PURESTORAGE®

PURESTORAGE®

How to Build Exascale Storage with QLC

Riley Thomasson

Member of Technical Staff
Pure Storage

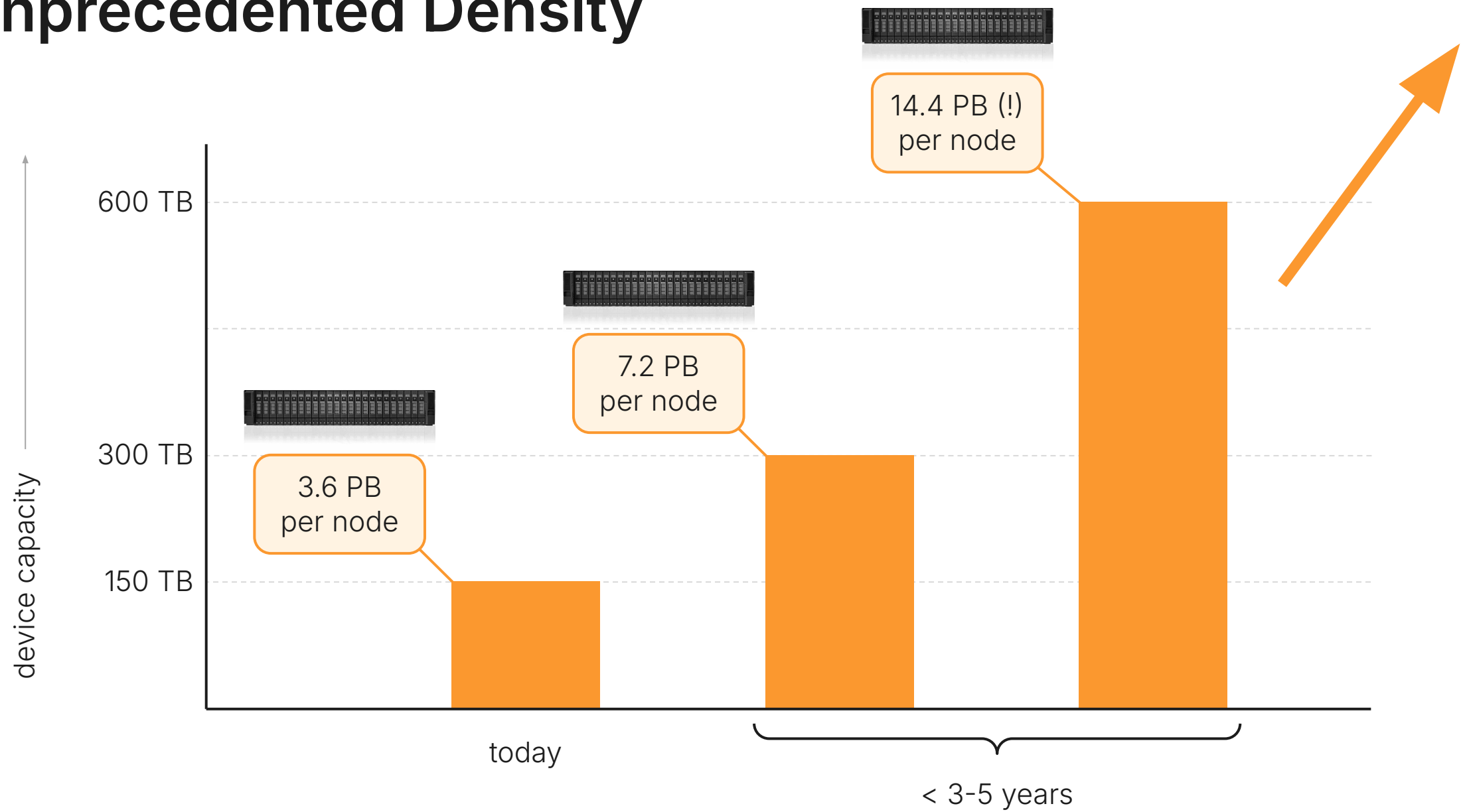
@purestorage

Sept 16, 2025

Agenda

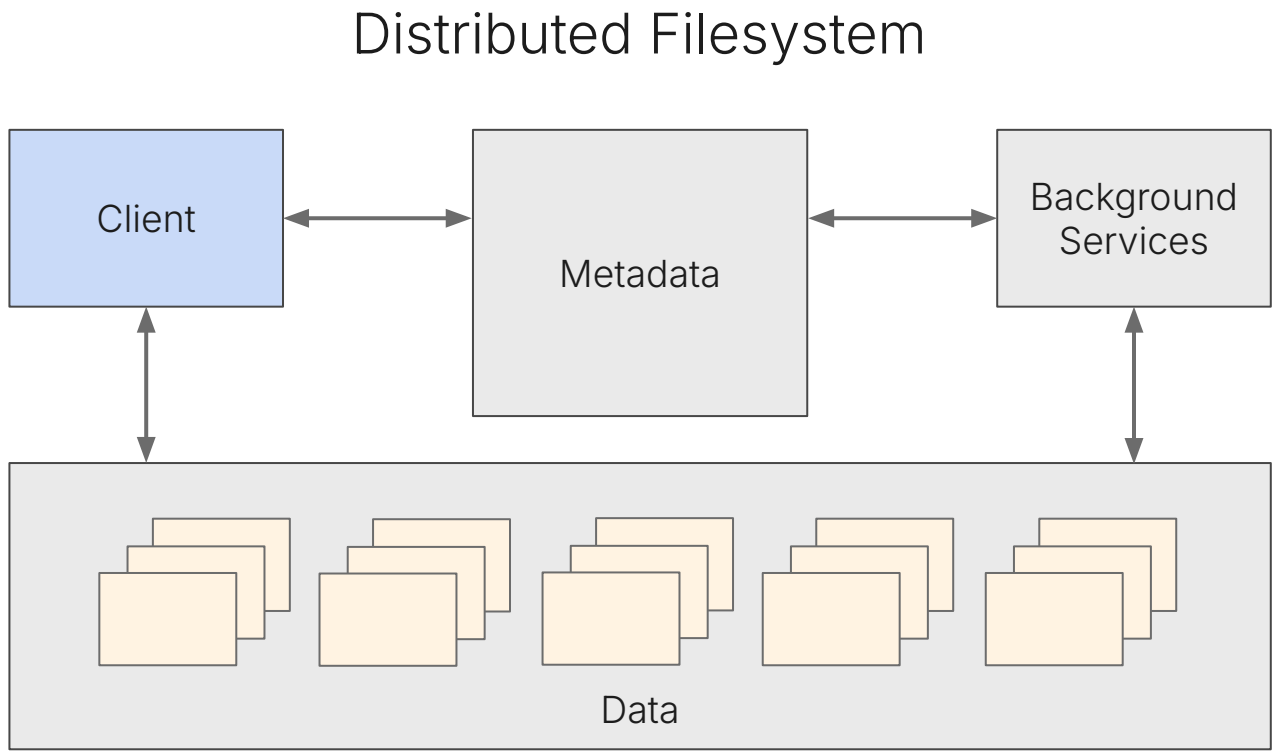
Context	Exascale Storage
Problem	QLC Challenges
Solution	Architectural Perspective

Unprecedented Density

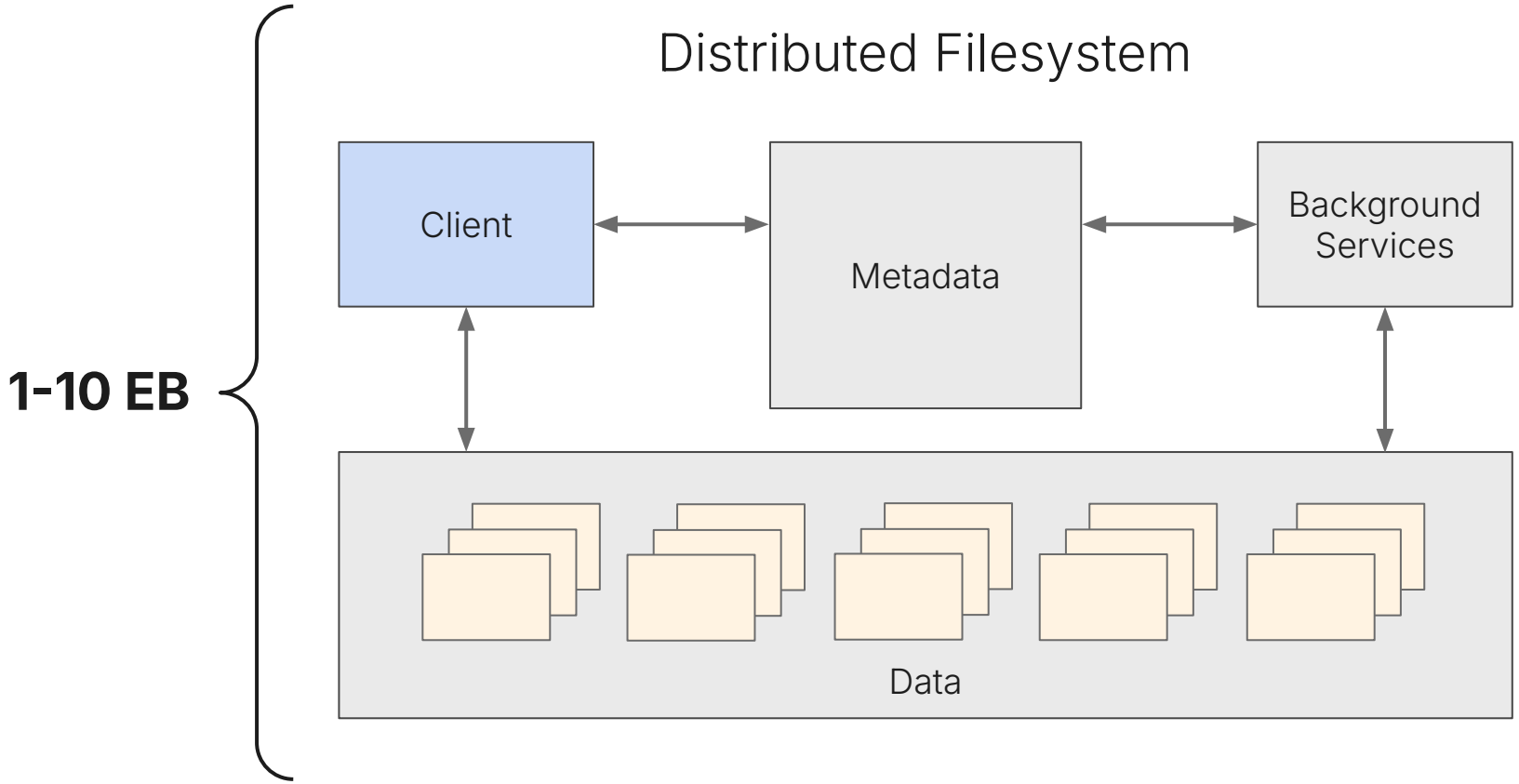


Exascale Storage

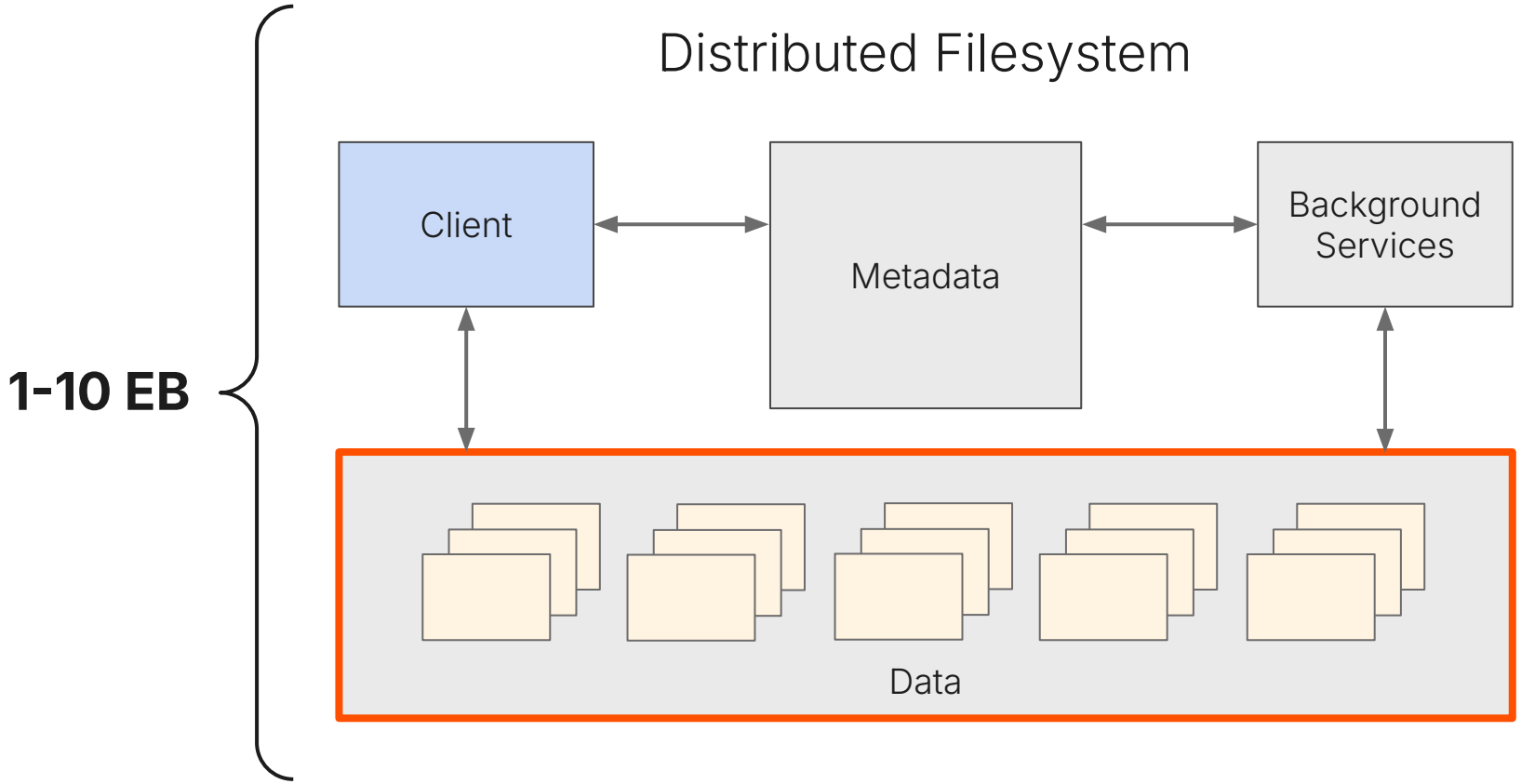
Use Cases
- GenAI
- Warm/Cold Object
- Archive
- Database



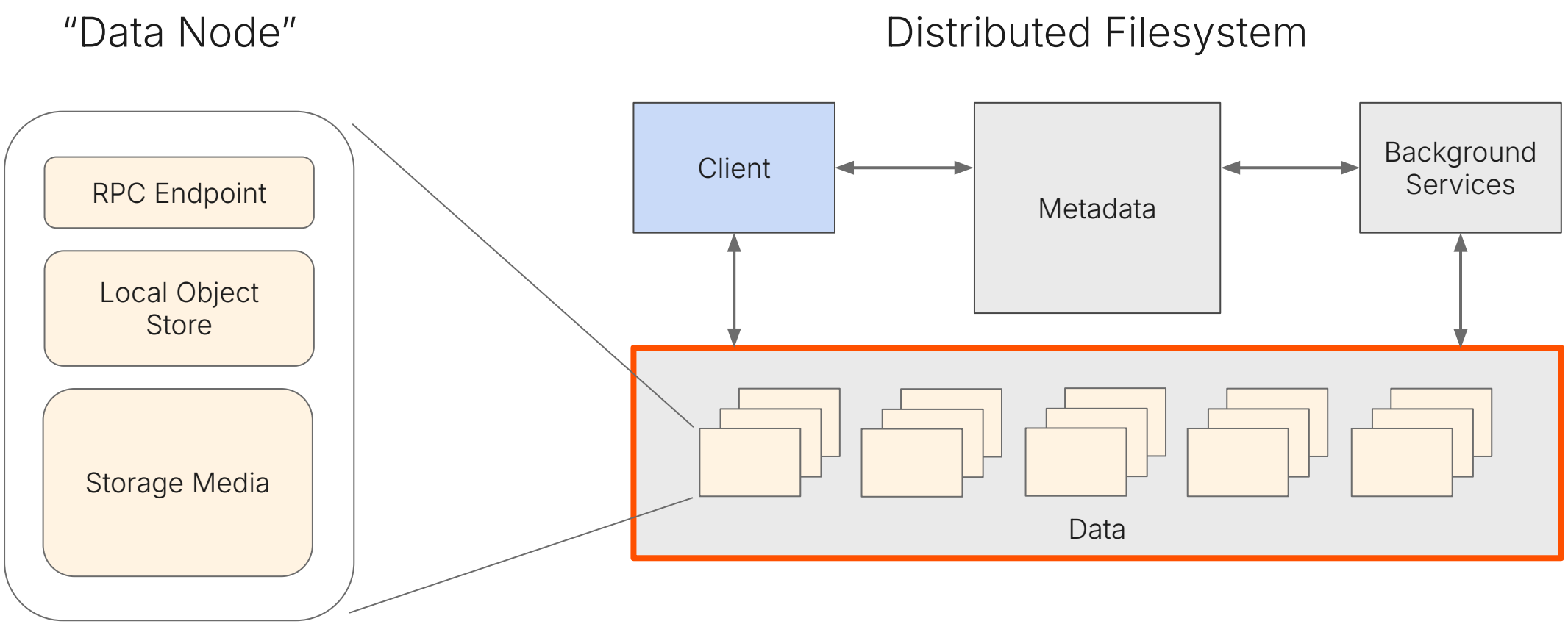
Exascale Storage



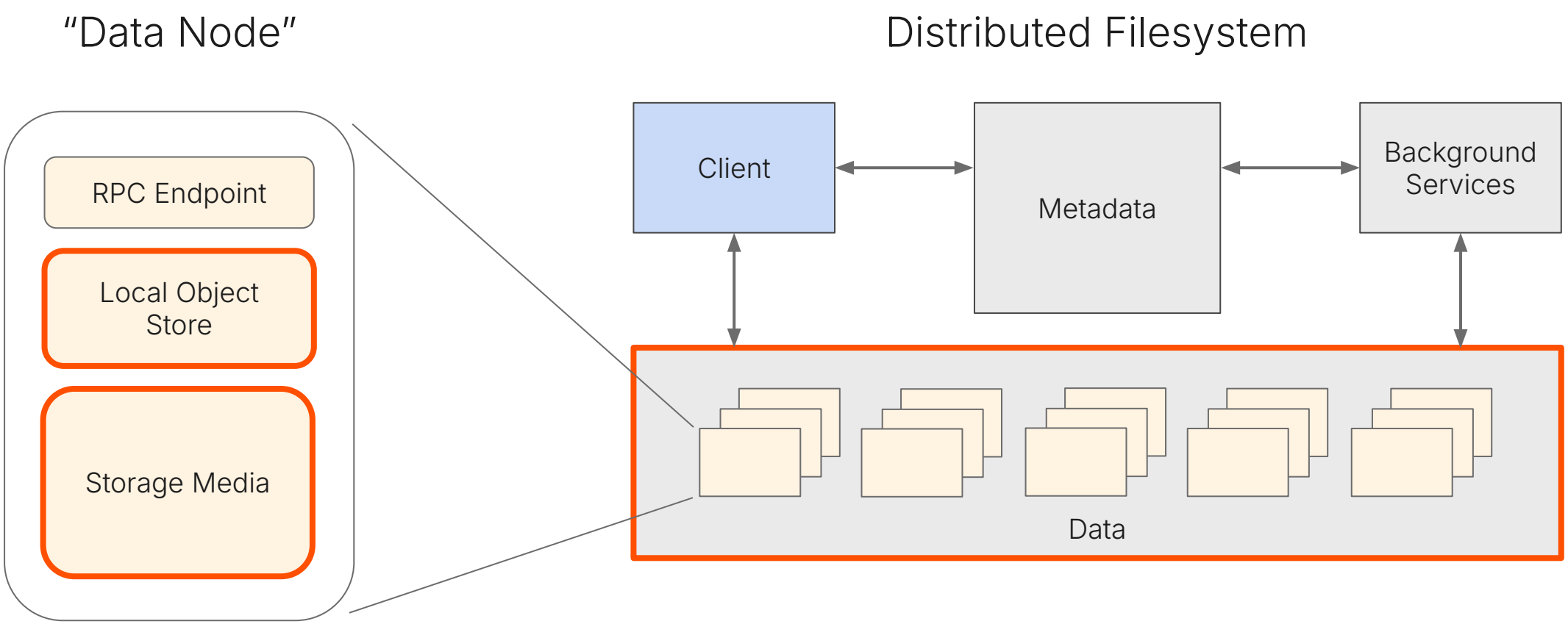
Exascale Storage



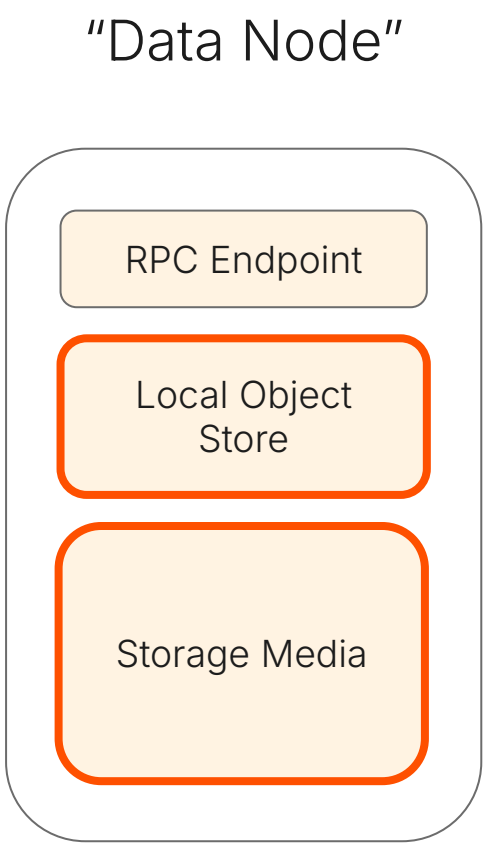
Exascale Storage



Exascale Storage



Exascale Storage

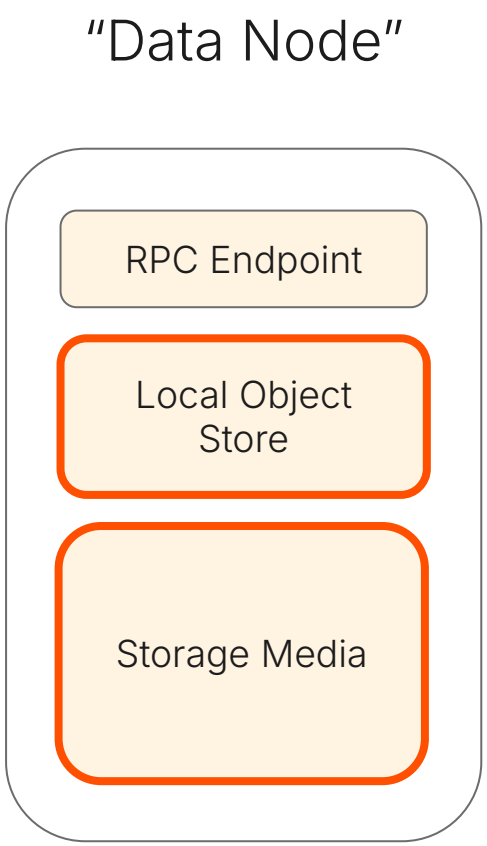


“Local Object Store”
- flat namespace
- append-only objects
- protected by cluster-level ECC
- no standard, many proprietary implementations

Not: **general purpose storage**
- filesystems (xfs, ext4, ntfs)
- fully-featured, hierarchical, overwrite-in-place
- standardized (POSIX)

Not: **cloud “object storage”**
- HA
- external, standardized API
- served by a large distributed cluster

Exascale Storage



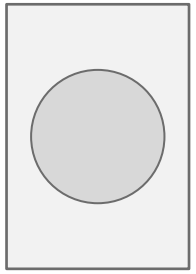
At exascale, efficiency is paramount!

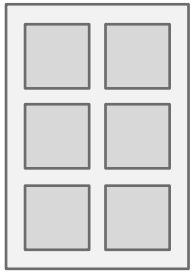
- power/TB
- performance/TB
- cost (media, CPU, memory, racks…)
- end-to-end perspective

Opportunities for efficiency:

1. leverage QLC media
2. customize for the “data node” workload

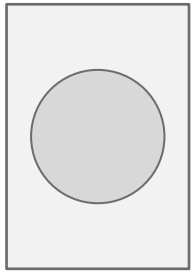
QLC

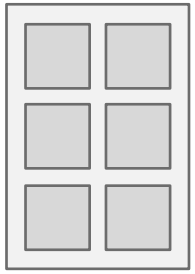


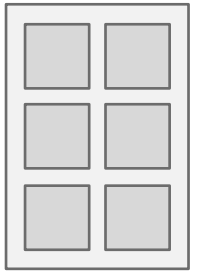


		HAMR HDD 30 TB	TLC 30 TB
per TB	$	1x	6x
	power	1x	2.8x
	read throughput	1x	280x
	write throughput	1x	90x
	IOPS	1x	30,000x

QLC







	HAMR HDD 30 TB	QLC 150 TB	TLC 30 TB
per TB $	1x	3x	6x
per TB power	1x	0.5x	2.8x
per TB read throughput	1x	60x	280x
per TB write throughput	1x	7x	90x
per TB IOPS	1x	2,000x	30,000x

Agenda

Context Exascale Storage

Problem QLC Challenges

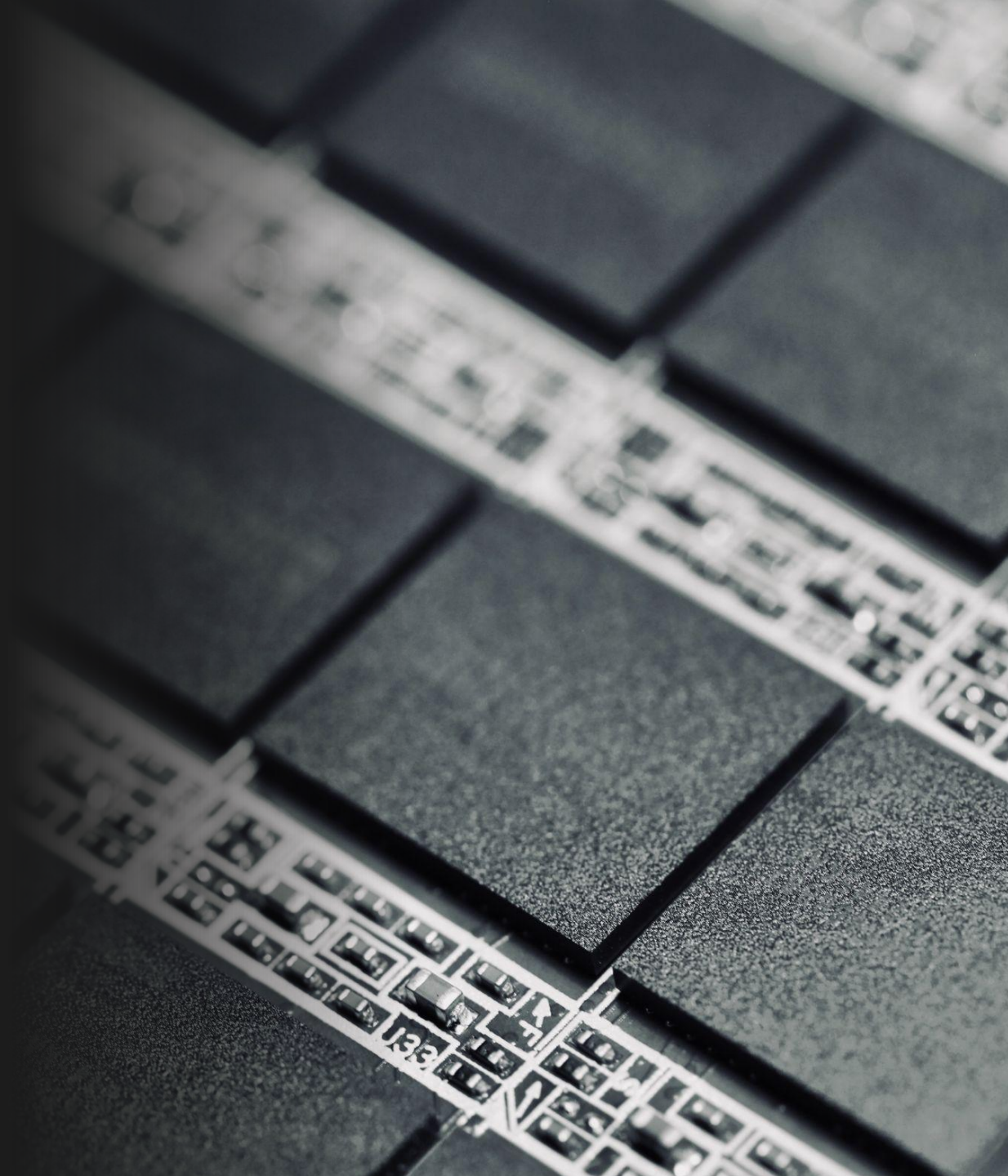
Solution Architectural Perspective

QLC Challenges

Scale

Endurance

Reliability

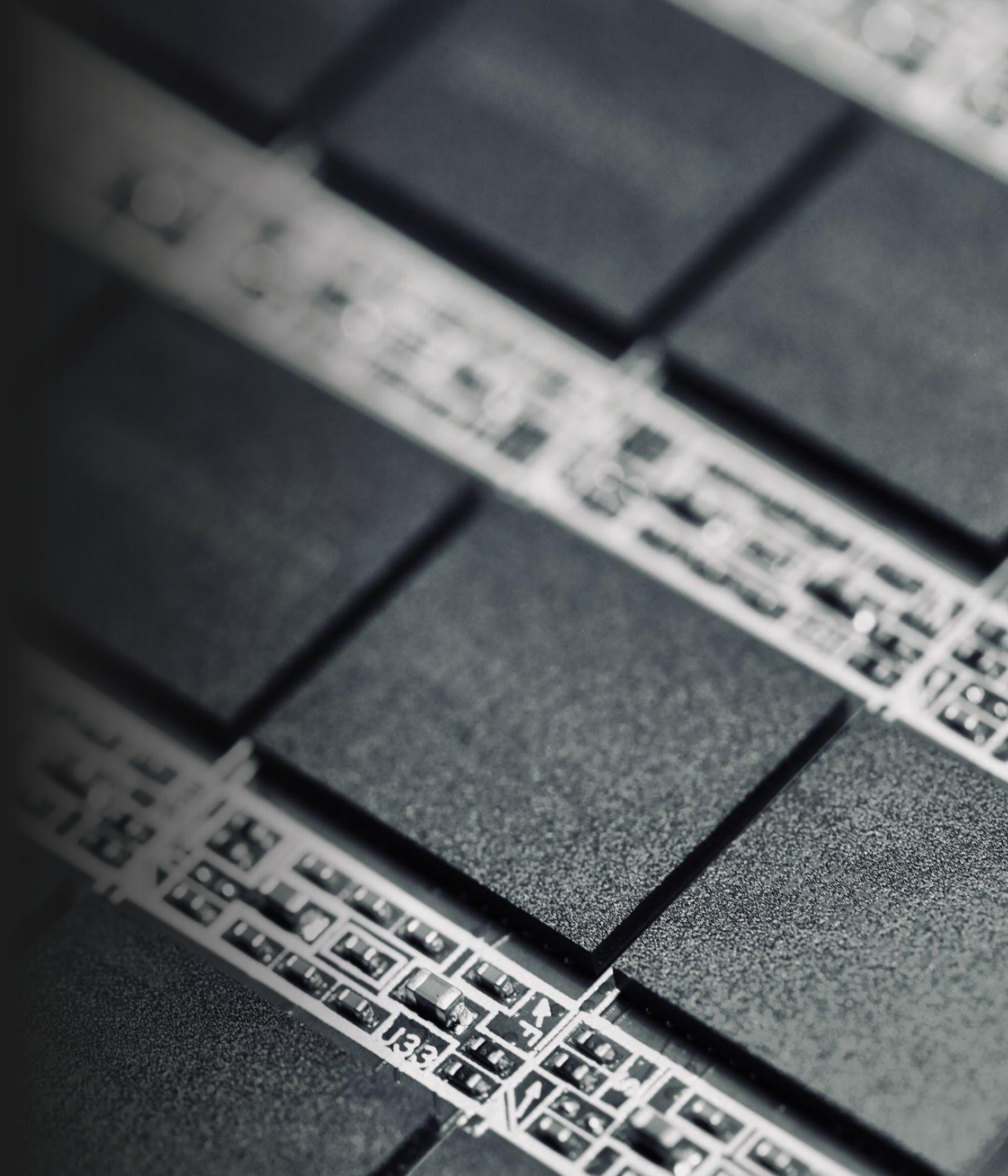


QLC Challenges

Scale

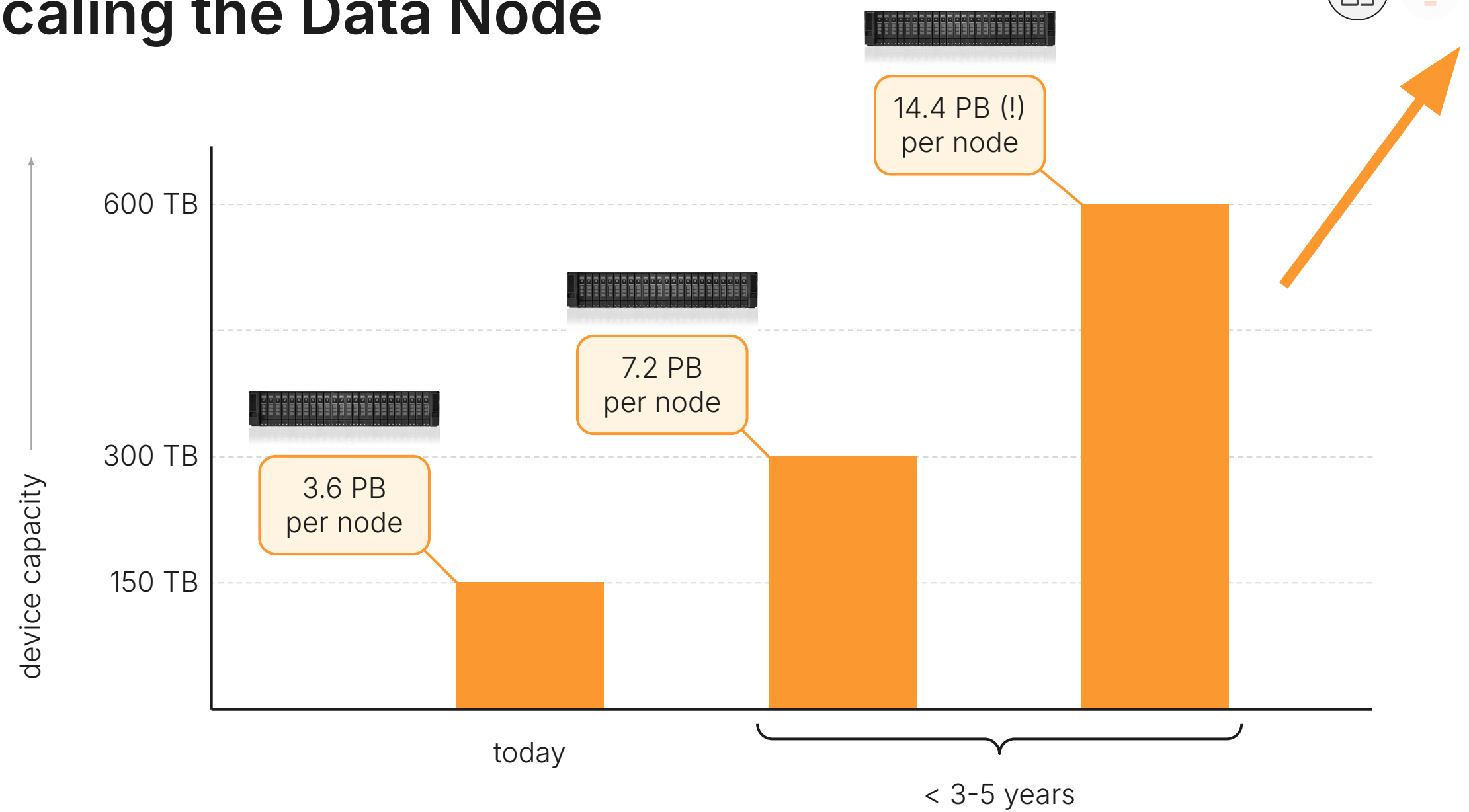
Endurance

Reliability



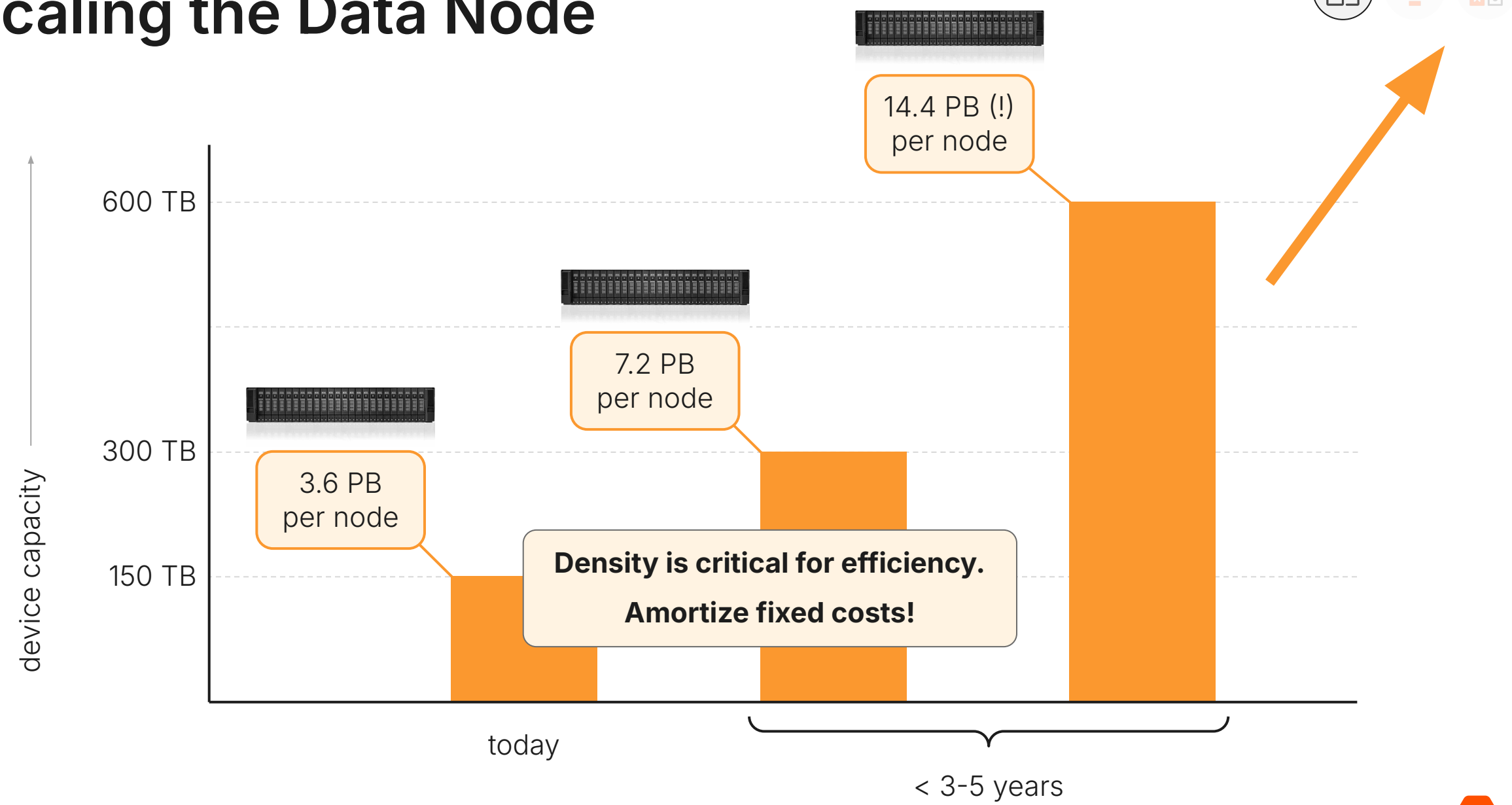
Scaling the Data Node

Scale

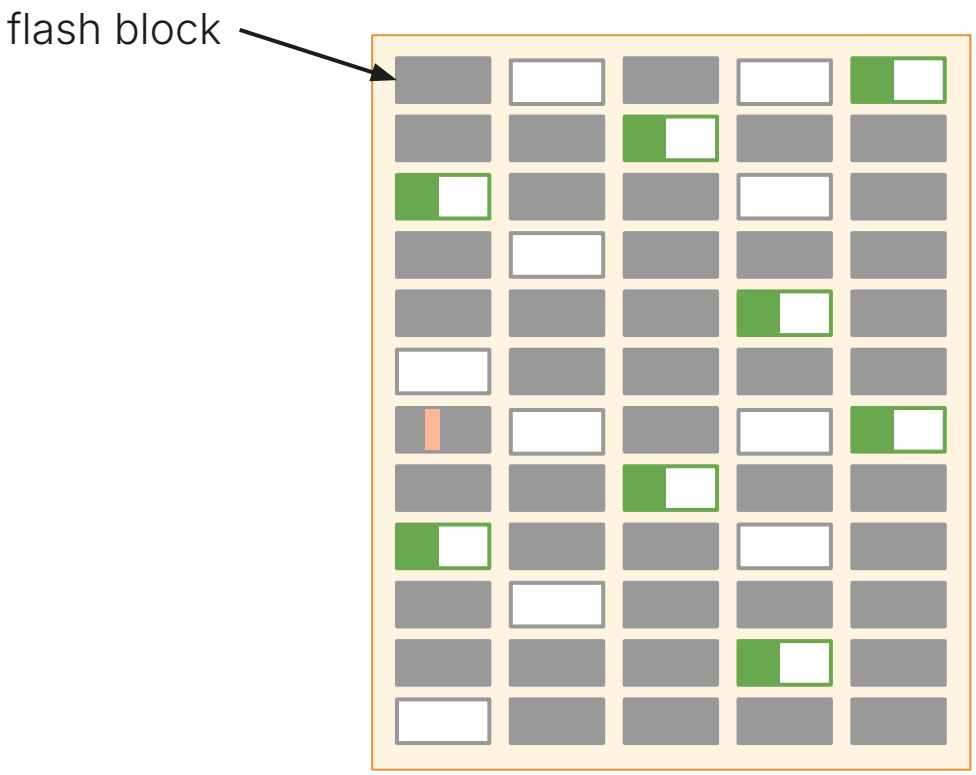


Scaling the Data Node

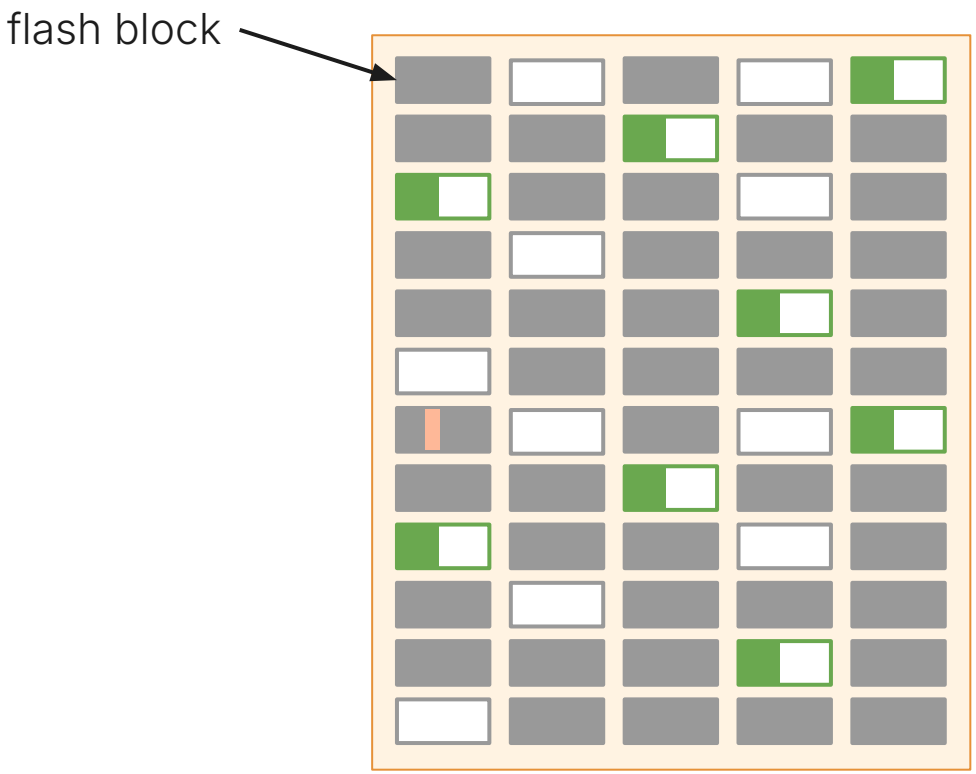
Scale



Review: FTL



Review: FTL

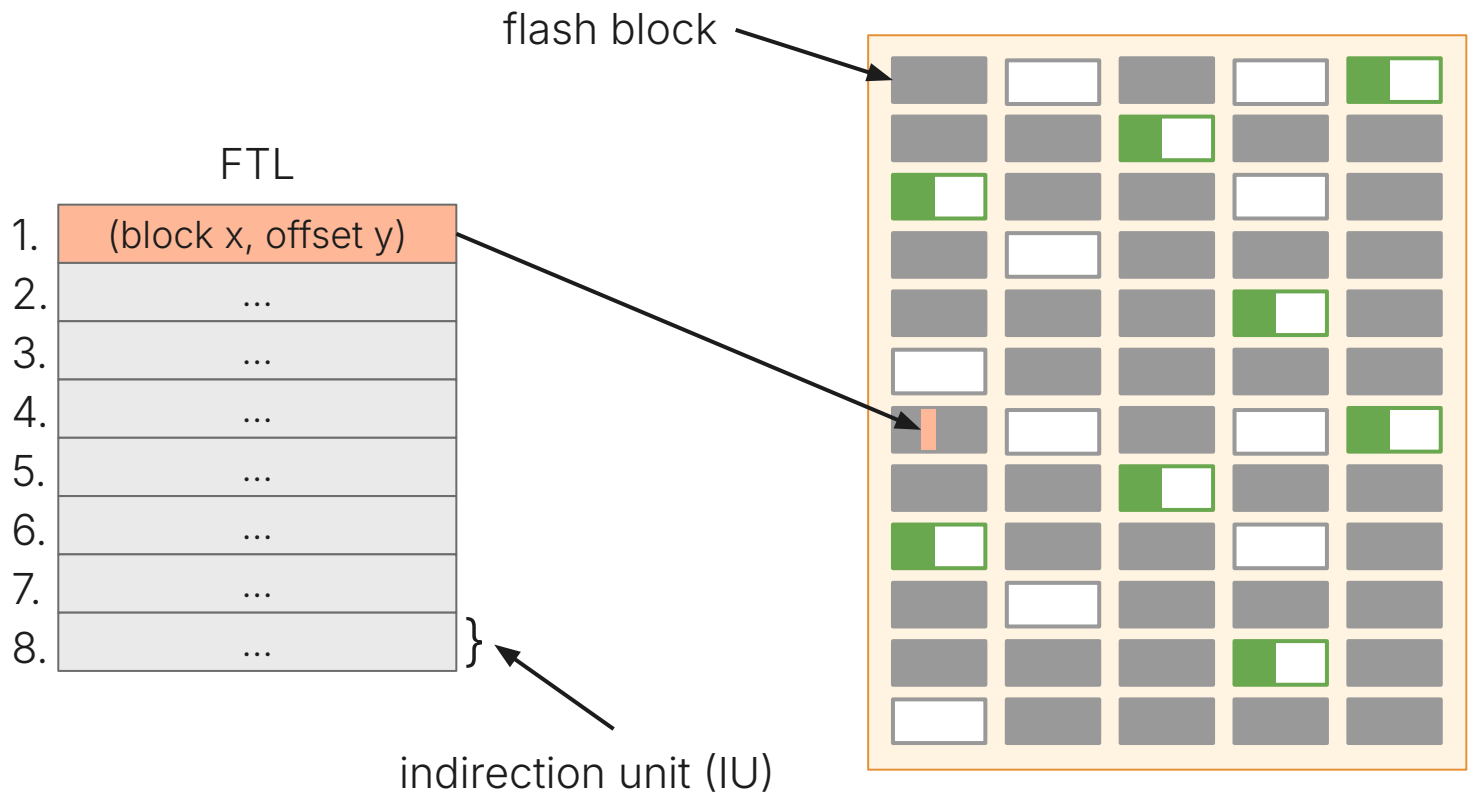


Open: accepting new writes via append

Full: no new writes, reads are ok

Erased: empty, ready to write

Review: FTL



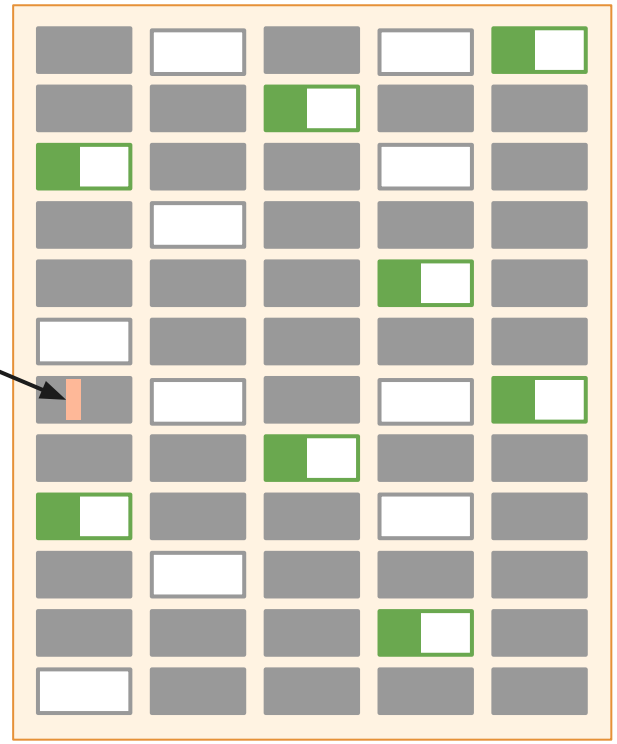
Open: accepting new writes via append

Full: no new writes, reads are ok

Erased: empty, ready to write

Review: FTL


```
write: {
  lba: 5,
  data: 0xfa...
}
```

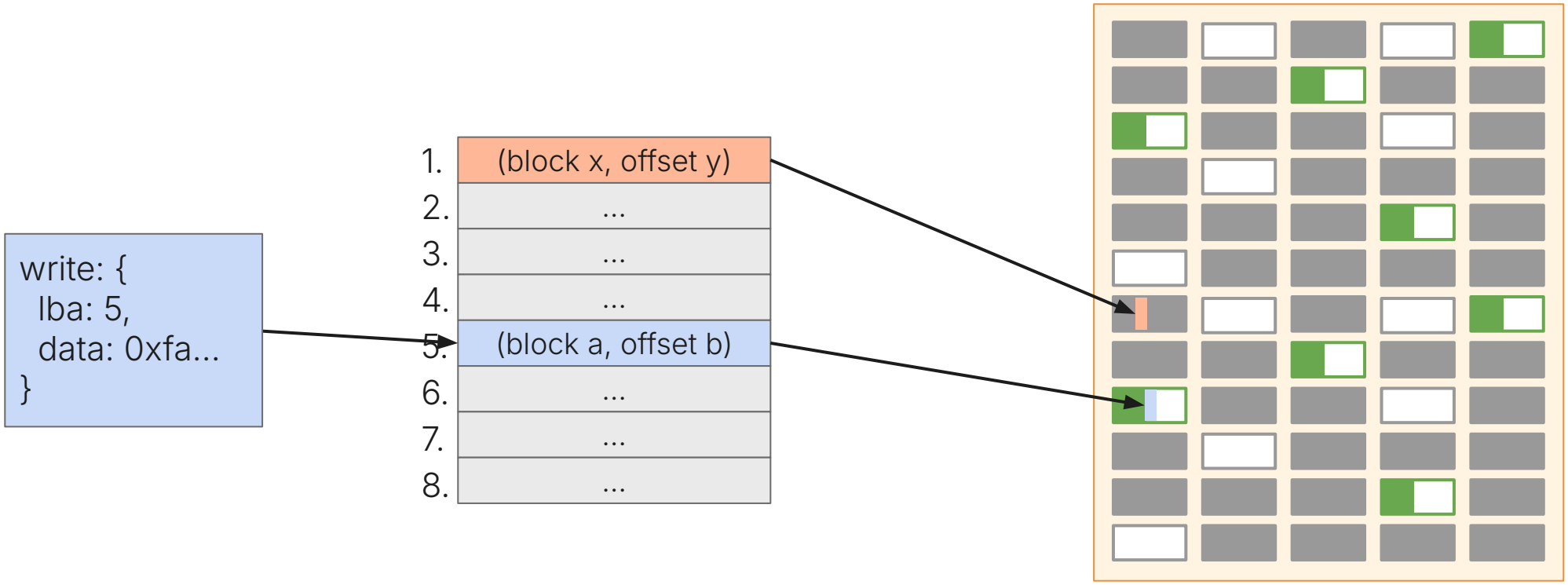


Open: accepting new writes via append

Full: no new writes, reads are ok

Erased: empty, ready to write

Review: FTL



Open: accepting new writes via append

Full: no new writes, reads are ok

Erased: empty, ready to write

Scaling the FTL

4B per IU mapping: addresses 2^{32} IUs, aligns nicely in memory

IU size	**flash:memory ratio**	**total memory** *per 14 PB node*	**addressable capacity** *per FTL partition*	**# FTL partitions**
4 KiB	1024:1	~14 TB	16 TiB	~800

Scaling the FTL

4B per IU mapping: addresses 2^{32} IUs, aligns nicely in memory

IU size	**flash:memory ratio**	**total memory** *per 14 PB node*	**addressable capacity** *per FTL partition*	**# FTL partitions**
4 KiB	1024:1	~14 TB	16 TiB	~800

Expensive!
- Reduce mapping bits? More partitions!
- Page FTL into flash? Tail latency!

Inflexible!
- Increase mapping bits? More memory!

Scaling the FTL

4B per IU mapping: addresses 2^{32} IUs, aligns nicely in memory

IU size	flash:memory ratio	total memory *per 14 PB node*	addressable capacity *per FTL partition*	# FTL partitions
4 KiB	1024:1	~14 TB	16 TiB	~800

Expensive!
- Reduce mapping bits? More partitions!
- Page FTL into flash? Tail latency!

Inflexible!
- Increase mapping bits? More memory!

Increase IU size!

Scaling the FTL

	IU size	flash:memory ratio	total memory *per 14 PB node*	addressable capacity *per FTL partition*	# FTL partitions
status quo	4 KiB	1024:1	~14 TB	16 TiB	~800

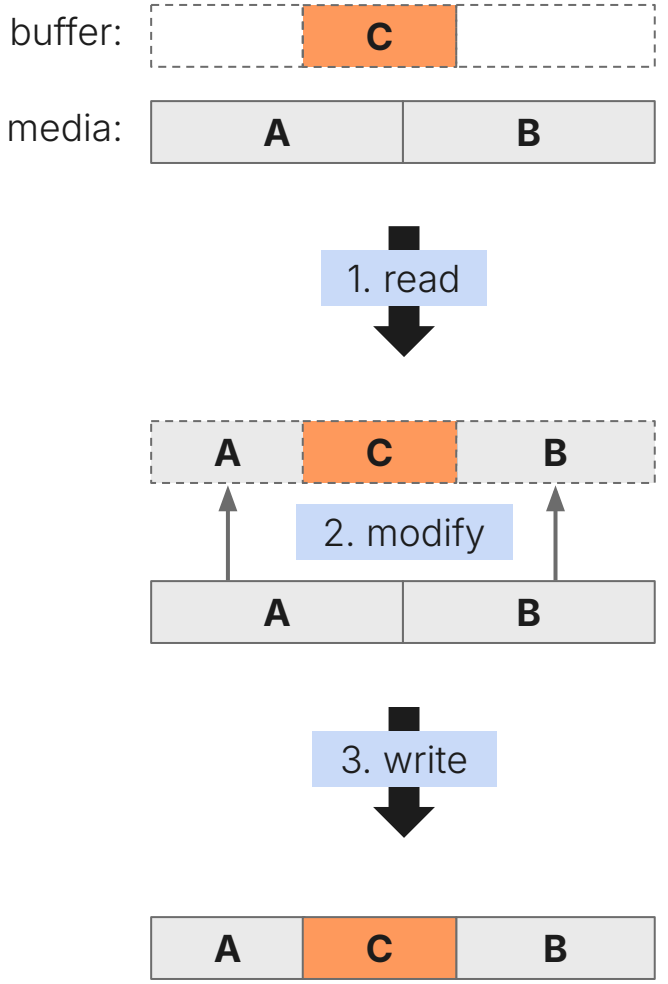
Scaling the FTL

	IU size	flash:memory ratio	total memory *per 14 PB node*	addressable capacity *per FTL partition*	# FTL partitions
status quo	4 KiB	1024:1	~14 TB	16 TiB	~800
new high-cap SSDs	16 KiB	4096:1	~3.4 TB	64 TiB	~200
	64 KiB	16384:1	~850 GB	256 TiB	~50

Scaling the FTL

	IU size	flash:memory ratio	total memory *per 14 PB node*	addressable capacity *per FTL partition*	# FTL partitions
status quo	4 KiB	1024:1	~14 TB	16 TiB	~800
new high-cap SSDs	16 KiB	4096:1	~3.4 TB	64 TiB	~200
	64 KiB	16384:1	~850 GB	256 TiB	~50
the future of ultra-dense QLC	512 KiB	131072:1	~100 GB	2 PiB	~6
	...				

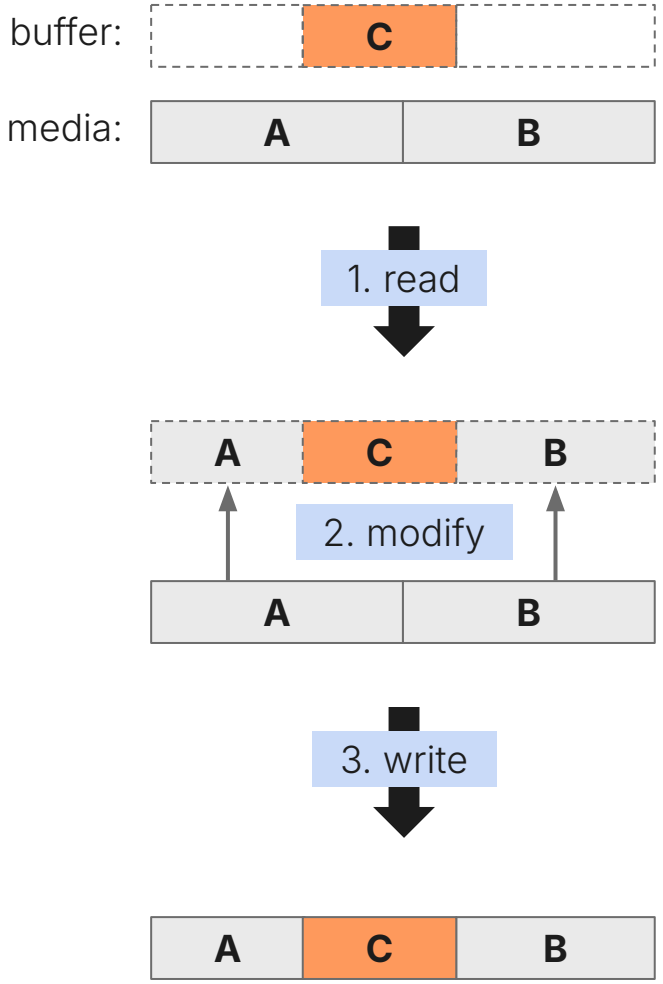
Pitfall: Read-Modify-Write



Pitfall: Read-Modify-Write

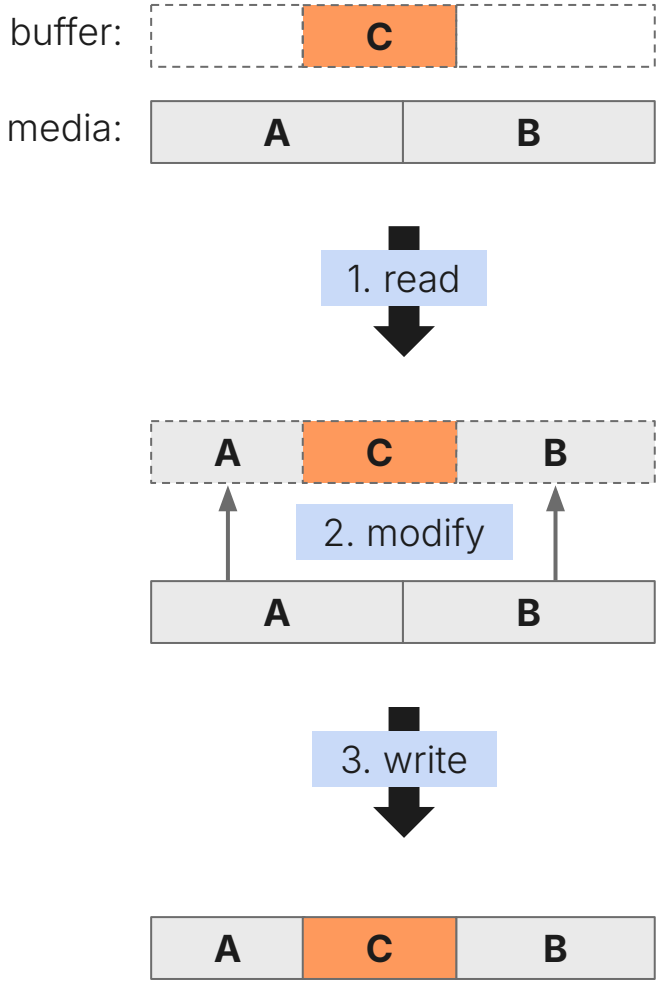
Scale

- extra read and write load → **reduced perf, endurance**
- extra media round-trip → **increased latency**
- write size **and alignment** are important
- overhead **scales with IU size**
- this is a real problem for **general-purpose** systems



Pitfall: Read-Modify-Write

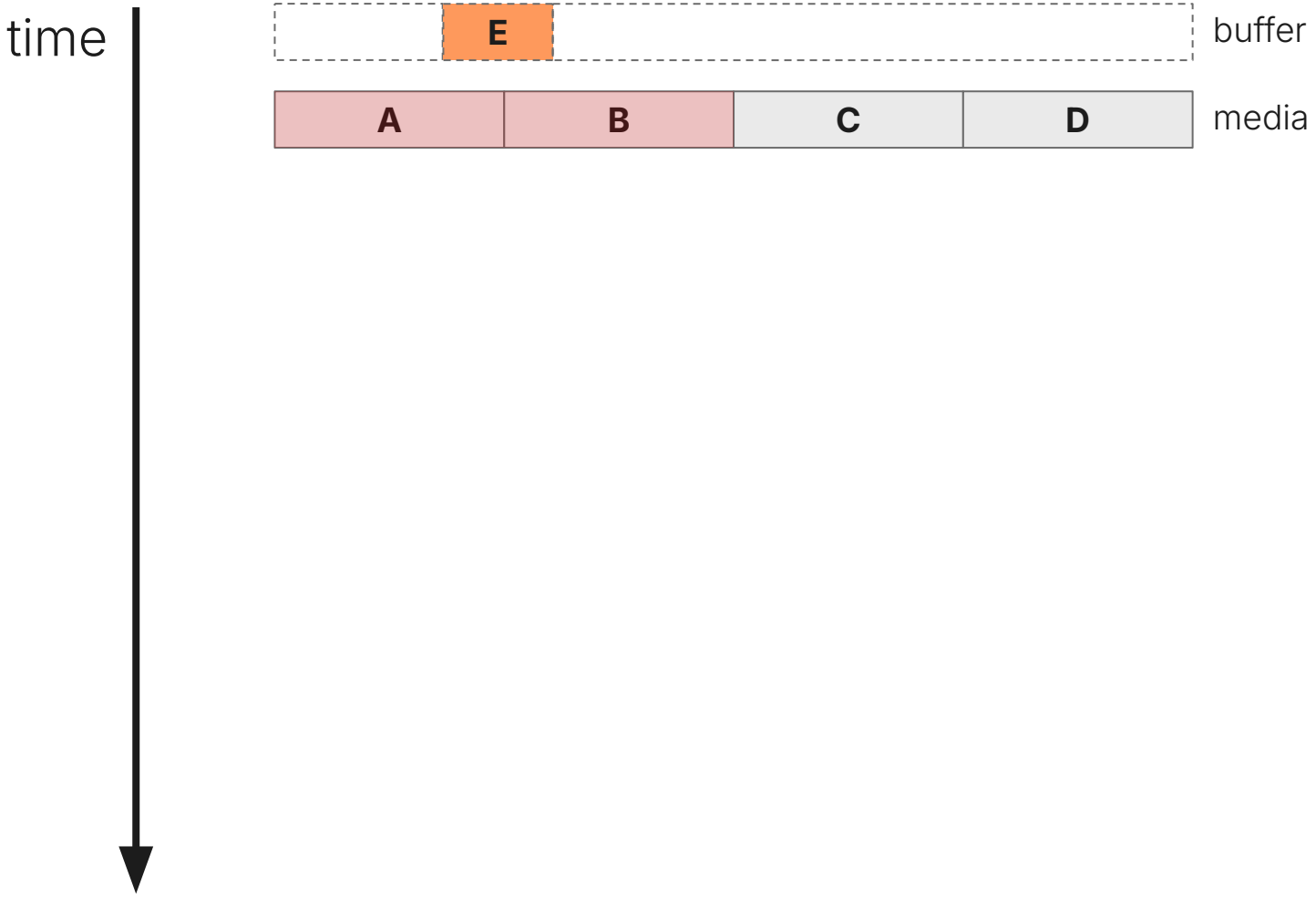
- extra read and write load → **reduced perf, endurance**
- extra media round-trip → **increased latency**
- write size **and alignment** are important
- overhead **scales with IU size**
- this is a real problem for **general-purpose** systems



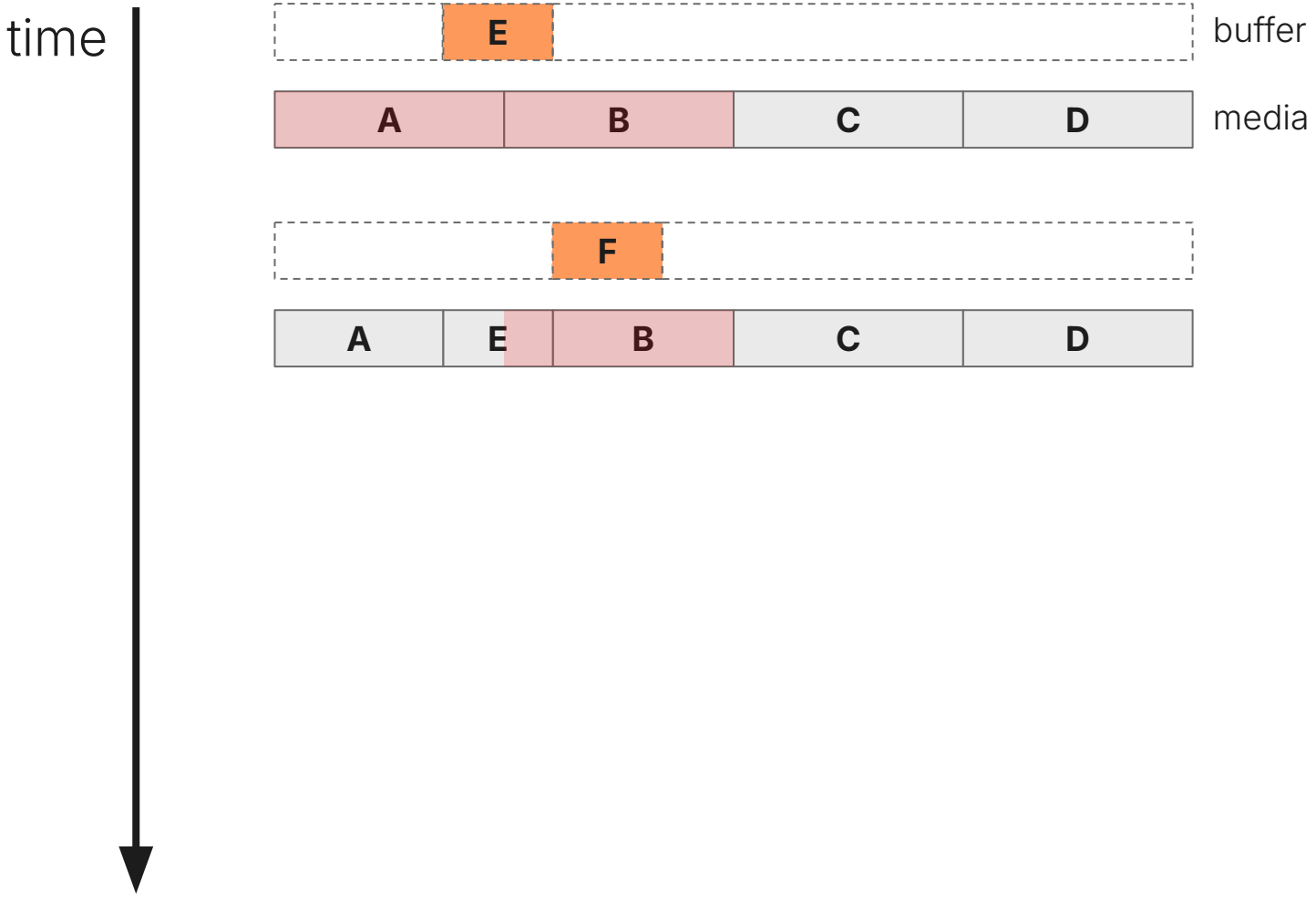
Key Insight: Take advantage of exascale workload patterns to deliver a large IU with minimal RMW.

Write Coalescing

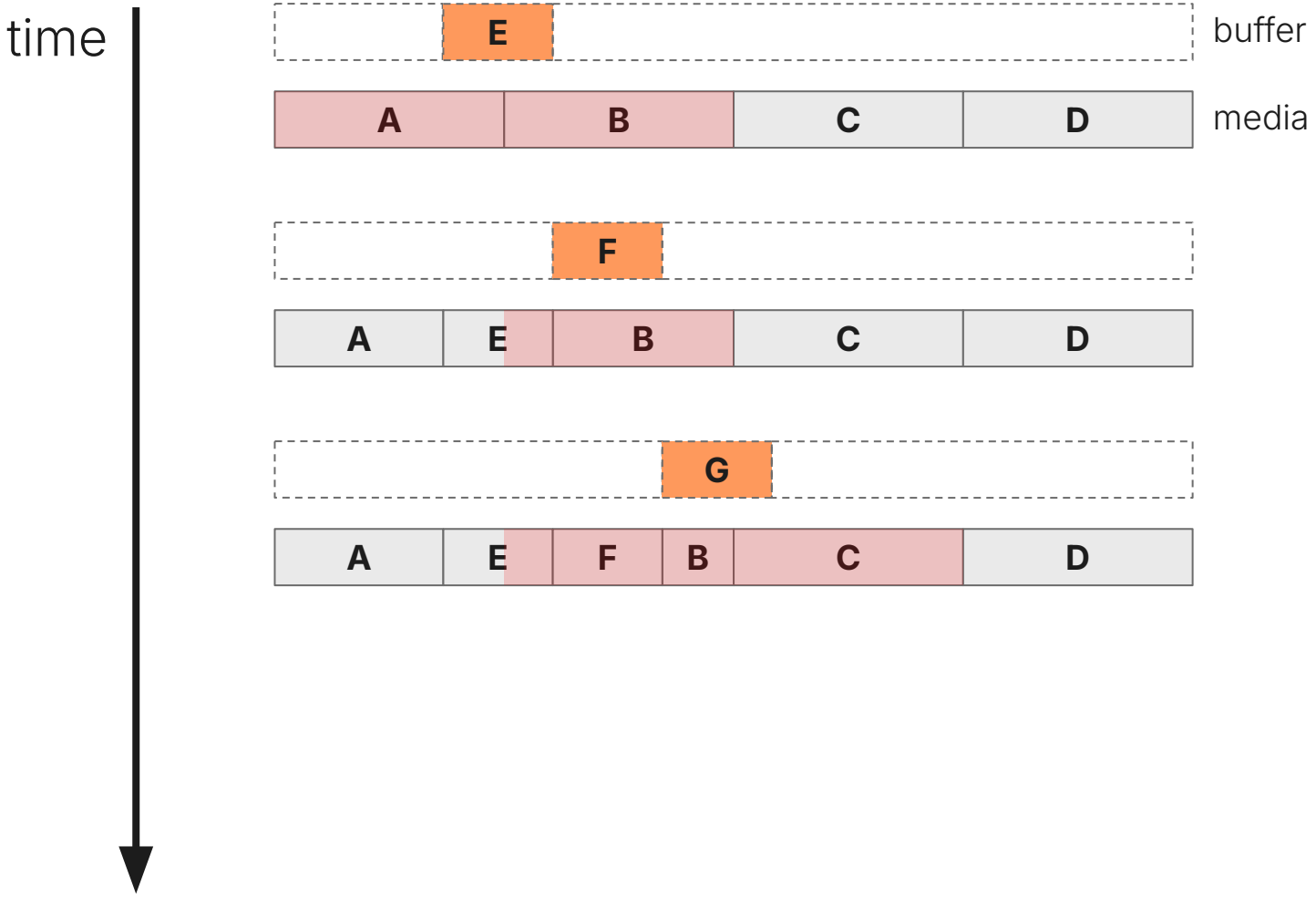
Scale



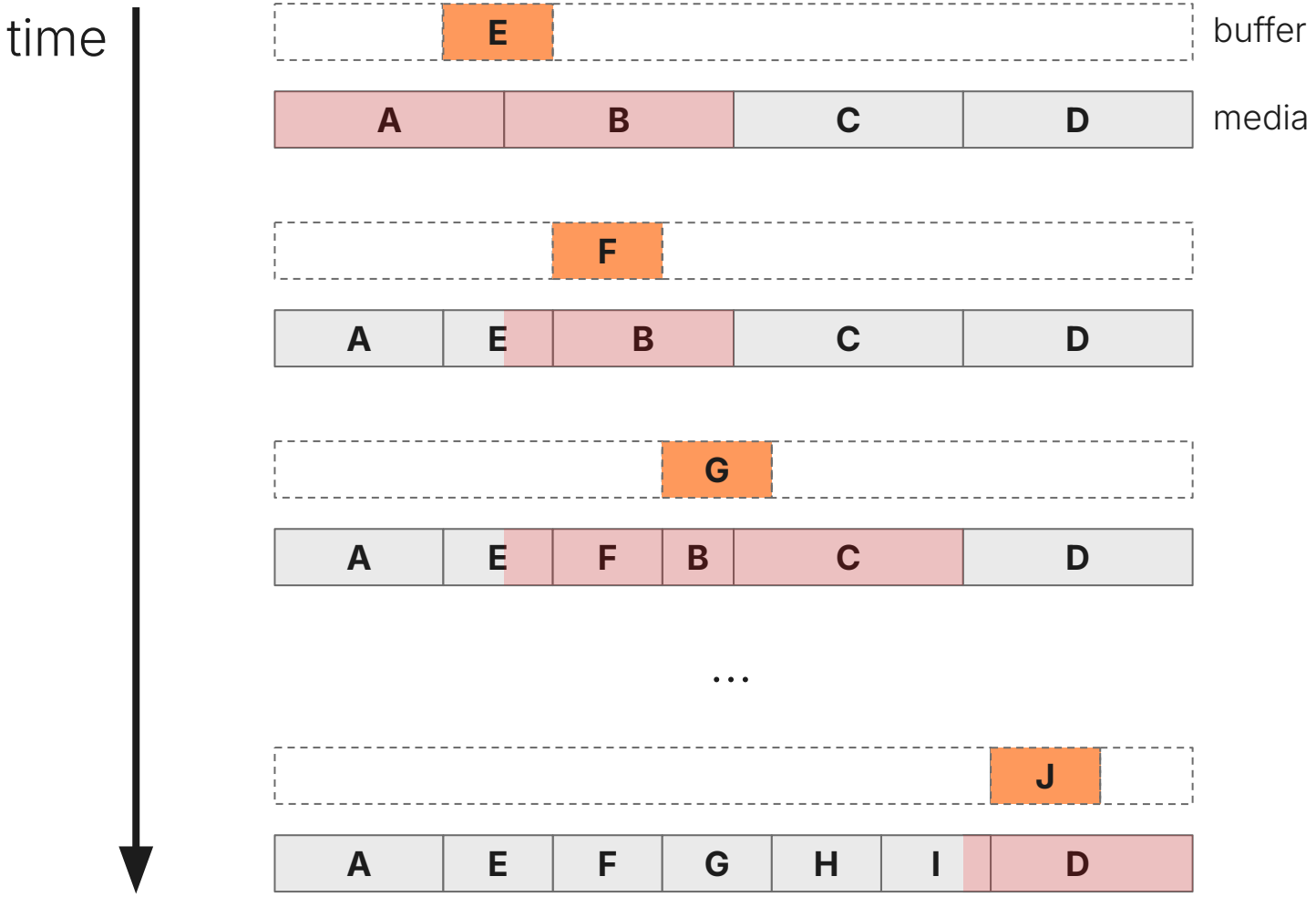
Write Coalescing



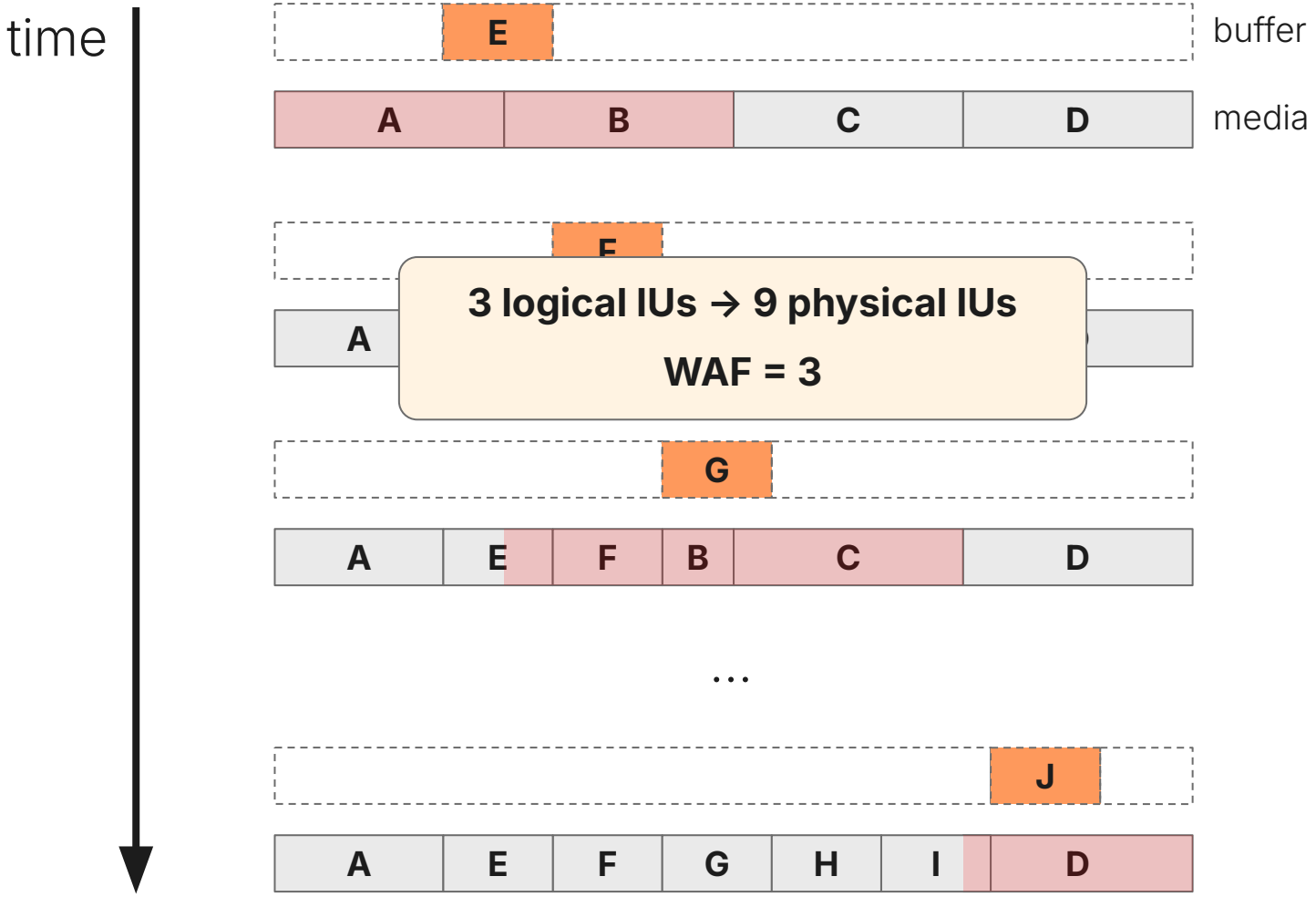
Write Coalescing



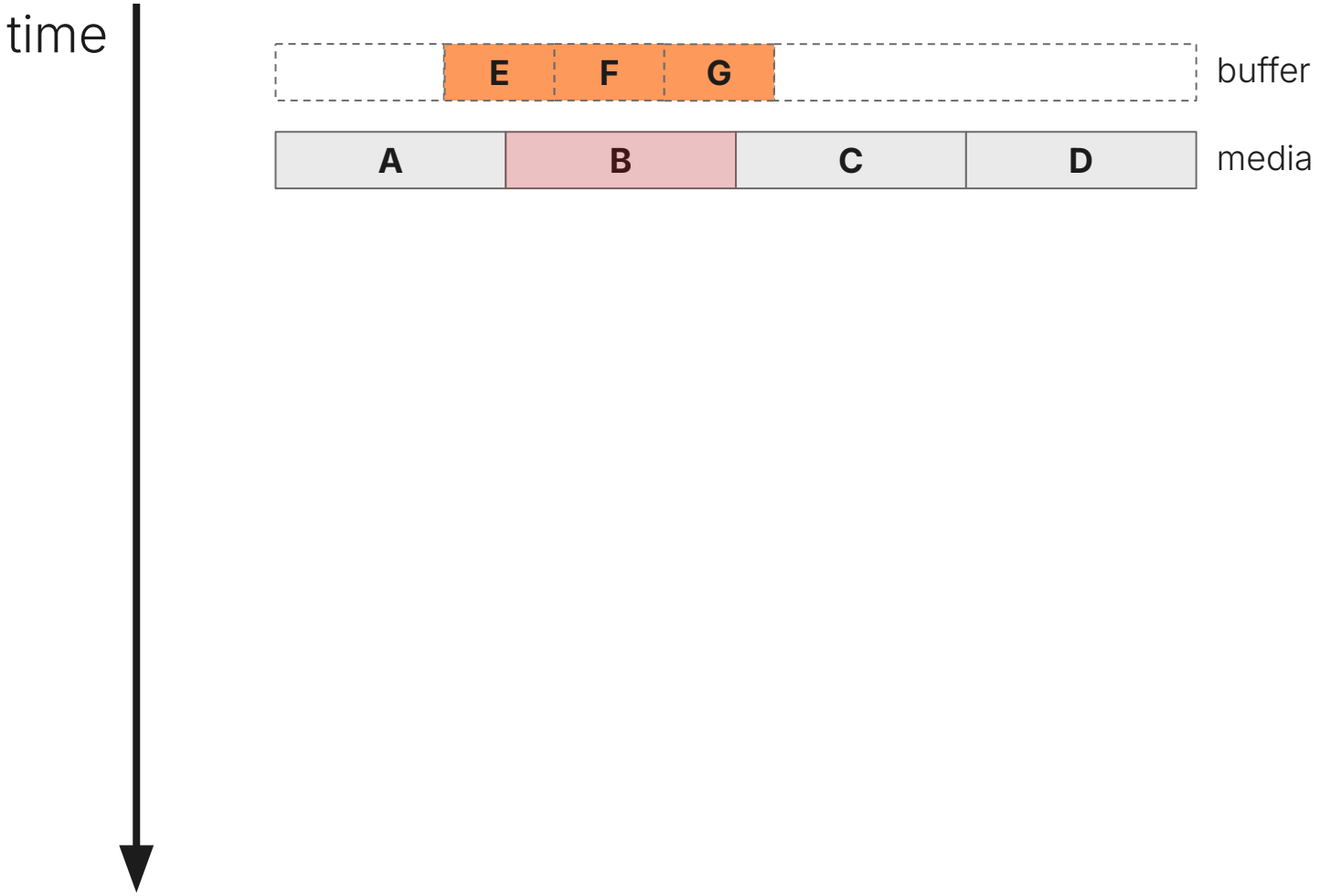
Write Coalescing



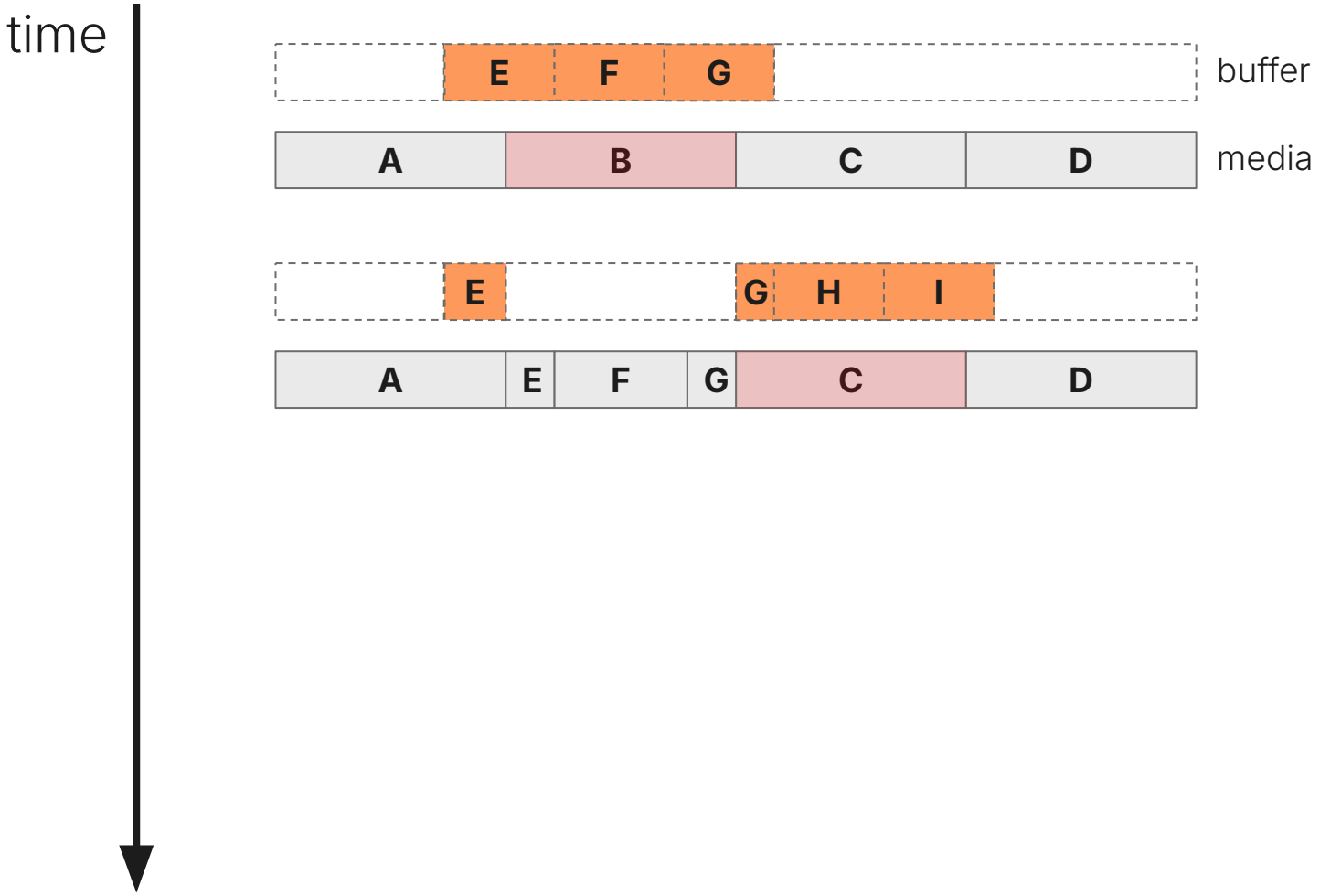
Write Coalescing



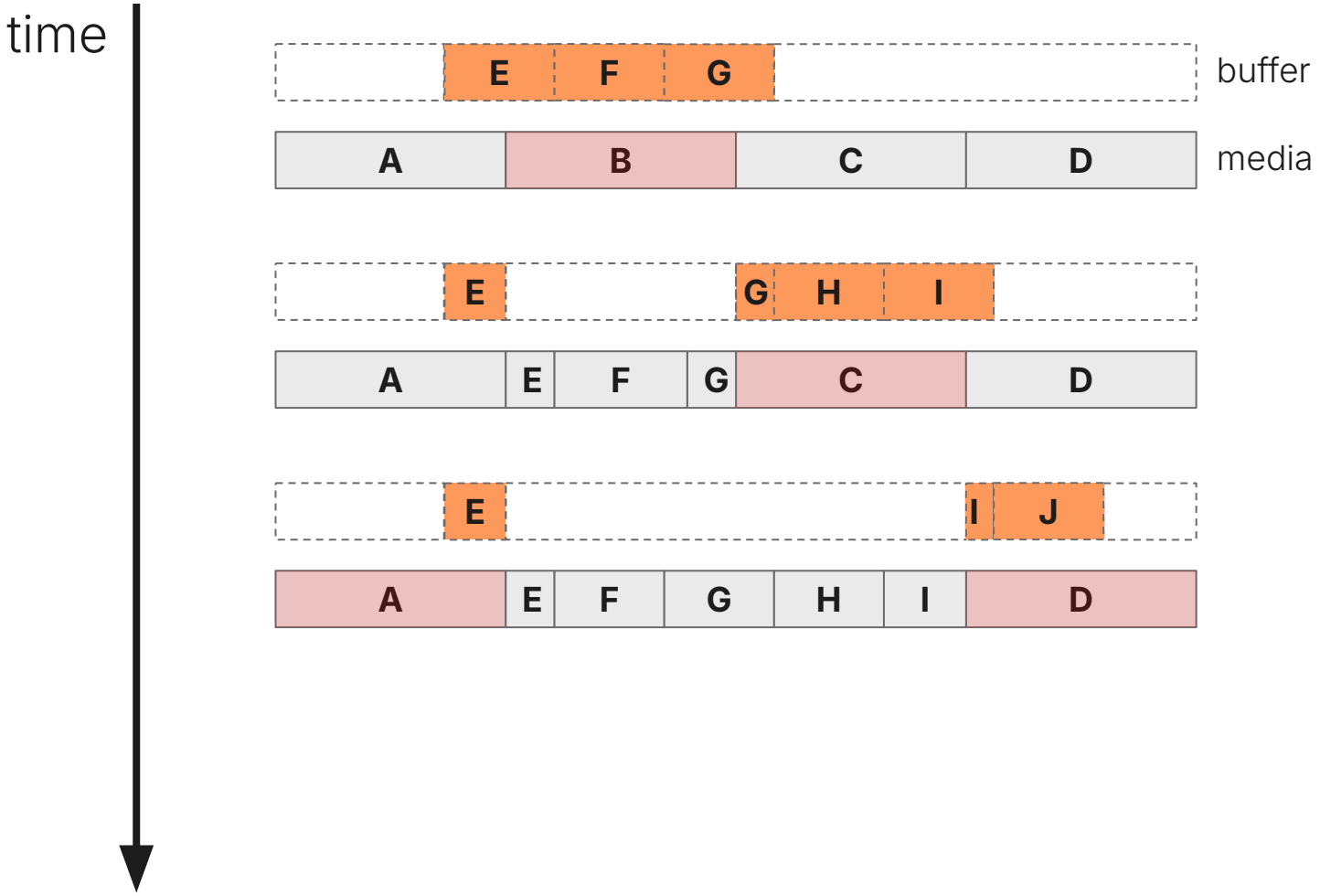
Write Coalescing



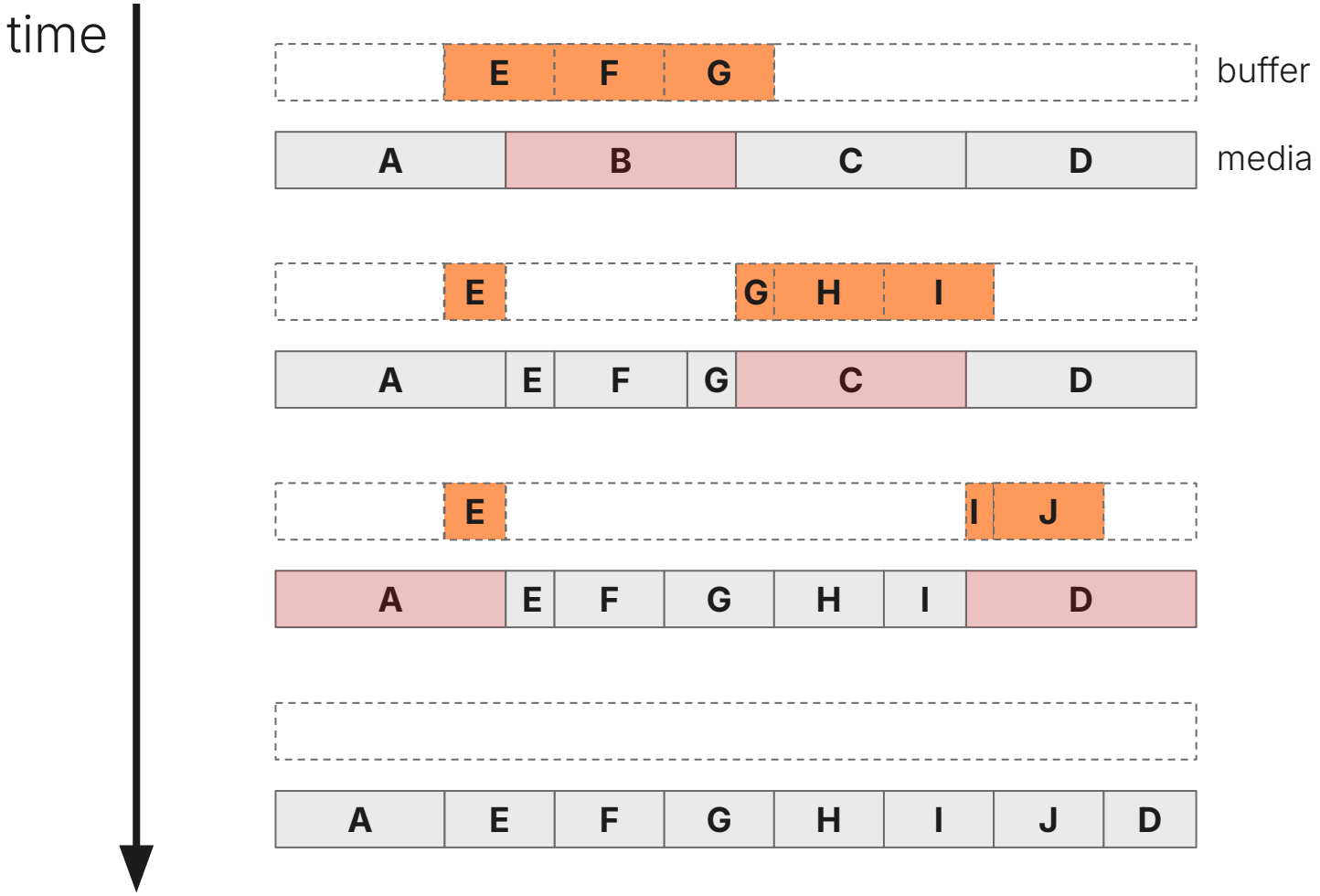
Write Coalescing



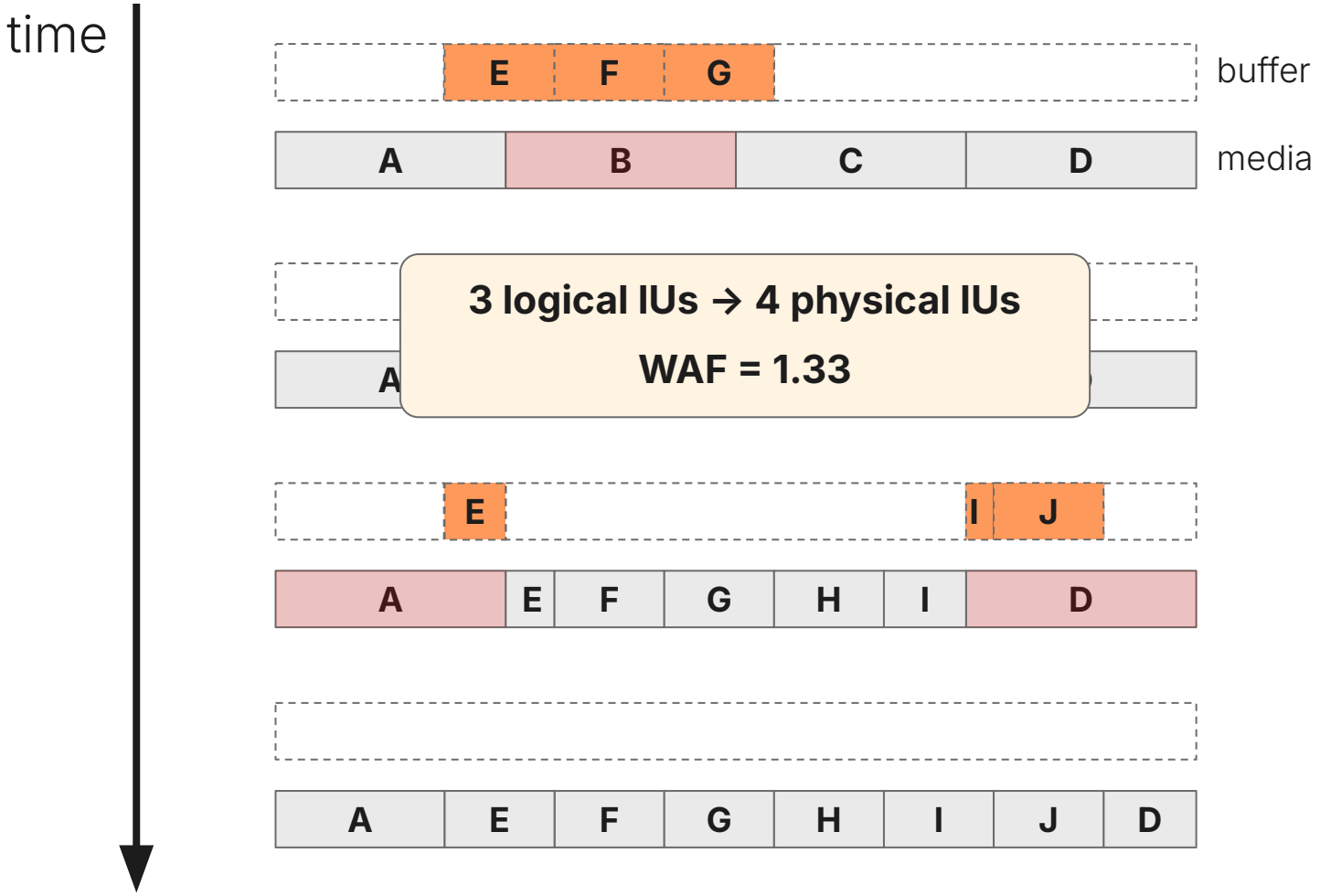
Write Coalescing



Write Coalescing

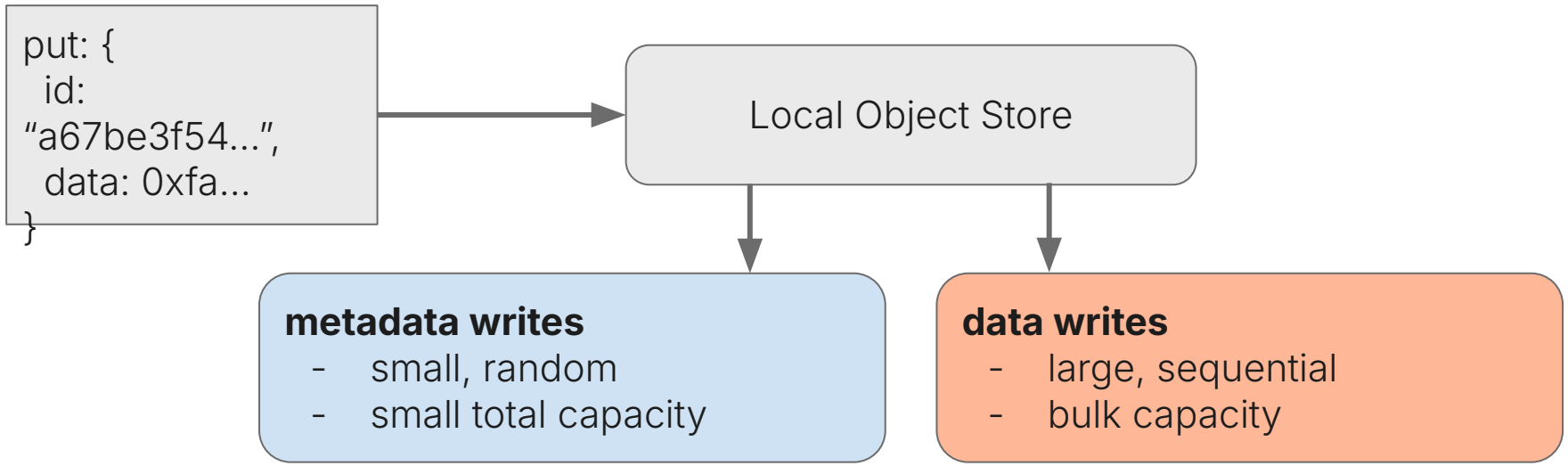


Write Coalescing

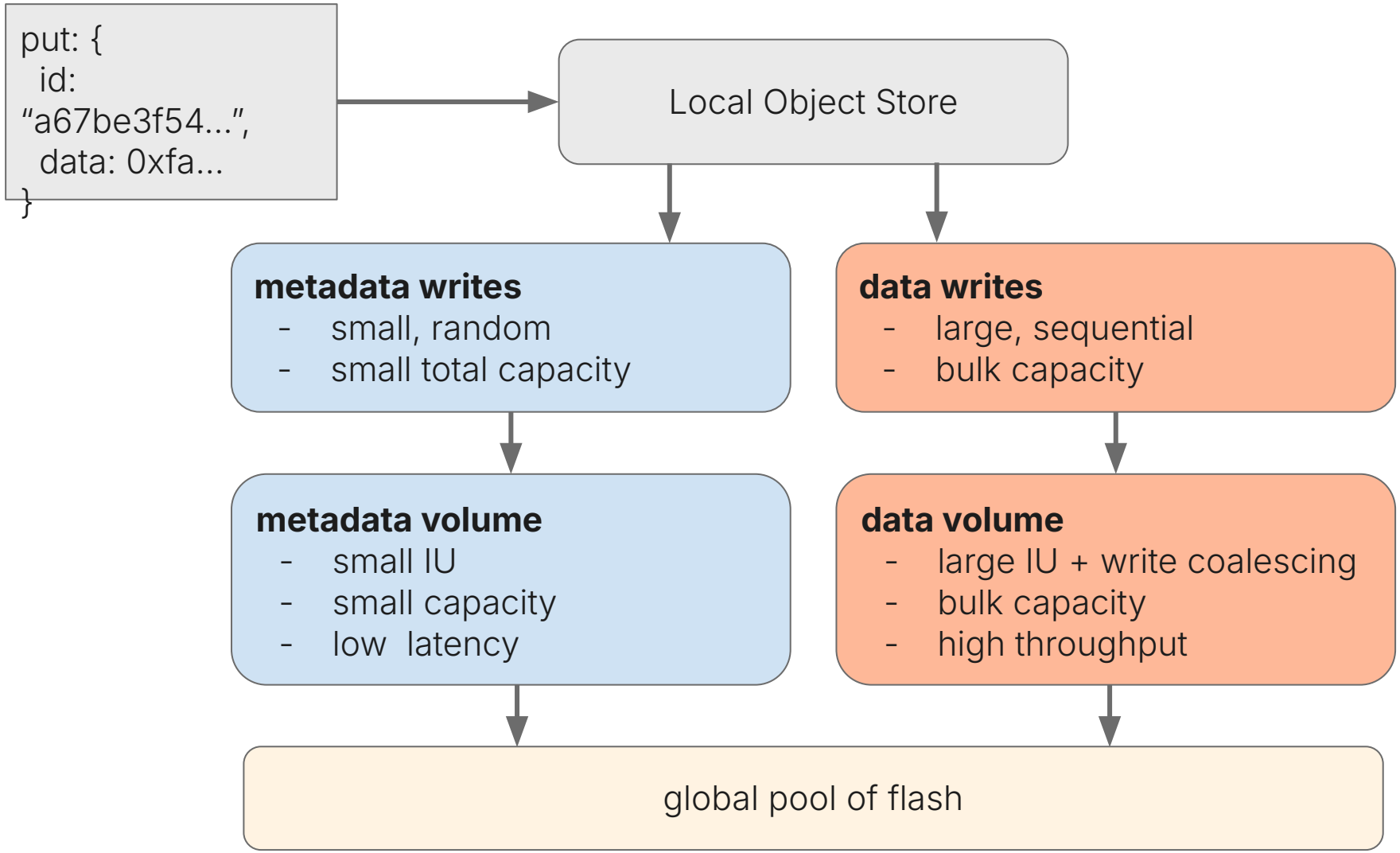


Configurable IUs

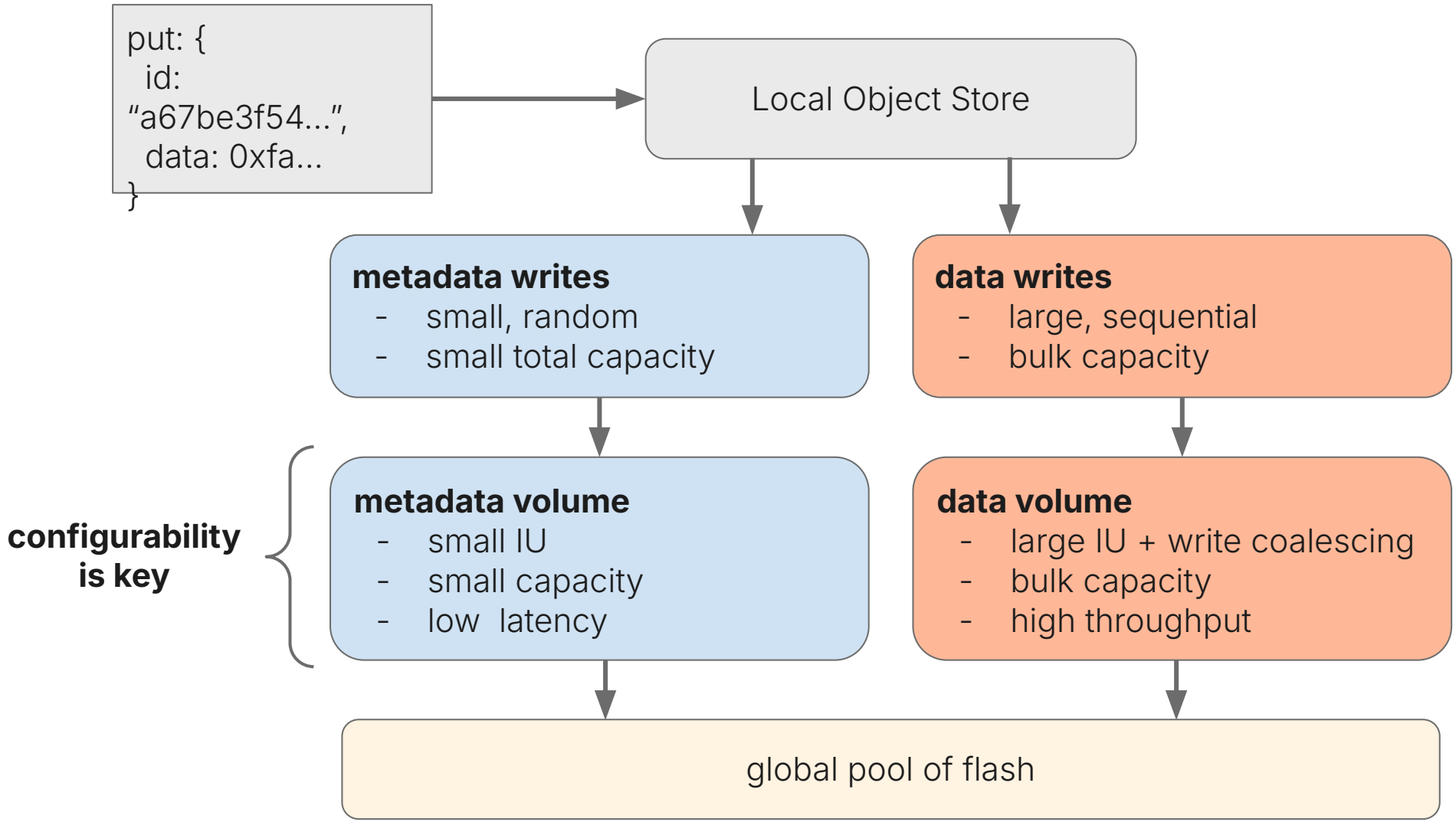
Scale



Configurable IUs



Configurable IUs

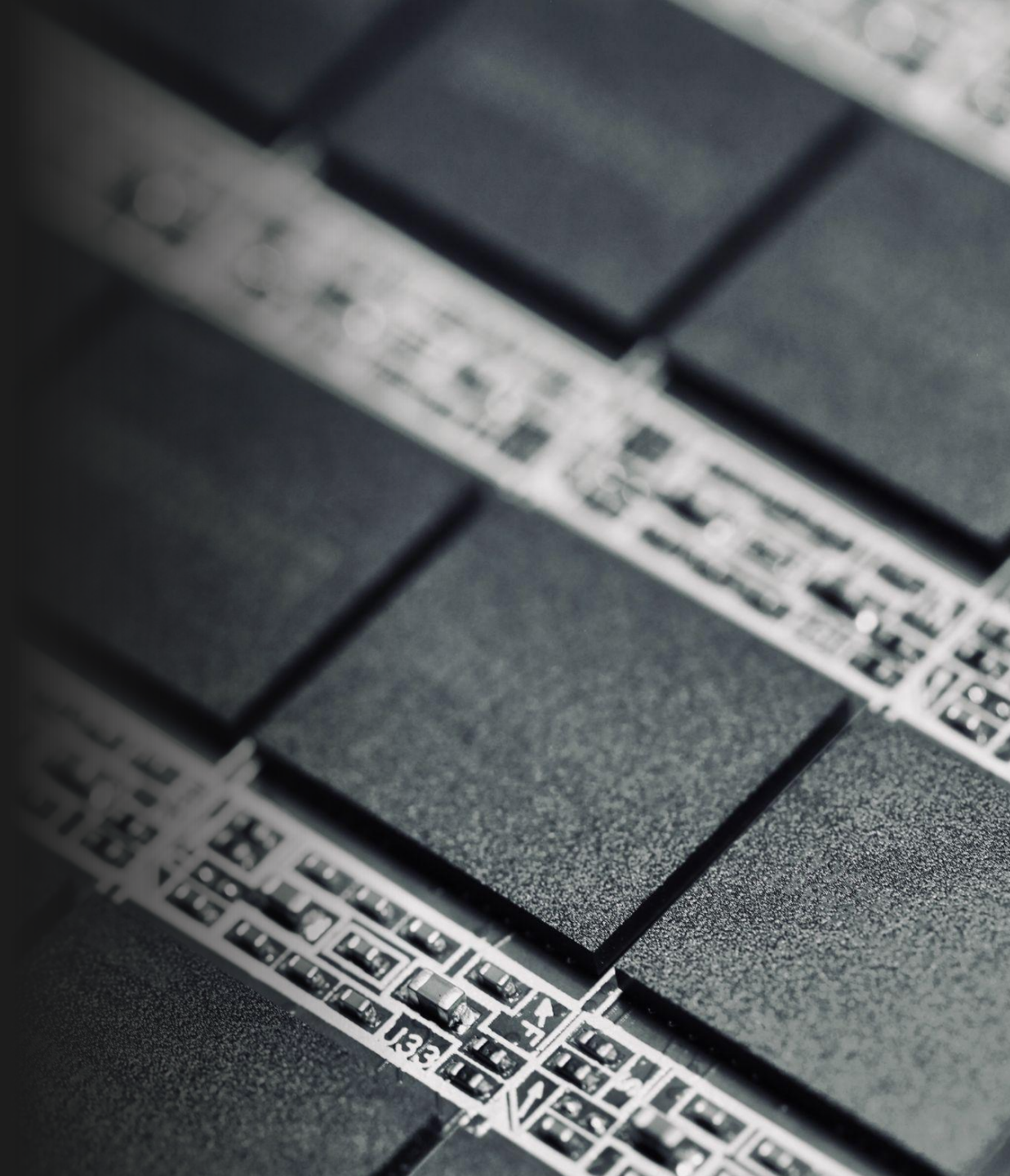


QLC Challenges

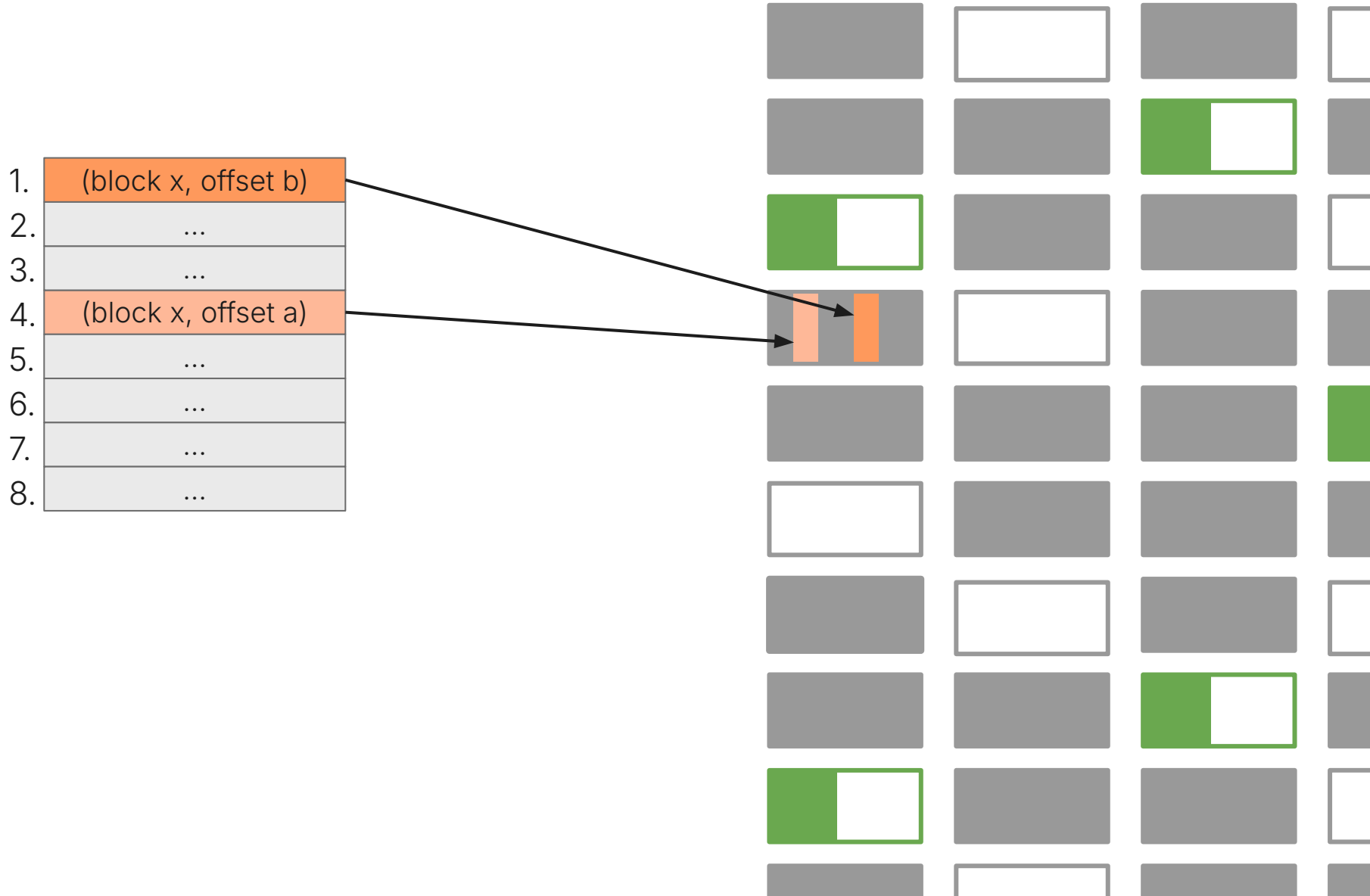
Scale

Endurance

Reliability



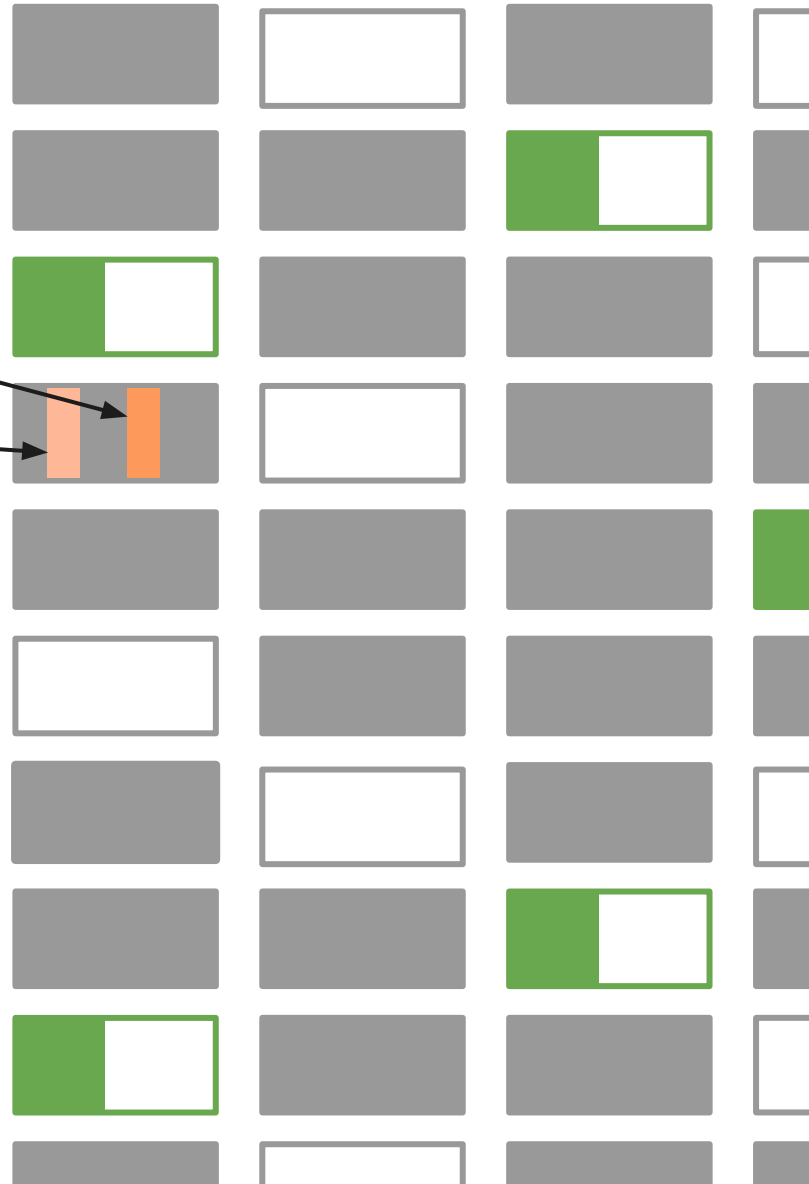
Review: Garbage Collection



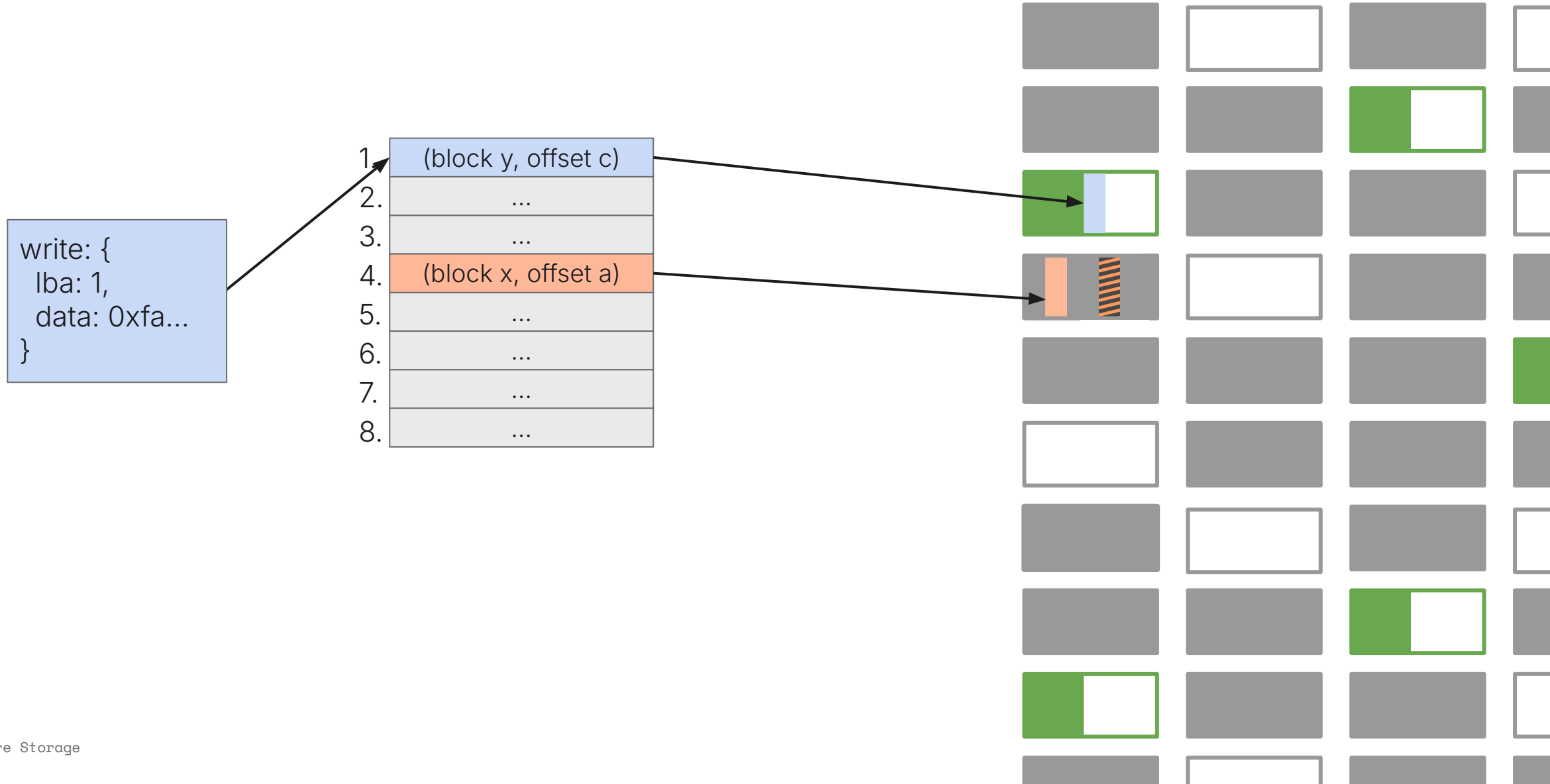
Review: Garbage Collection


```
write: {
  lba: 1,
  data: 0xfa...
}
```

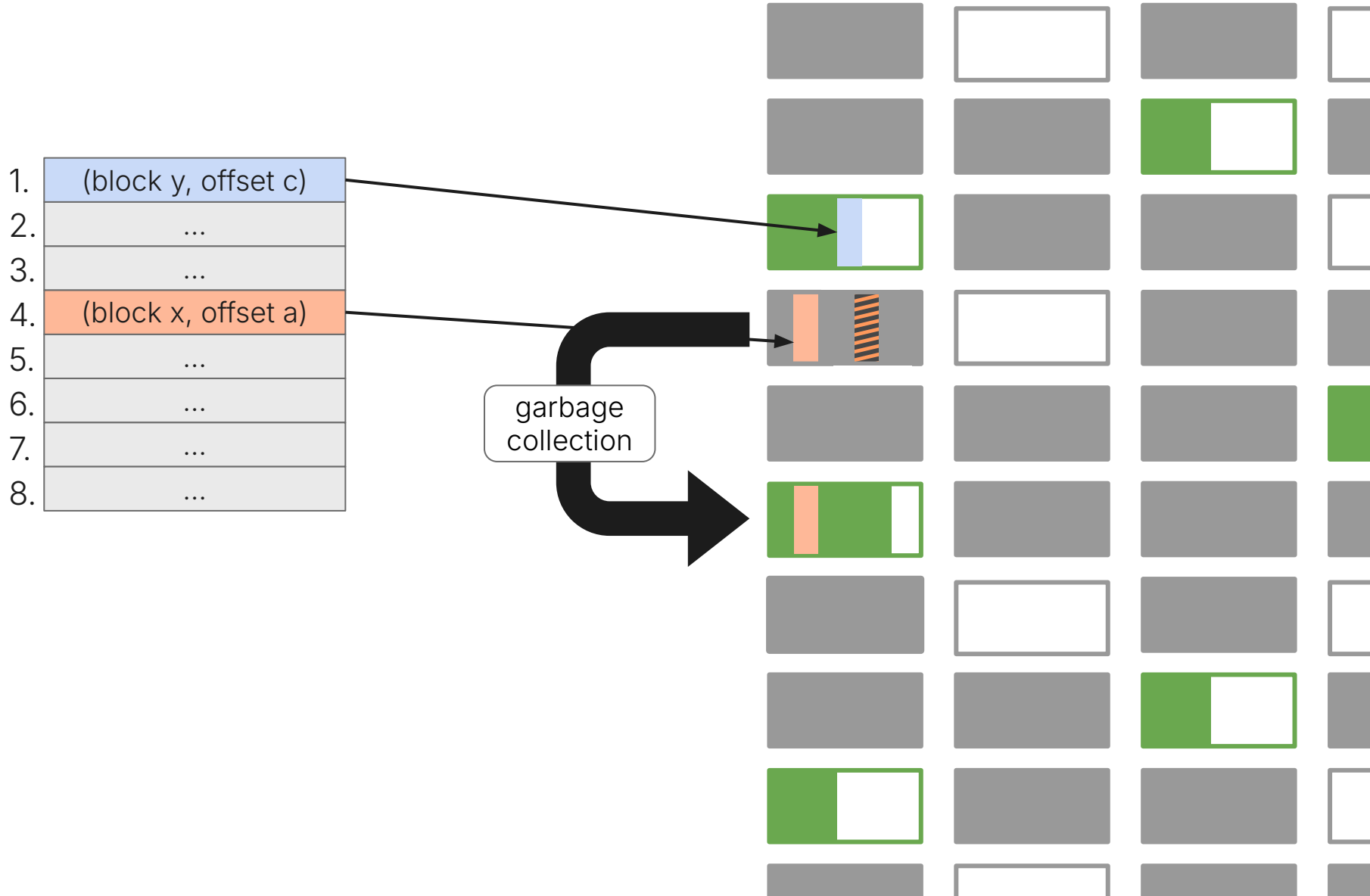
1.	(block x, offset b)
2.	...
3.	...
4.	(block x, offset a)
5.	...
6.	...
7.	...
8.	...



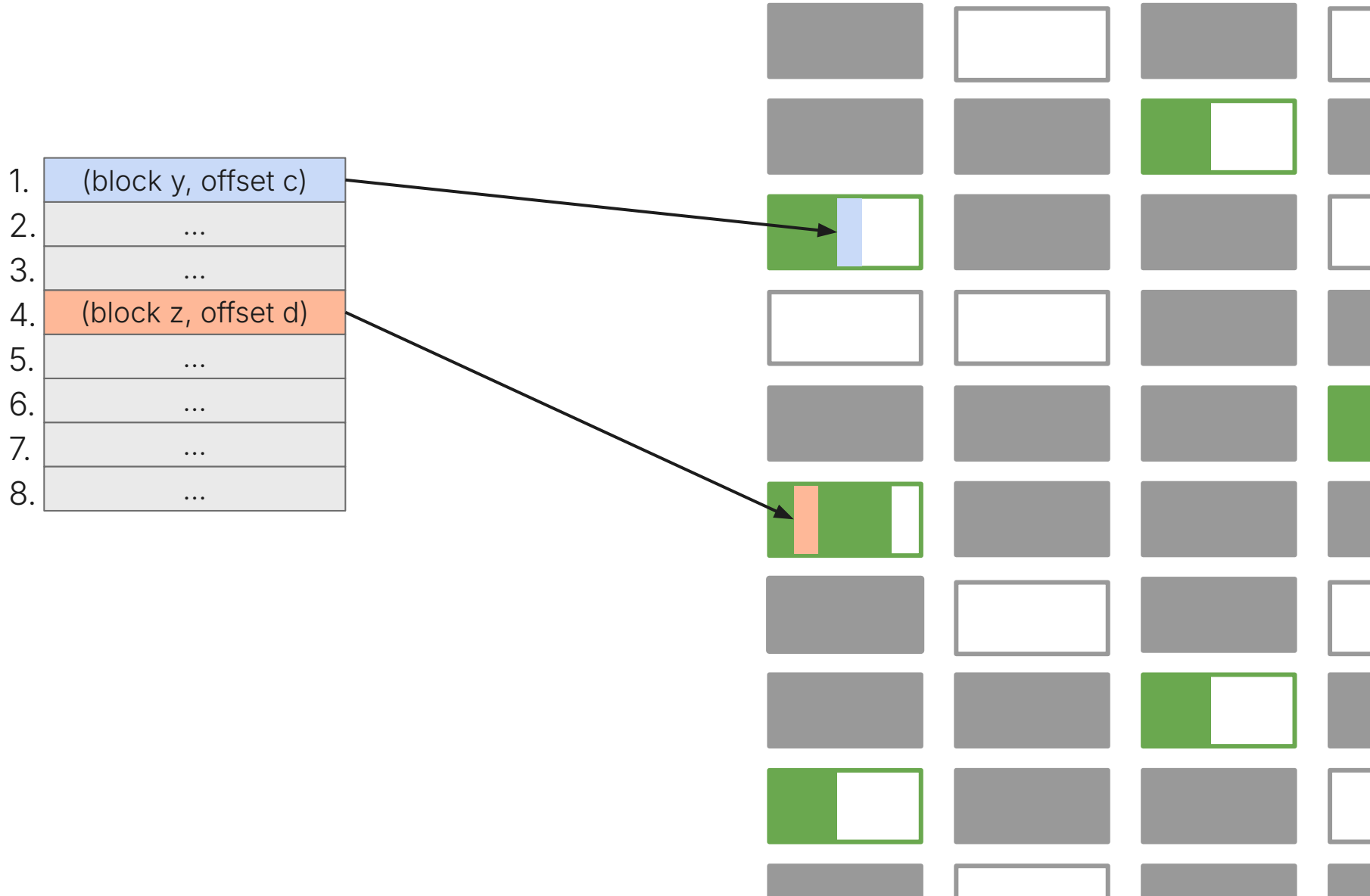
Review: Garbage Collection



Review: Garbage Collection



Review: Garbage Collection



Write Amplification

$$\text{WAF} = \frac{\text{physical written}}{\text{logical written}}$$

- QLC has relatively **low endurance**: O(thousands)
- WAF reduces media lifetime (and performance!) linearly
  e.g. WAF=2 → ½ total lifetime bytes written by user
- GC is the **dominant contributor** to WAF (others: RMW, retention)
- Option: increase over-provisioning? → direct impact to cost!
- Option: use smarter GC algorithms? → no help to random-write benchmarks!

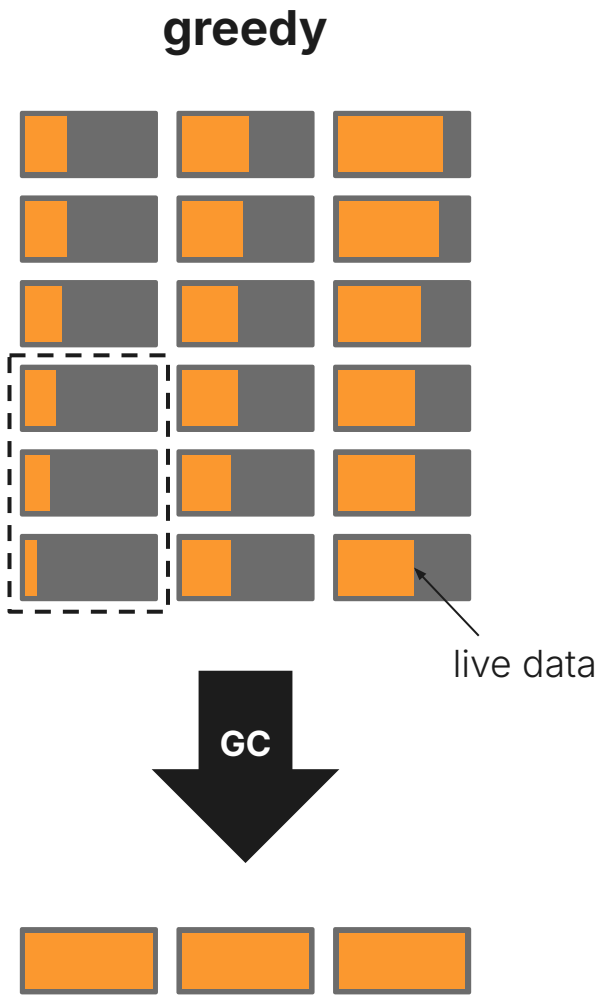
Write Amplification

- QLC has relatively **low endurance**: O(thousands)
- WAF reduces media lifetime (and performance!) linearly
  e.g. WAF=2 → ½ total lifetime bytes written by user
- GC is the **dominant contributor** to WAF (others: RMW, retention)
- Option: increase over-provisioning? → direct impact to cost!
- Option: use smarter GC algorithms? → no help to random-write benchmarks!

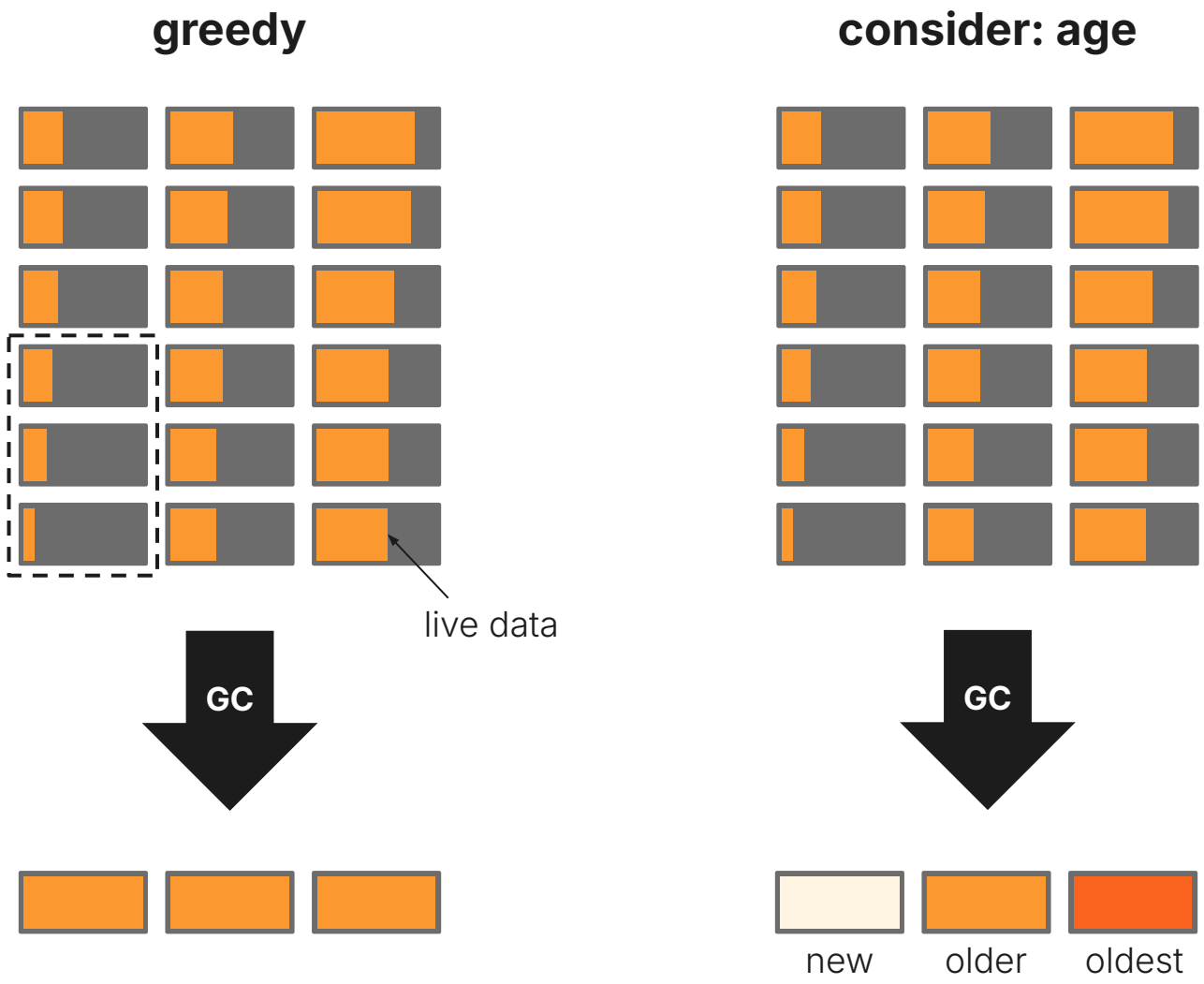
$$\text{WAF} = \frac{\text{physical written}}{\text{logical written}}$$

Key Insight: Target real-world workloads with smarter algorithms to minimize WAF.

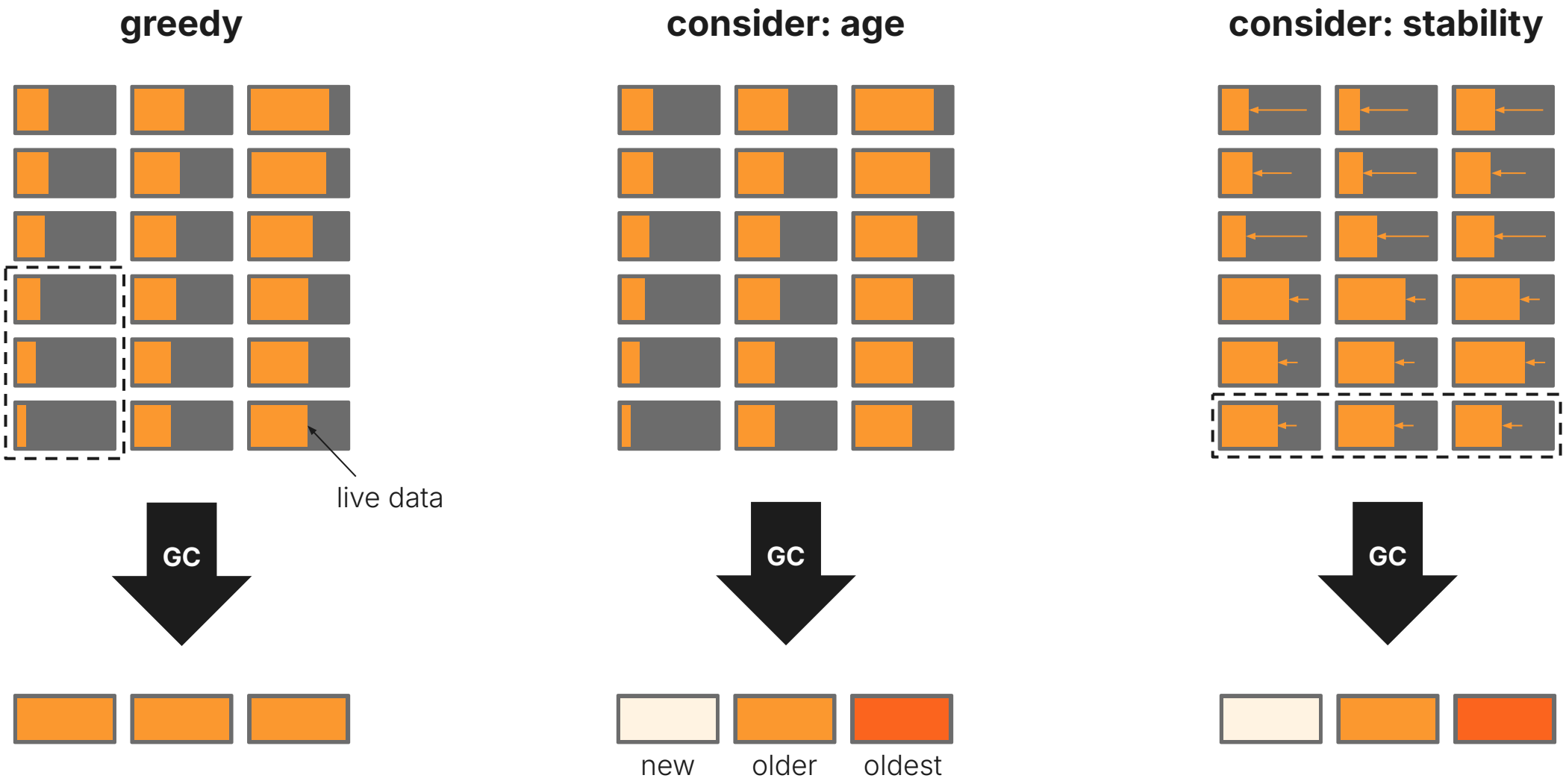
Smarter Algorithms



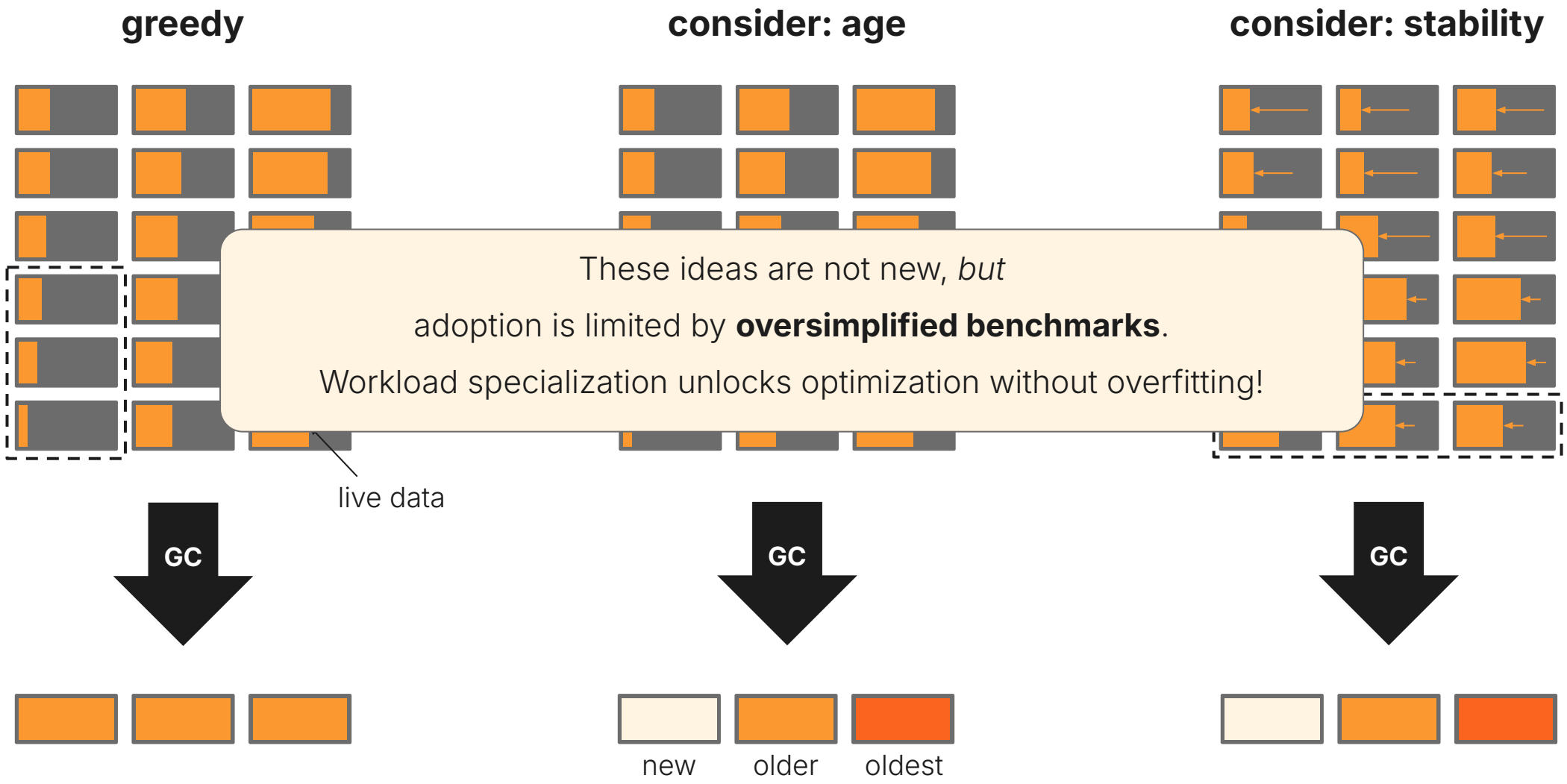
Smarter Algorithms



Smarter Algorithms



Smarter Algorithms

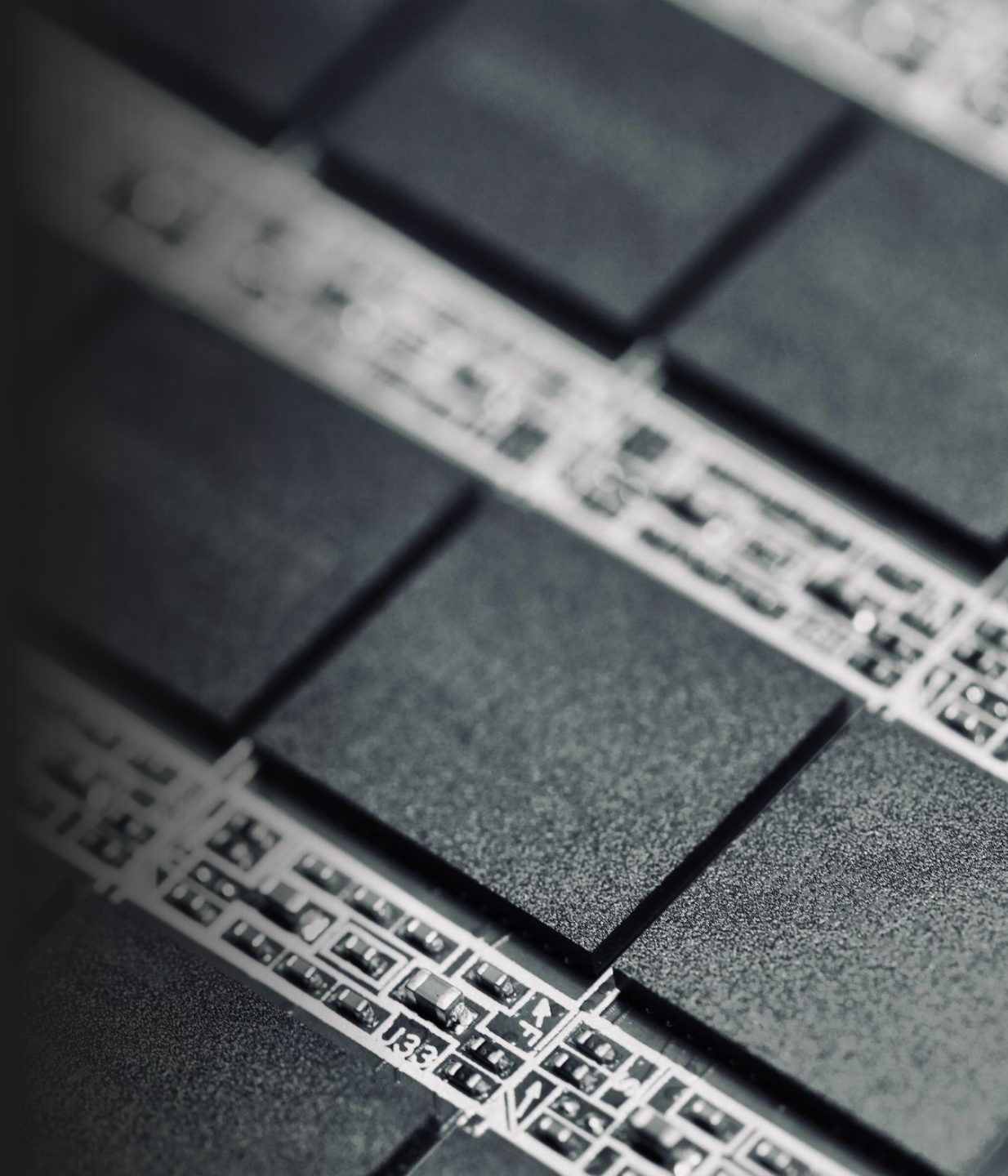


QLC Challenges

Scale

Endurance

Reliability

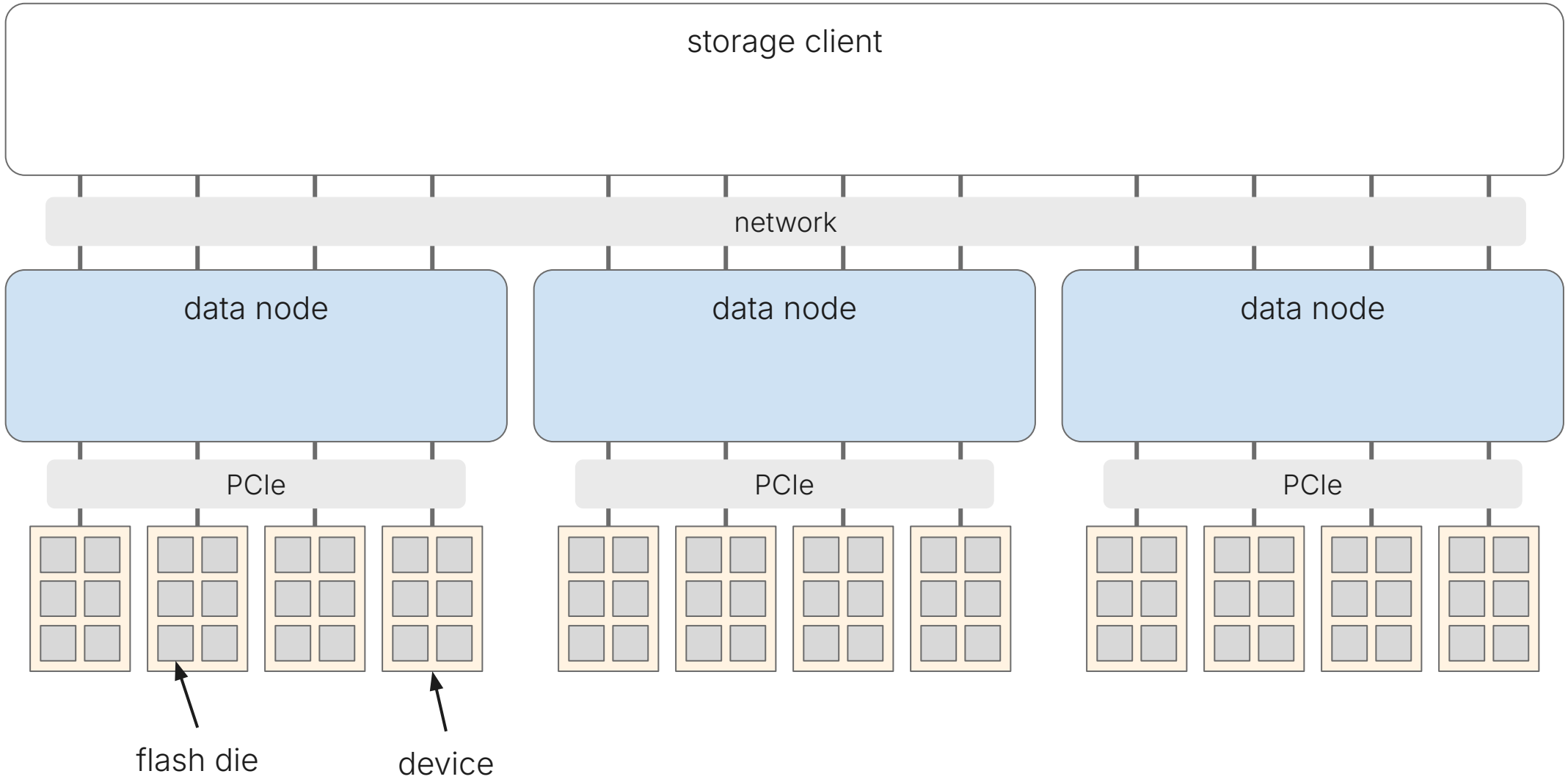


Blast Radius

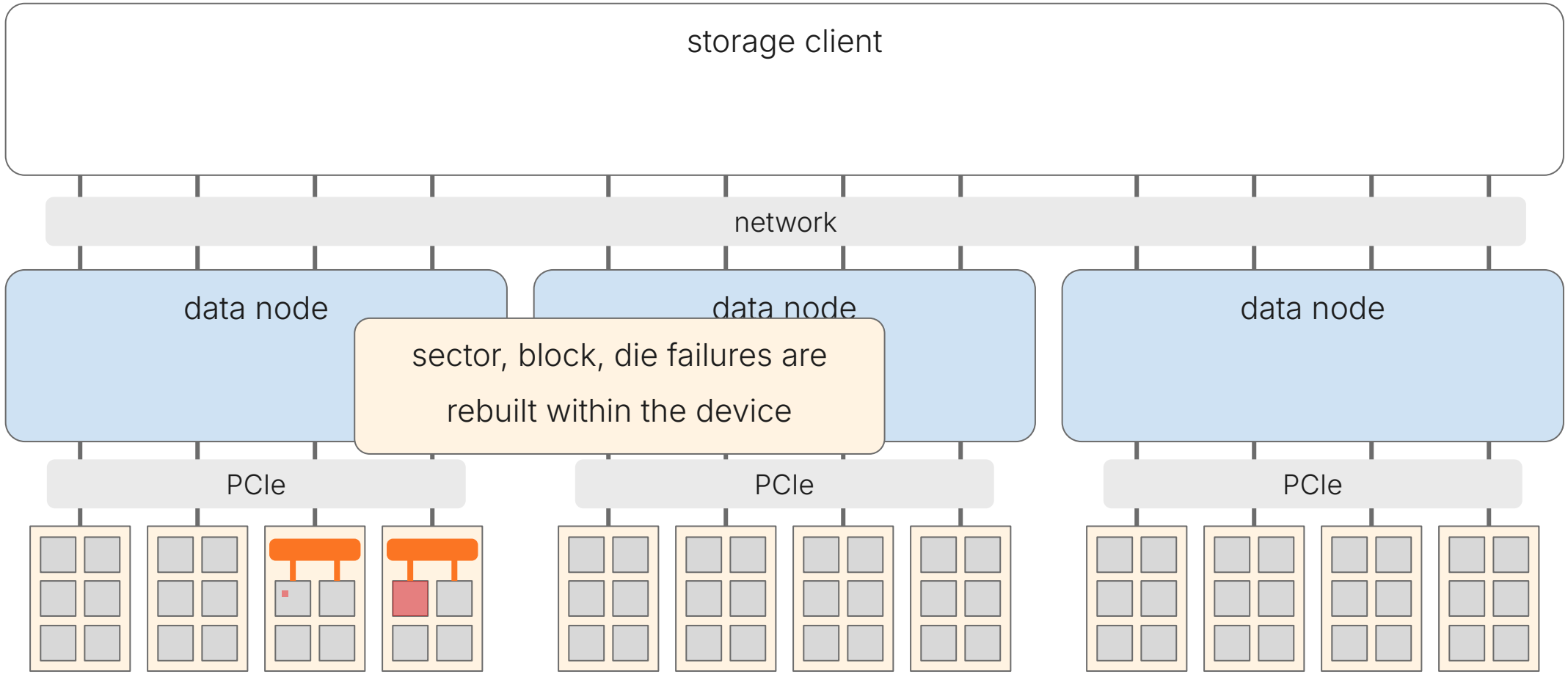
- failure modes:
  - sector ~KB
  - flash block ~100s MB
  - flash die ~100s GB
  - device ~100s TB
  - data node ~PB

- QLC density → large devices/nodes → increased impact of failures

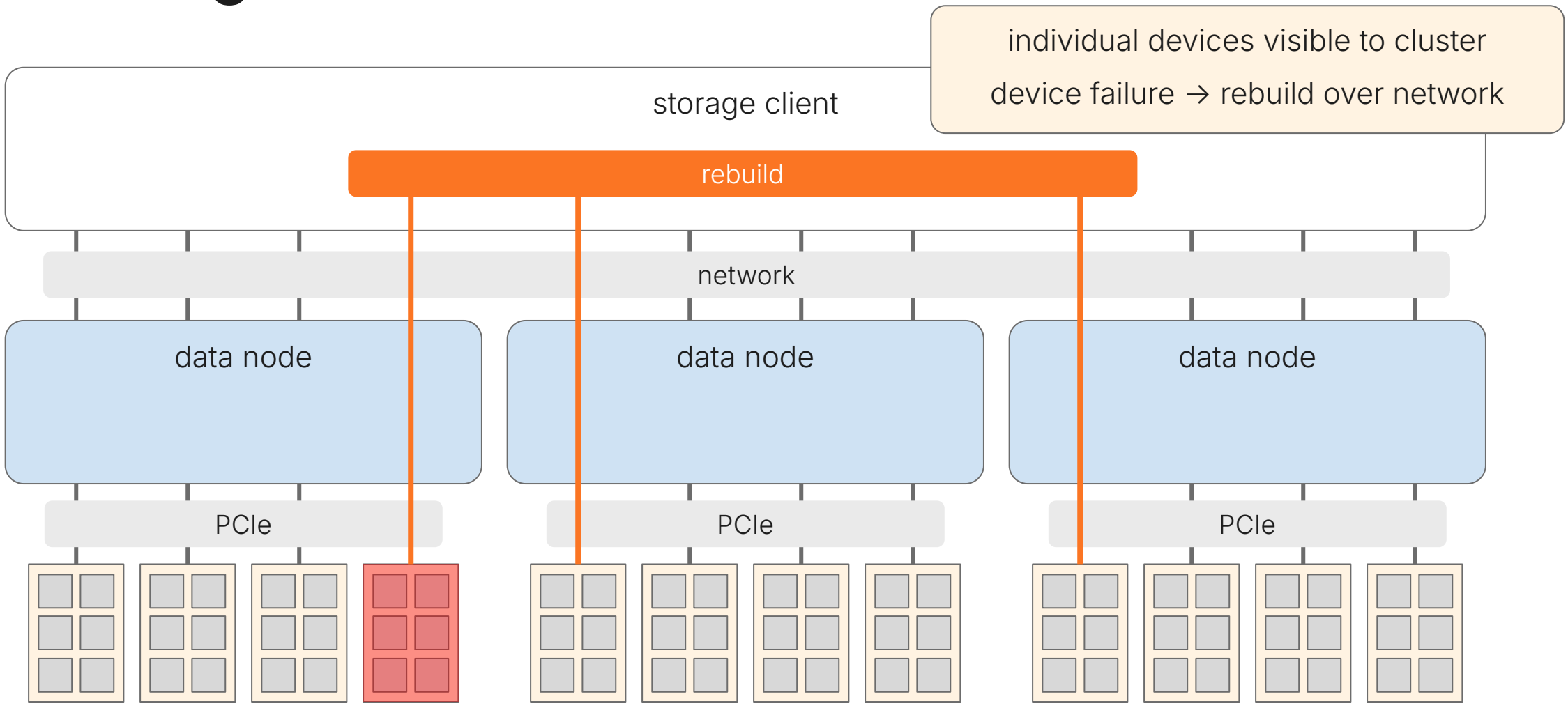
Rebuilding Failures



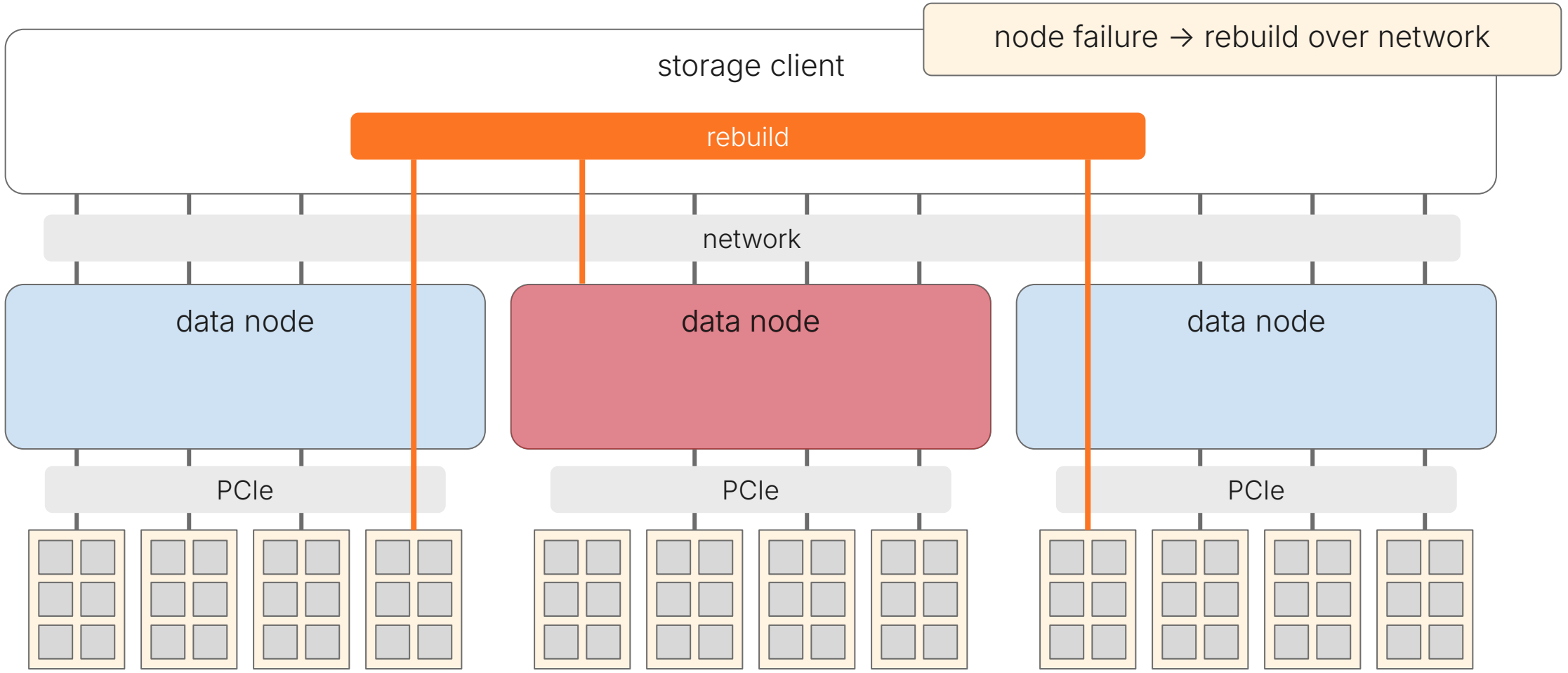
Rebuilding Failures



Rebuilding Failures



Rebuilding Failures



Blast Radius

- failure modes:
  - sector ~KB
  - flash block ~100s MB
  - flash die ~100s GB
  - device ~100s TB
  - data node ~PB

- QLC density → large devices/nodes → increased impact of failures

Blast Radius

- failure modes:
  - sector ~KB
  - flash block ~100s MB **← most common, small impact**
  - flash die ~100s GB **← rare, moderate impact**
  - device ~100s TB **← moderate frequency, huge impact!**
  - data node ~PB

- QLC density → large devices/nodes → increased impact of failures

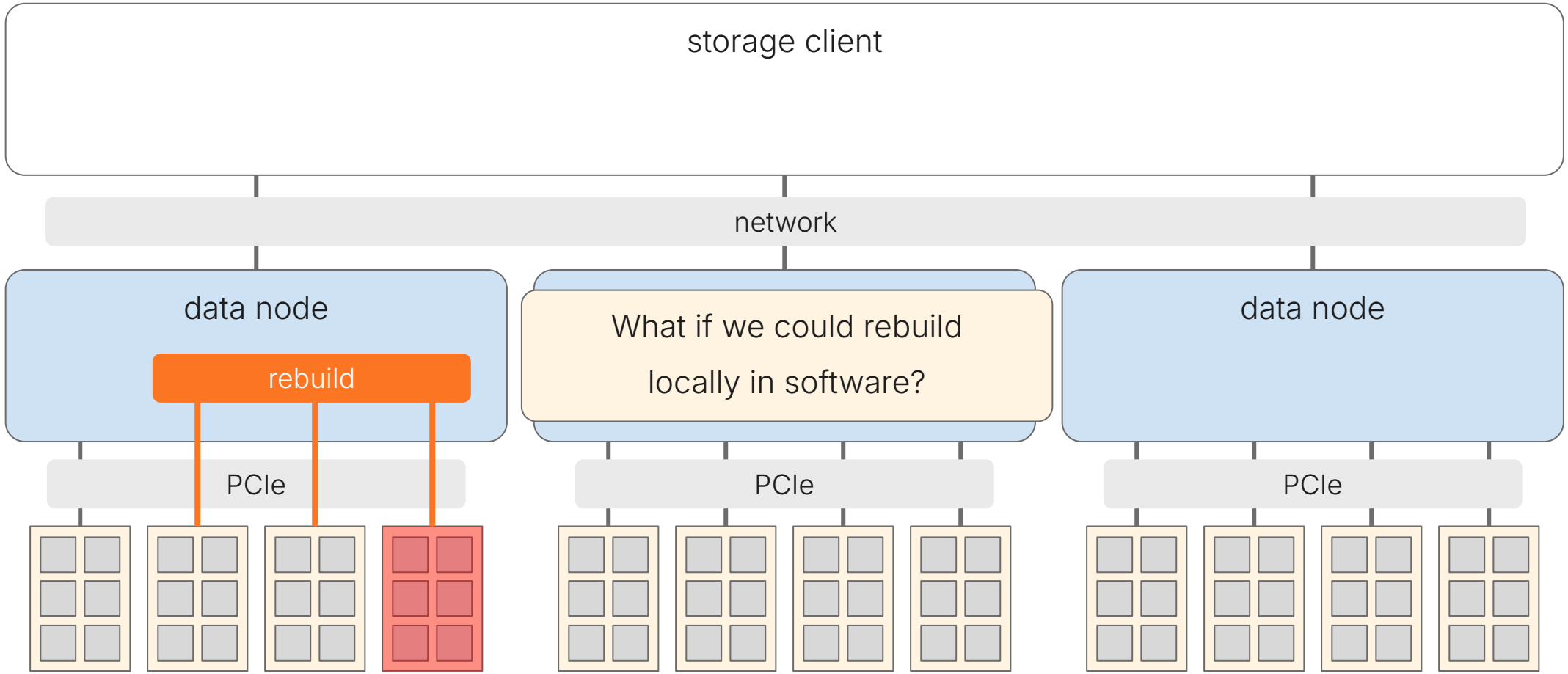
Blast Radius

- failure modes:
  - sector ~KB
  - flash block ~100s MB **← most common, small impact**
  - flash die ~100s GB **← rare, moderate impact**
  - device ~100s TB **← moderate frequency, huge impact!**
  - data node ~PB

- QLC density → large devices/nodes → increased impact of failures

Key Insight: Protect efficiently against dominant failure modes.

Node-Local Parity?



Recap: QLC Challenges

Scale

- large IUs
- write coalescing
- data/metadata

Endurance

- real workloads
- smarter algos
- minimize WAF

Reliability

- failure modes
- blast radius
- local parity

Recap: QLC Challenges

Scale

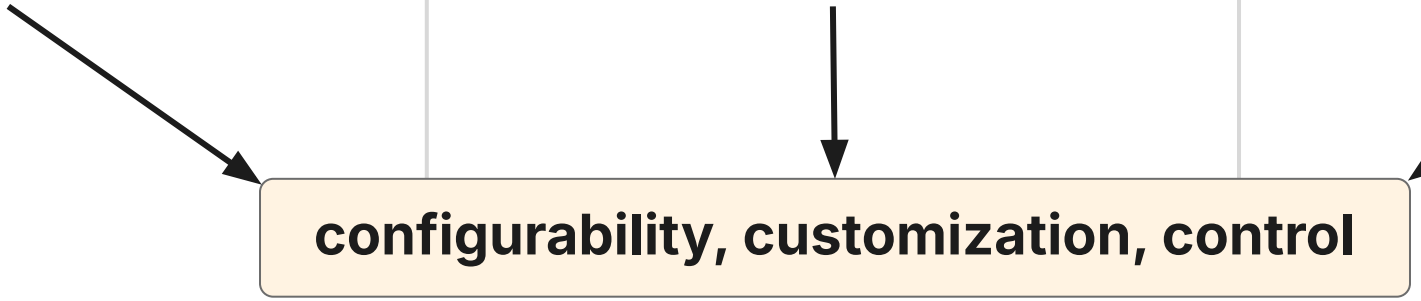
- large IUs
- write coalescing
- data/metadata

Endurance

- real workloads
- smarter algos
- minimize WAF

Reliability

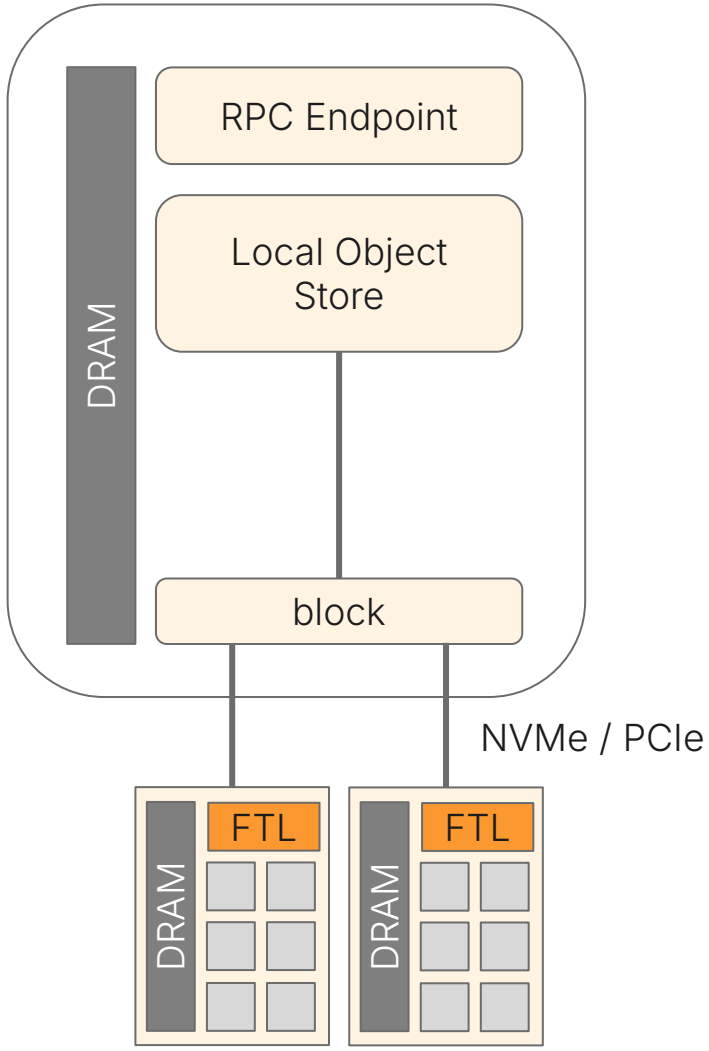
- failure modes
- blast radius
- local parity



Agenda

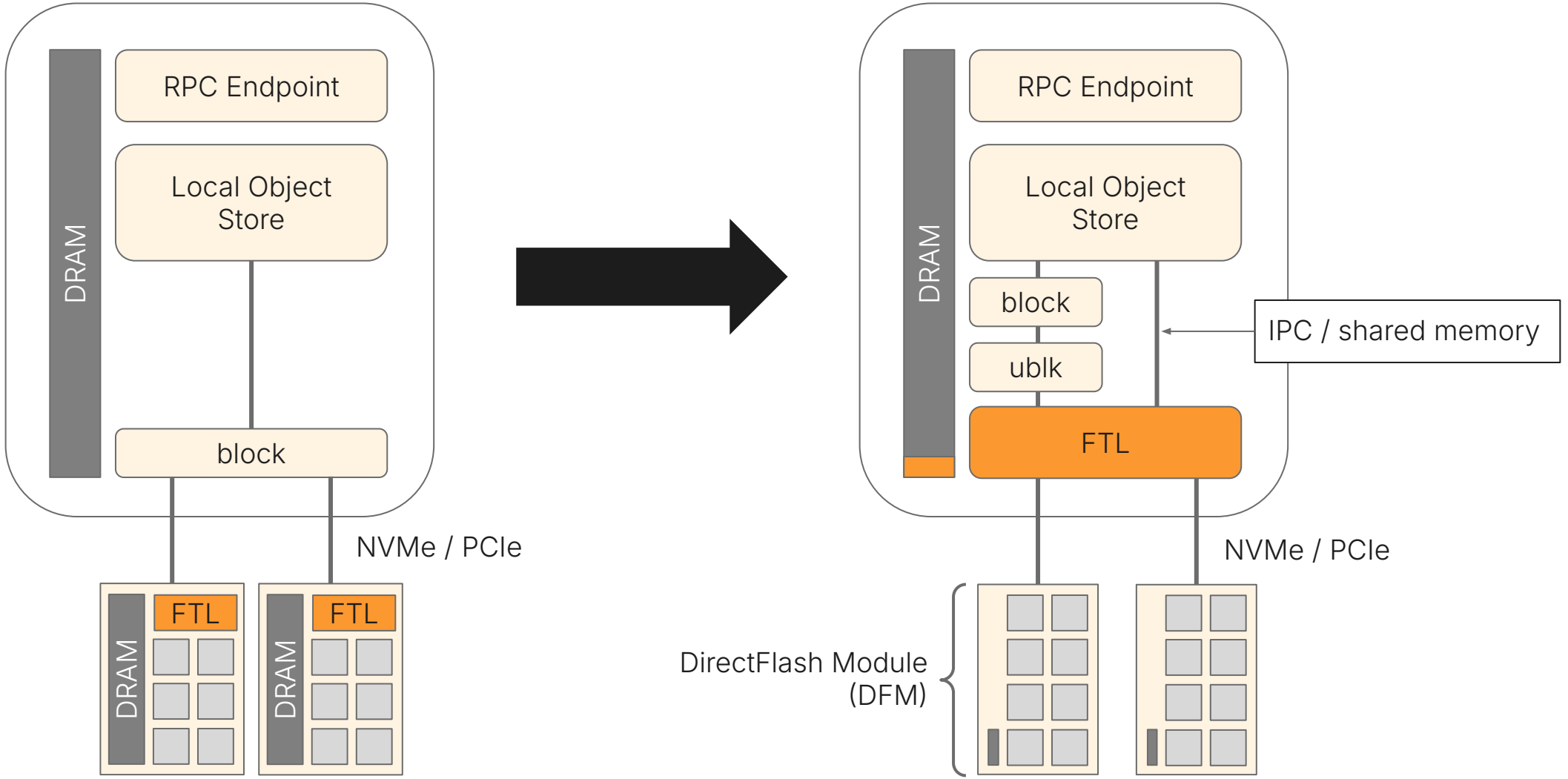
Context	Exascale Storage
Problem	QLC Challenges
Solution	Architectural Perspective

Host-Based FTL



NVMe / PCIe

Host-Based FTL



Late-Binding Configuration

Scale

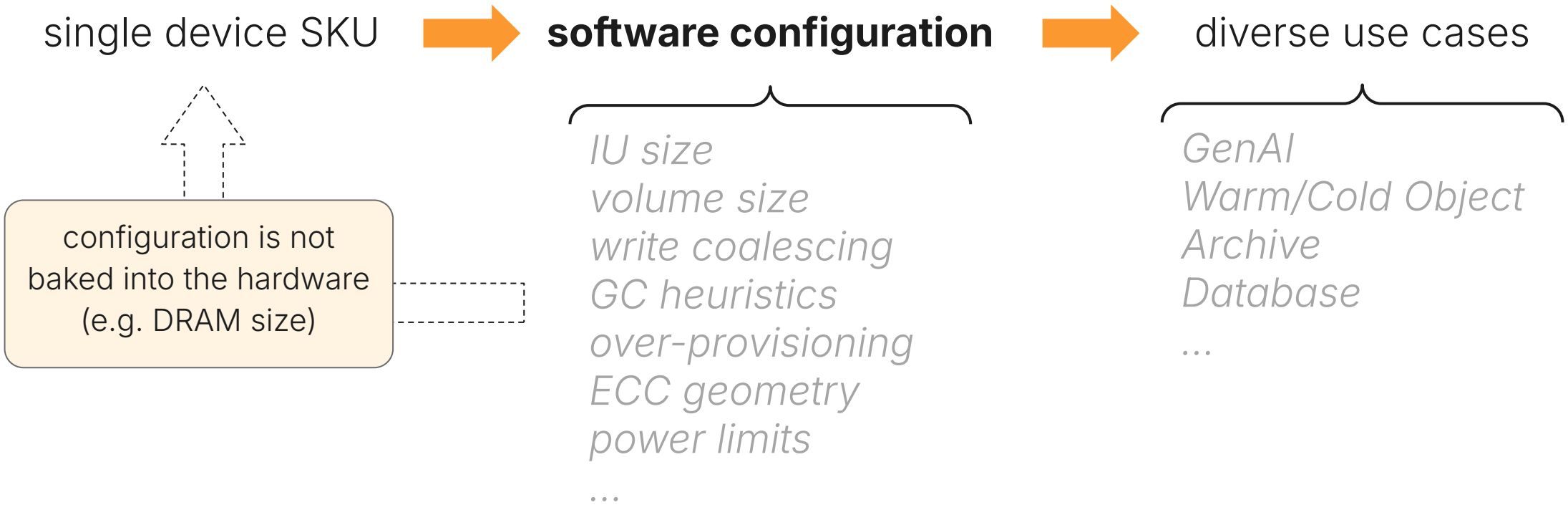
single device SKU → **software configuration** → diverse use cases

IU size
volume size
write coalescing
GC heuristics
over-provisioning
ECC geometry
power limits
...

GenAI
Warm/Cold Object
Archive
Database
...

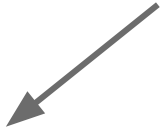
Late-Binding Configuration

Scale



What Next?

driving efficiency in exascale storage



configurability

Scale drives the balance from abstraction to full-stack optimization.

Configure storage media to match the natural patterns of the local object store.

What Next?

driving efficiency in exascale storage

configurability

Scale drives the balance from abstraction to full-stack optimization.

Configure storage media to match the natural patterns of the local object store.

media lifetime

low WAF → media lasts **10 years**

How can we fully leverage this, technically and operationally?

PURESTORAGE®