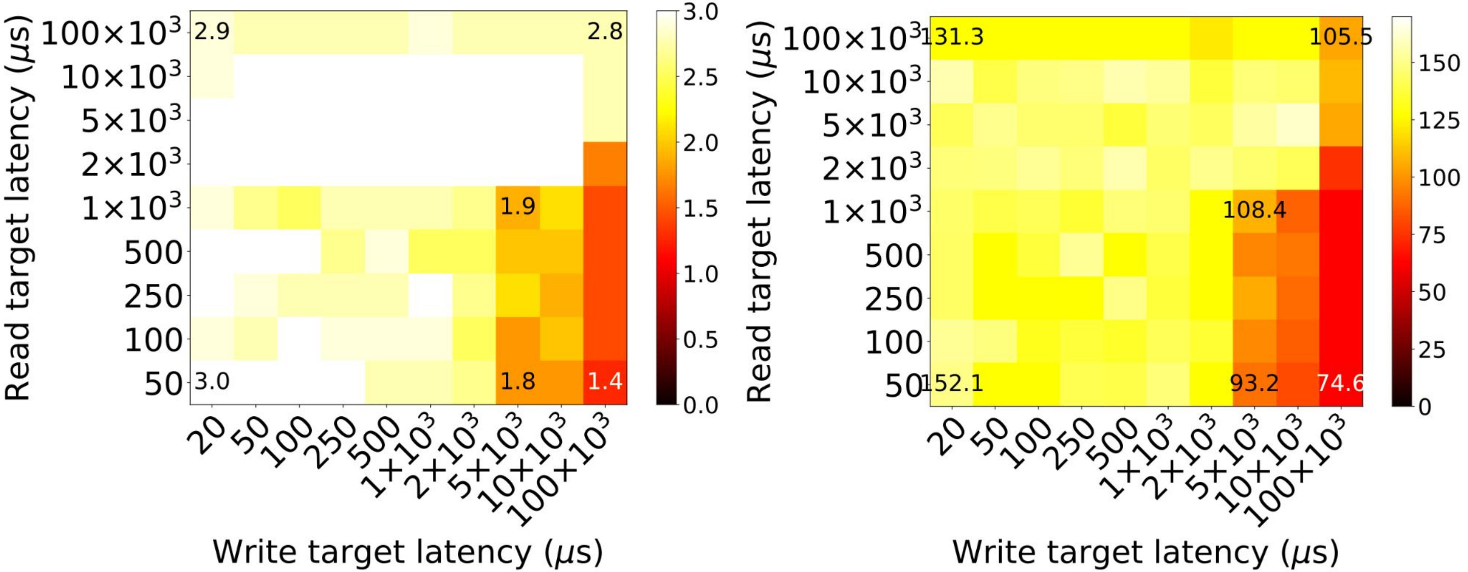
Backup Slides

Baseline Performance

#	Workload	R TP (in KIOPS)	W TP (in KIOPS)	R P99 Lat (in μs)	W P99 Lat (in μs)
1	R1	17.0	-	77.5	-
2	R256	364.3	-	793.8	-
3	W1	-	62.3	-	23.1
4	W256	-	70.0	-	15,794.2
5	R1–W1	4.0	65.0	1,879.2	26.8
6	R1–W256	0.3	68.9	15,217.5	15,558.2
7	R256–W1	302.6	61.5	3,044.1	32.1
8	R256–W256	83.2	93.1	15,283.0	15,938.4

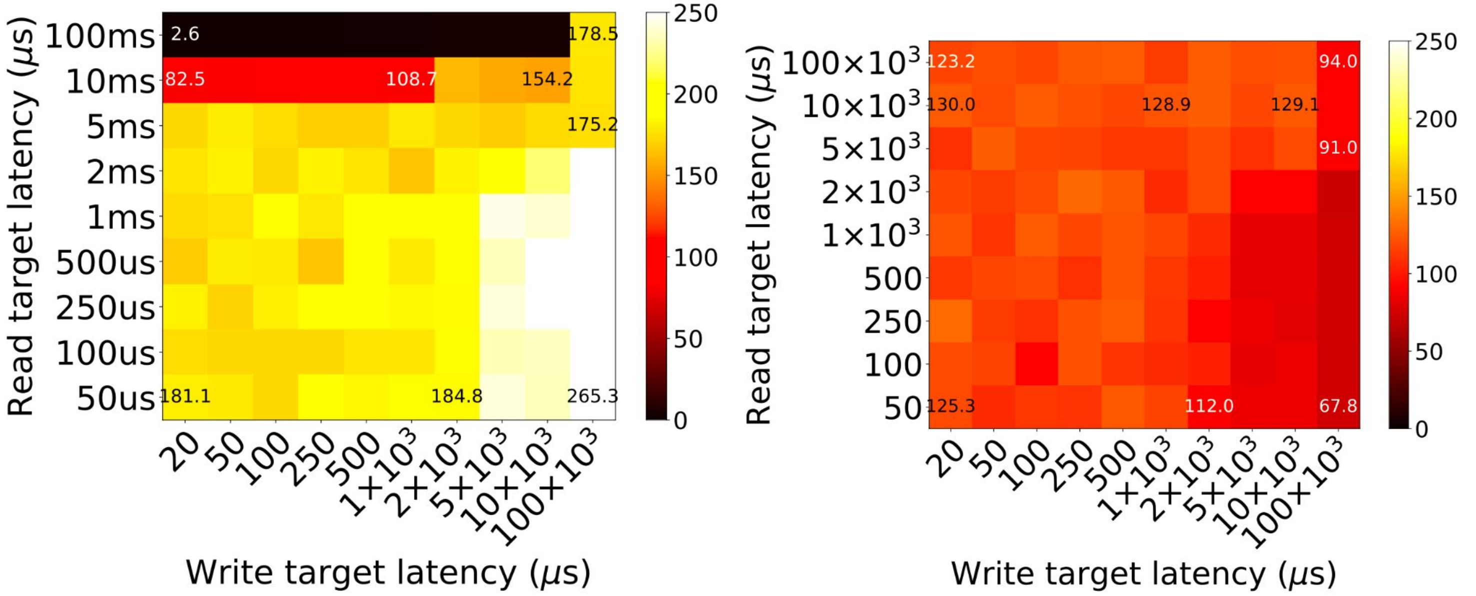
Baseline performance of Samsung 980 PRO SSD with the None scheduler.

Block Interface: L-app (Read) + T-app (Write)



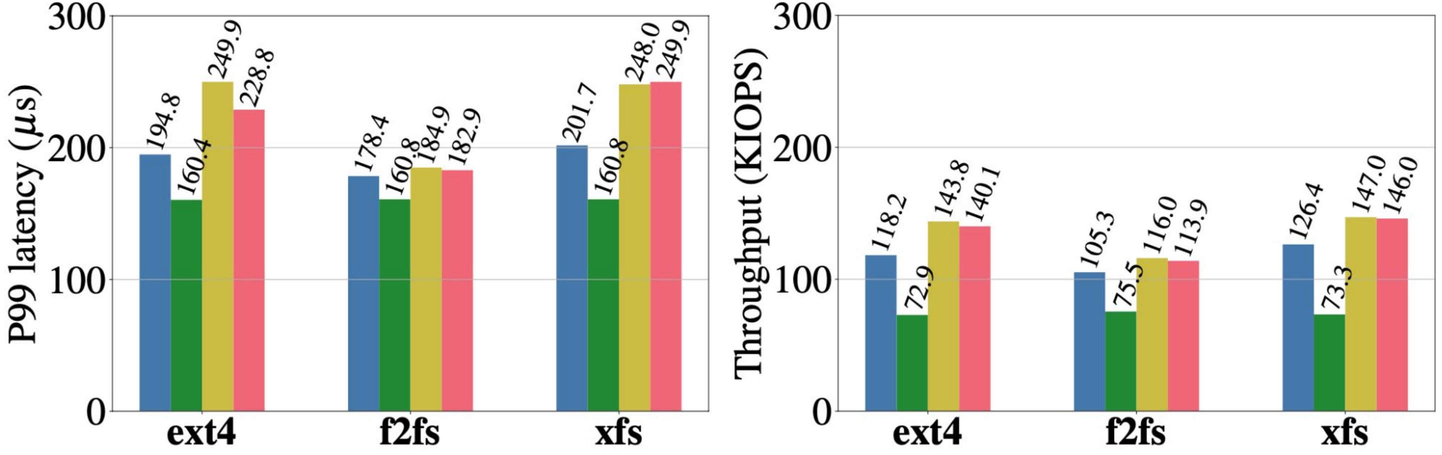
Performance of the combination of L-app (read) and T-app (write) with different Kyber configurations.

Block Interface: T-app (Read) + T-app (Write)



Performance of the combination of T-app (read) and T-app (write) with different Kyber configurations.

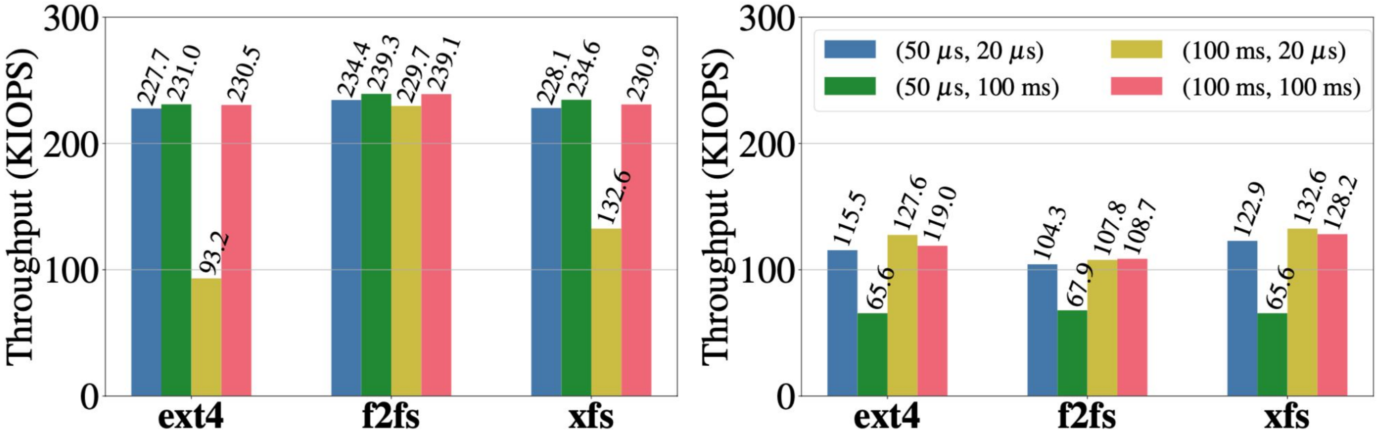
FS: R-app (Read) + T-app (Write)



(a) R latency in R1–W256 **(b) W throughput in R1–W256**

Performance of the combination of R1–W256 with different Kyber configurations with file systems.

T-app (Read) + T-app (Write)



(c) R throughput in R256–W256 **(d) W throughput in R256–W256**

Performance of the combination of R256–W256 with different Kyber configurations with file systems.

Unused Slides

SSD Performance: Asymmetric R/W Performance

Workload	Read Throughput (KIOPS)	Read Latency (us)
L-app (R)	17.0	77.5
T-app (R)	364.3	793.8
L-app (R) + L-app (W)	4.0	1,879.2
L-app (R) + T-app (W)	0.3	15,217,5
T-app (R) + T-app (W)	83.2	15,283.0