A Parametric I/O Model for Modern Storage Devices

Tarikul Islam Papon, Manos Athanassoulis

Overview

Why an I/O model?

Why not the traditional I/O model?

Asymmetry & Concurrency of Modern Storage Devices

The parametric I/O model

A [much better] bufferpool policy

Experimental Evaluation

Future work & Conclusions

Overview

Why an I/O model?

Why not the traditional I/O model?

Asymmetry & Concurrency of Modern Storage Devices

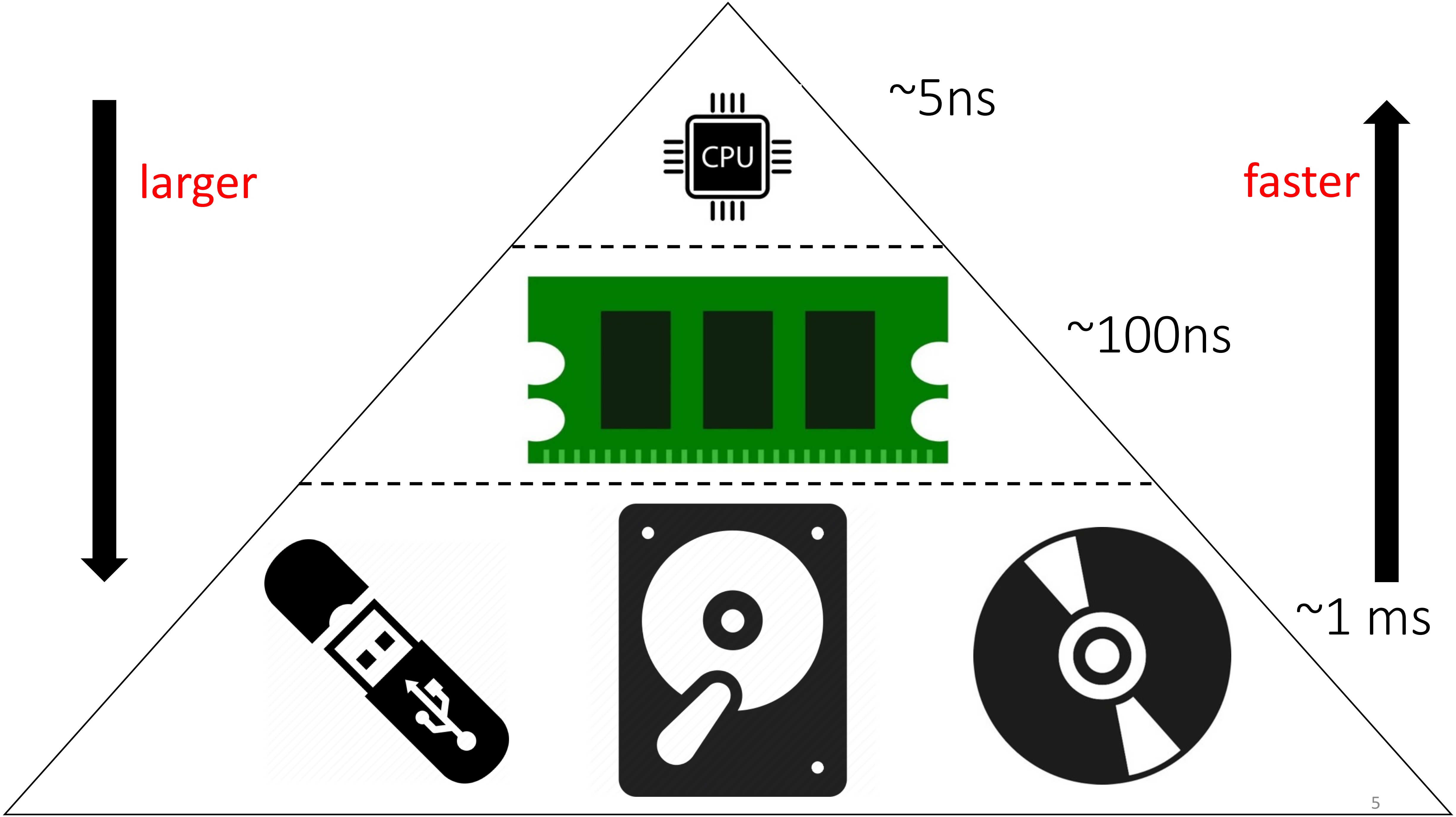
The parametric I/O model

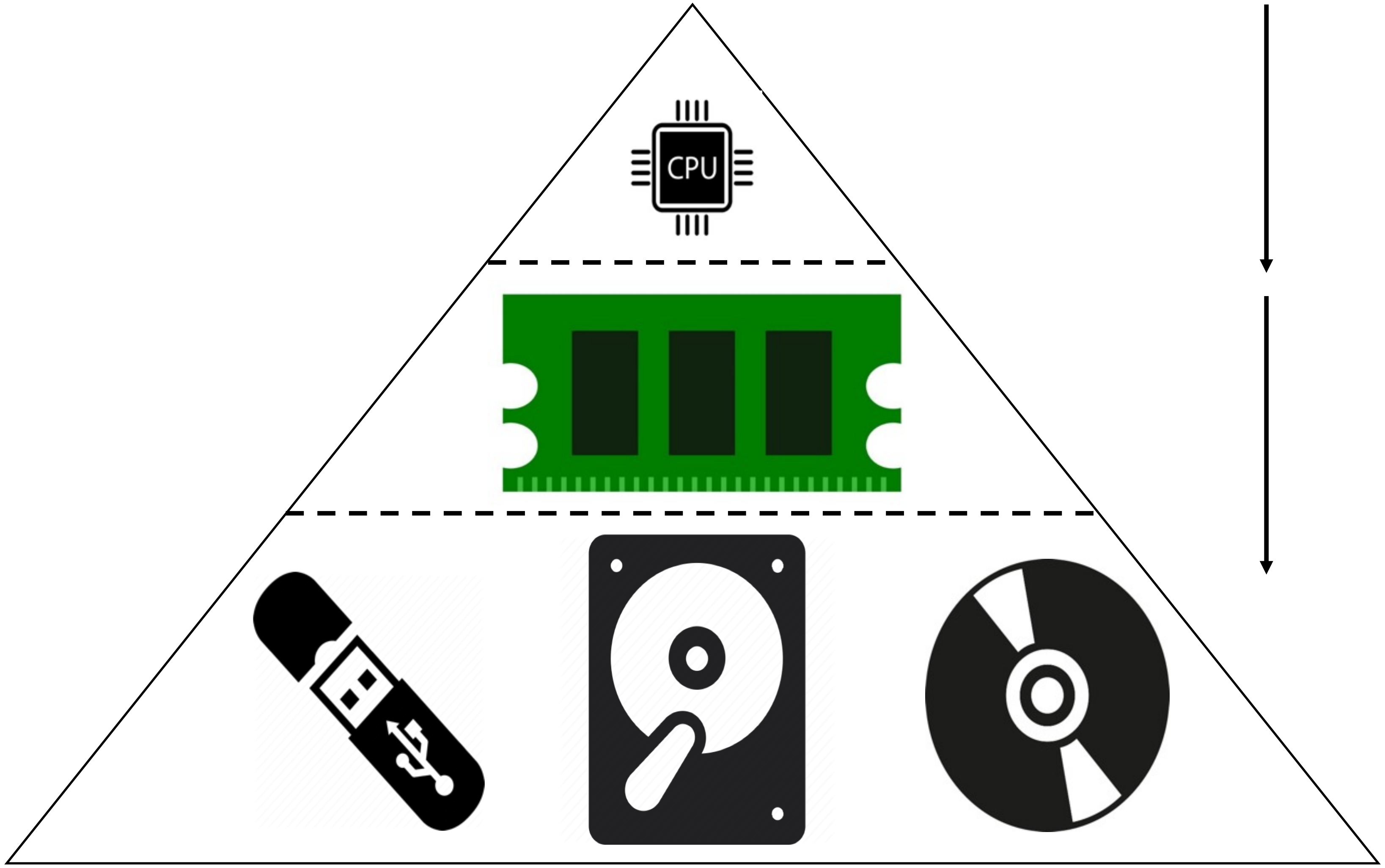
A [much better] bufferpool policy

Experimental Evaluation

Future work & Conclusions

Why an I/O Model?


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

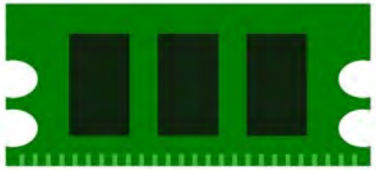

CPU


STOP

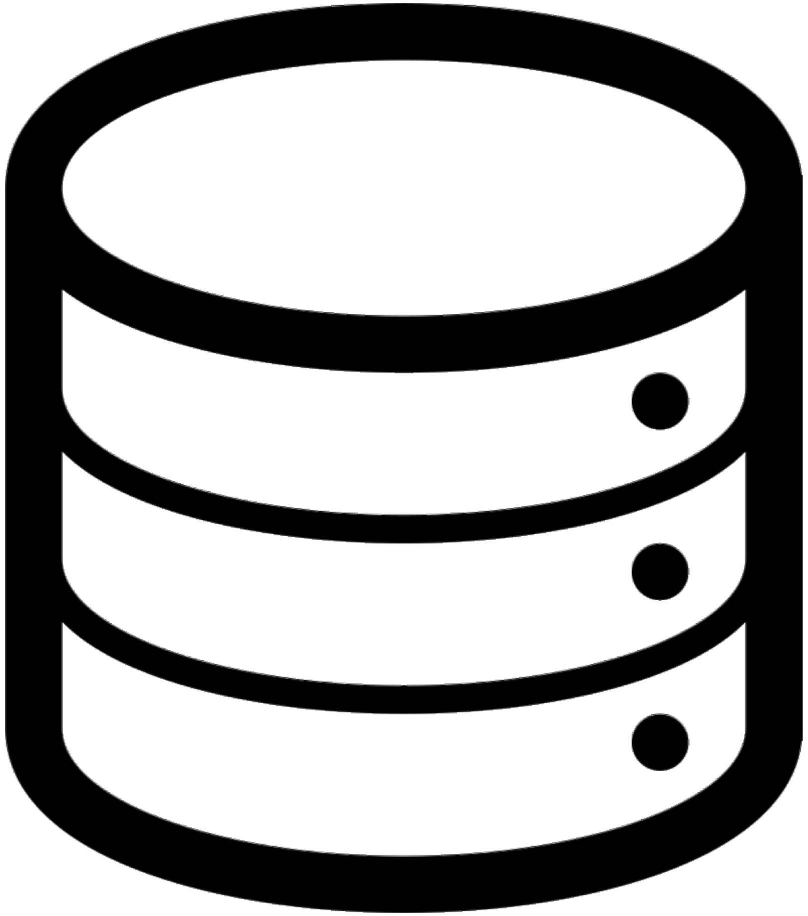
Traditional I/O Model

Traditional I/O Model (EM Model)

One I/O at a time



Small, fast main memory
(size M)



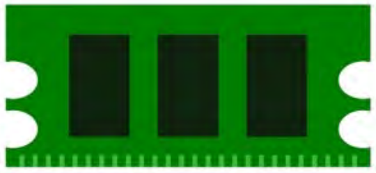
Large, slow external memory

Traditional I/O Model (EM Model)

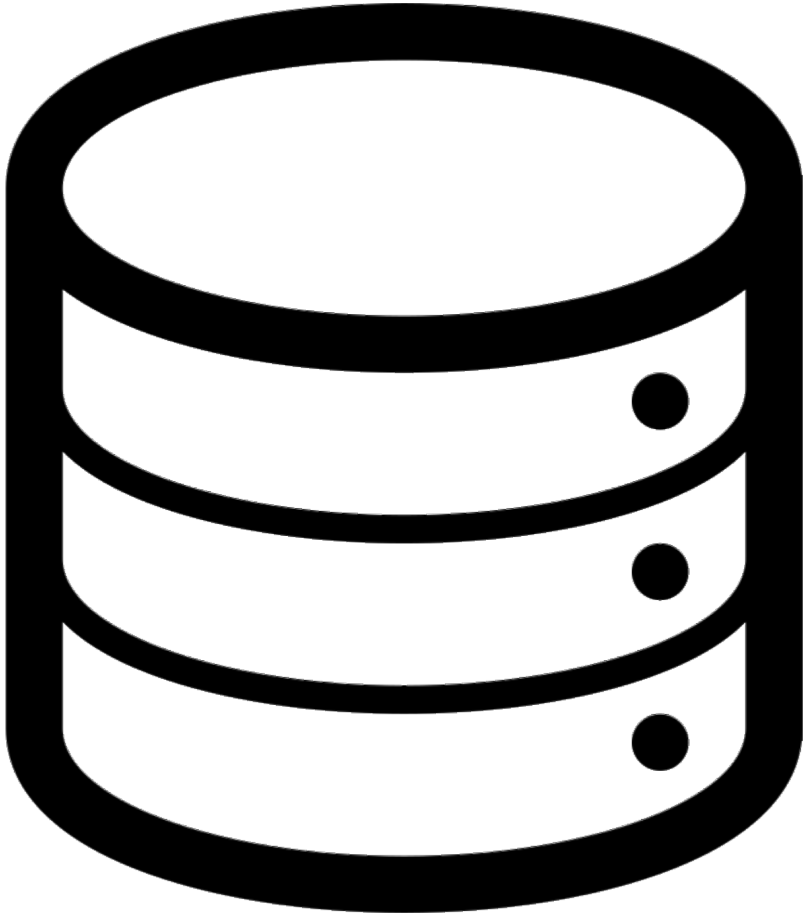
So, Total cost $\cong$ total number of read/write to disk

Transfer cost 1 unit

0 access cost



Small, fast main memory (size M)

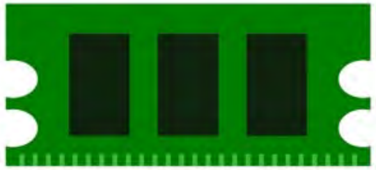


Large, slow external memory

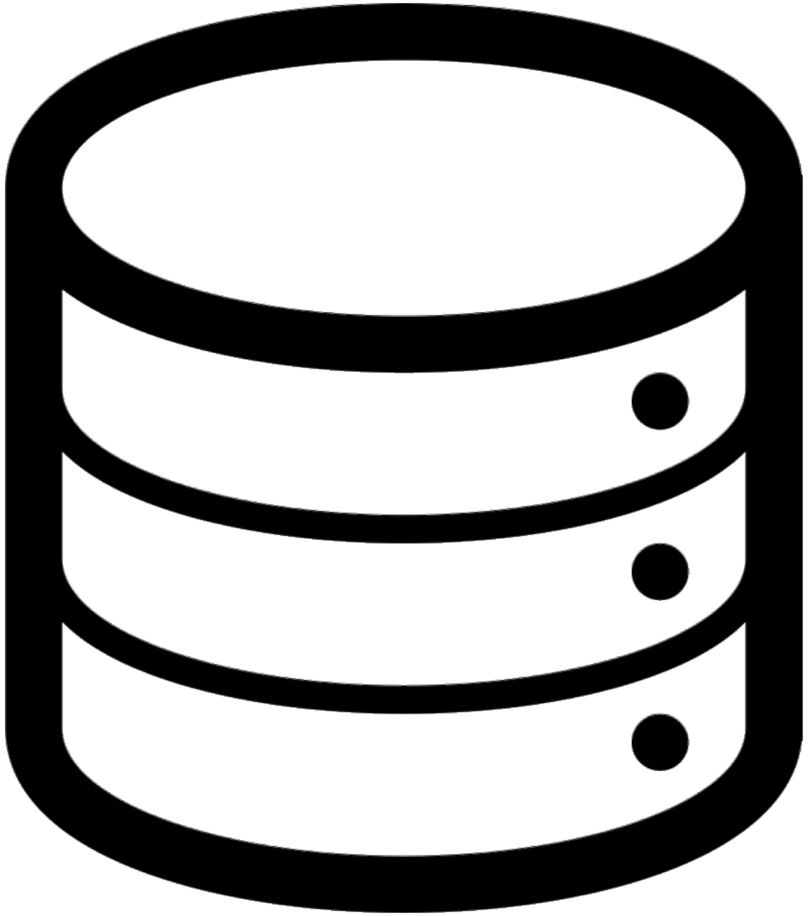
Traditional I/O Model (EM Model)

Two assumptions

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



Small, fast main memory (size M)

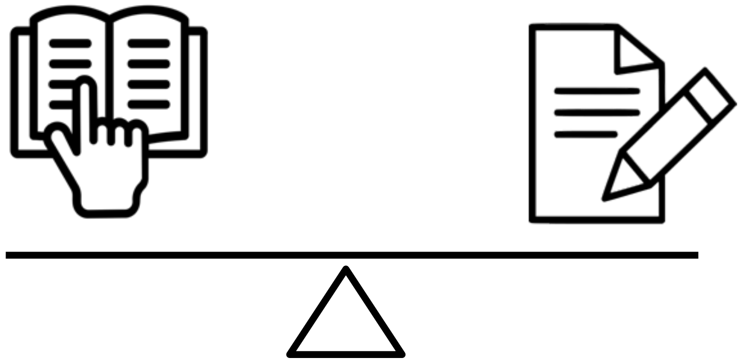


Large, slow external memory

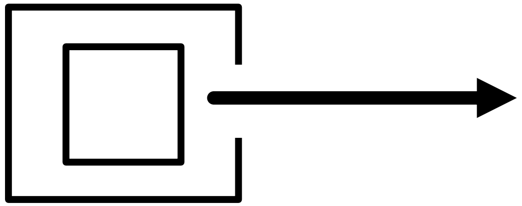
Hard Disk Drives

Hard Disk Drives

Two assumptions of EM 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?

Overview

Why an I/O model?

Why not the traditional I/O model?

Asymmetry & Concurrency of Modern Storage Devices

The parametric I/O model

A [much better] bufferpool policy

Experimental Evaluation

Future work & Conclusions

Solid-State Drives
&
Non-Volatile Memories

No Mechanical Movement!

Solid State Drives & NVMs

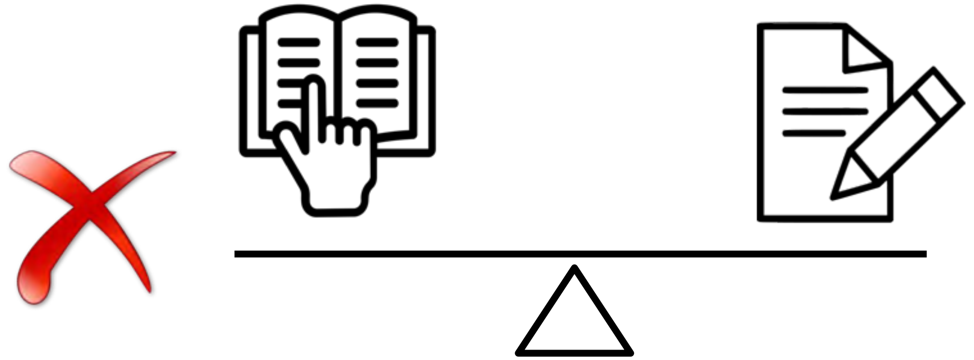
SSDs

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

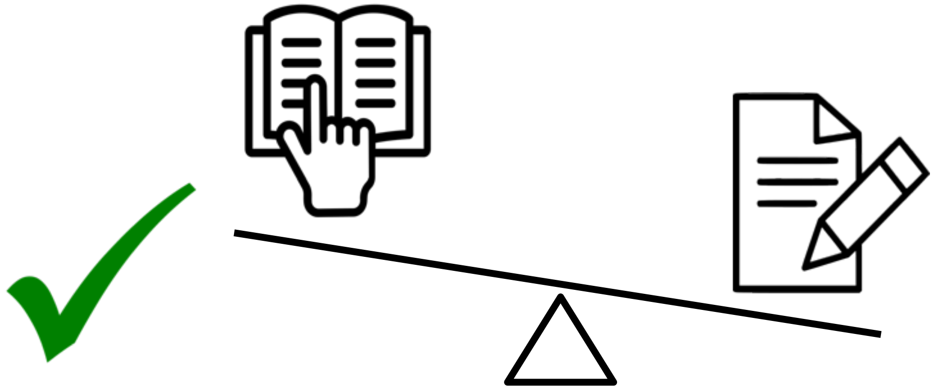
NVMs

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

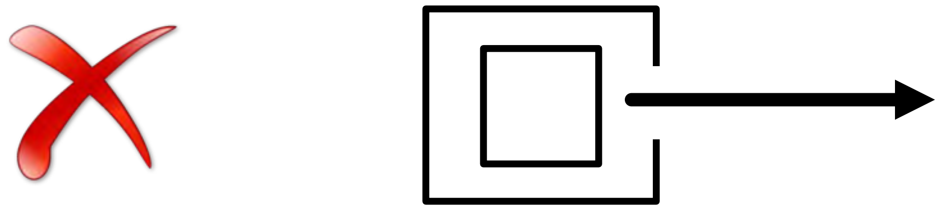
Modern Storage Devices



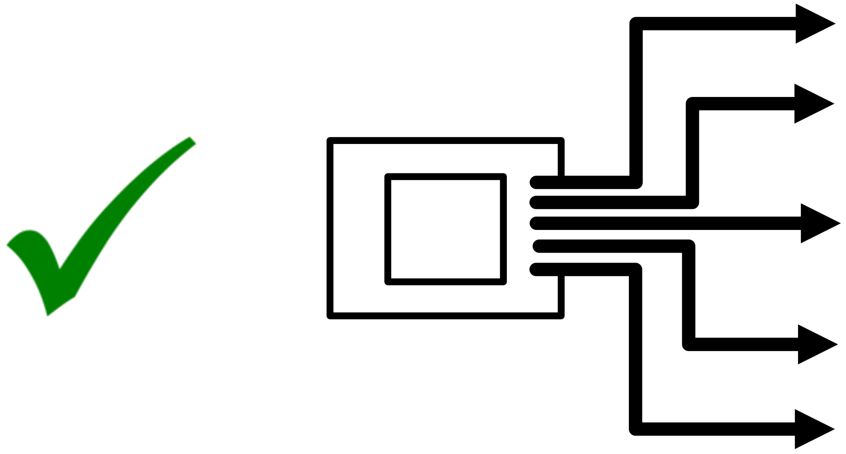
Symmetric cost for Read & Write



Read/Write Asymmetry

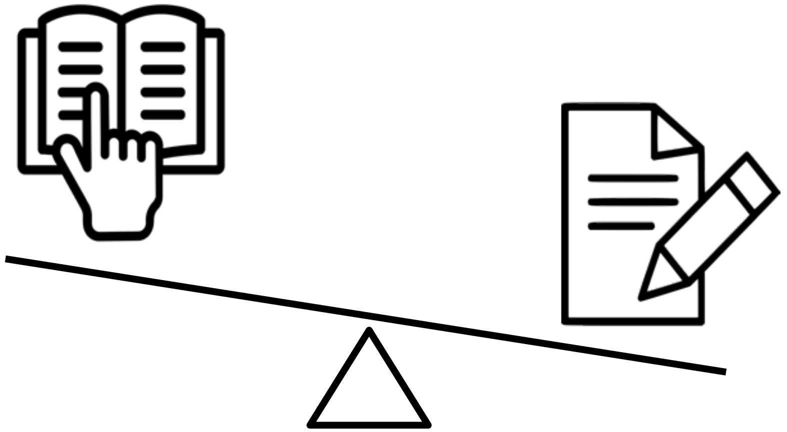


One I/O at a time



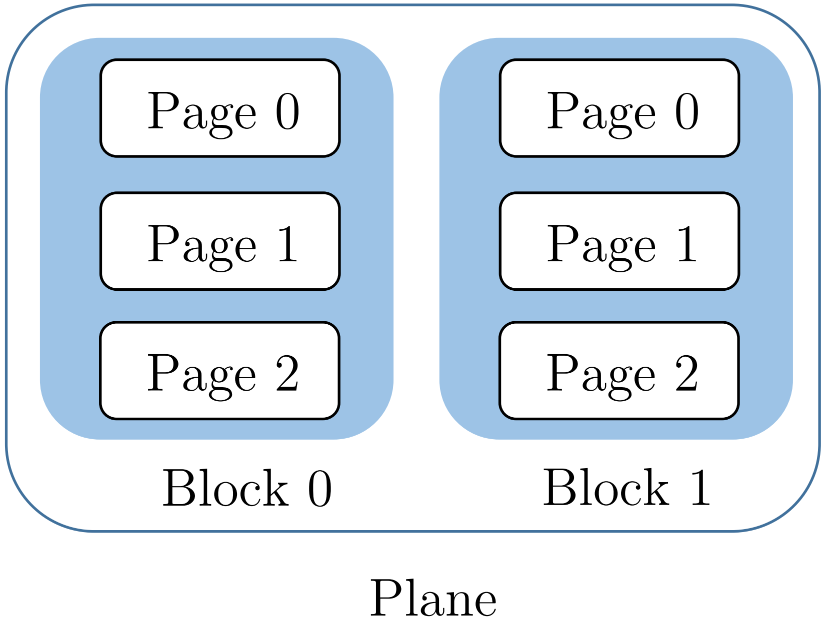
Concurrency

Read/Write Asymmetry



Solid-State Storage Devices

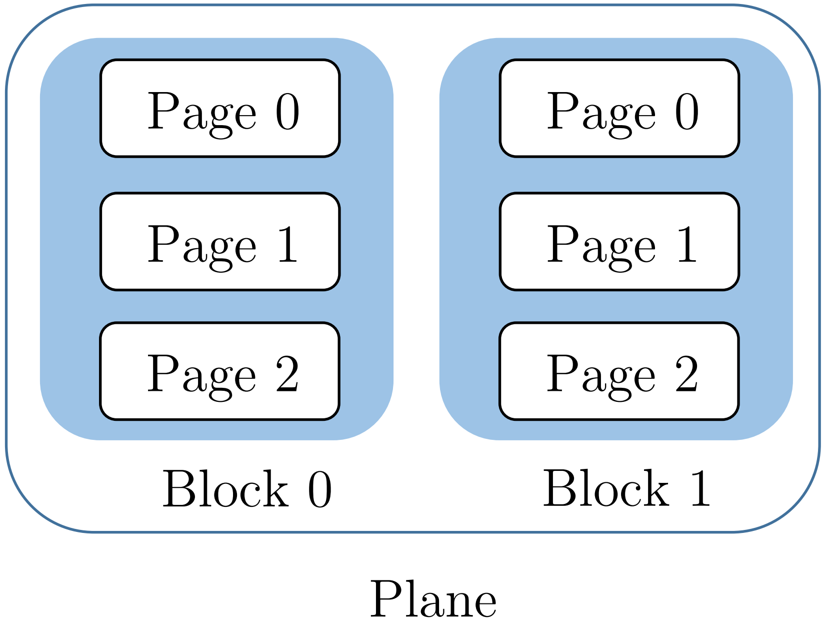
- Both reads & writes are fast in empty drive
- **Out-of-place** updates cause invalidation
- Invalidation causes garbage collection due to **erase-before-write**
- Limited device lifetime



Solid-State Storage Devices

Reasons behind read/write asymmetry

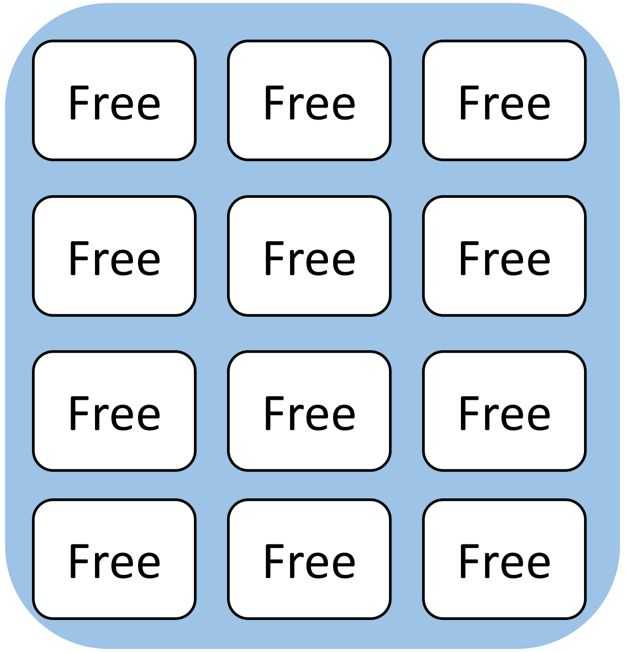
- erase-before-write
- large erasure granularity
- garbage collection



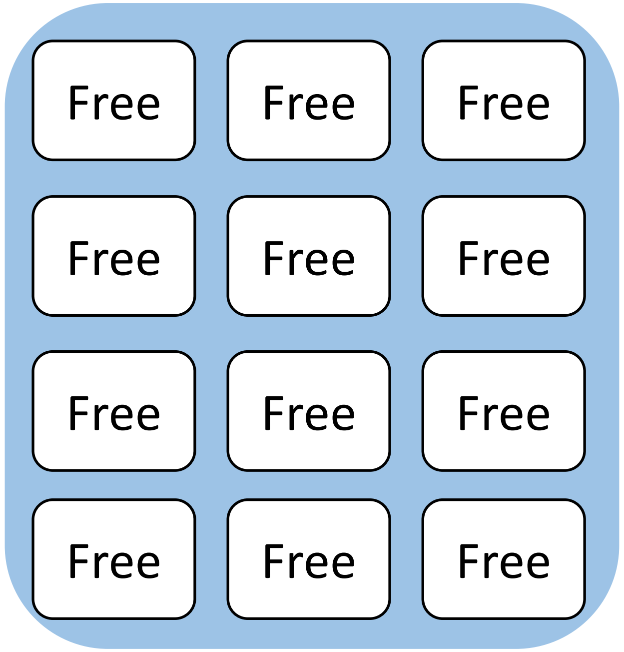
Solid-State Storage Devices

Writes

come

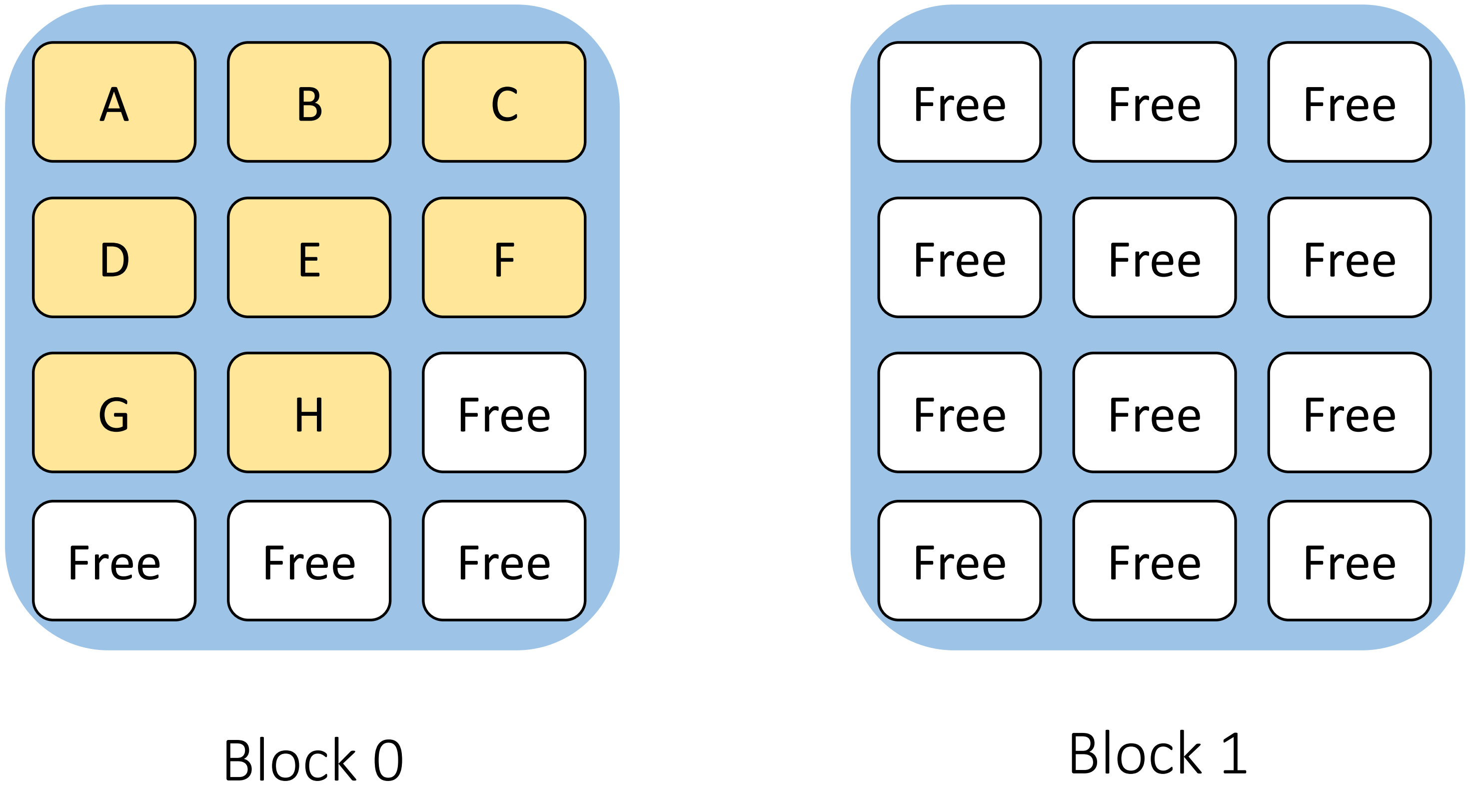


Block 0



Block 1

Solid-State Storage Devices

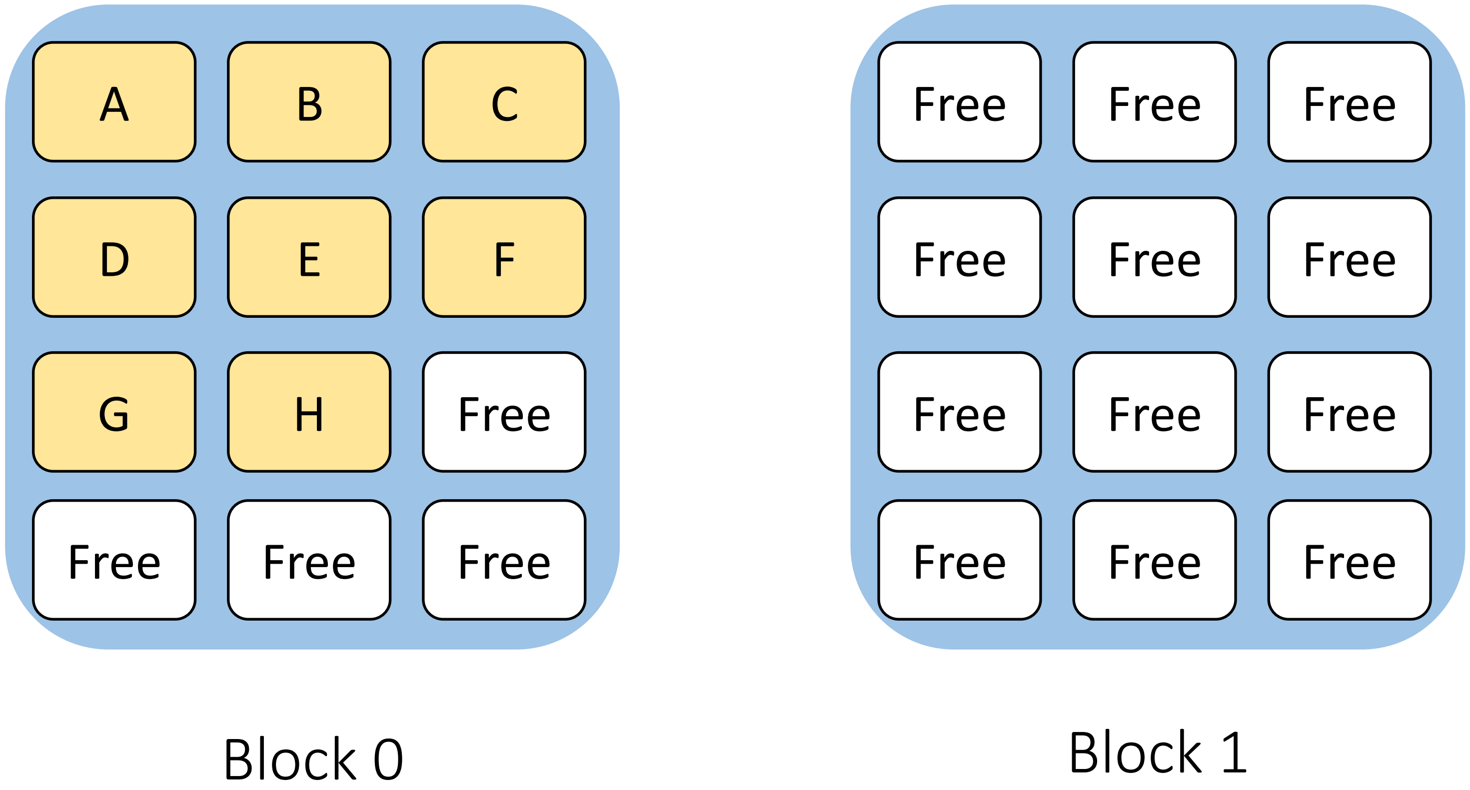


Writing in a free page isn’t costly!

Solid-State Storage Devices

Update

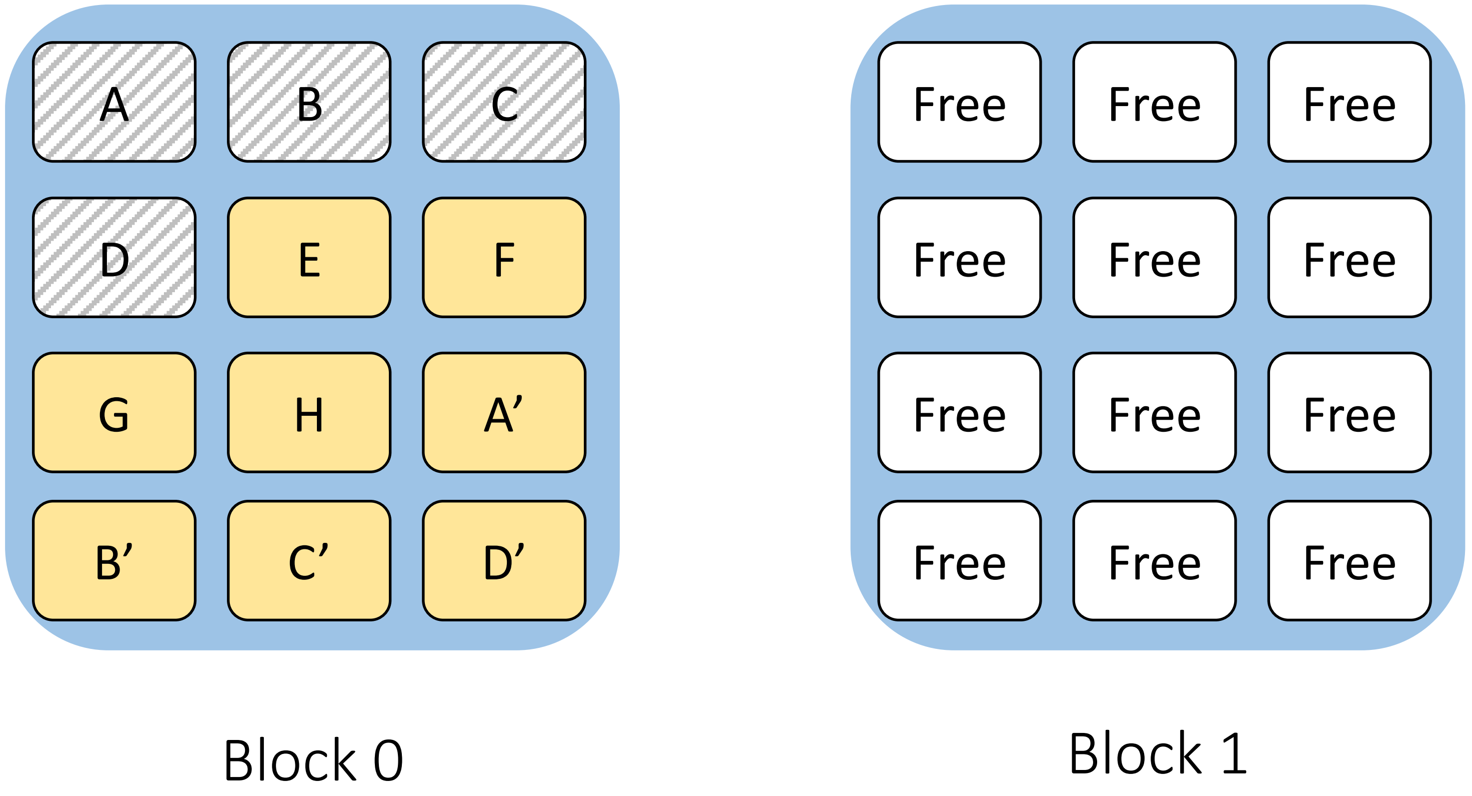
A, B, C, D



Solid-State Storage Devices

Update

A, B, C, D

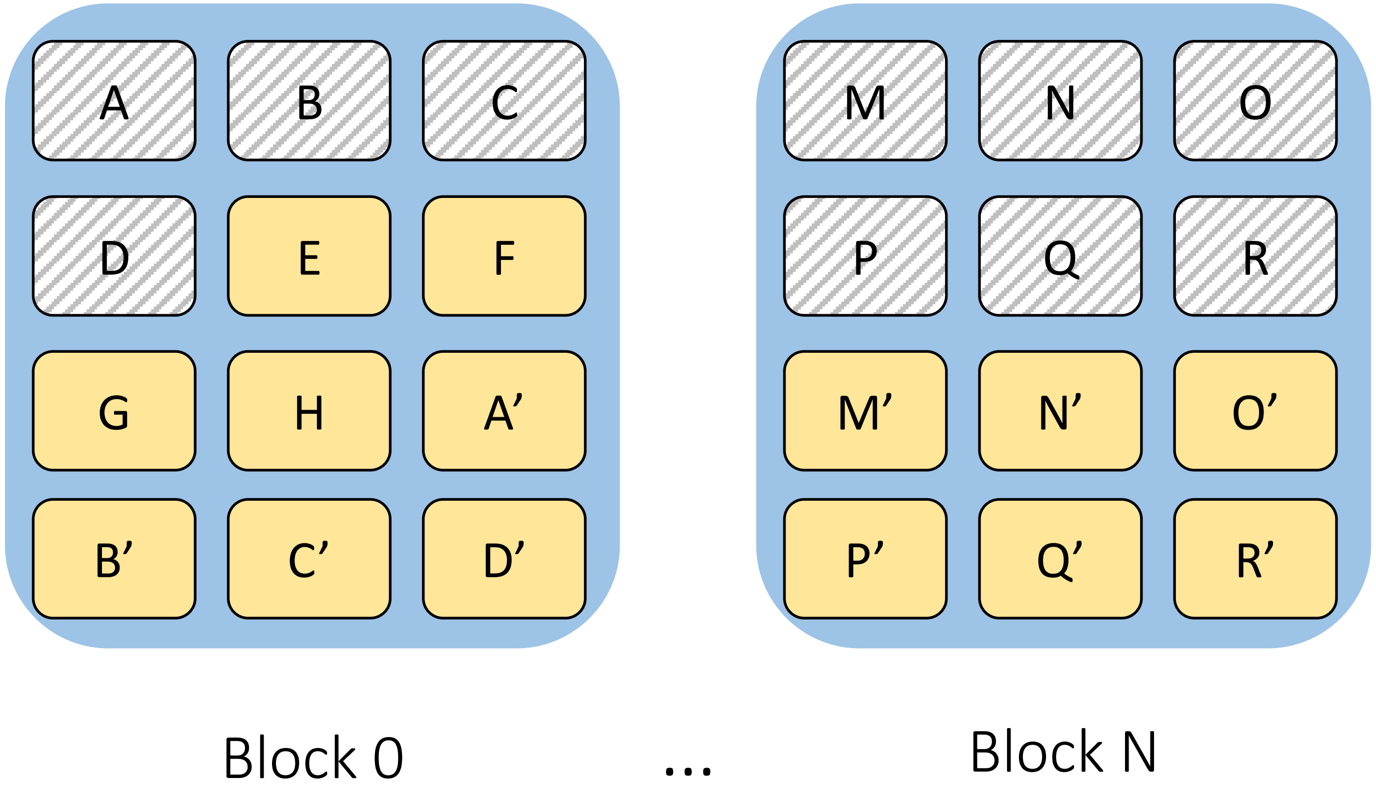


Not all updates are costly!

Solid-State Storage Devices

What if there is no space?

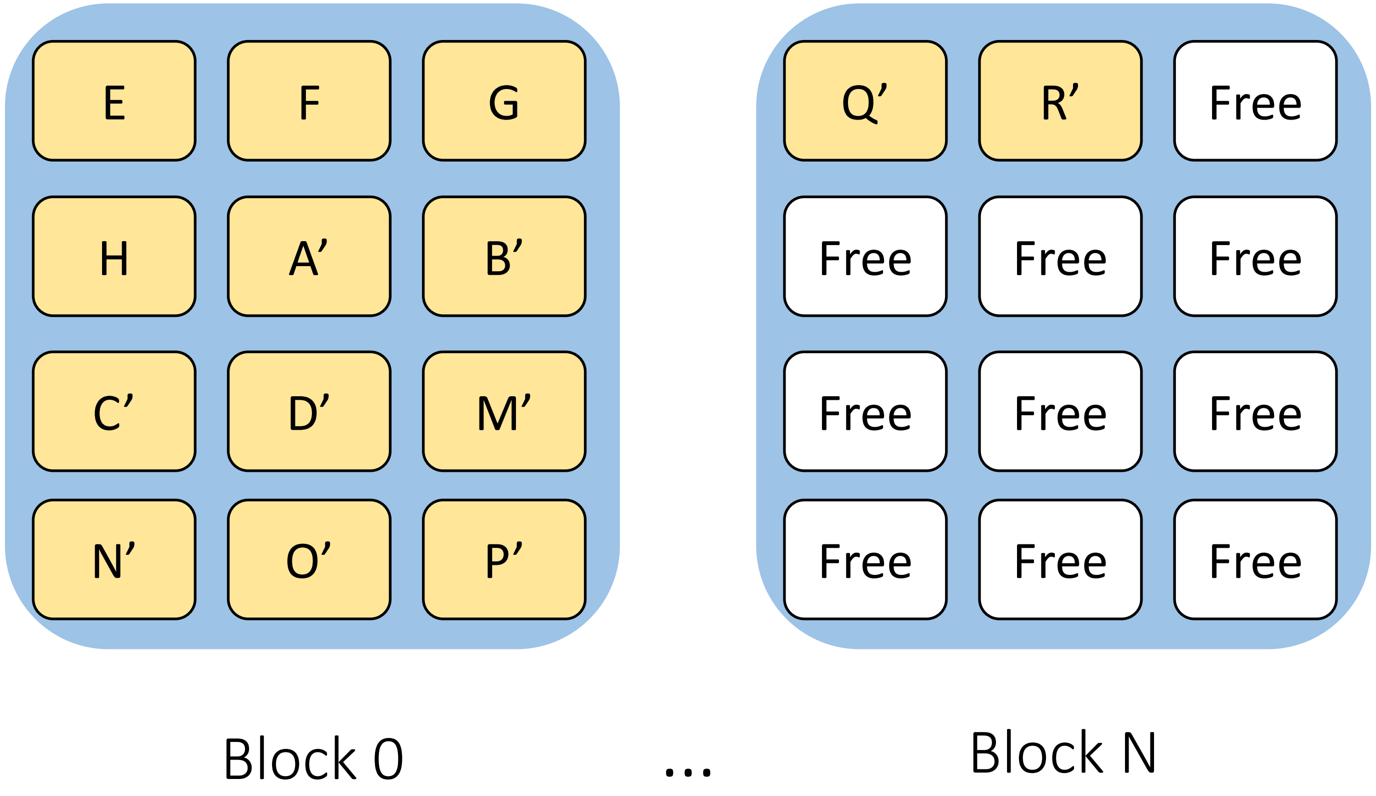
Garbage Collection!



Solid-State Storage Devices

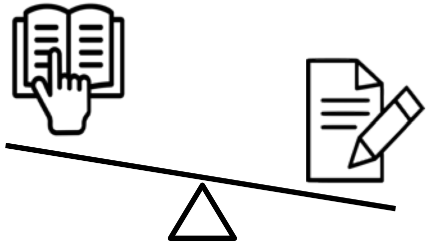
What if there is no space?

Garbage Collection!



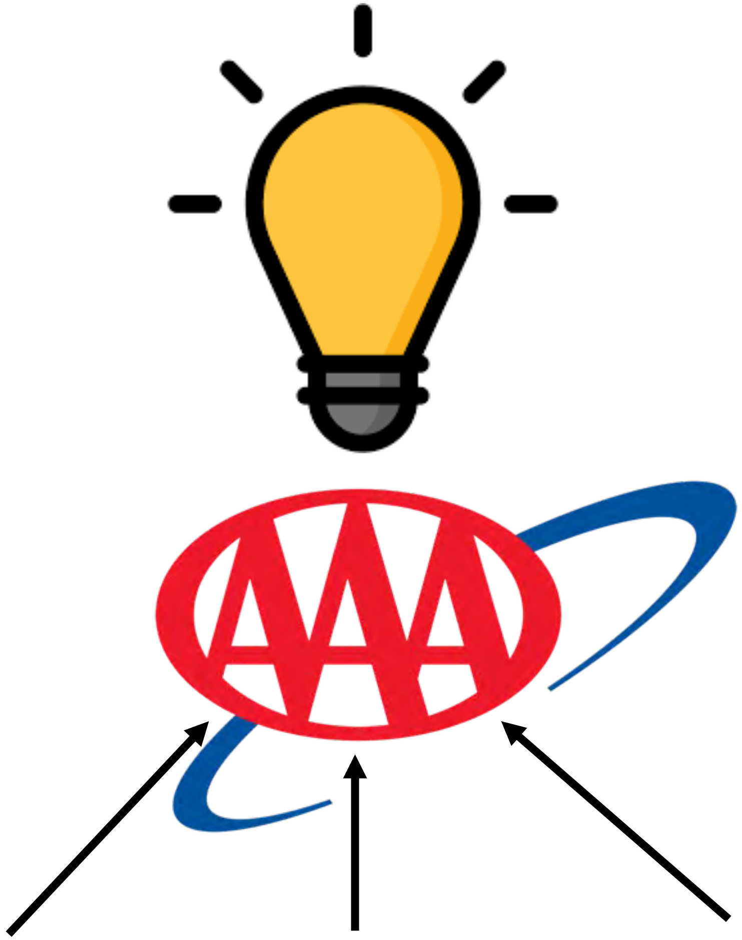
On average updates have higher cost (due to GC) → ***Read/Write asymmetry***

Read/Write Asymmetry - Example



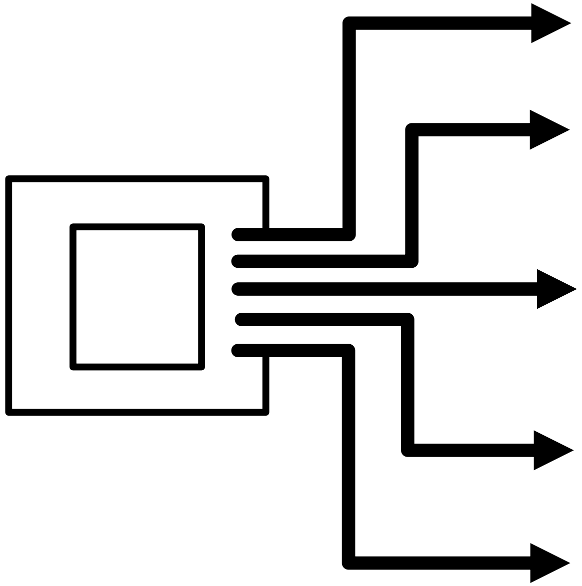
Intel Device	Advertised Random Read IOPS	Advertised Random Write IOPS	Advertised Asymmetry
D5-P4320	427k	36k	11.9
DC-P4500	626k	51k	12.3
DC-P4610	643k	199k	3.2
Optane 900P	550k	500k	1.1
Optane H10	330k	250k	1.3

Read/Write Asymmetry

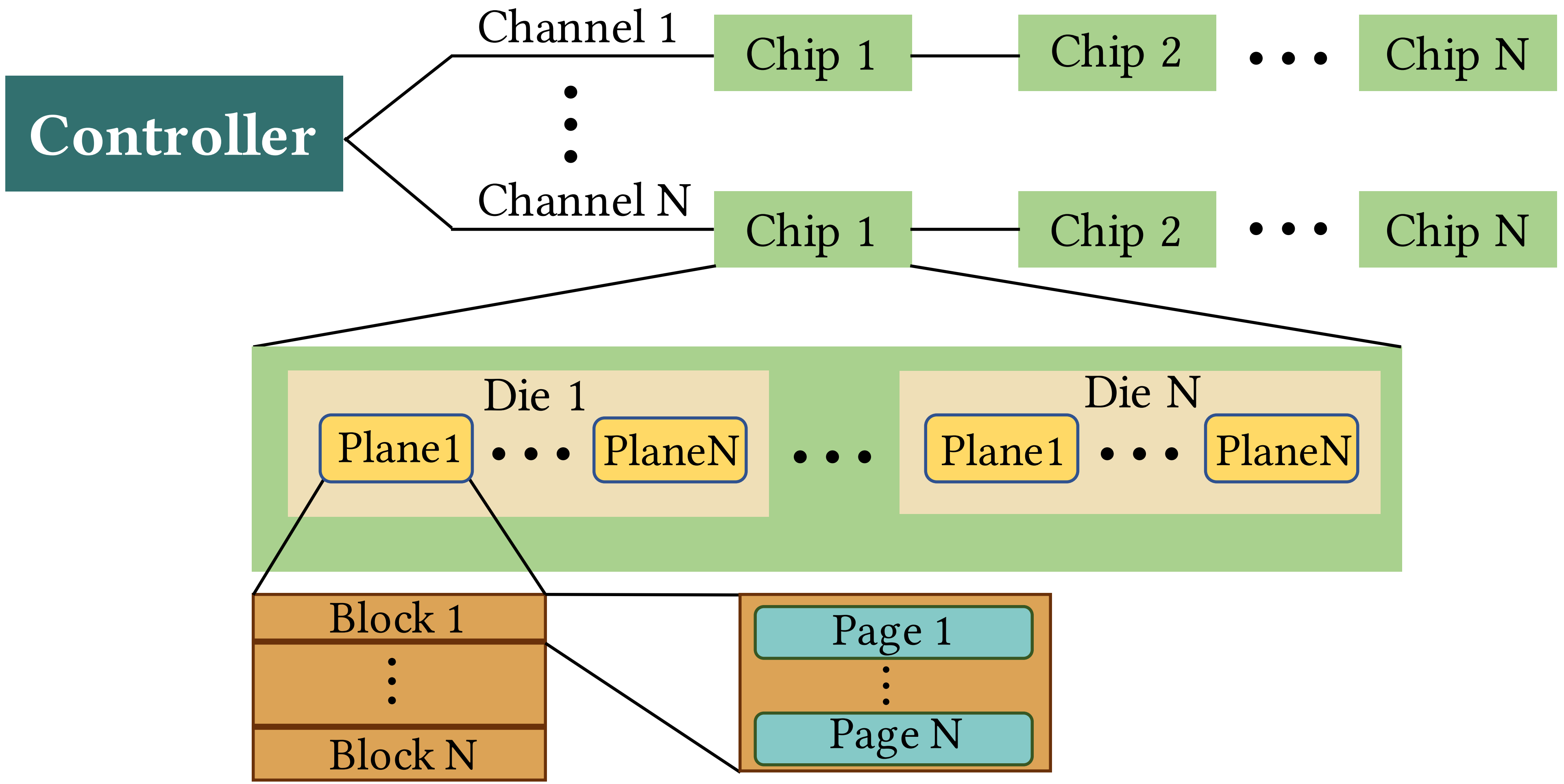


Asymmetry-Aware Algorithms

Concurrency

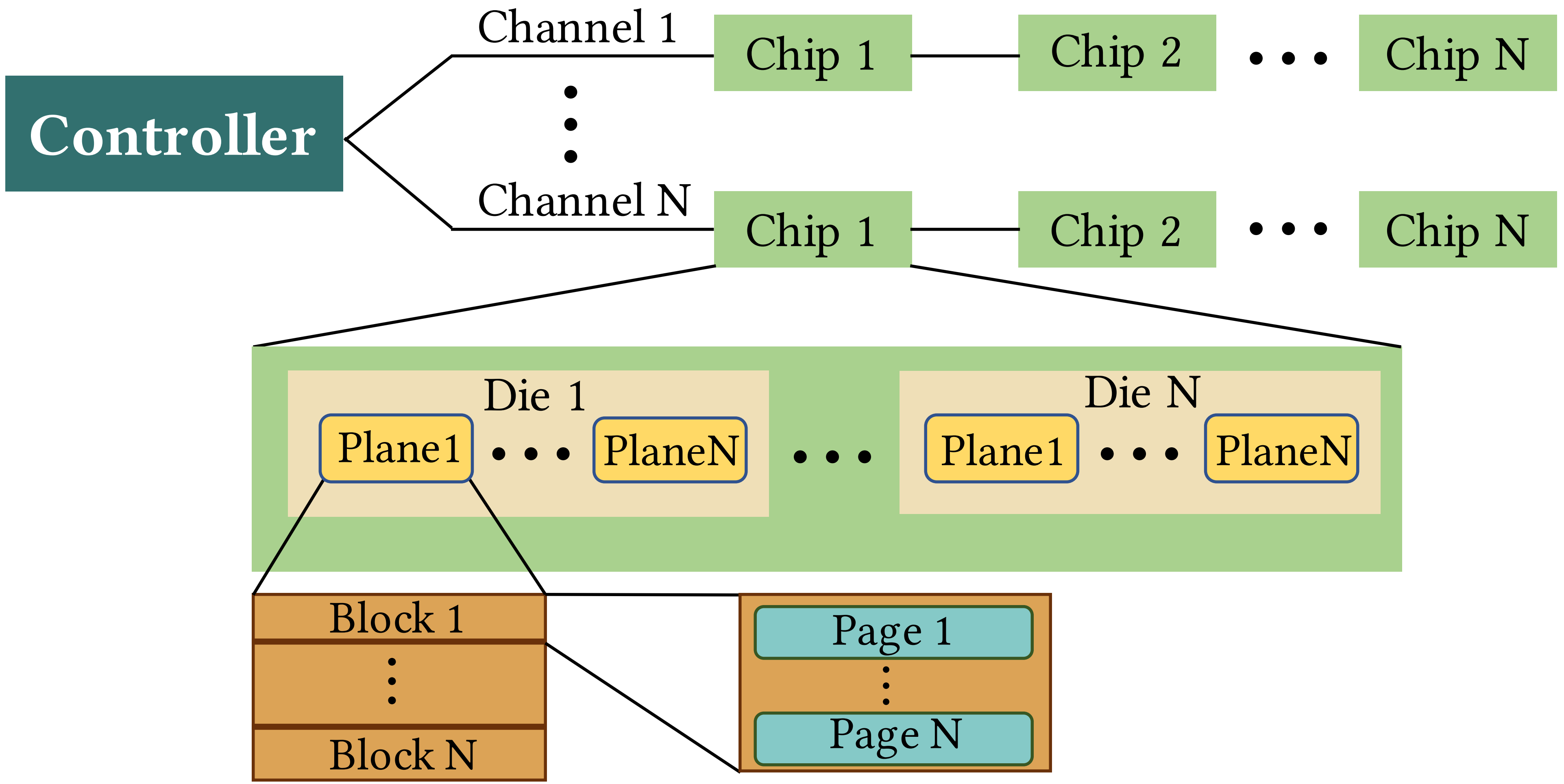


Internals of an SSD



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

How can Best Performance be Achieved?



Controller
Channel 1
Channel N
Chip 1
Chip 2
Chip N
Chip 1
Chip 2
Chip N
Die 1
Die N
Plane1
PlaneN
Plane1
PlaneN
Block 1
Block N
Page 1
Page N

Benchmarking

Benchmarking

Tools

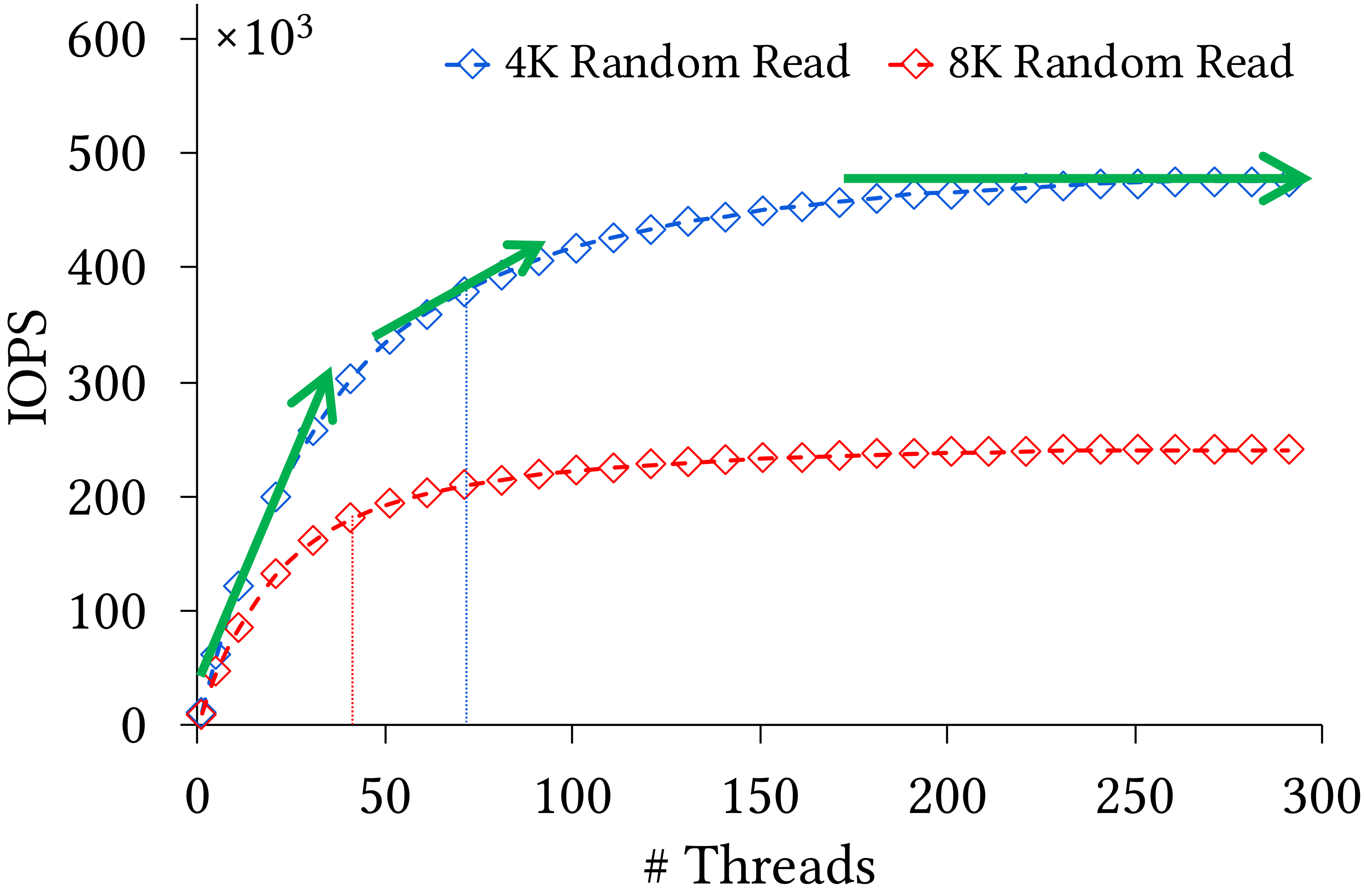
- Custom micro-benchmarking infrastructure
- fio
- Intel’s SPDK

Setup

- With File System
- Without File System

Measuring Asymmetry/Concurrency (With FS)

Device: Dell P4510 (1TB)



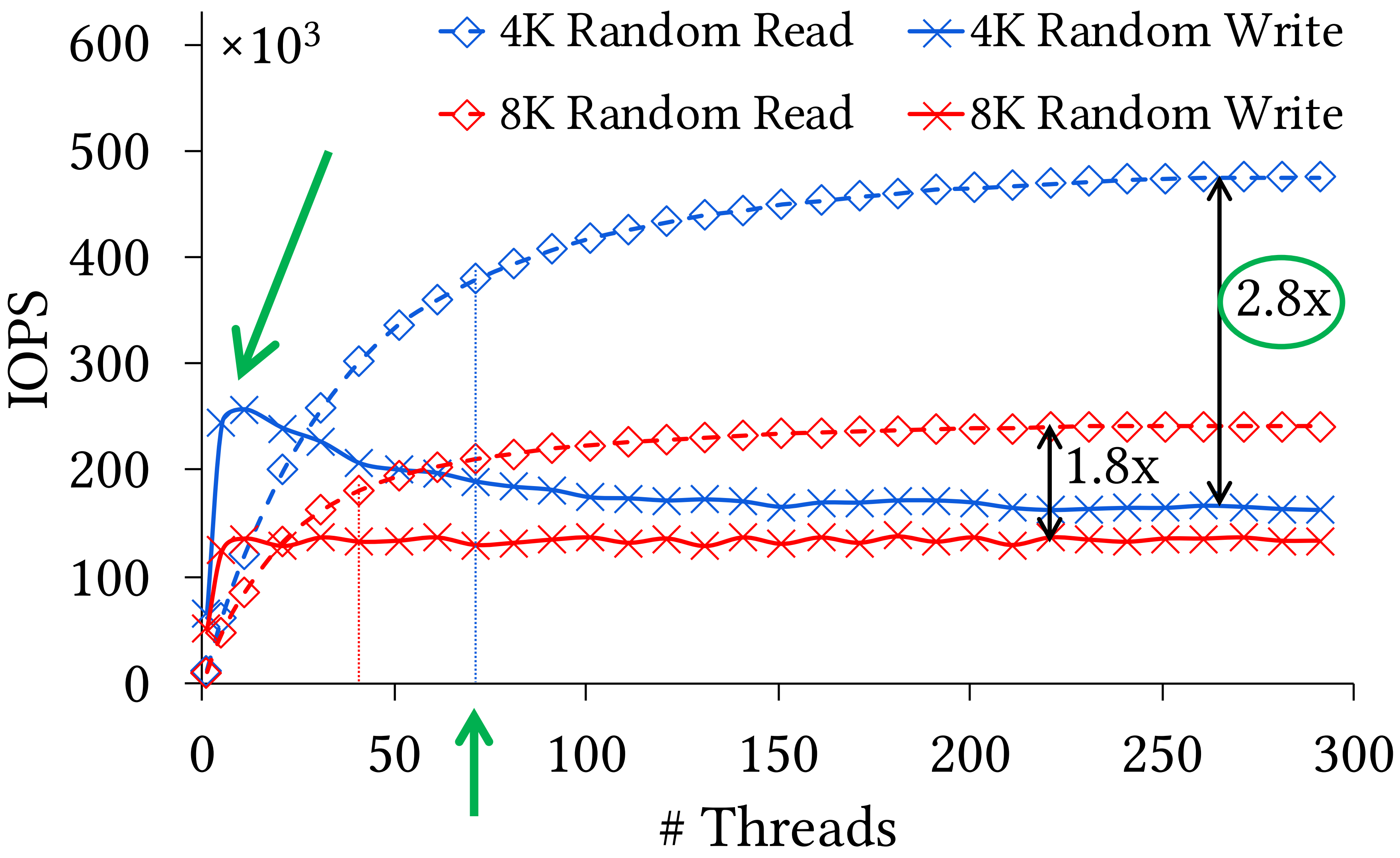
Measuring Asymmetry/Concurrency (With FS)

Device: Dell P4510 (1TB)

For 4K random read,

Asymmetry: 2.8

Concurrency: 70



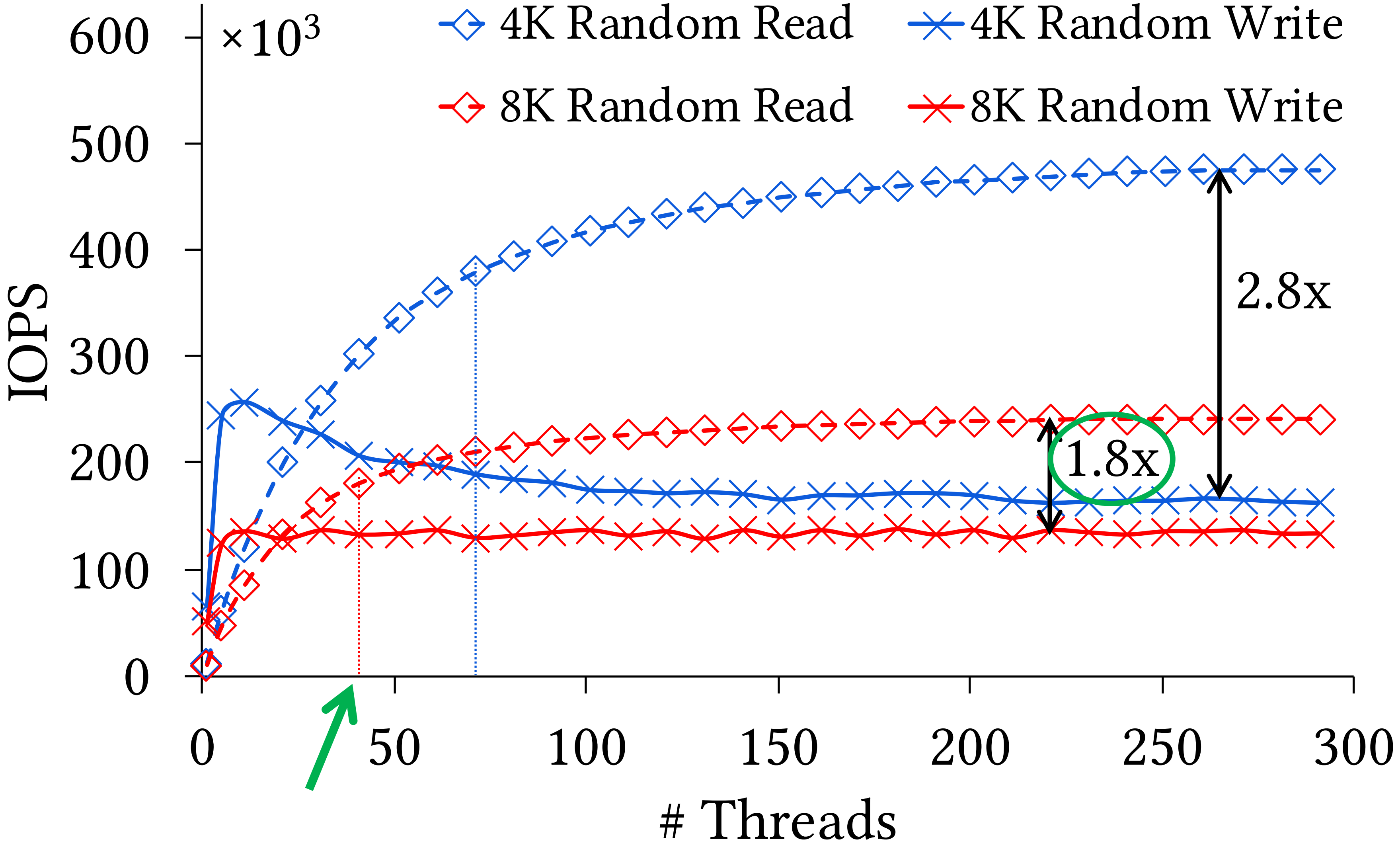
Measuring Asymmetry/Concurrency (With FS)

Device: Dell P4510 (1TB)

For 8K random write,

Asymmetry: 1.8

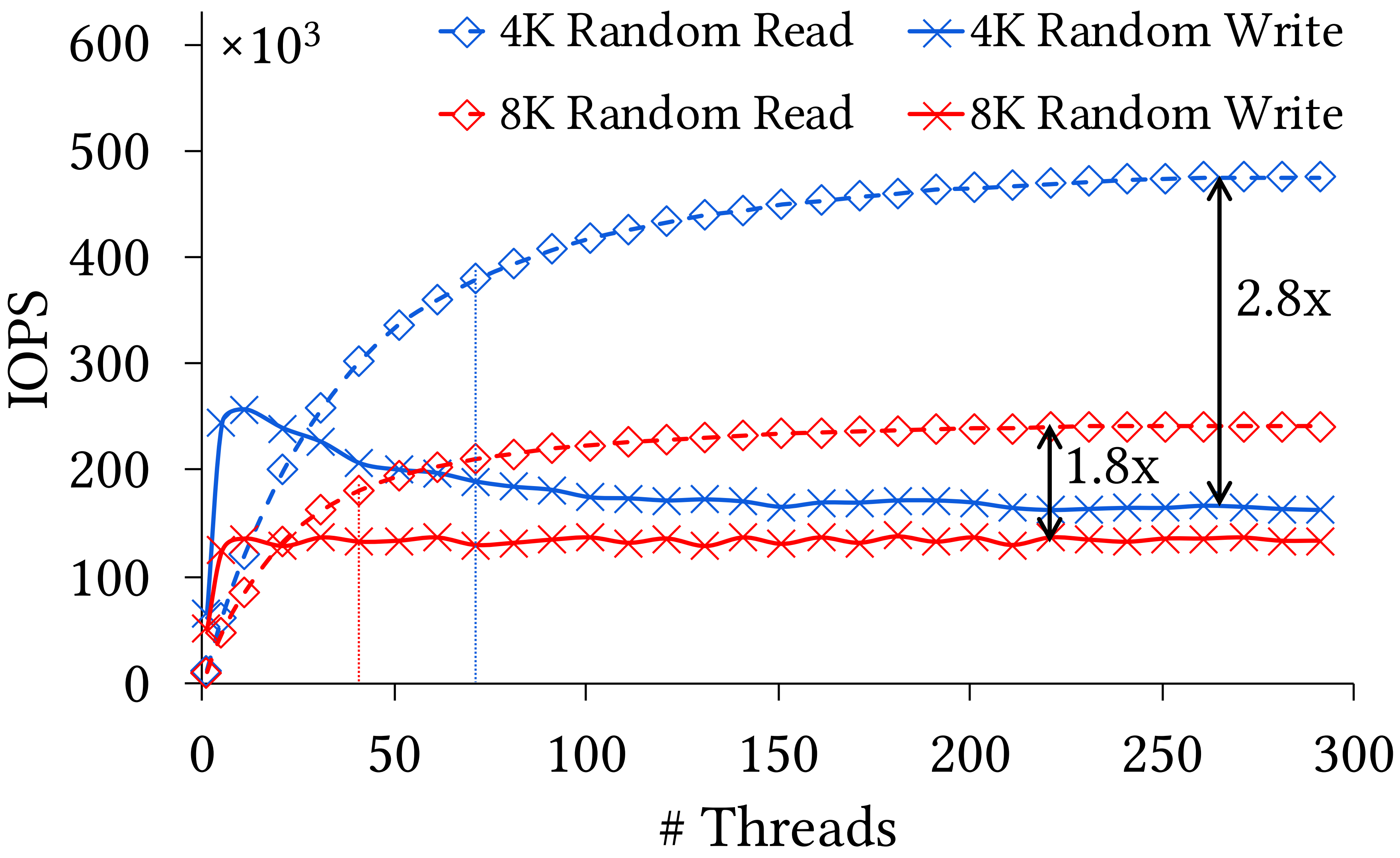
Concurrency: 10



Measuring Asymmetry/Concurrency (With FS)

Device: Dell P4510 (1TB)

Asymmetry and concurrency depends on request type and access granularity



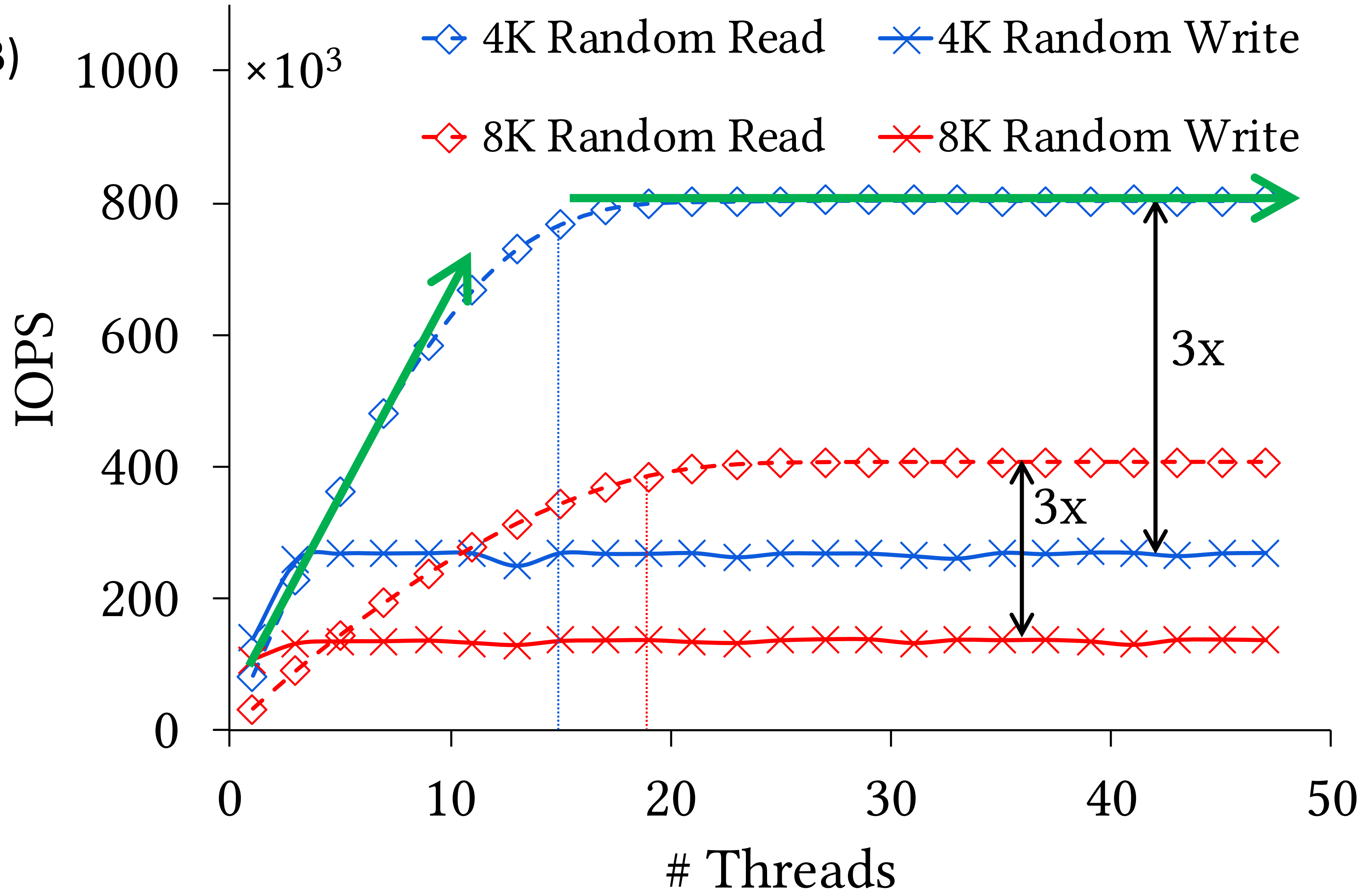
Measuring Asymmetry/Concurrency (Without FS)

Device: Dell P4510 (1TB)

For 4K random reads,

Asymmetry: 3

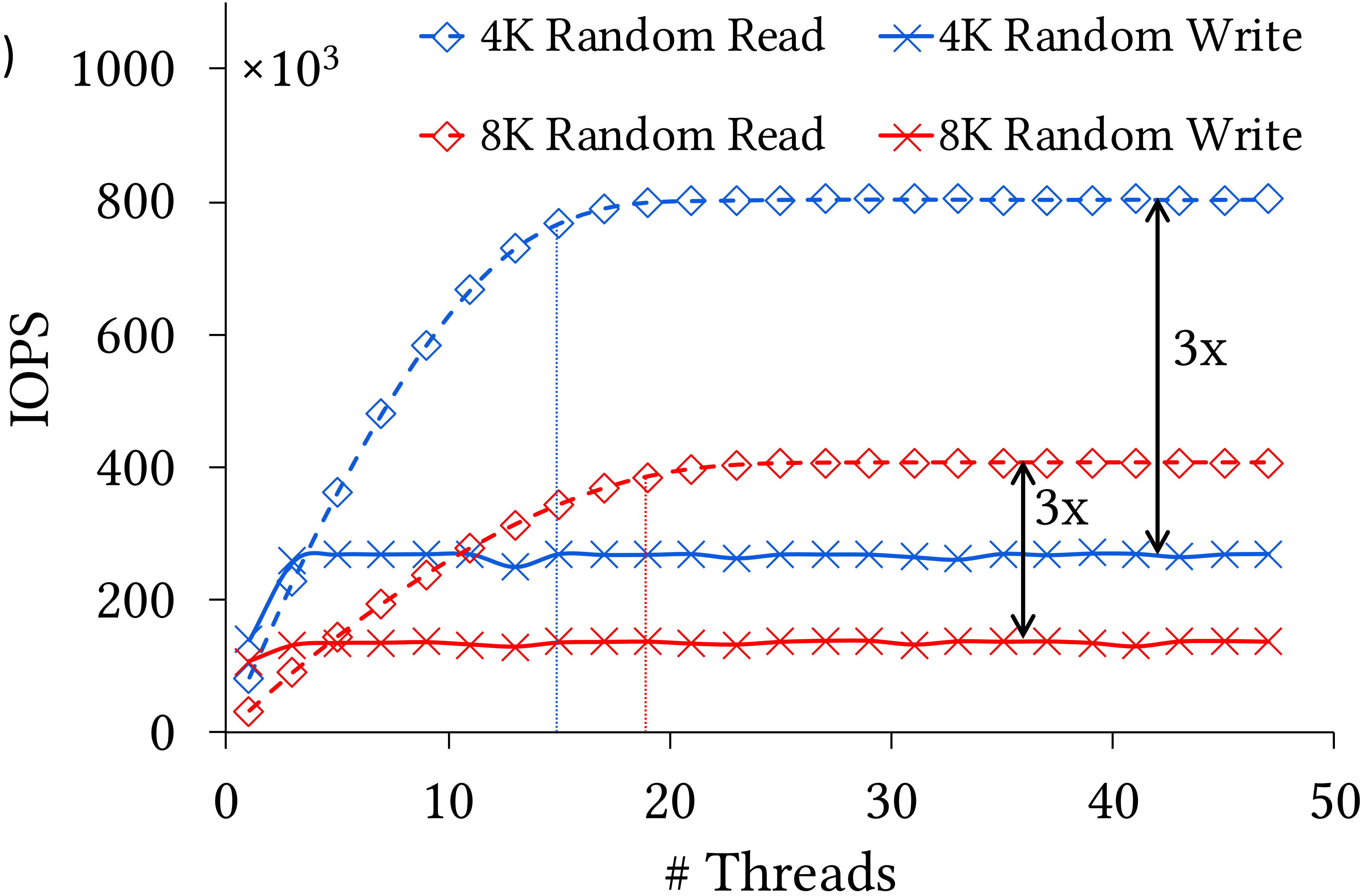
Concurrency: 14



Measuring Asymmetry/Concurrency (Without FS)

Device: Dell P4510 (1TB)

More stable performance without the file system!



Measuring Asymmetry/Concurrency

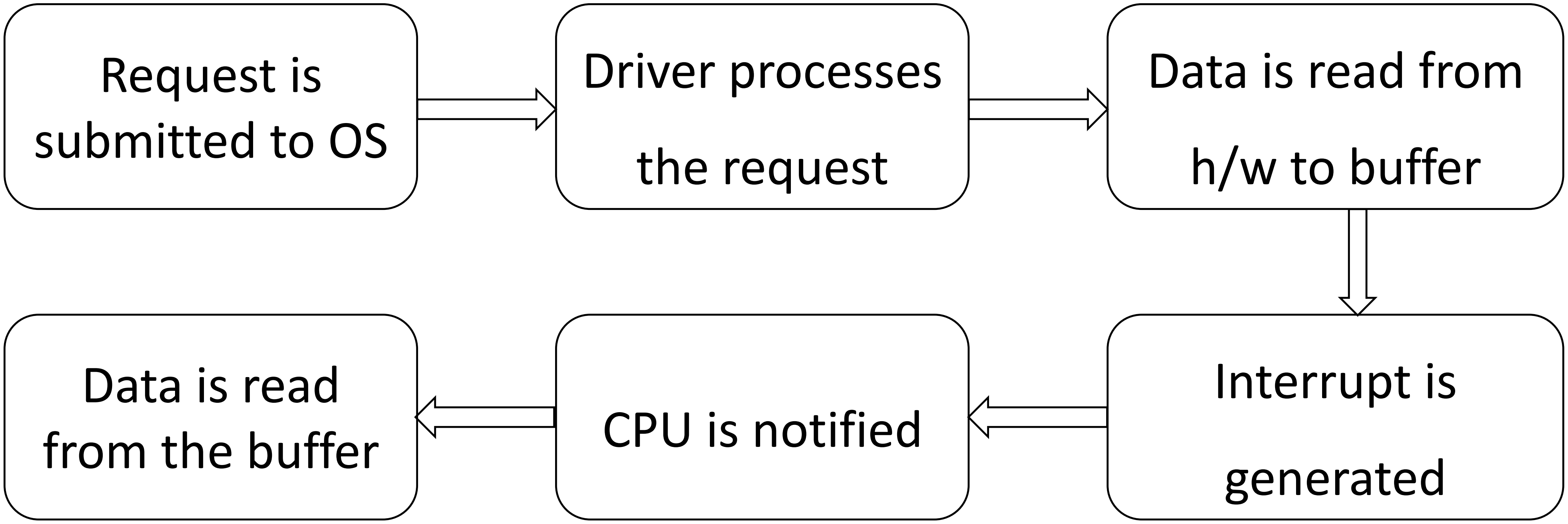
Table 1: Empirical *Asymmetry* and *Concurrency*

	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	70	8	1.8	40	7
PCIe SSD (w/o FS)	3.0	14	6	3.0	18	4
SATA SSD	1.5	13	9	1.3	17	5
Virtual SSD	2.0	11	21	1.9	6	10

Why does Performance Change Depending on the File System?

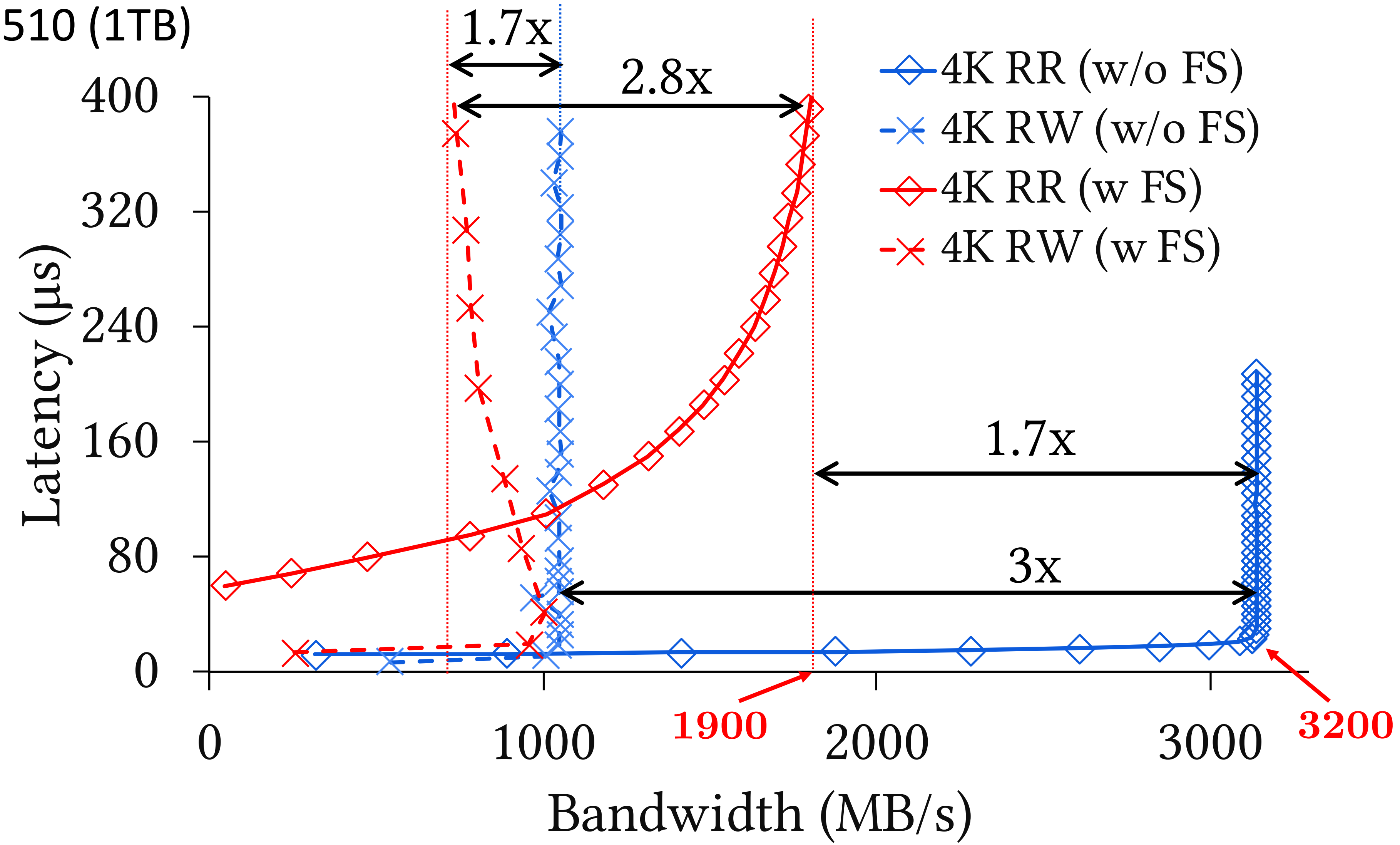
Can the File System be the Bottleneck?

Interrupt-based model

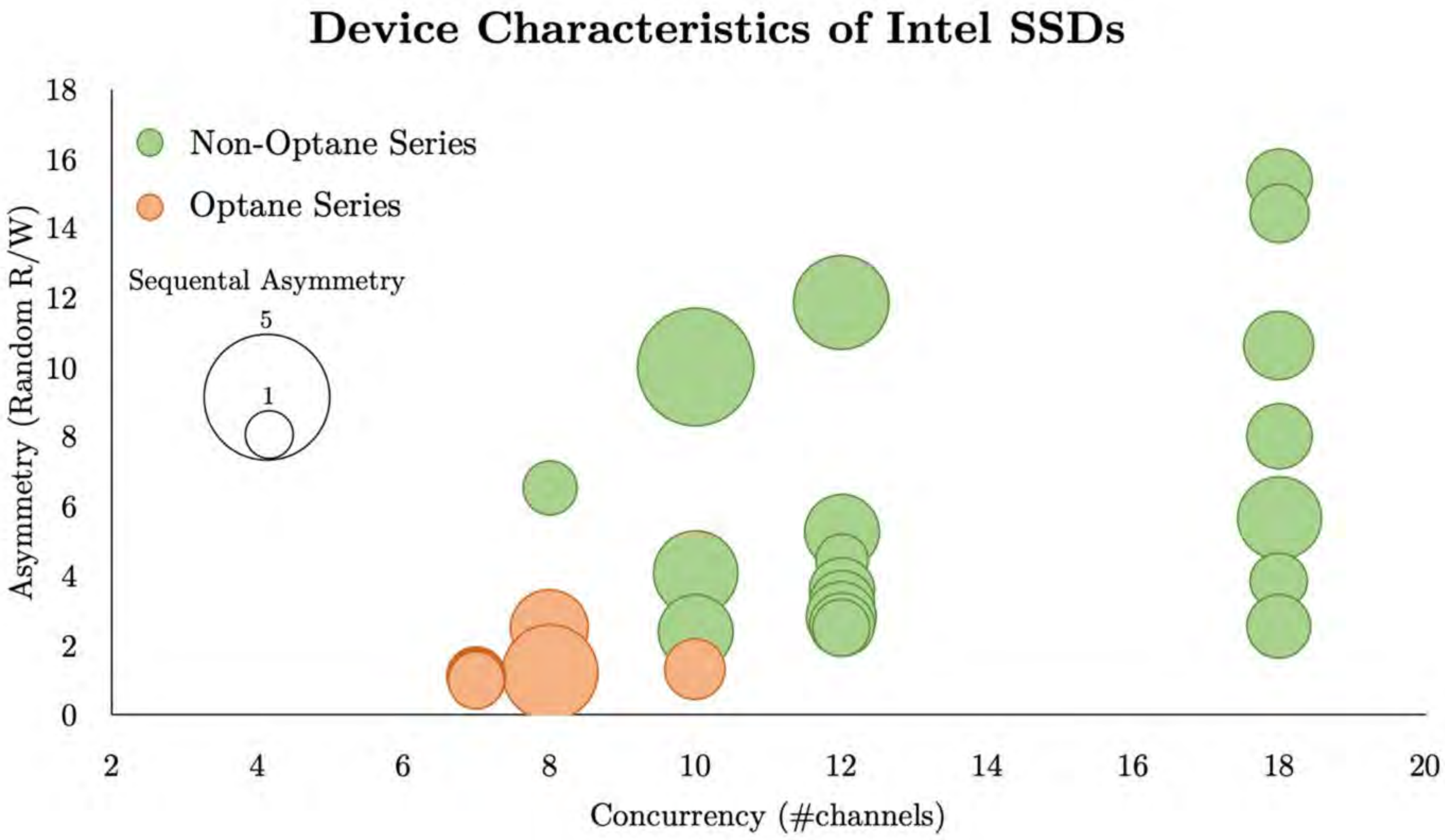


Can the File System be the Bottleneck?

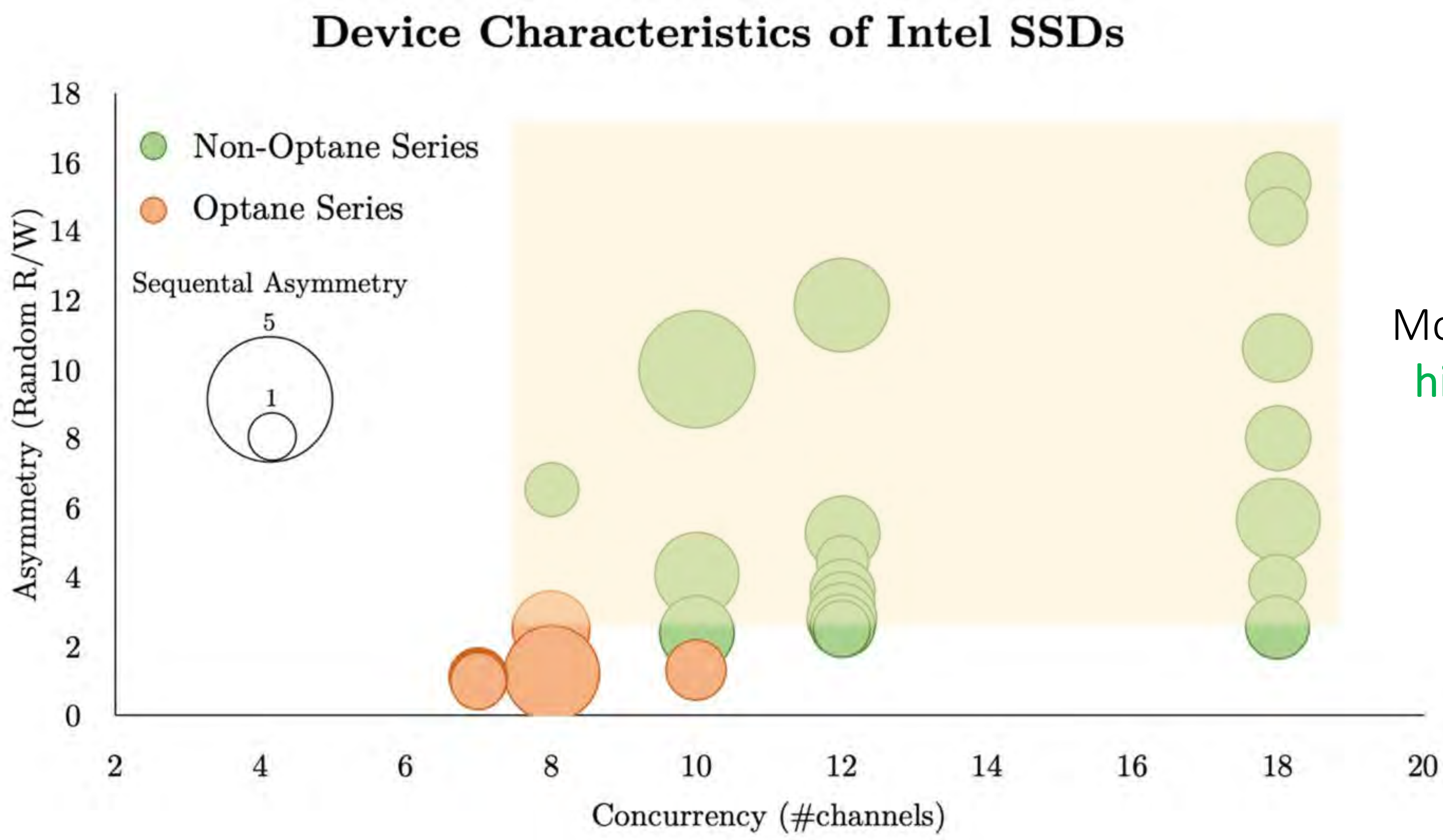
Device: Dell P4510 (1TB)



Modern Storage Devices



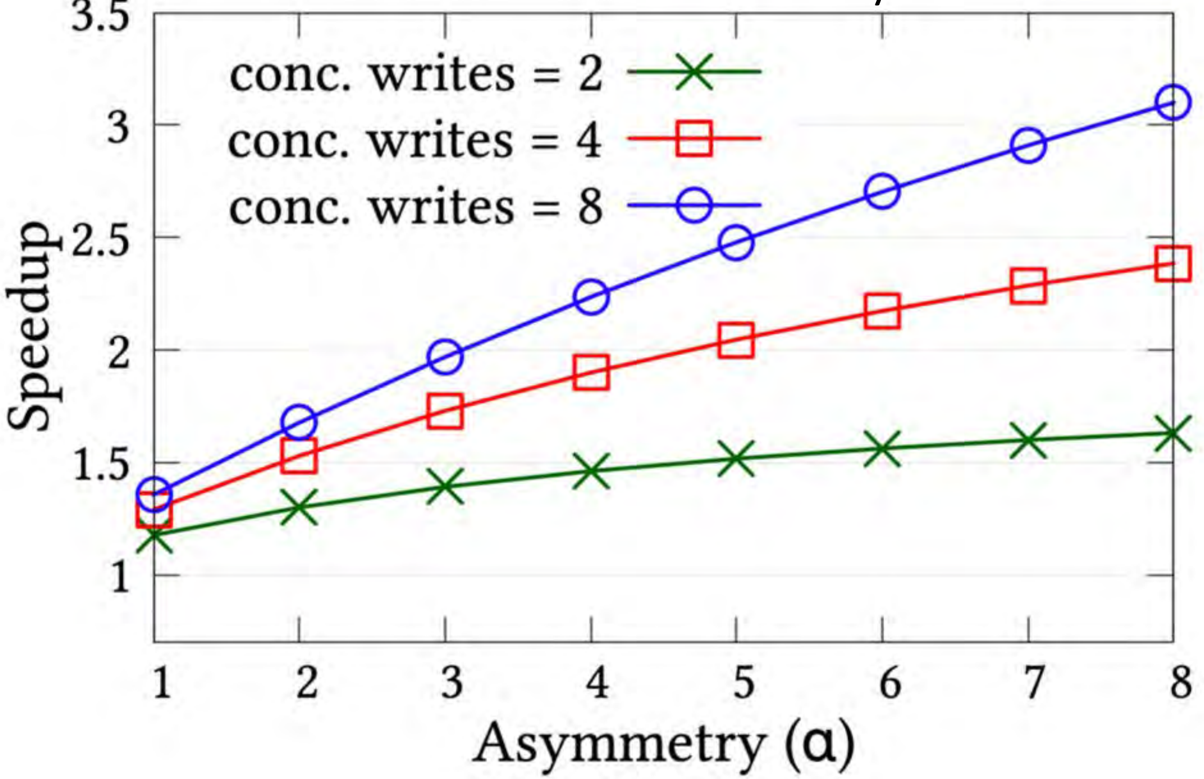
Modern Storage Devices



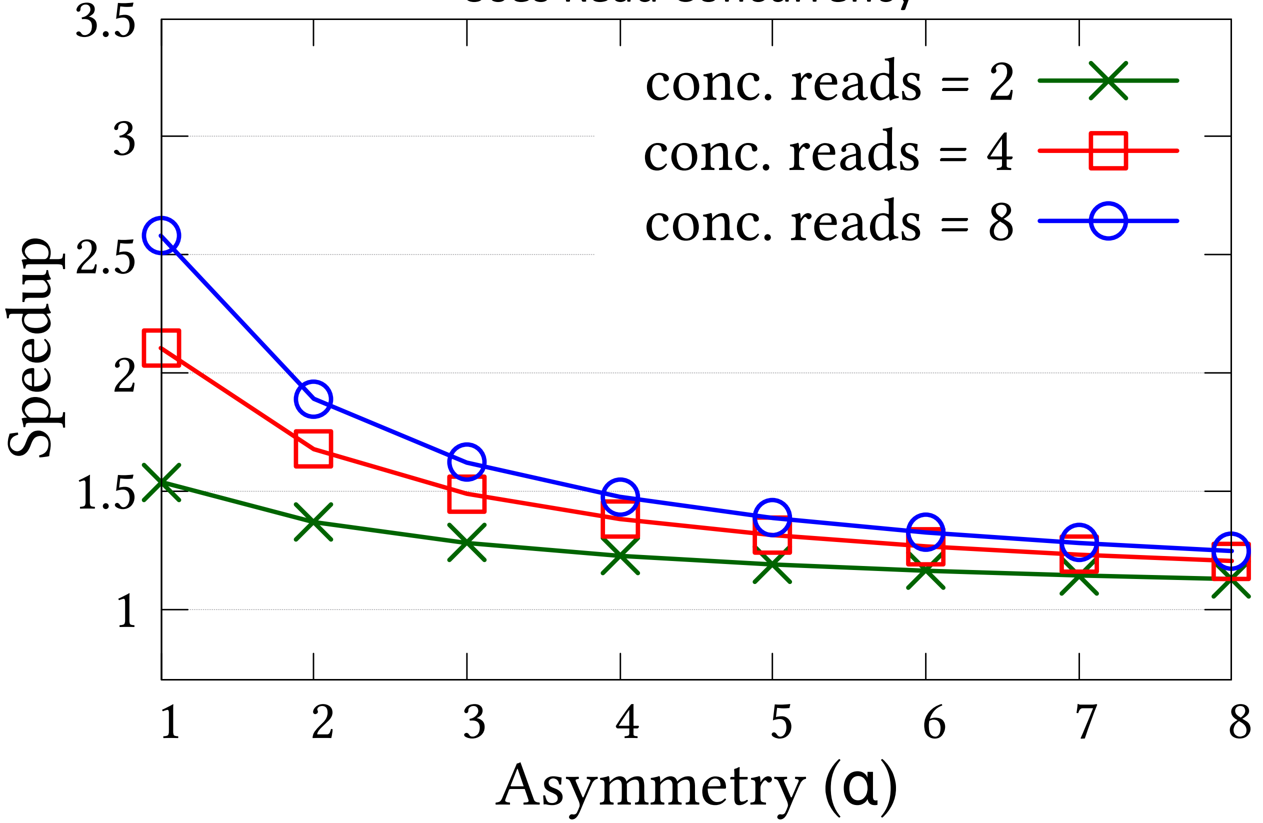
Most devices have high asymmetry

Impact of Asymmetry/Concurrency

Uses Write Concurrency



Uses Read Concurrency



Speedup increases as more concurrent I/Os are used, and α dictates the gain.

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

read/write asymmetry and concurrency?

Overview

Why an I/O model?

Why not the traditional I/O model?

Asymmetry & Concurrency of Modern Storage Devices

The parametric I/O model

A [much better] bufferpool policy

Experimental Evaluation

Future work & Conclusions

Parametric I/O Model

PIO (M, k, α)

Main Memory Size

Concurrency

Asymmetry

Buffer Pool Page Eviction Algorithm

Classical

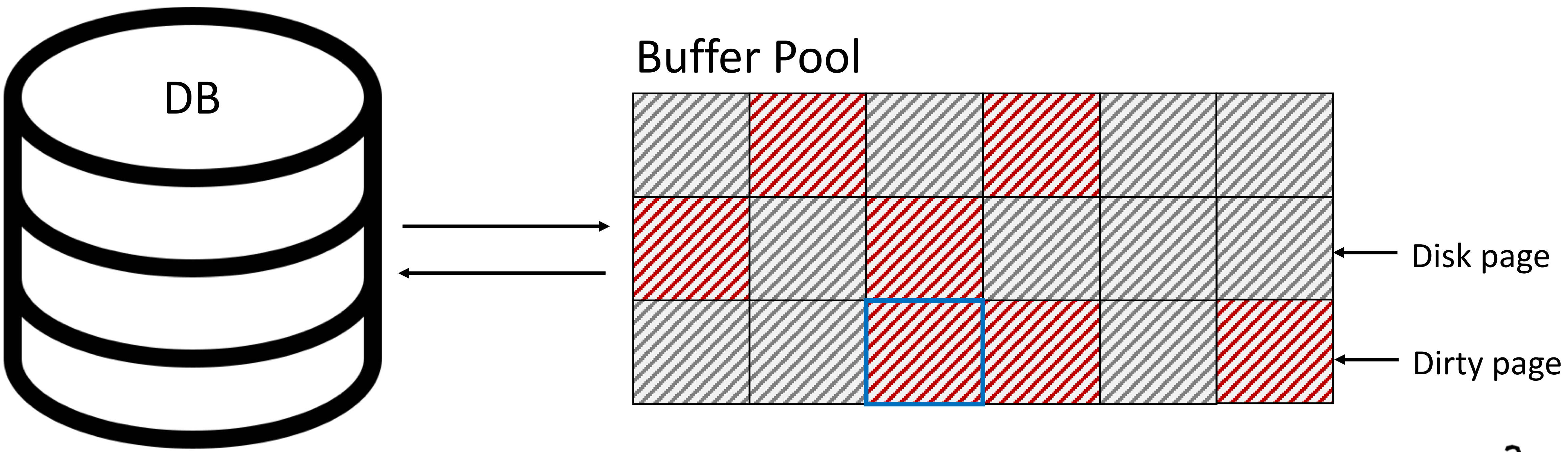
```
Request(page);
If (page in BP) -> return page
Else
    // Miss! Bring the page from Disk
    If BP not full -> Read requested page from Disk
    Else
        - Select a page for eviction based on replacement policy
        - If the candidate page is dirty, write to disk
        - Drop the candidate page from BP
        - Read requested page
```

[if the request is a **write**, an **in-memory update** takes place that set the **dirty bit** as well]

Popular Page Replacement Algorithms

LRU	(Most Popular)
LFU, FIFO	(Simple)
Clock	(Commercial)
CFLRU	Flash-Friendly
LRU-WSR	Flash-Friendly

Traditional Buffer Pool Manager



All these policies exchange one read for one write!

Is this Fair?

NO!

- Since device has read/write asymmetry, it is **NOT** fair to perform one write for one read
- Since writes are now α times costlier, for x writes and y reads, the total weighted I/O cost = $\mathbf{y + \alpha x}$

Overview

Why an I/O model?

Why not the traditional I/O model?

Asymmetry & Concurrency of Modern Storage Devices

The parametric I/O model

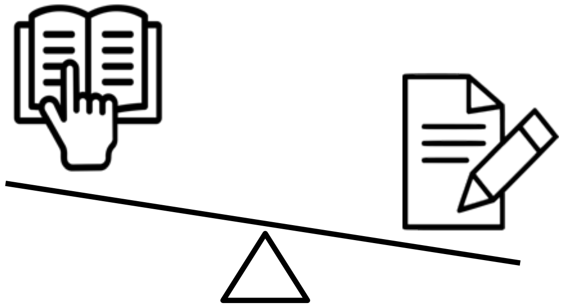
A [much better] bufferpool policy

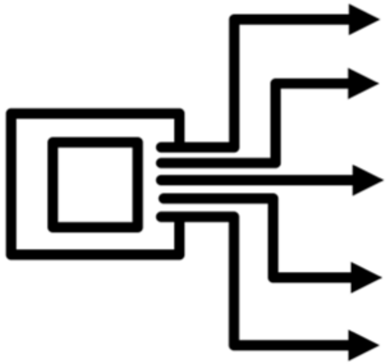
Experimental Evaluation

Future work & Conclusions

New Buffer Pool Page Eviction Algorithm

Use device's properties





New Buffer Pool Page Eviction Algorithms

Instead of evicting one dirty page, evict α pages at one go

1 read ⇒ (max) α evictions

Instead of evicting α pages, evict **one** page and write back α dirty pages at one go

1 read ⇒ (max) α write backs

New Buffer Pool Page Eviction Algorithm(s)

COW (Concurrent Write-back)

COW(n)

n depends on the device concurrency

Concurrent Write-back **dirty** pages from the eviction window of size n

COW-X(n)

Concurrent Write-back eXactly n **dirty** pages

New Buffer Pool Page Eviction Algorithm(s)

COW (Concurrent Write-back)

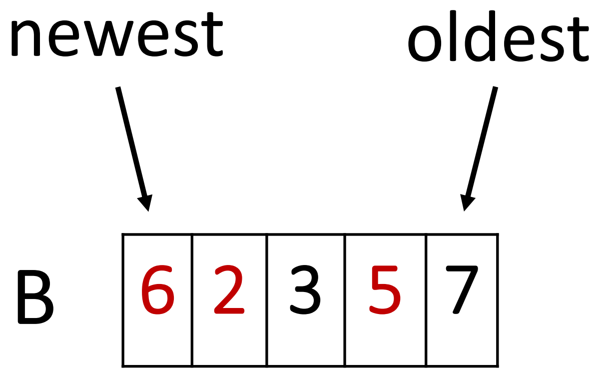
COW(n, E)

Concurrent Eviction of **dirty** pages from the eviction window of size n

COW-X(n, E)

Concurrent Eviction of eXactly n **dirty** pages

Let’s Take a Look at an Example



Let’s assume: $\alpha = 3$ & red indicates dirty page

Read request of page 8 comes

Let's Take a Look at an Example

Candidate for eviction

↓

B	6	2	3	5	7

COW kicks in only when evicting a dirty page

After Eviction:

B	8	6	2	3	5

Write request of page 1 comes

Let's Take a Look at an Example (n = 3)

LRU

Candidate ↓

B	8	6	2	3	5

After Eviction:

B	1	8	6	2	3

COW(n)

Eviction Window ↓

8	6	2	3	5

Let’s Take a Look at an Example (n = 3)

LRU

Candidate ↓

B

8	6	2	3	5

After Eviction:

B

1	8	6	2	3

COW(n)

Eviction Window ↓

8	6	2	3	5

5 & 2 are concurrently written

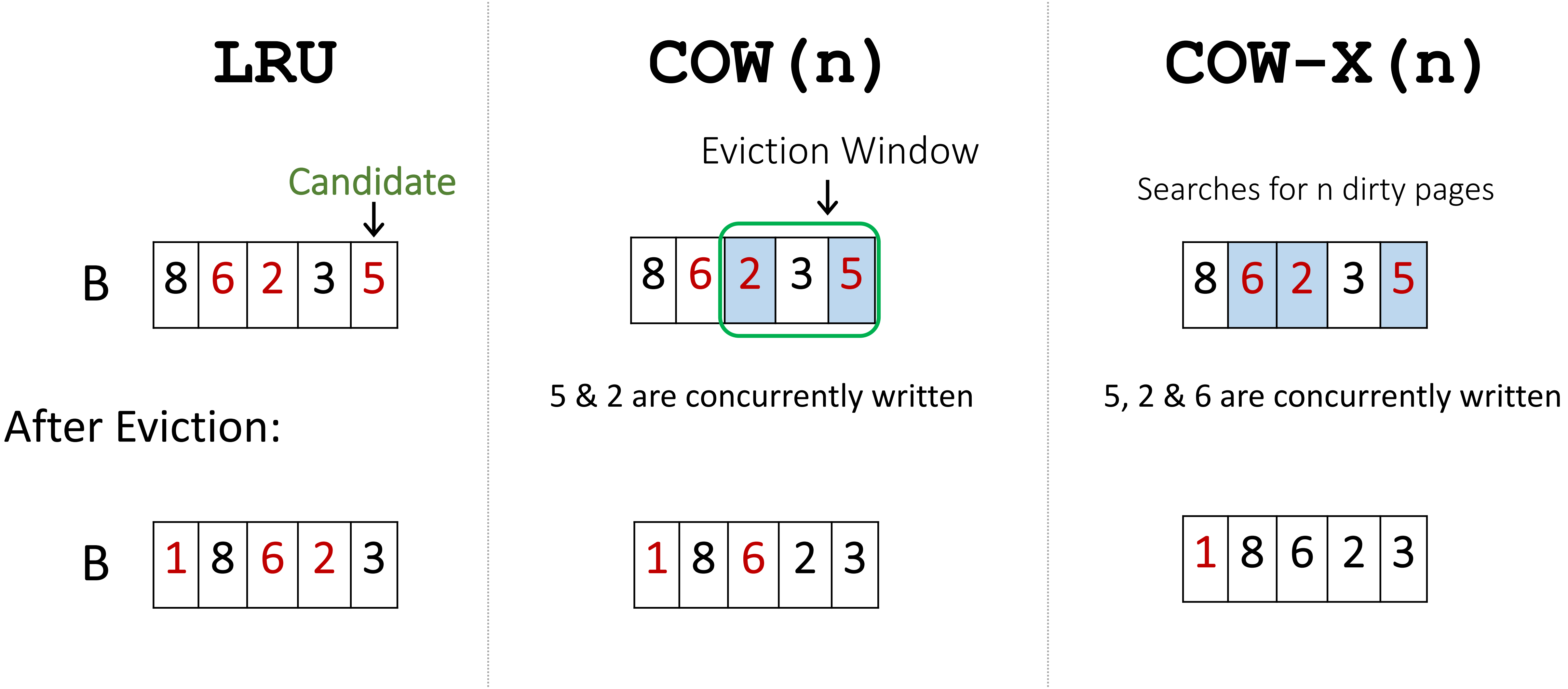
1	8	6	2	3

COW-X(n)

Searches for n dirty pages

8	6	2	3	5

Let’s Take a Look at an Example (n = 3)



What's the tradeoff?

COW(n)

Eviction Window

↓

8	6	2	3	5

5 & 2 are concurrently written

1	8	6	2	3

COW-X(n)

Searches for n dirty pages

8	6	2	3	5

5, 2 & 6 are concurrently written

1	8	6	2	3

Overview

Why an I/O model?

Why not the traditional I/O model?

Asymmetry & Concurrency of Modern Storage Devices

The parametric I/O model

A [much better] bufferpool policy

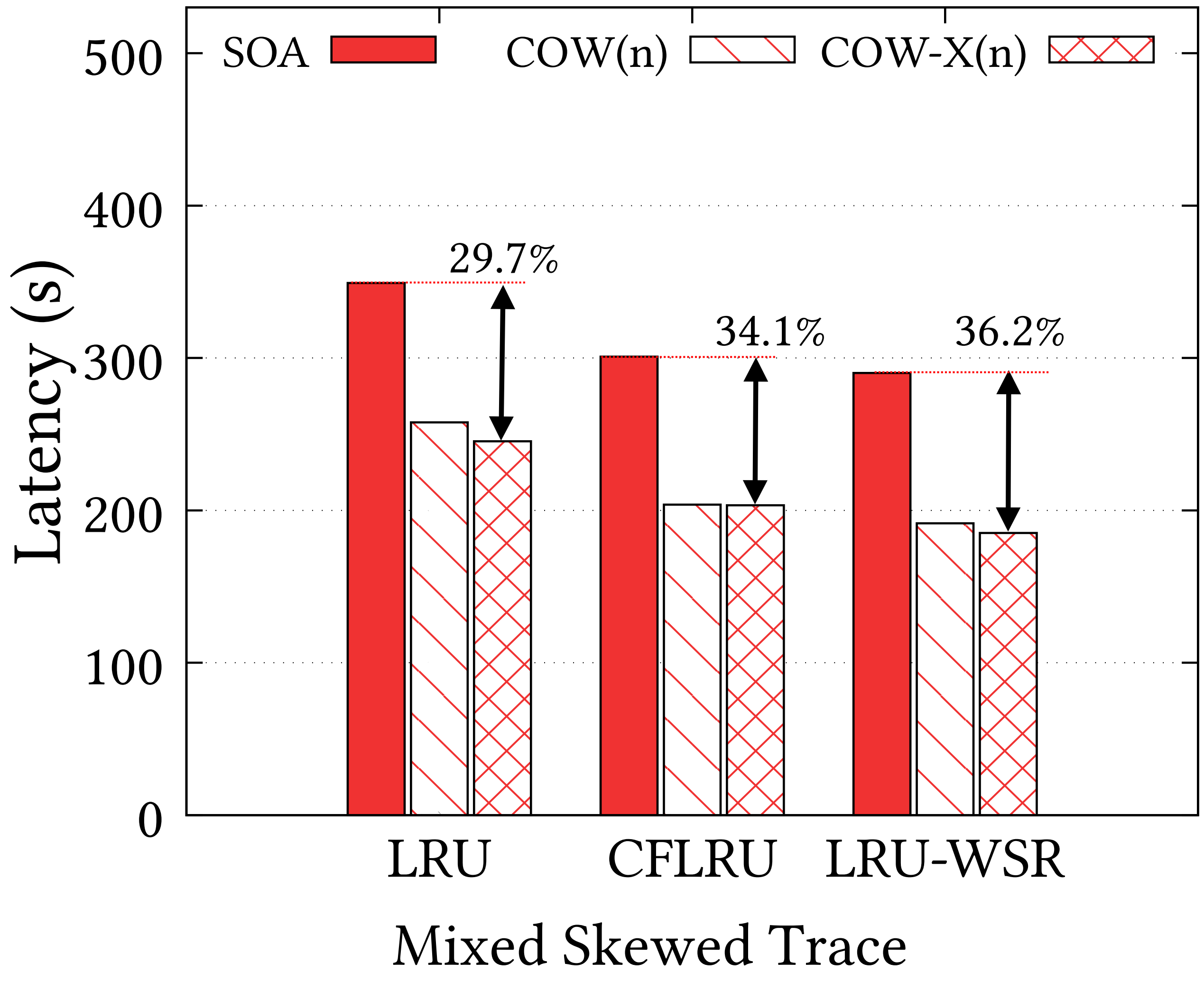
Experimental Evaluation

Future work & Conclusions

Experimental Evaluation

- Comparison w.r.t. LRU, CFLRU, LRU-WSR and their COW counterparts
- Evaluation on 4 synthesized traces and TPC-C benchmark
- 3 storage devices: PCIe SSD, Regular SSD, Virtual SSD

Experimental Evaluation



Device: PCIe SSD

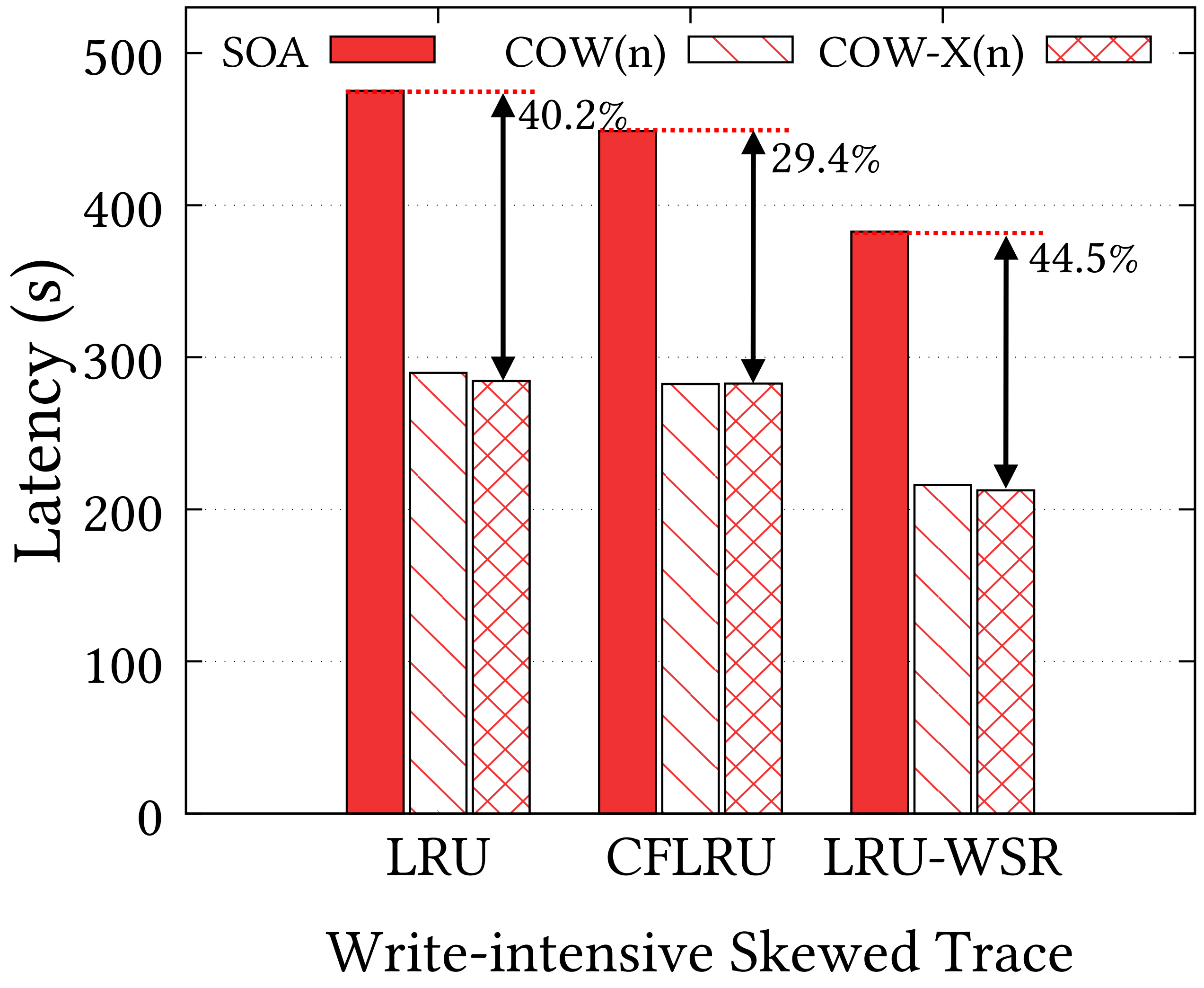
$\alpha = 3$

COW improves runtime significantly

COW-X(n) performs better than COW(n)

Negligible increase in buffer miss (<0.004%)

Experimental Evaluation

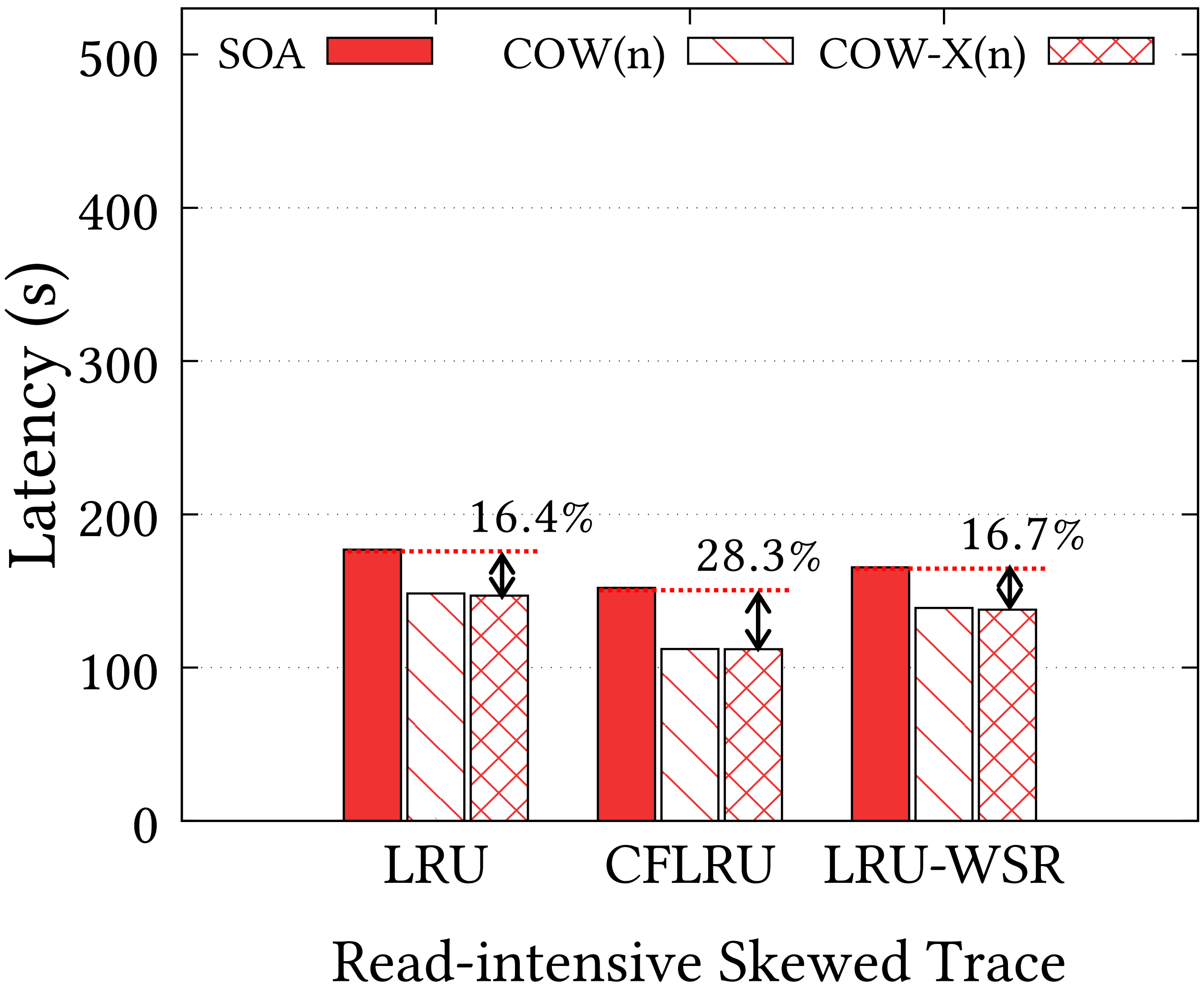


Device: PCIe SSD

α = 3

Gain is more for write-intensive workloads because of efficient writing

Experimental Evaluation



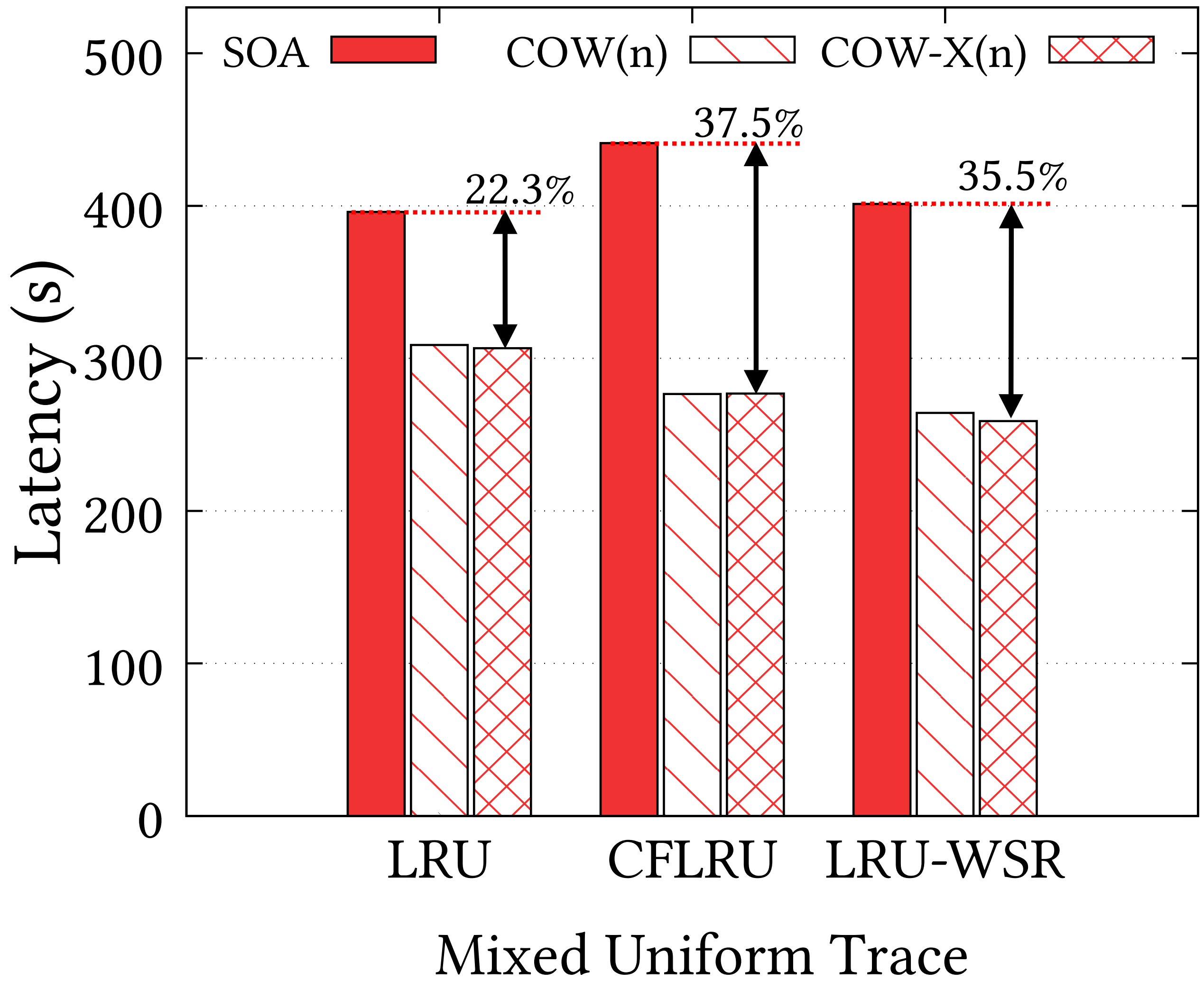
Device: PCIe SSD

$\alpha = 3$

Even read-intensive workload

have substantial gain

Experimental Evaluation

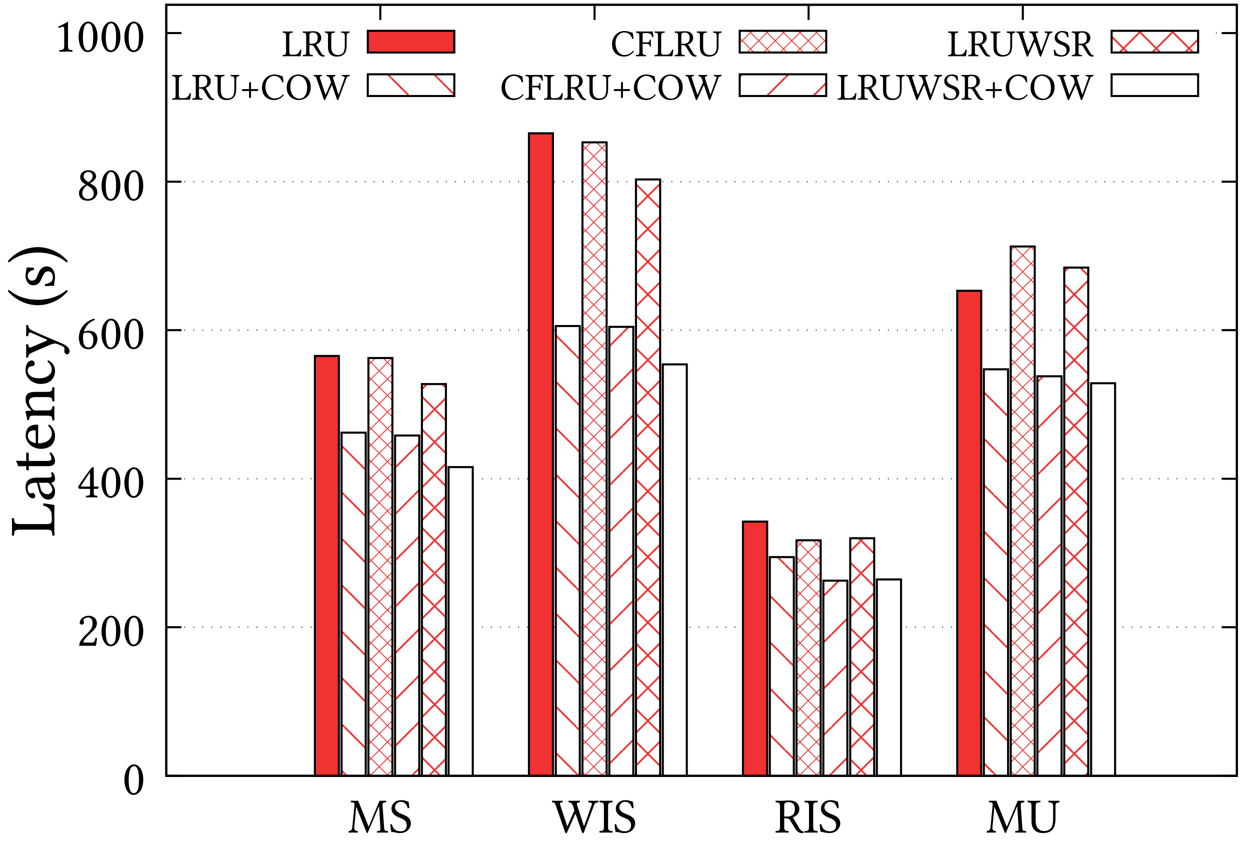


Device: PCIe SSD

$\alpha = 3$

Significant gain for uniform workloads

Experimental Evaluation

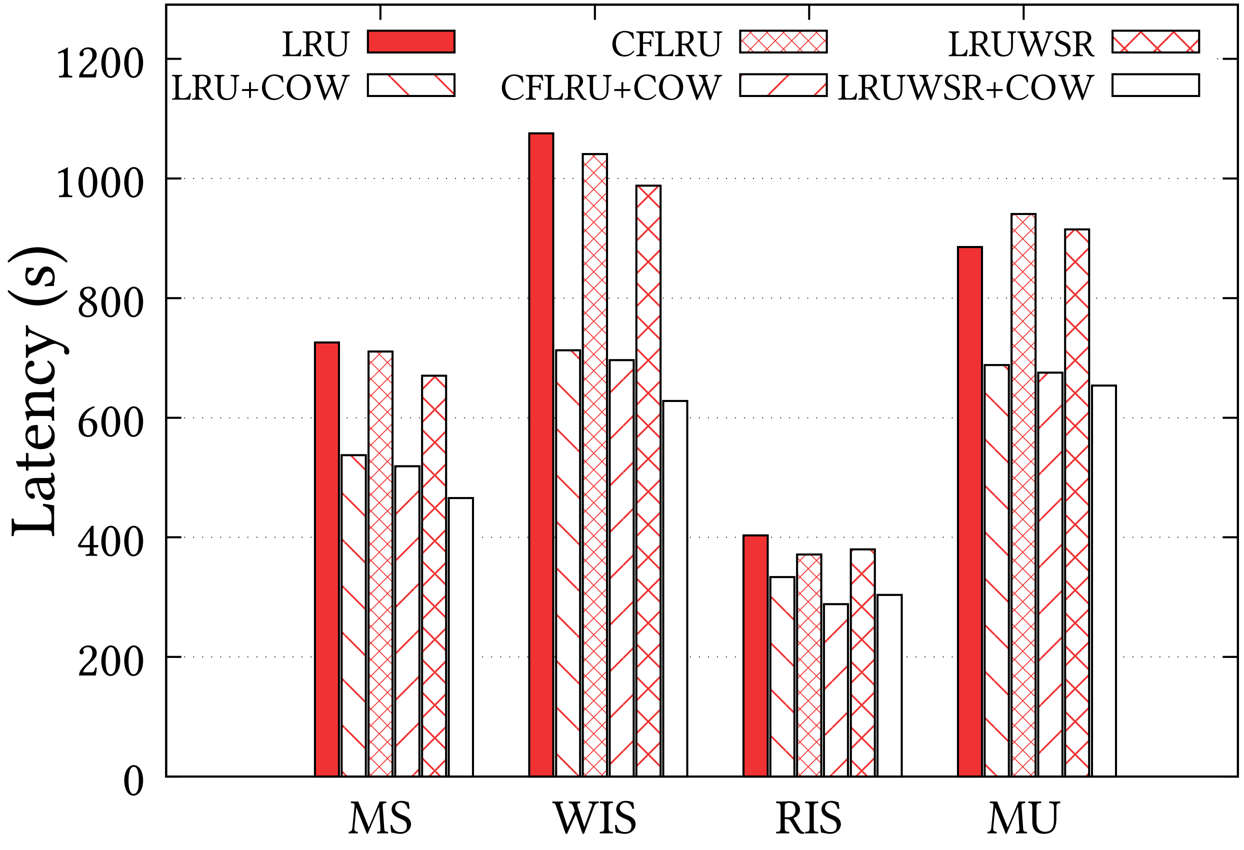


Device: Regular SSD

$\alpha = 1.5$

COW has substantial gain for devices with low asymmetry

Experimental Evaluation

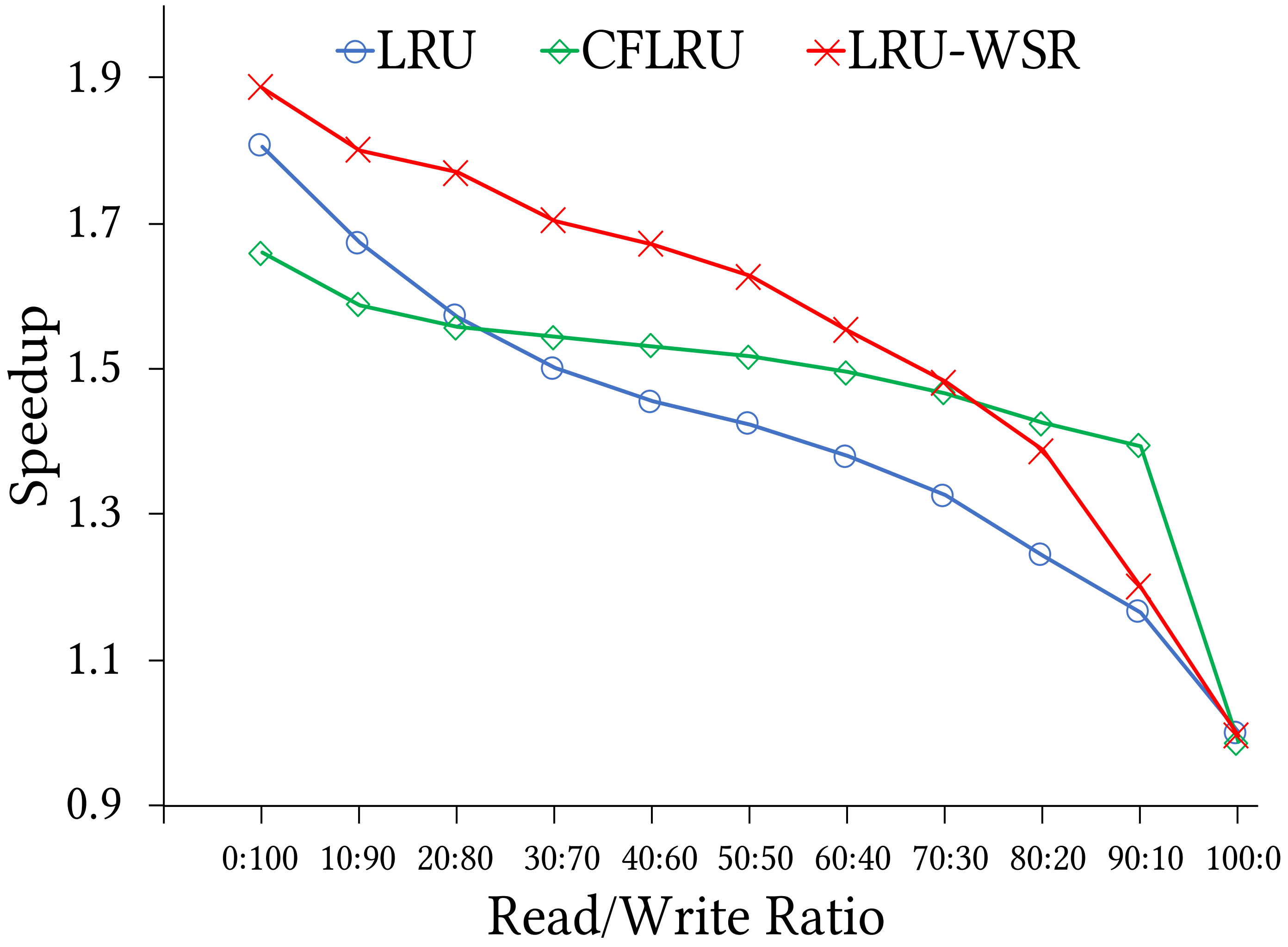


Device: Virtual SSD

α = 2.0

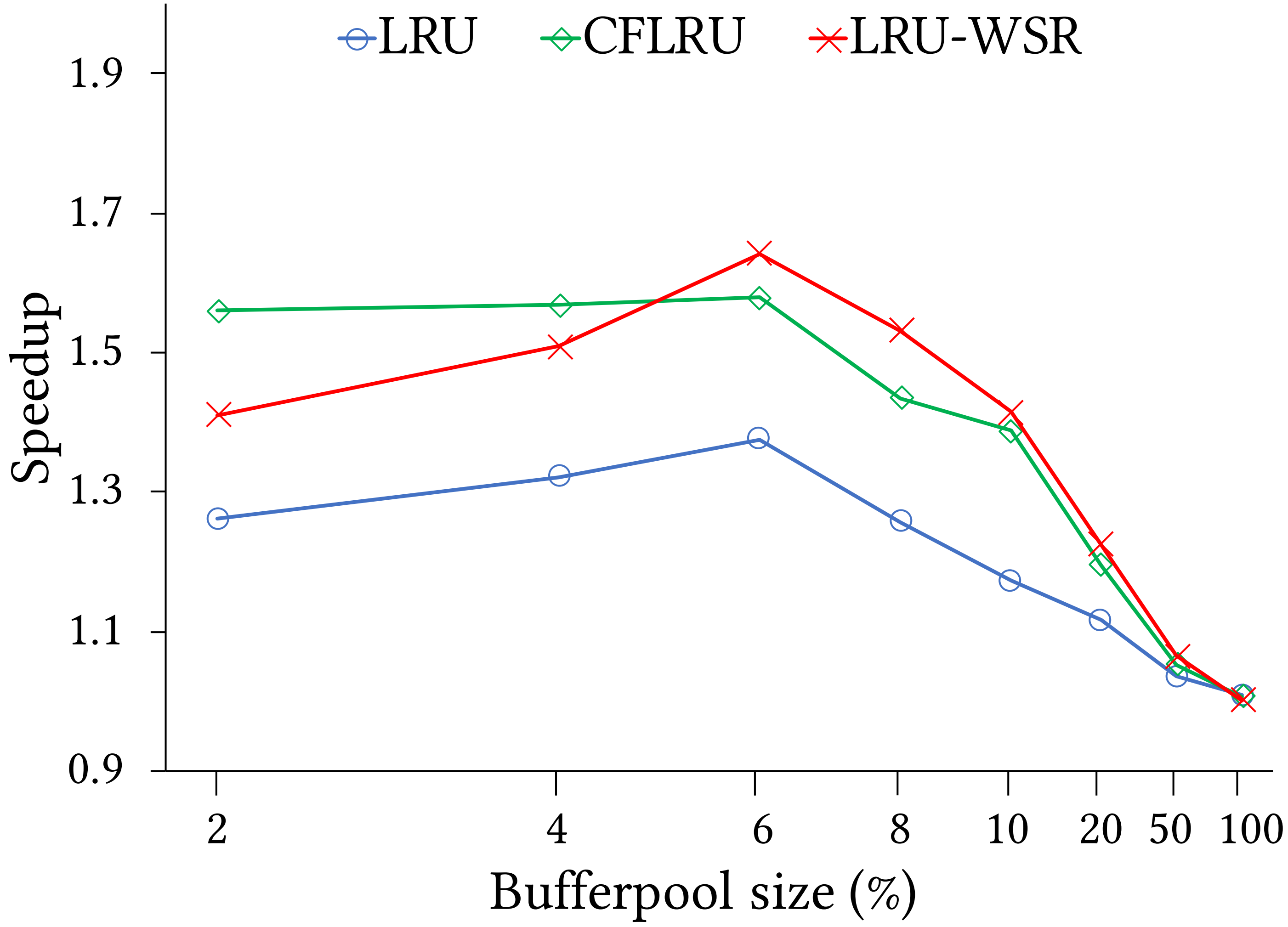
COW has substantial gain for devices with low asymmetry

Impact of Read/Write in Workload



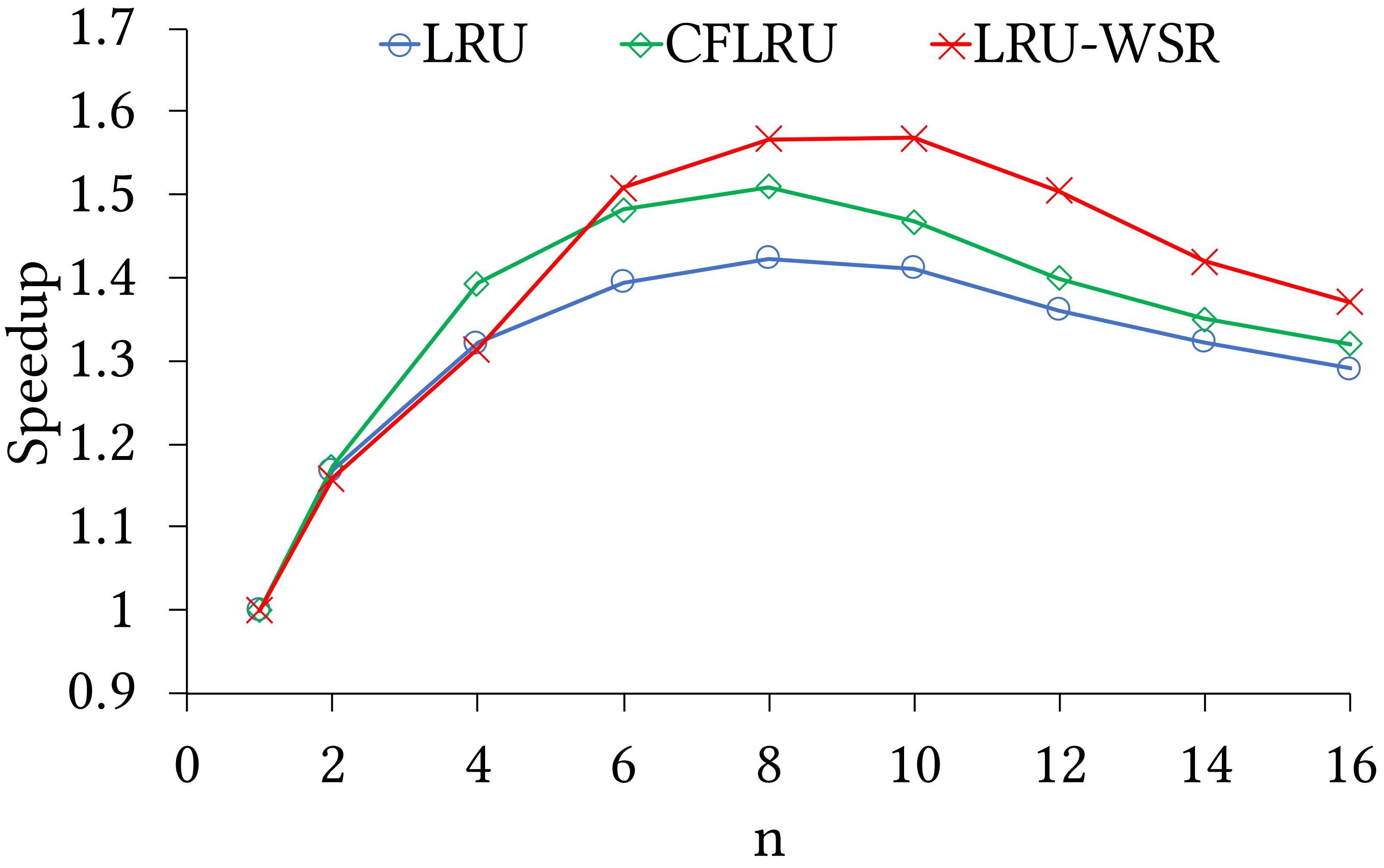
For write heavy workloads, gain of COW can be as high as 2x

Impact of Memory Pressure



In general, COW has better gain for smaller bufferpool size

Impact of Concurrency

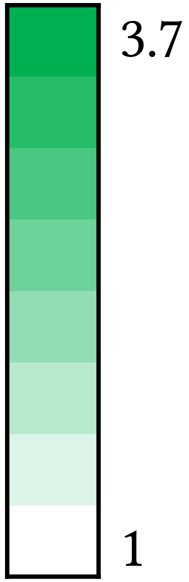


There is an optimal concurrency for each device.

Impact of Asymmetry

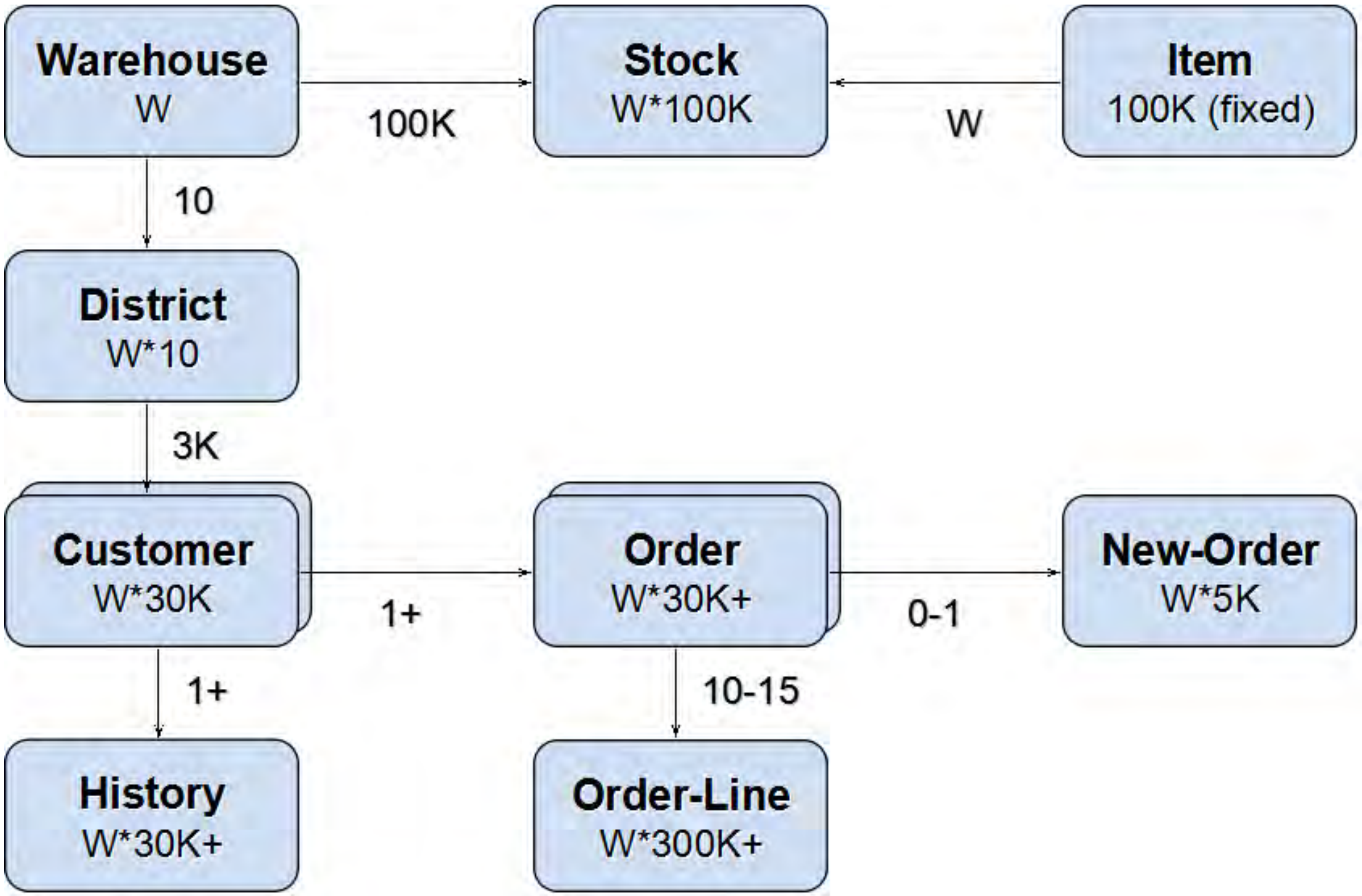
Speedup

n \ α	1	2	3	4	5	6	7	8
8	1.50	1.93	2.31	2.64	2.94	3.20	3.44	3.66
7	1.48	1.89	2.25	2.55	2.82	3.06	3.28	3.47
6	1.46	1.85	2.17	2.45	2.69	2.90	3.08	3.25
5	1.44	1.79	2.07	2.31	2.52	2.69	2.85	2.98
4	1.40	1.70	1.94	2.14	2.30	2.44	2.55	2.65
3	1.34	1.58	1.76	1.90	2.01	2.10	2.18	2.24
2	1.23	1.38	1.48	1.55	1.60	1.65	1.68	1.71
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00



Gain is higher for devices with higher asymmetry

Experimental Evaluation (TPC-C)



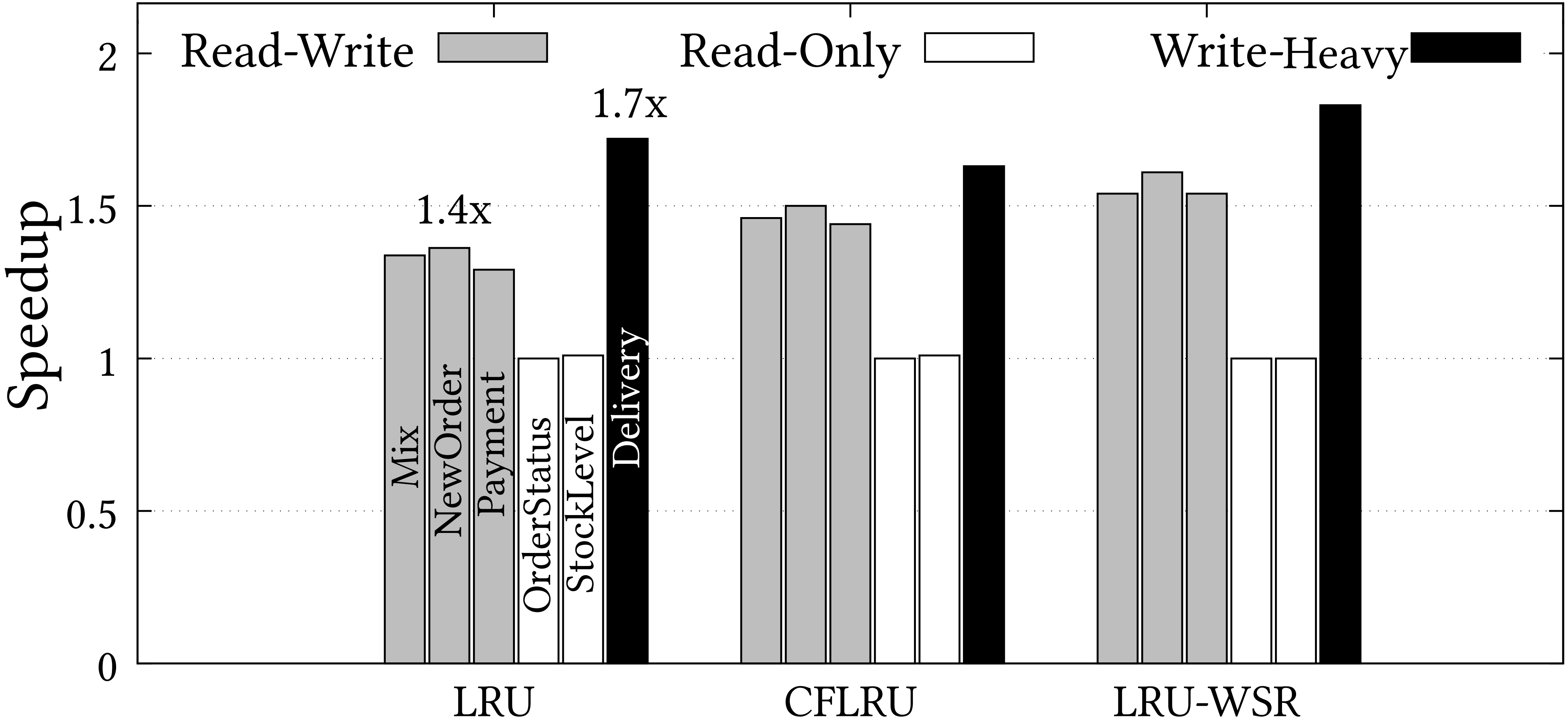
Experimental Evaluation (TPC-C)

Trace was collected from PostgreSQL database

TPC-C consists of 5 transactions

NewOrder (45%)	R/W Mix
Payment (43%)	R/W Mix
OrderStatus (4%)	R-only
StockLevel (4%)	R-only
Delivery (4%)	W-heavy

Experimental Evaluation (TPC-C)



Overview

Why an I/O model?

Why not the traditional I/O model?

Asymmetry & Concurrency of Modern Storage Devices

The parametric I/O model

A [much better] bufferpool policy

Experimental Evaluation

Future work & Conclusions

Guidelines for Algorithm Design

Know Thy Device

Exploit Device Concurrency

Use Concurrency with Care

Asymmetry Controls Performance

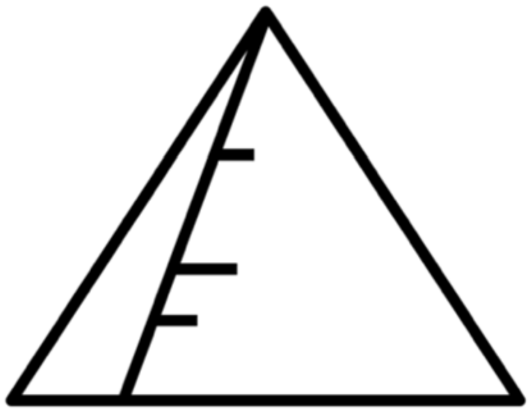
Future Work

Make asymmetry and concurrency part of algorithm design

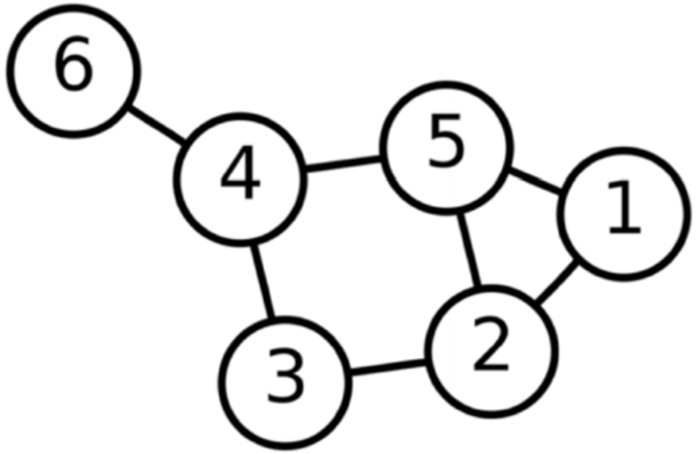
... not simply an engineering optimization

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

index structures

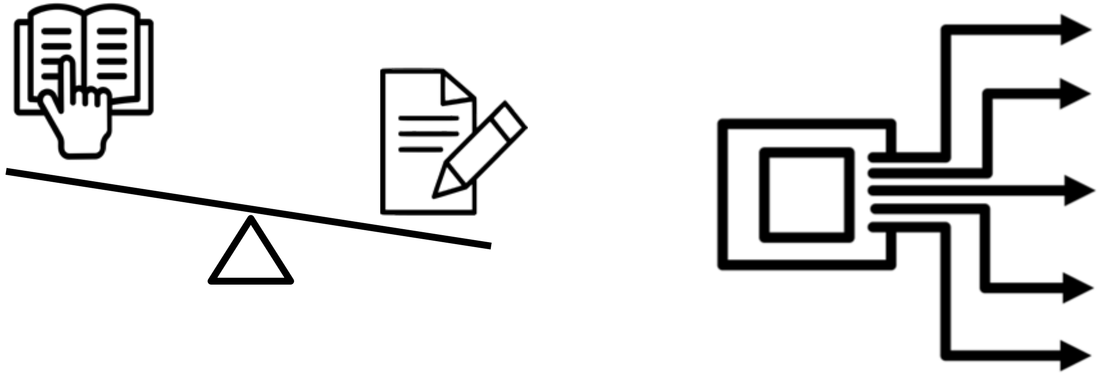


graph traversal algorithms



Conclusion

Modern Storage Devices



Read/Write Asymmetry

Concurrency

Need for a new parametric I/O model

PIO (M, k, α)

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

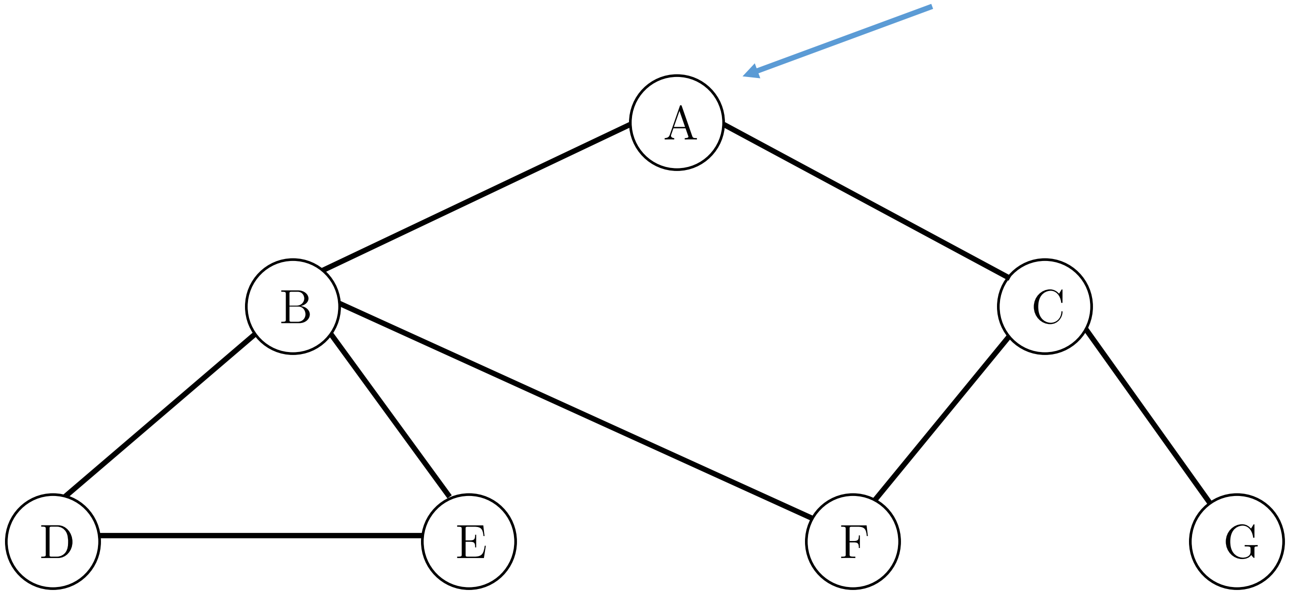
Benefits of PIO (M, k, α)

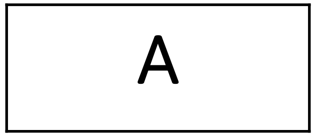
- algorithms tailored to new devices
- Can capture ***any*** new device

Thank You!!!

Questions?

Traditional Graph Traversal (DFS)



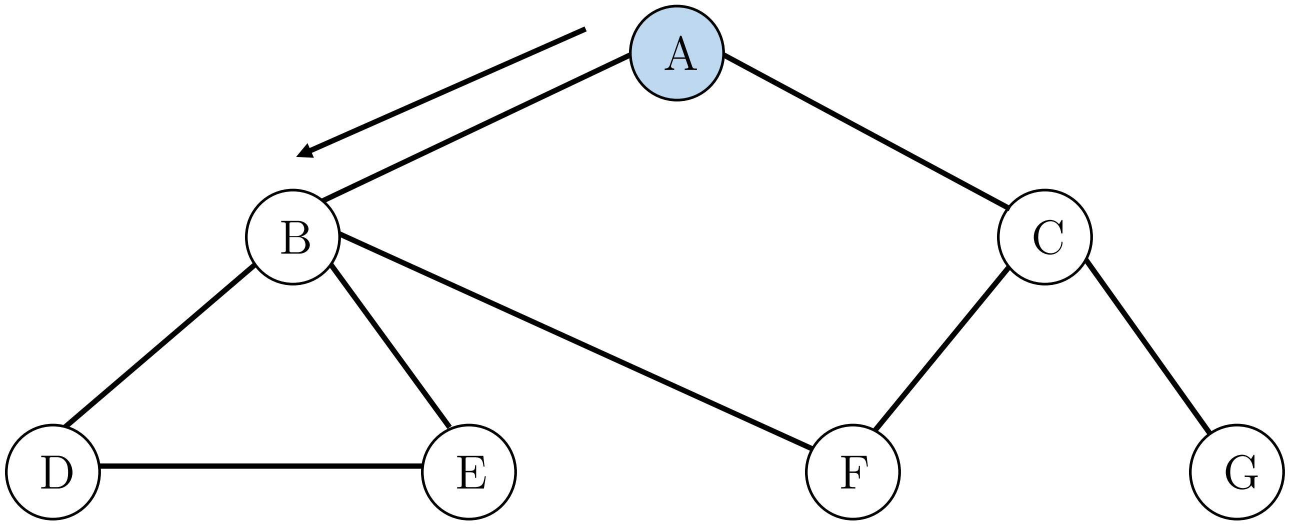


Stack

Traditional Graph Traversal (DFS)

B
C

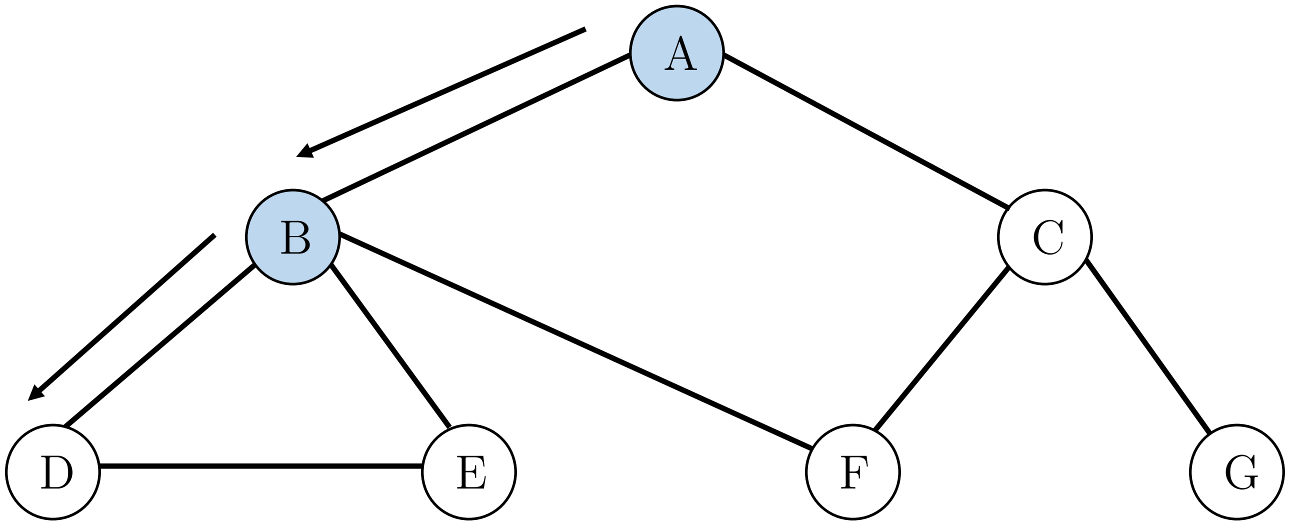
Stack



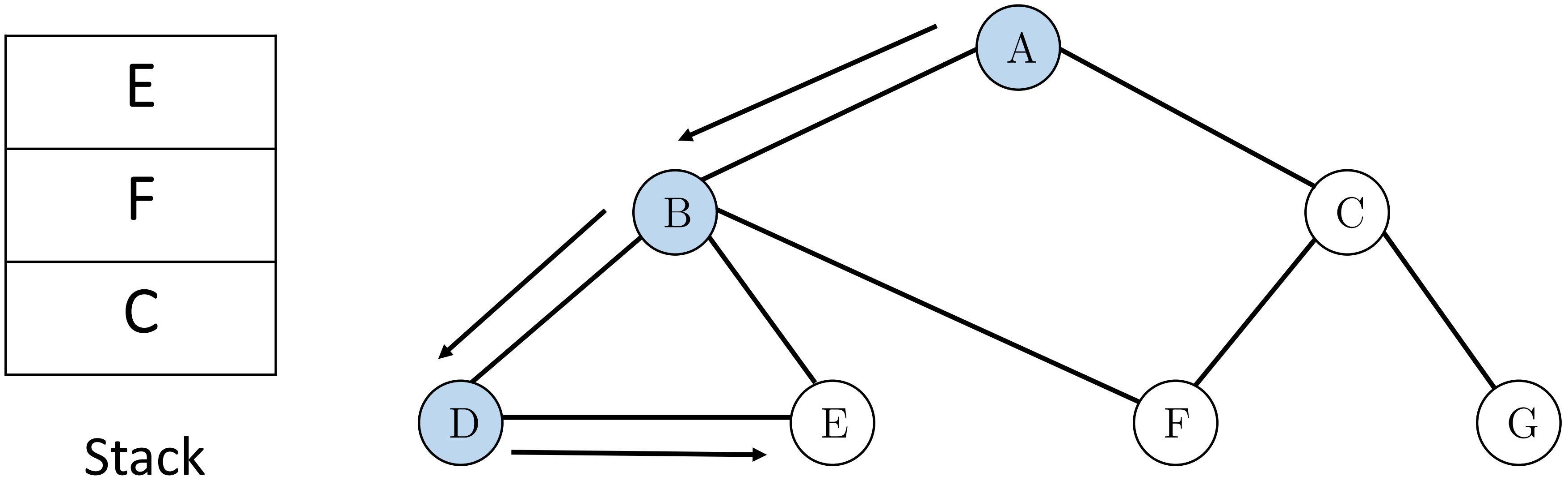
Traditional Graph Traversal (DFS)

D
E
F
C

Stack



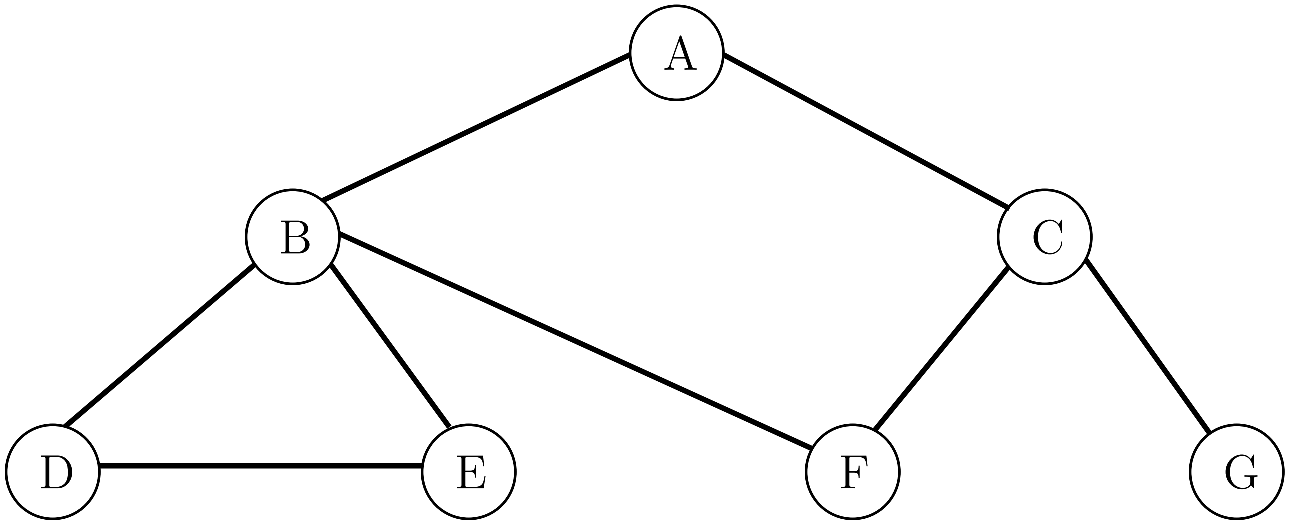
Traditional Graph Traversal (DFS)



And so on…

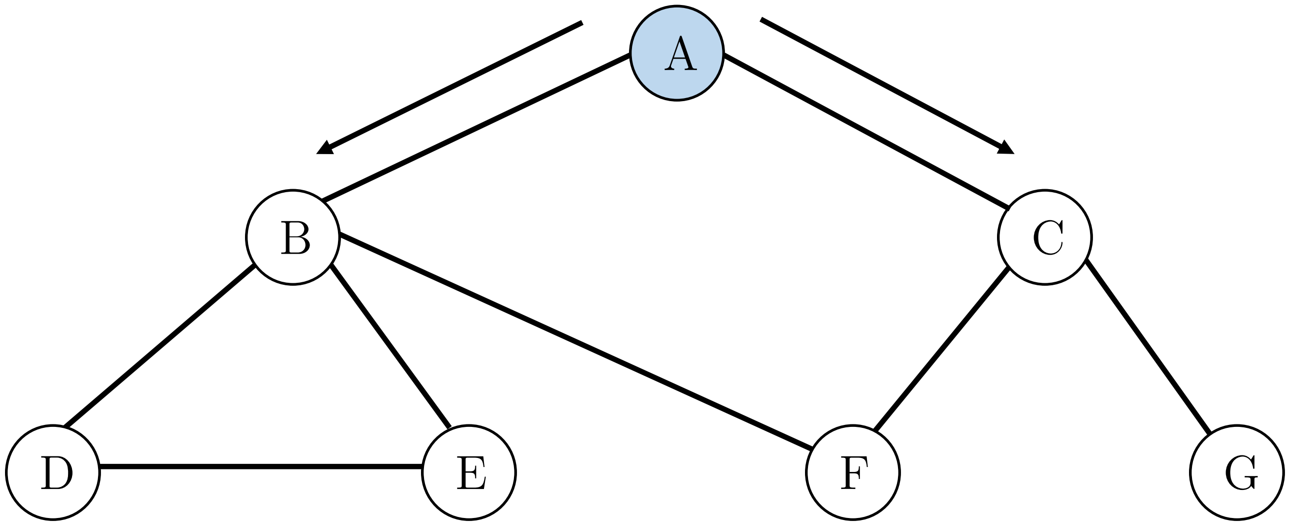
Graph Traversal (DFS) with Concurrency

A



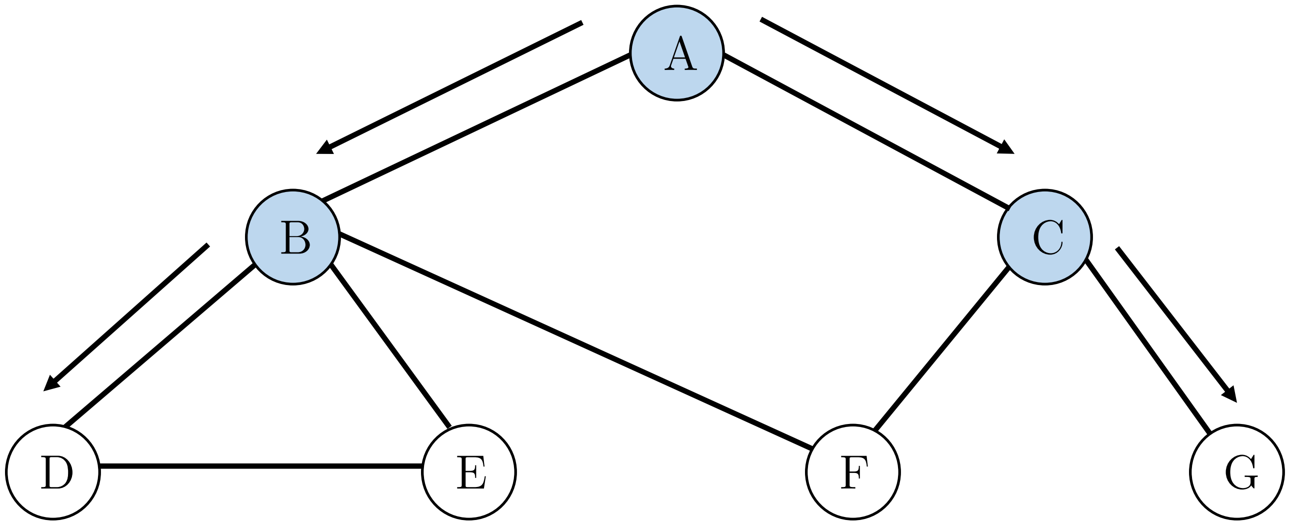
Graph Traversal (DFS) with Concurrency

B
C

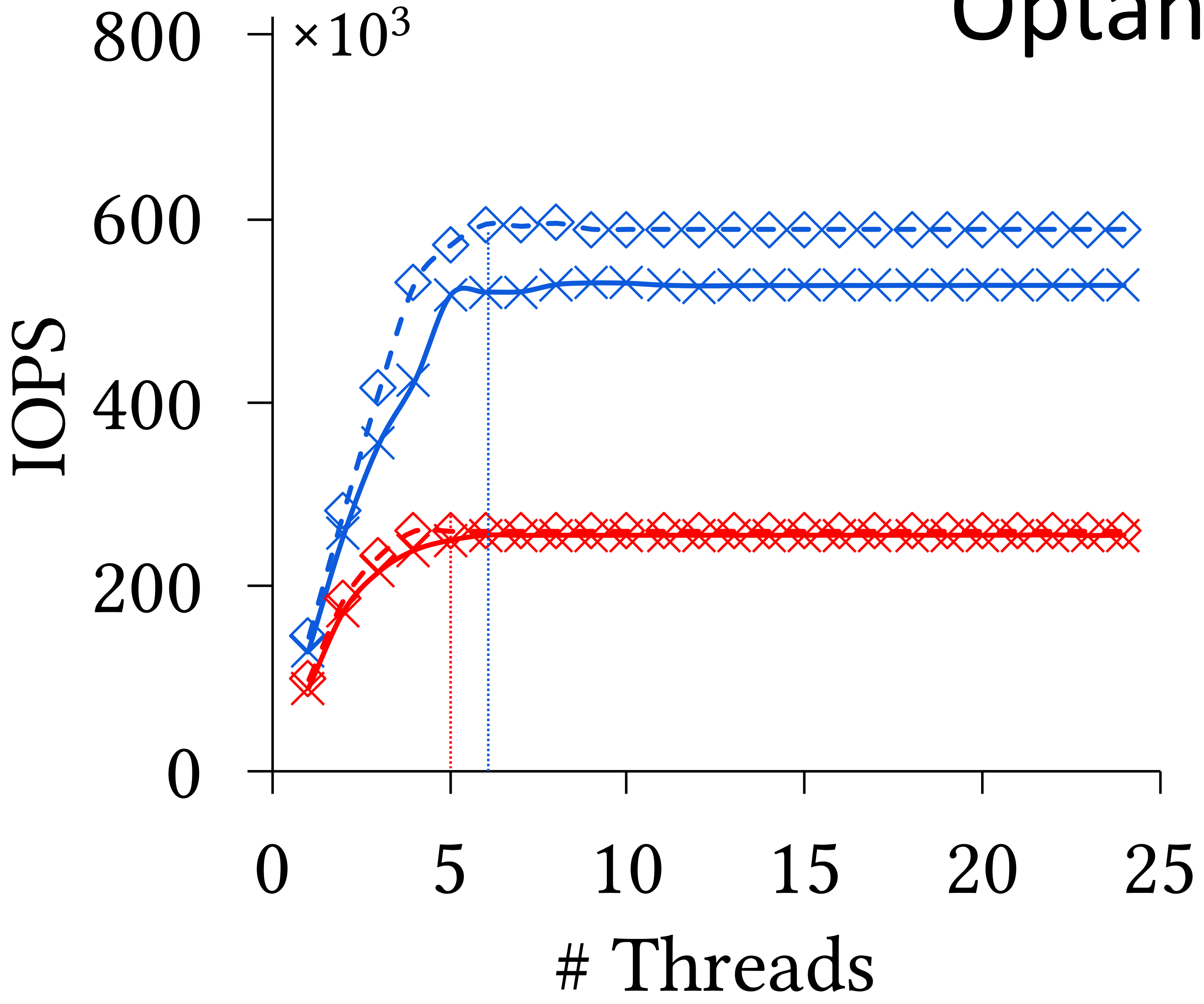


Graph Traversal (DFS) with Concurrency

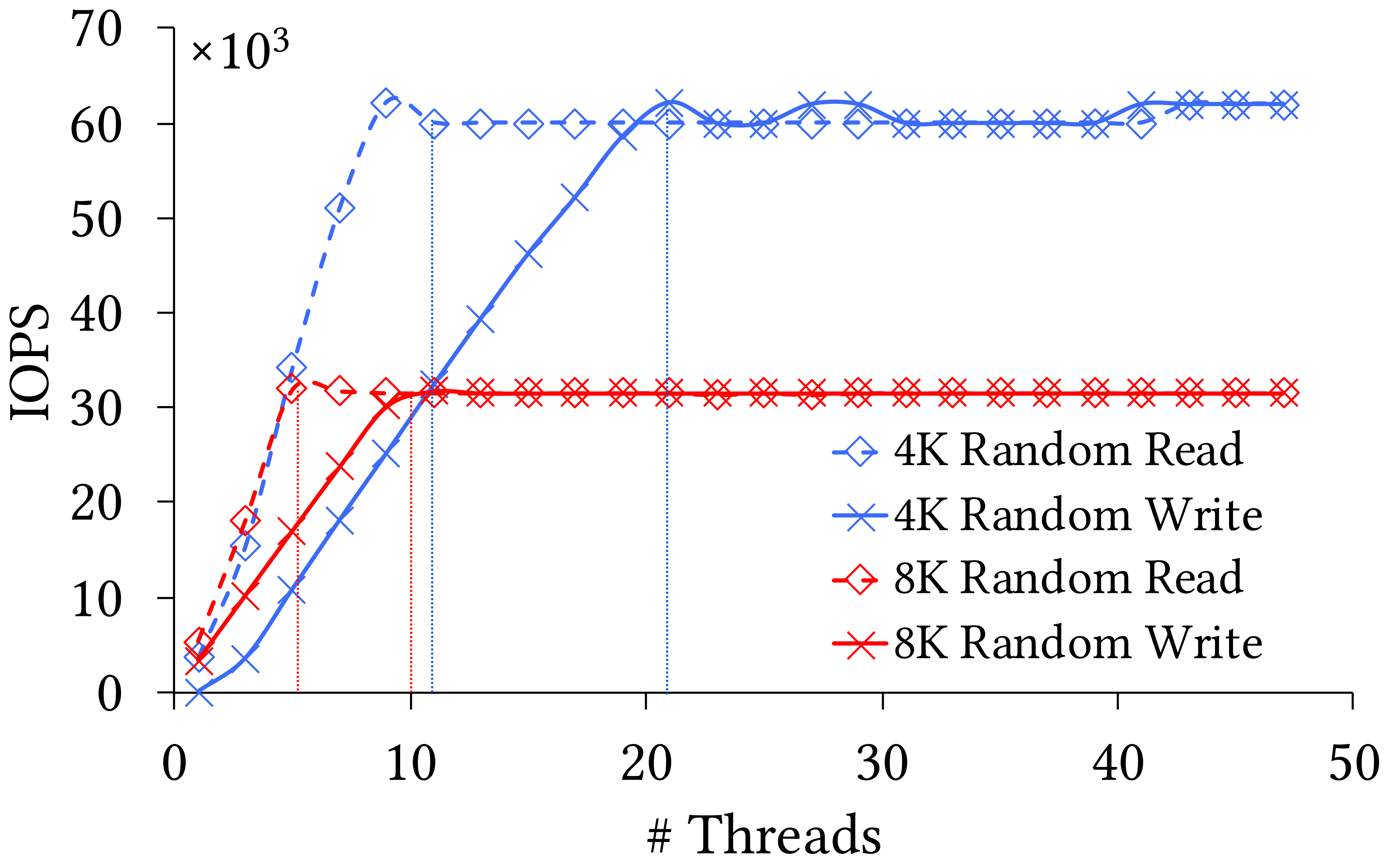
D
E
F
G



Optane SSD

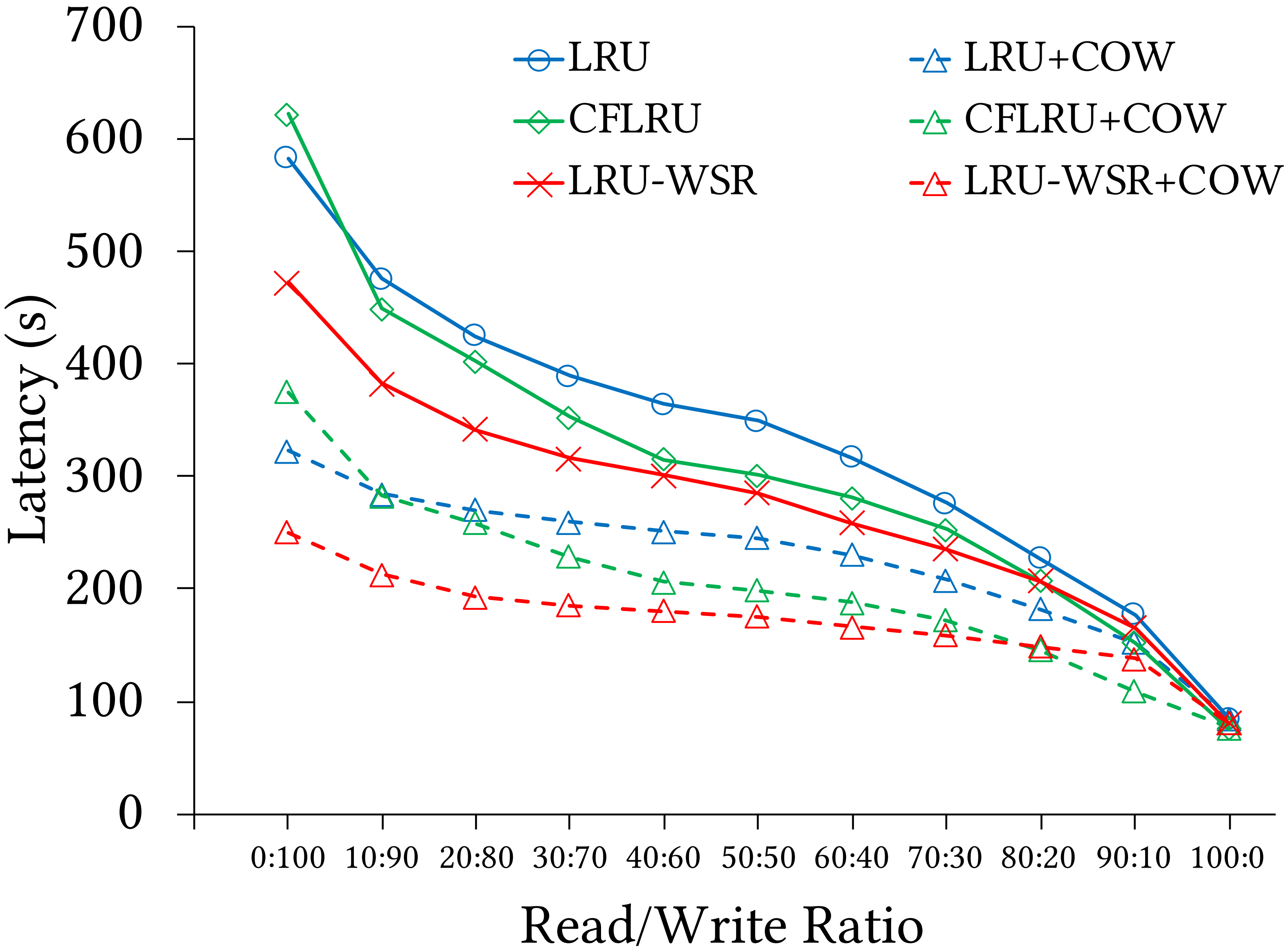


Virtual SSD

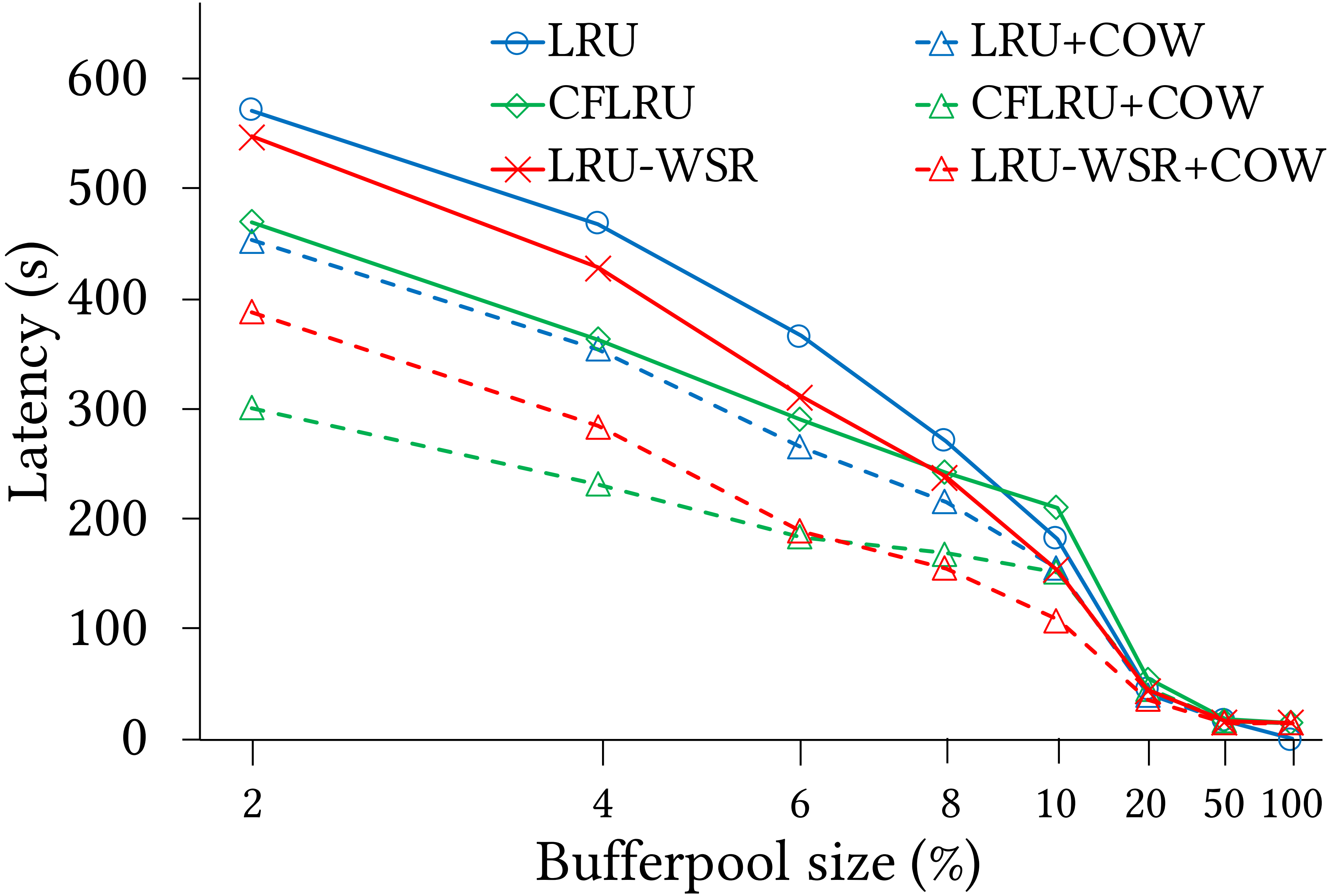

×10³
IOPS
Threads
0
10
20
30
40
50
60
70
0
10
20
30
40
50
4K Random Read
4K Random Write
8K Random Read
8K Random Write

Trace	#Op	Buffer Size (#page)	Disk Size (#page)	R/W Ratio	Locality
Mixed Skewed (MS)	2M	20K	344K	50/50	90/10
Write-Intensive Skewed (WIS)	2M	20K	344K	10/90	90/10
Read-Intensive Skewed (RIS)	2M	20K	344K	90/10	90/10
Mixed Uniform (MU)	2M	20K	344K	50/50	50/50

Trace	LRU		LRU+COW(n)		LRU+COW-X(n)		CFLRU		CFLRU+COW(n)		CFLRU+COW-X(n)		LRU-WSR		LRU-WSR+COW(n)		LRU-WSR+COW-X(n)	
	#miss	#write	Δ#miss	Δ#write	Δ#miss	Δ#write	#miss	#write	Δ#miss	Δ#write	Δ#miss	Δ#write	#miss	#write	Δ#miss	Δ#write	Δ#miss	Δ#write
MS	1097316	686599	● 0	▲ 71	● 0	▲ 89	1087389	619836	▼ -8	▲ 100	▼ -8	▲ 100	1087716	622895	▼ -6	▲ 44	▲ 2	▲ 145
WIS	1081962	1011848	● 0	▲ 108	● 0	▲ 144	1095380	995609	▲ 4	▲ 119	▲ 4	▲ 119	1095082	992068	● 0	▲ 84	▲ 6	▲ 140
RIS	1101735	180232	● 0	▲ 28	● 0	▲ 40	1069567	134890	▲ 32	▲ 30	▲ 32	▲ 30	1087372	160951	▼ -10	▲ 4	▲ 47	▲ 105
MU	1884058	960019	● 0	▼ -31	● 0	▼ -27	1884065	947231	● 0	▲ 17	● 0	▲ 17	1884030	947167	▲ 1	▲ 7	▲ 3	▲ 17



Latency (s)
700
600
500
400
300
200
100
0
LRU
LRU+COW
CFLRU
CFLRU+COW
LRU-WSR
LRU-WSR+COW
0:100
10:90
20:80
30:70
40:60
50:50
60:40
70:30
80:20
90:10
100:0
Read/Write Ratio


LRU
LRU+COW
CFLRU
CFLRU+COW
LRU-WSR
LRU-WSR+COW
Latency (s)
600
500
400
300
200
100
0
2
4
6
8
10
20
50
100
Bufferpool size (%)