A Parametric I/O Model for Modern Storage Devices

Tarikul Islam Papon
papon@bu.edu

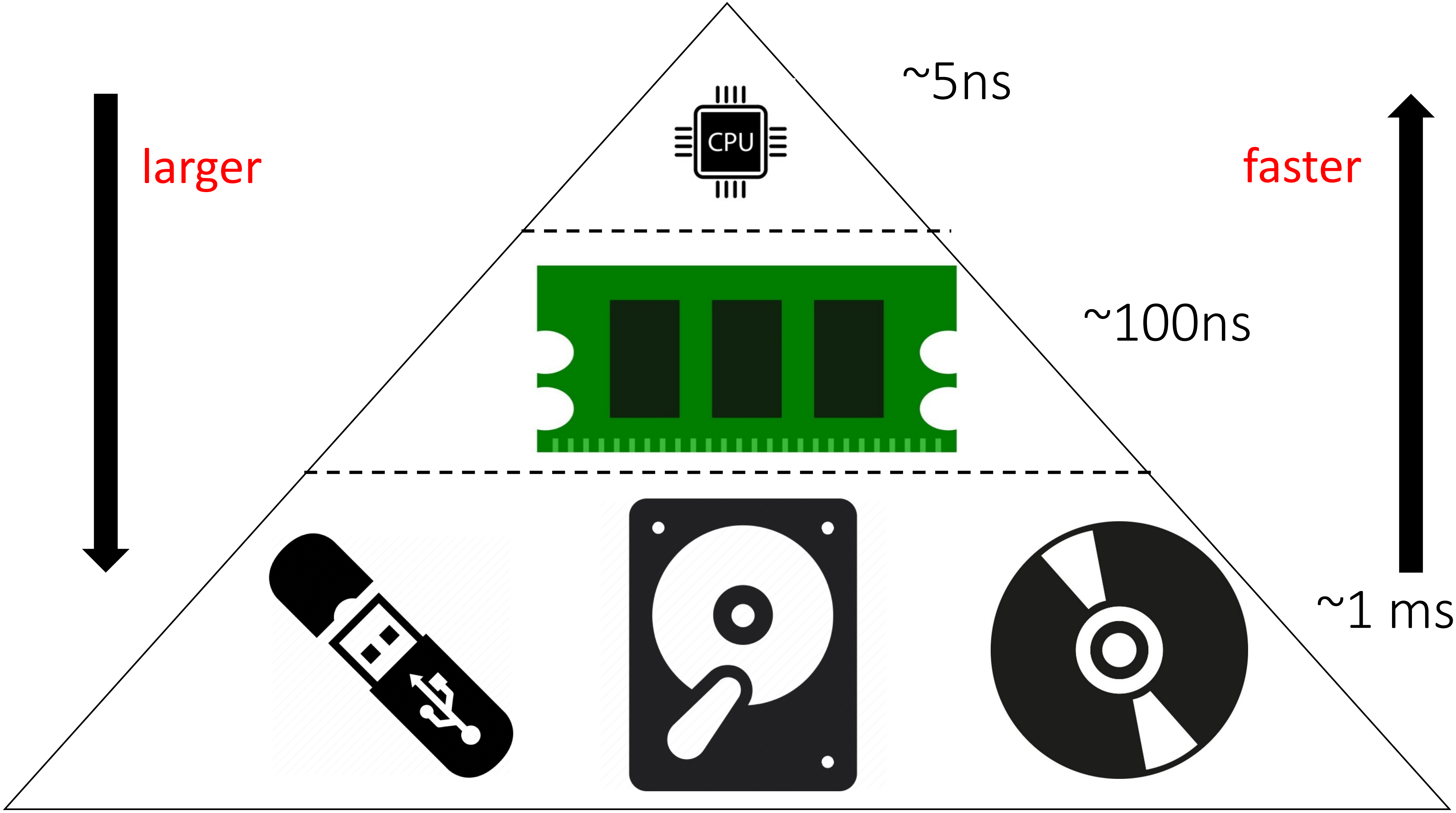
Manos Athanassoulis
mathan@bu.edu

presentation at DaMoN 2021

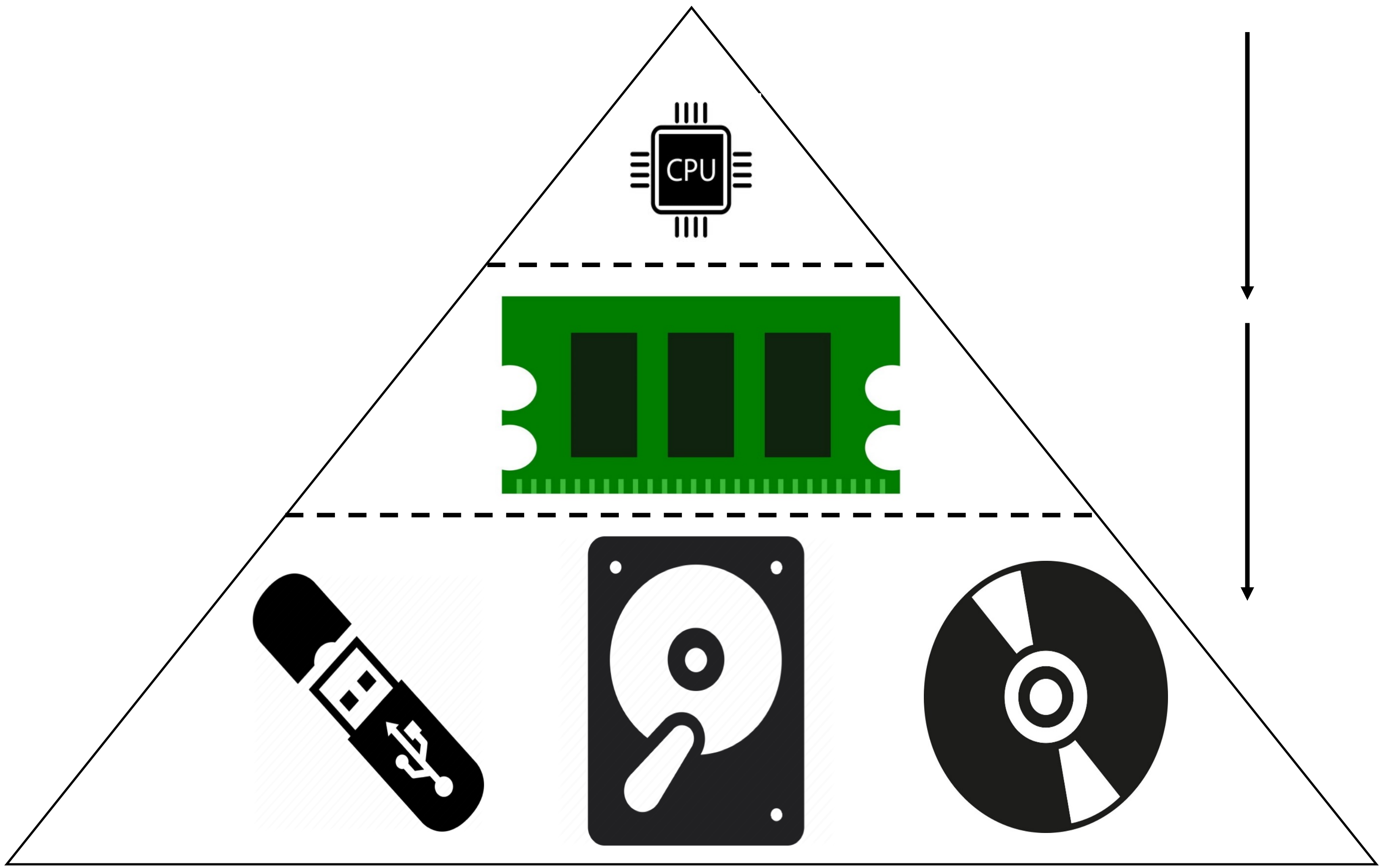
Modeling Performance

"Algorithm/Data Structure **X** has $O(f(N))$ performance, where N is the number of data pages on disk"

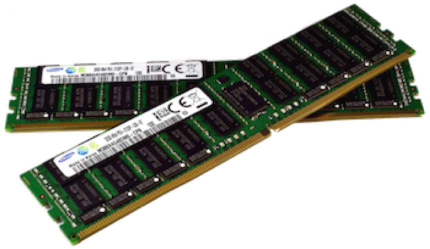
… is probably one of the most commonly read phrases in SIGMOD papers.


larger
CPU
~5ns
~100ns
~1 ms
faster

Memory Hierarchy

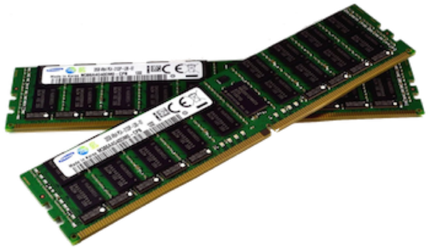


Traditional I/O Model



Small, fast main memory
(size M)

Traditional I/O Model

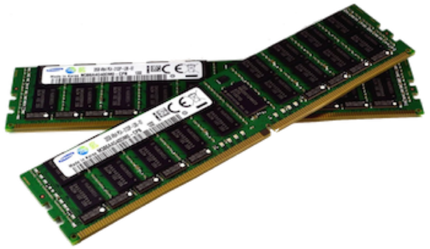


Small, fast main memory
(size M)

Large, slow external memory

Traditional I/O Model

One I/O at a time

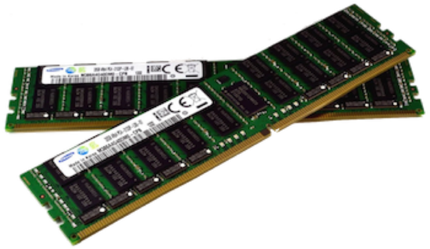


Small, fast main memory
(size M)

Large, slow external memory

Traditional I/O Model

0 access cost



Small, fast main memory
(size M)

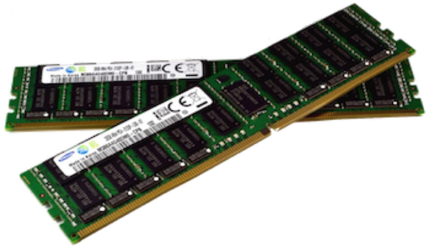
Large, slow external memory

Traditional I/O Model

Transfer cost

1 unit

0 access cost



Small, fast main memory

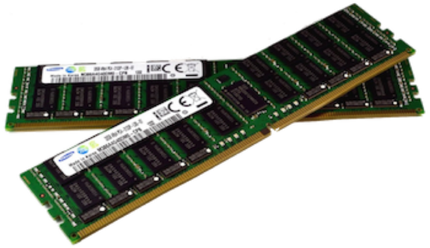
(size M)

Large, slow external memory

Traditional I/O Model

total cost $\simeq$ total # reads/writes to disk

0 access cost



Small, fast main memory
(size M)

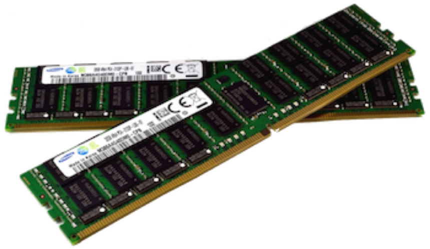
Transfer cost
1 unit

Large, slow external memory

Traditional I/O Model

Two (outdated) assumptions

- **Symmetric** cost for **Read & Write** to disk
- **One I/O** at a time



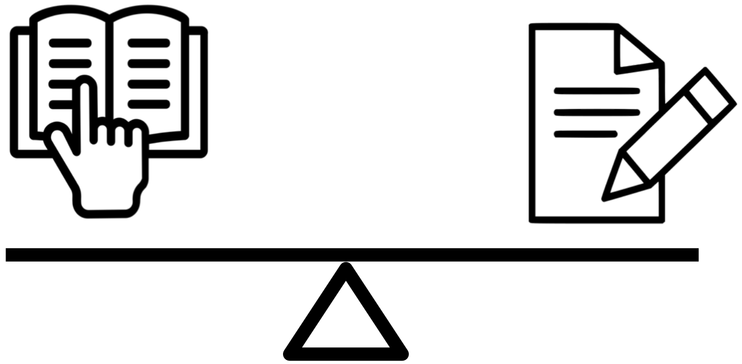
Small, fast main memory
(size M)

Large, slow external memory

Hard Disk Drives

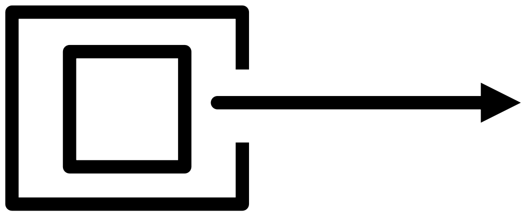
Hard Disk Drives

Two assumptions of Traditional I/O Model



Symmetric cost for **Read & Write** to disk

✓



One I/O at a time

✓

HDD Stopped Evolving

- Generally, the slowest component
- Slowest increase in performance

Device	Size	Seq B/W	Time to read
HDD 1980	100 MB	1.2 MB/s	~ 1 min
HDD 2020	4 TB	125 MB/s	~ 9 hours

HDDs are moving deeper in the memory hierarchy, and new algorithms are designed for new faster storage devices

How do these modern storage devices perform?

Solid State Drives & NVMs

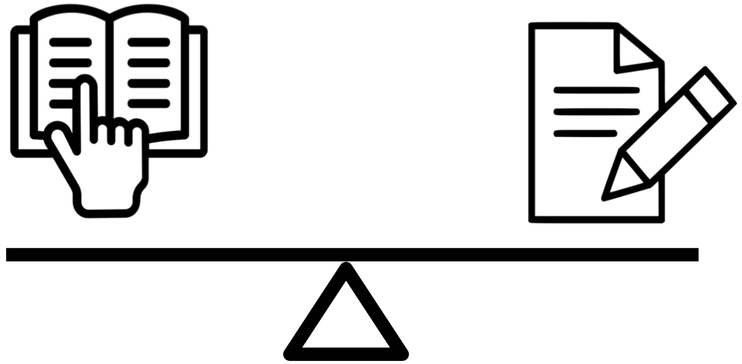
SSDs

- SATA SSDs
- PCIe SSDs (NVMe SSDs)
- Zoned SSDs
- Open SSDs

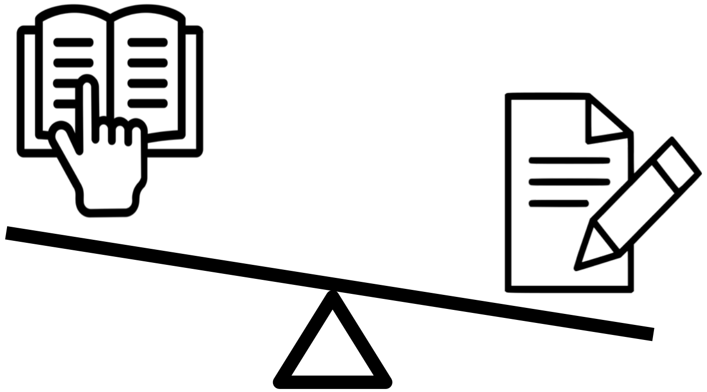
NVMs

- PCM
- MRAM
- STT-RAM
- 3D Xpoint (Intel's Optane)

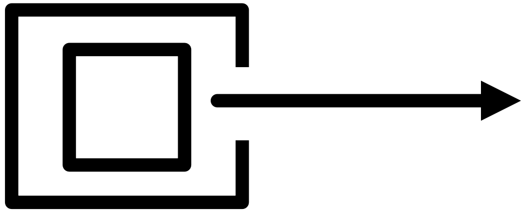
Modern Storage Devices



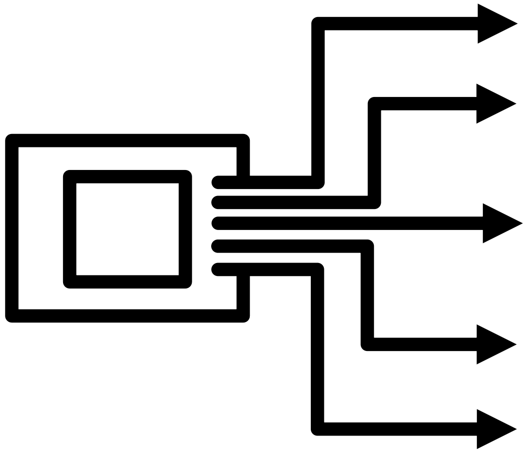
Symmetric cost for Read & Write



Read/Write Asymmetry

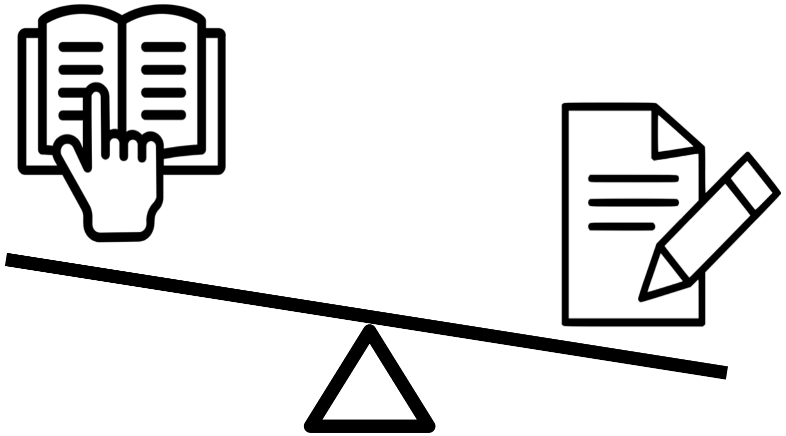


One I/O at a time



Concurrency

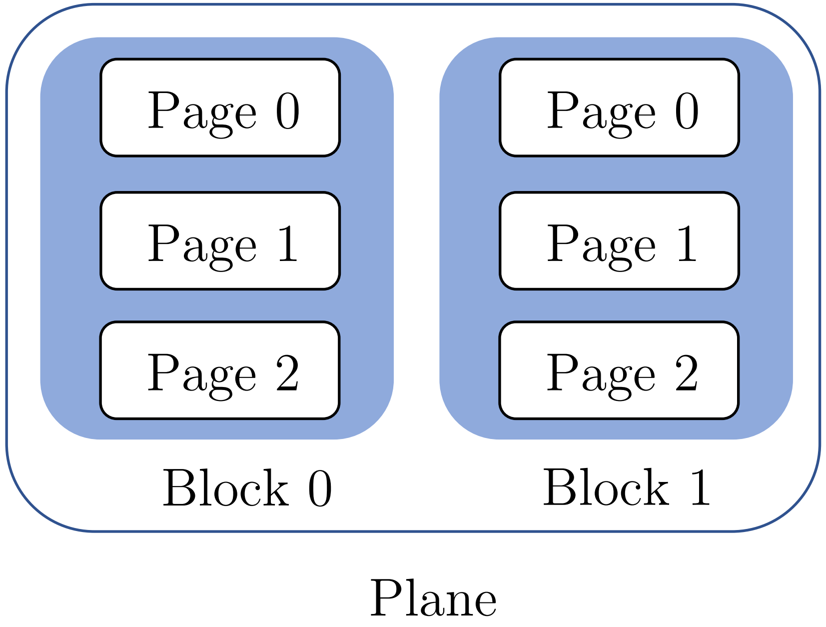
Read/Write Asymmetry



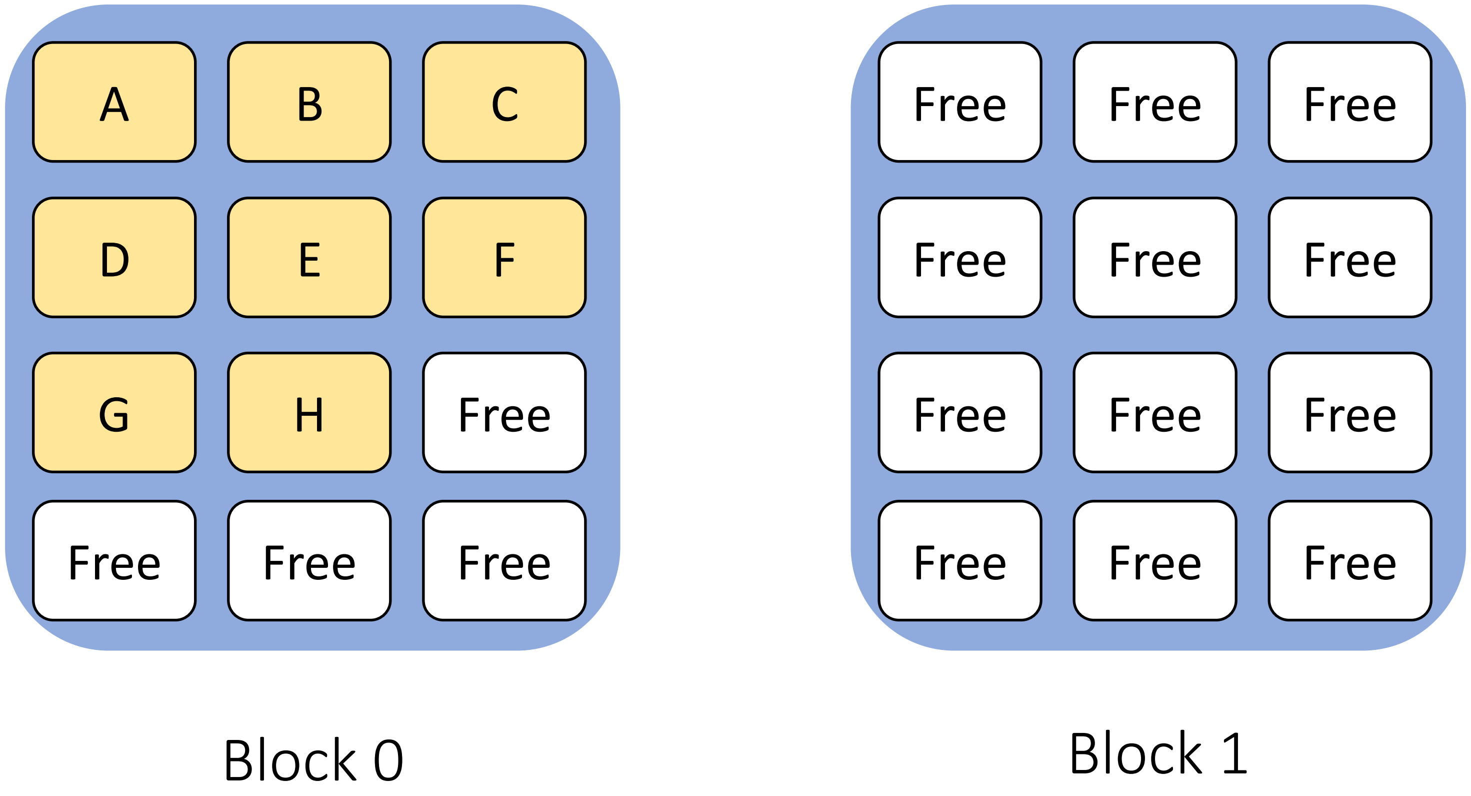
Writes in SSD

Out-of-place updates cause invalidation

Invalidation causes **garbage collection**



Writes in SSD

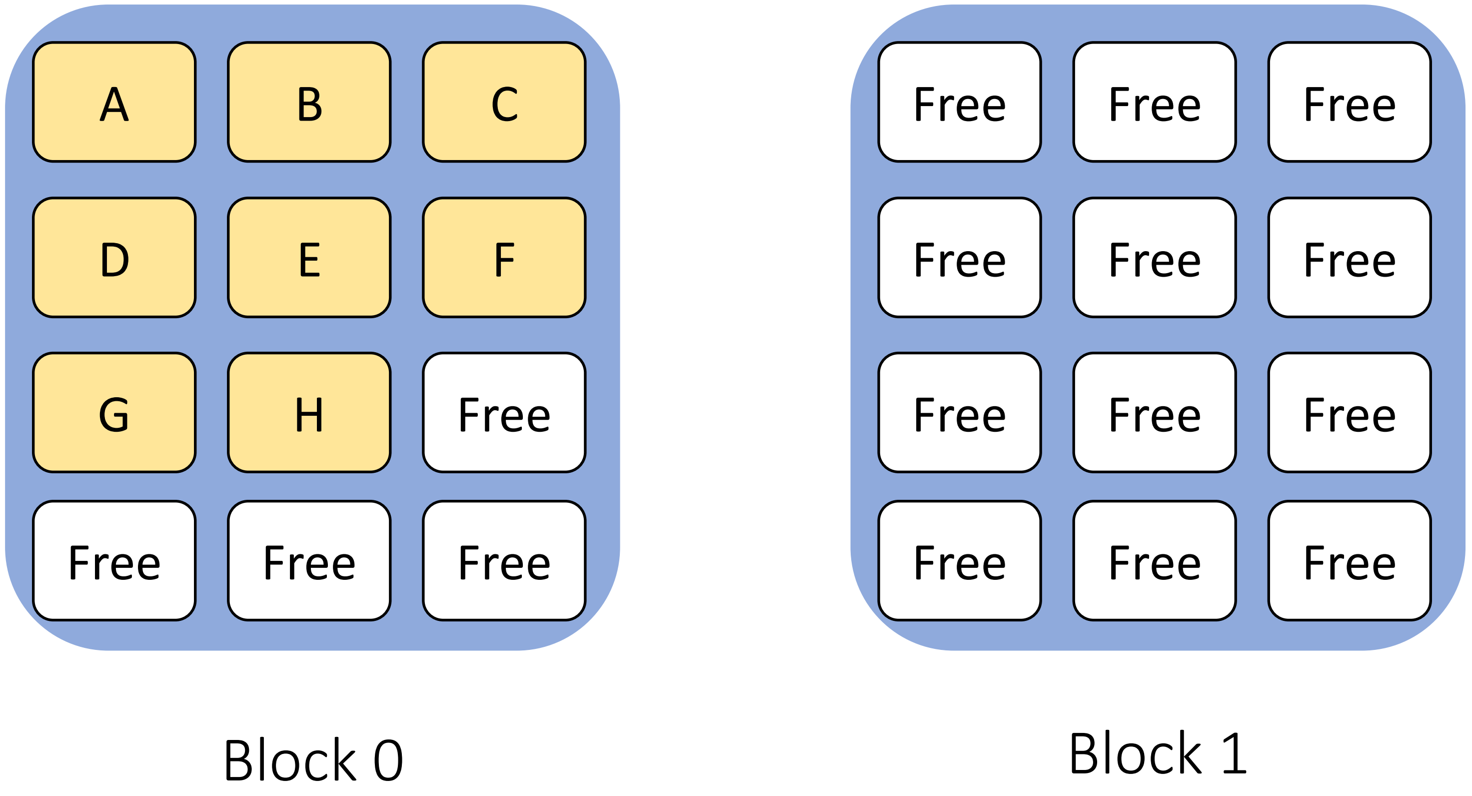


Writing in a free page isn't costly!

Writes in SSD

Update

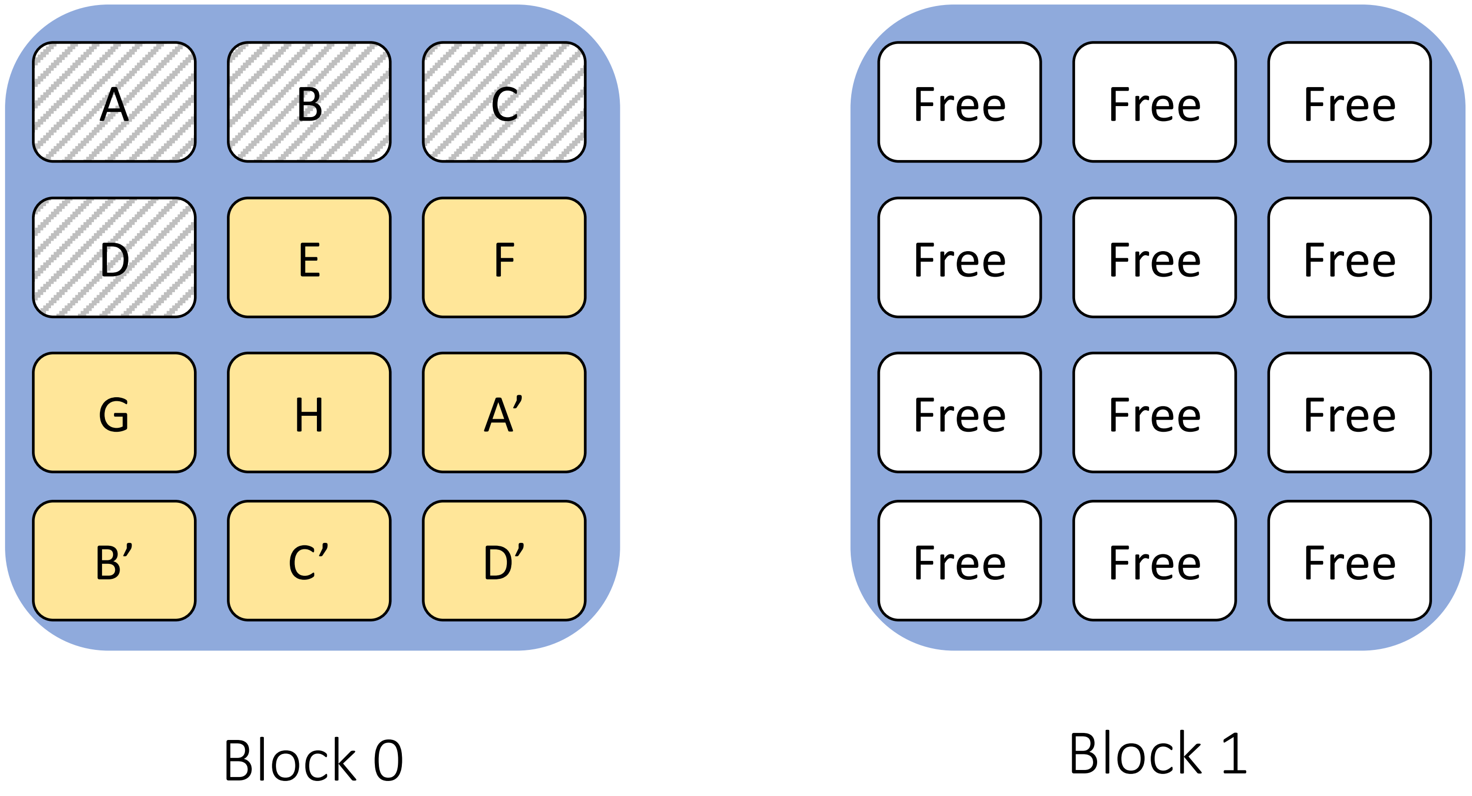
A, B, C, D



Writes in SSD

Update

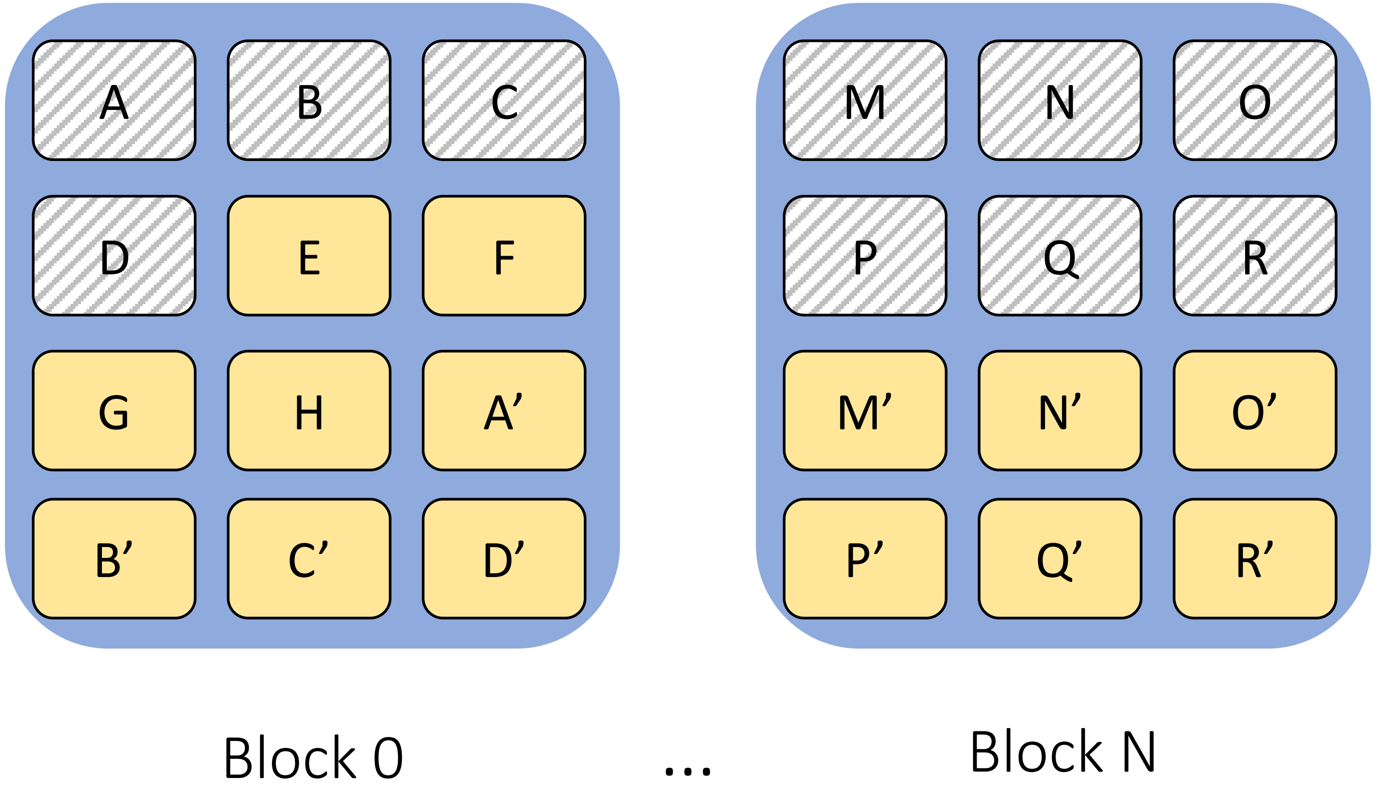
A, B, C, D



Not all updates are costly!

Writes in SSD

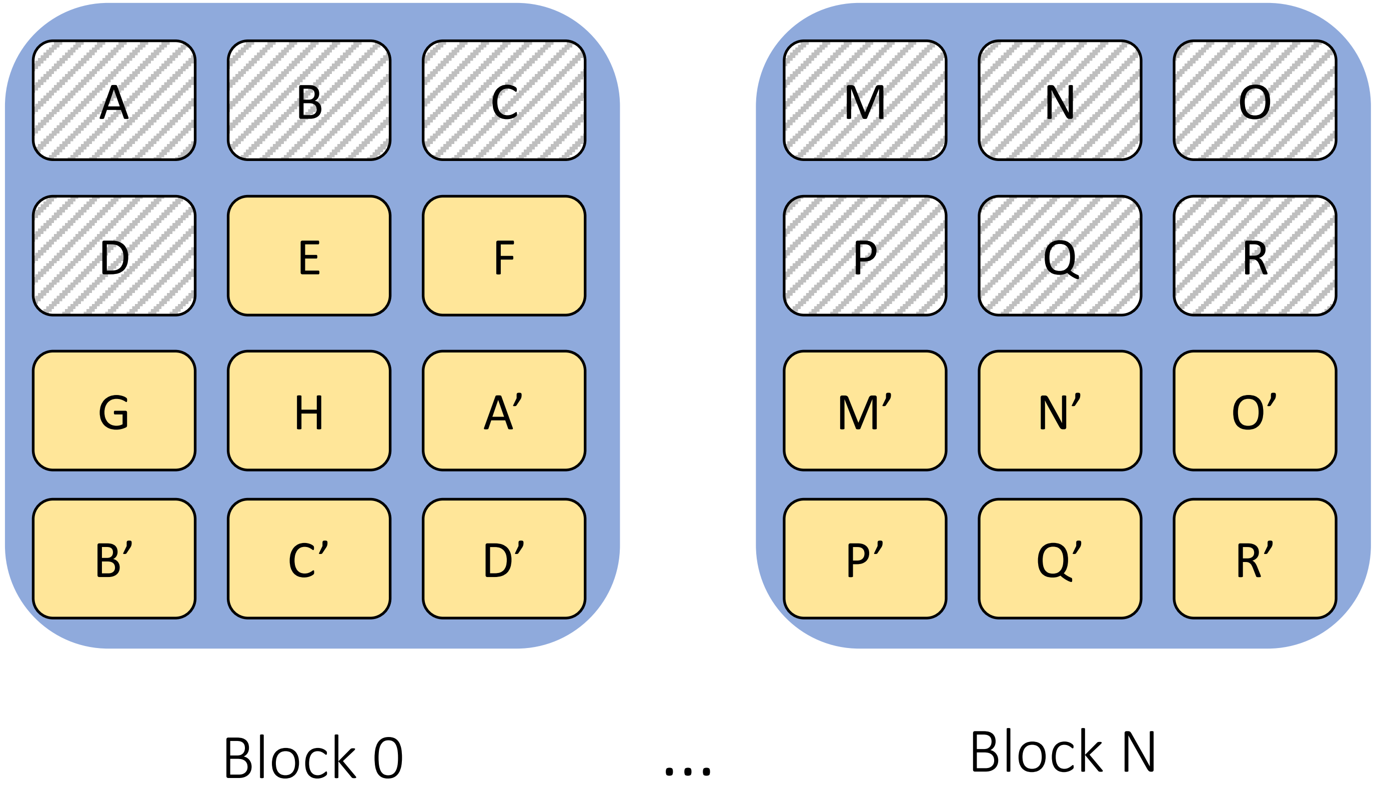
What if there is no space?



Writes in SSD

What if there is no space?

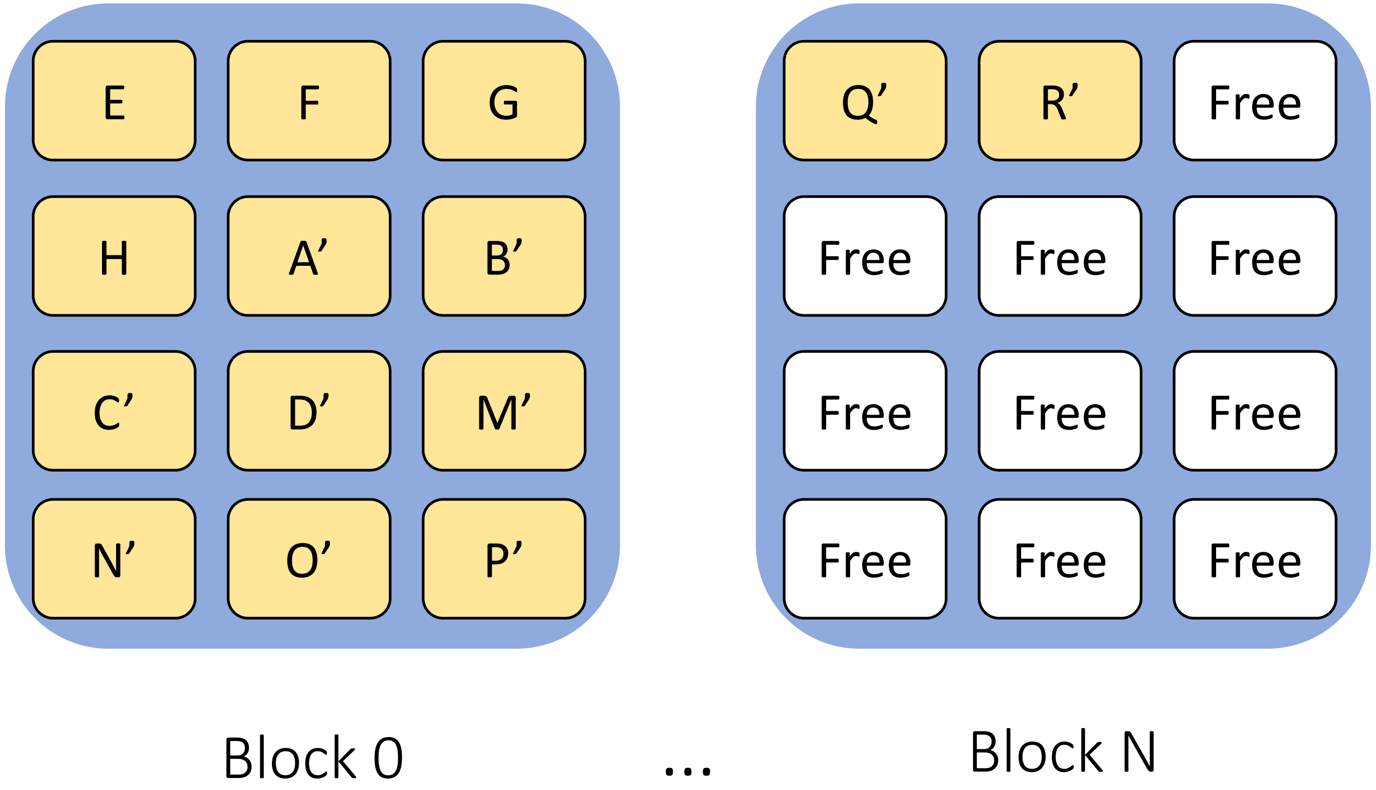
Garbage Collection!



Writes in SSD

What if there is no space?

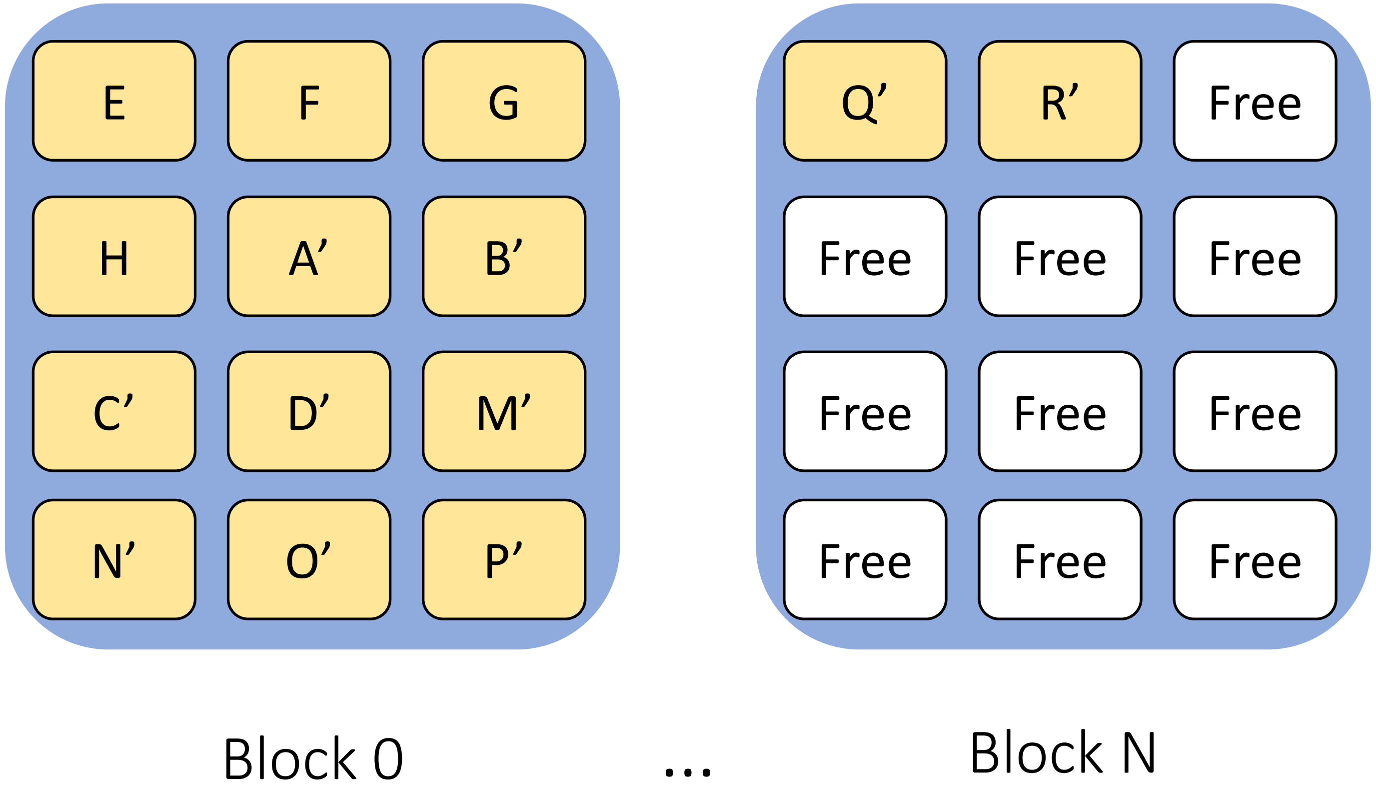
Garbage Collection!



Writes in SSD

What if there is no space?

Garbage Collection!

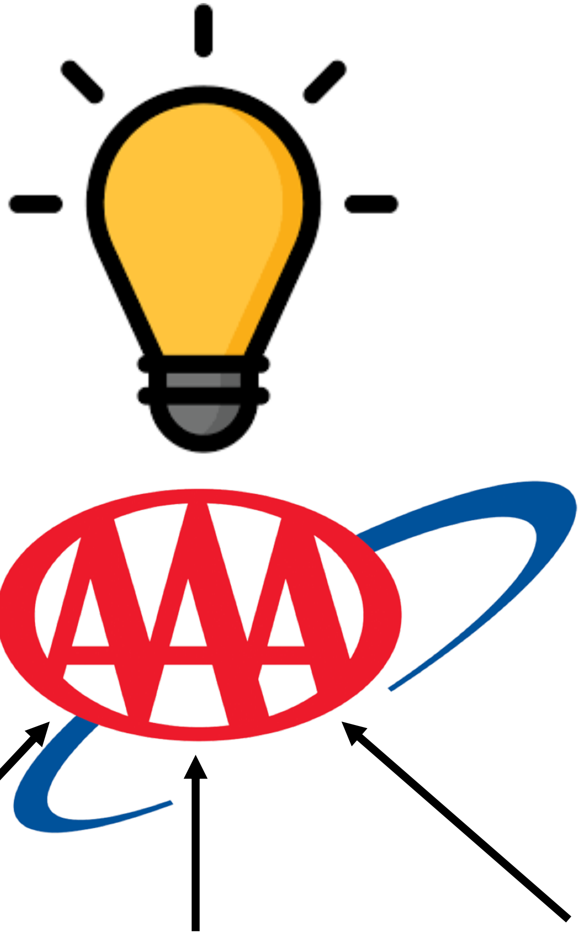


Higher average update cost (due to GC) → ***Read/Write asymmetry***

Read/Write Asymmetry - Example

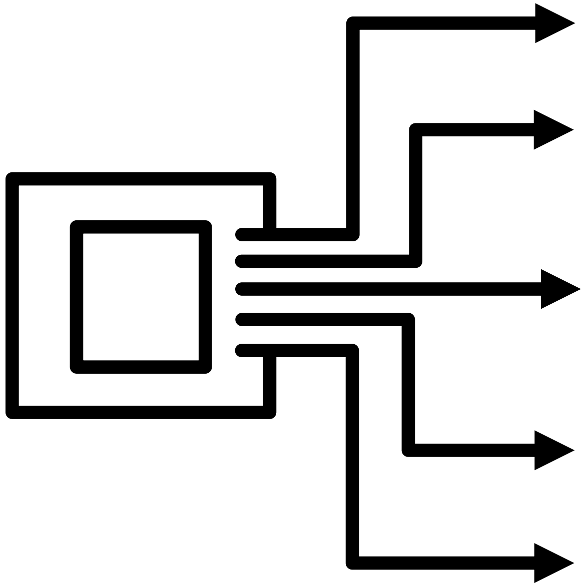
Intel Device	Advertised Random Read IOPS	Advertised Random Write IOPS	Advertised Asymmetry
D5-P4320	427k	36k	11.9
DC-P4500	626k	51k	12.3
DC-P4610	643k	199k	3.2
Optane 900P	550k	500k	1.1
Optane H10	330k	250k	1.3

Read/Write Asymmetry

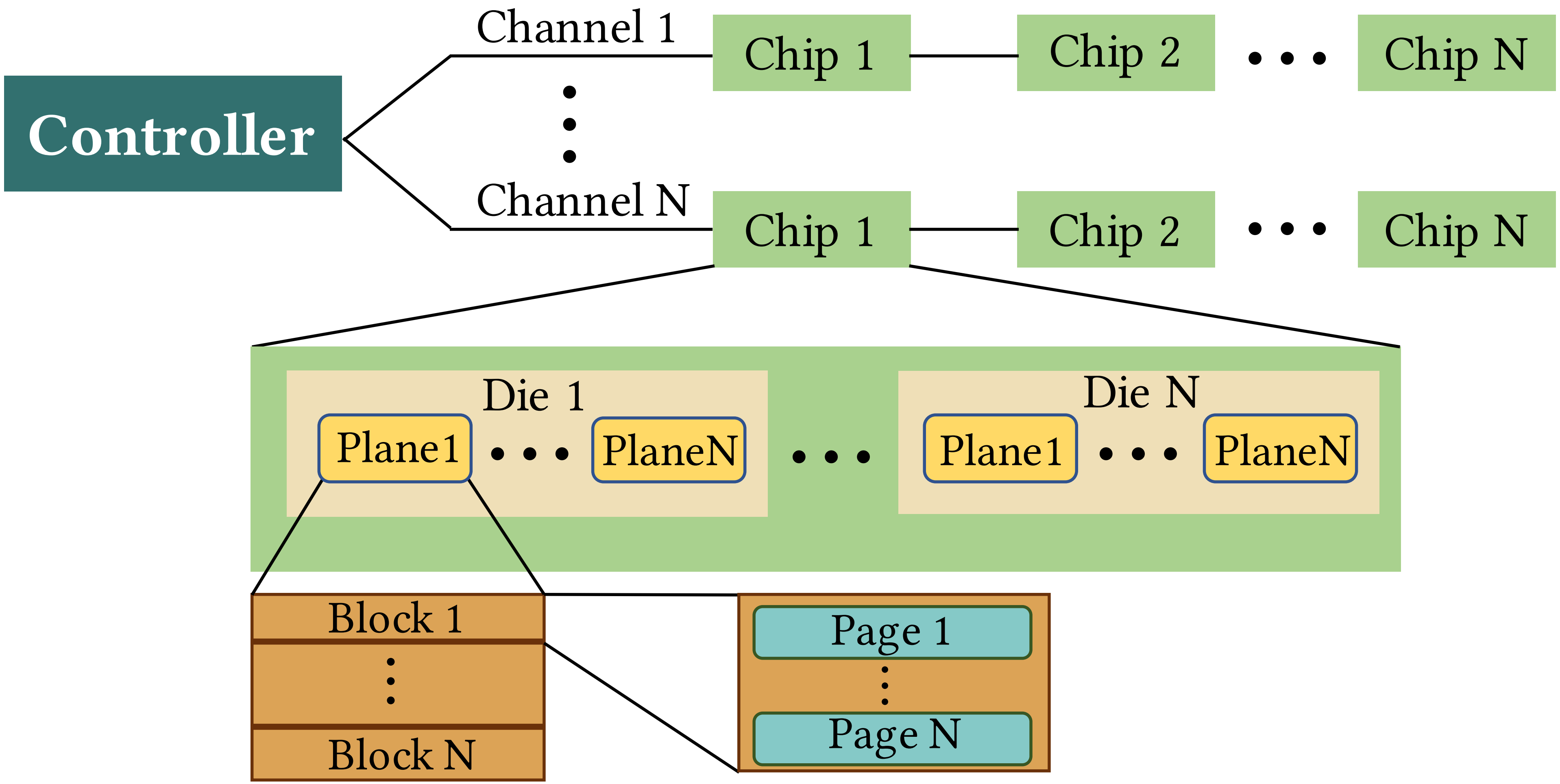


Asymmetry-Aware Algorithms

Concurrency

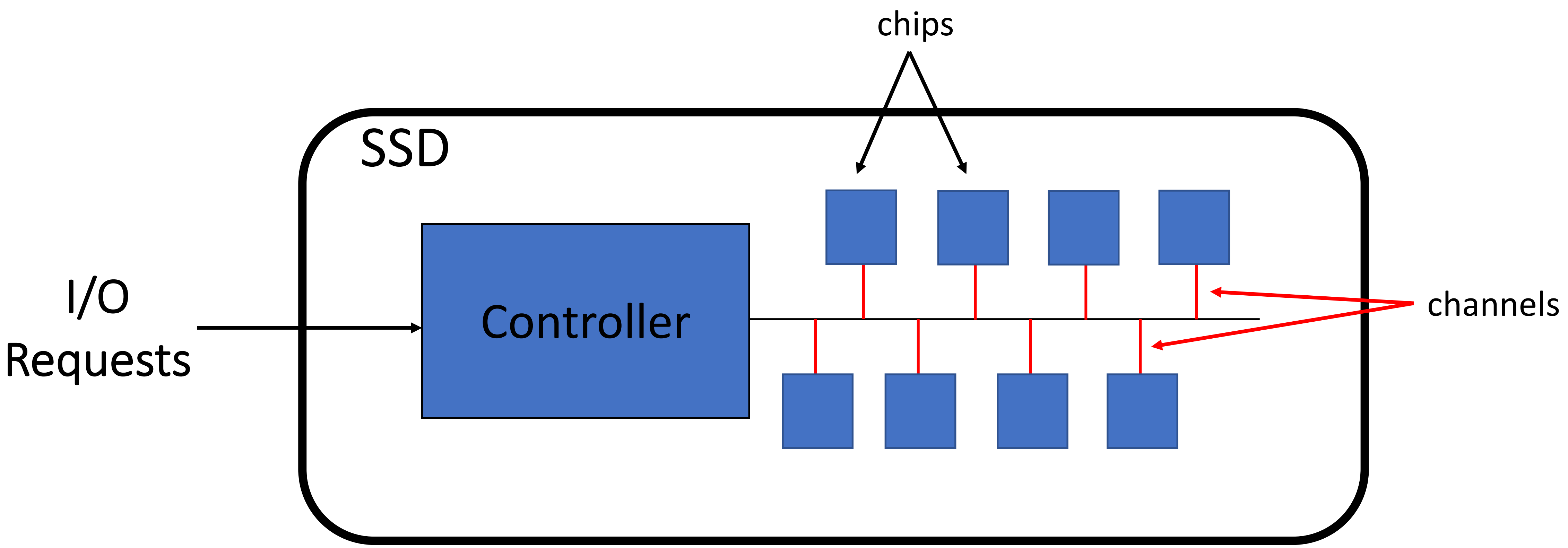


Internals of an SSD



Parallelism at different levels (e.g. channel, chip, die, plane block, page)

Concurrency in SSD (simplified)



Benchmarking

Benchmarking

Tools

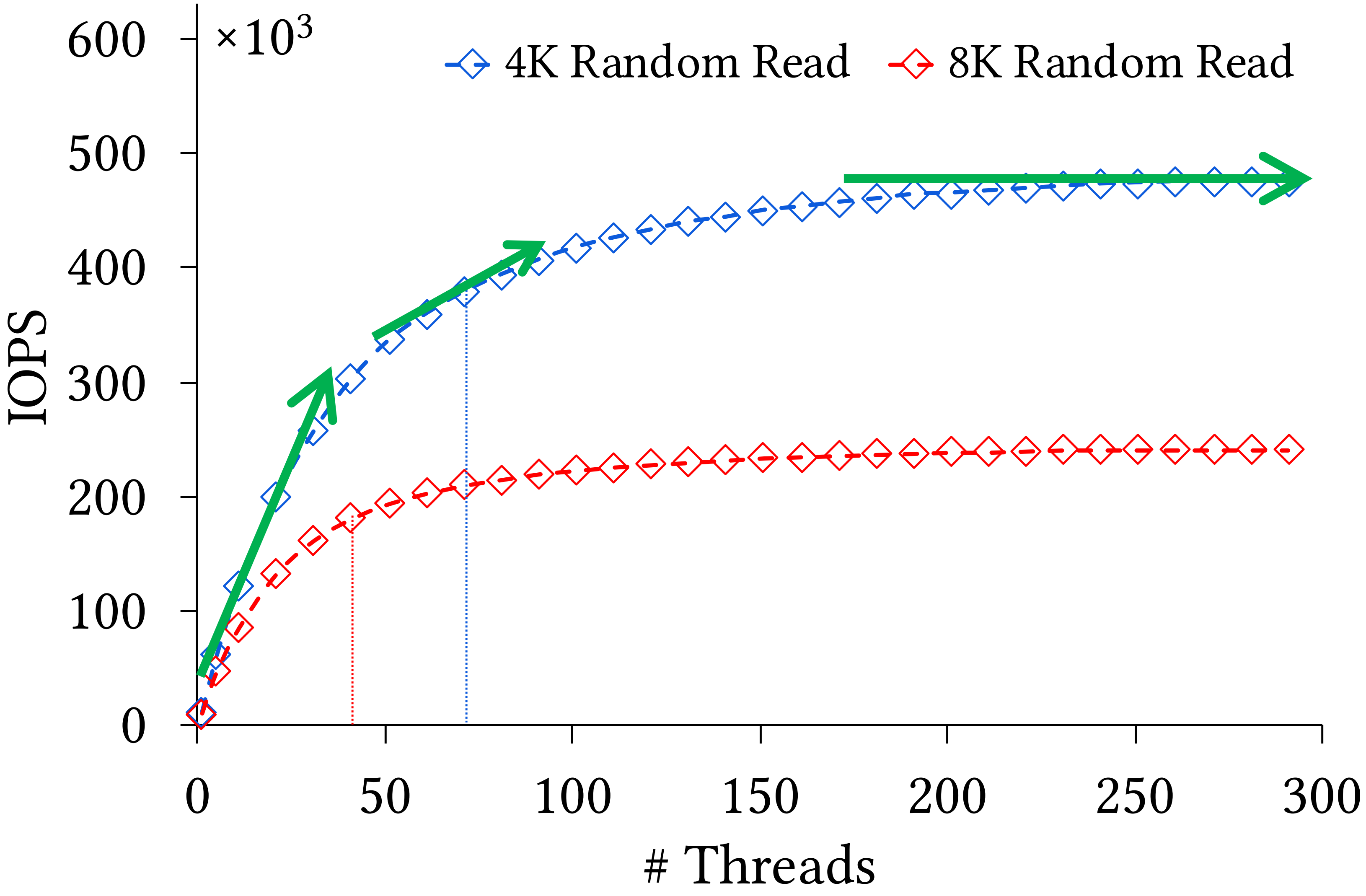
- Custom micro-benchmarking infrastructure
- fio
- Intel’s SPDK

Setup

- With File System
- Without File System

Measuring Asymmetry/Concurrency (With FS)

Device: Dell P4510 (1TB)



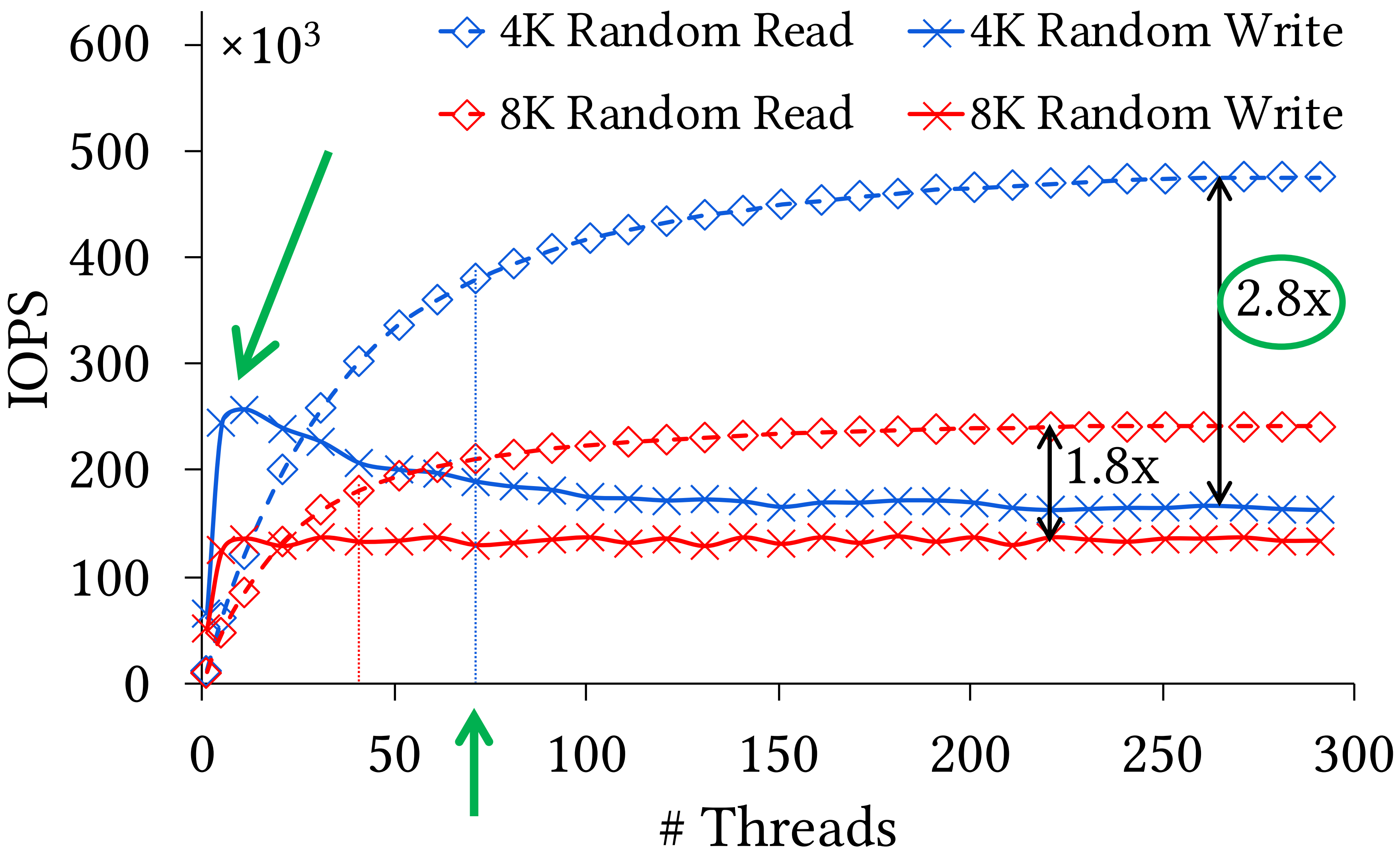
Measuring Asymmetry/Concurrency (With FS)

Device: Dell P4510 (1TB)

For 4K random read,

Asymmetry: 2.8

Concurrency: 70



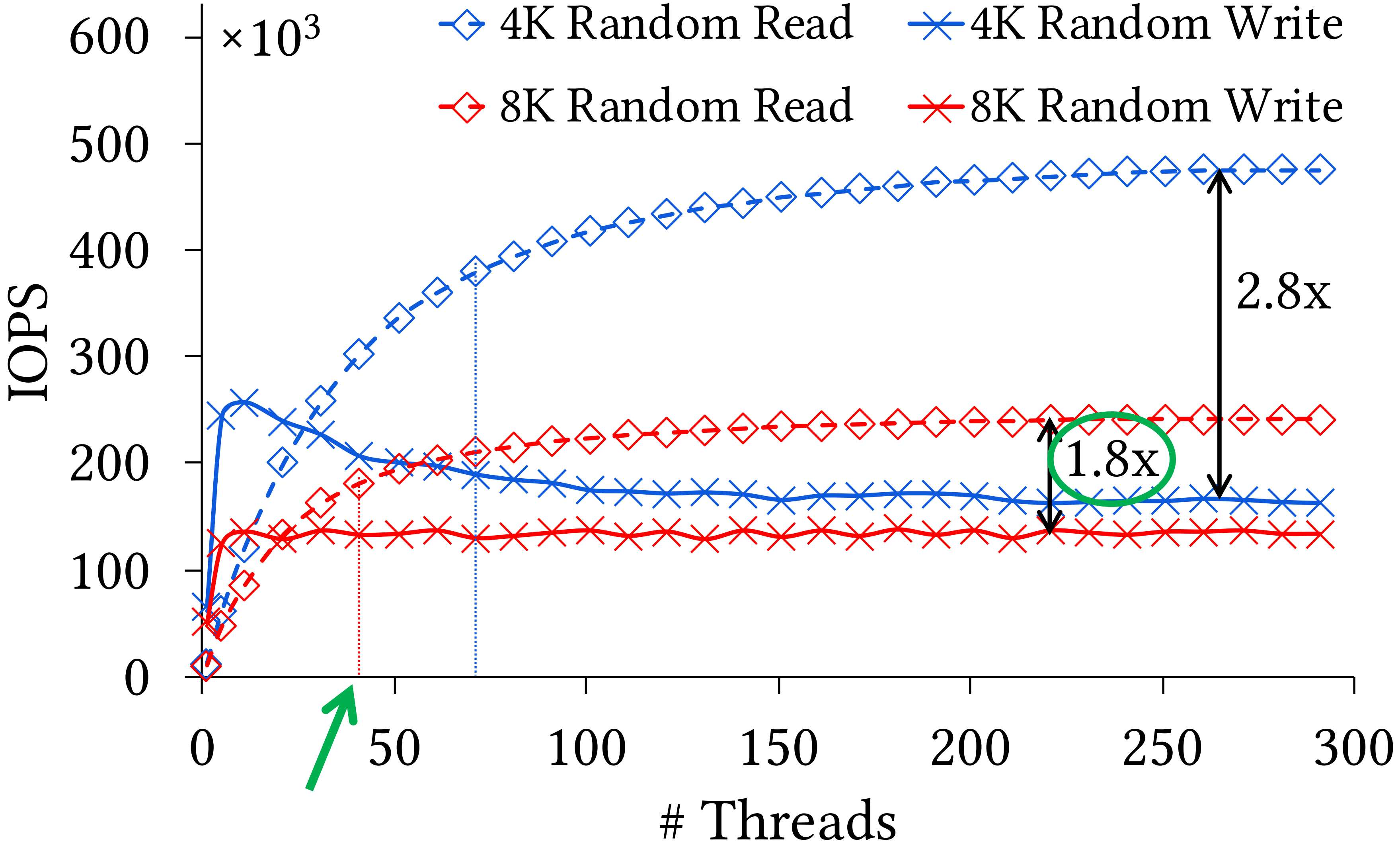
Measuring Asymmetry/Concurrency (With FS)

Device: Dell P4510 (1TB)

For 8K random write,

Asymmetry: 1.8

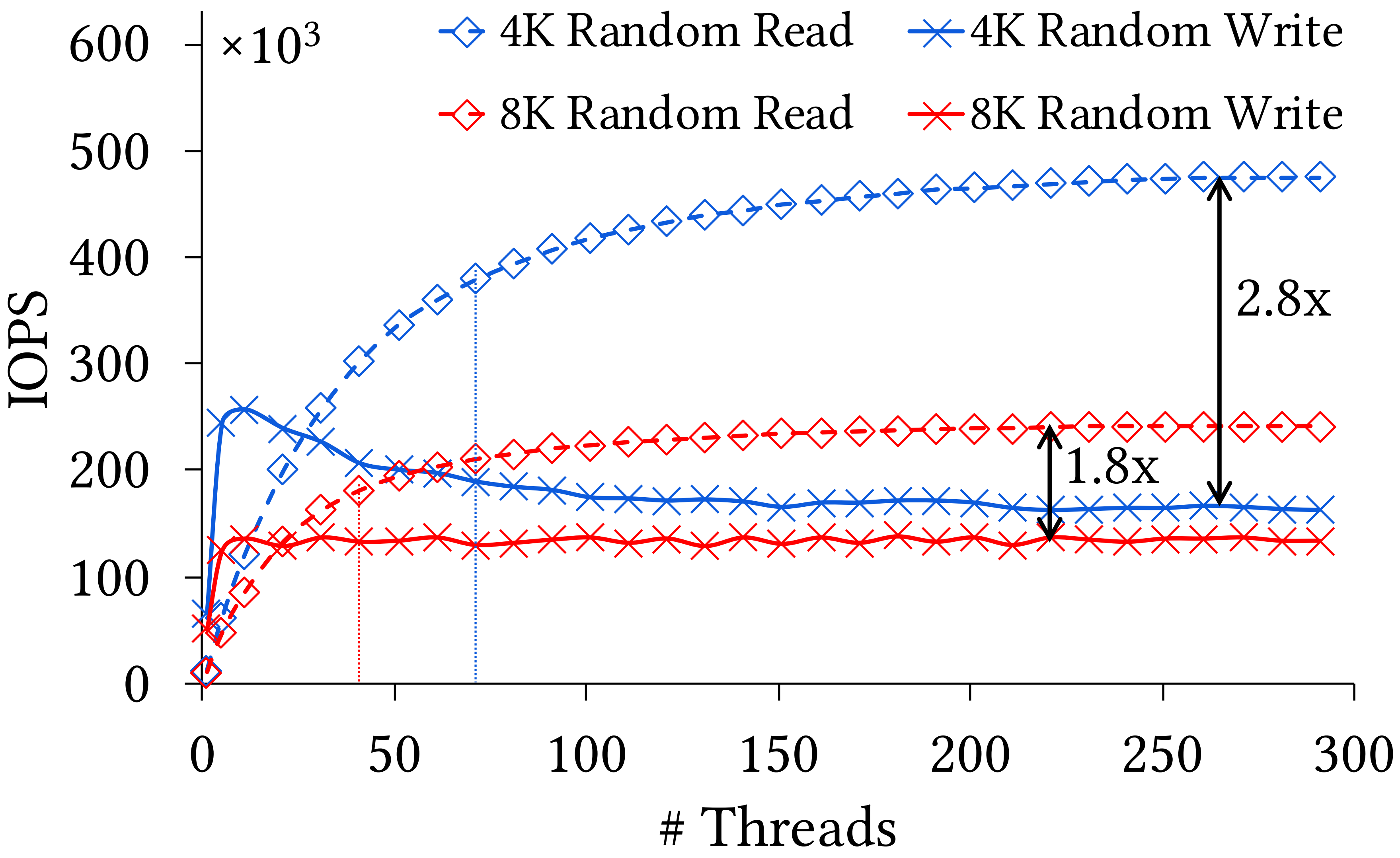
Concurrency: 10



Measuring Asymmetry/Concurrency (With FS)

Device: Dell P4510 (1TB)

Asymmetry and concurrency depends on request type and access granularity



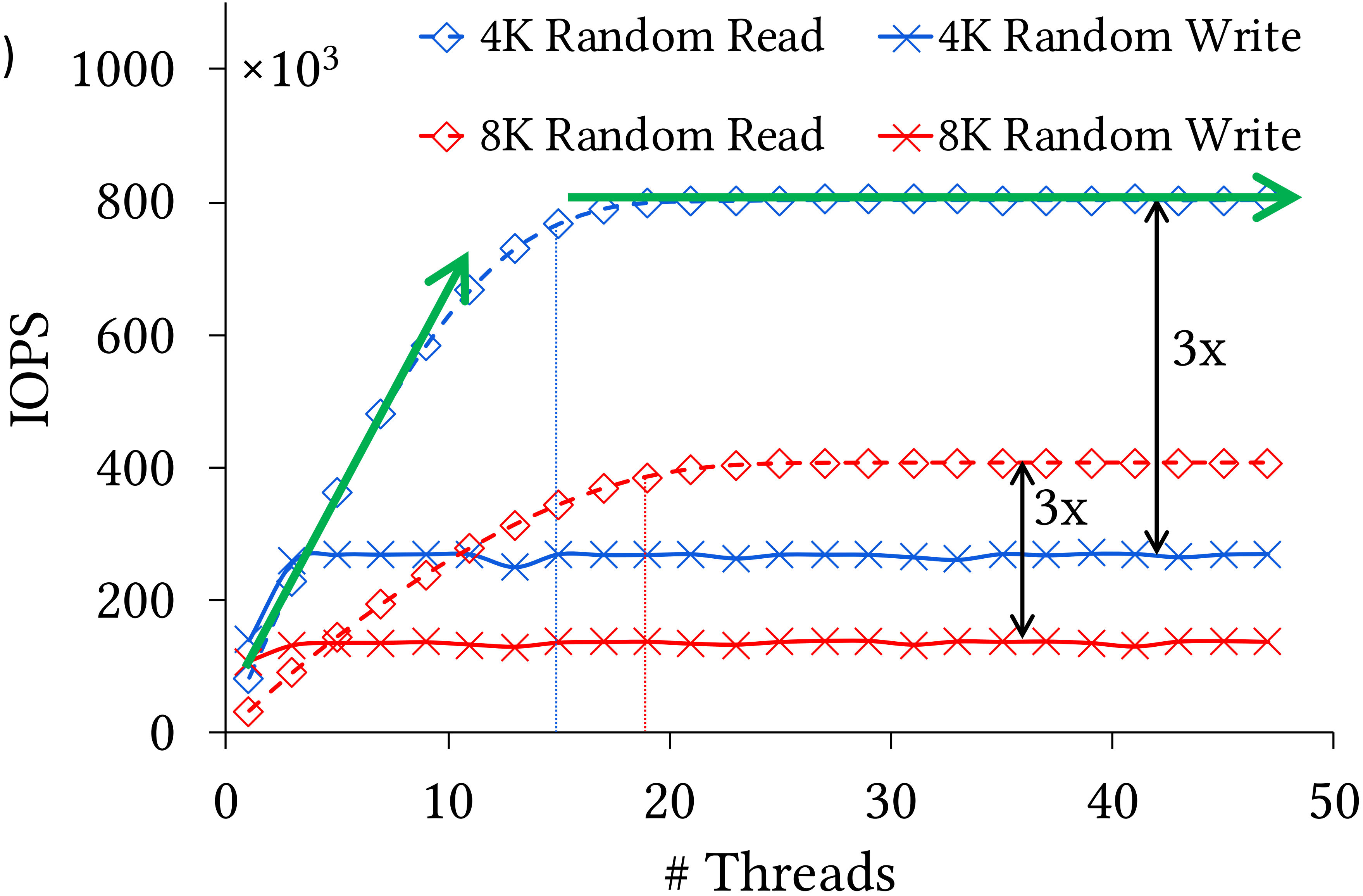
Measuring Asymmetry/Concurrency (Without FS)

Device: Dell P4510 (1TB)

For 4K random reads,

Asymmetry: 3

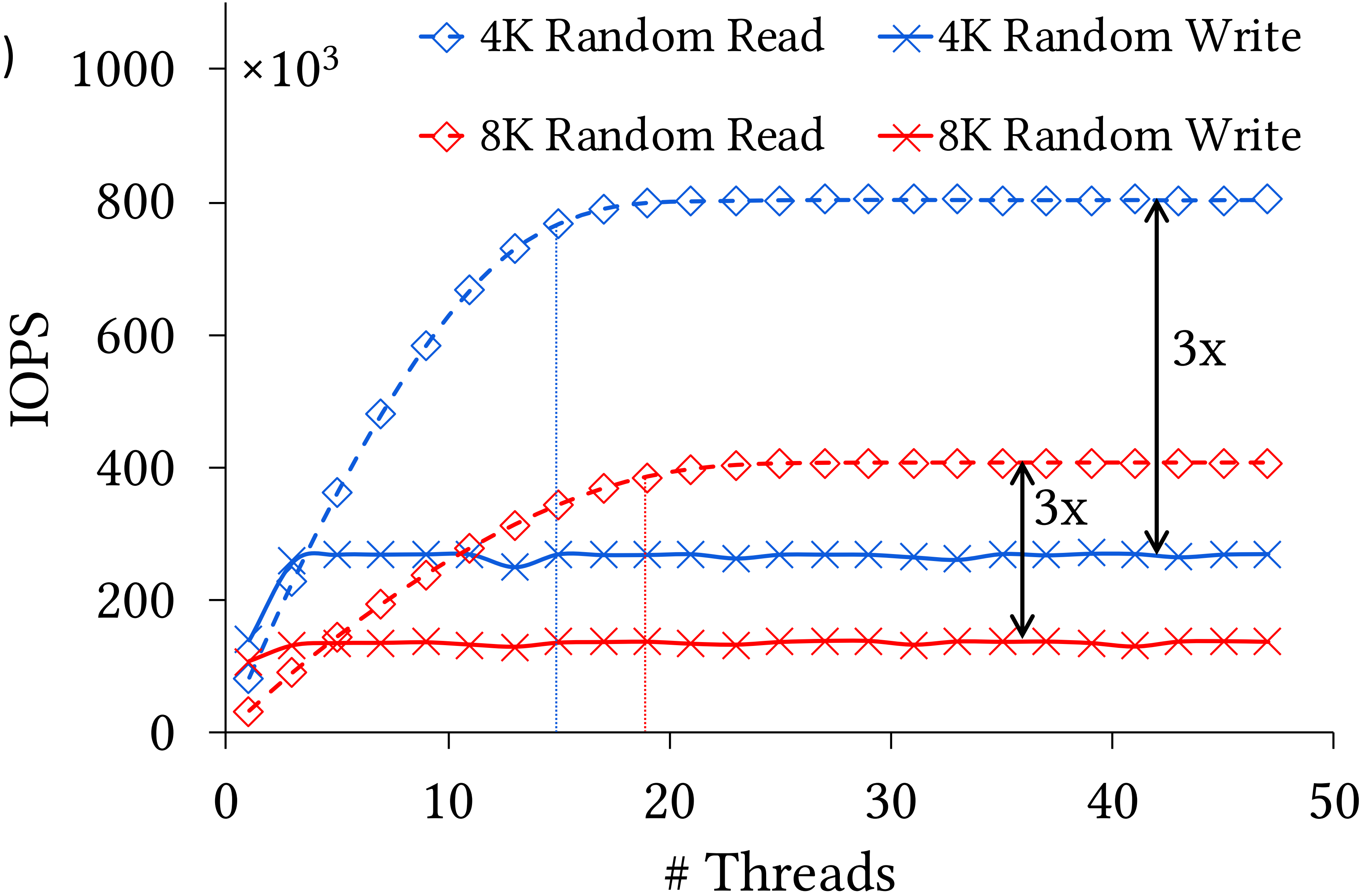
Concurrency: 14



Measuring Asymmetry/Concurrency (Without FS)

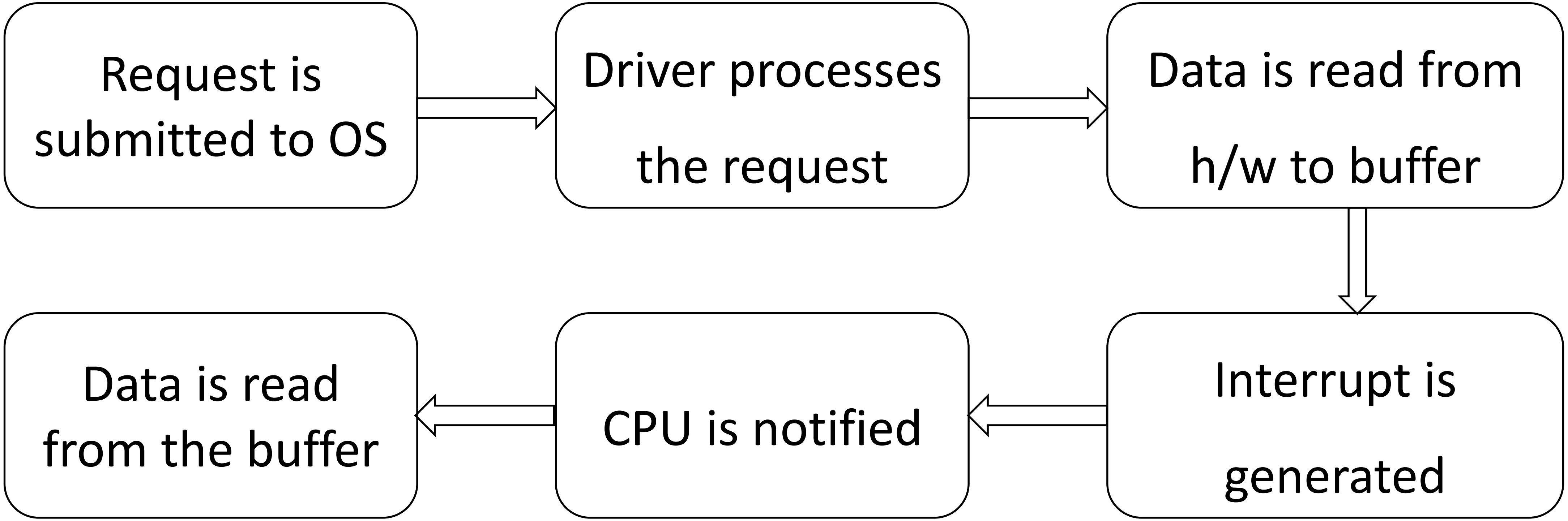
Device: Dell P4510 (1TB)

Much stable performance without the file system!



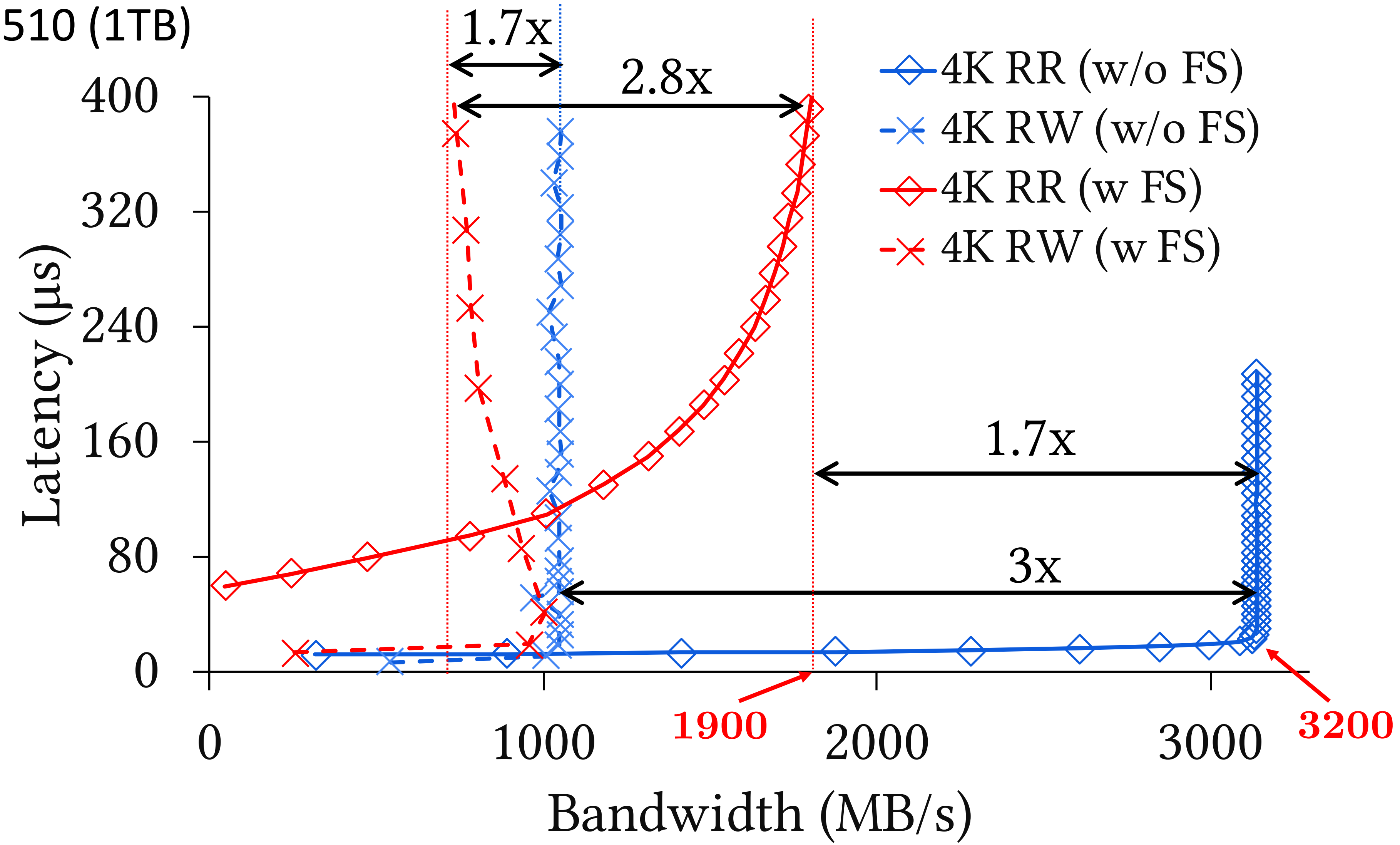
Can the File System be the Bottleneck?

Interrupt-based model



Can the File System be the Bottleneck?

Device: Dell P4510 (1TB)

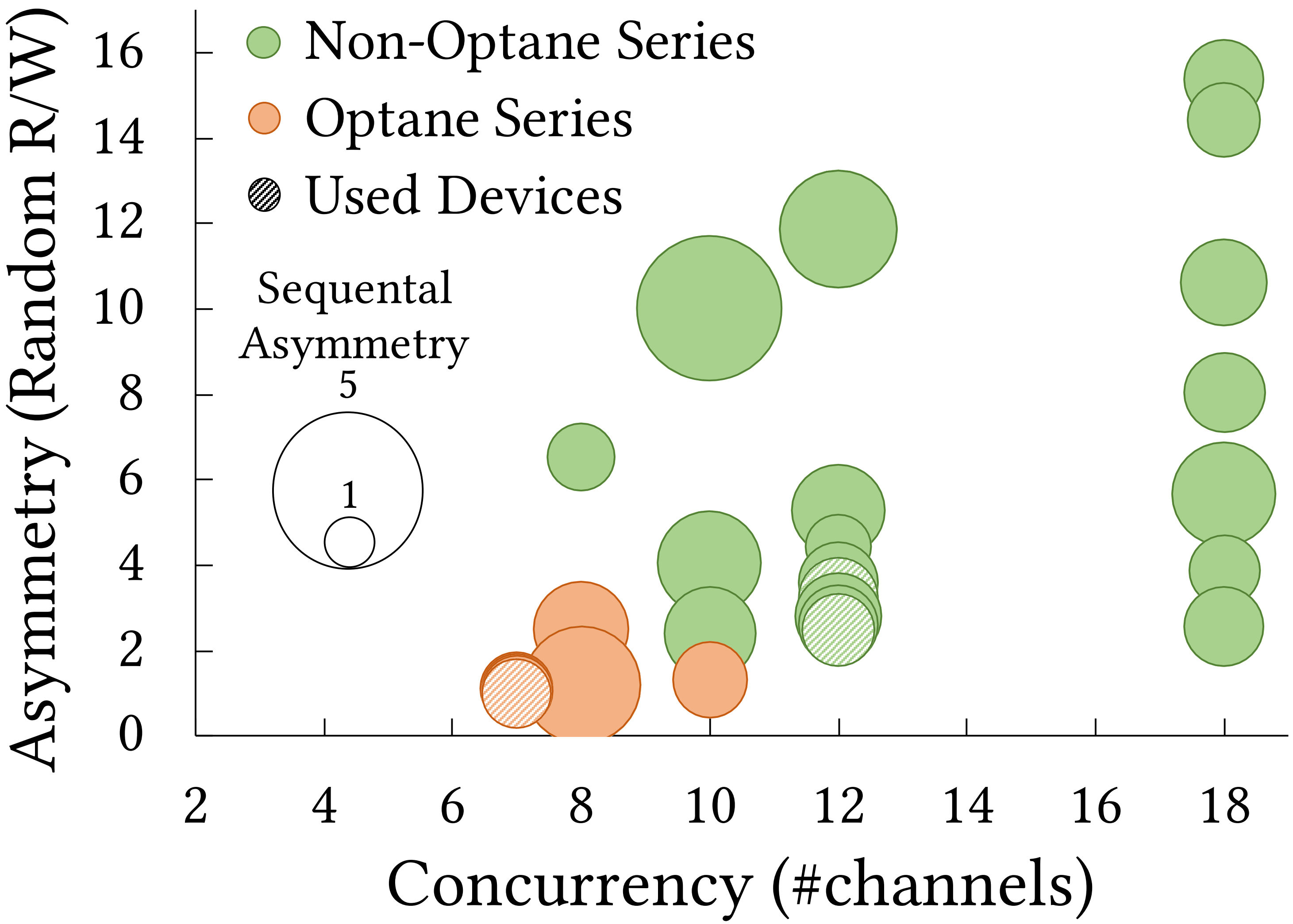


Measuring Asymmetry/Concurrency

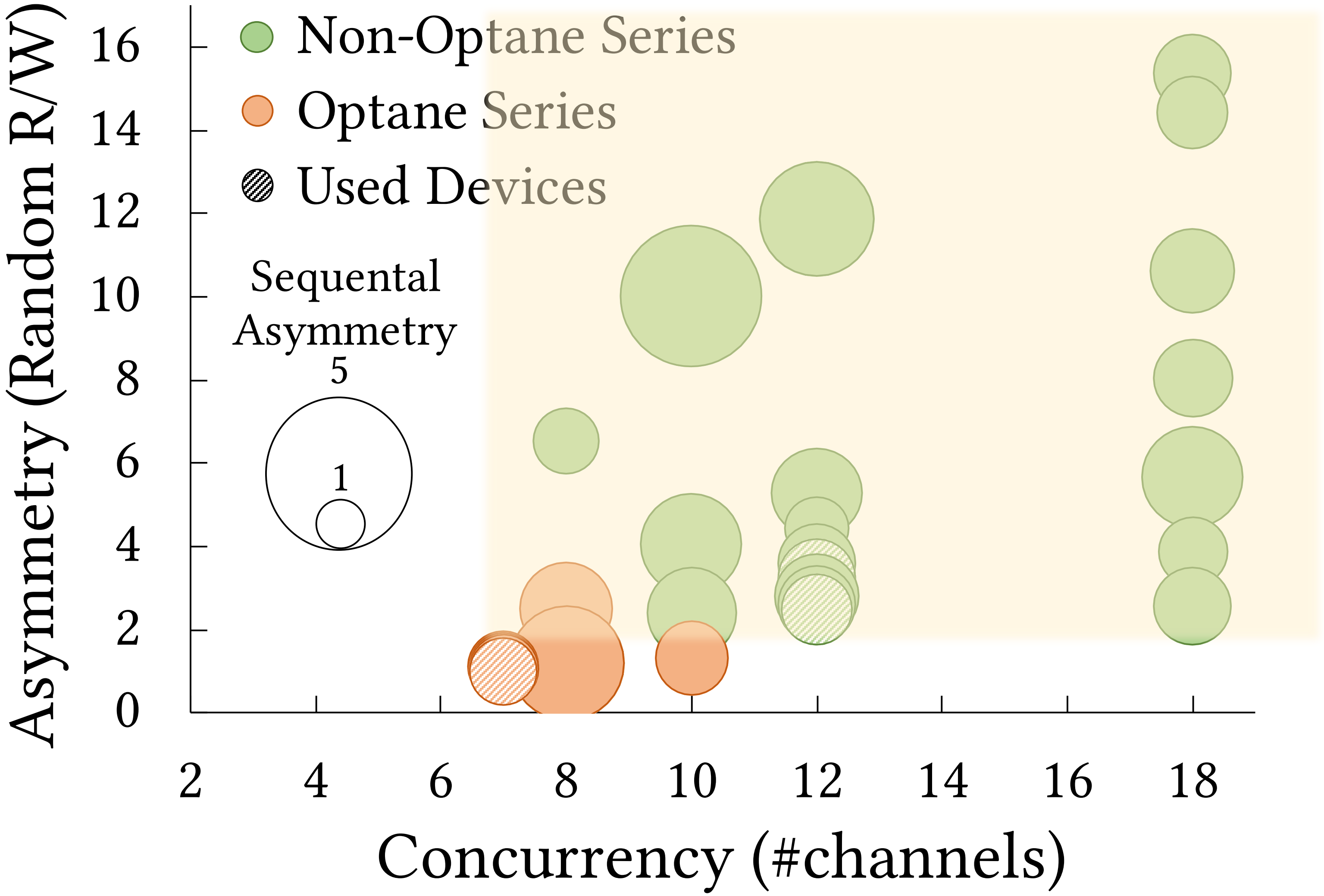
Table 2: Empirical *Asymmetry* and *Concurrency*.

	4KB			8KB		
Device	α	k_r	k_w	α	k_r	k_w
Optane SSD	1.1	6	5	1.0	4	4
PCIe SSD (with FS)	2.8	80	8	1.9	40	7
PCIe SSD (w/o FS)	3.0	16	6	3.0	15	4
SATA SSD	1.5	25	9	1.3	21	5
Virtual SSD	2.0	11	19	1.9	6	10

Modern Storage Devices



Modern Storage Devices



Most devices have high asymmetry

How should the I/O model be adapted in light of

read/write asymmetry and **concurrency**?

Parametric I/O Model

PIO (M, k, α)

Main Memory Size — Concurrency — Asymmetry

$\boldsymbol{PIO(M, k_r, k_w, \alpha)}$ *assumes a fast main memory with capacity M, and storage of unbounded capacity that has* ***read/write asymmetry*** α, *and* ***read (write) concurrency*** k_r (k_w).

Performance Analysis

We classify storage-intensive applications into four classes

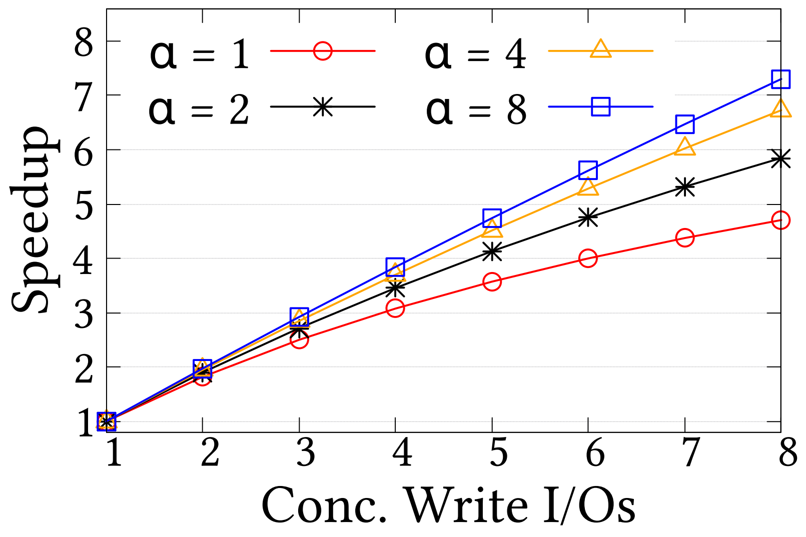
- Unbatchable Reads, Unbatchable Writes
- Unbatchable Reads, Batchable Writes
- Batchable Reads, Unbatchable Writes
- Batchable Reads, Batchable Writes

Unbatchable Reads, Batchable Writes

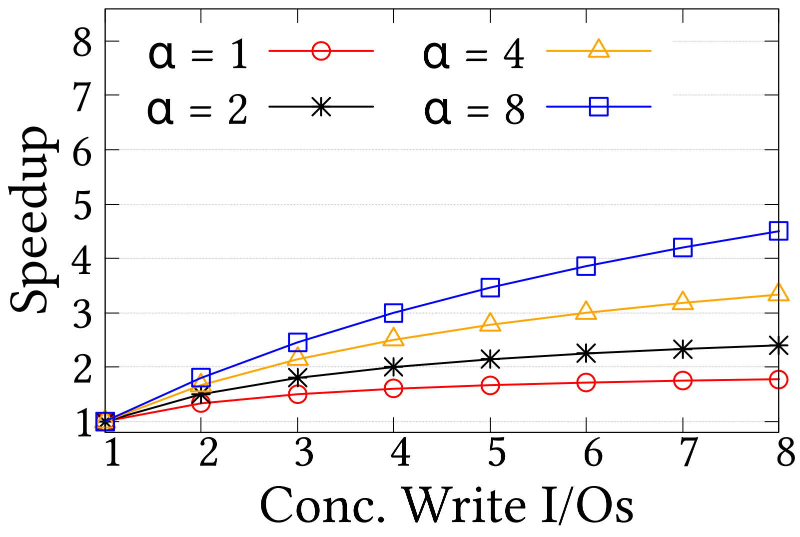
- Can exploit write concurrency (k_w) by batching writes
- Amortized cost per write following PIO is $\frac{\alpha}{k_w}$
- Example: DBMS bufferpool

Unbatchable Reads, Batchable Writes

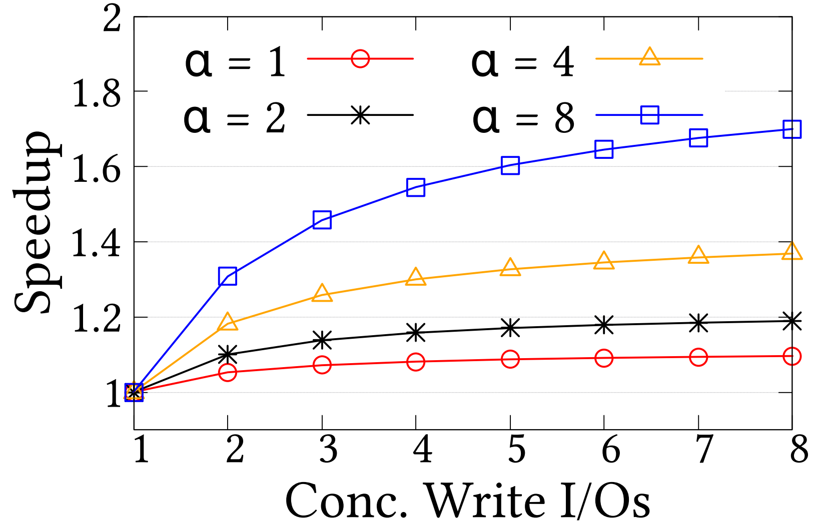
10% reads



50% reads



90% reads



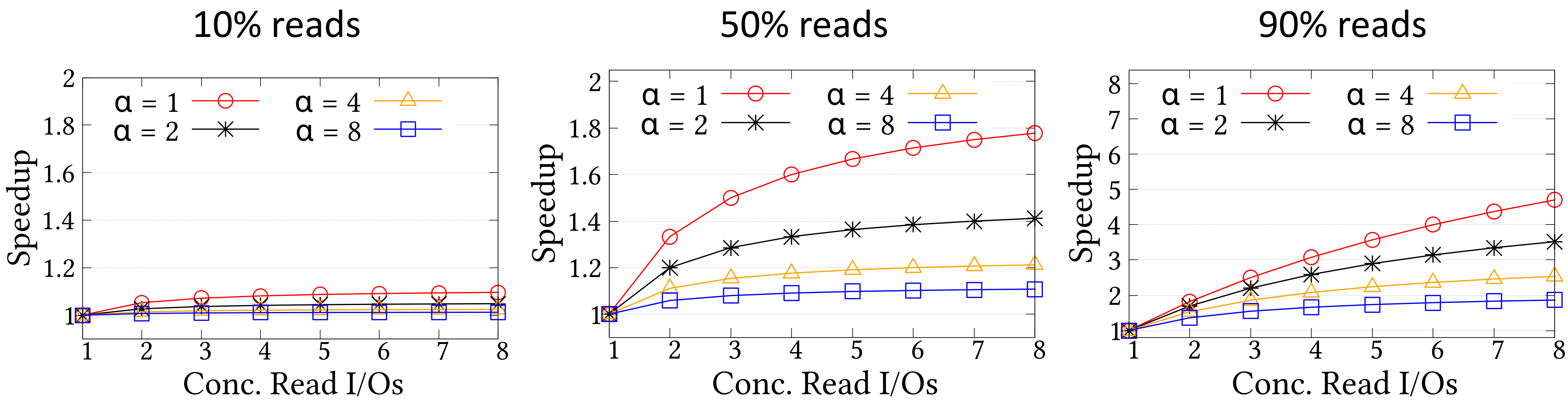
Speedup increases with increasing concurrent I/Os

Speedup depends on asymmetry – gain is ***higher*** for a device with ***higher*** asymmetry

Batchable Reads, Unbatchable Writes

- Can exploit read concurrency (k_r) by batching read
- Amortized cost per read following PIO is $\frac{1}{k_r}$
- Example: Graph traversal

Batchable Reads, Unbatchable Writes



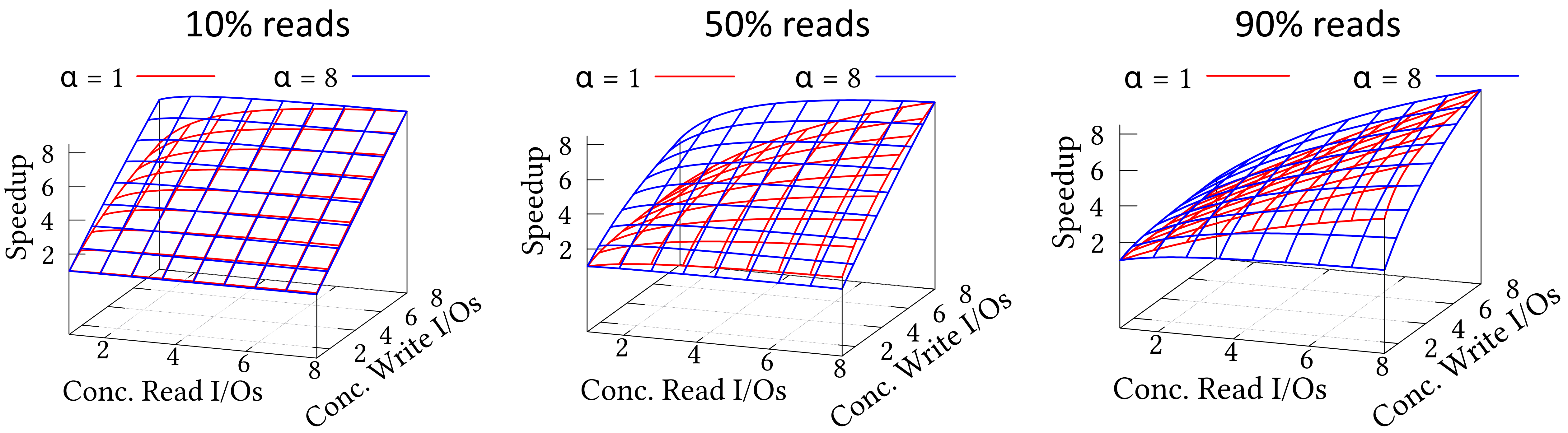
Speedup increases with increasing concurrent I/Os

Speedup depends on asymmetry – gain is ***higher*** for a device with ***lower*** asymmetry

Batchable Reads, Unbatchable Writes

- Can exploit both read and write concurrency $(k_r\,,\,k_w)$
- Amortized cost per read following PIO is $\frac{1}{k_r}$
- Amortized cost per write following PIO is $\frac{\alpha}{k_w}$
- Example: LSM compaction

Batchable Reads, Batchable Writes



Speedup increases with increasing concurrent I/Os, and depends on asymmetry

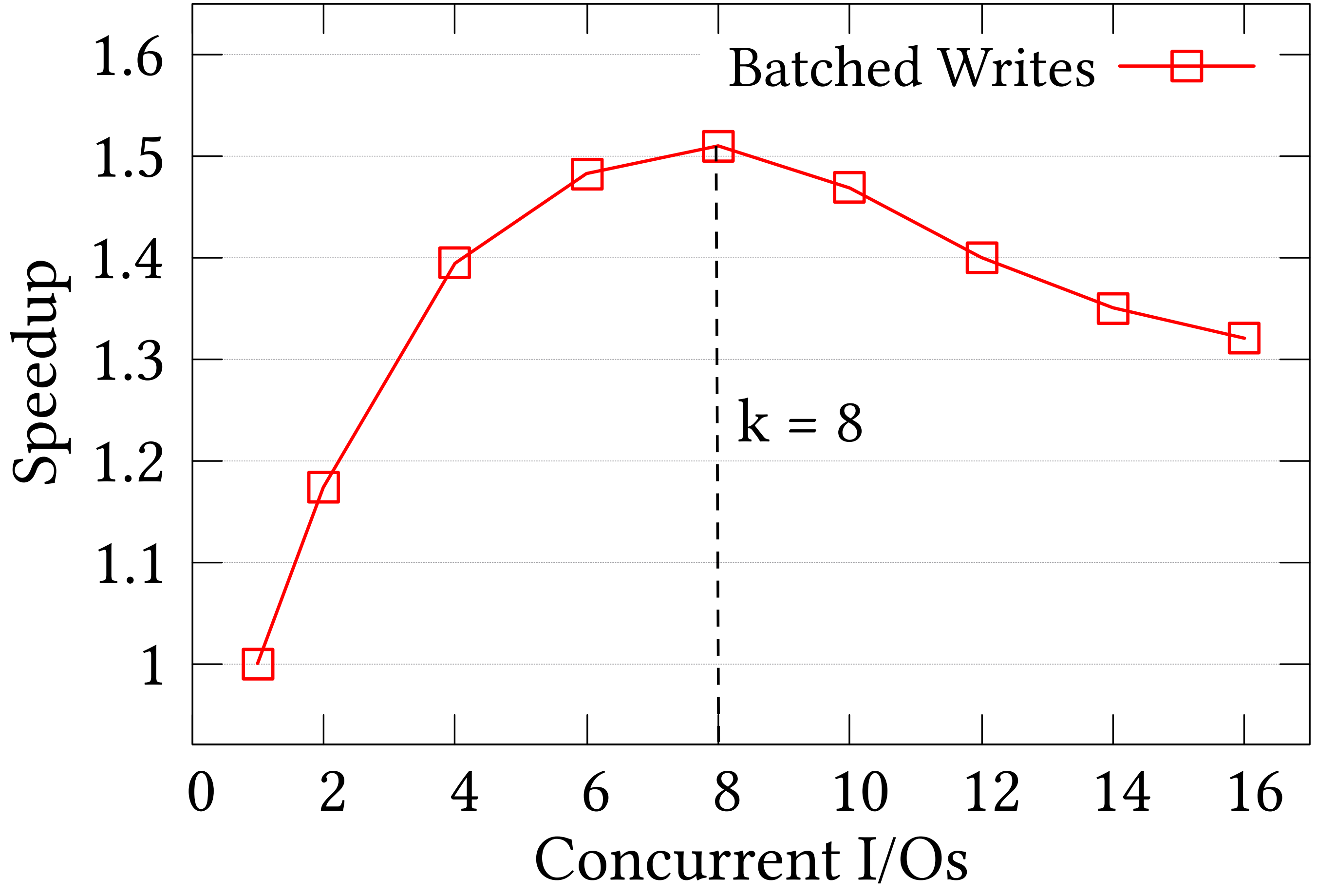
Impact of utilizing write concurrency is higher than utilizing read concurrency

Importance of using Proper k

A sample application with unbatchable reads and batchable writes

We use PCIe SSD ($k_w = 8$) to run this concurrency-aware application

Optimal speedup at the device concurrency.



Guidelines for Algorithm Design

Know Thy Device

Exploit Device Concurrency

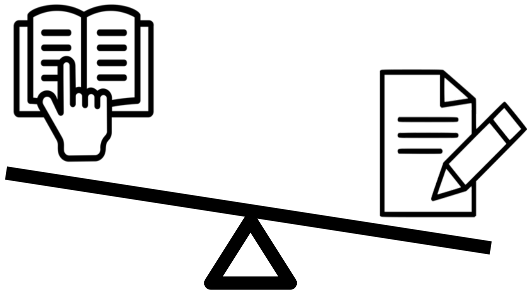
Use Concurrency with Care

It is suboptimal to treat a read and a write equally for a device with asymmetry

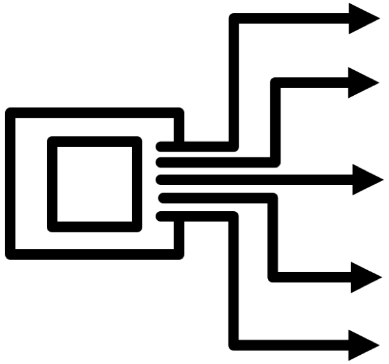
Asymmetry Controls Performance

Conclusion

Modern Storage Devices



Read/Write Asymmetry



Concurrency

Need for a new parametric I/O model

PIO (M, k, α)

Benefits of PIO (M, k, α)

- algorithms tailored to new devices
- Can capture ***any*** new device

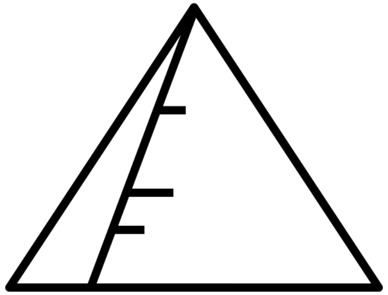
Prerequisite: quantify ***k*** and ***α***

Make *asymmetry and concurrency* part of *algorithm design*

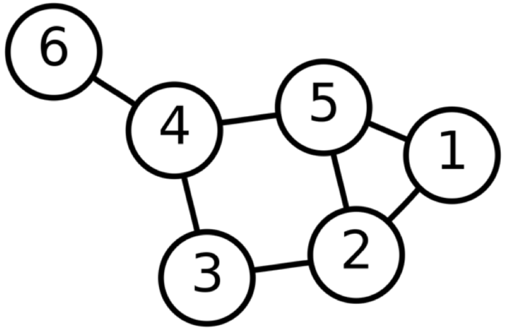
… not simply an engineering optimization

Build algorithms/data structures for storage devices
with asymmetry α and concurrency k

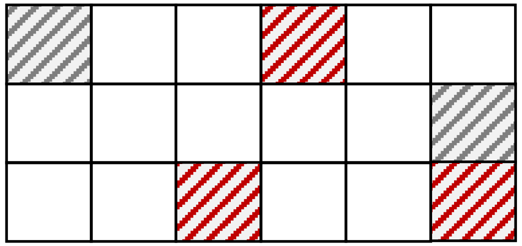
index structures



graph traversal algorithms



bufferpool management

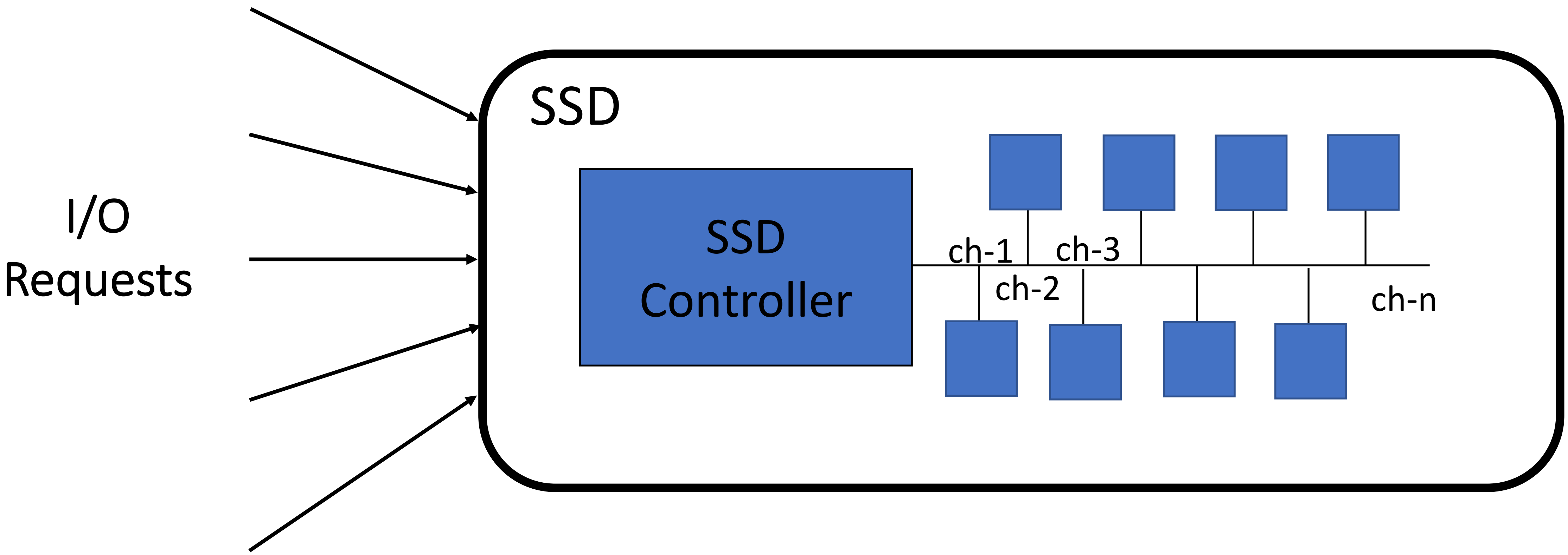


Thank You!

disc.bu.edu/pio

Backup Slides

Expected Parallelism



How many of the channels will be occupied for uniform distribution?

Expected Parallelism

E_n = expected number of empty channels after n I/Os are uniformly distributed

We use E_{n-1} and consider where the n^{th} I/O will be routed

it will be on an empty channel reducing them

with probability

$$E_n = \frac{E_{n-1}}{n}(E_{n-1} - 1) + \left(1 - \frac{E_{n-1}}{n}\right)E_{n-1} = \frac{n-1}{n}E_{n-1}$$

with probability

it will be on a non-empty channel

Since $E_0 = n$; $\quad E_n = \left(\frac{n-1}{n}\right)^n n$

Fraction of empty channels $= {}^{E_n}/_{n} = \left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e}$

So, on average $\left(1 - \frac{1}{e}\right)$ = **63.2%** channels will be accessed in parallel

Unbatchable Reads, Batchable Writes