The Need for a New I/O Model

Tarikul Islam Papon
papon@bu.edu

Manos Athanassoulis
mathan@bu.edu

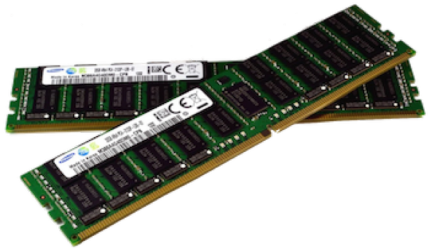
presentation at CIDR 2021

Modeling Performance

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

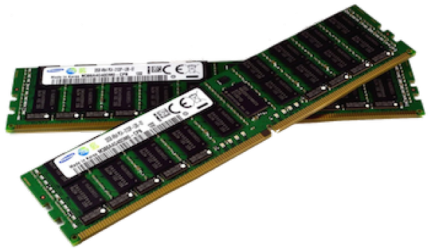
… is probably one of the most commonly read phrases in SIGMOD papers.

Traditional I/O Model



Small, fast main memory
(size M)

Traditional I/O Model

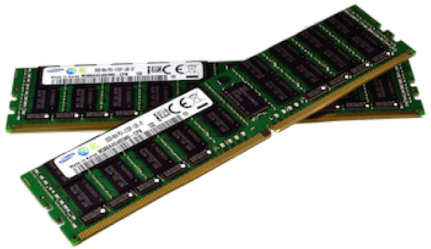


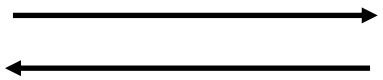
Small, fast main memory
(size M)

Large, slow external memory

Traditional I/O Model

One I/O at a time



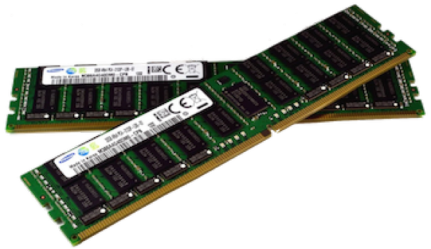


Small, fast main memory
(size M)

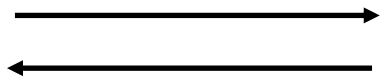
Large, slow external memory

Traditional I/O Model

0 access cost



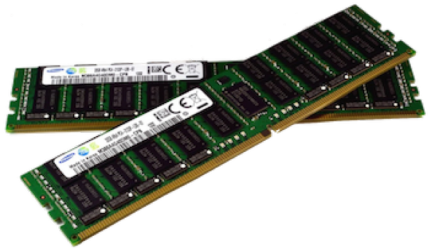
Small, fast main memory
(size M)



Large, slow external memory

Traditional I/O Model

0 access cost



Small, fast main memory
(size M)

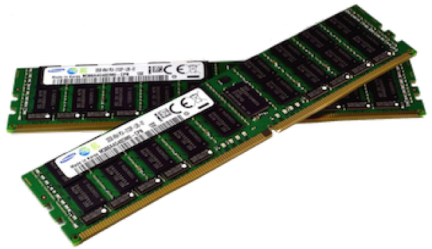
Transfer cost
1 unit

Large, slow external memory

Traditional I/O Model

total cost ≅ total # reads/writes to disk

0 access cost



Small, fast main memory
(size M)

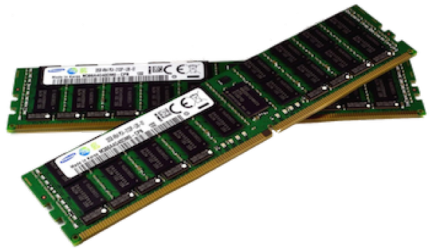
Transfer cost
1 unit

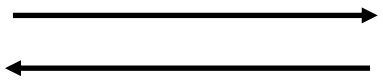
Large, slow external memory

Traditional I/O Model

Two (outdated) assumptions

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





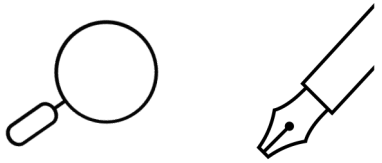
Small, fast main memory
(size M)

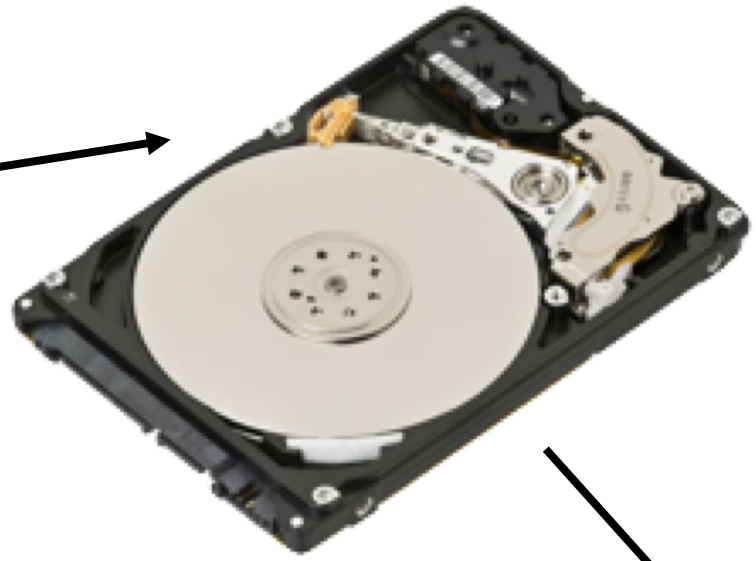
Large, slow external memory

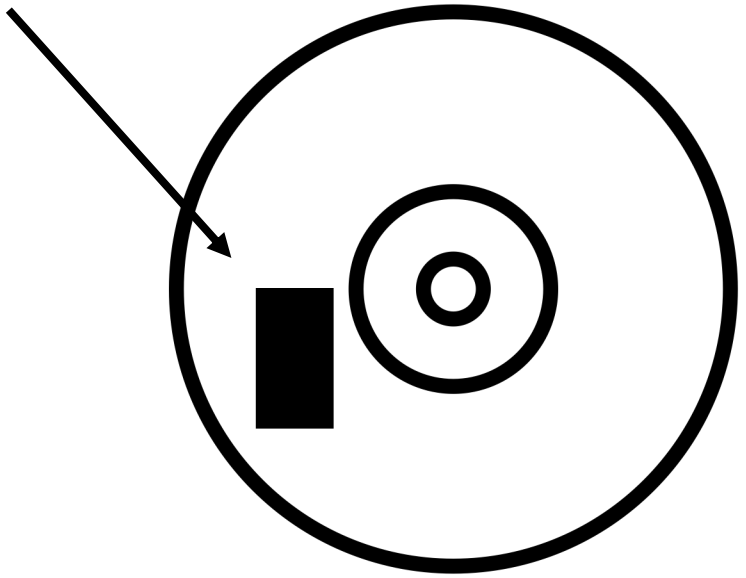
HDD vs. SSD

Read/Write a page from/to HDD

read or write page 5







Accessing a page from SSD

read page 5

Navigate to

- Channel
- Chip
- Die
- Plane
- Block

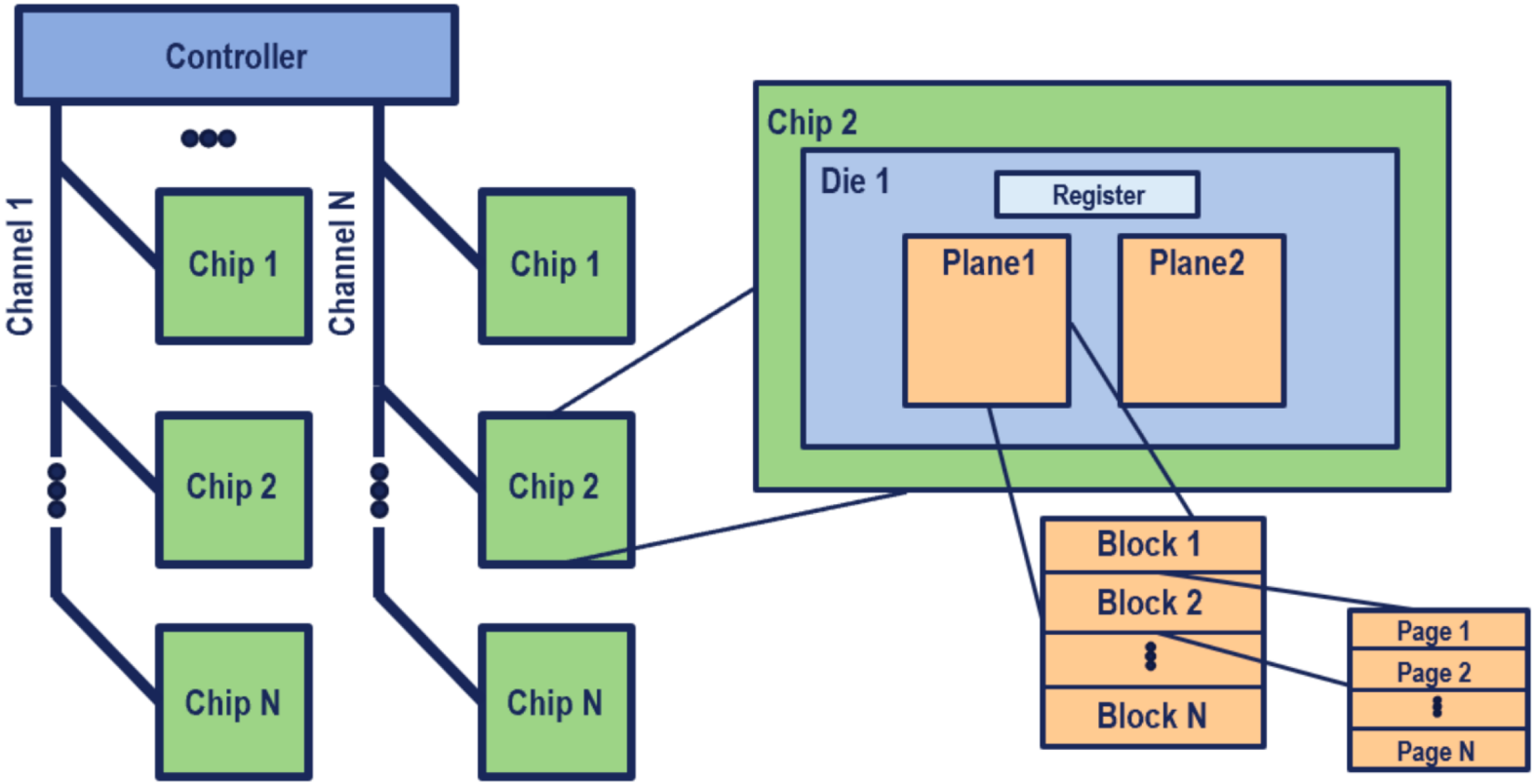


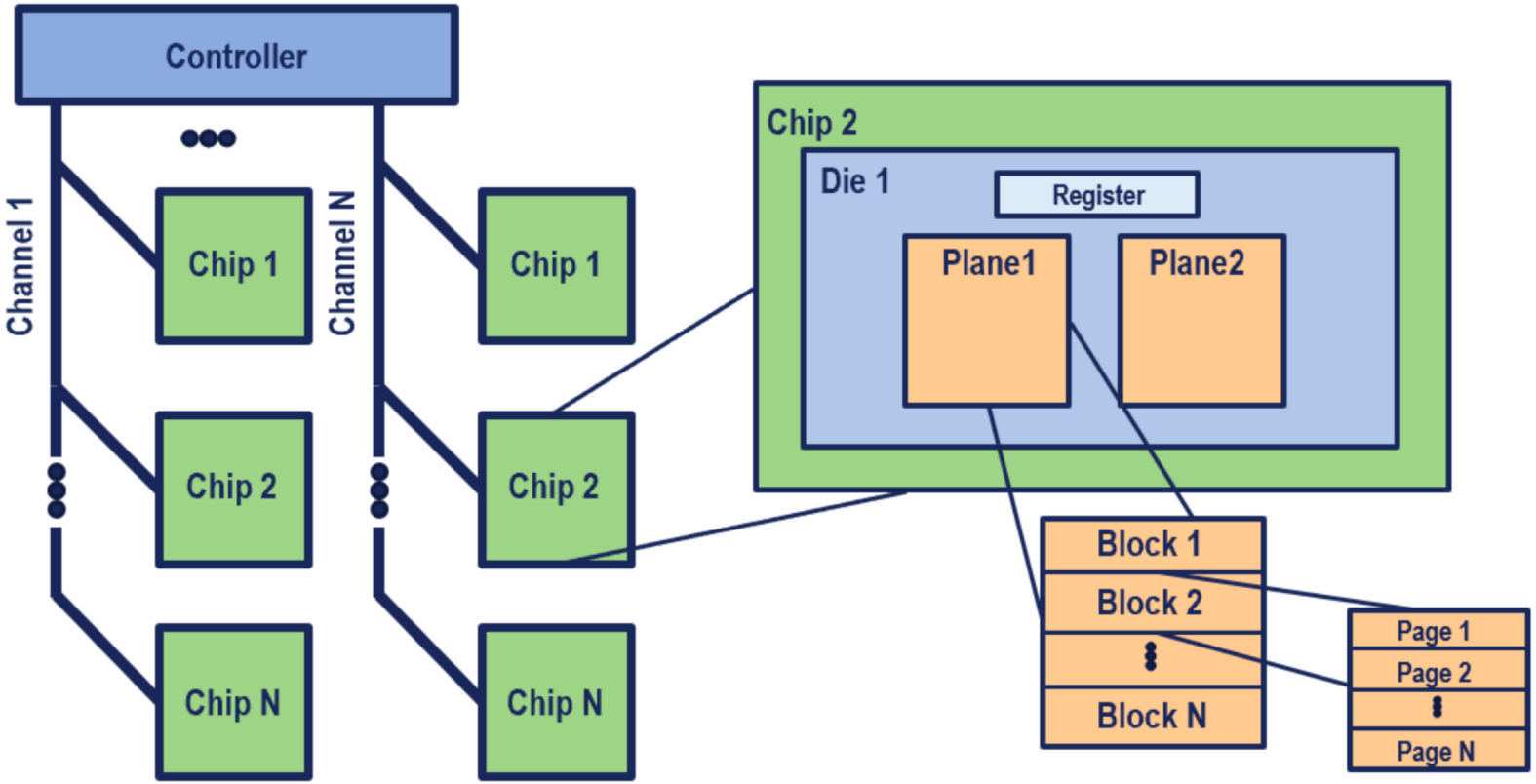
Image Source: Garrett et al. “Enabling Intra-Plane ParallelBlock Erase in NAND Flash to Alleviate the Impact of Garbage Collection”

Accessing a page from SSD

read page 5

Navigate to

- Channel
- Chip
- Die
- Plane
- Block



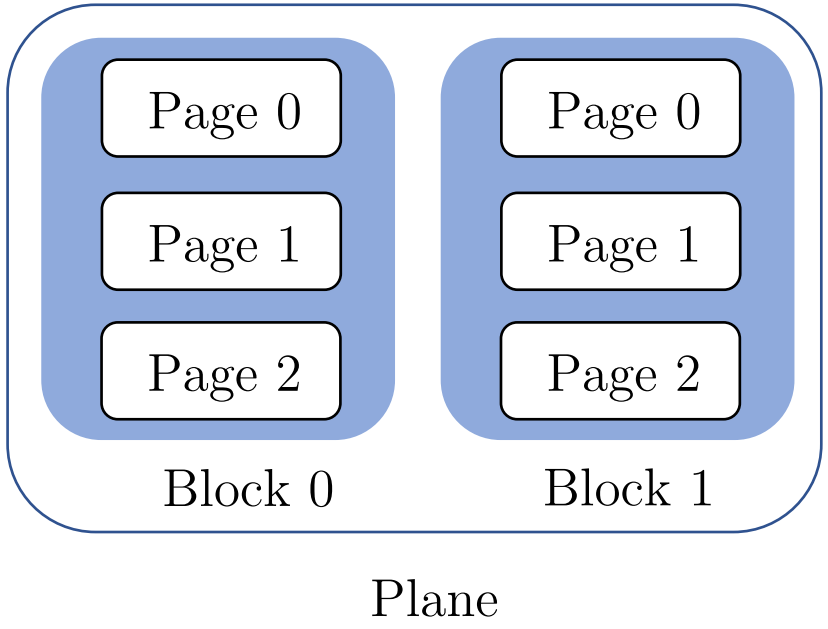
Opportunities for *concurrently* reading or writing multiple pages

Image Source: Garrett et al. “Enabling Intra-Plane ParallelBlock Erase in NAND Flash to Alleviate the Impact of Garbage Collection”

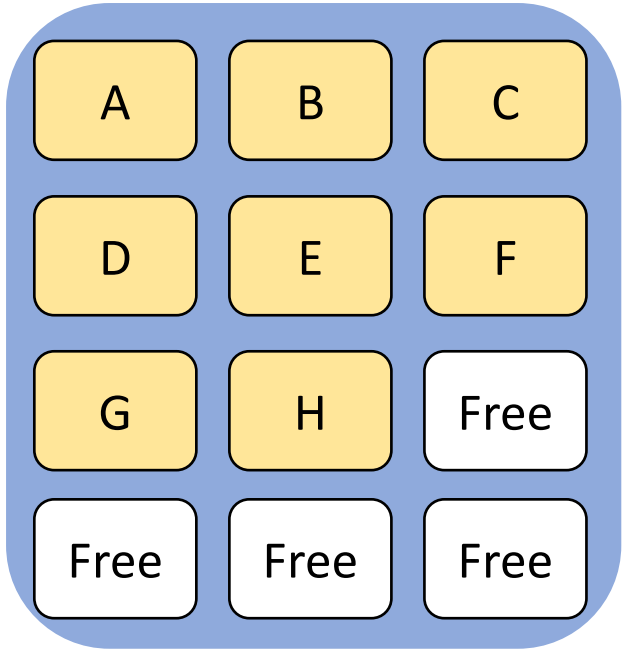
Writes on SSD

Out-of-place updates cause invalidation

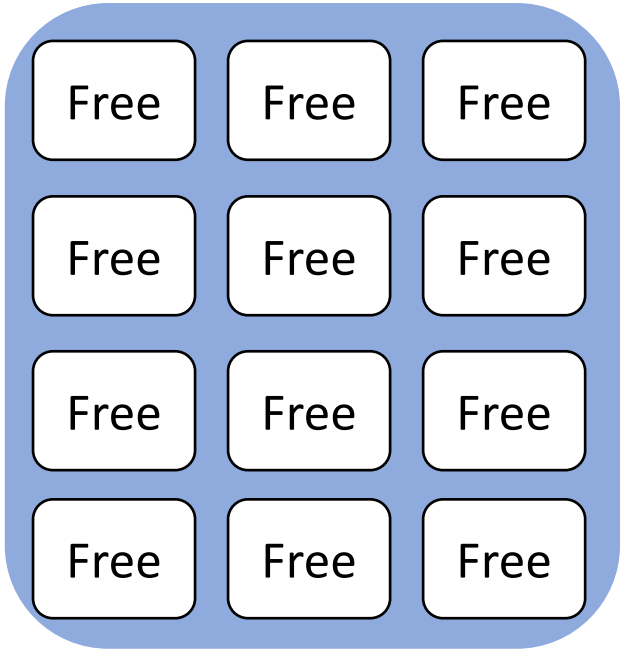
Invalidation causes **garbage collection**



Writes on SSD



Block 0



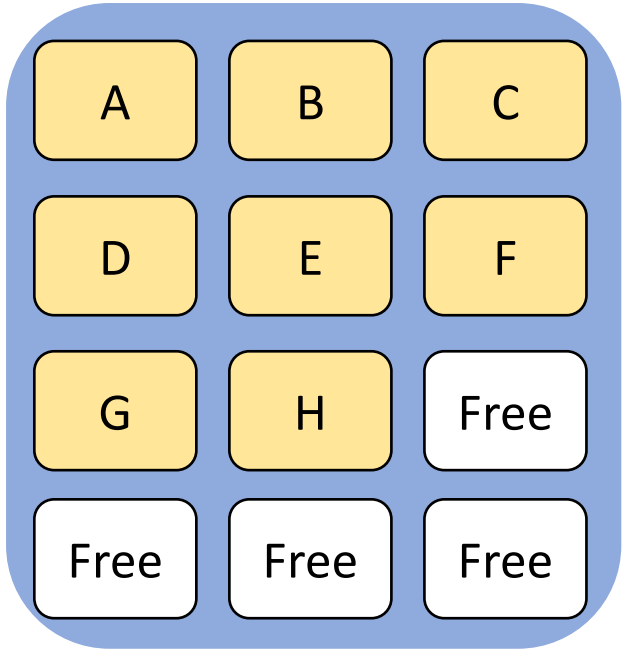
Block 1

Writing in a free page isn't costly!

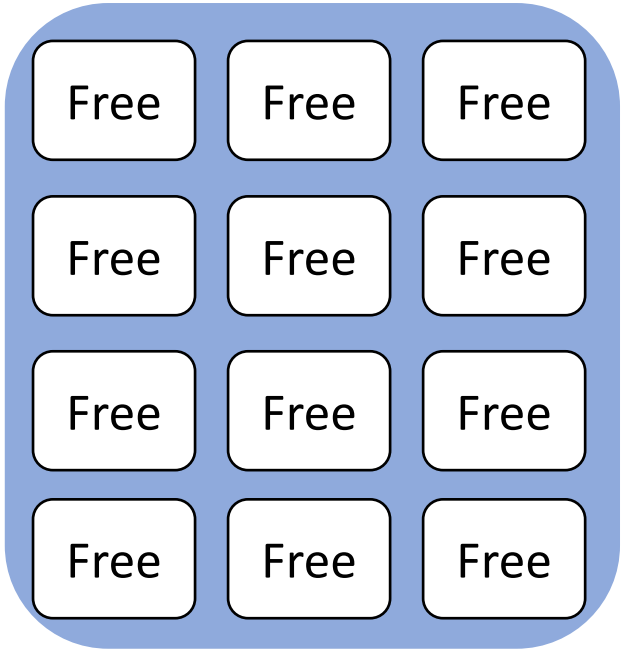
Writes on SSD

Update

A, B, C, D



Block 0

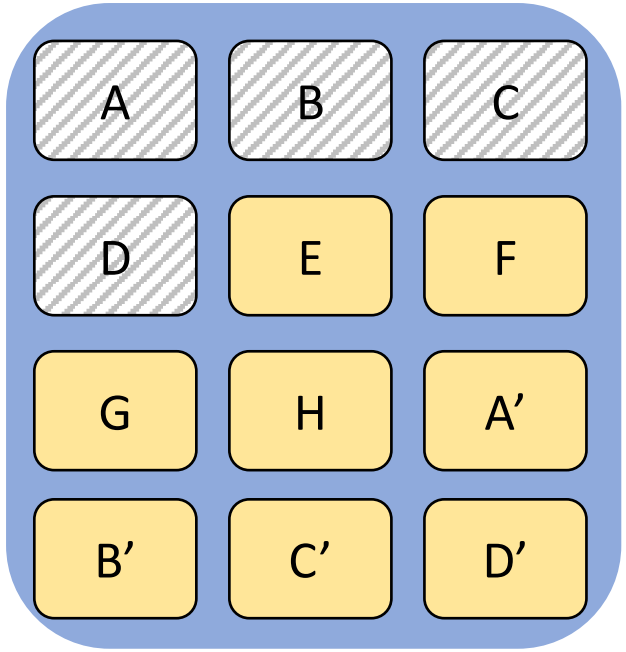


Block 1

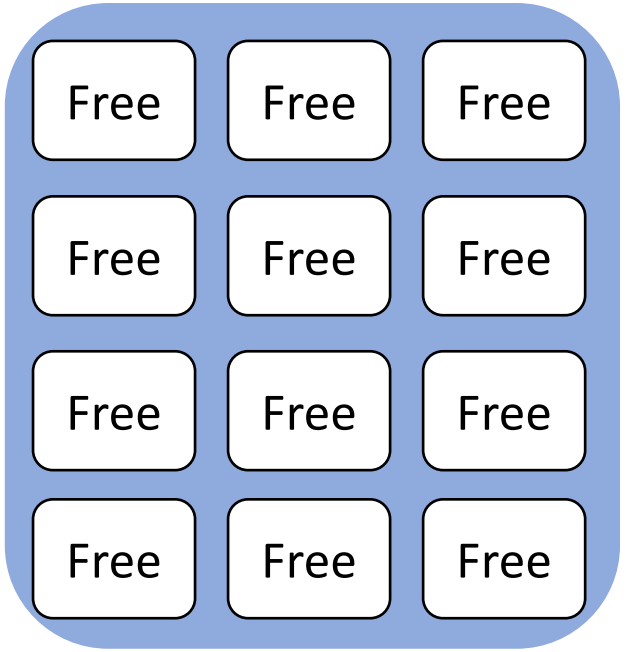
Writes on SSD

Update

A, B, C, D



Block 0

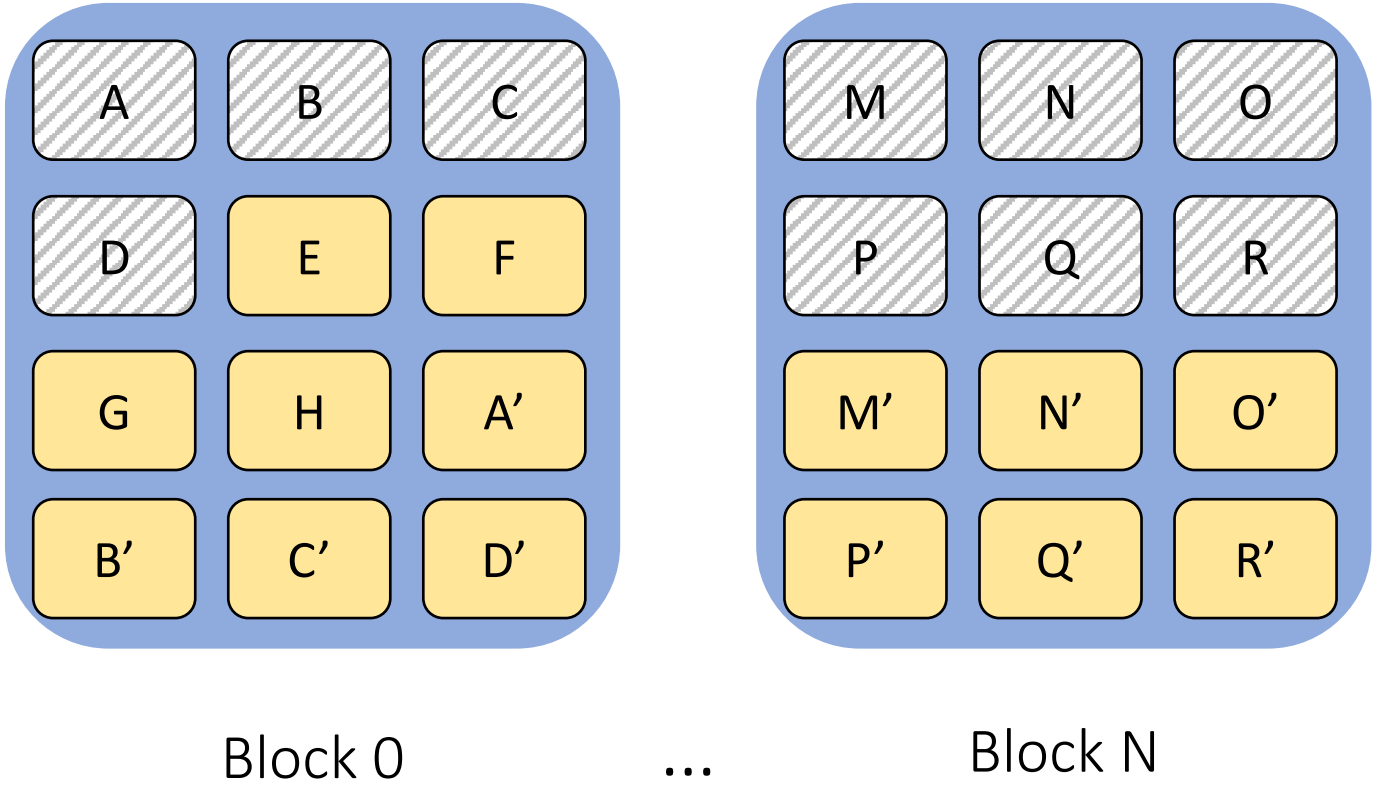


Block 1

Not all updates are costly!

Writes on SSD

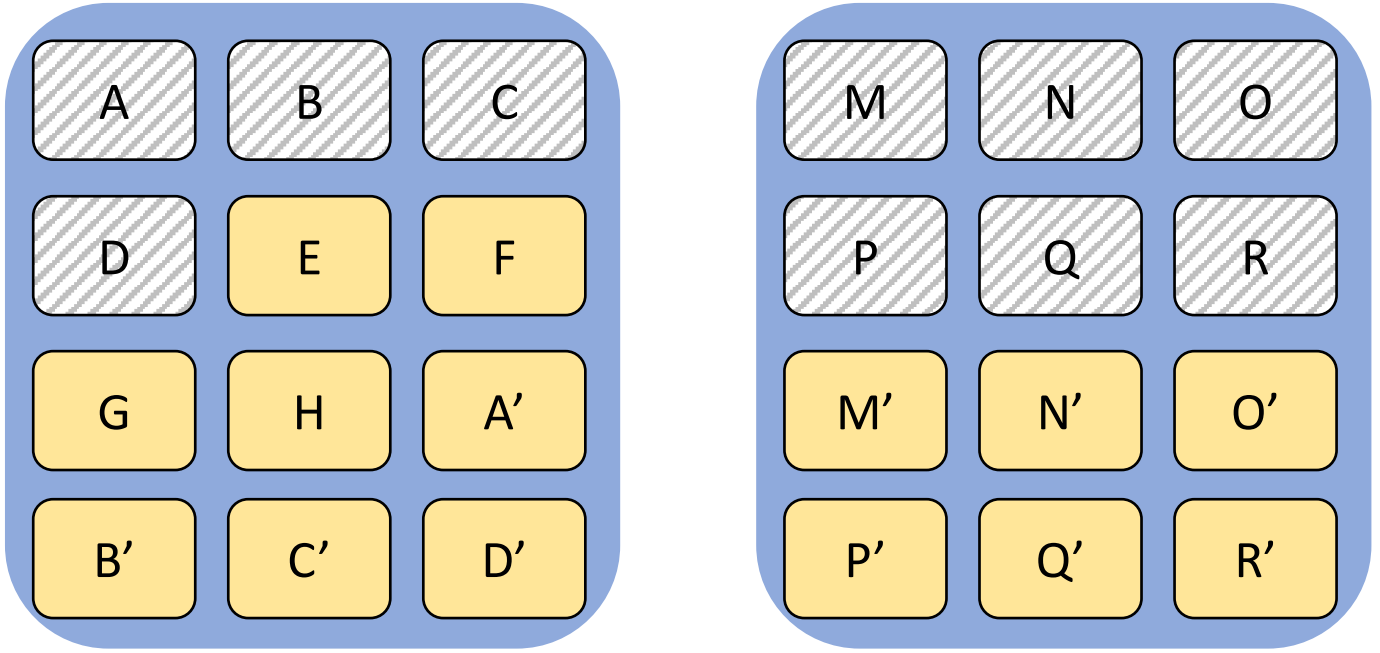
What if there is no space?



Writes on SSD

What if there is no space?

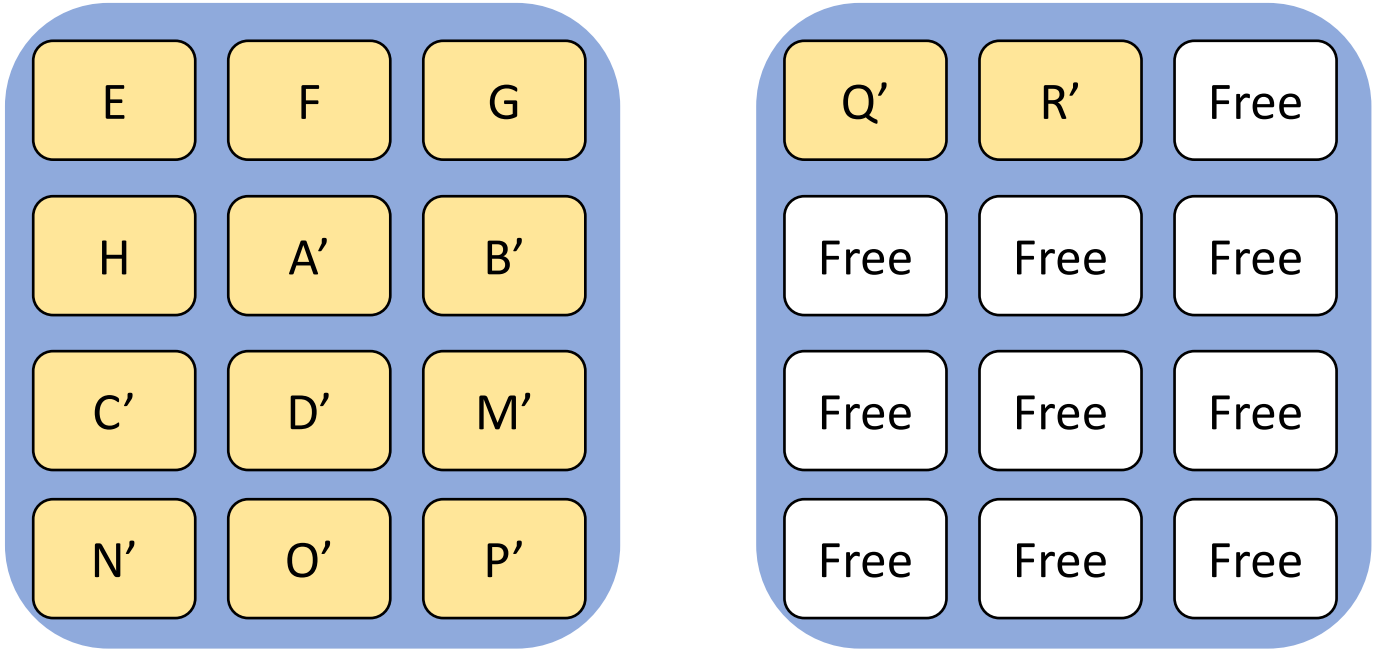
Garbage Collection!



Writes on SSD

What if there is no space?

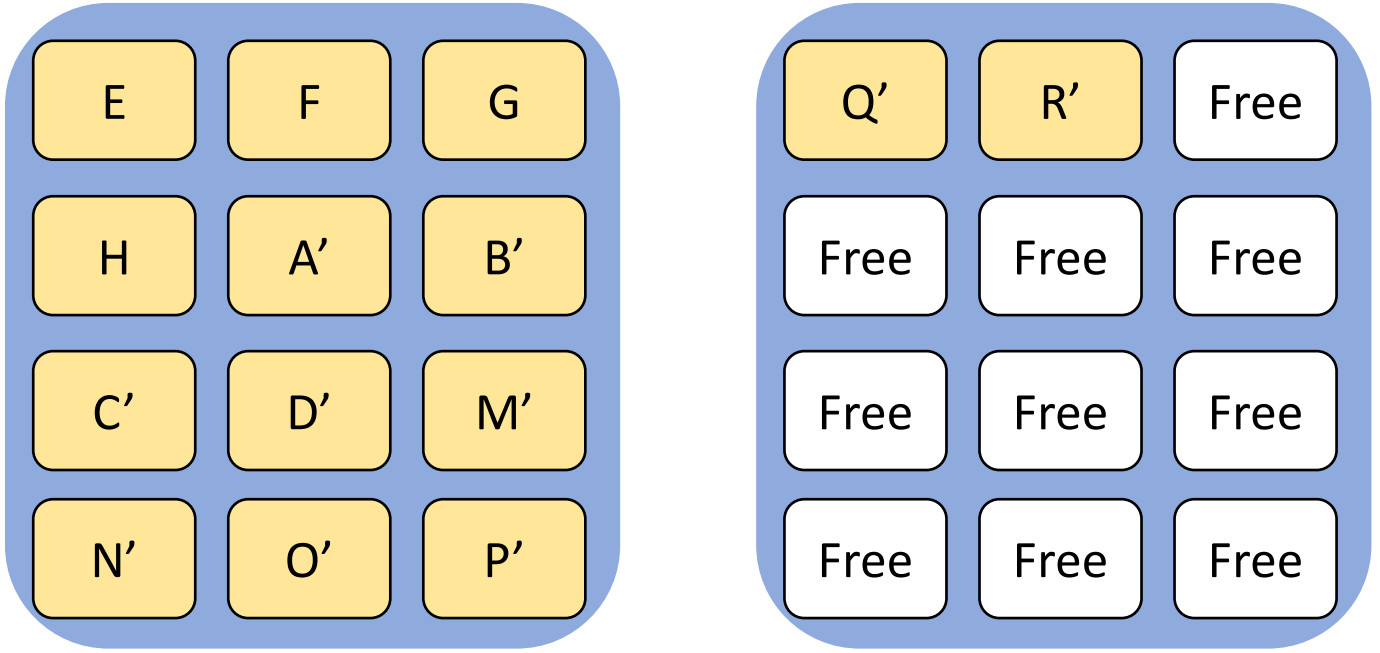
Garbage Collection!



Writes on SSD

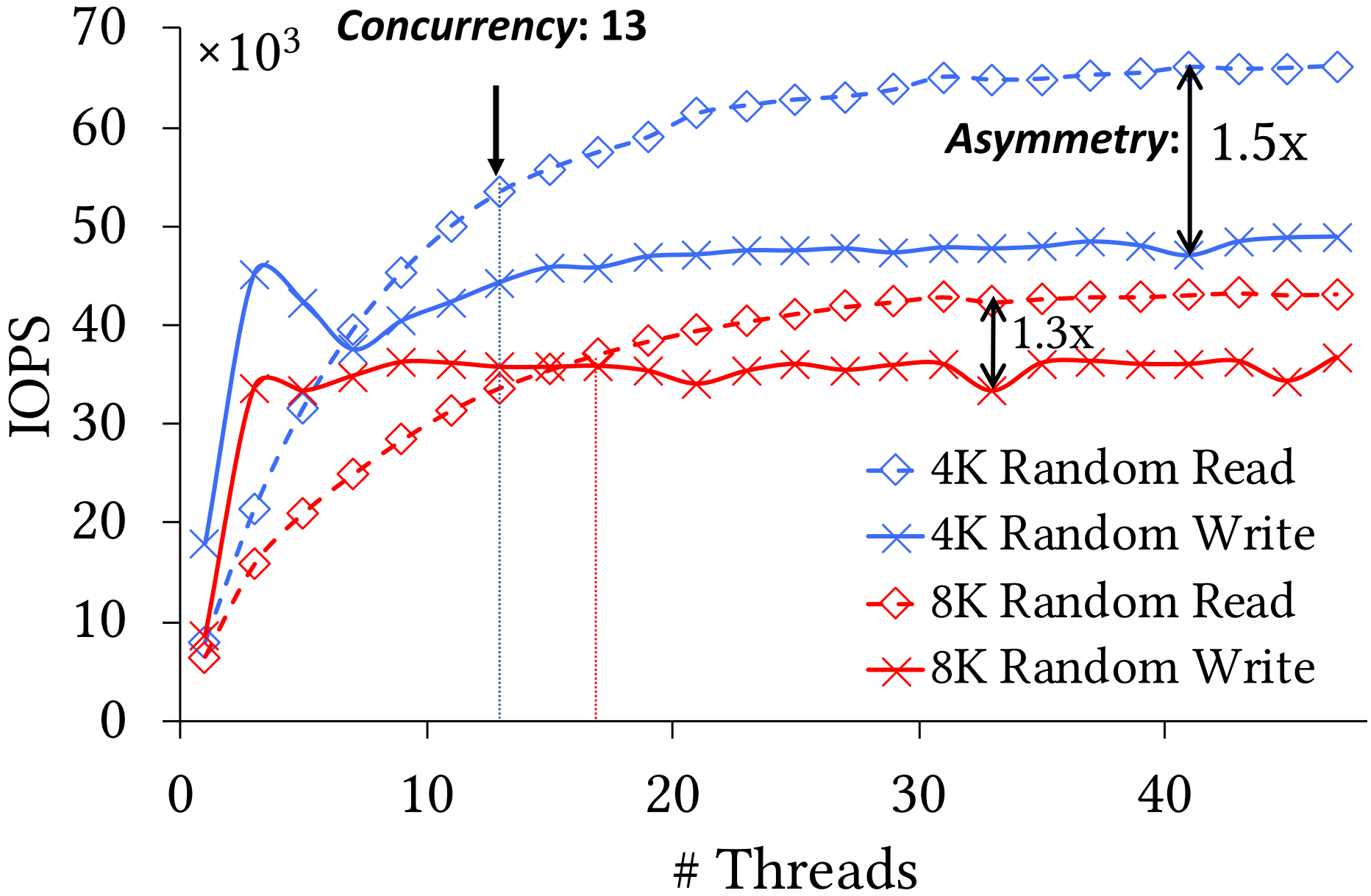
What if there is no space?

Garbage Collection!

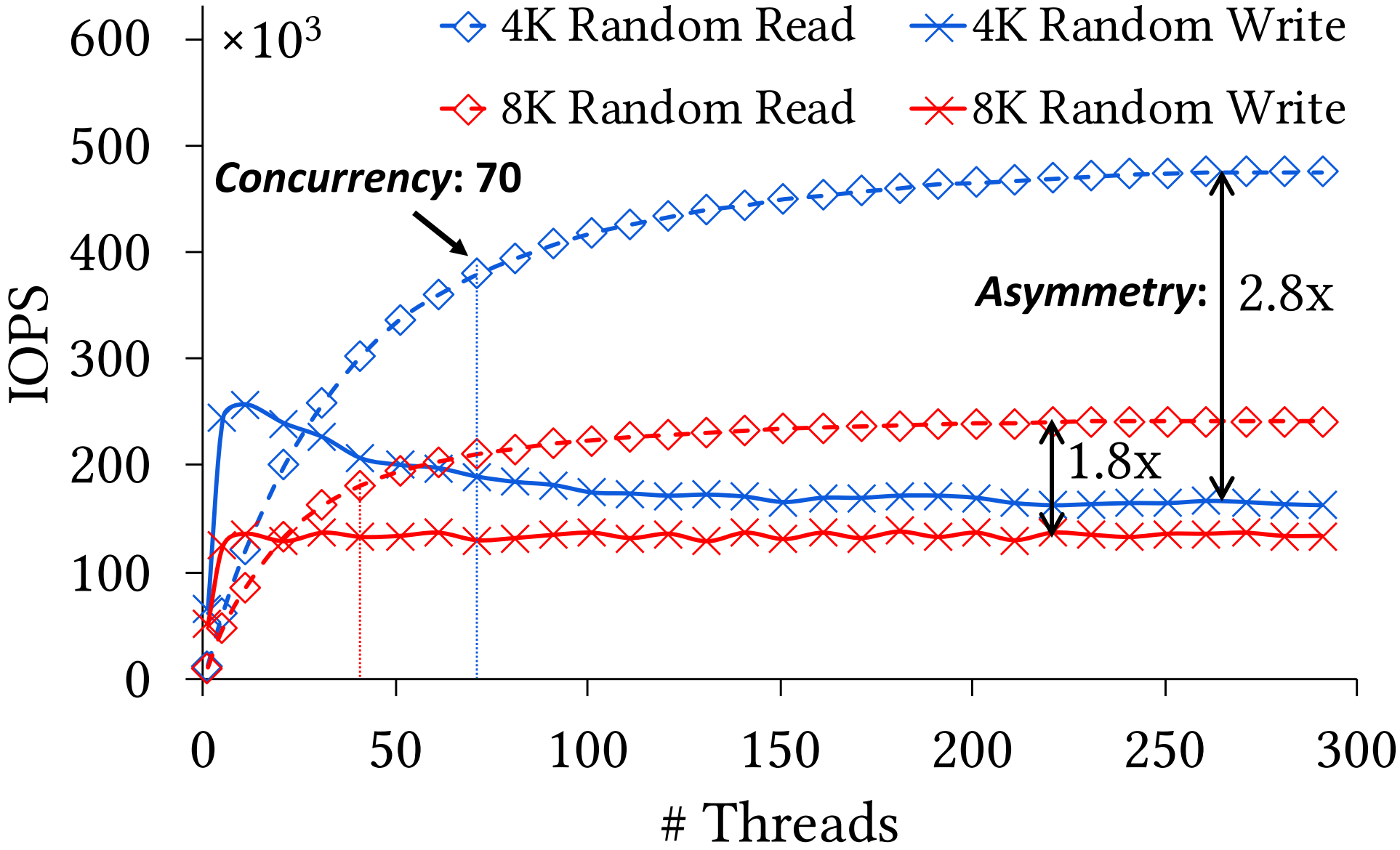


Higher average update cost (due to GC) → ***Read/Write asymmetry***

Measuring Asymmetry/Concurrency in off-the-shelf SSD



Measuring Asymmetry/Concurrency in NVMe SSD

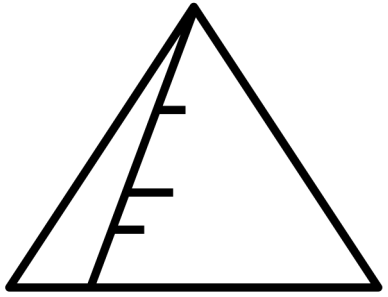


Make *asymmetry and concurrency* part of *algorithm design*

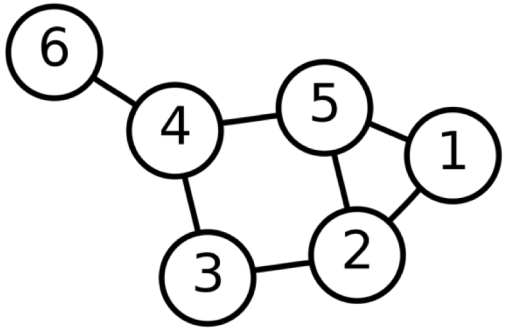
… not simply an engineering optimization

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

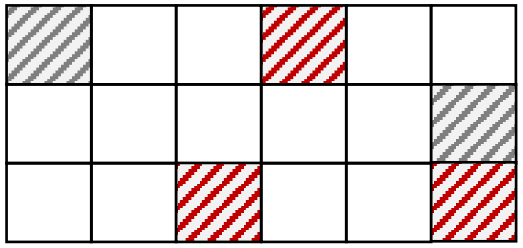
index structures



graph traversal algorithms



bufferpool management



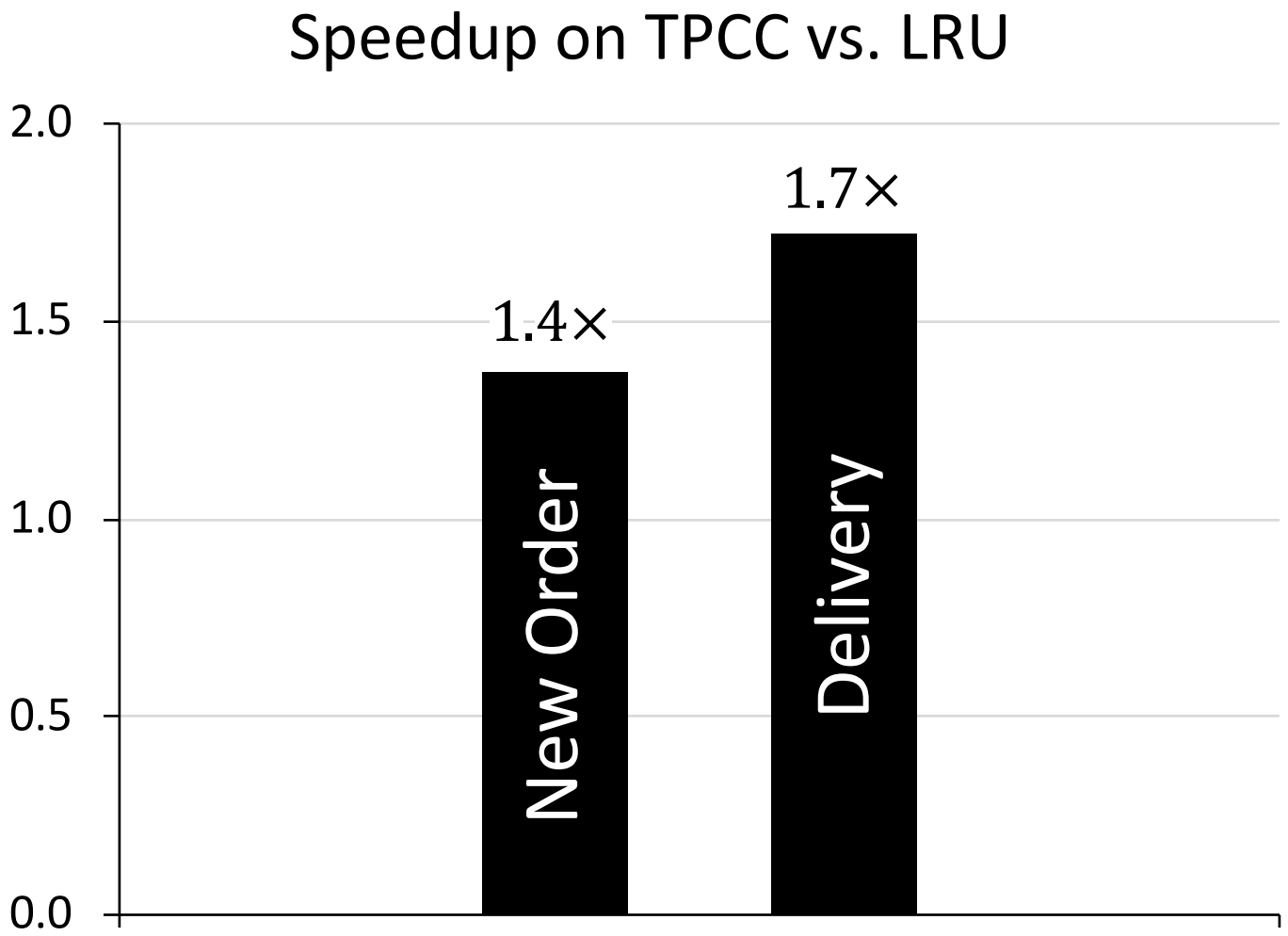
Asymmetry & Concurrency Aware Bufferpool Strategy

exploit **device parallelism**

concurrent write-back without evicting

bridge read/write **asymmetry**

SSD with asymmetry: 1.5x & concurrency: 9



Asymmetry & Concurrency Aware Bufferpool Strategy

exploit **device parallelism**

concurrent write-back without evicting

bridge read/write **asymmetry**

Thank you!

DiSC lab

disc.bu.edu/pio

SSD with asymmetry: 1.5x & concurrency: 9