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"

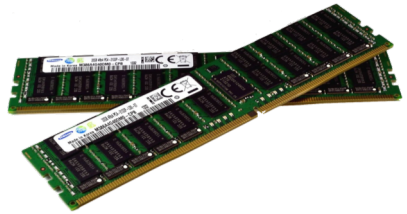
… is probably one of the most commonly read phrases in SIGMOD papers.


intel
Core™ i7

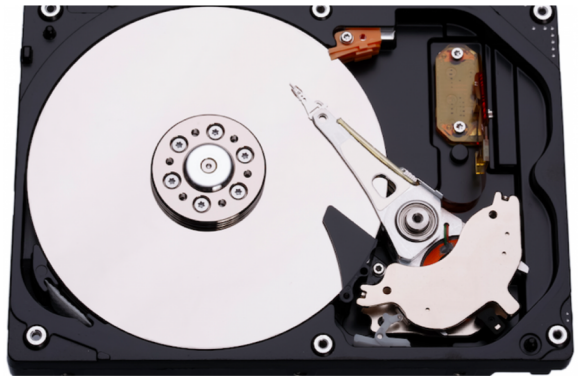

intel
Core™ i7

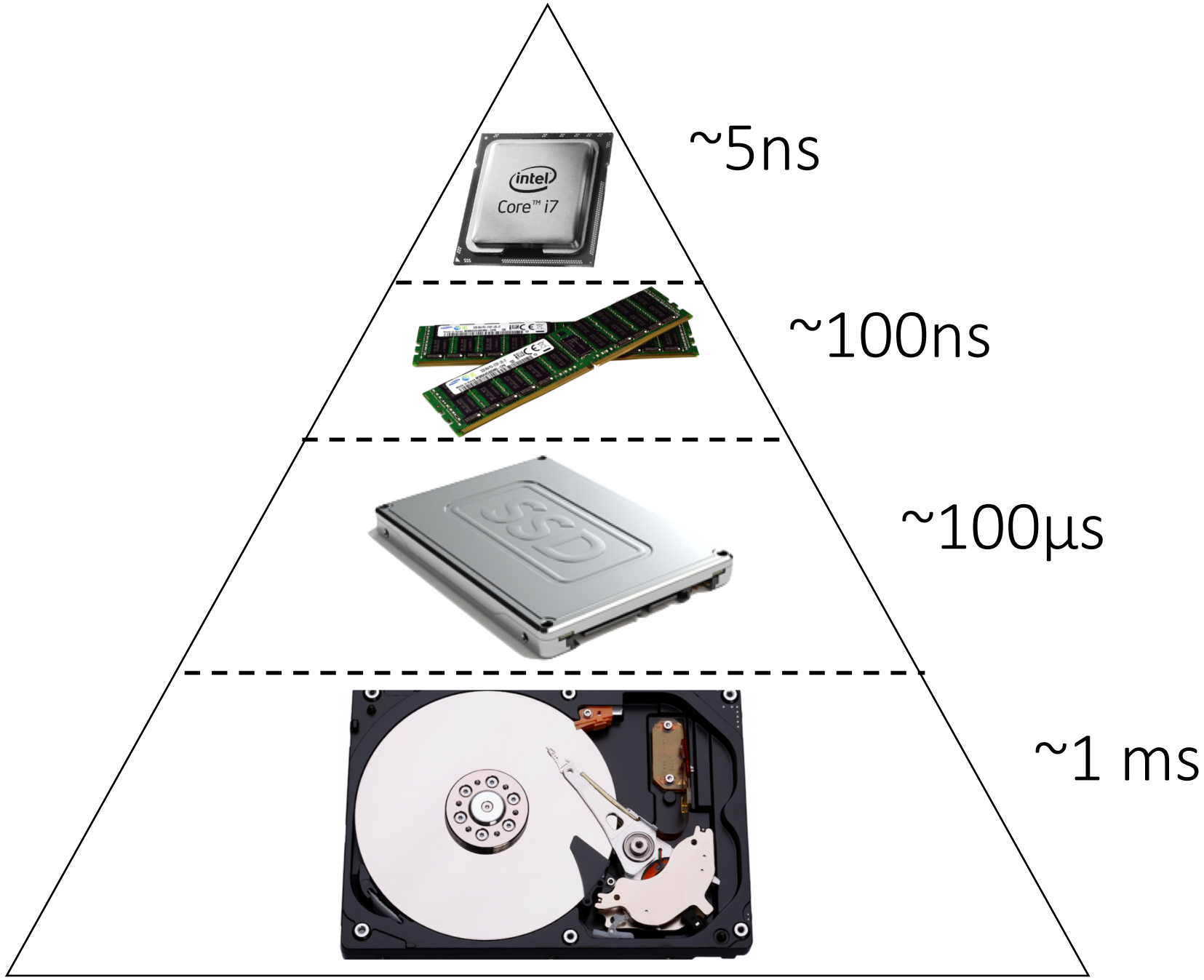

intel
Core™ i7
SSD

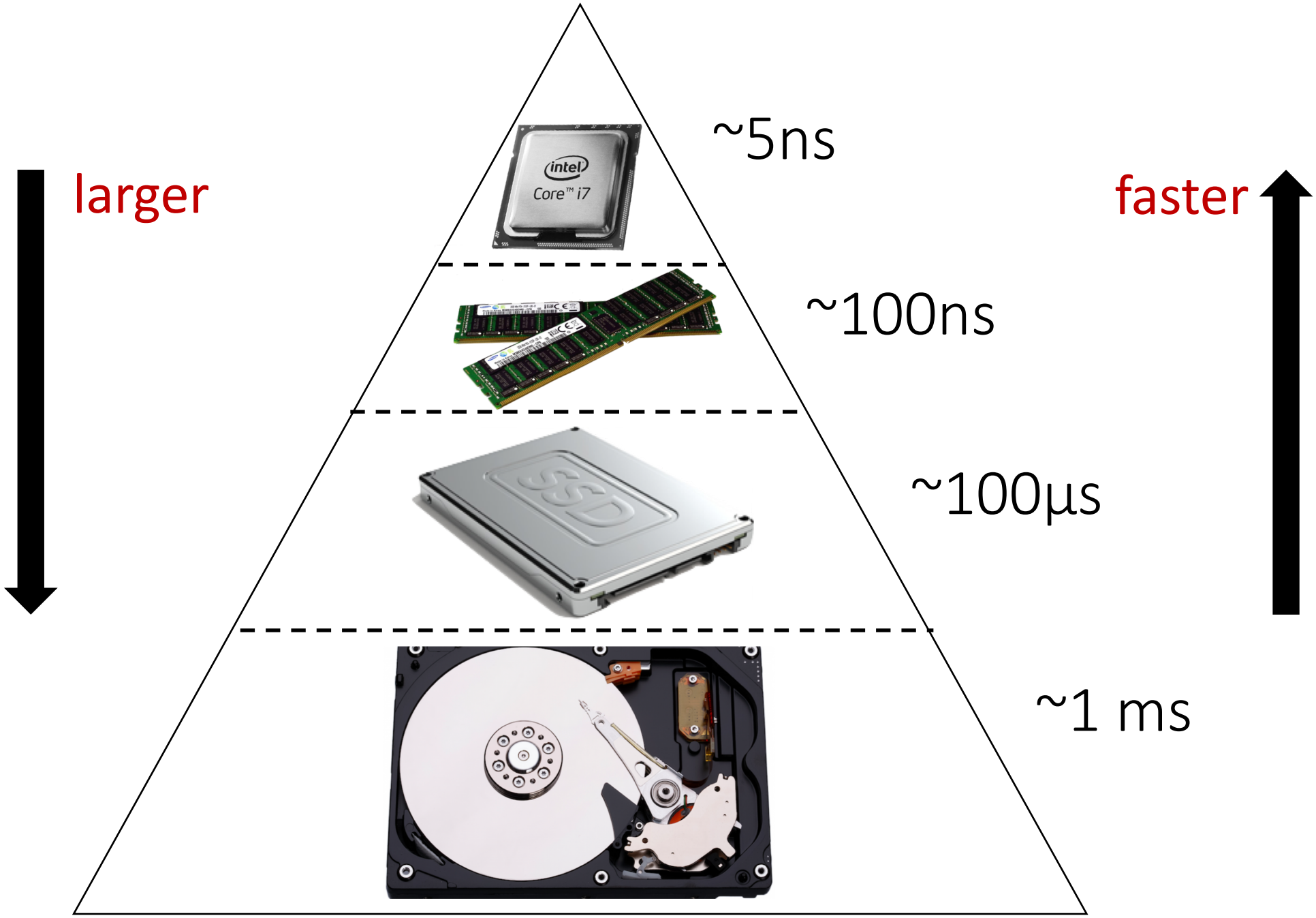

intel
Core™ i7



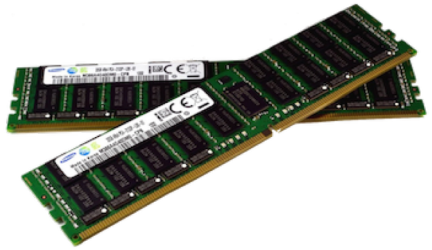

SSD




~5ns
~100ns
~100µs
~1 ms

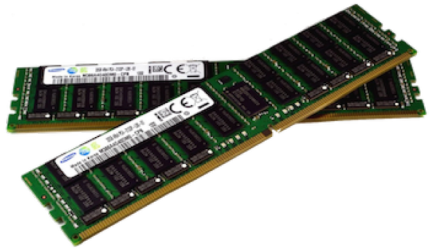

larger
faster
~5ns
~100ns
~100µs
~1 ms
intel
Core™ i7
SSD

Traditional I/O Model



Small, fast main memory
(size M)

Traditional I/O Model

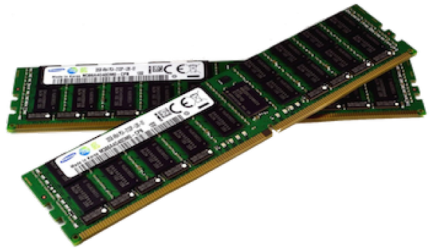


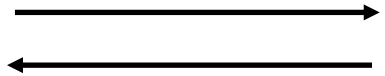
Small, fast main memory
(size M)

Large, slow external memory

Traditional I/O Model

One I/O at a time



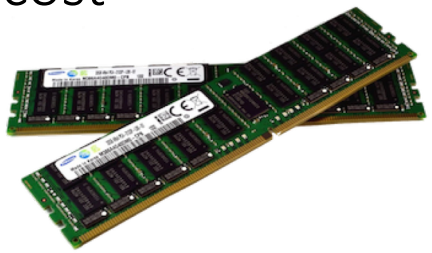


Small, fast main memory (size M)

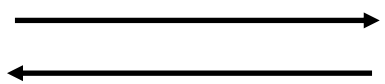
Large, slow external memory

Traditional I/O Model

0 access cost



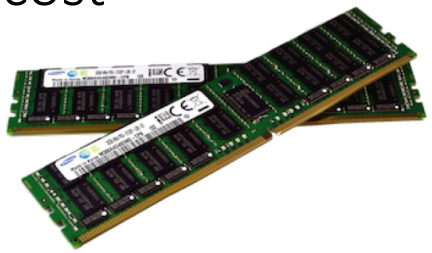
Small, fast main memory
(size M)



Large, slow external memory

Traditional I/O Model

0 access cost



Small, fast main memory
(size M)

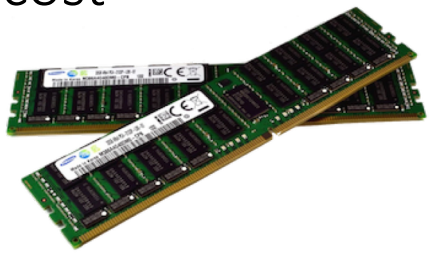
Transfer cost
1 unit

Large, slow external memory

Traditional I/O Model

total cost ≅ total # reads/writes to disk

0 access cost



Small, fast main memory
(size M)

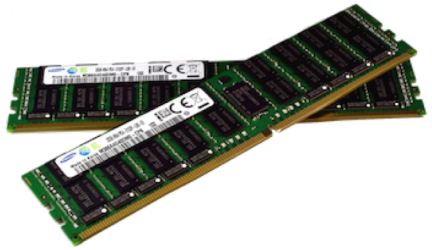
Transfer cost
1 unit

Large, slow external memory

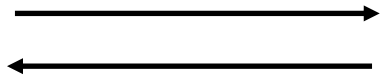
Traditional I/O Model

Two (outdated) assumptions

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



Small, fast main memory
(size M)

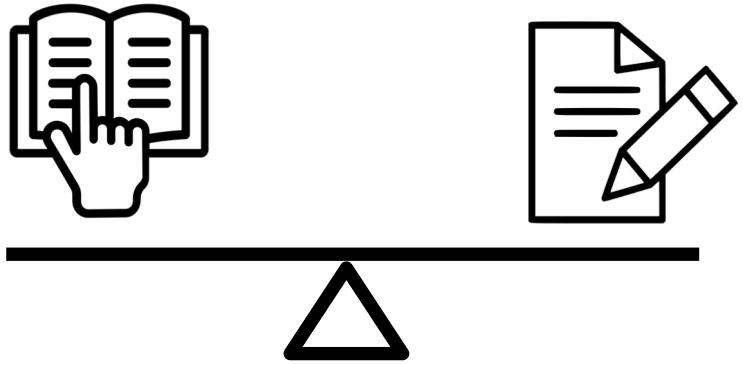


Large, slow external memory

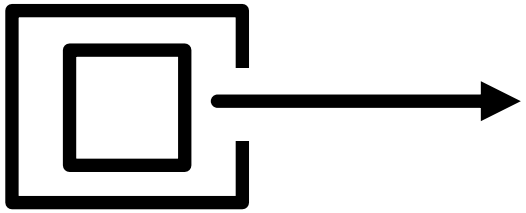
Hard Disk Drives

Hard Disk Drives

Two assumptions of the Traditional I/O Model



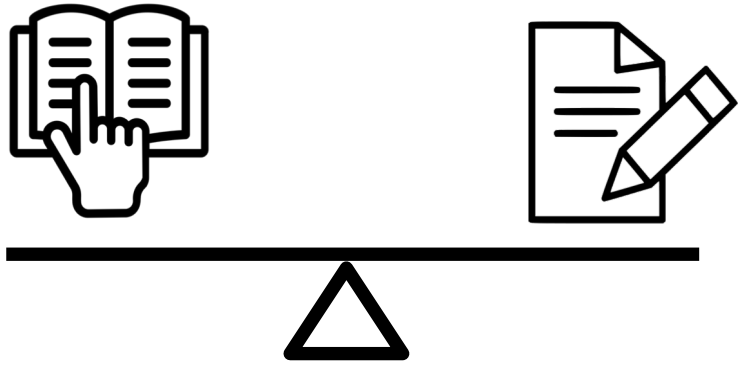
Symmetric cost for **Read & Write** to disk



One I/O at a time

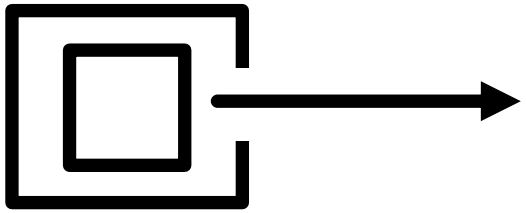
Hard Disk Drives

Two assumptions of the Traditional I/O Model



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

✓



One I/O at a time

✓

"Tape is Dead. Disk is Tape.

Flash is Disk."

- Jim Gray

“Tape is Dead. Disk is Tape. Flash is Disk.”

- Jim Gray

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

“Tape is Dead. Disk is Tape. Flash is Disk.”

- Jim Gray

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

“Tape is Dead. Disk is Tape. Flash is Disk.”

- Jim Gray

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 Disk (SSD)

No mechanical movement

Solid-State Disk (SSD)

No mechanical movement

Fast access, High chip density, Low energy consumption

Solid-State Disk (SSD)

- ✓ SATA SSD

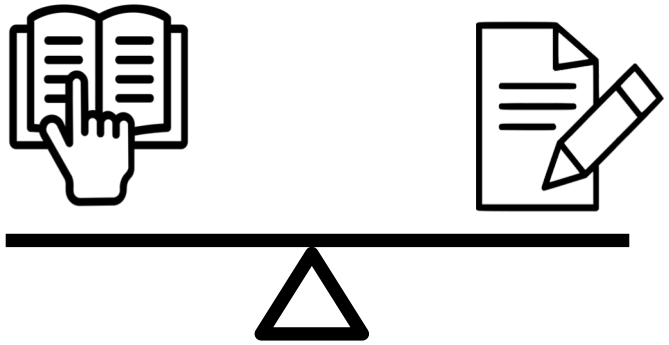
Solid-State Disk (SSD)

- ✓ SATA SSD
- ✓ PCIe SSD

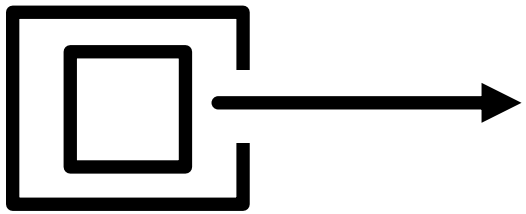
Solid-State Disk (SSD)

- ✓ SATA SSD
- ✓ PCIe SSD
- ✓ Optane SSD

Modern Storage Devices

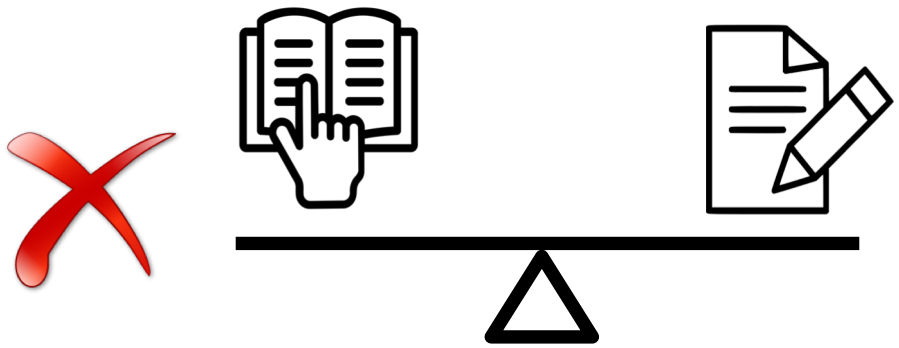


Symmetric cost for Read & Write

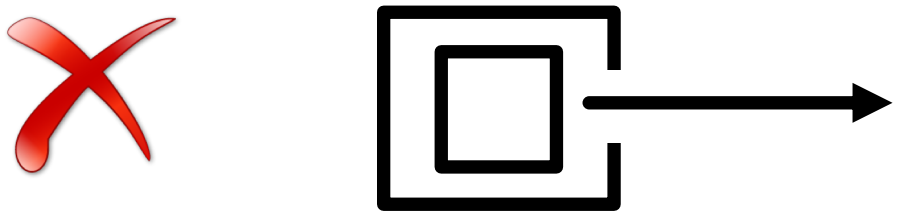


One I/O at a time

Modern Storage Devices

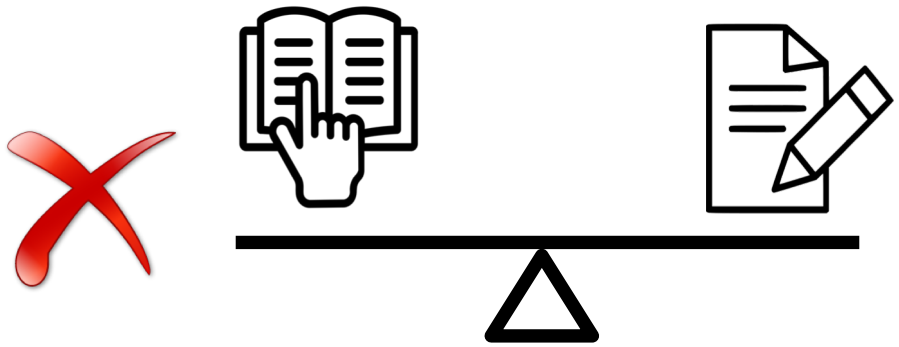


Symmetric cost for Read & Write

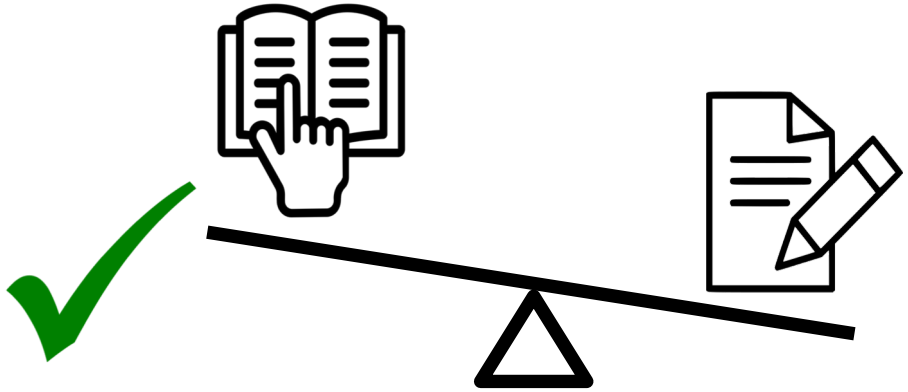


One I/O at a time

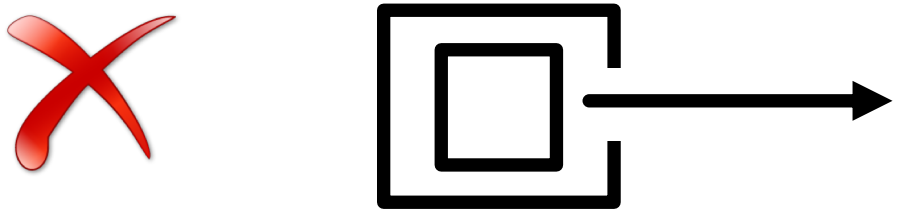
Modern Storage Devices



Symmetric cost for Read & Write

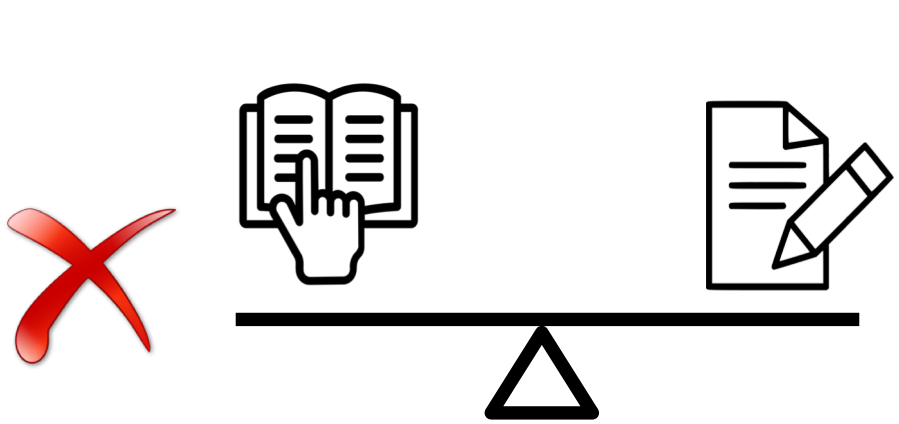


Read/Write Asymmetry

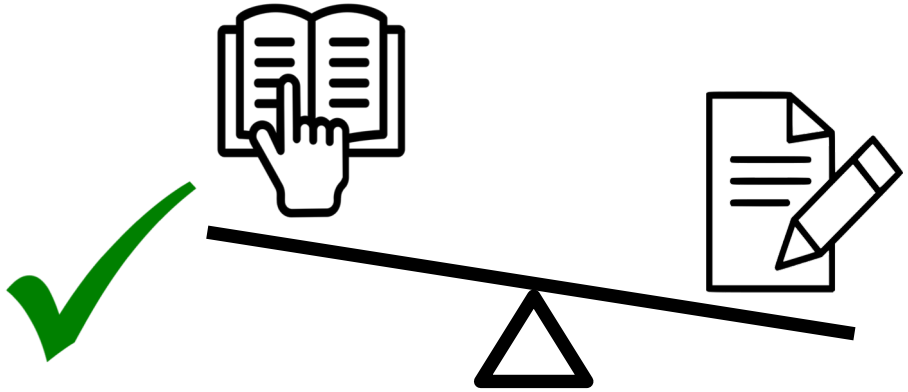


One I/O at a time

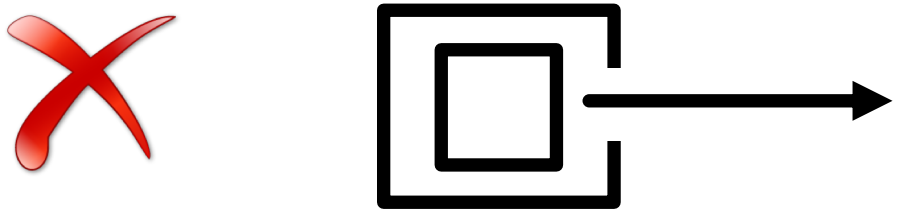
Modern Storage Devices



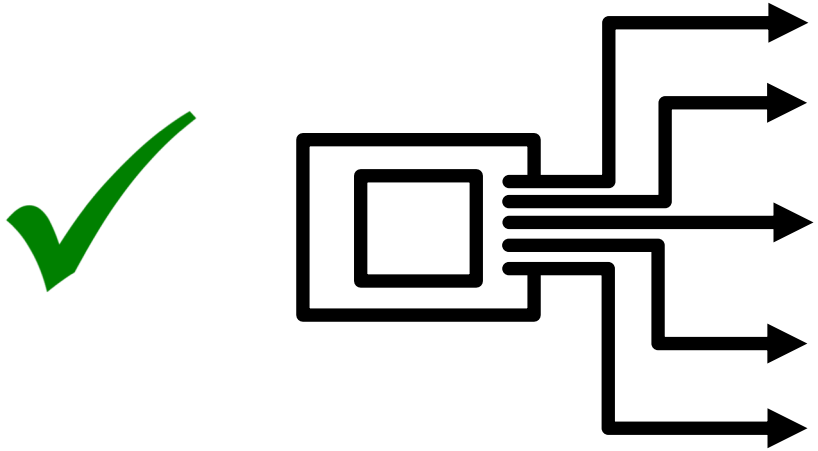
Symmetric cost for Read & Write



Read/Write Asymmetry



One I/O at a time



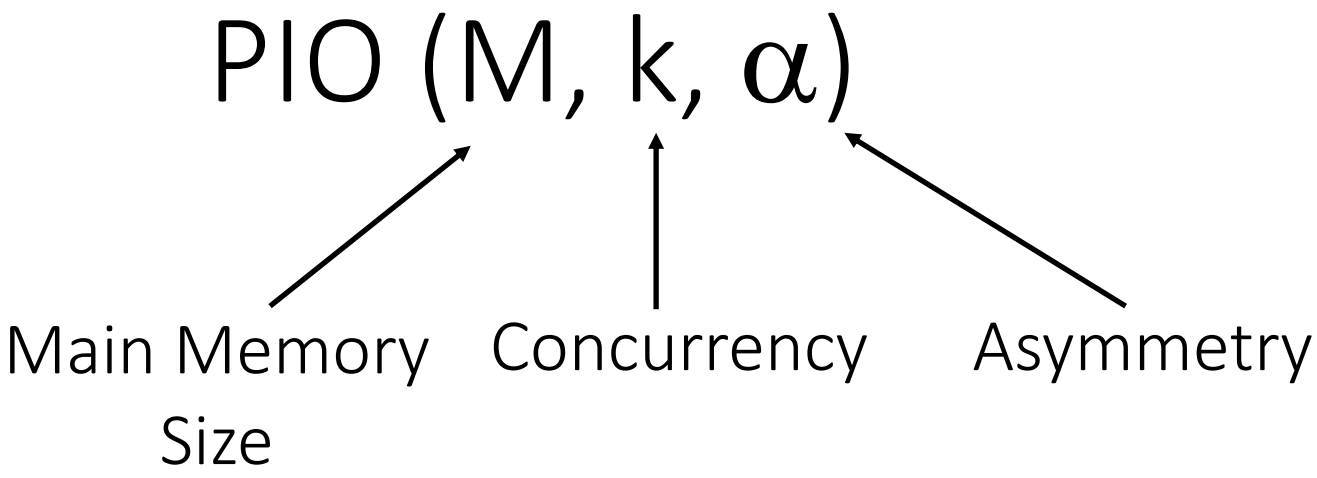
Concurrency

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

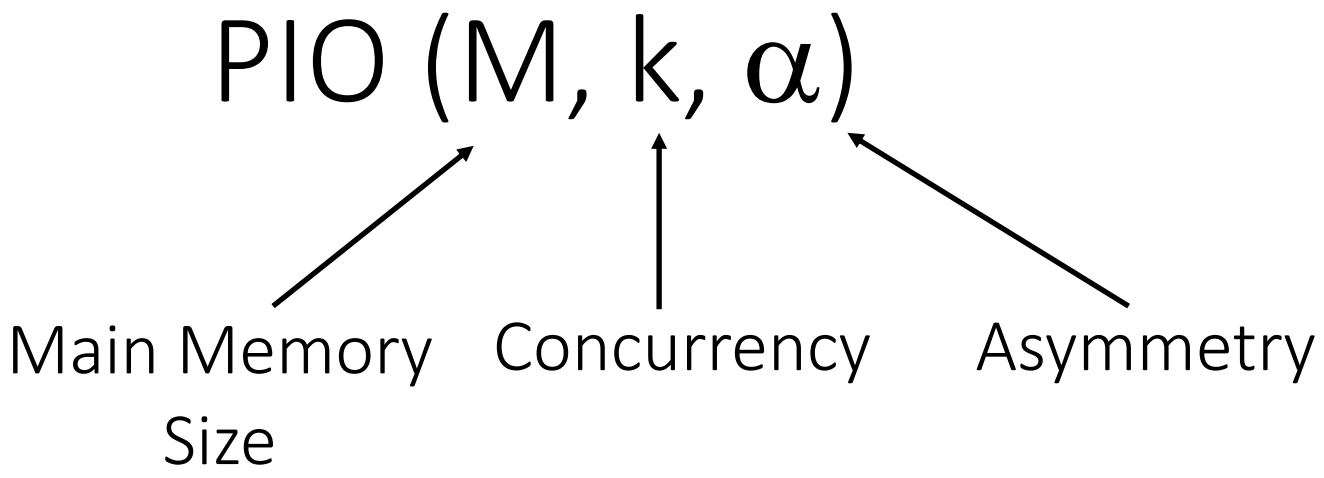
read/write asymmetry and concurrency?

Parametric I/O Model

Parametric I/O Model

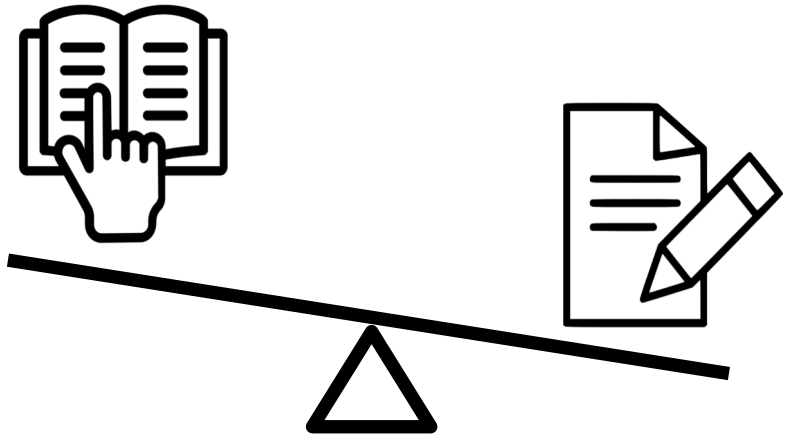


Parametric I/O Model



PIO*(M, k_r, k_w, α)** *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).

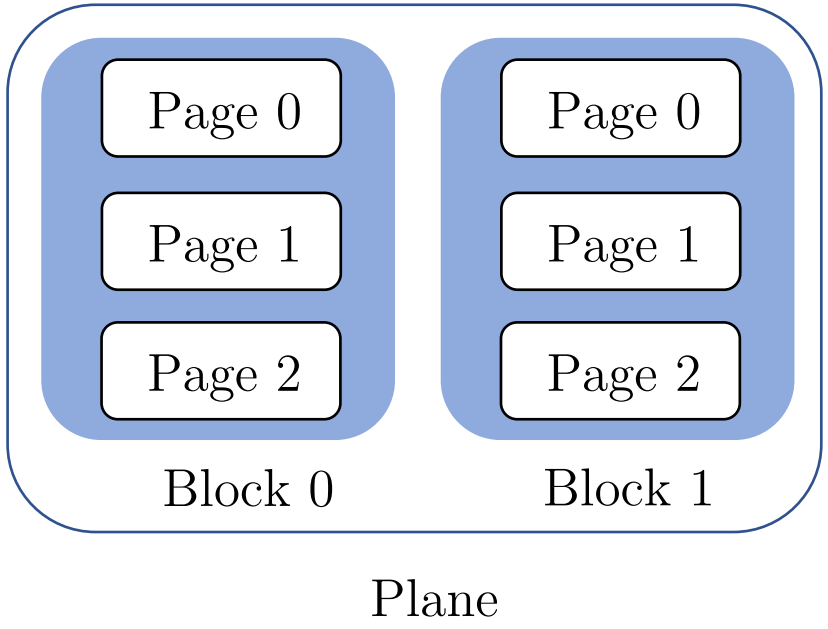
Read/Write Asymmetry



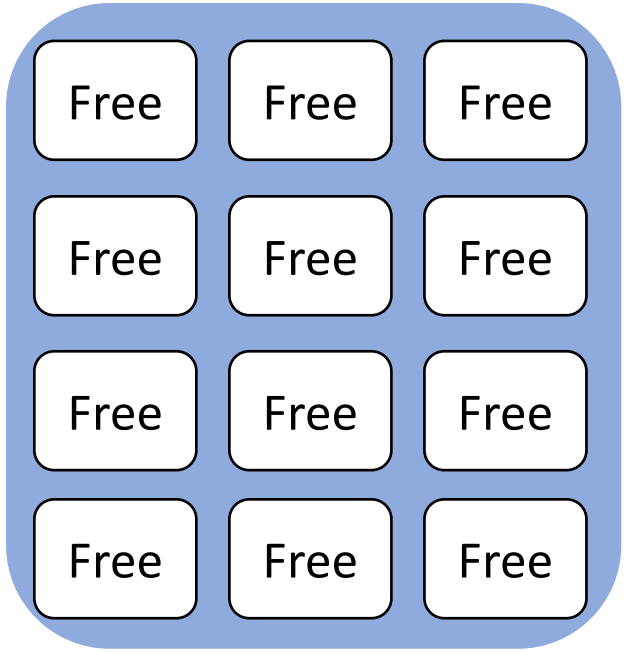
Writes in SSD

Out-of-place updates cause invalidation

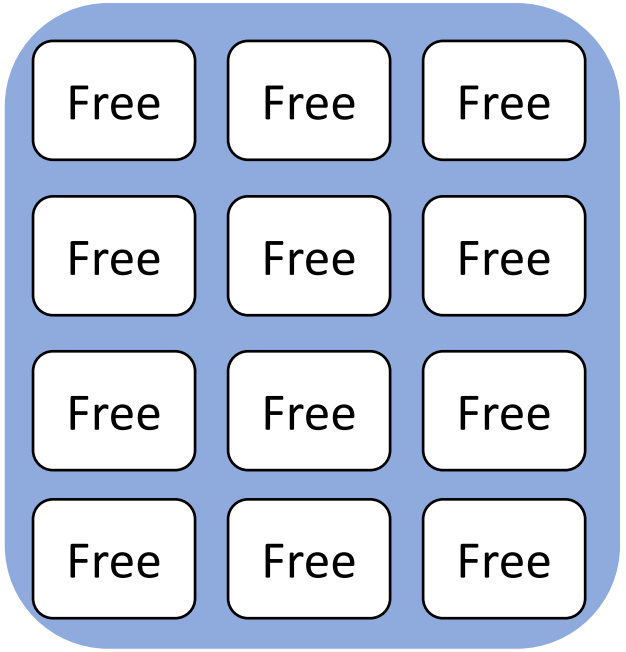
Invalidation causes garbage collection.



Writes in SSD

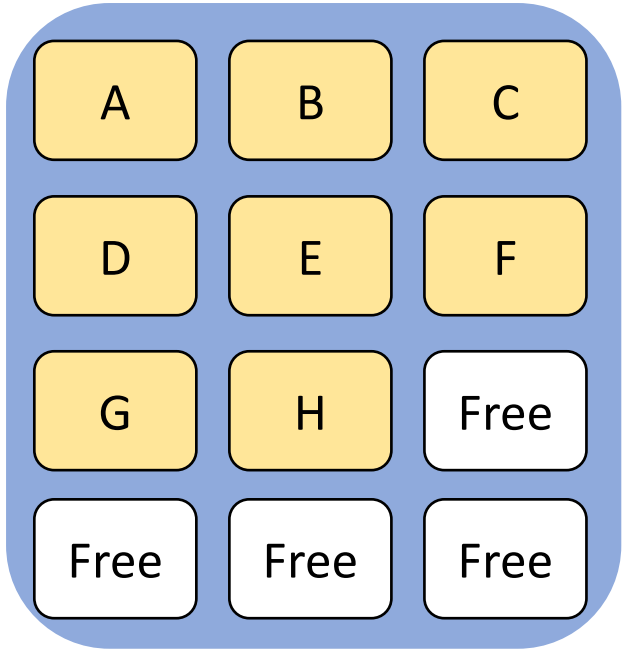


Block 0

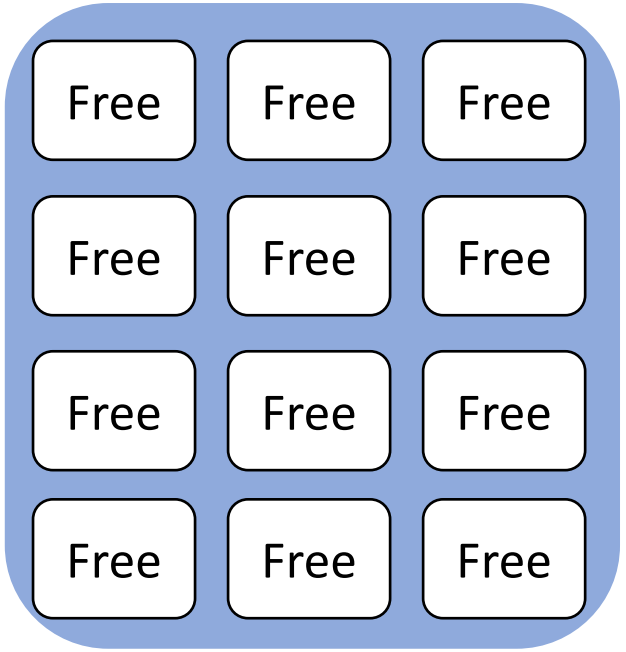


Block 1

Writes in SSD



Block 0



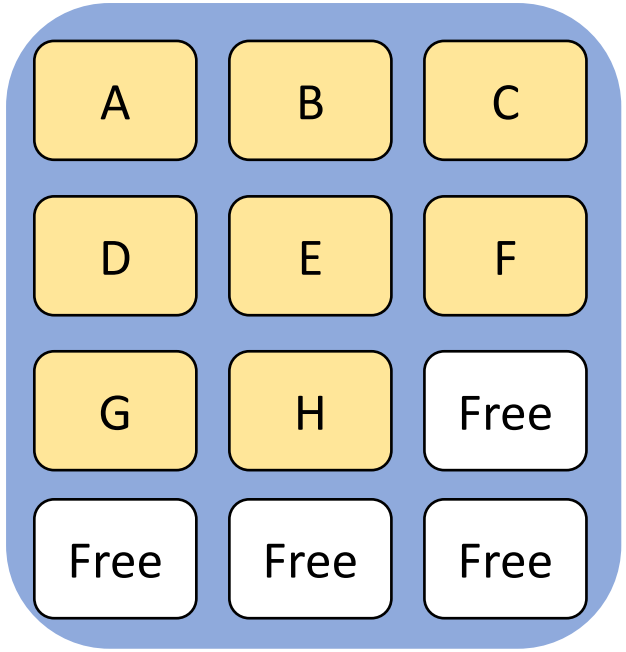
Block 1

Writing in a free page isn't costly!

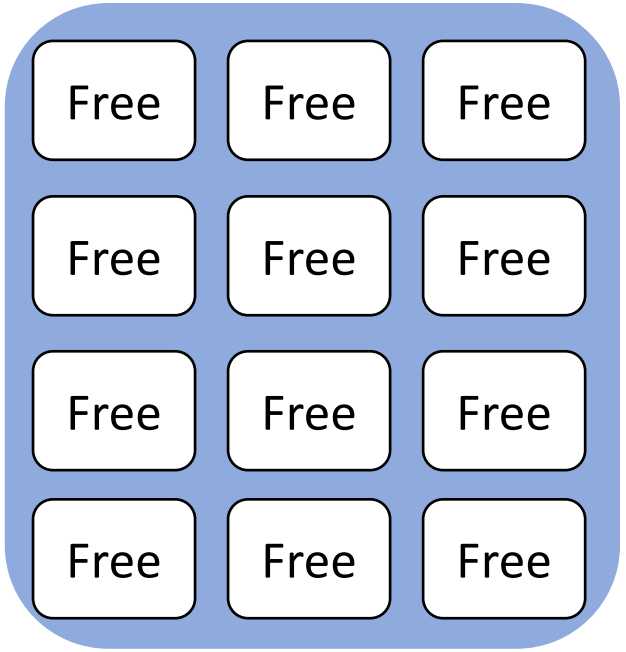
Writes in SSD

Update

A, B, C, D



Block 0

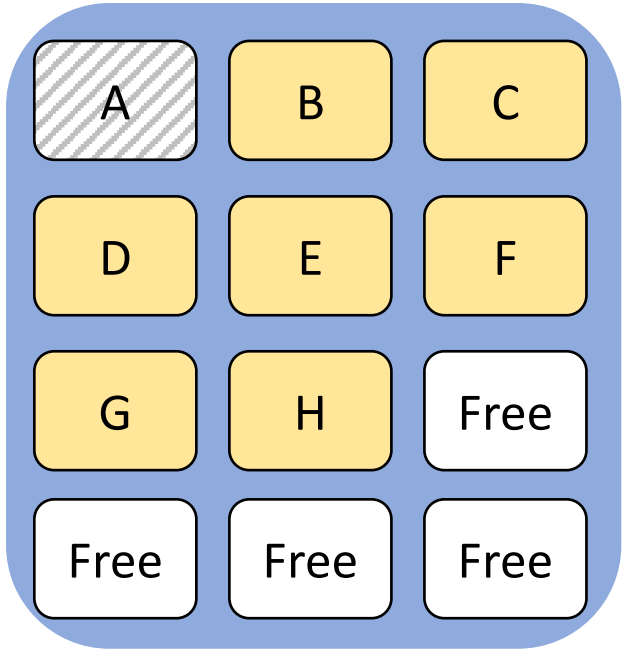


Block 1

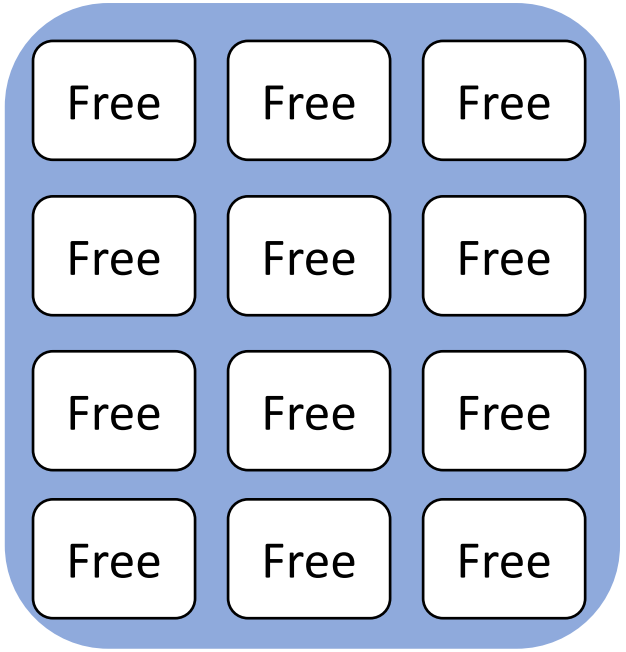
Writes in SSD

Update

A, B, C, D



Block 0

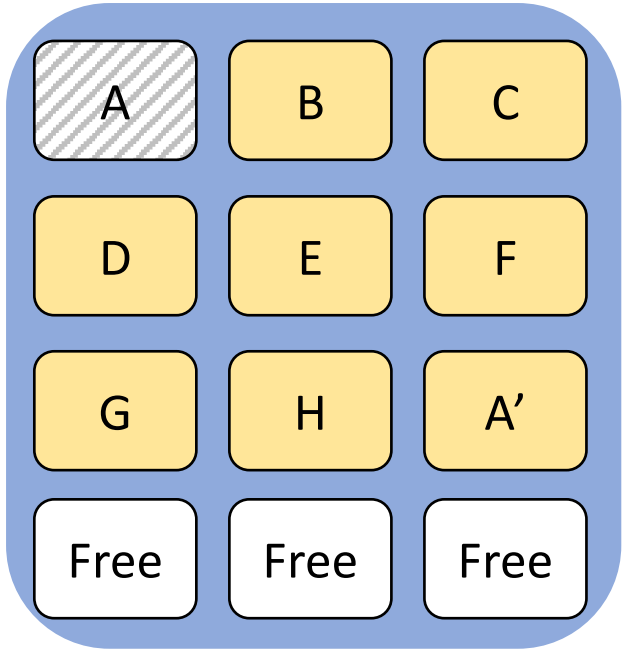


Block 1

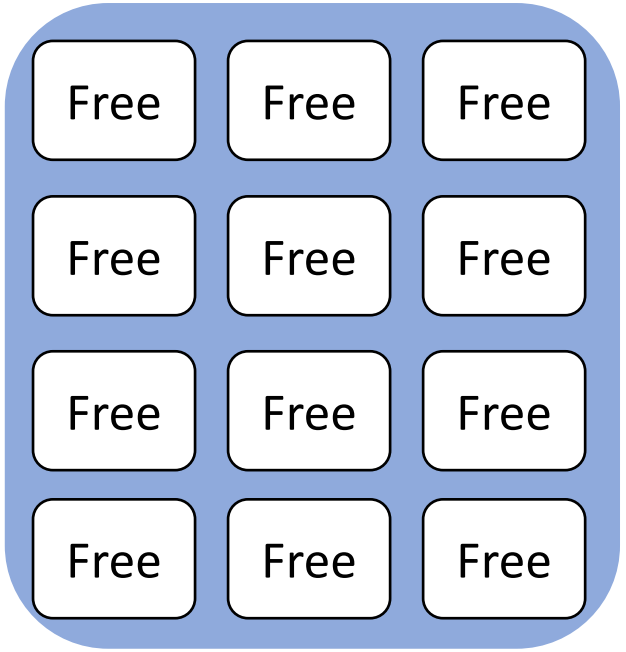
Writes in SSD

Update

A, B, C, D



Block 0

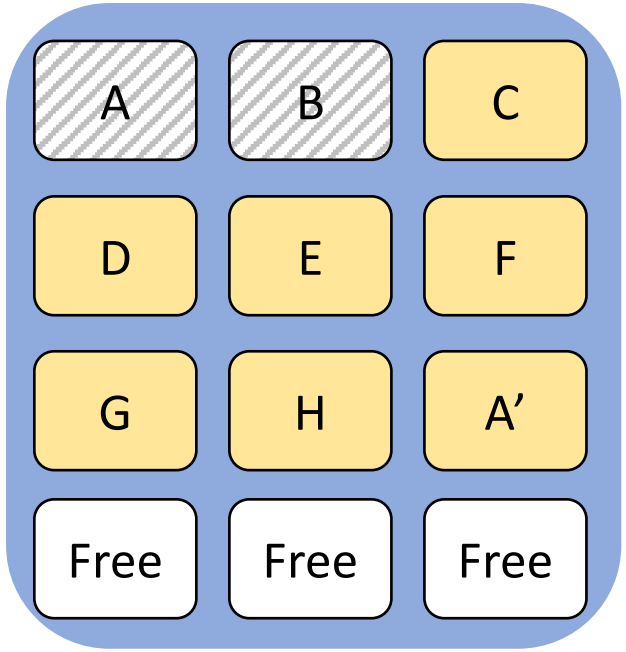


Block 1

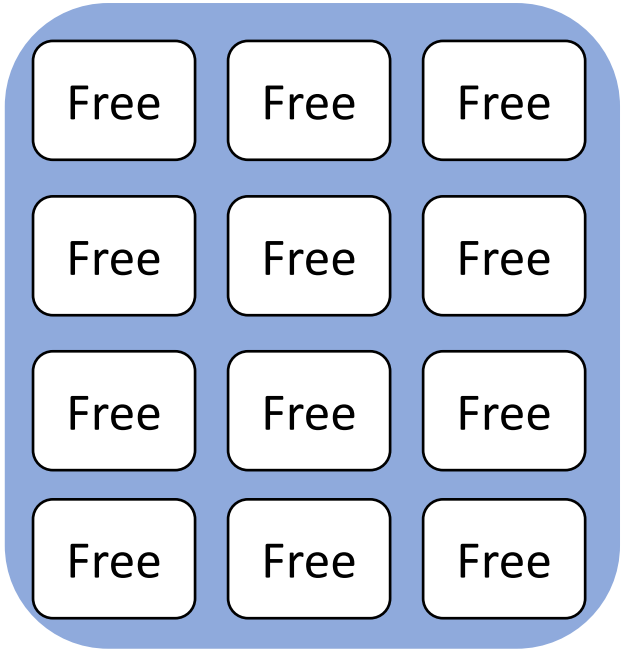
Writes in SSD

Update

A, **B**, C, D



Block 0

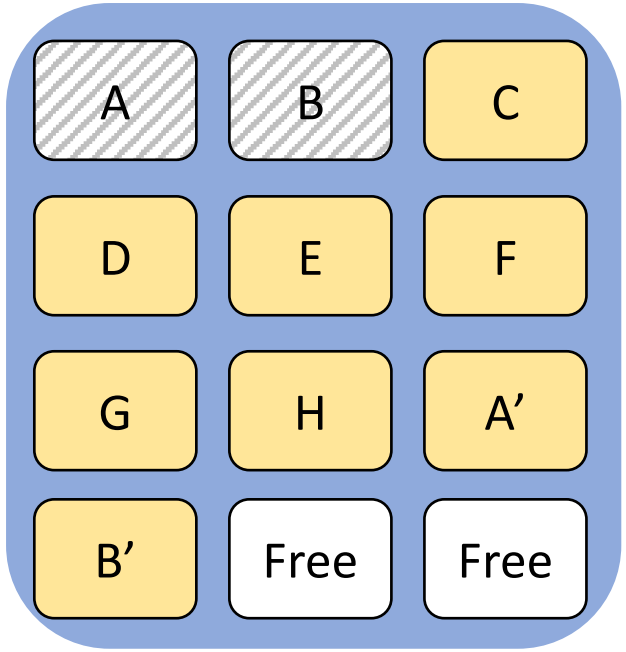


Block 1

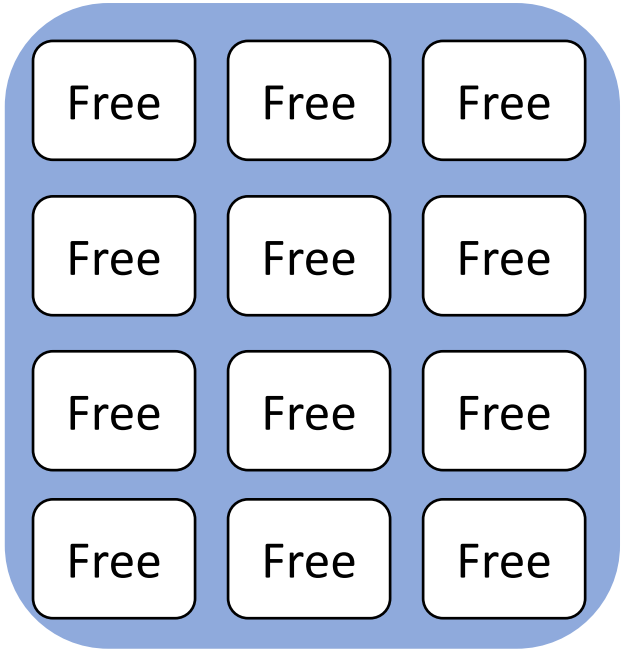
Writes in SSD

Update

A, **B**, C, D



Block 0

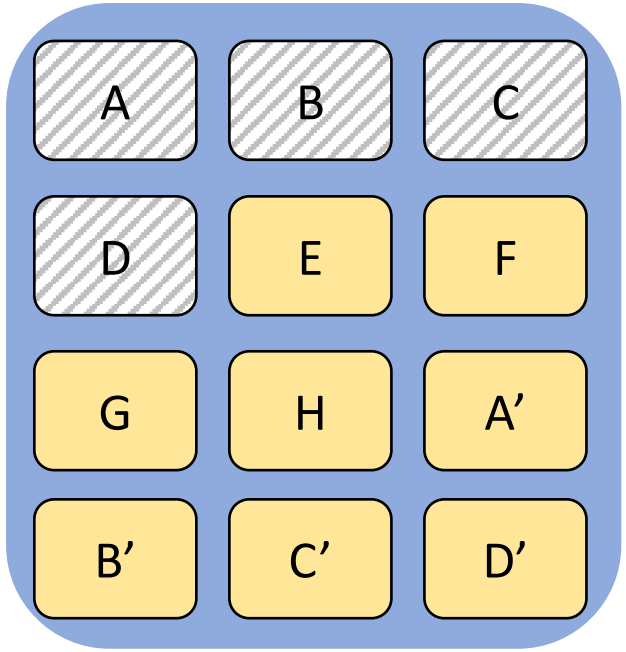


Block 1

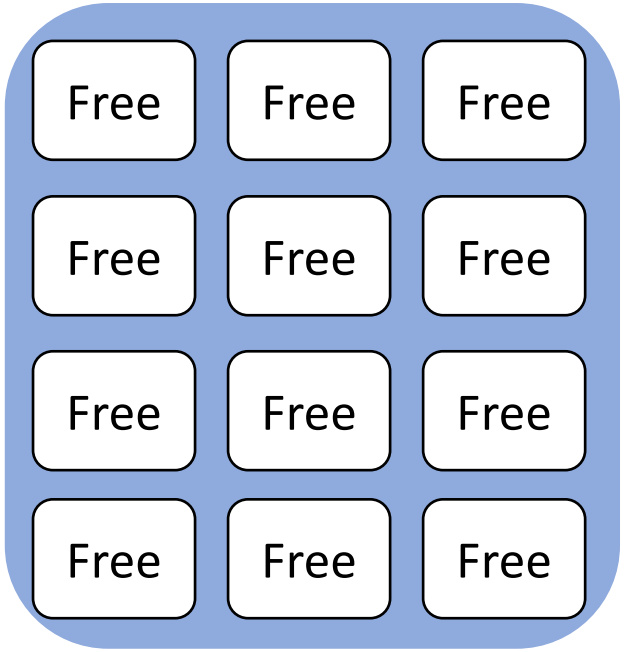
Writes in SSD

Update

A, B, C, D



Block 0

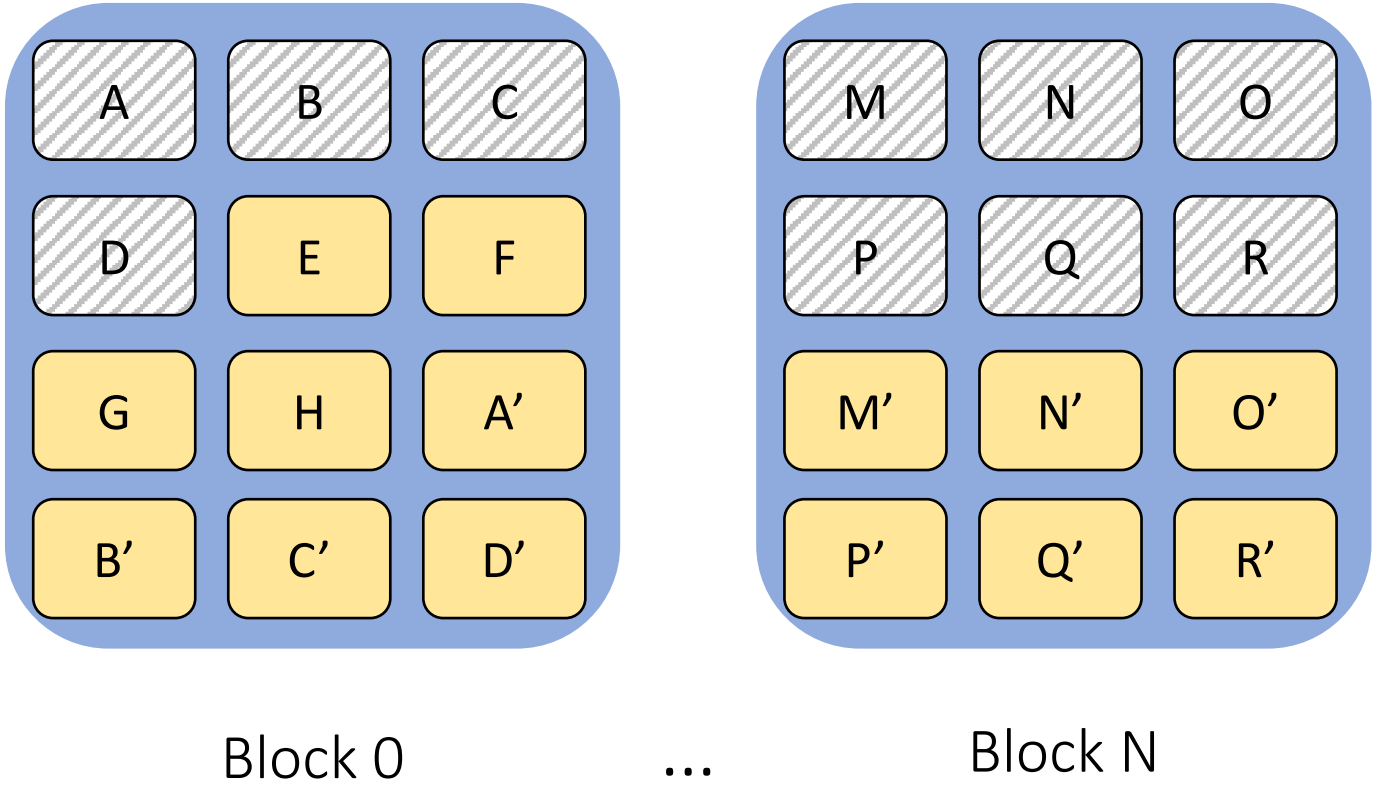


Block 1

Not all updates are costly!

Writes in SSD

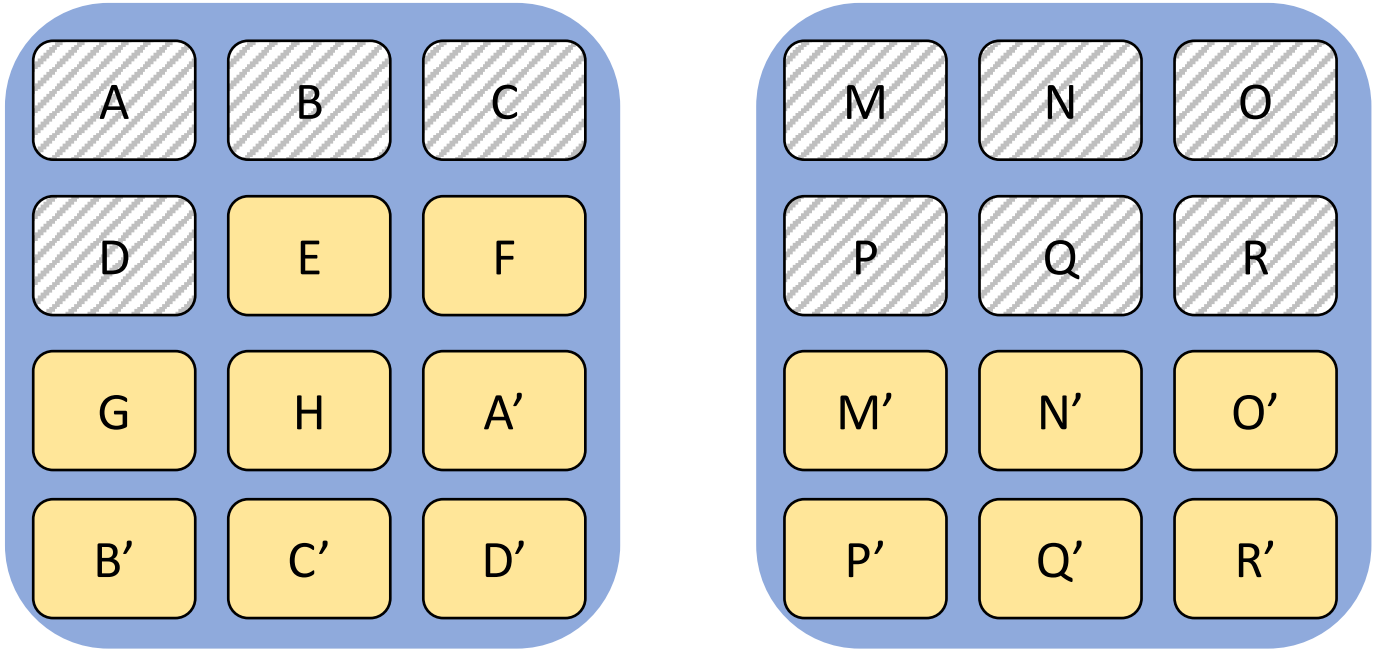
What if there is no space?



Writes in SSD

What if there is no space?

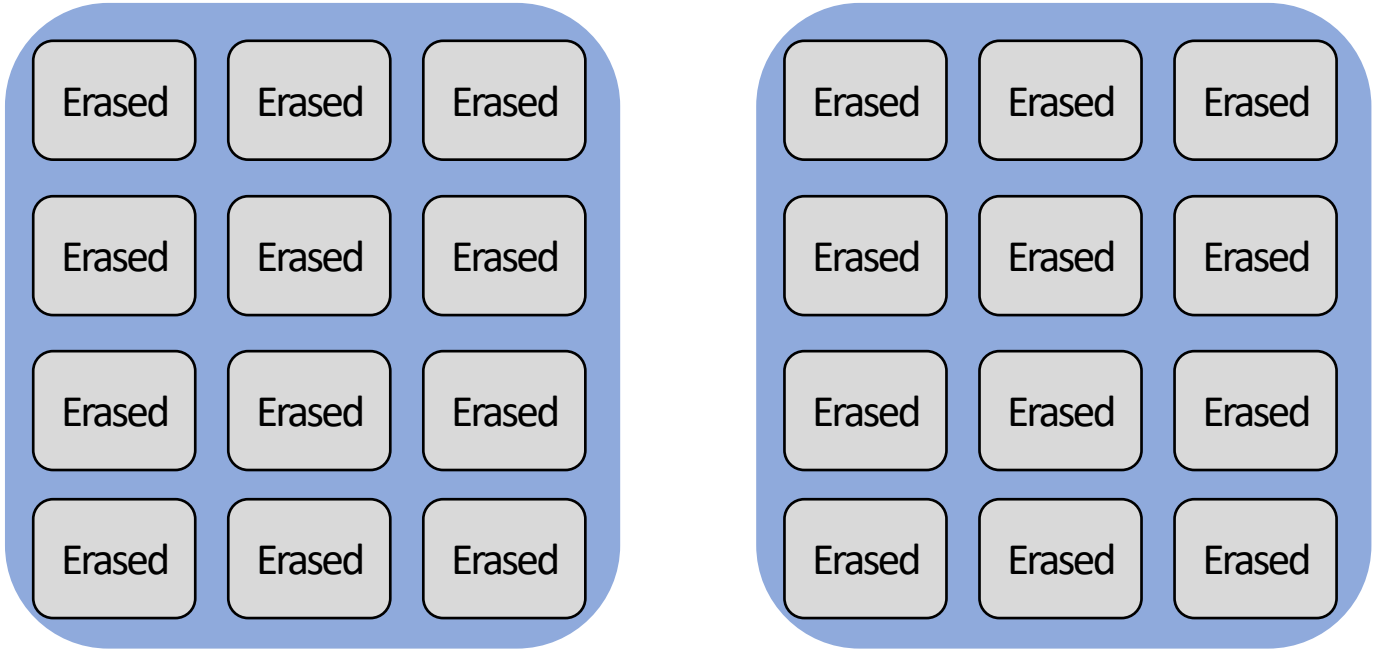
Garbage Collection!



Writes in SSD

What if there is no space?

Garbage Collection!



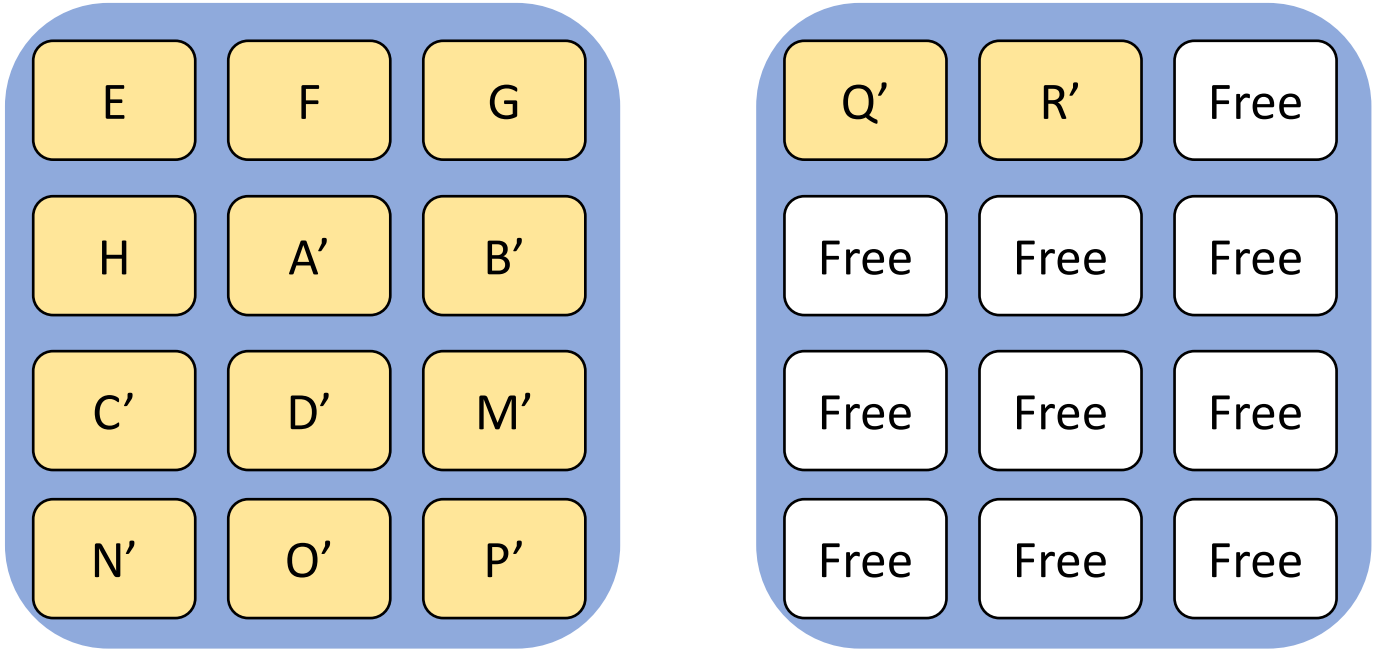
Valid pages:

E	F	G	H	A’	B’	C’	D’	M’	N’	O’	P’	Q’	R’

Writes in SSD

What if there is no space?

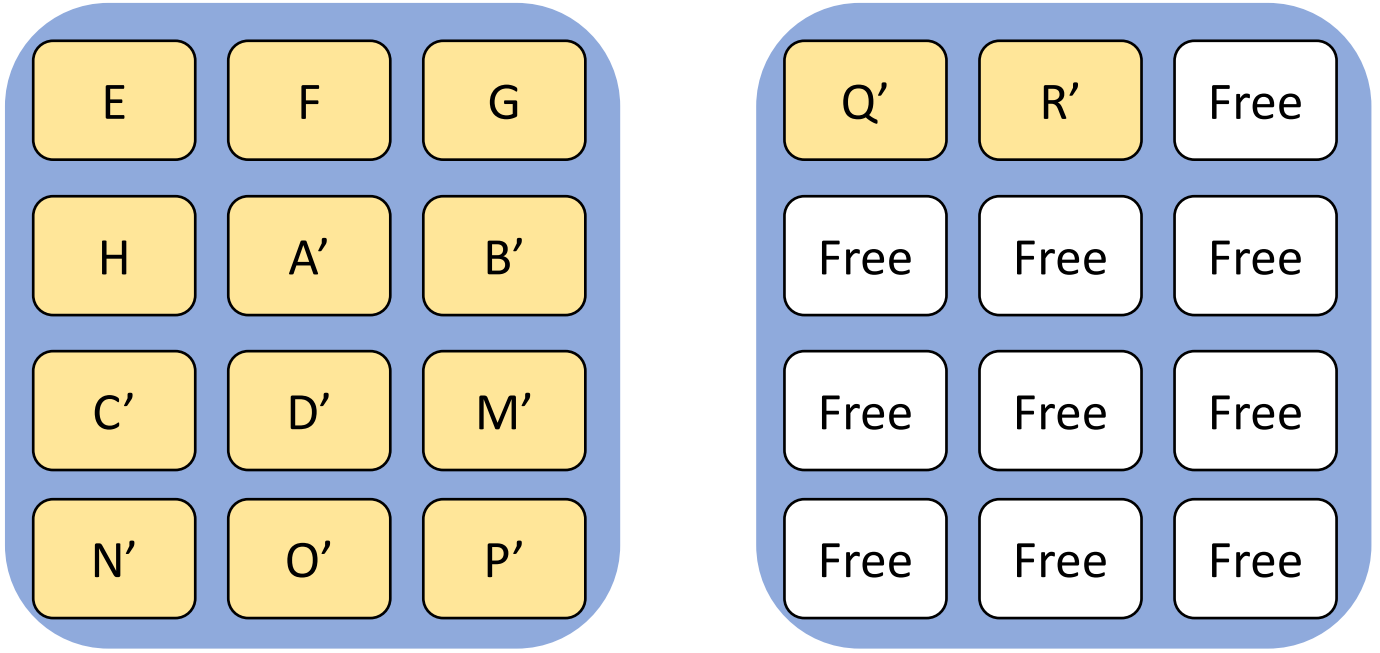
Garbage Collection!



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

Device	Advertised Rand Read IOPS	Advertised Rand Write IOPS	Advertised Asymmetry
PCIe D5-P4320	427k	36k	11.9
PCIe DC-P4500	626k	51k	12.3
PCIe P4510	465k	145k	3.2
SATA D3-S4610	92k	28k	3.3
Optane P4800X	550k	500k	1.1

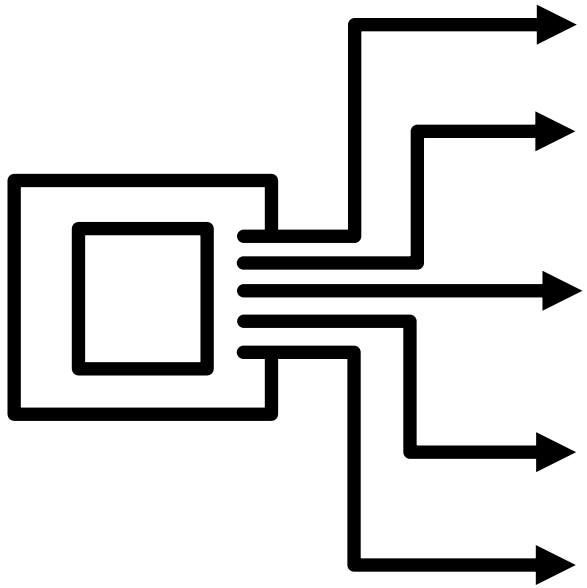
Read/Write Asymmetry - Example

Device	Advertised Rand Read IOPS	Advertised Rand Write IOPS	Advertised Asymmetry
PCIe D5-P4320	427k	36k	11.9
PCIe DC-P4500	626k	51k	12.3
PCIe P4510	465k	145k	3.2
SATA D3-S4610	92k	28k	3.3
Optane P4800X	550k	500k	1.1

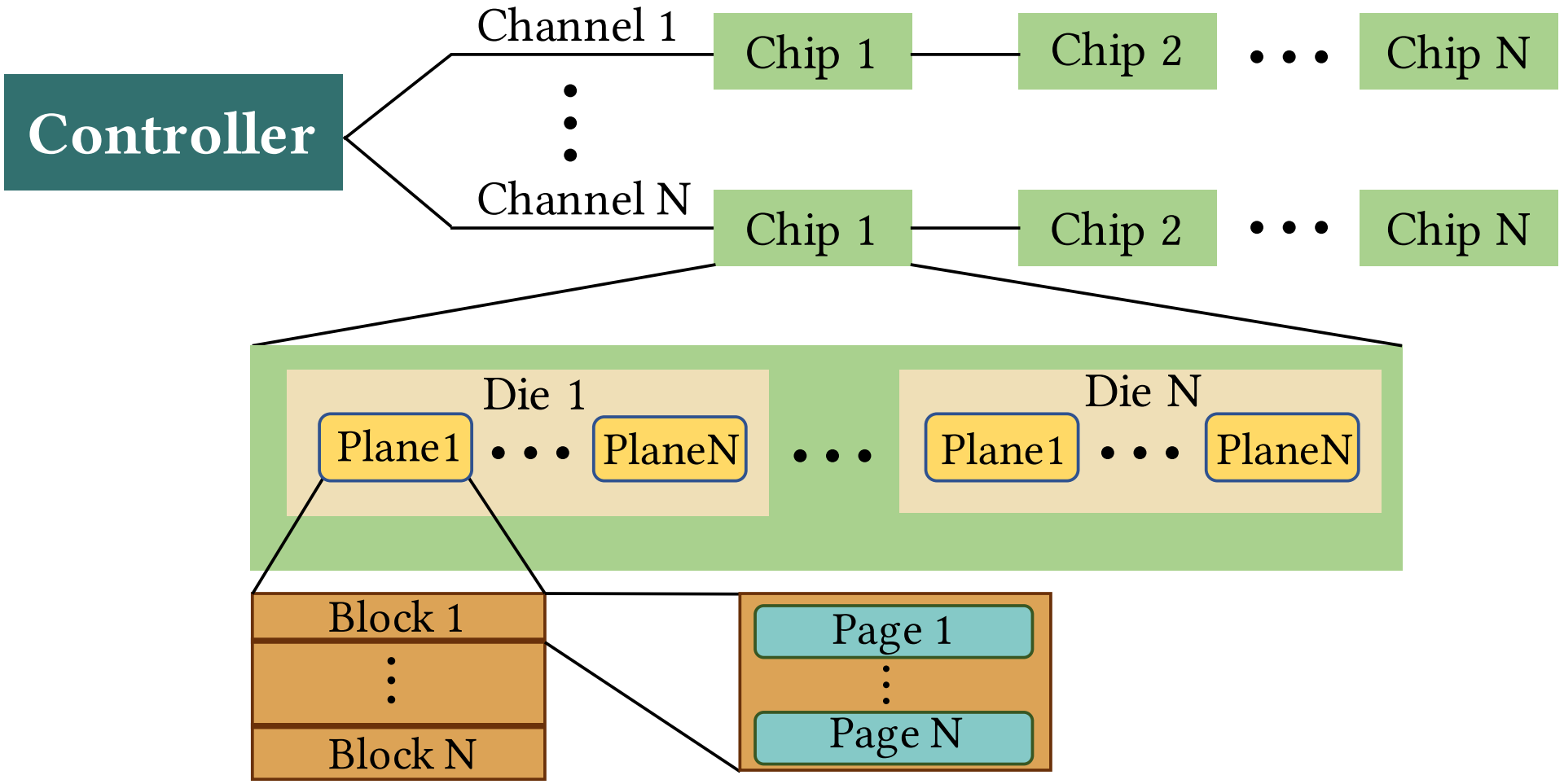
Read/Write Asymmetry - Example

Device	Advertised Rand Read IOPS	Advertised Rand Write IOPS	Advertised Asymmetry
PCIe D5-P4320	427k	36k	11.9
PCIe DC-P4500	626k	51k	12.3
PCIe P4510	465k	145k	3.2
SATA D3-S4610	92k	28k	3.3
Optane P4800X	550k	500k	1.1

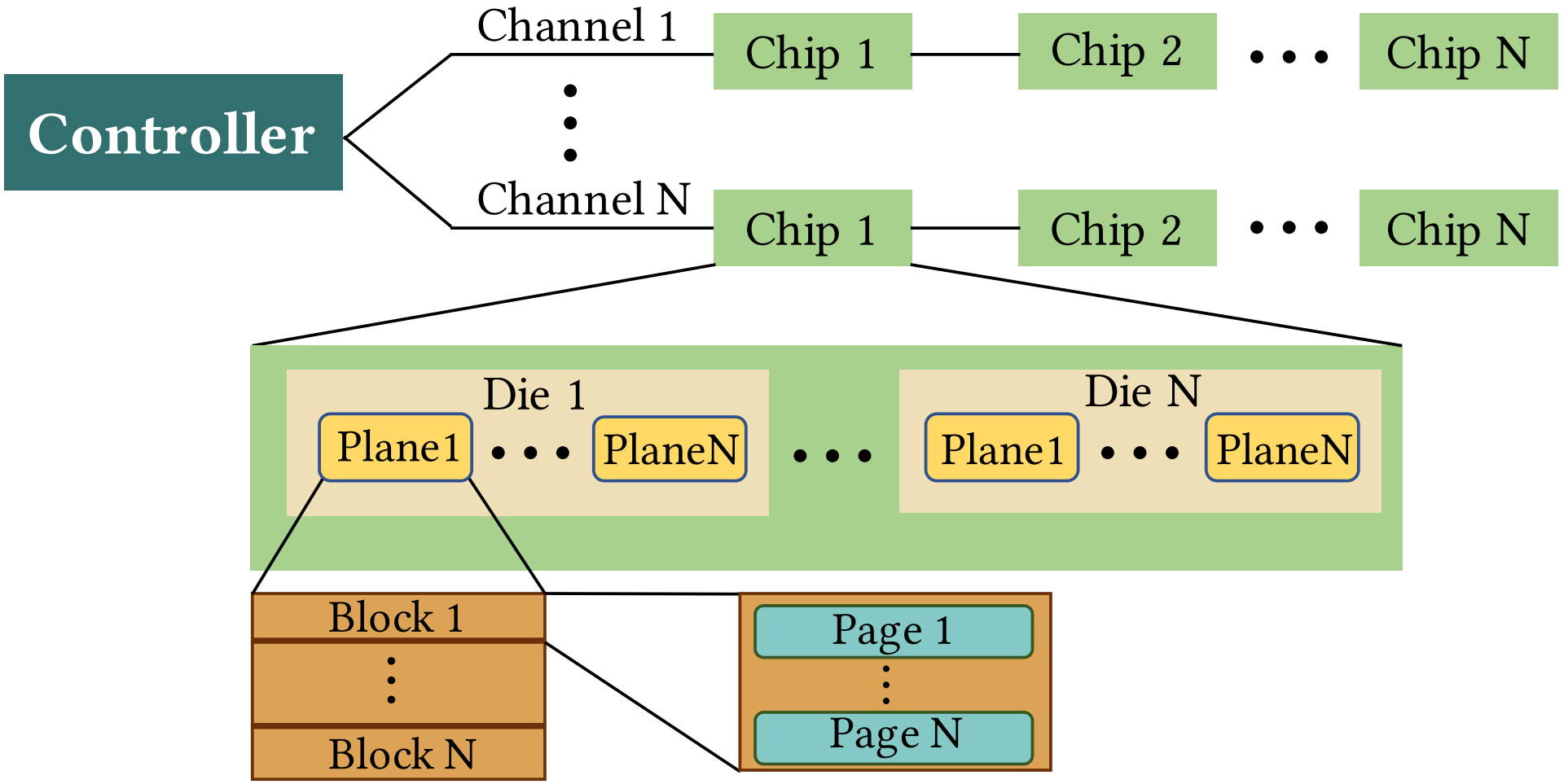
Concurrency



Internals of an SSD

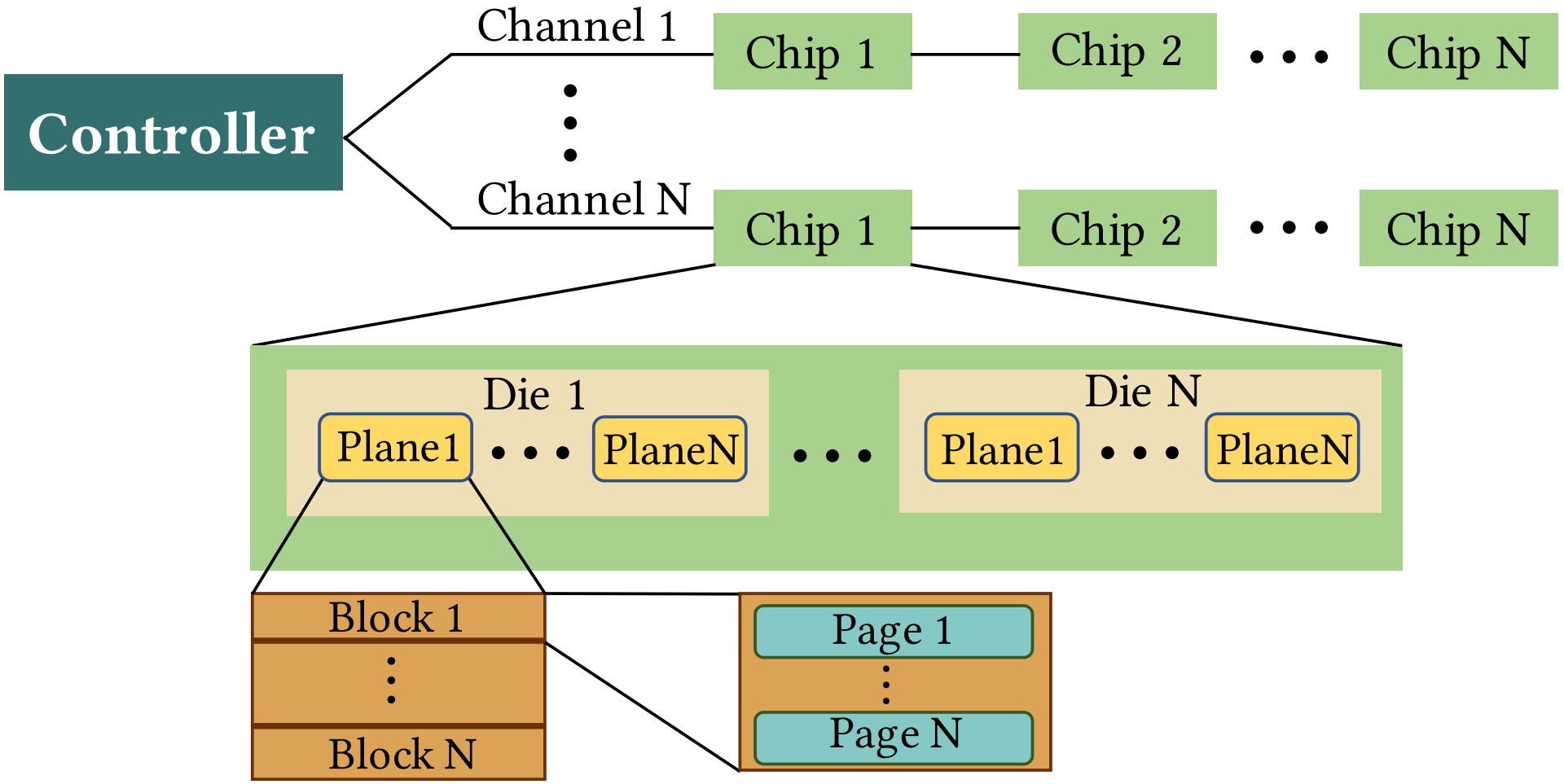


Internals of an SSD



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

Internals of an SSD



Essential to know the concurrency of a device!

How can we quantify

read/write asymmetry and concurrency?

Benchmarking

Benchmarking

Tools

- ✓ Custom micro-benchmarking
- ✓ Fio
- ✓ Intel's SPDK

Benchmarking

Tools

- ✓ Custom micro-benchmarking
- ✓ Fio
- ✓ Intel's SPDK

Setup

with and **without** FS

Benchmarking

Devices

- ✓ Optane SSD
- ✓ PCIe SSD **with** & **w/o** FS
- ✓ SATA SSD

Benchmarking

Devices

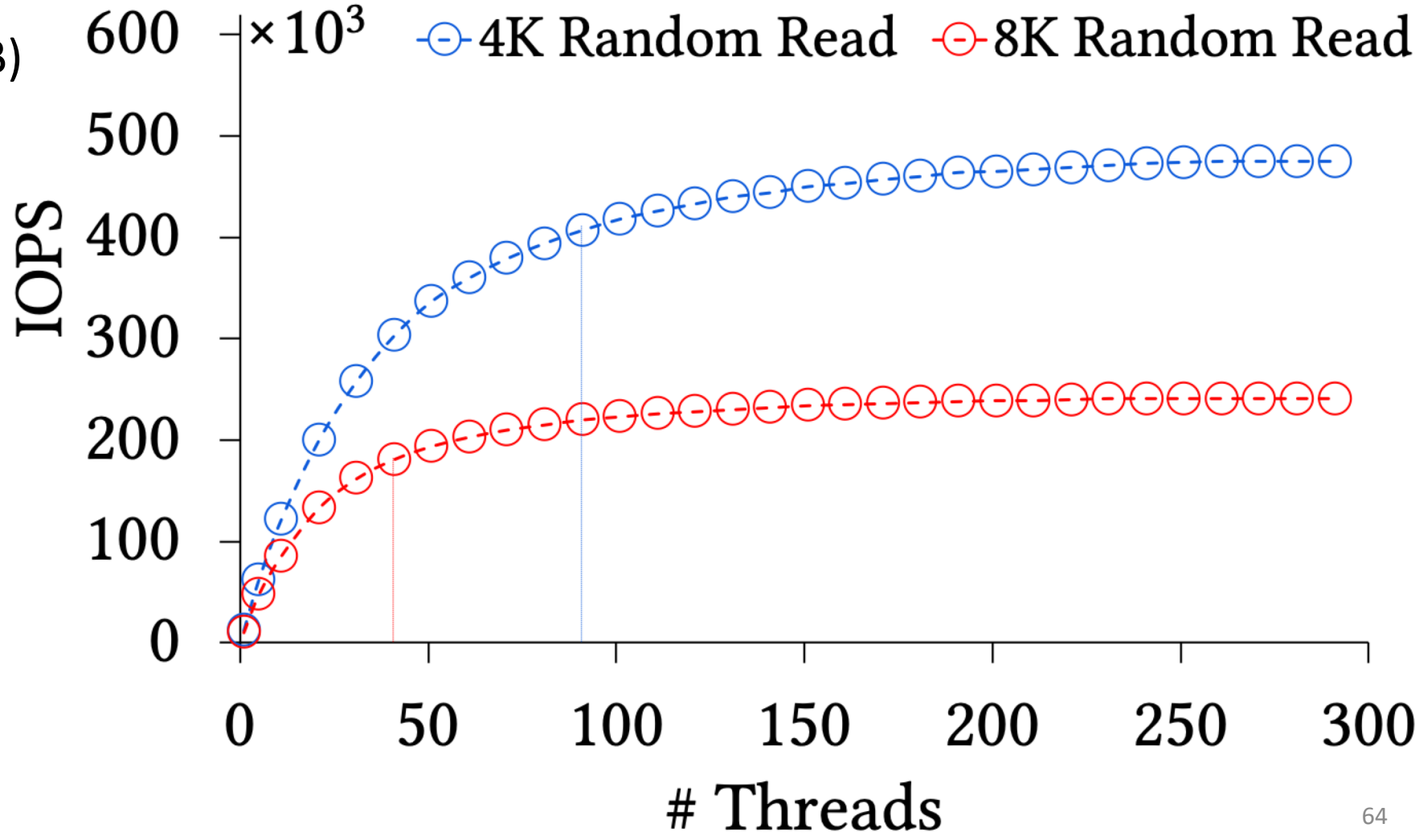
- ✓ Optane SSD
- ✓ PCIe SSD **with** & **w/o** FS
- ✓ SATA SSD

Setup

- ✓ Increase **#threads** issuing I/Os
- ✓ Report: **latency** and **throughput**

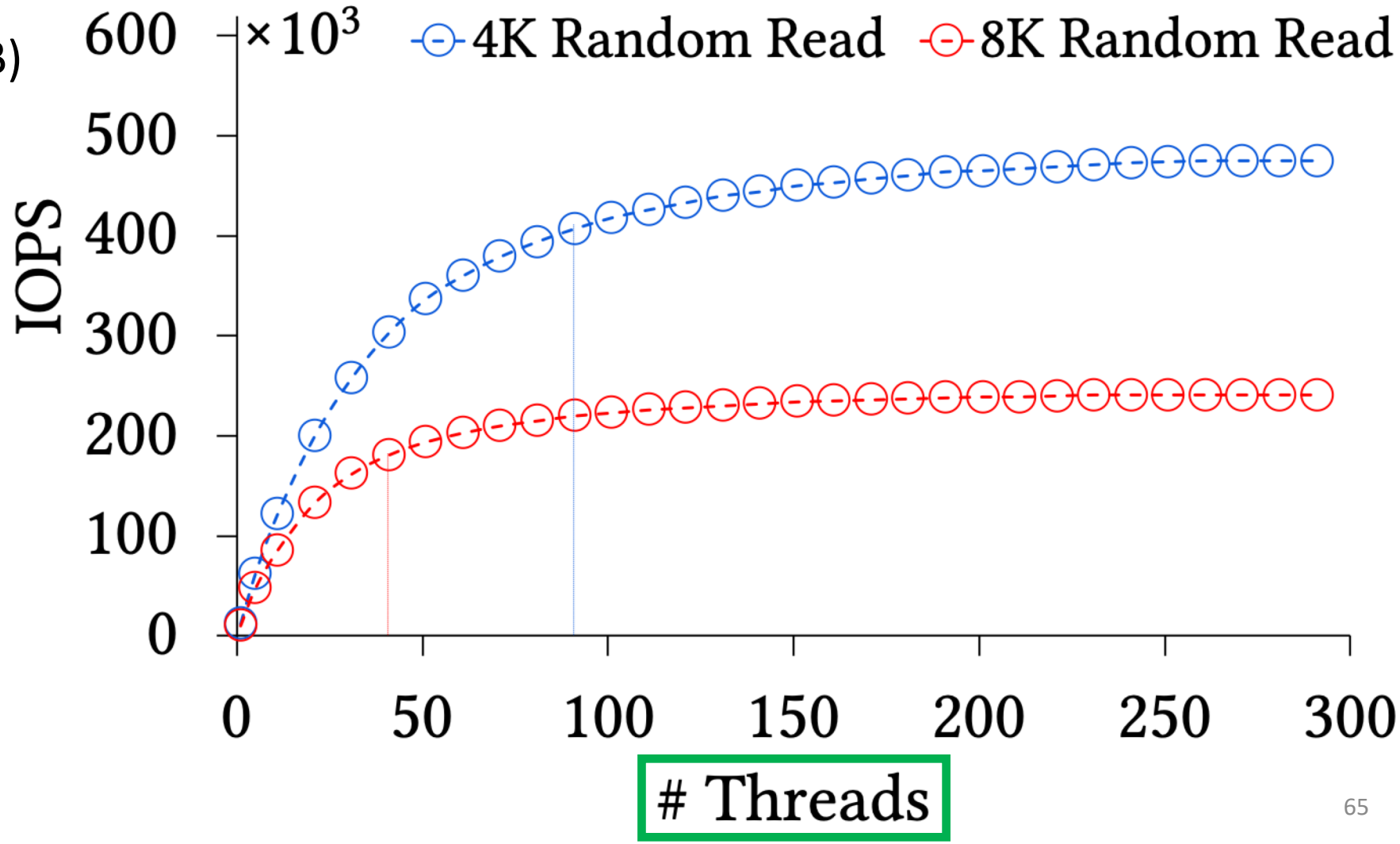
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



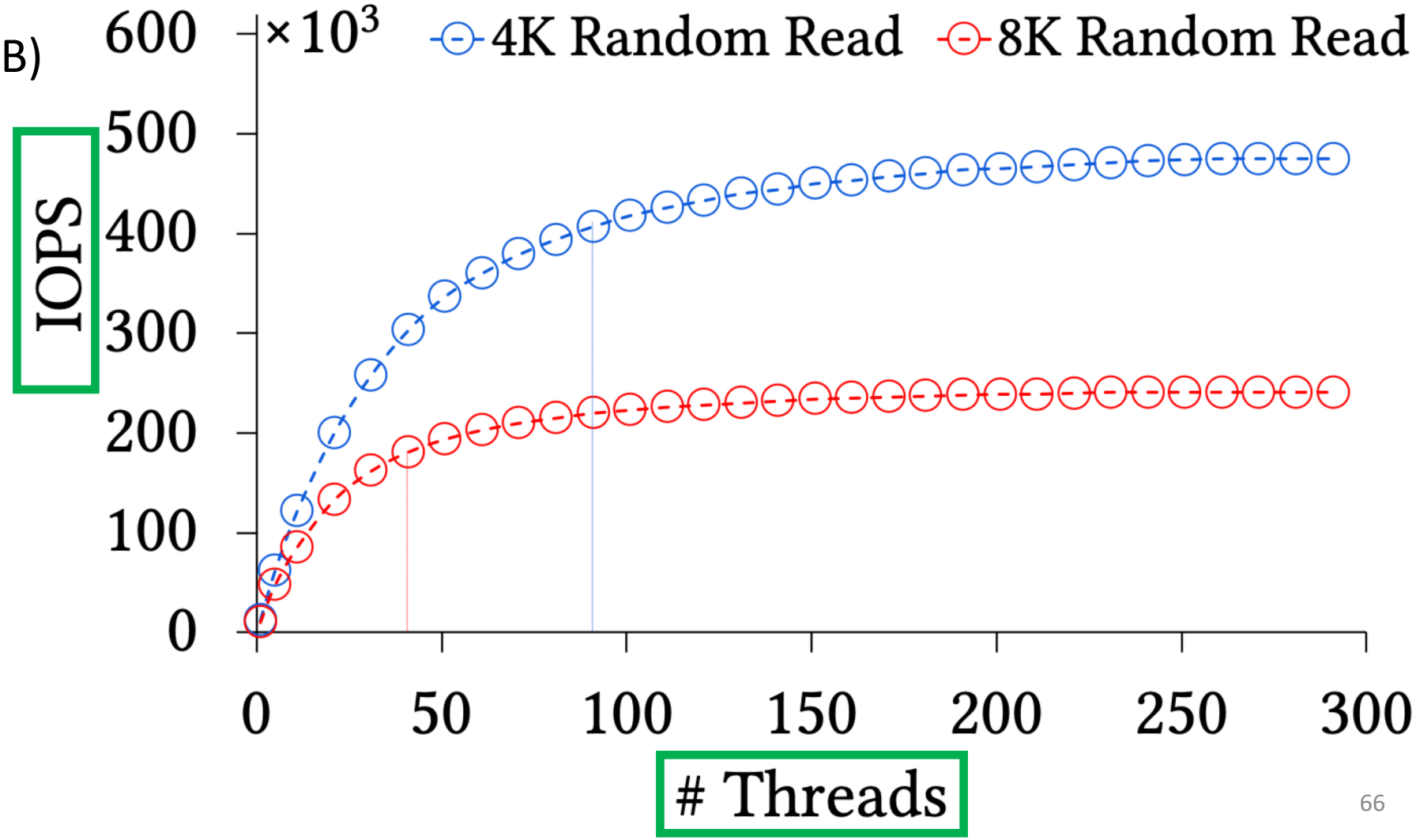
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



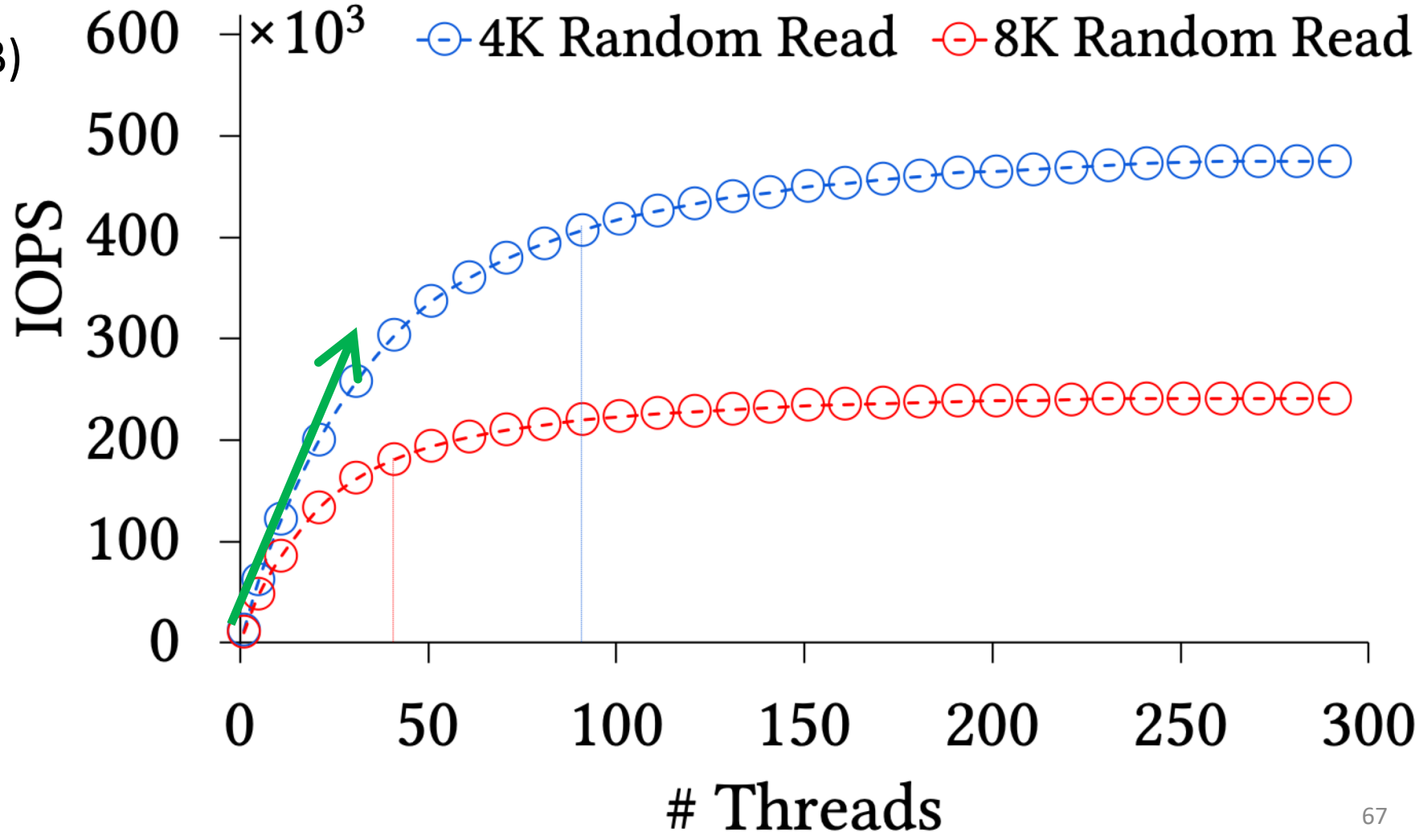
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



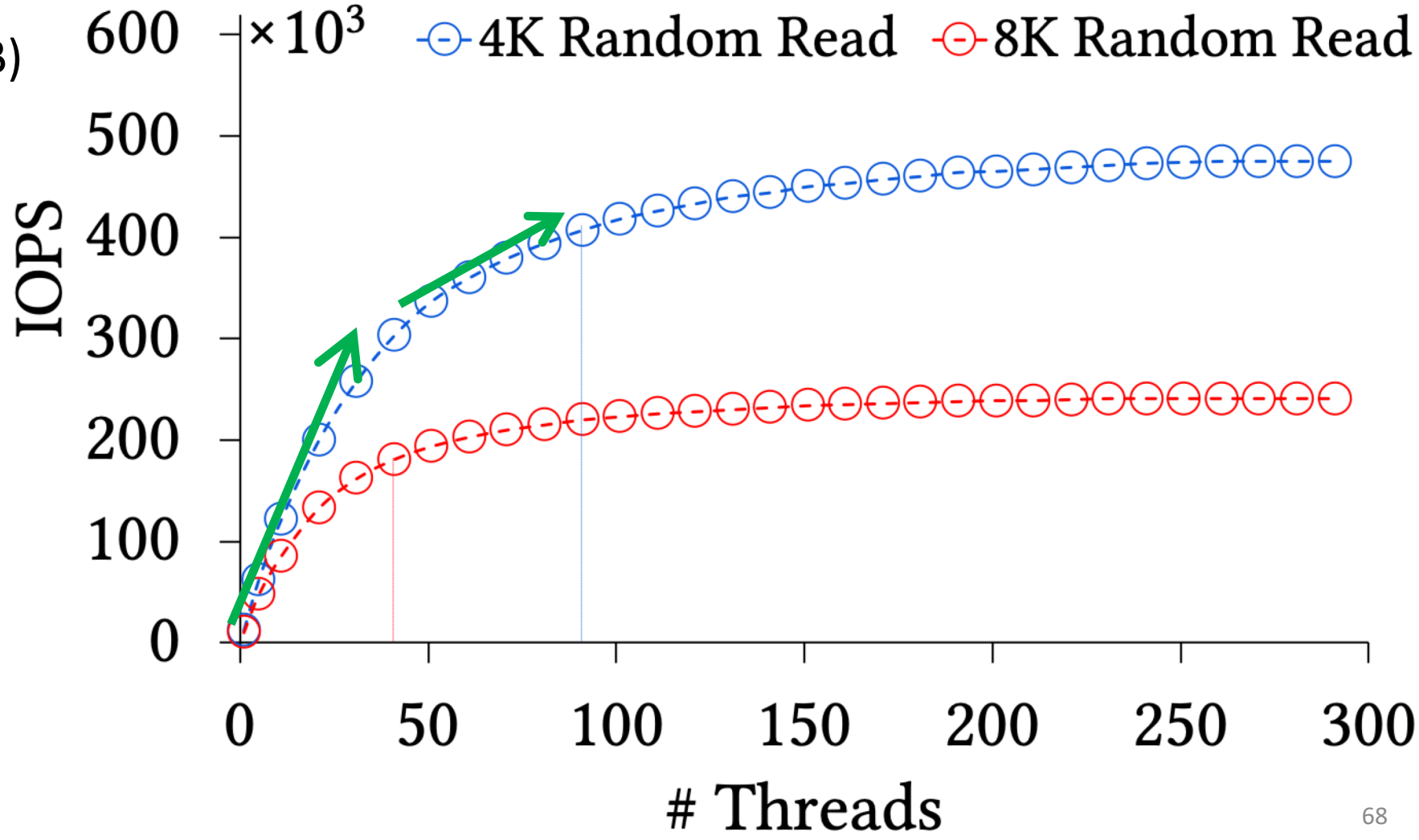
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



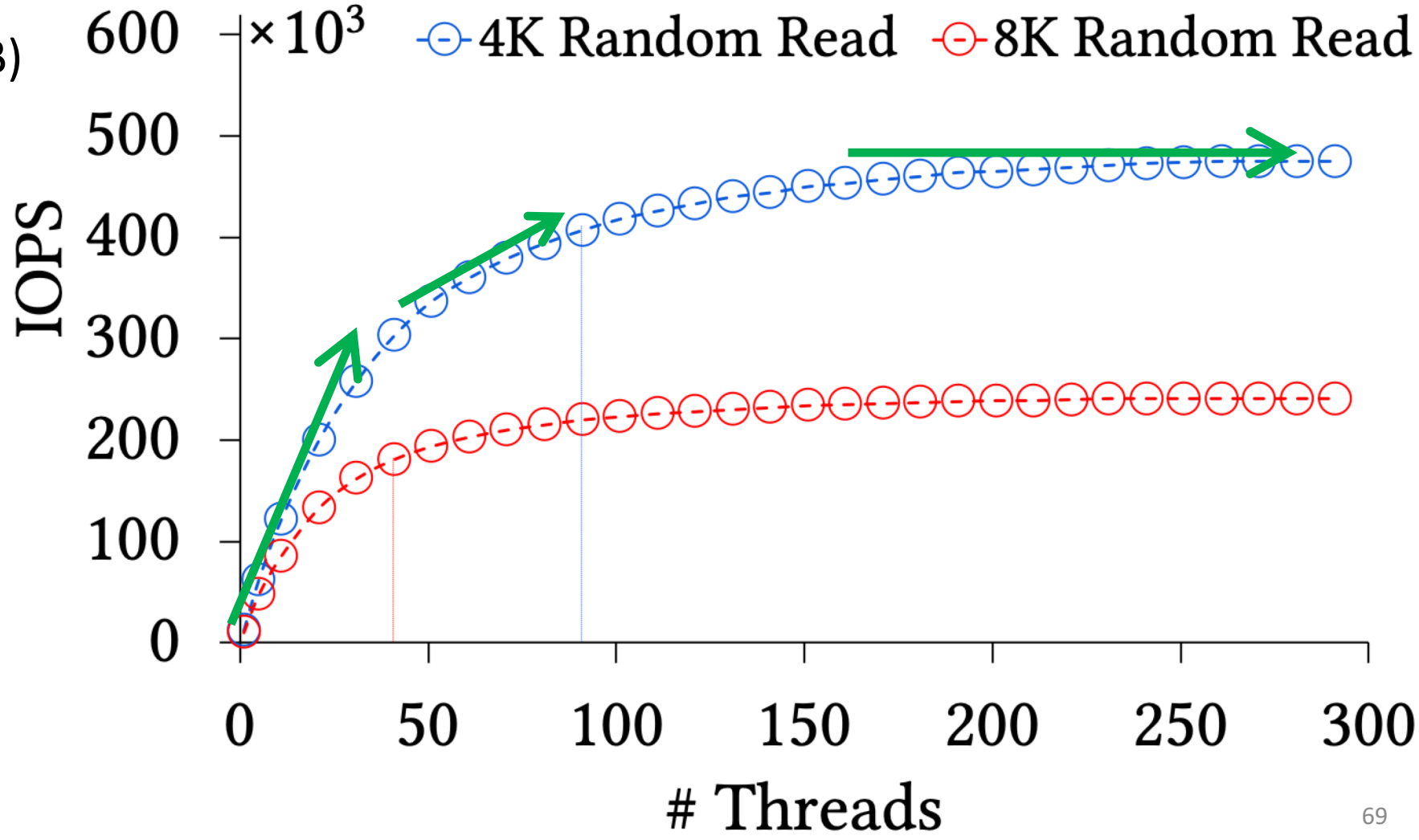
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



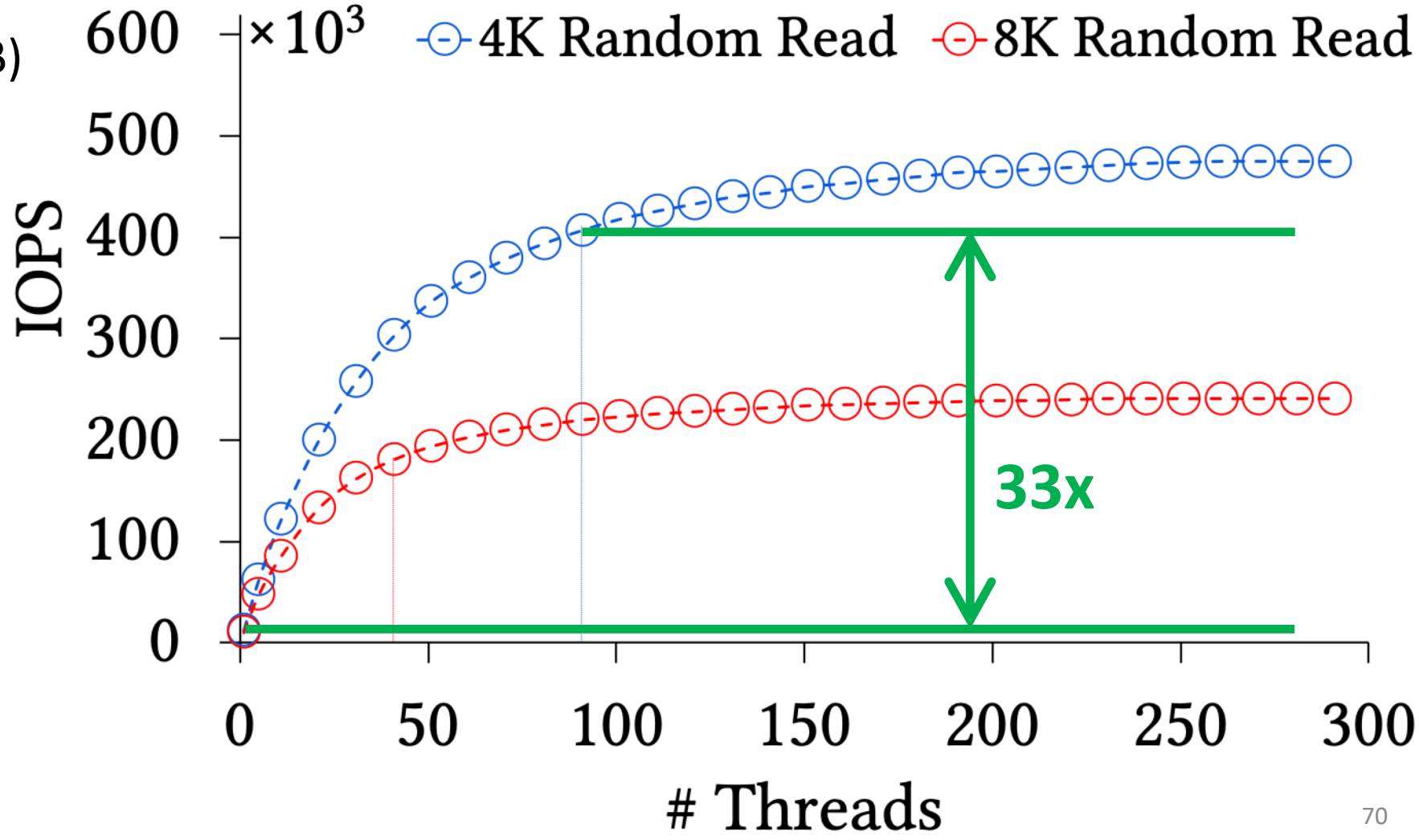
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



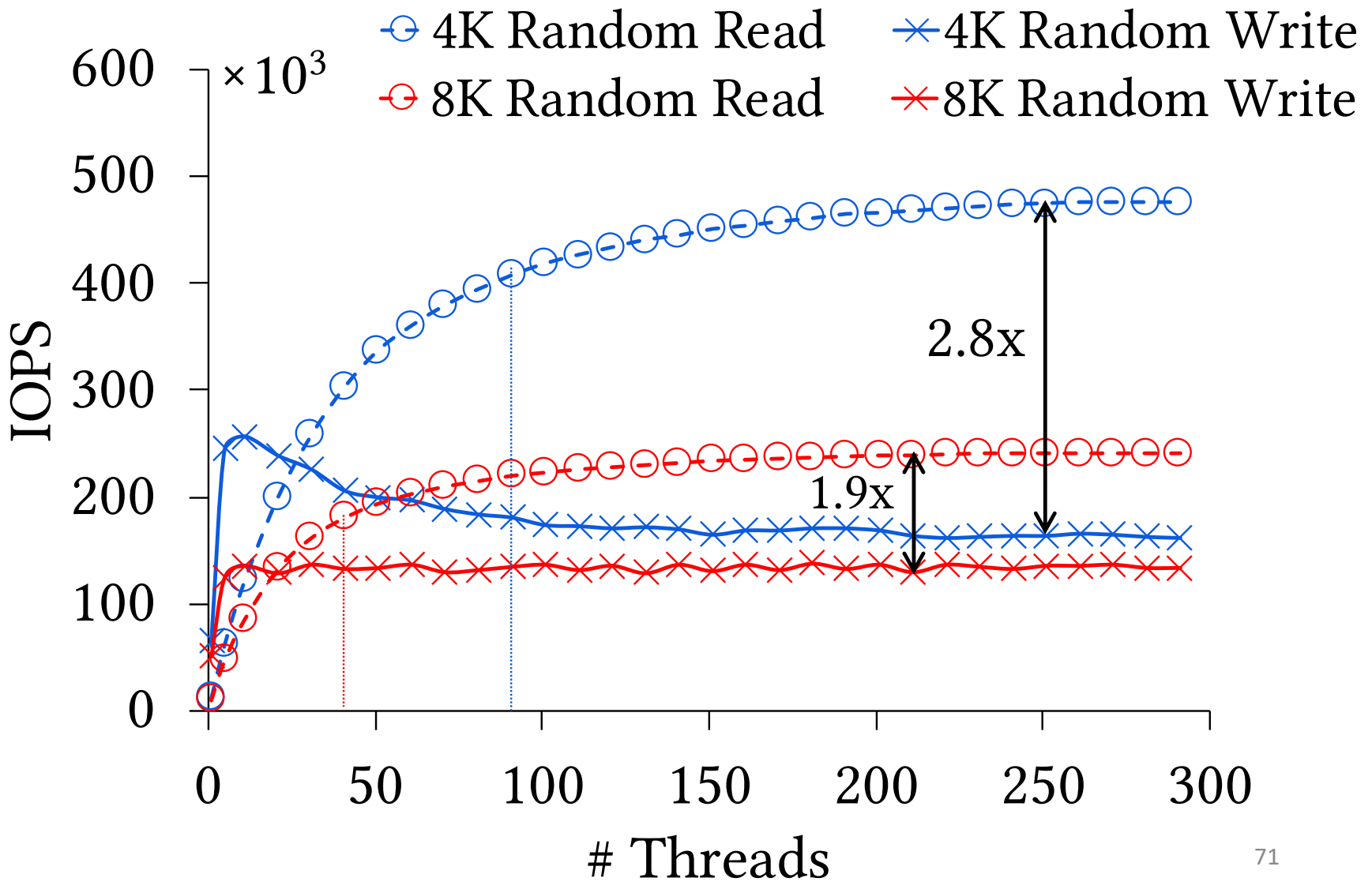
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



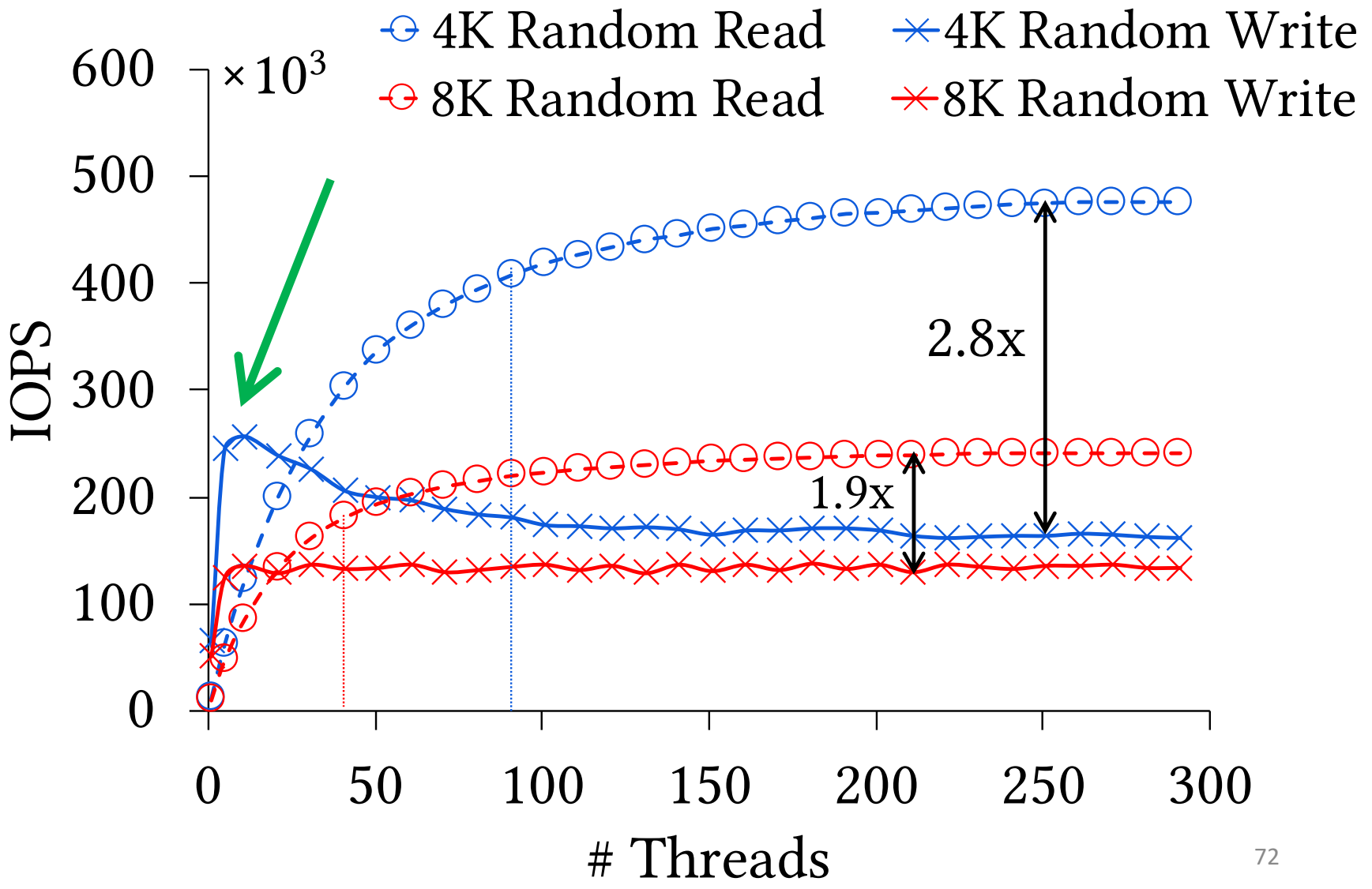
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



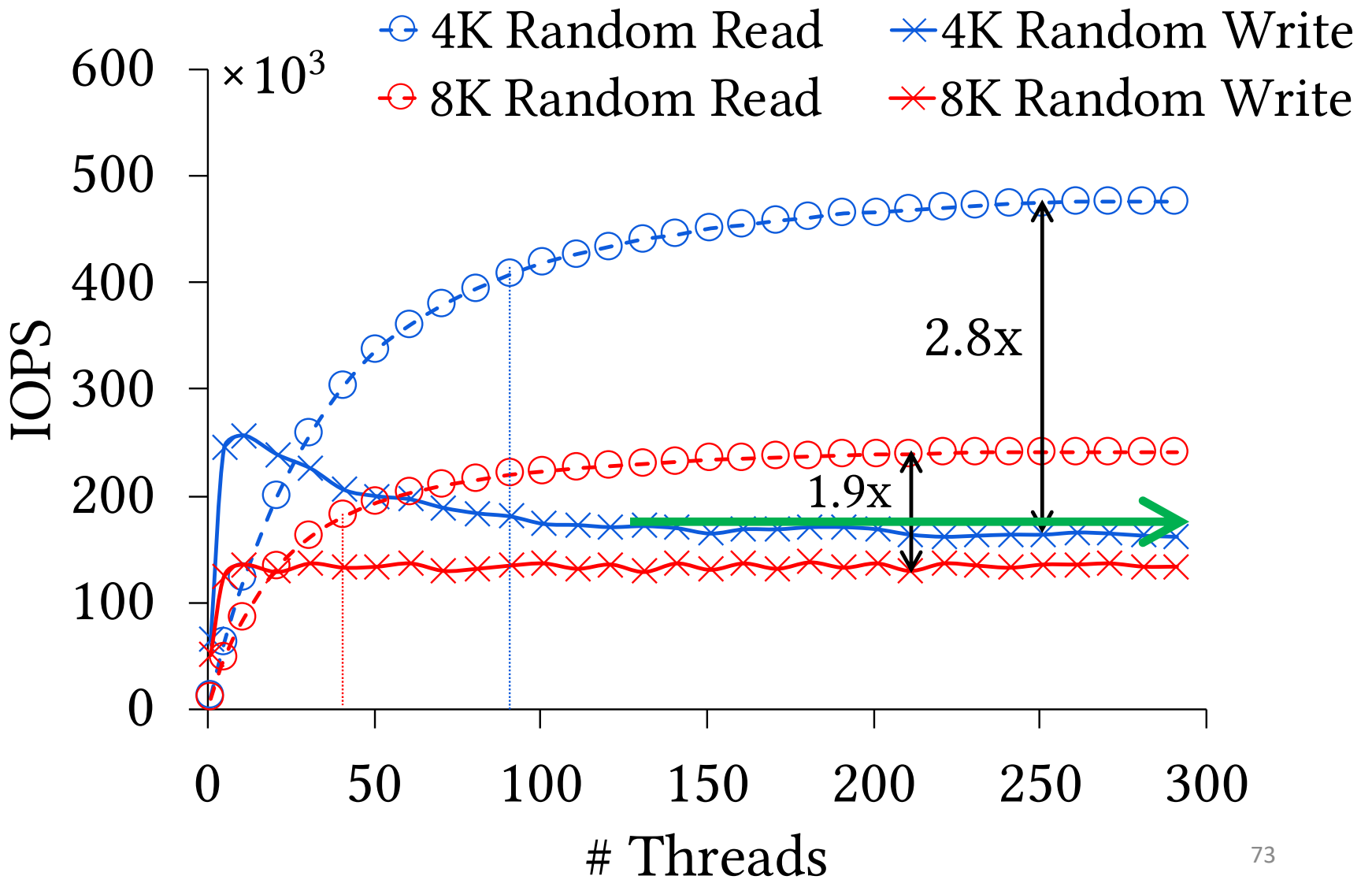
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)



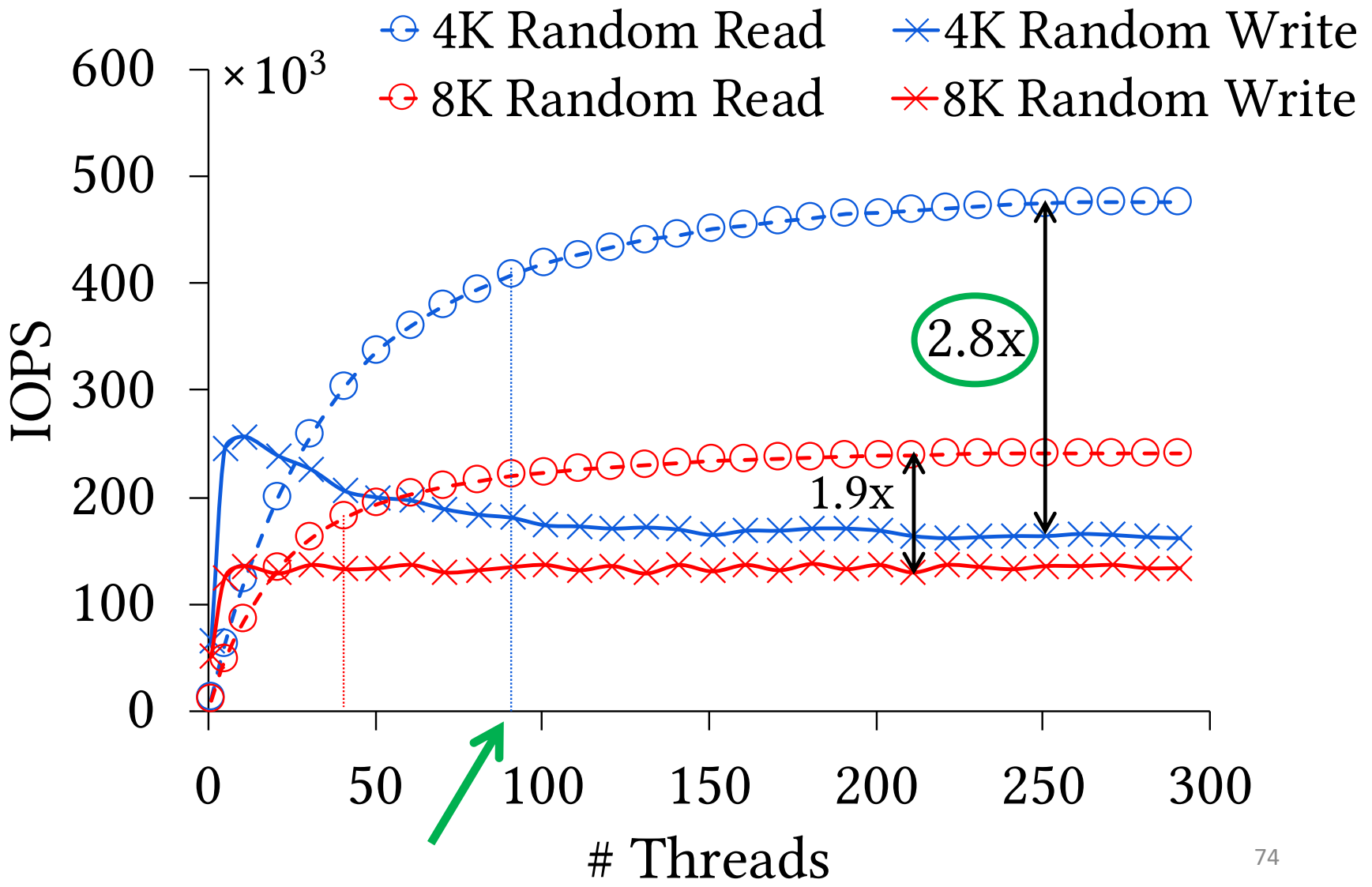
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)

For 4K random read,

Asymmetry: 2.8

Concurrency: 80



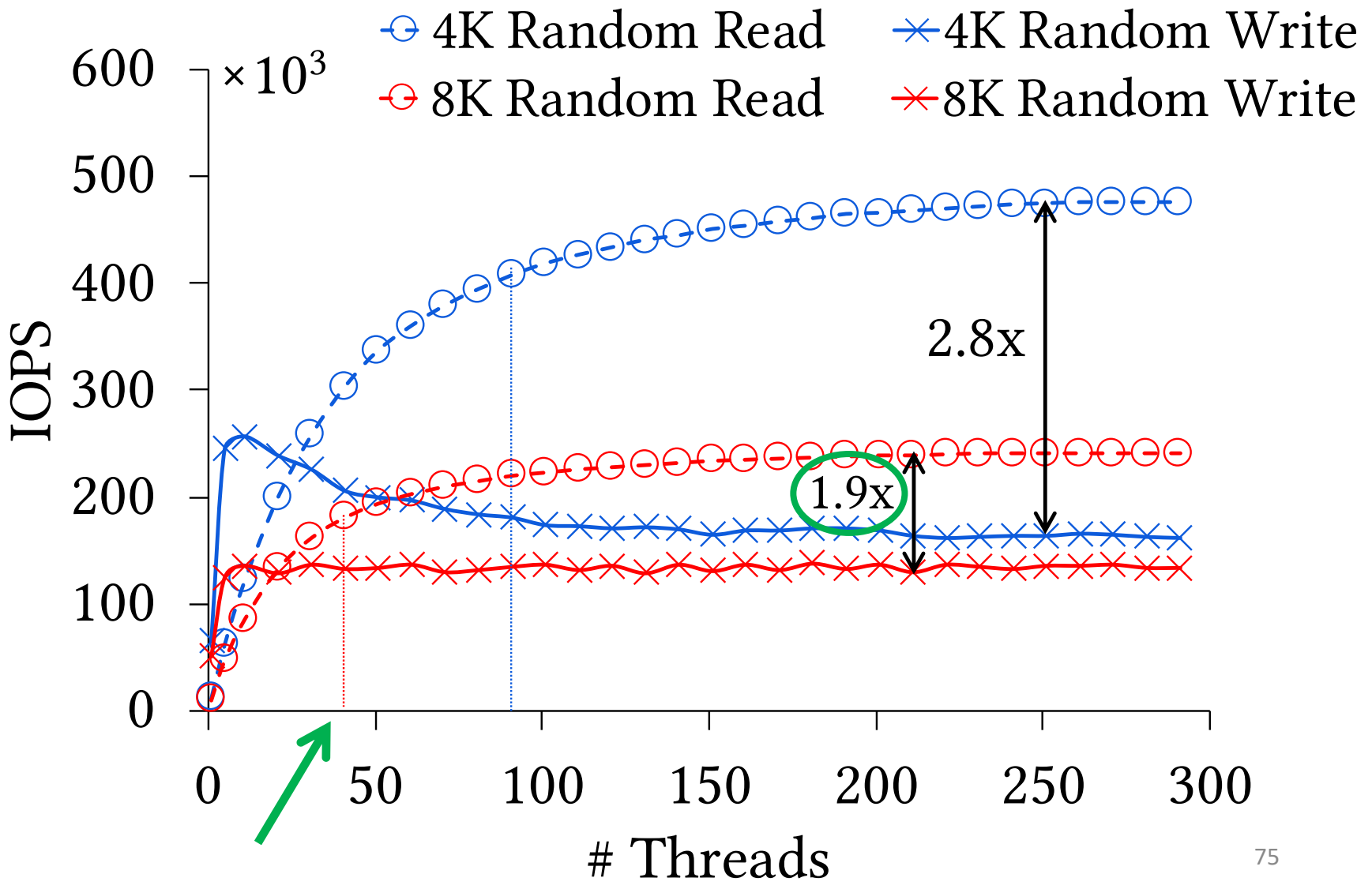
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)

For 8K random read,

Asymmetry: 1.9

Concurrency: 40



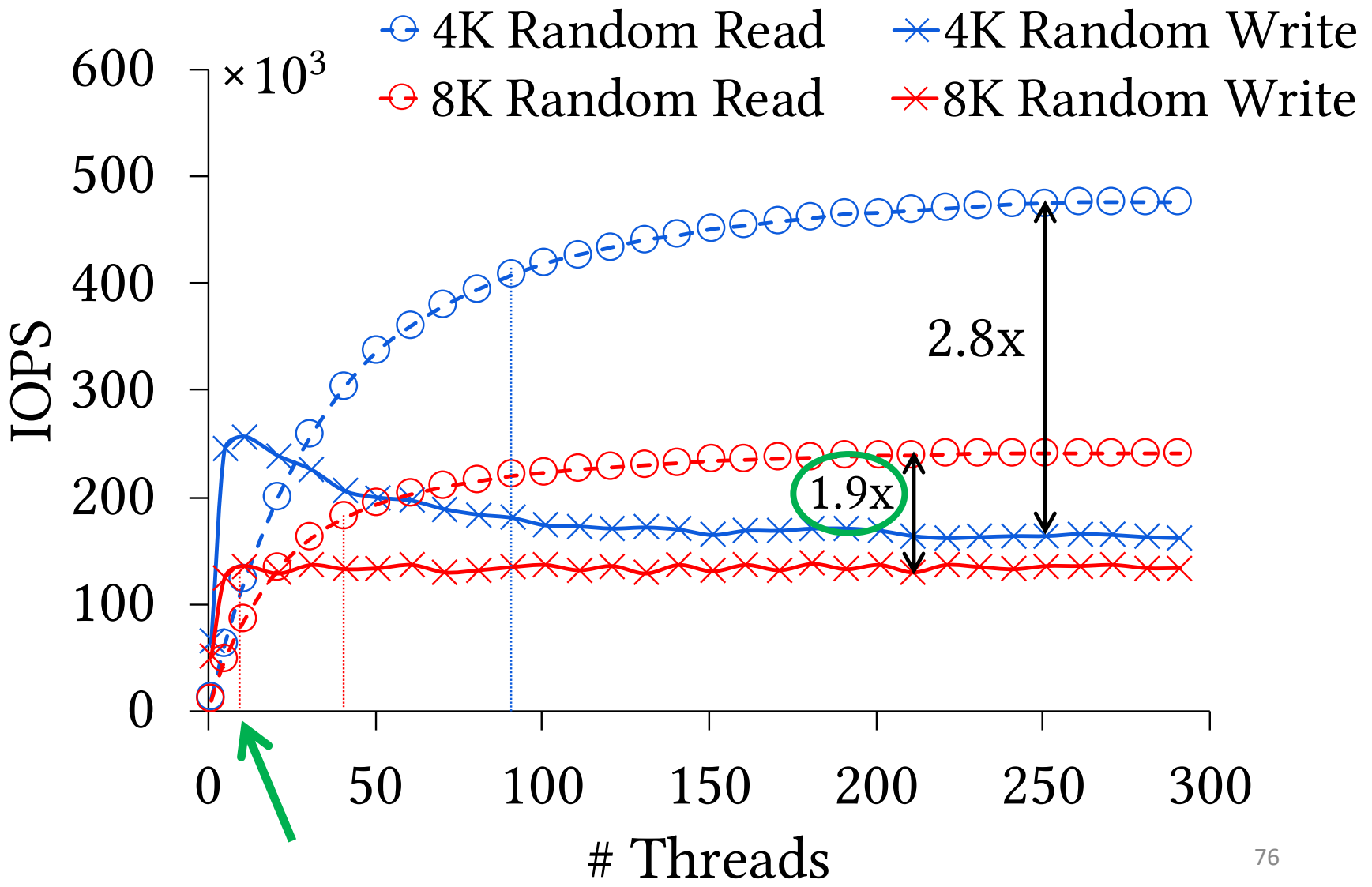
Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)

For 8K random write,

Asymmetry: 1.9

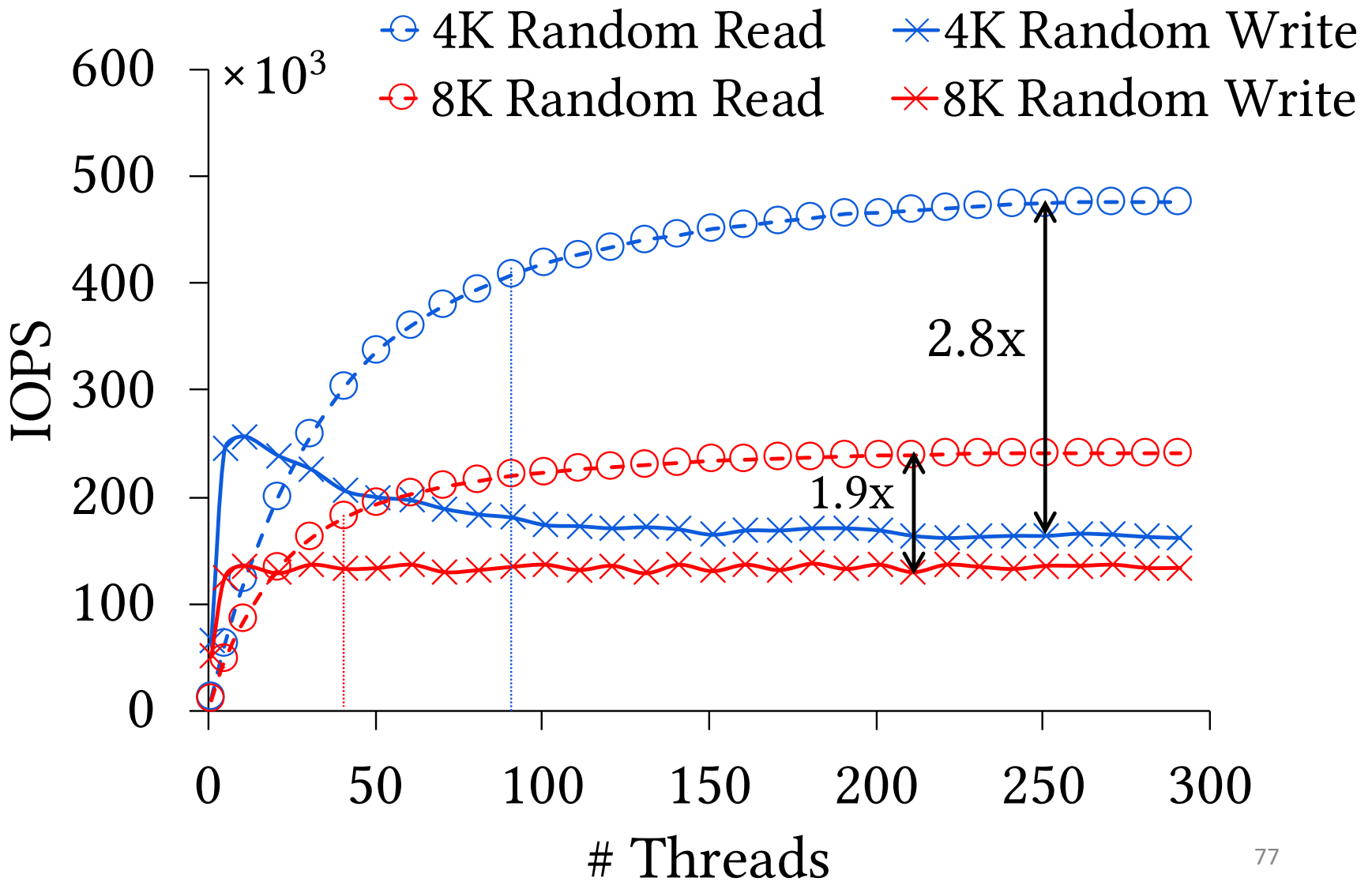
Concurrency: 7



Measuring Asymmetry/Concurrency (With FS)

Device
PCIe SSD - P4510 (1TB)

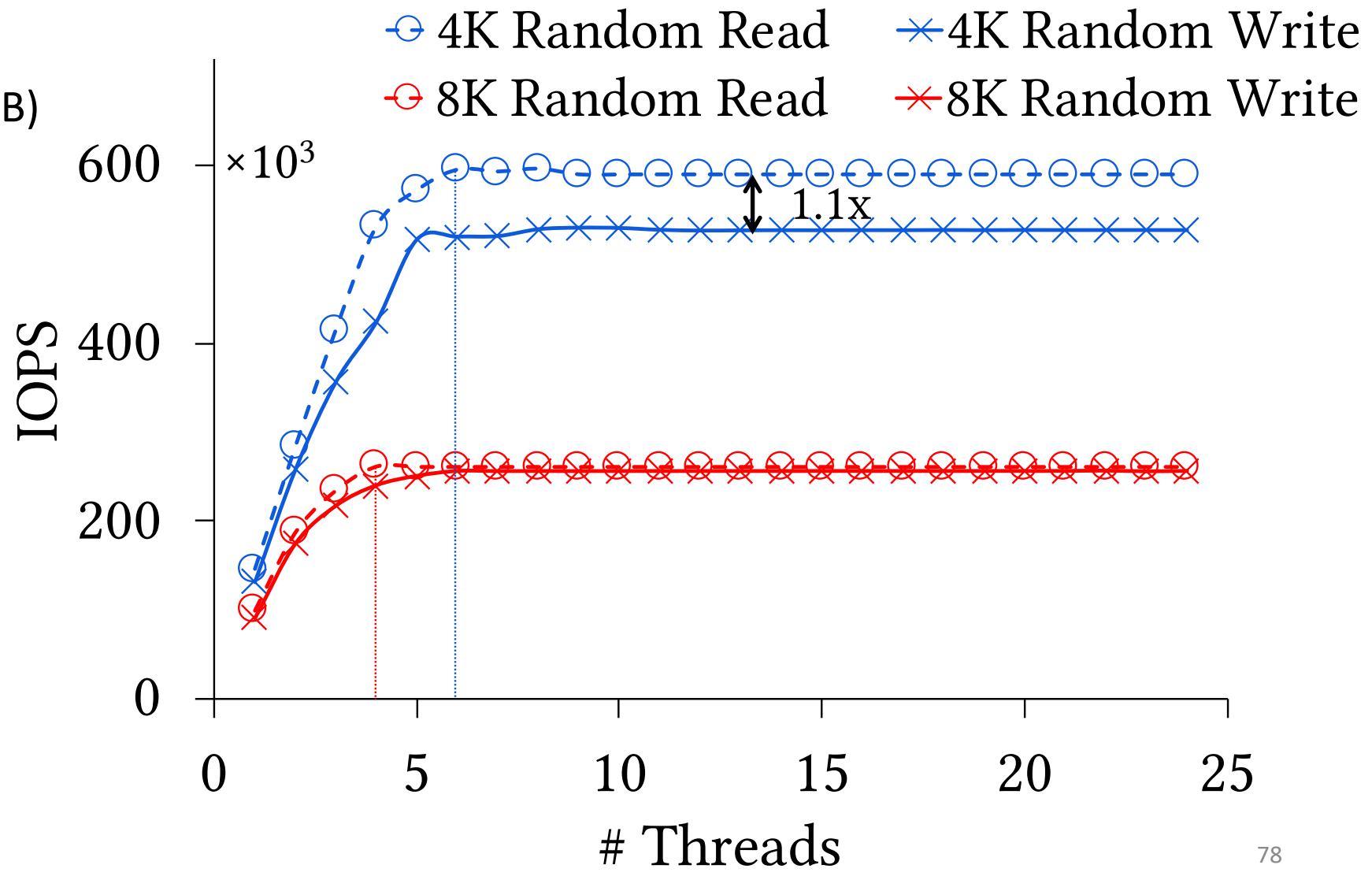
Asymmetry and concurrency depends on request type and access granularity.



Measuring Asymmetry/Concurrency (With FS)

Device

Optane SSD - P4800X (375GB)



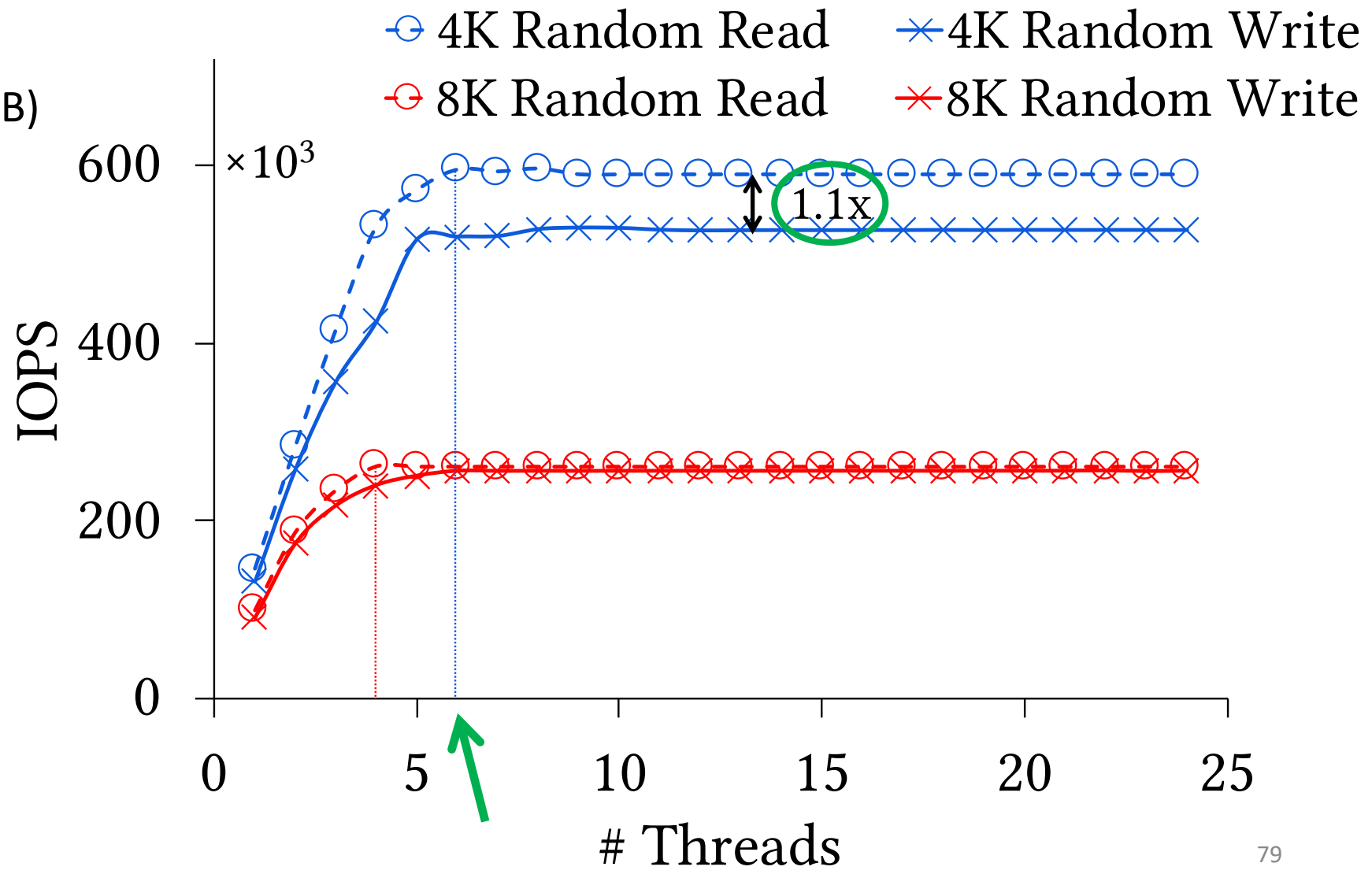
Measuring Asymmetry/Concurrency (With FS)

Device
Optane SSD - P4800X (375GB)

For 4K random read,

Asymmetry: 1.1

Concurrency: 6



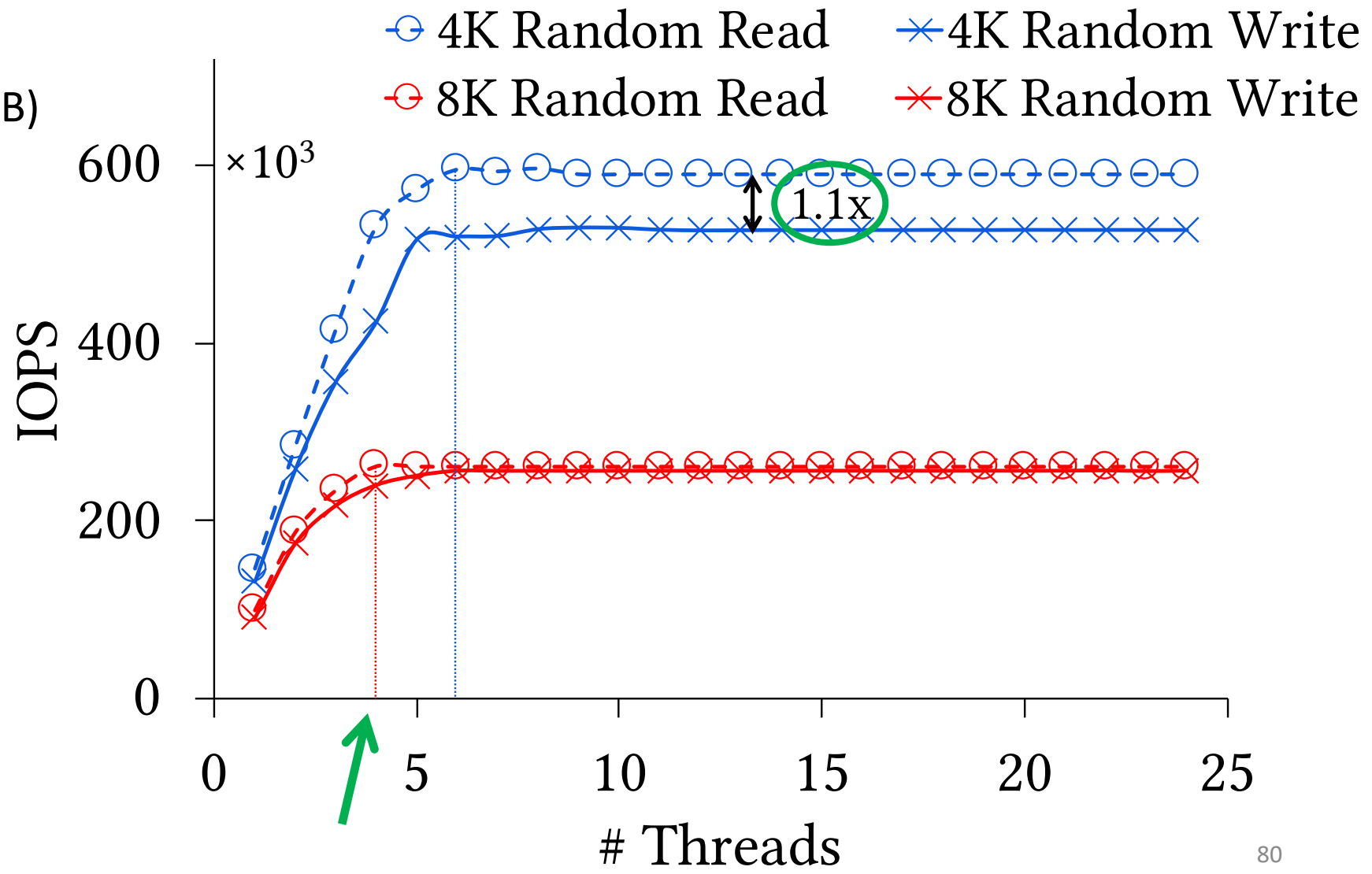
Measuring Asymmetry/Concurrency (With FS)

Device
Optane SSD - P4800X (375GB)

For 8K random read,

Asymmetry: 1.1

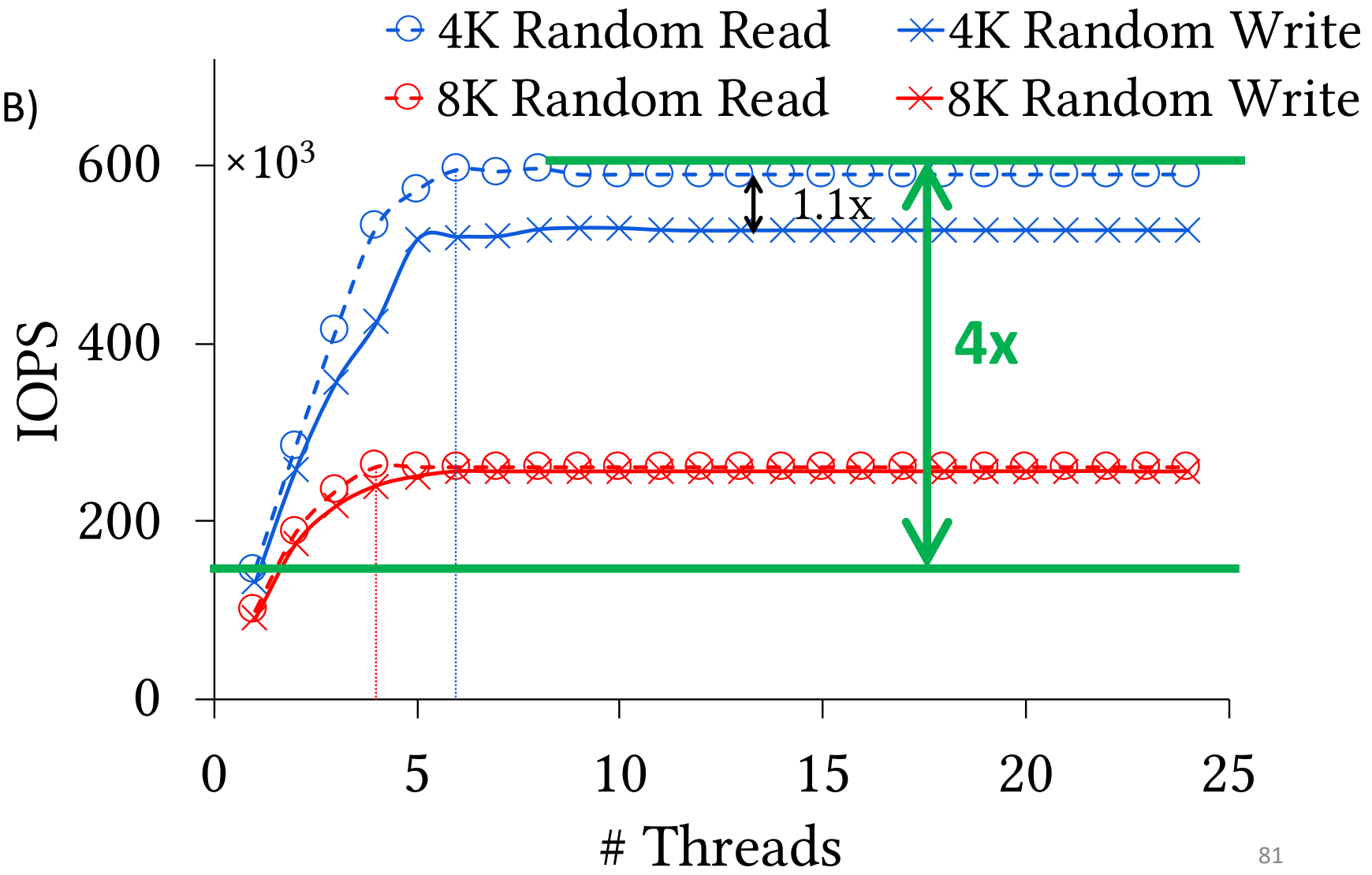
Concurrency: 4



Measuring Asymmetry/Concurrency (With FS)

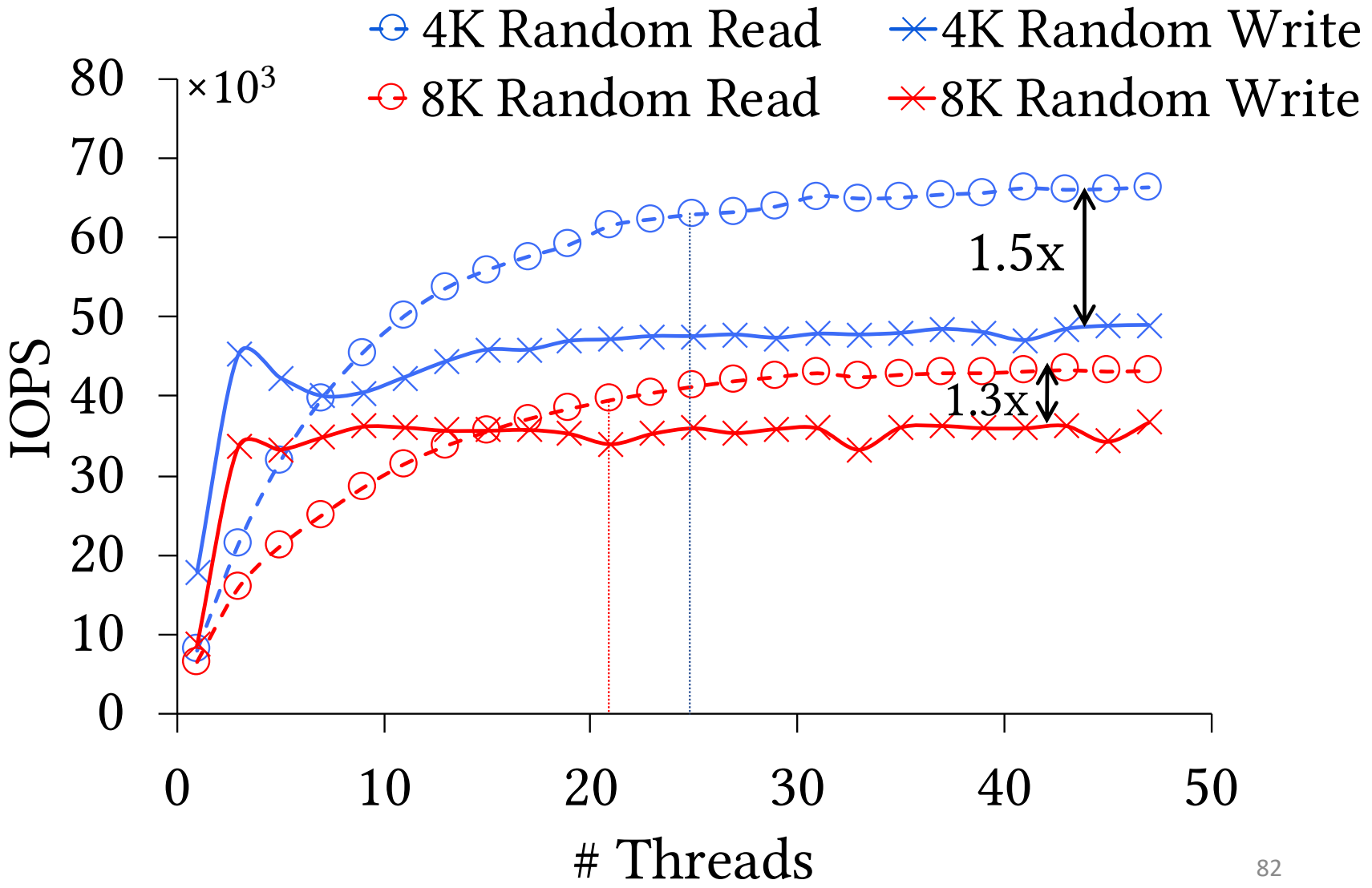
Device

Optane SSD - P4800X (375GB)



Measuring Asymmetry/Concurrency (With FS)

Device
SATA SSD - S4610 (240GB)



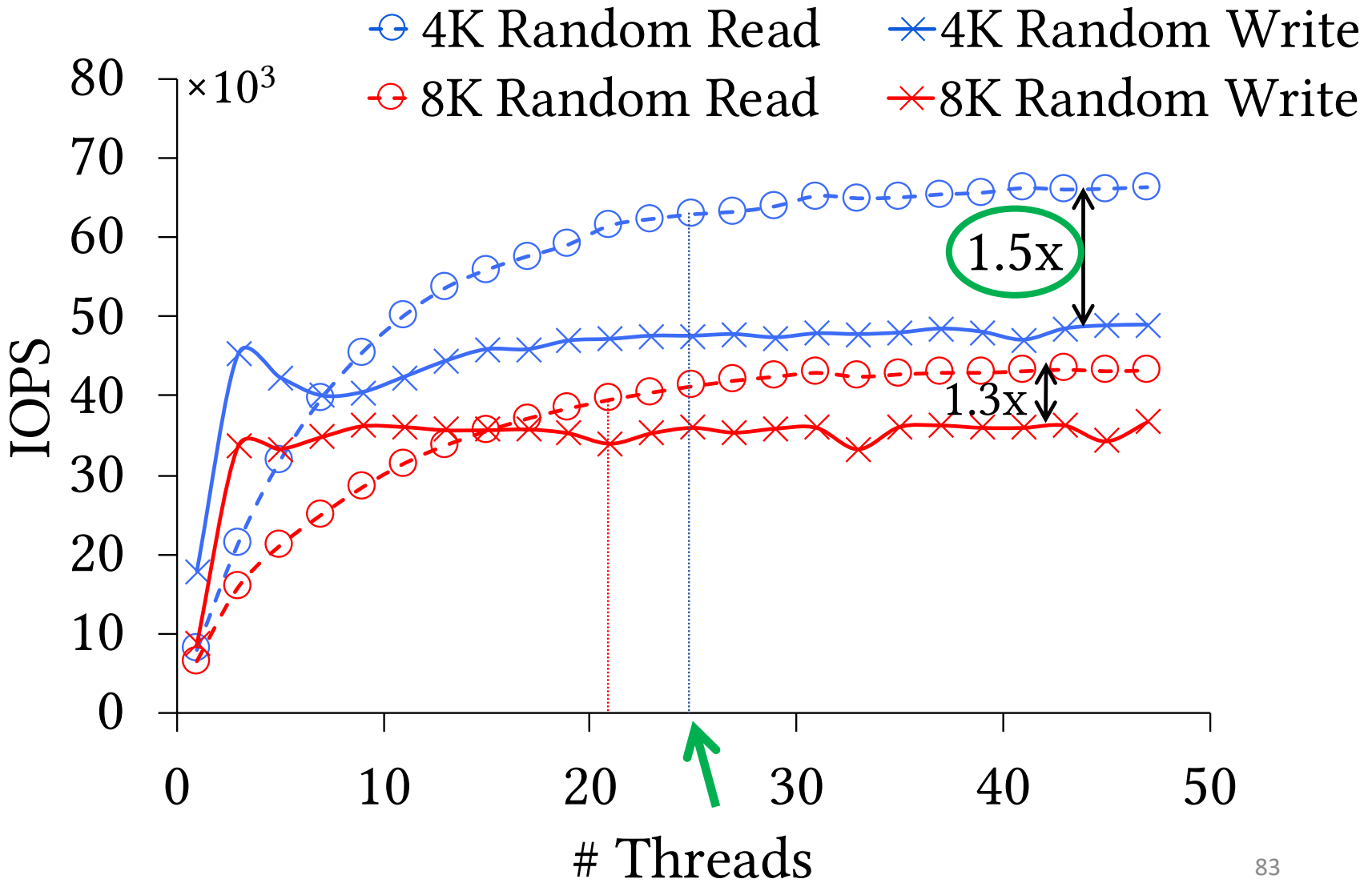
Measuring Asymmetry/Concurrency (With FS)

Device
SATA SSD - S4610 (240GB)

For 4K random read,

Asymmetry: 1.5

Concurrency: 25



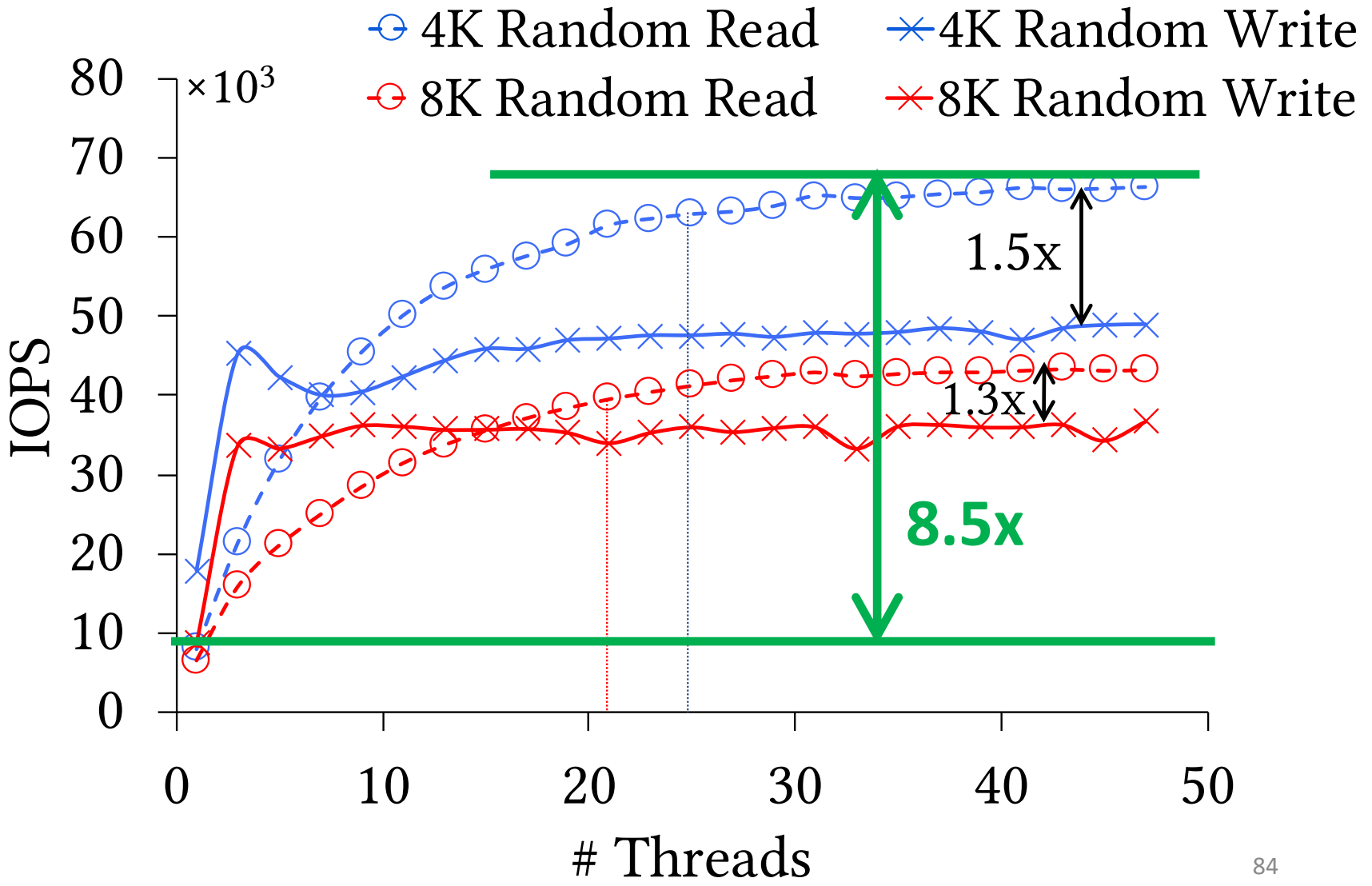
Measuring Asymmetry/Concurrency (With FS)

Device
SATA SSD - S4610 (240GB)

For 4K random read,

Asymmetry: 1.5

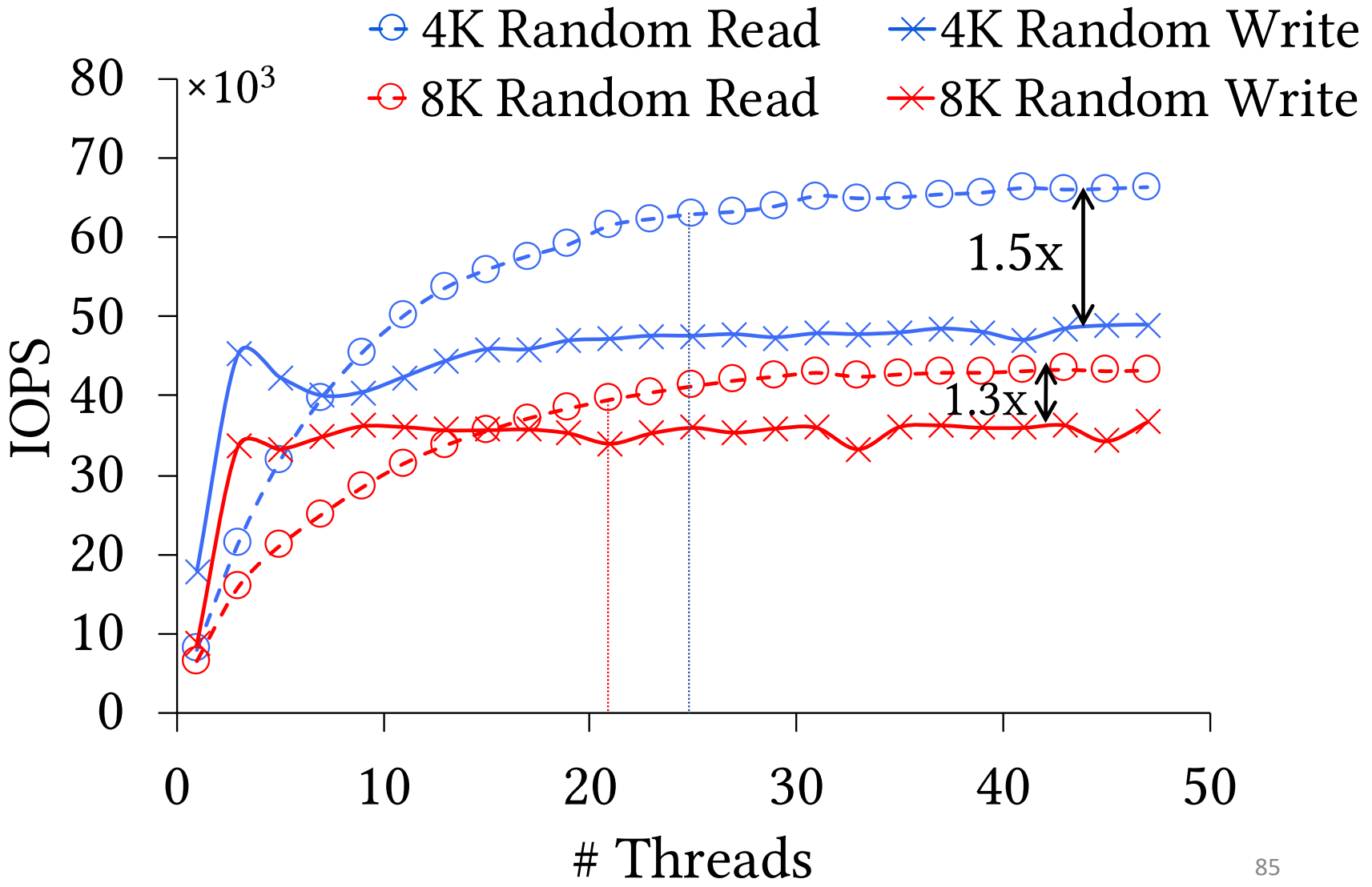
Concurrency: 25



Measuring Asymmetry/Concurrency (With FS)

Device
SATA SSD - S4610 (240GB)

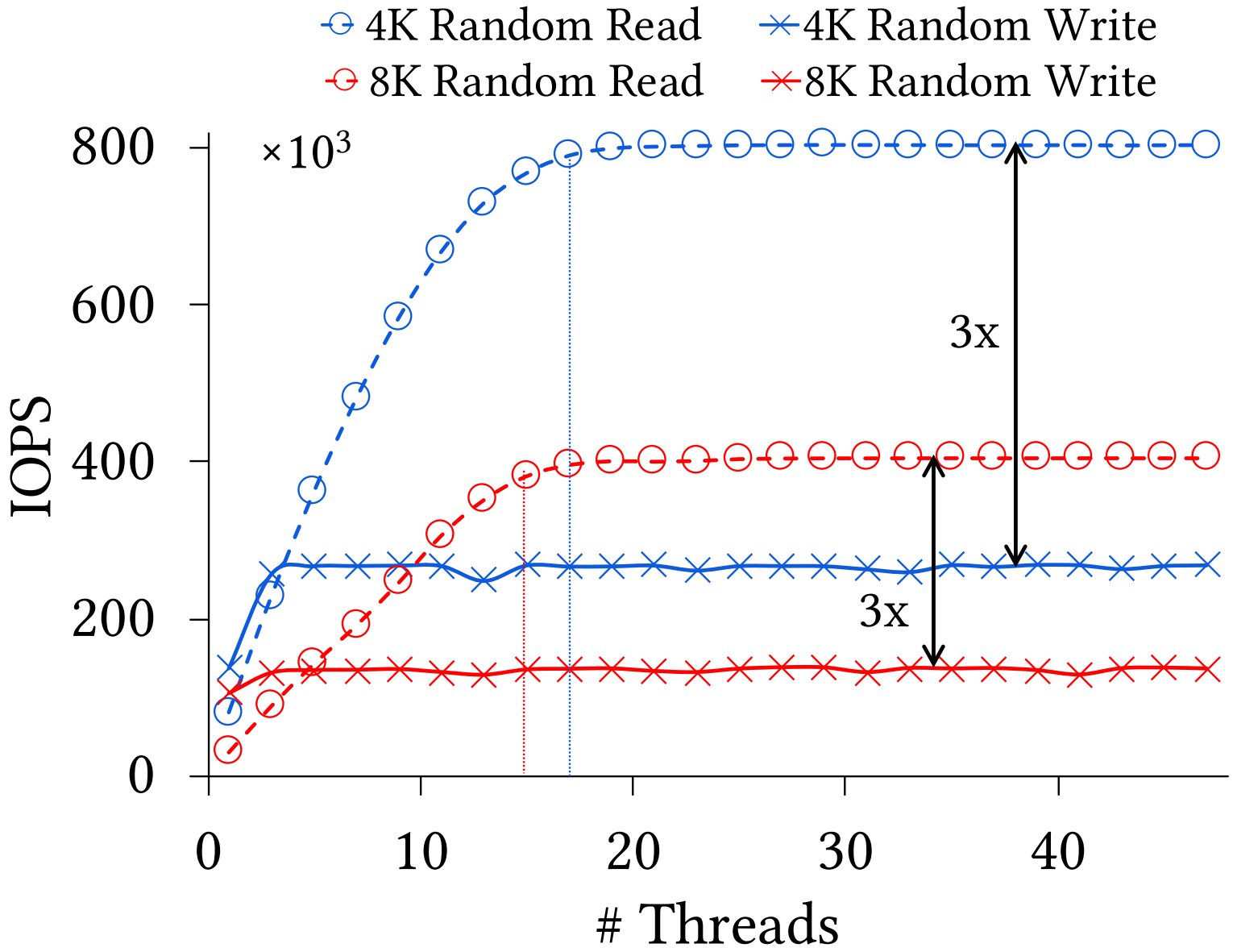
The impact of concurrency is high for any device.



Measuring Asymmetry/Concurrency (Without FS)

Device

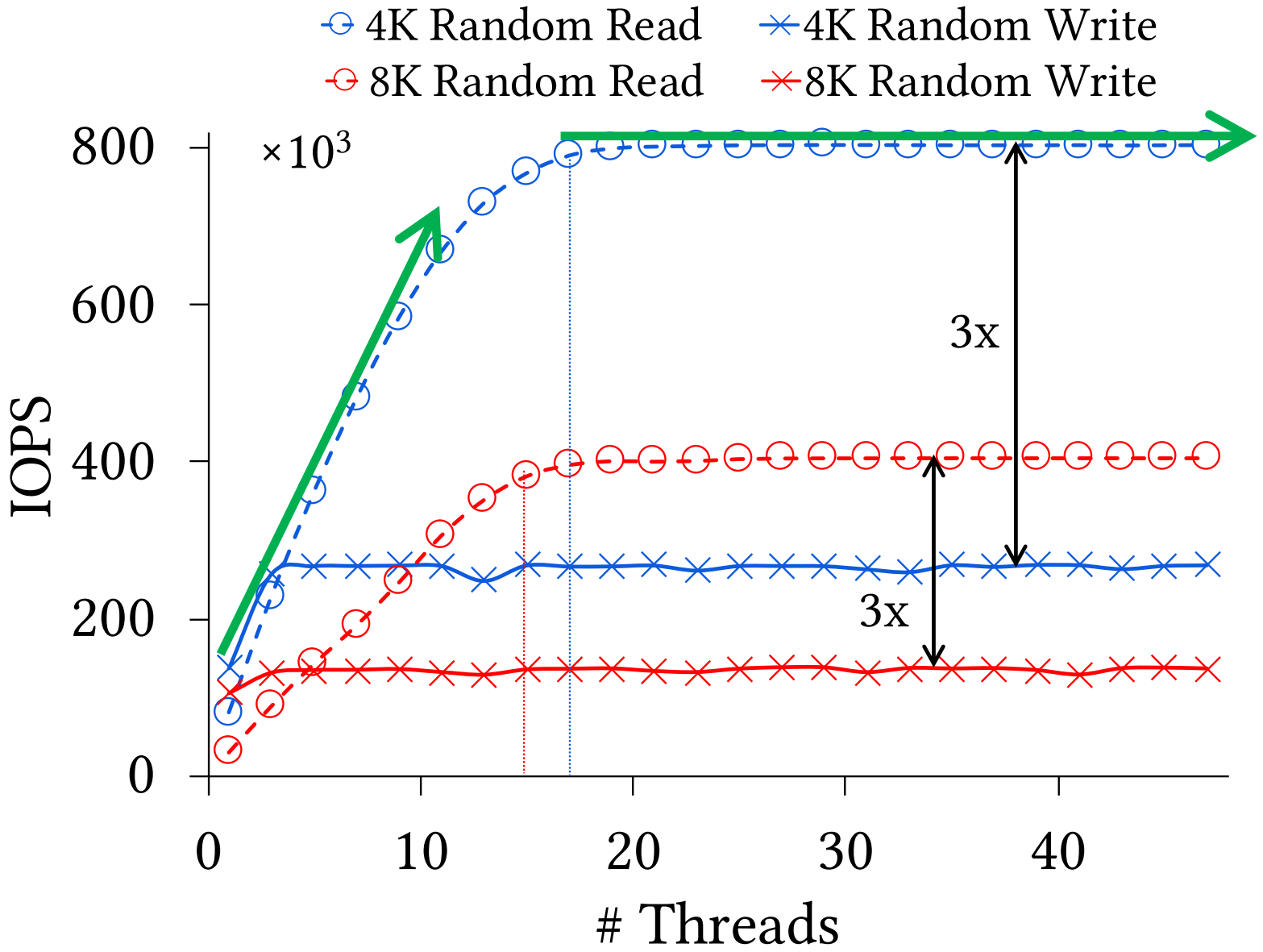
PCIe SSD - P4510 (1TB)



Measuring Asymmetry/Concurrency (Without FS)

Device

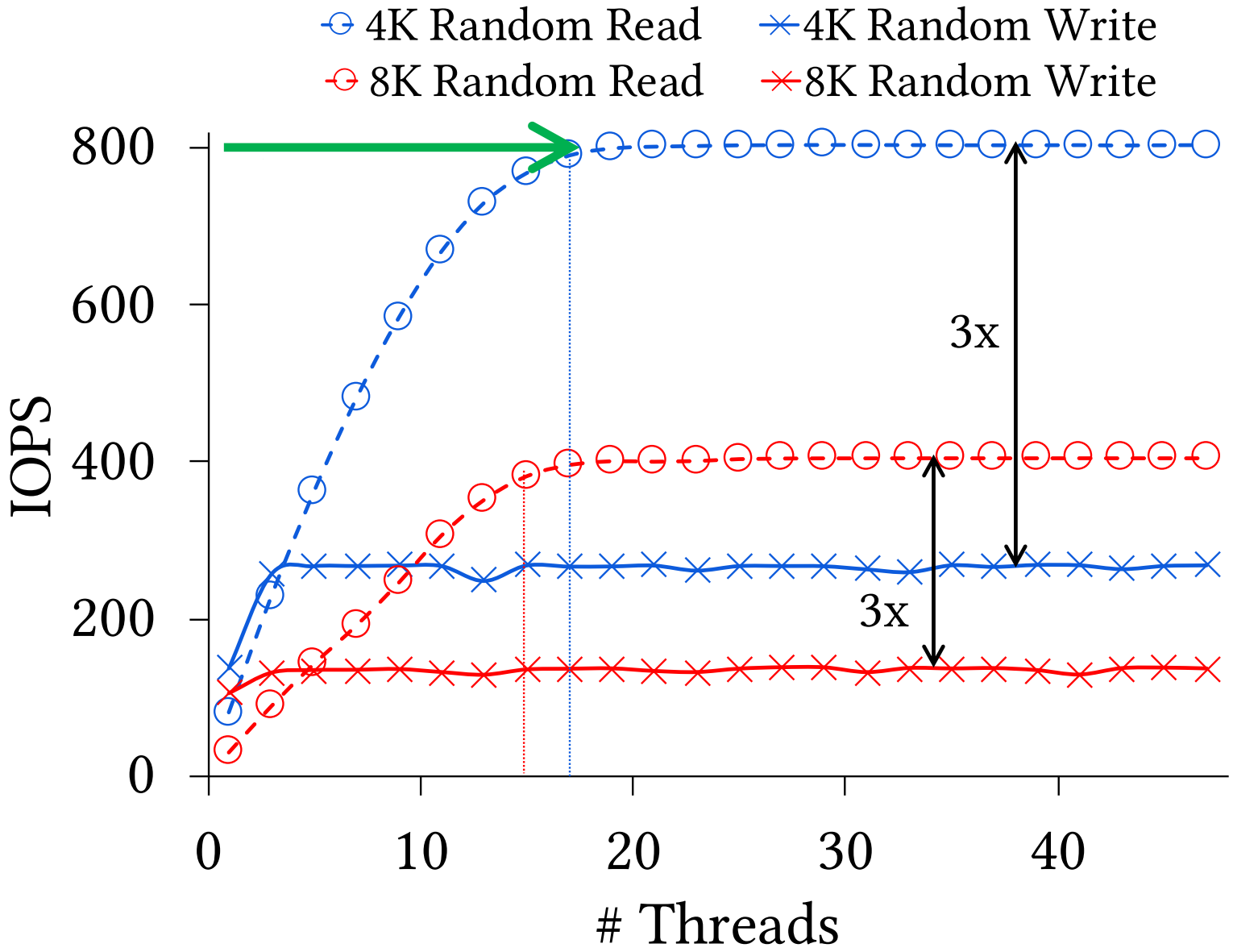
PCIe SSD - P4510 (1TB)



Measuring Asymmetry/Concurrency (Without FS)

Device

PCIe SSD - P4510 (1TB)



Measuring Asymmetry/Concurrency (Without FS)

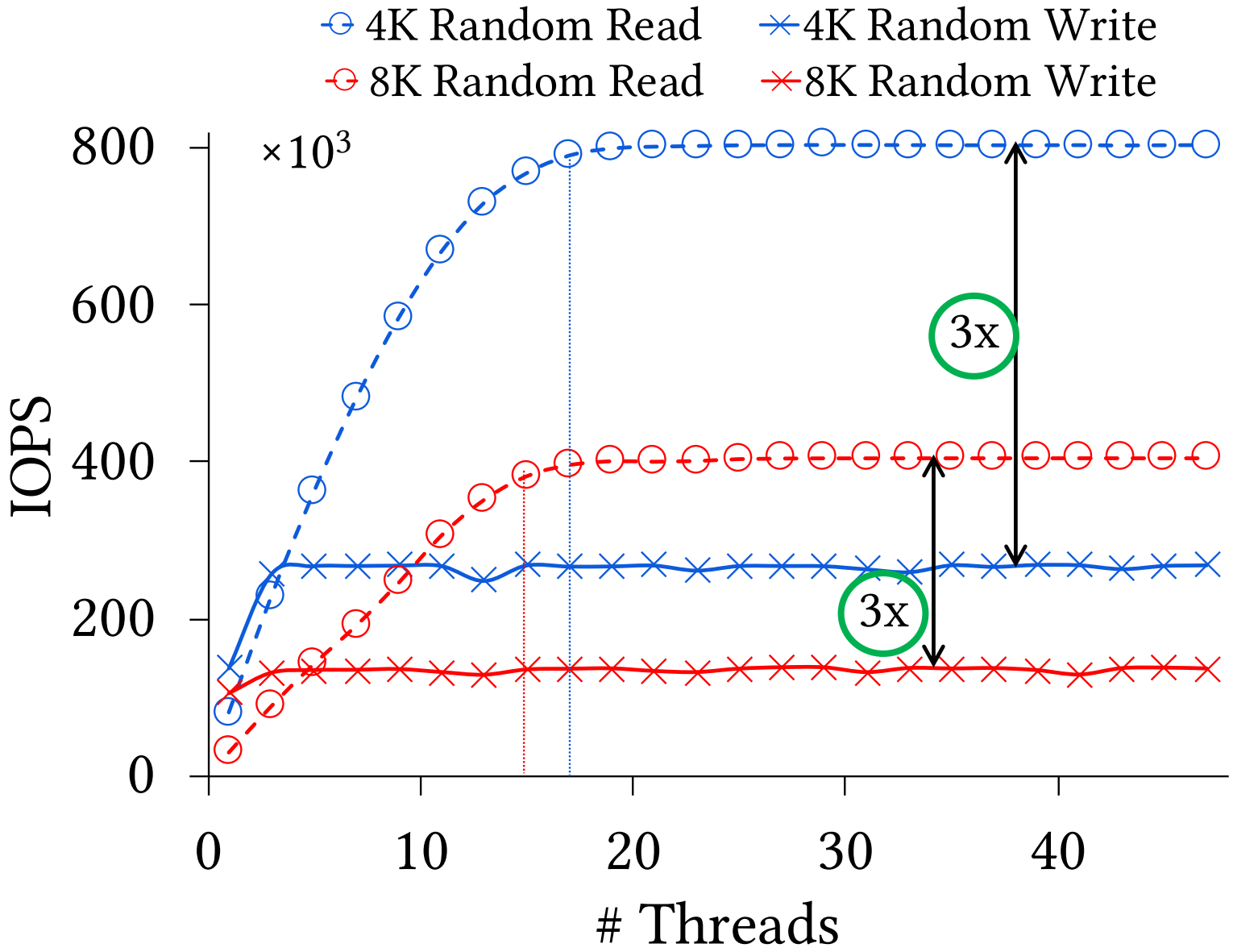
Device

PCIe SSD - P4510 (1TB)

For 4K random read,

Asymmetry: 3

Concurrency: 17

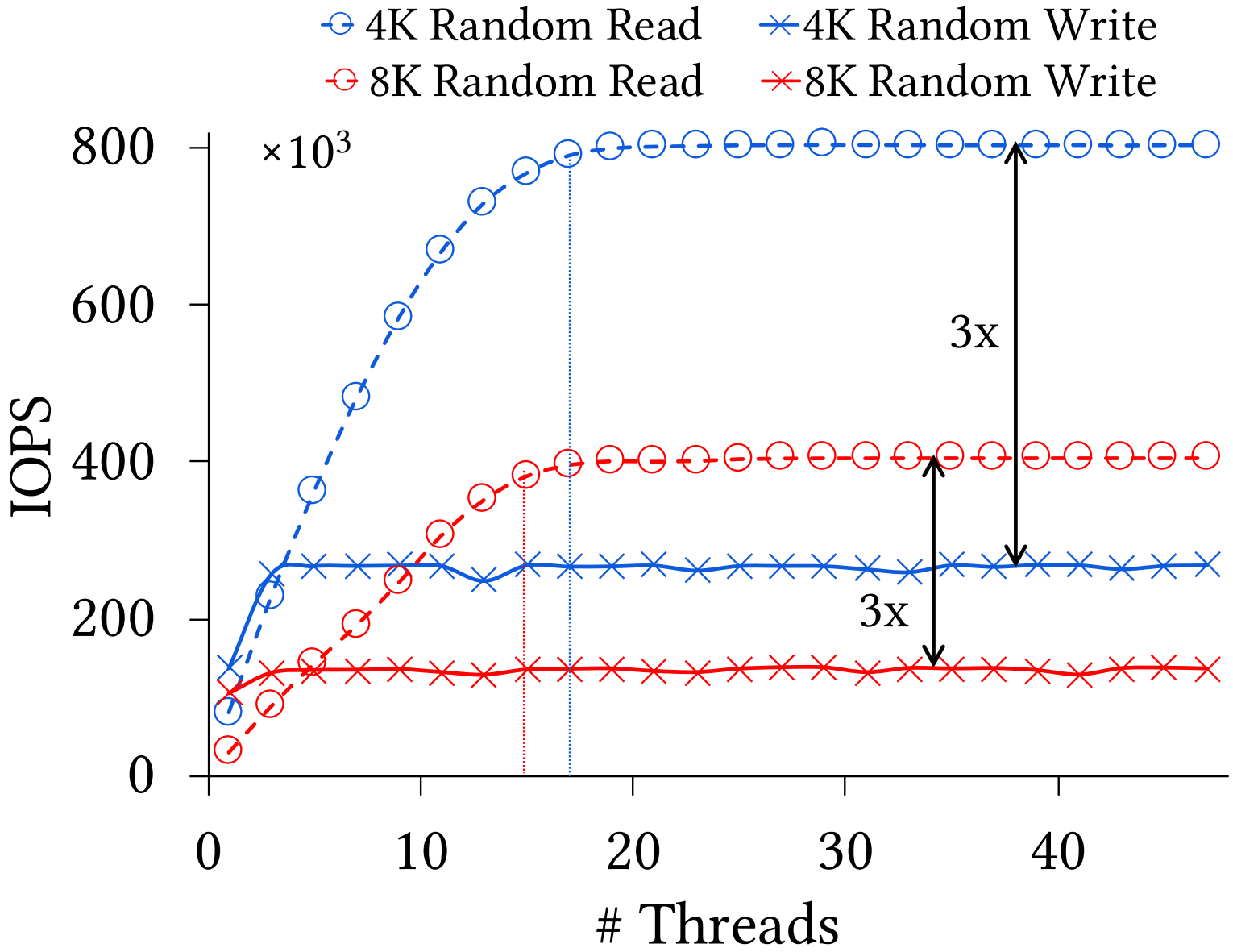


Measuring Asymmetry/Concurrency (Without FS)

Device

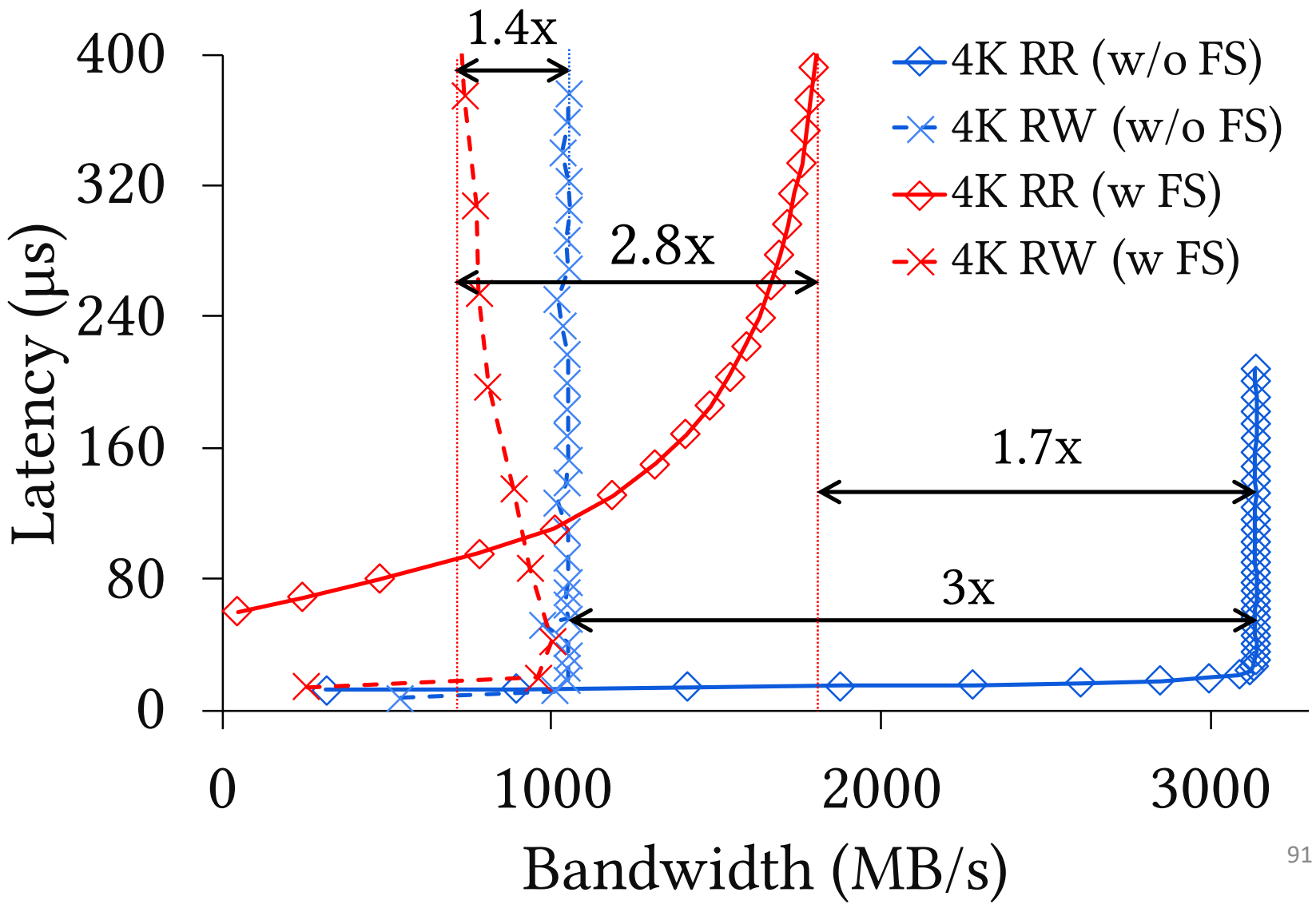
PCIe SSD - P4510 (1TB)

The observed α and k depends on the presence of the FS.



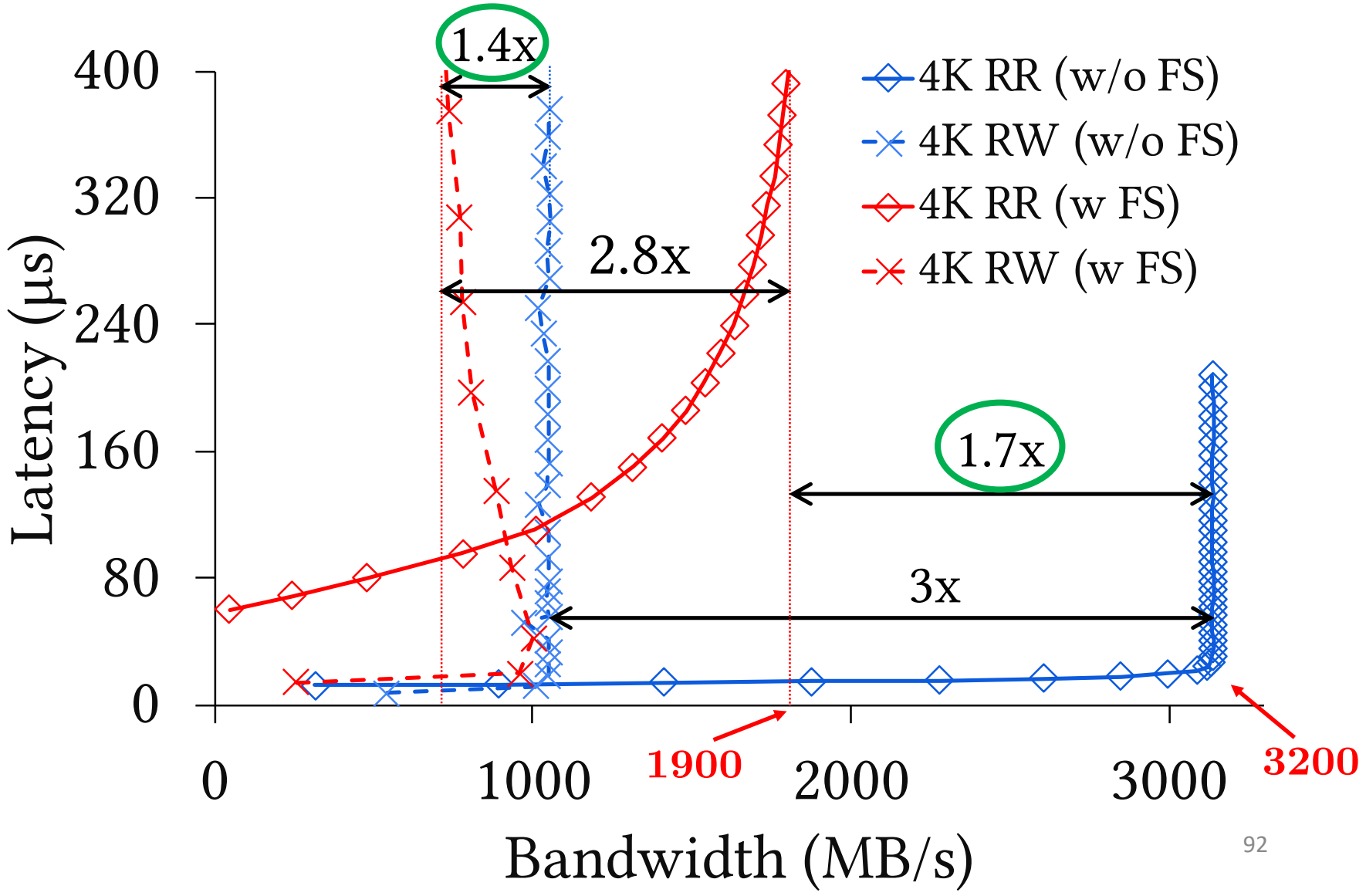
Impact of the File System

Device
PCIe SSD - P4510 (1TB)



Impact of the File System

Device
PCIe SSD - P4510 (1TB)



Impact of the File System

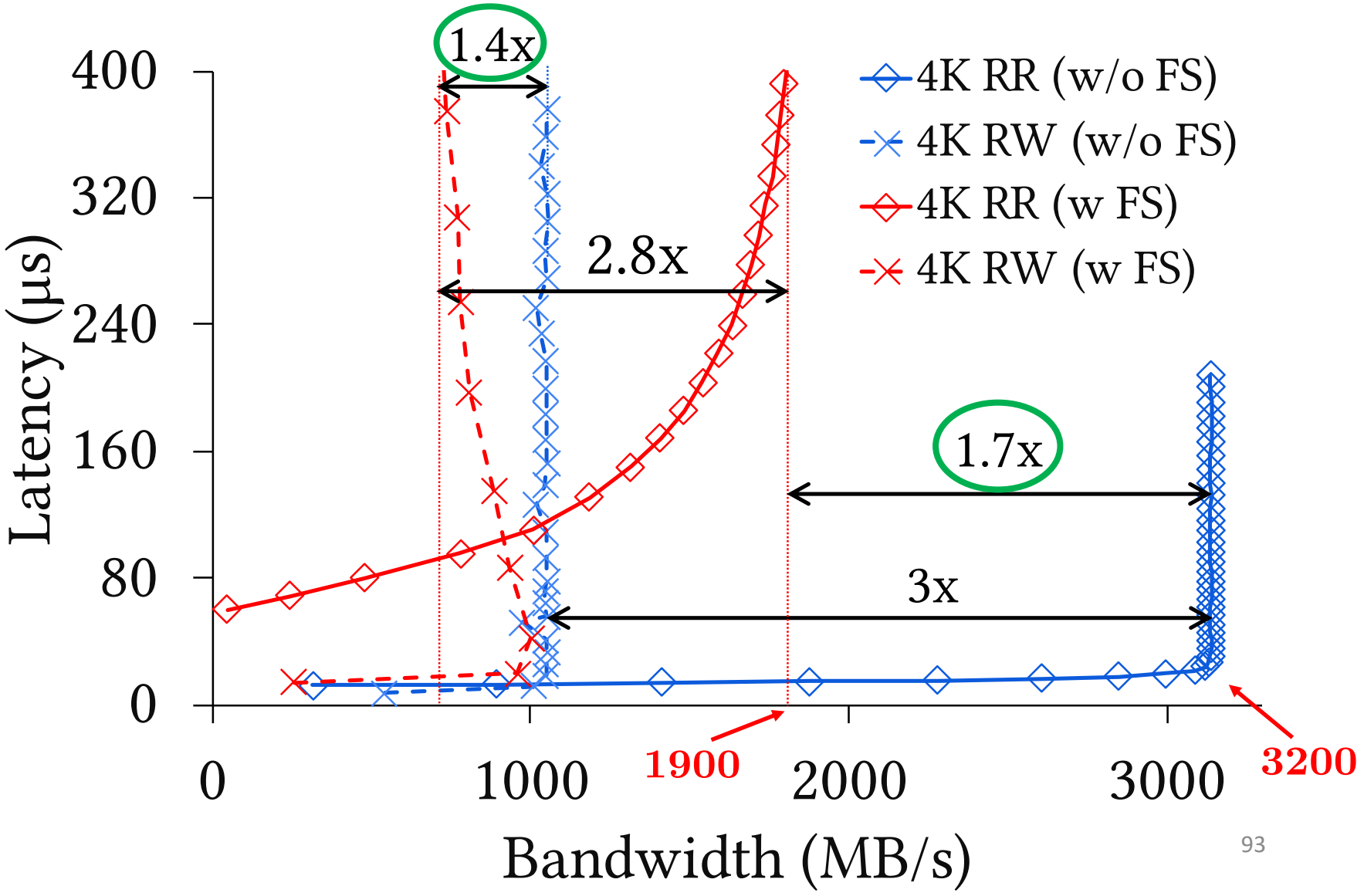
Device
PCIe SSD - P4510 (1TB)

FS affects α and k.

Much stable &

increased bandwidth

w/o the file system.



Empirical Asymmetry and Concurrency

Devices	4KB			8KB		
	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

Empirical Asymmetry and Concurrency

Devices	4KB			8KB		
	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**

Empirical Asymmetry and Concurrency

Devices	4KB			8KB		
	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

Performance Analysis following PIO (M, k, α)

Performance Analysis

We classify storage-intensive applications into three classes

- ✓ Batchable Writes
- ✓ Batchable Reads
- ✓ Batchable Reads and Writes

Batchable Writes

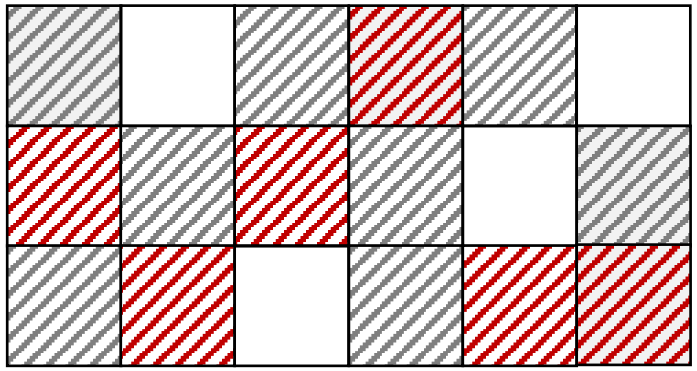
- ✓ Can exploit write concurrency (k_w) by batching writes

Batchable Writes

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

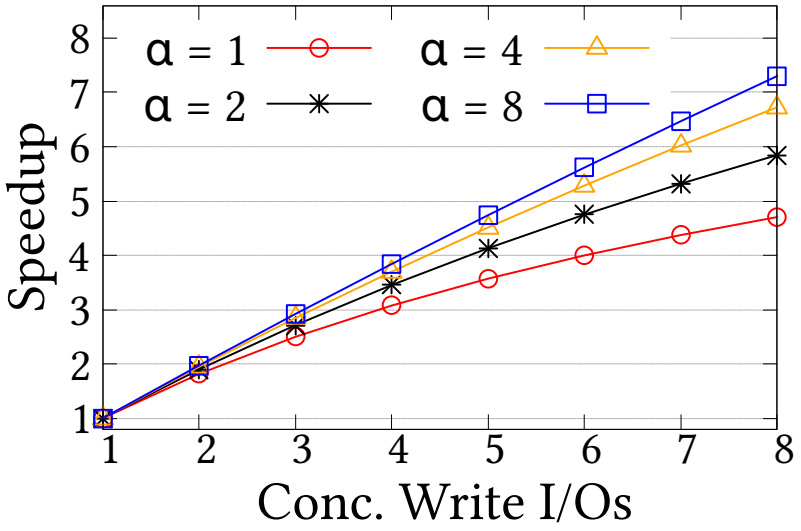
Batchable Writes

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

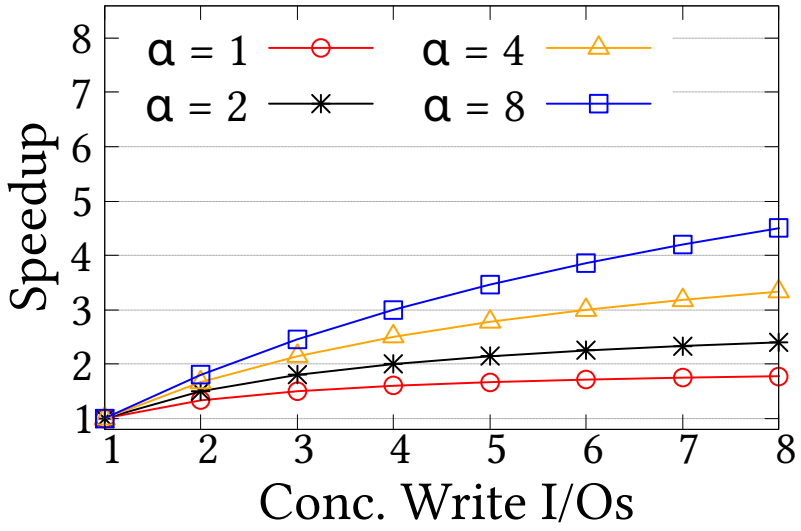


Batchable Writes

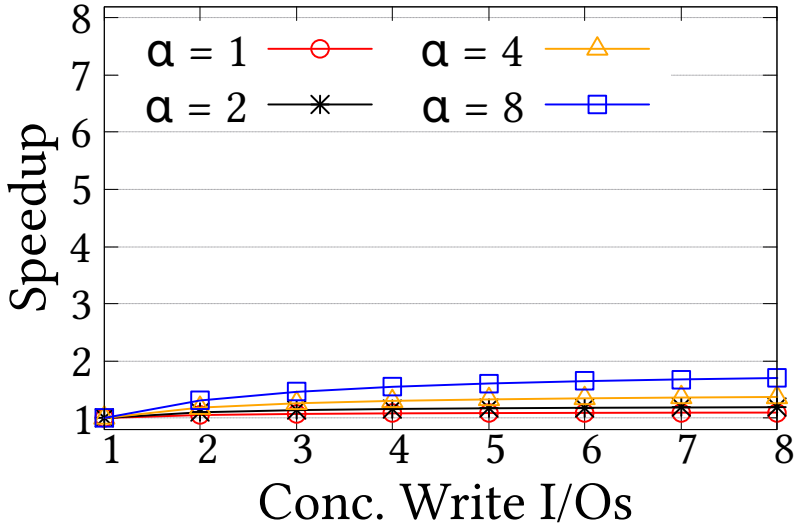
10% reads



50% reads

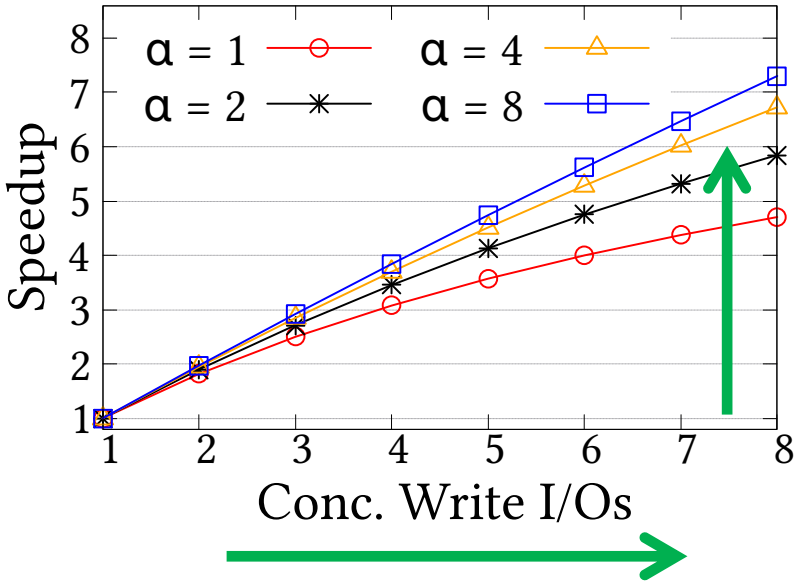


90% reads

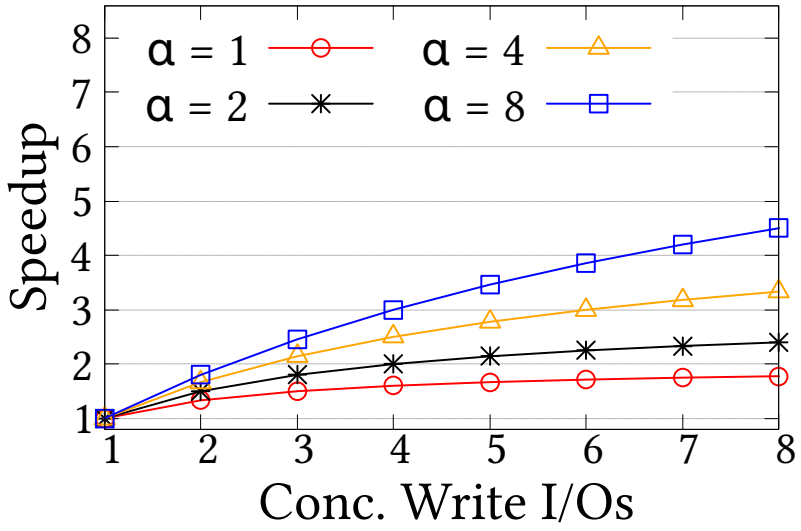


Batchable Writes

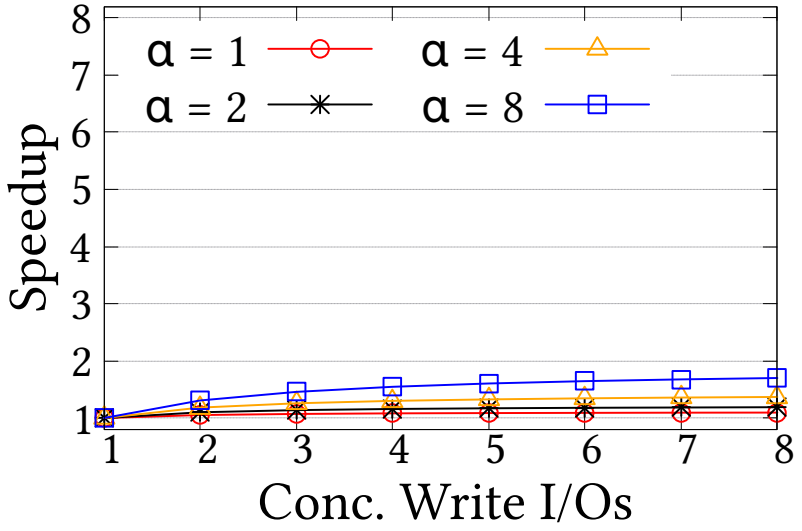
10% reads



50% reads



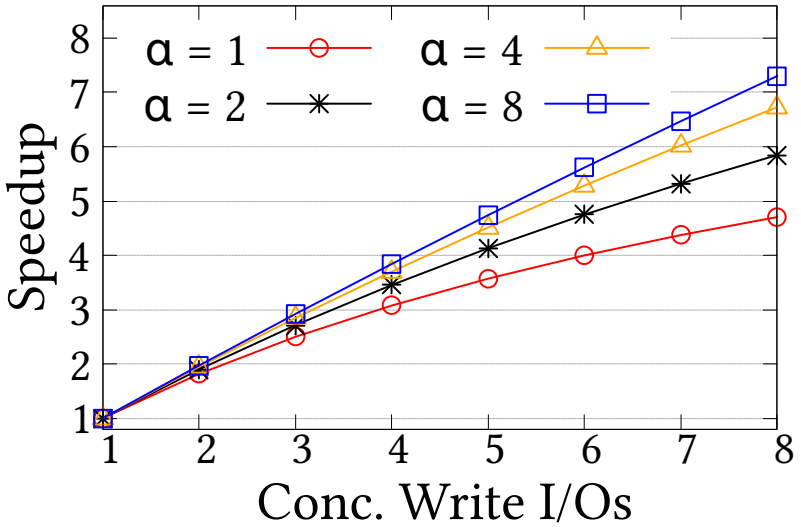
90% reads



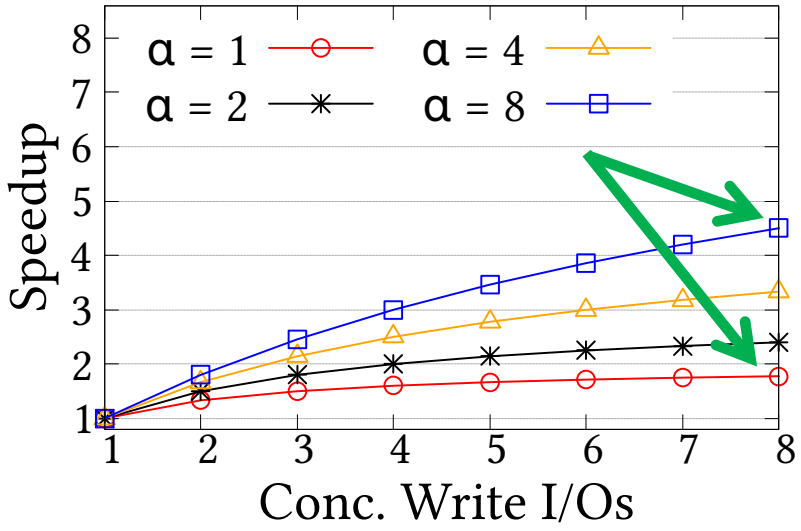
Speedup increases with increasing concurrent I/Os

Batchable Writes

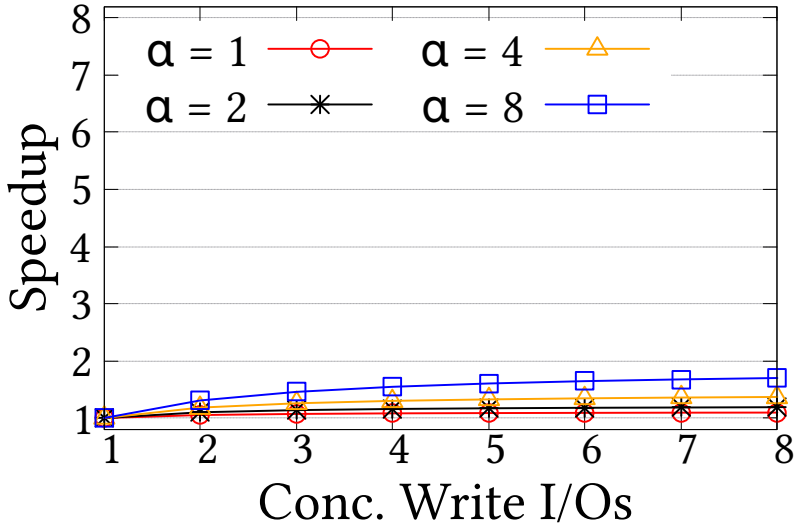
10% reads



50% reads



90% reads

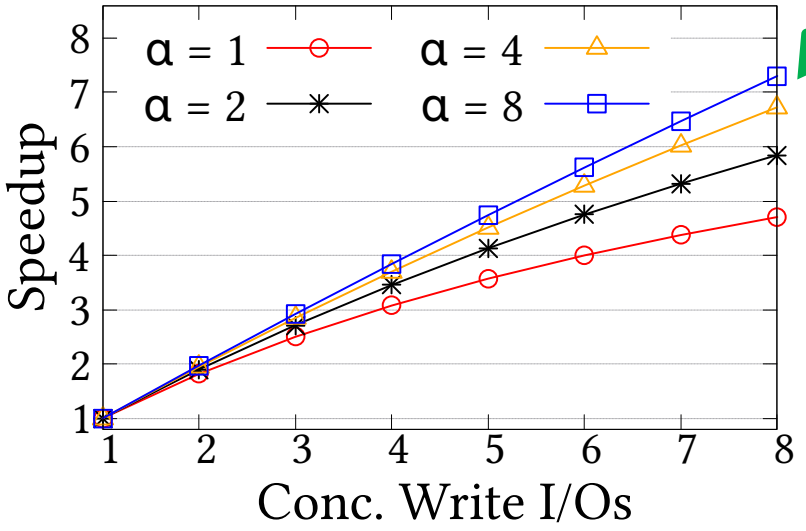


Speedup increases with increasing concurrent I/Os

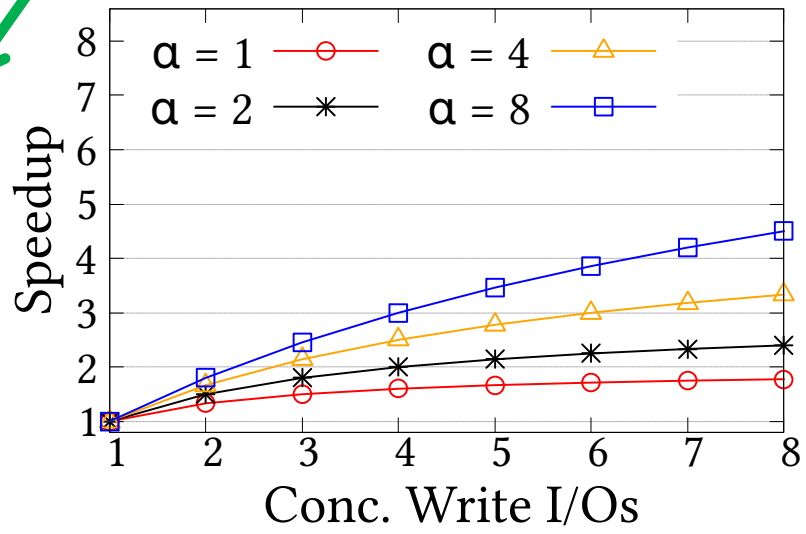
Speedup depends on asymmetry ($gain \propto \alpha$)

Batchable Writes

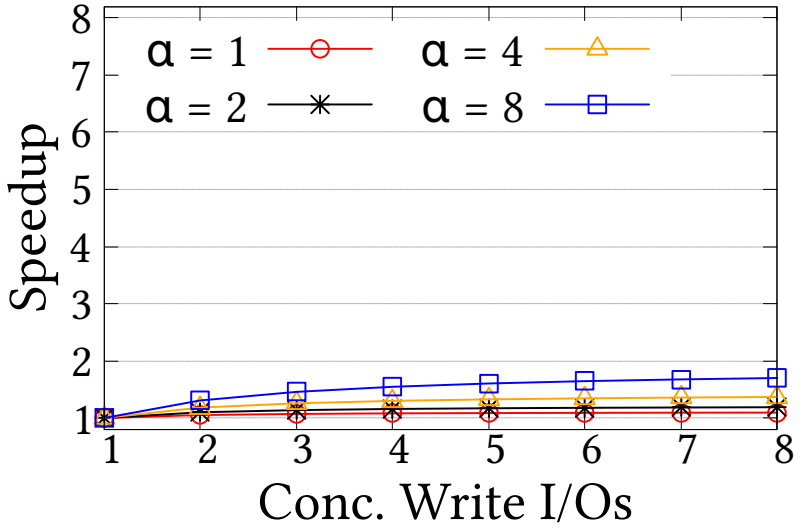
10% reads



50% reads



90% reads



Speedup increases with increasing concurrent I/Os

Speedup depends on asymmetry ($gain \propto \alpha$)

Speedup is highest for write-intensive workloads

Batchable Reads

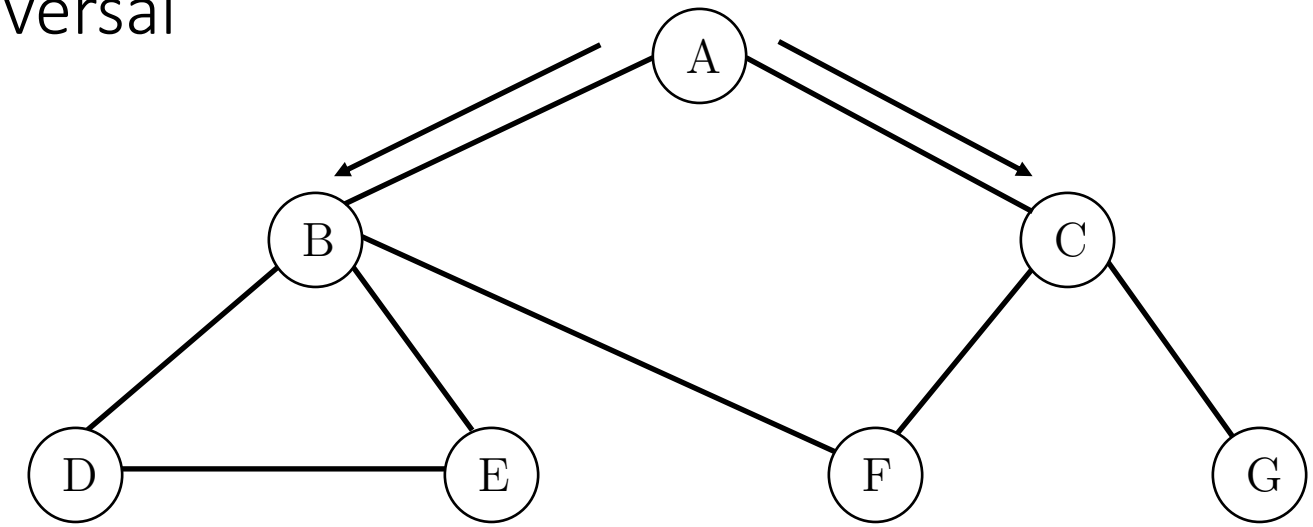
- ✓ Can exploit read concurrency (k_r) by batching reads

Batchable Reads

- ✓ Can exploit read concurrency (k_r) by batching reads
- ✓ Amortized cost per read following PIO is $\frac{1}{k_r}$

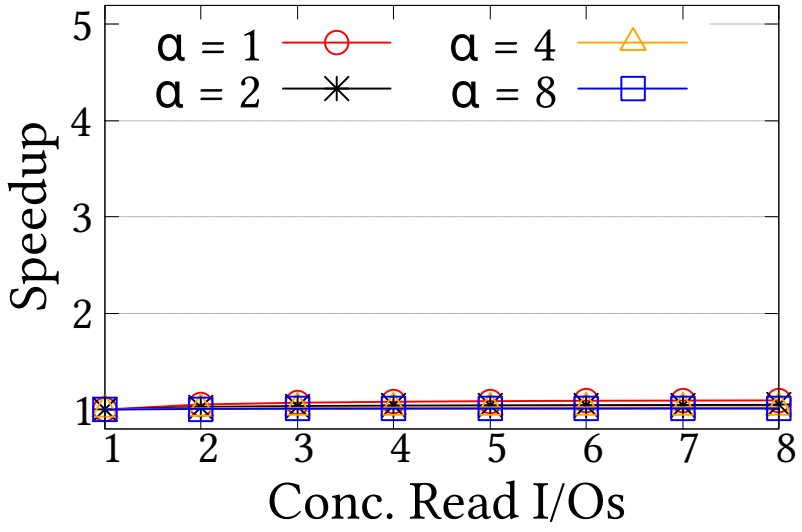
Batchable Reads

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

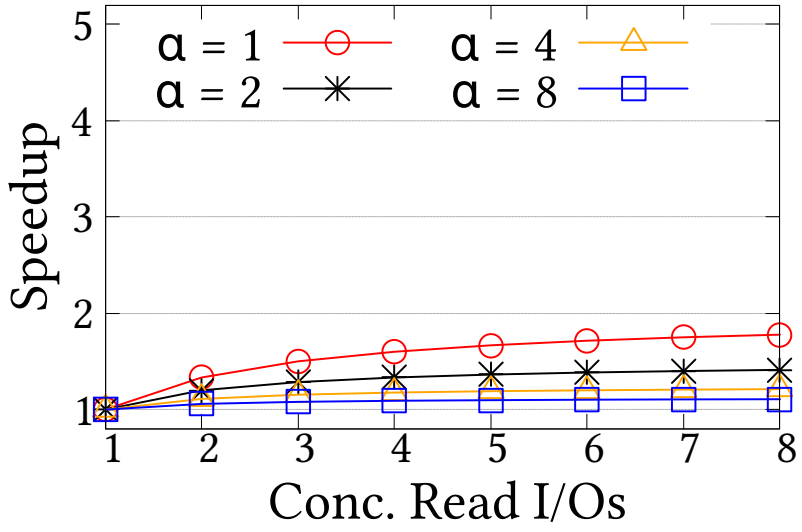


Batchable Reads

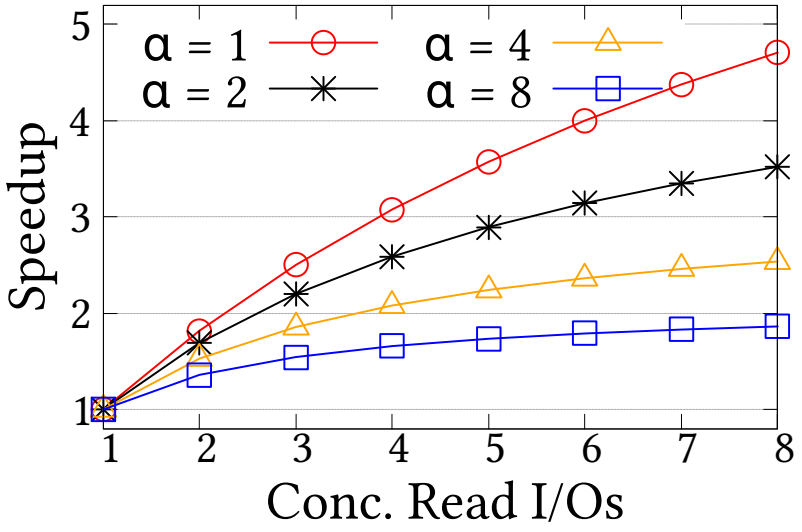
10% reads



50% reads

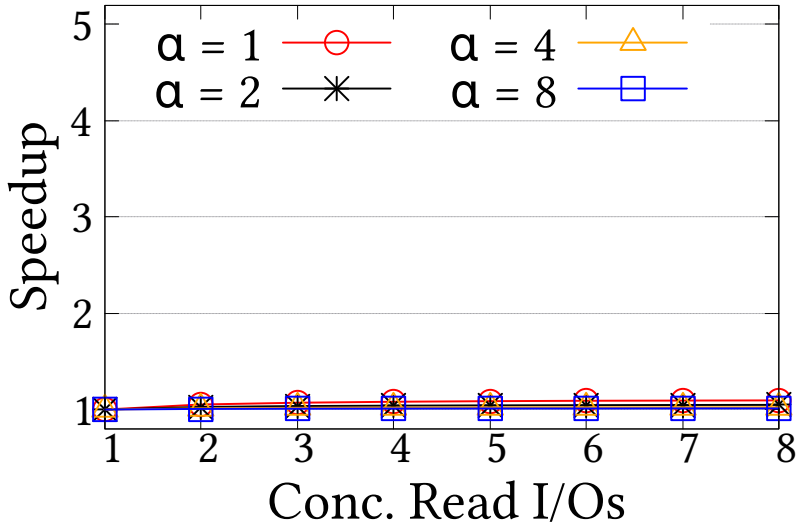


90% reads

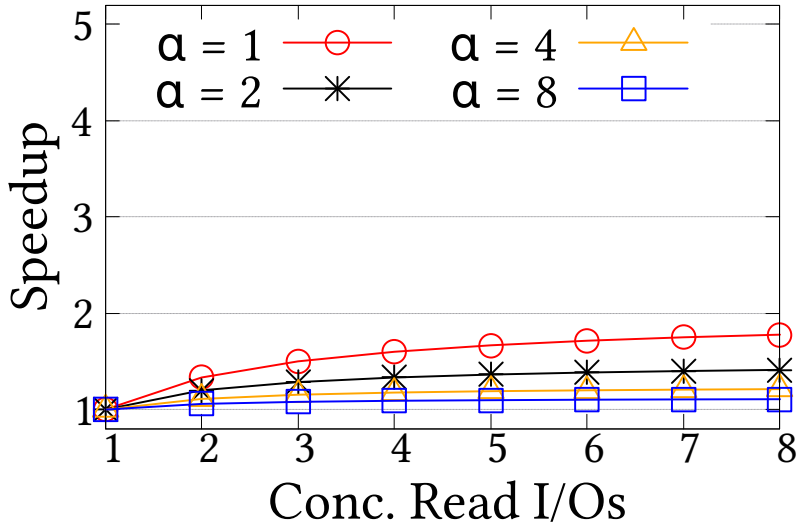


Batchable Reads

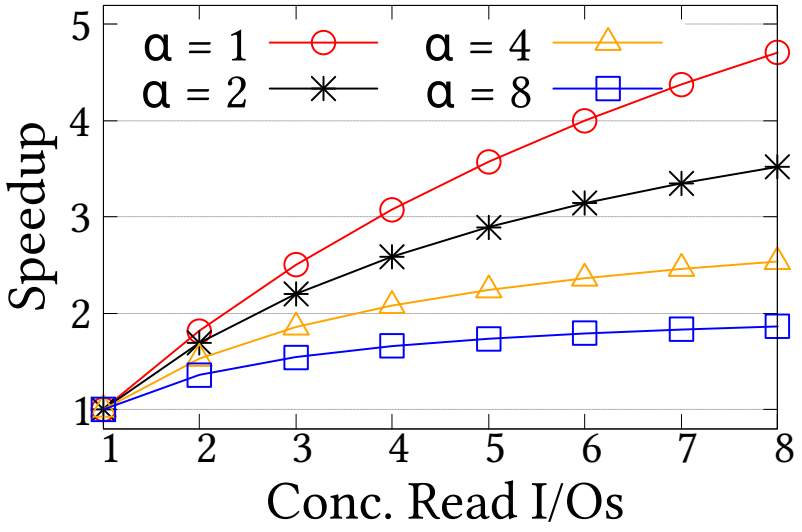
10% reads



50% reads



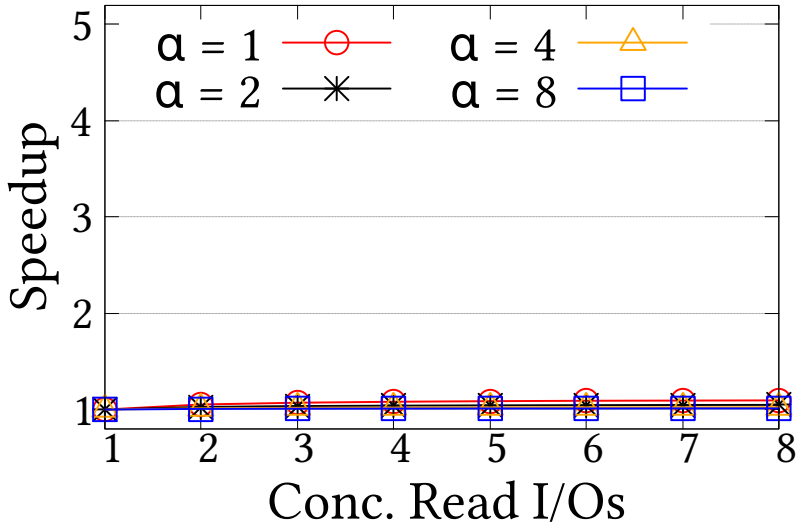
90% reads



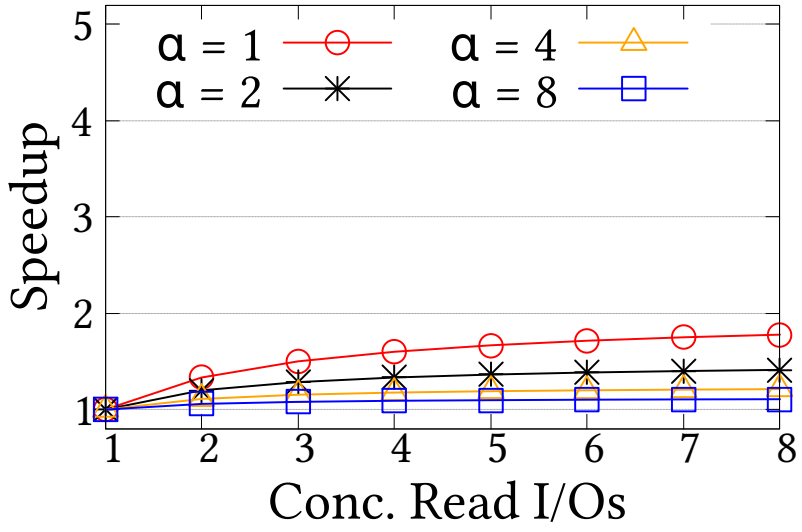
Speedup increases with increasing concurrent I/Os

Batchable Reads

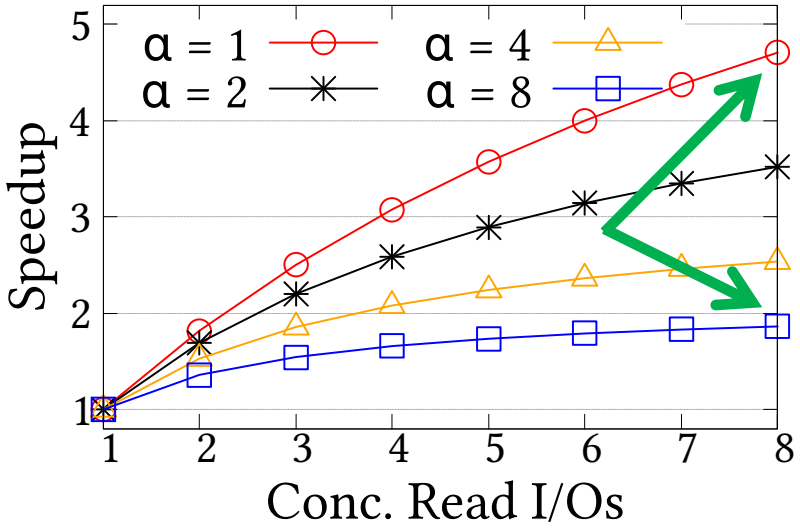
10% reads



50% reads



90% reads

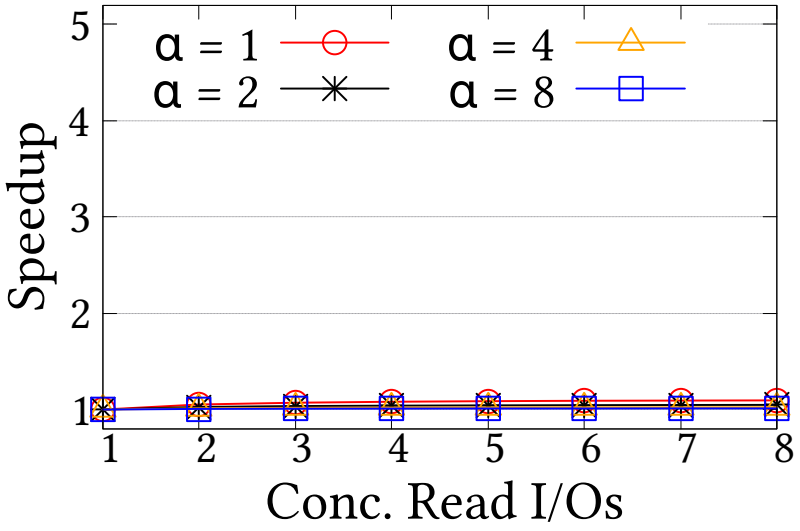


Speedup increases with increasing concurrent I/Os

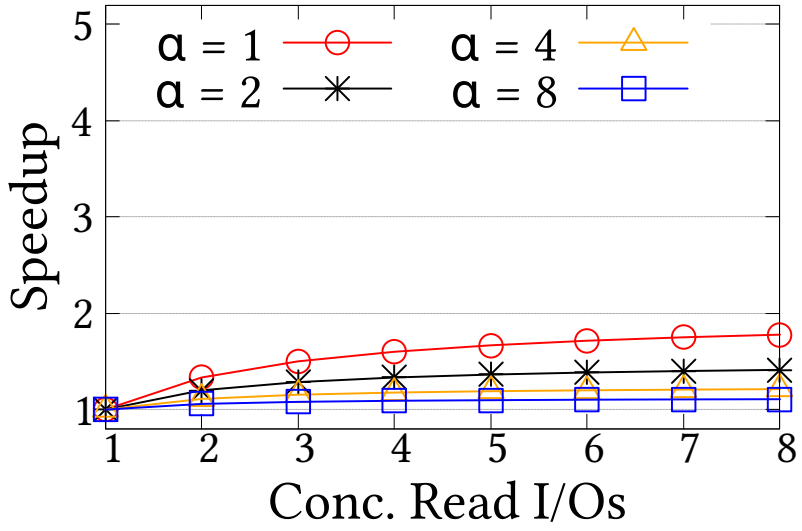
Speedup depends on asymmetry ($gain \propto {}^{1}/_{\alpha}$)

Batchable Reads

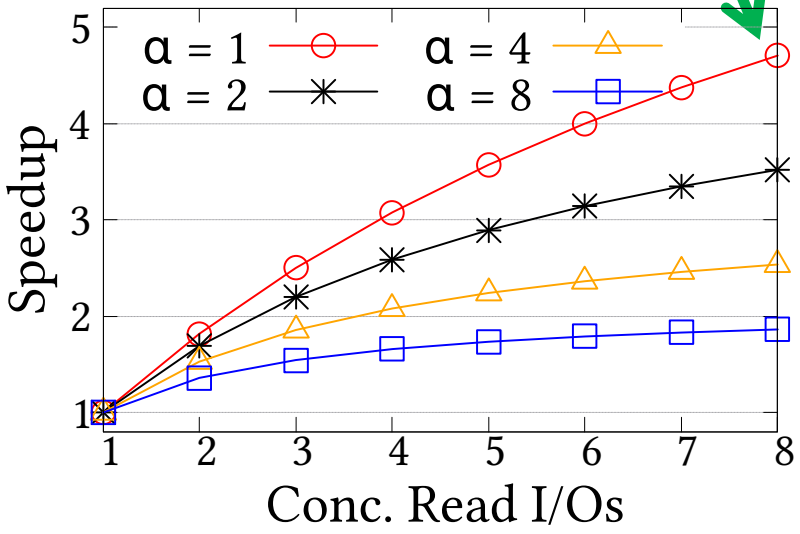
10% reads



50% reads



90% reads



Speedup increases with increasing concurrent I/Os

Speedup depends on asymmetry ($gain \propto {}^{1}/_{\alpha}$)

Speedup is highest for read-intensive workloads

Batchable Reads & Writes

- ✓ Can exploit both read and write concurrency (k_r , k_w)

Batchable Reads & Writes

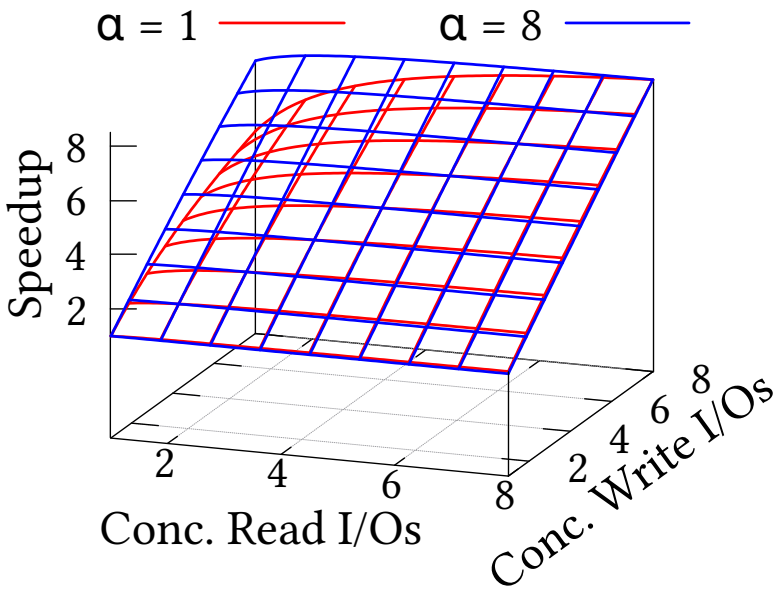
- ✓ Can exploit both read and write concurrency (k_r , k_w)
- ✓ Amortized cost per read following PIO is $\frac{1}{k_r}$
- ✓ Amortized cost per write following PIO is $\frac{\alpha}{k_w}$

Batchable Reads & Writes

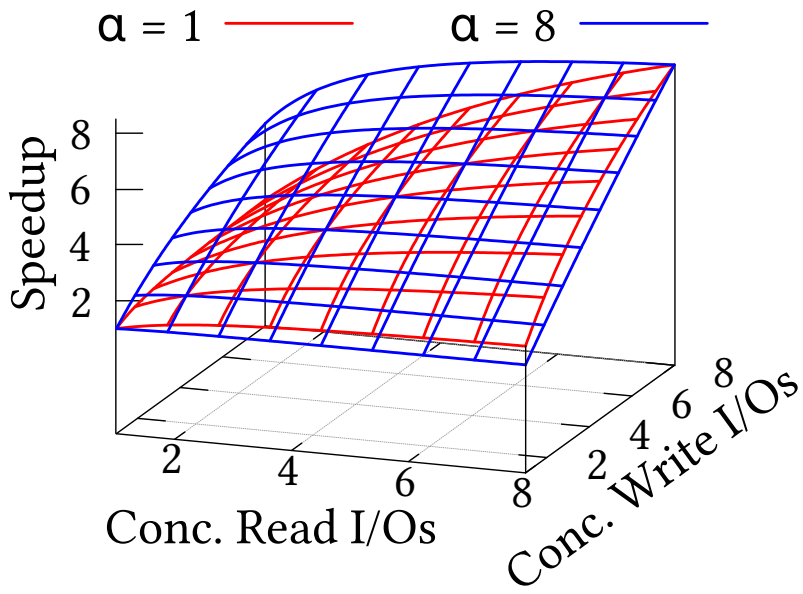
- ✓ Can exploit both read and write concurrency (k_r , k_w)
- ✓ Amortized cost per read following PIO is $\frac{1}{k_r}$
- ✓ Amortized cost per write following PIO is $\frac{\alpha}{k_w}$
- ✓ Example: LSM compaction

Batchable Reads & Writes

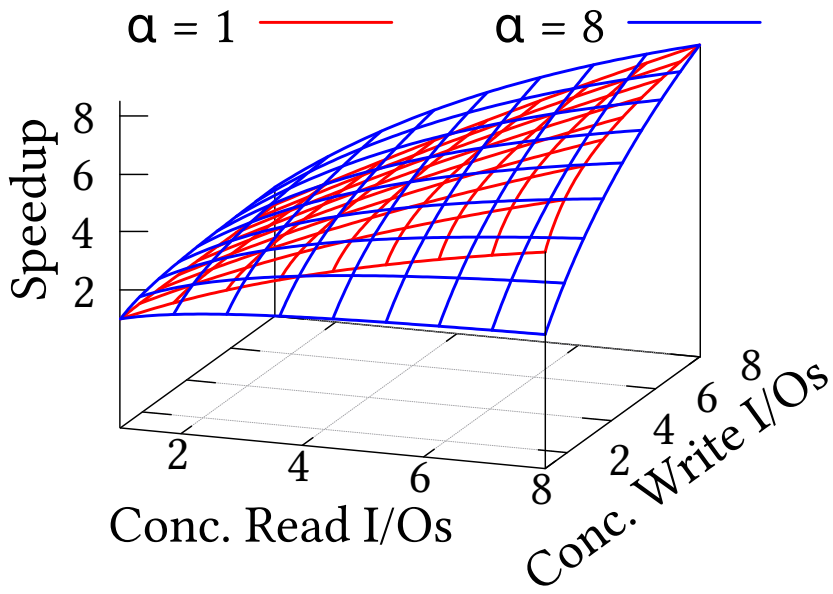
10% reads



50% reads

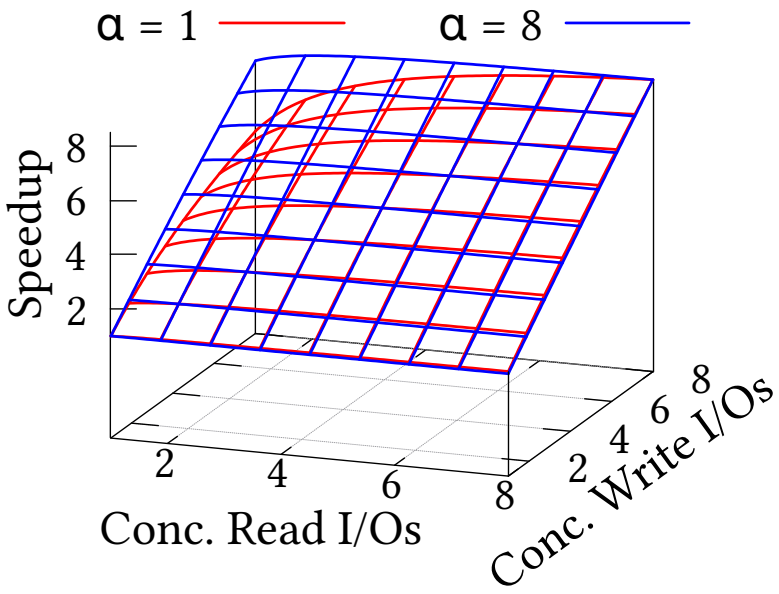


90% reads

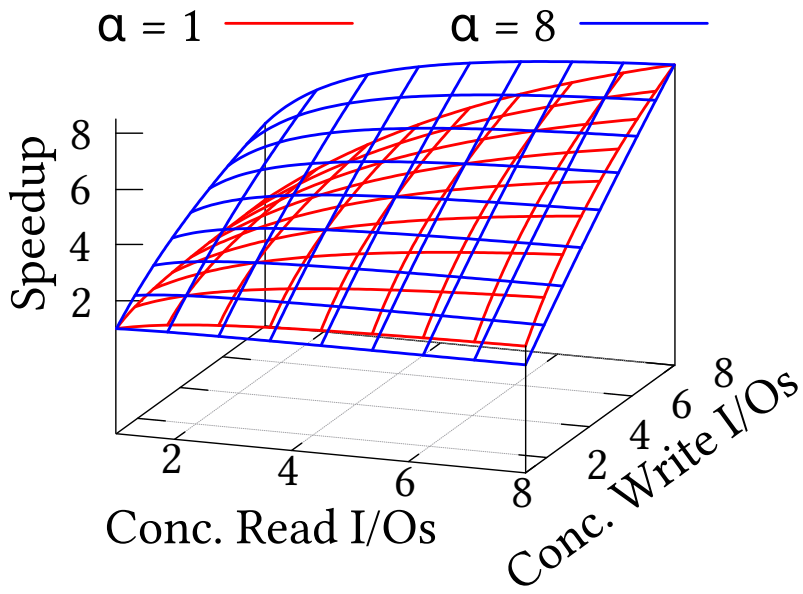


Batchable Reads & Writes

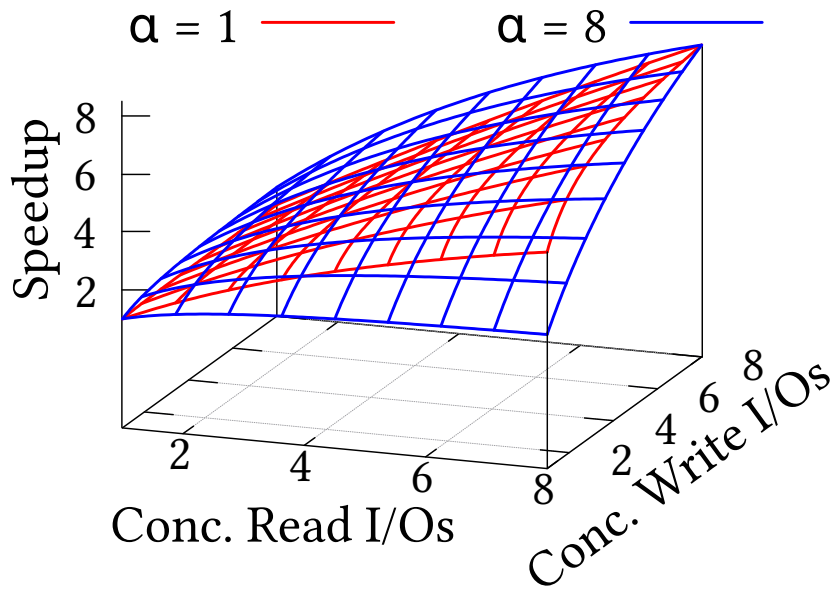
10% reads



50% reads



90% reads

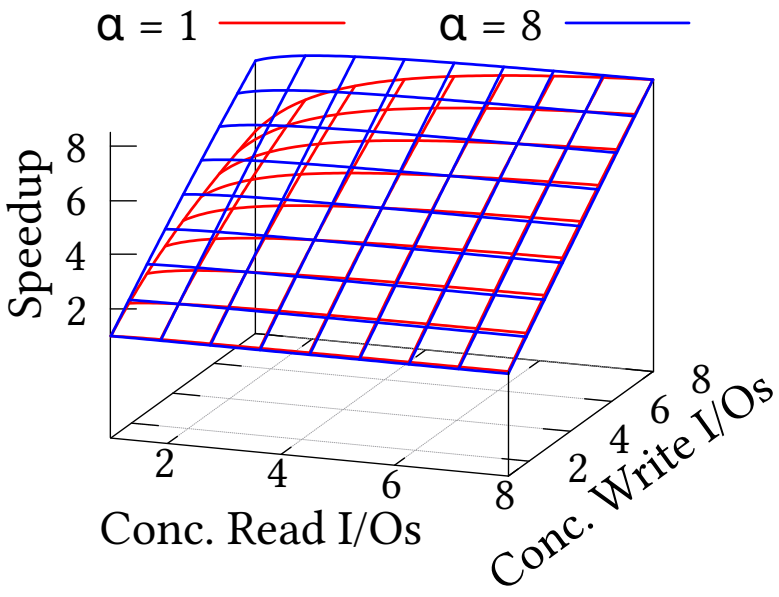


Speedup increases with increasing concurrent I/Os

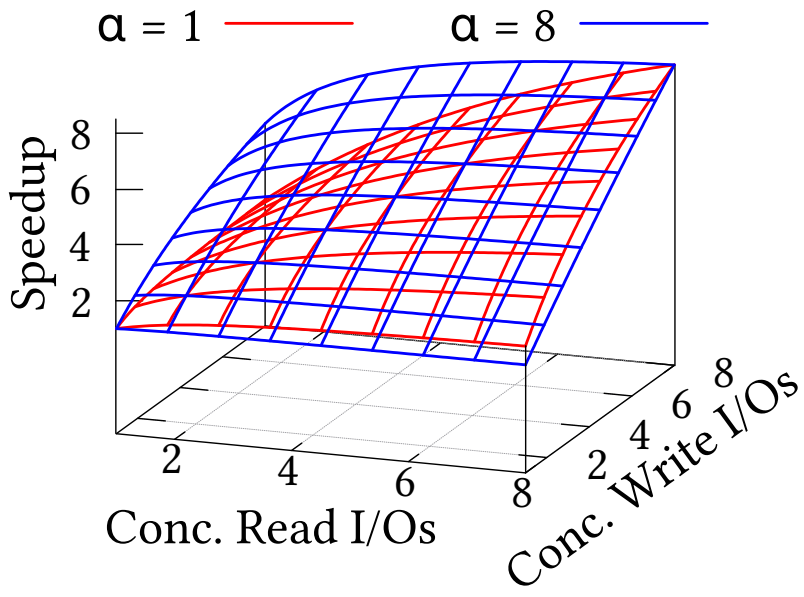
Speedup depends on asymmetry

Batchable Reads & Writes

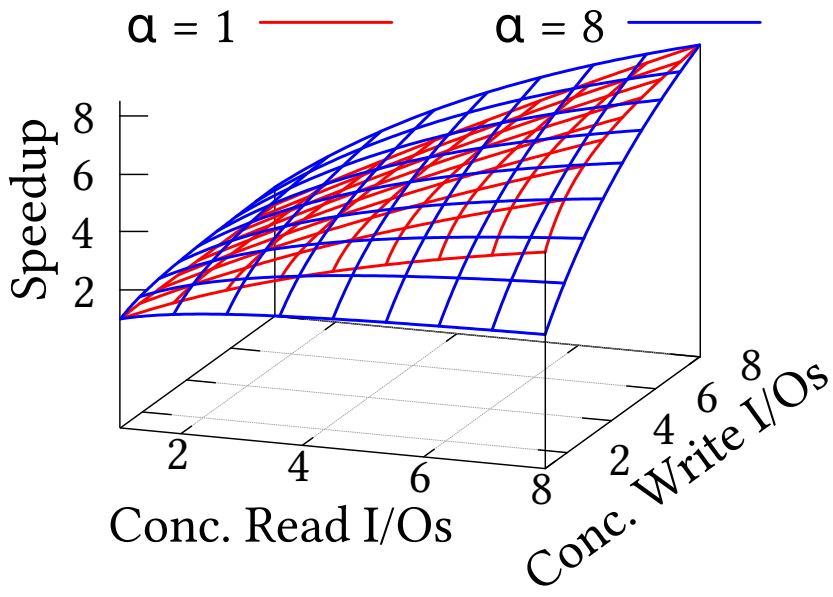
10% reads



50% reads



90% reads



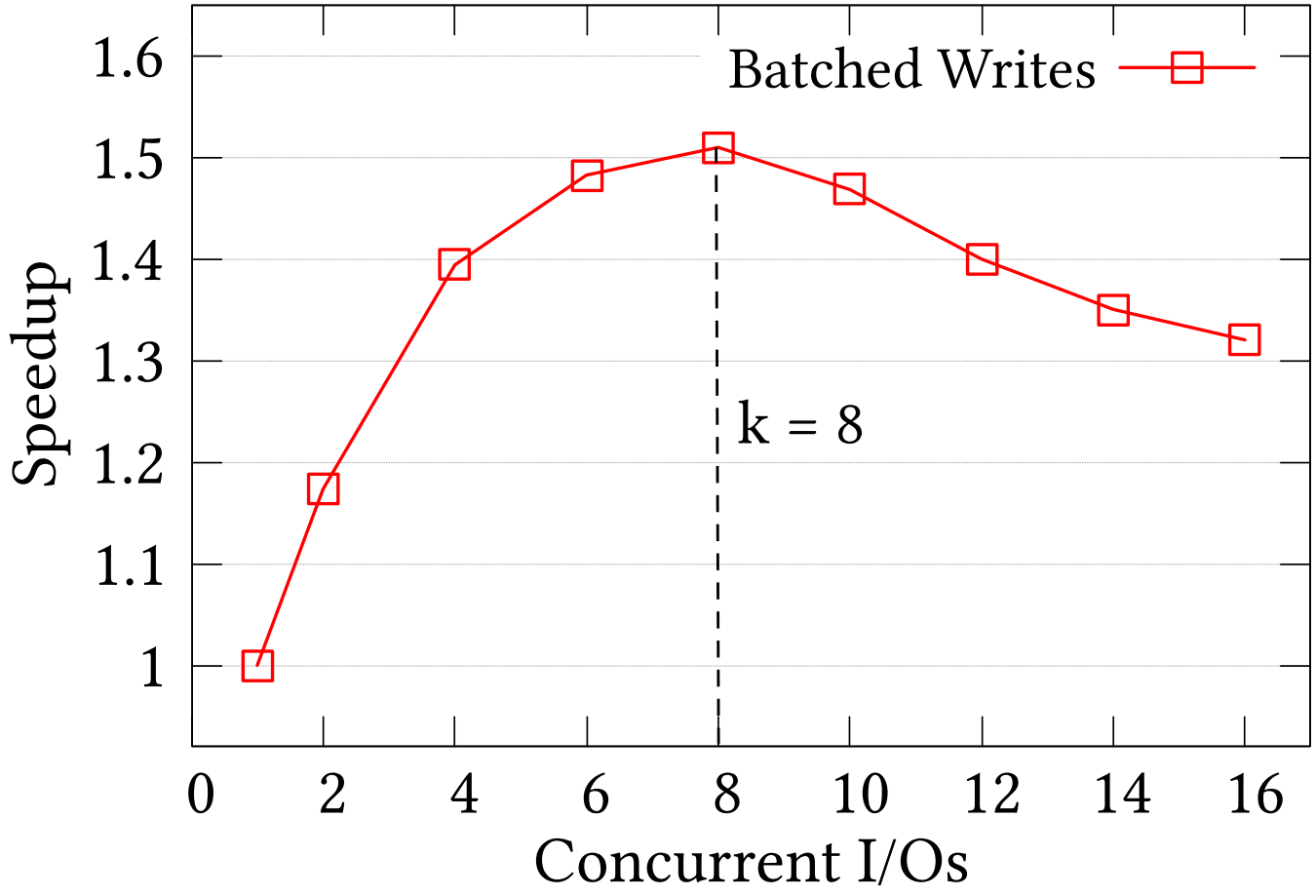
Speedup increases with increasing concurrent I/Os

Speedup depends on asymmetry

Impact of utilizing write concurrency is higher

Importance of using Proper k

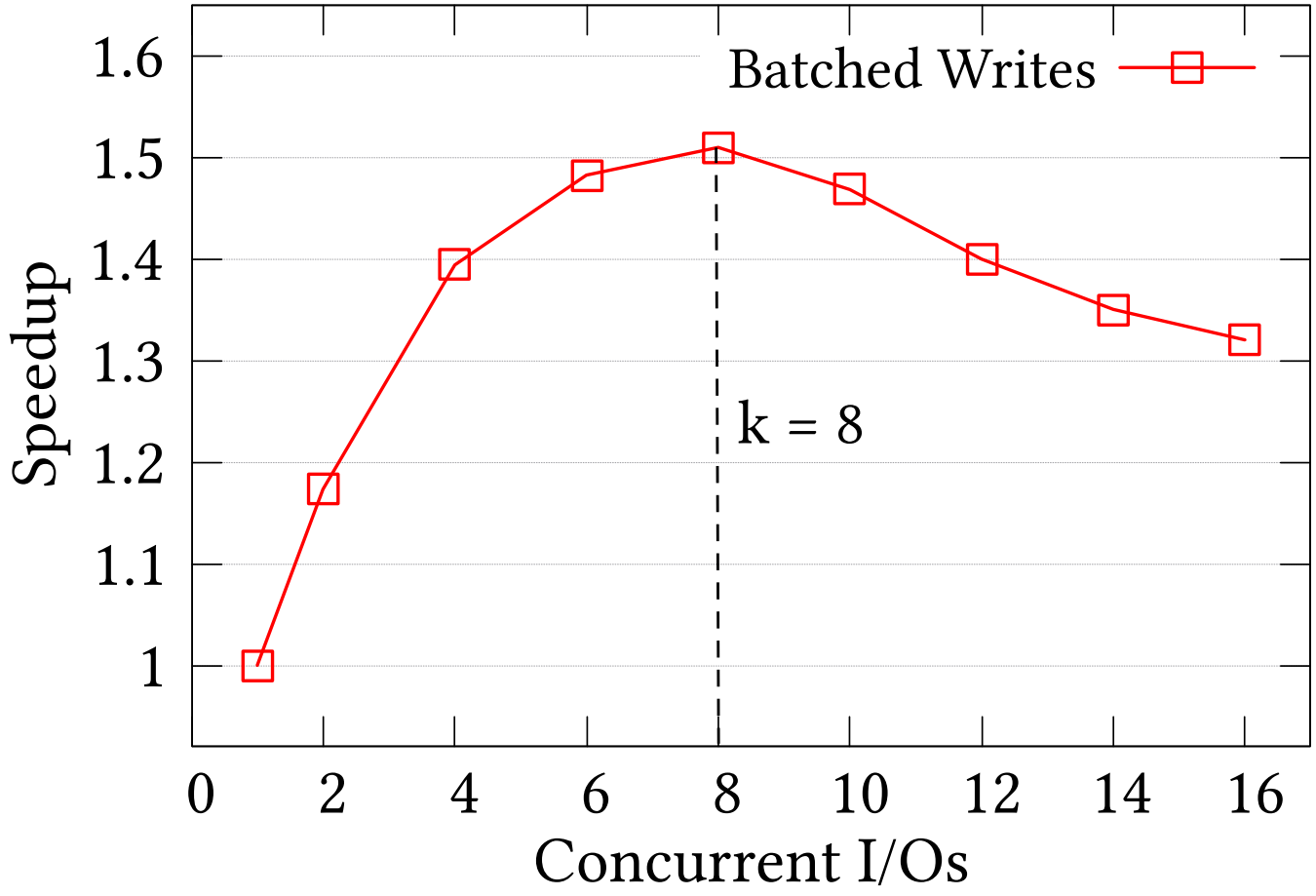
Sample application: batchable writes



Importance of using Proper k

Sample application: batchable writes

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

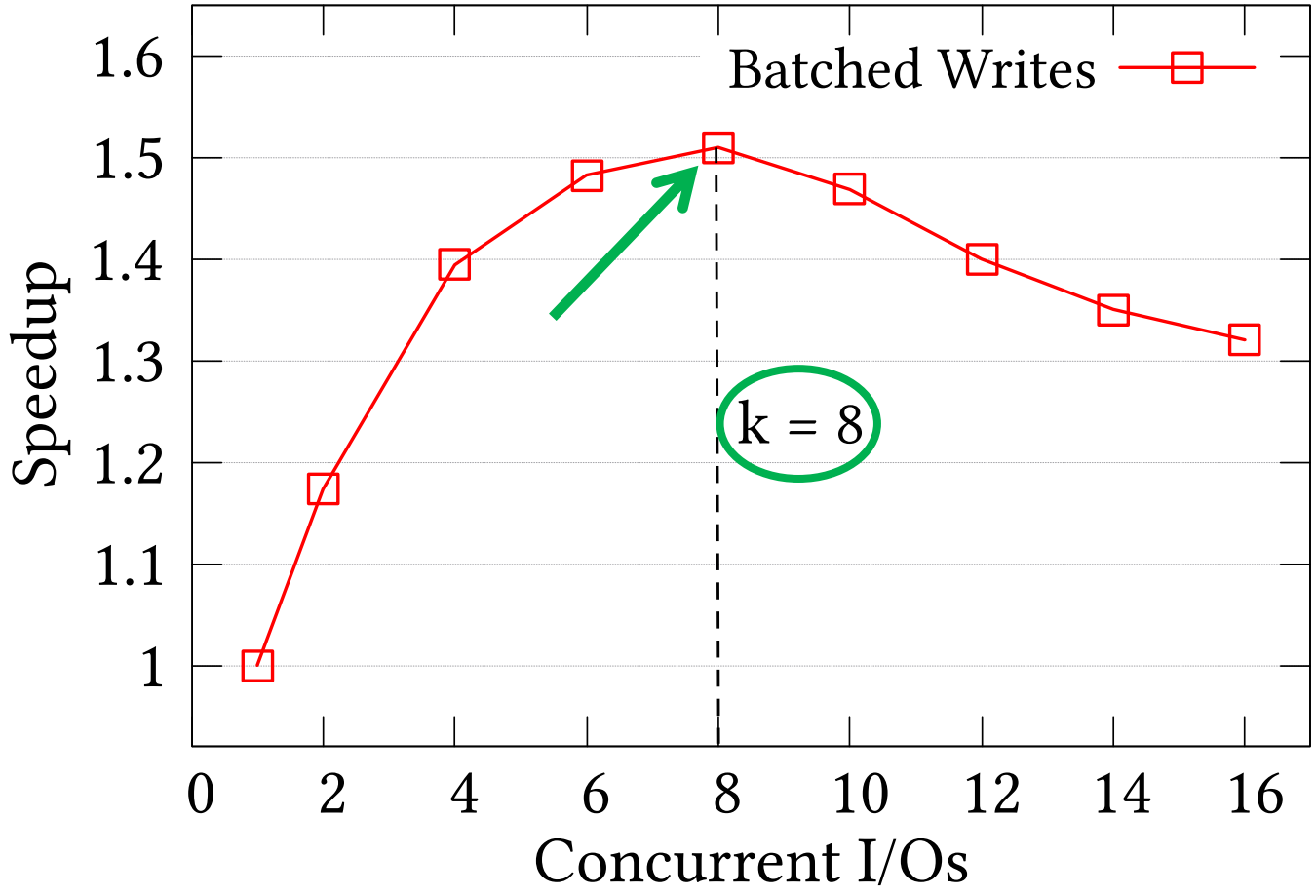


Importance of using Proper k

Sample application: batchable writes

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

Optimal speedup at the device concurrency.



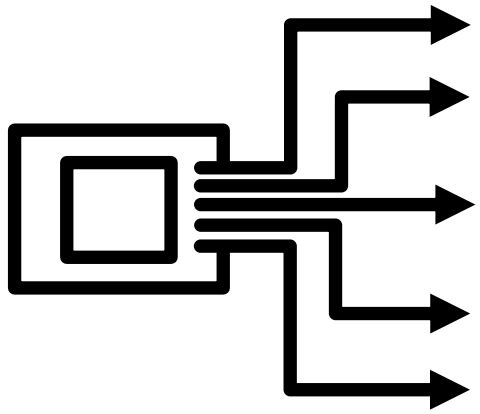
Guidelines for Algorithm Design

Guidelines for Algorithm Design

Know Thy Device

Guidelines for Algorithm Design

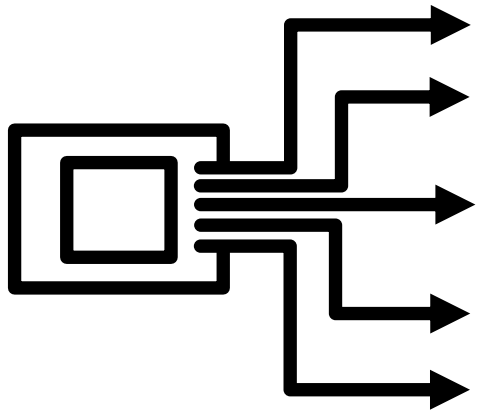
Know Thy Device



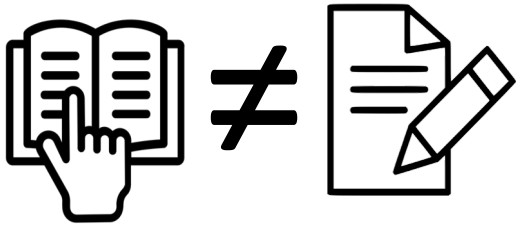
Exploit concurrency
(with care)

Guidelines for Algorithm Design

Know Thy Device



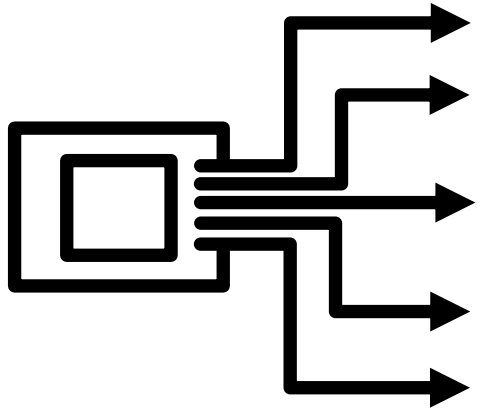
Exploit concurrency (with care)



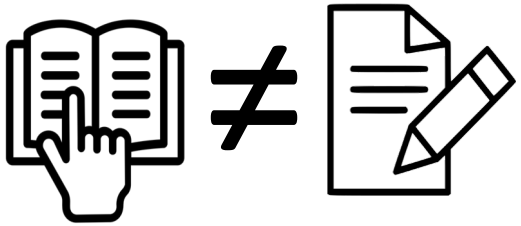
Treat read and write differently.

Guidelines for Algorithm Design

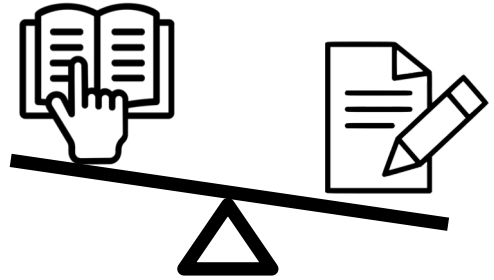
Know Thy Device



Exploit concurrency (with care)

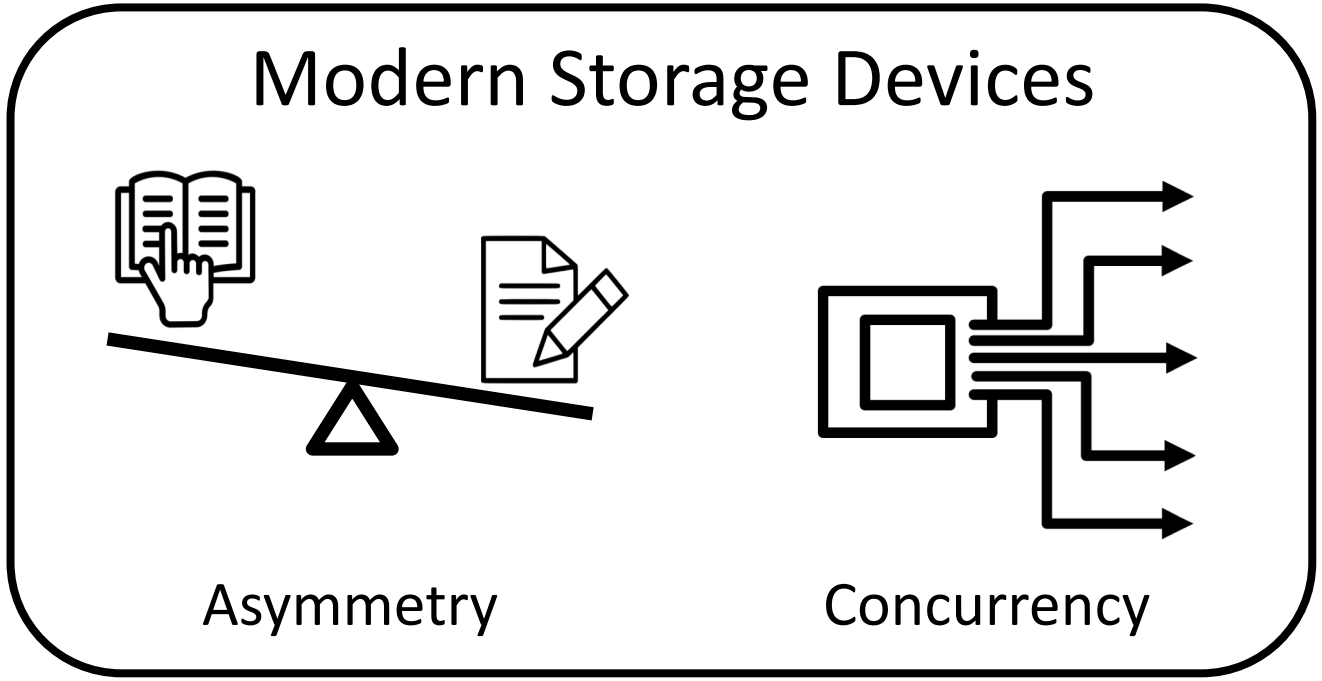


Treat read and write differently.

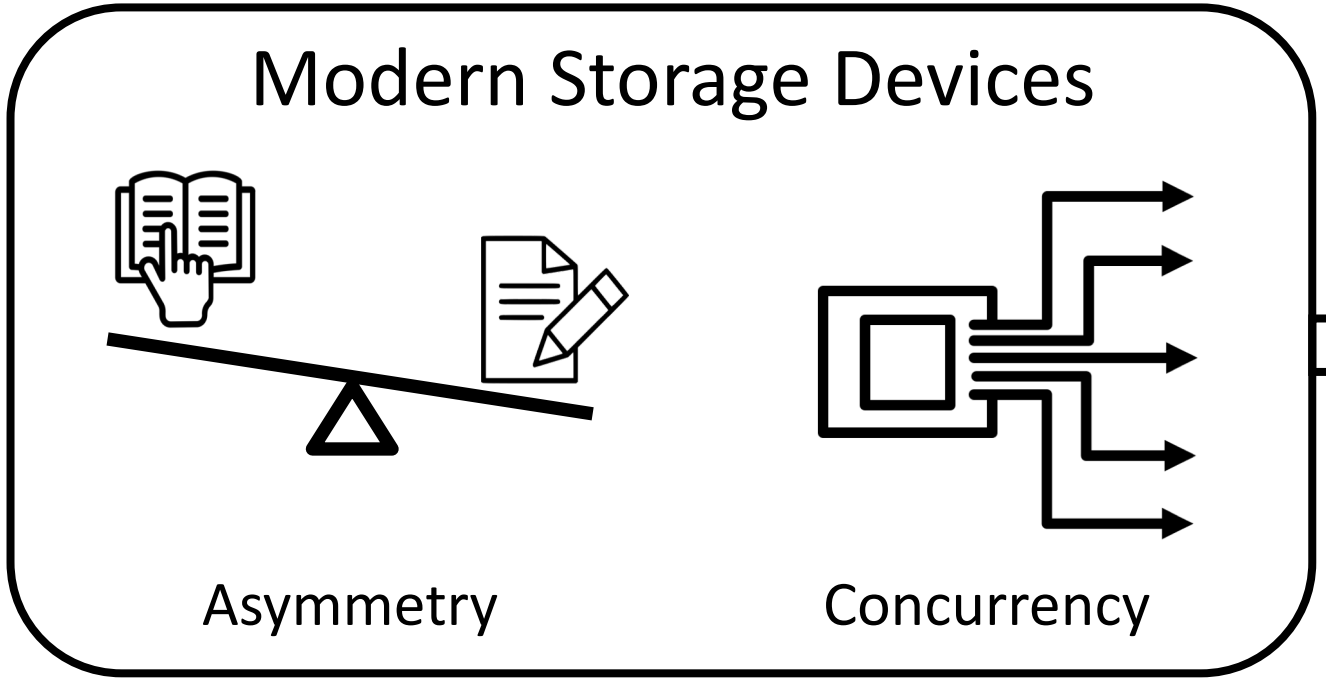


α controls performance

Conclusion & Future Work



Conclusion & Future Work



Need for a new parametric I/O model

PIO (M, k, α)

Conclusion & Future Work

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

… not simply an engineering optimization

Conclusion & Future Work

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

Conclusion & Future Work

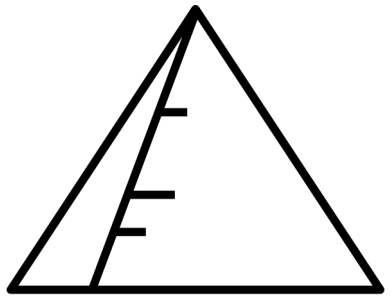
Make *asymmetry and concurrency* part of *algorithm design*

… not simply an engineering optimization

Build algorithms/data structures for storage devices

with asymmetry α and concurrency k

index structures



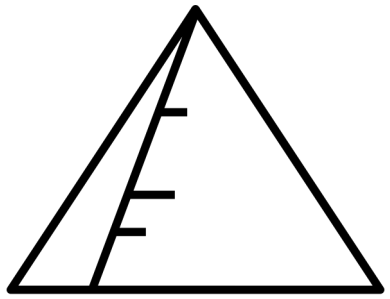
Conclusion & Future Work

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

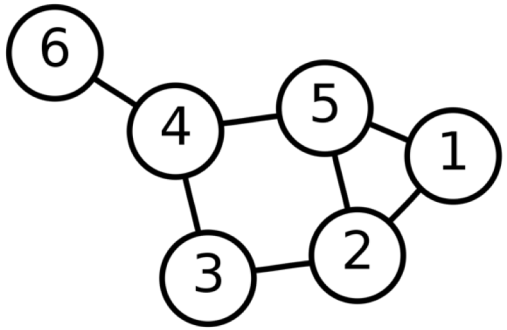
… not simply an engineering optimization

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

index structures



graph traversal algorithms



Conclusion & Future Work

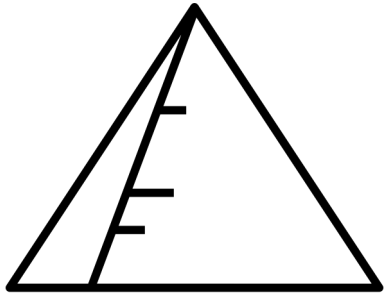
Make *asymmetry and concurrency* part of *algorithm design*

… not simply an engineering optimization

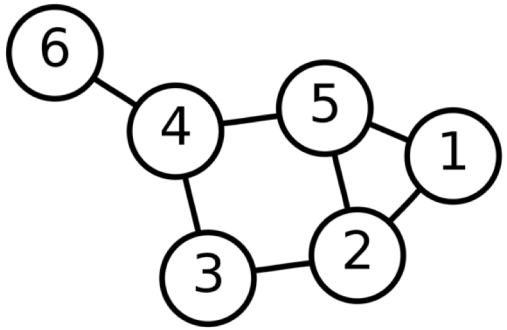
Build algorithms/data structures for storage devices

with asymmetry α and concurrency k

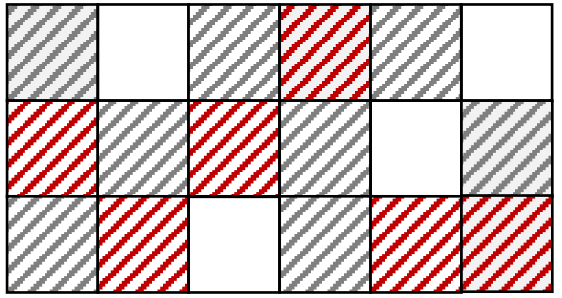
index structures



graph traversal algorithms



bufferpool management



Conclusion & Future Work

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

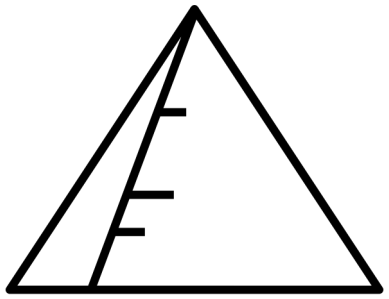
… not simply an engineering optimization

Build algorithms/data structures for storage devices

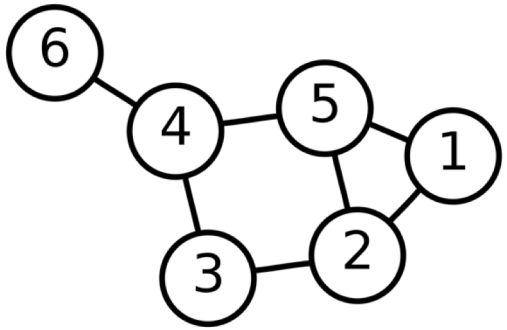
with asymmetry α and concurrency k

Stay Tuned!

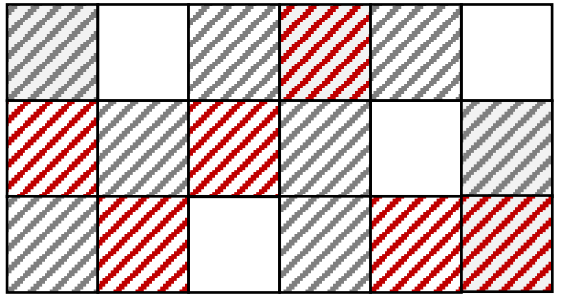
index structures



graph traversal algorithms



bufferpool management



Thank You!

disc.bu.edu/pio