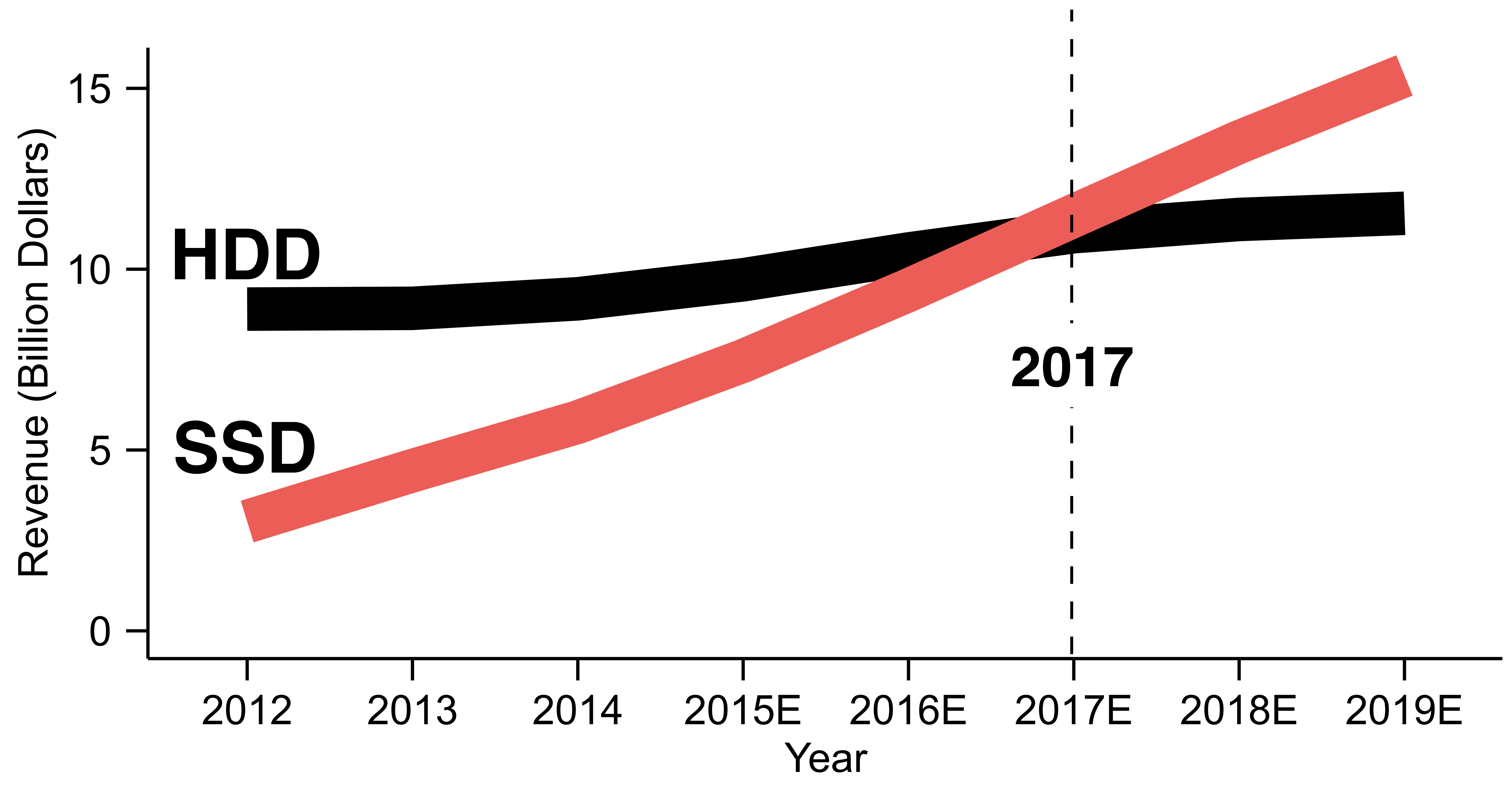
The Unwritten Contract of Solid State Drives

Jun He, Sudarsun Kannan,
Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau

Department of Computer Sciences, University of Wisconsin - Madison

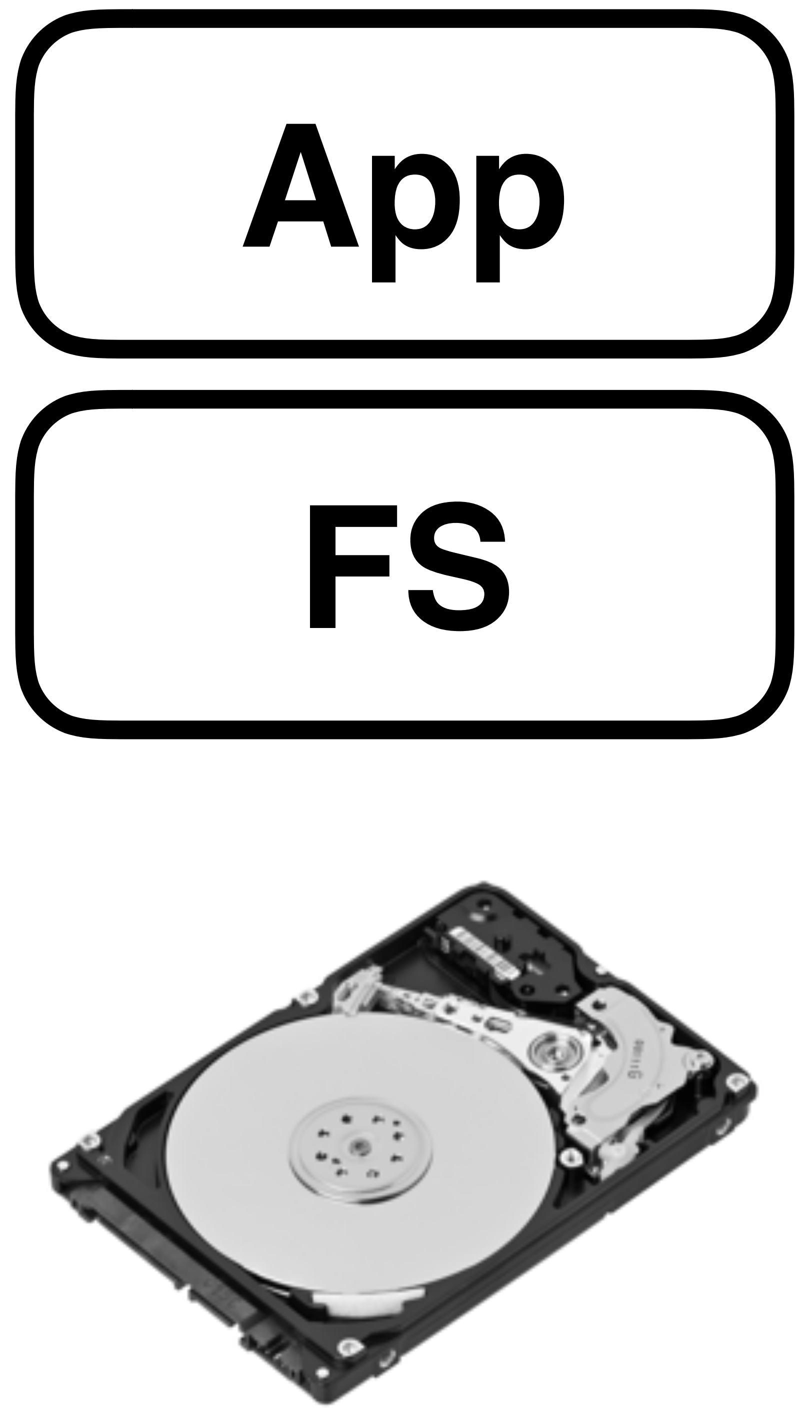
WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Enterprise SSD revenue is expected to exceed enterprise HDD in 2017

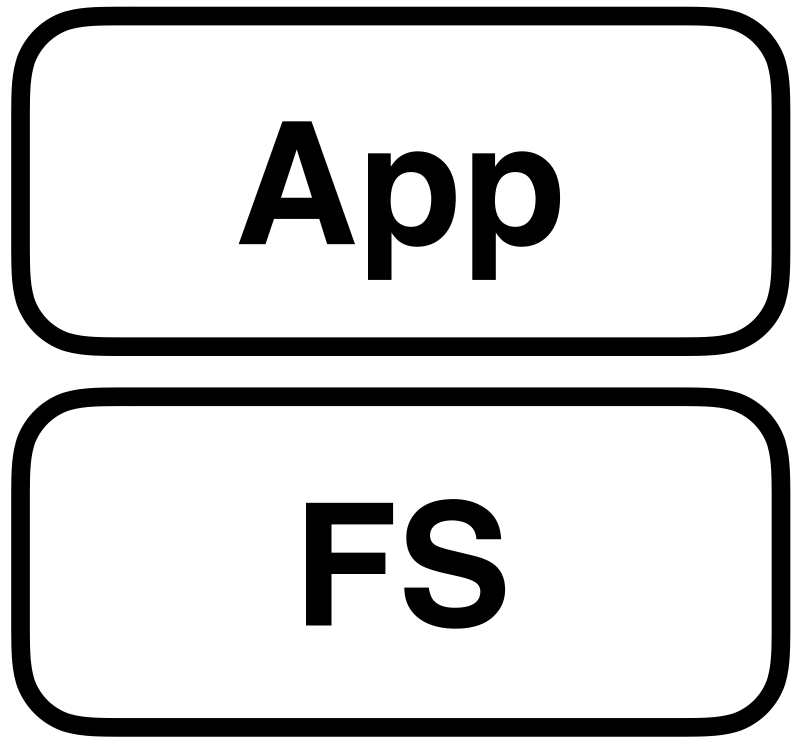


Source: Gartner, Stifel Estimates
https://www.theregister.co.uk/2016/01/07/gartner_enterprise_ssd_hdd_revenue_crossover_in_2017/

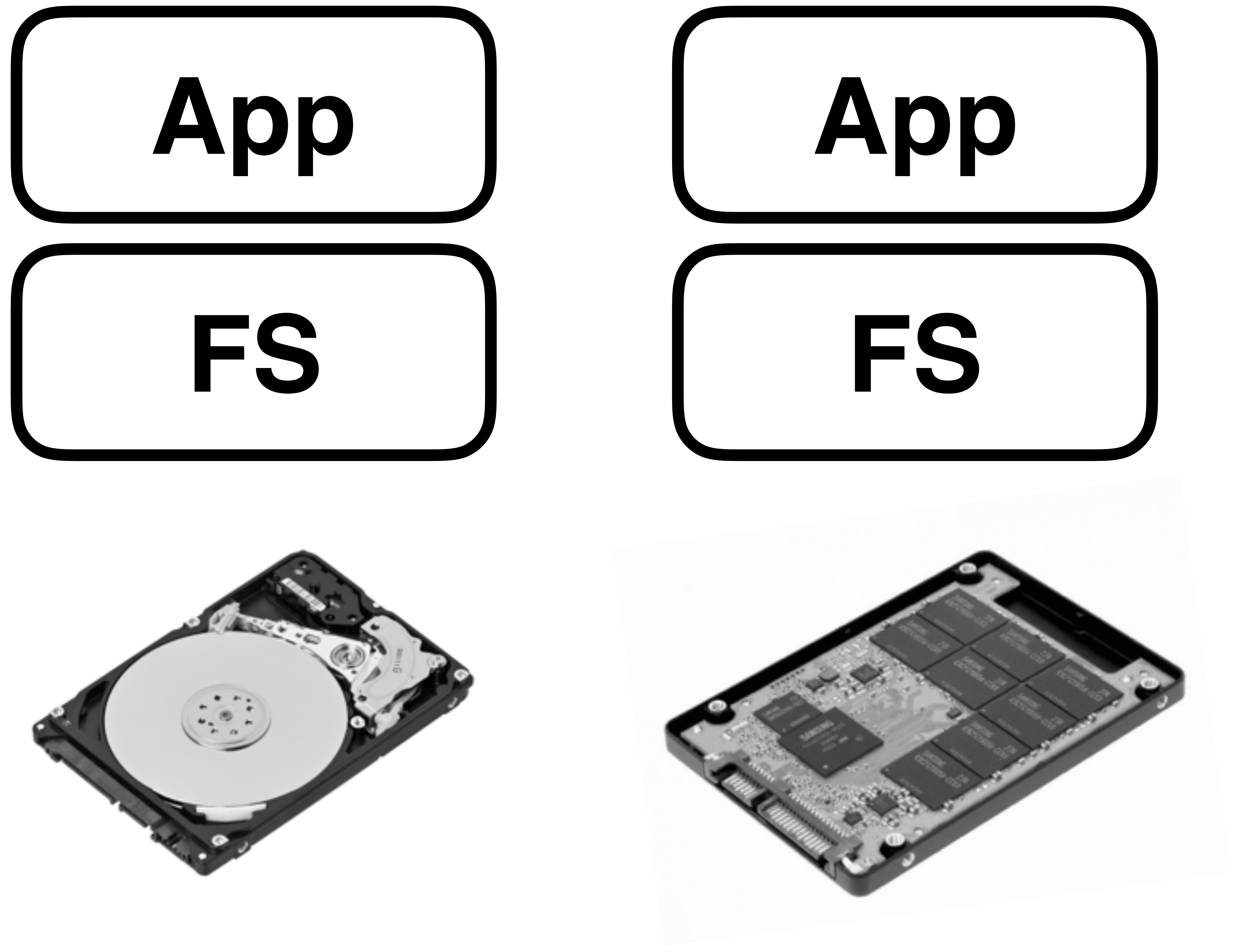
Storage stack is shifting from the HDD era to the SSD era



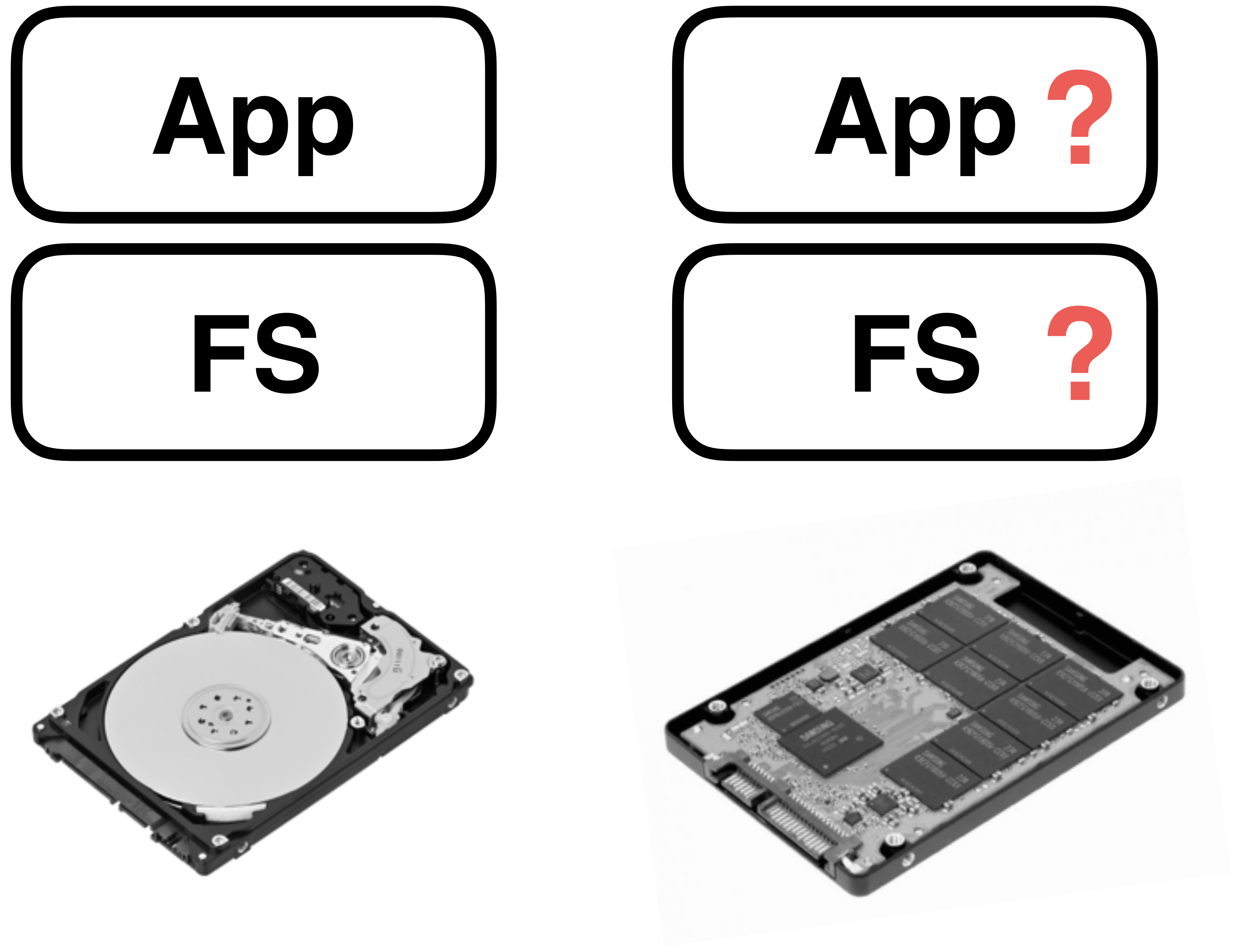
Storage stack is shifting from the HDD era to the SSD era



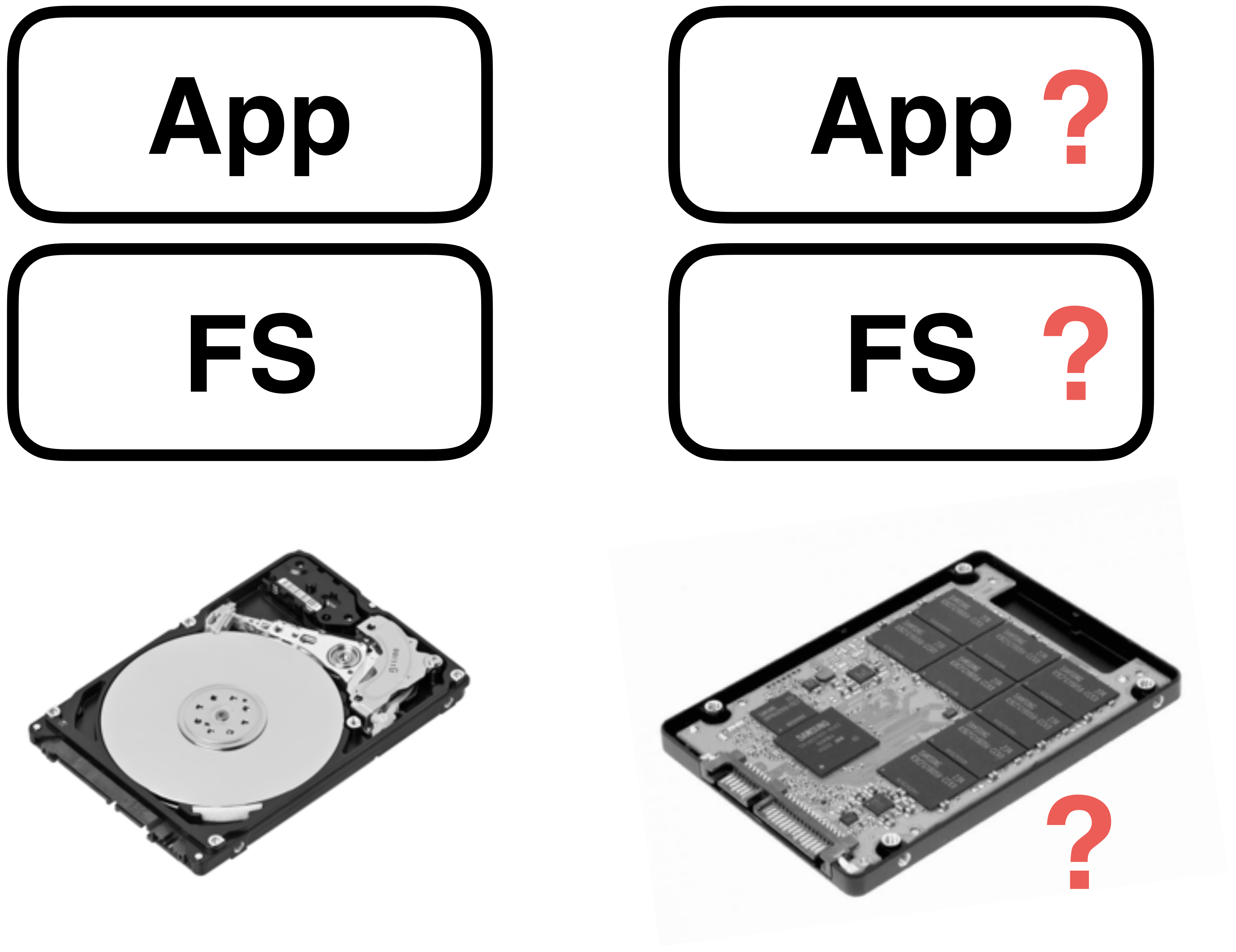
Storage stack is shifting from the HDD era to the SSD era



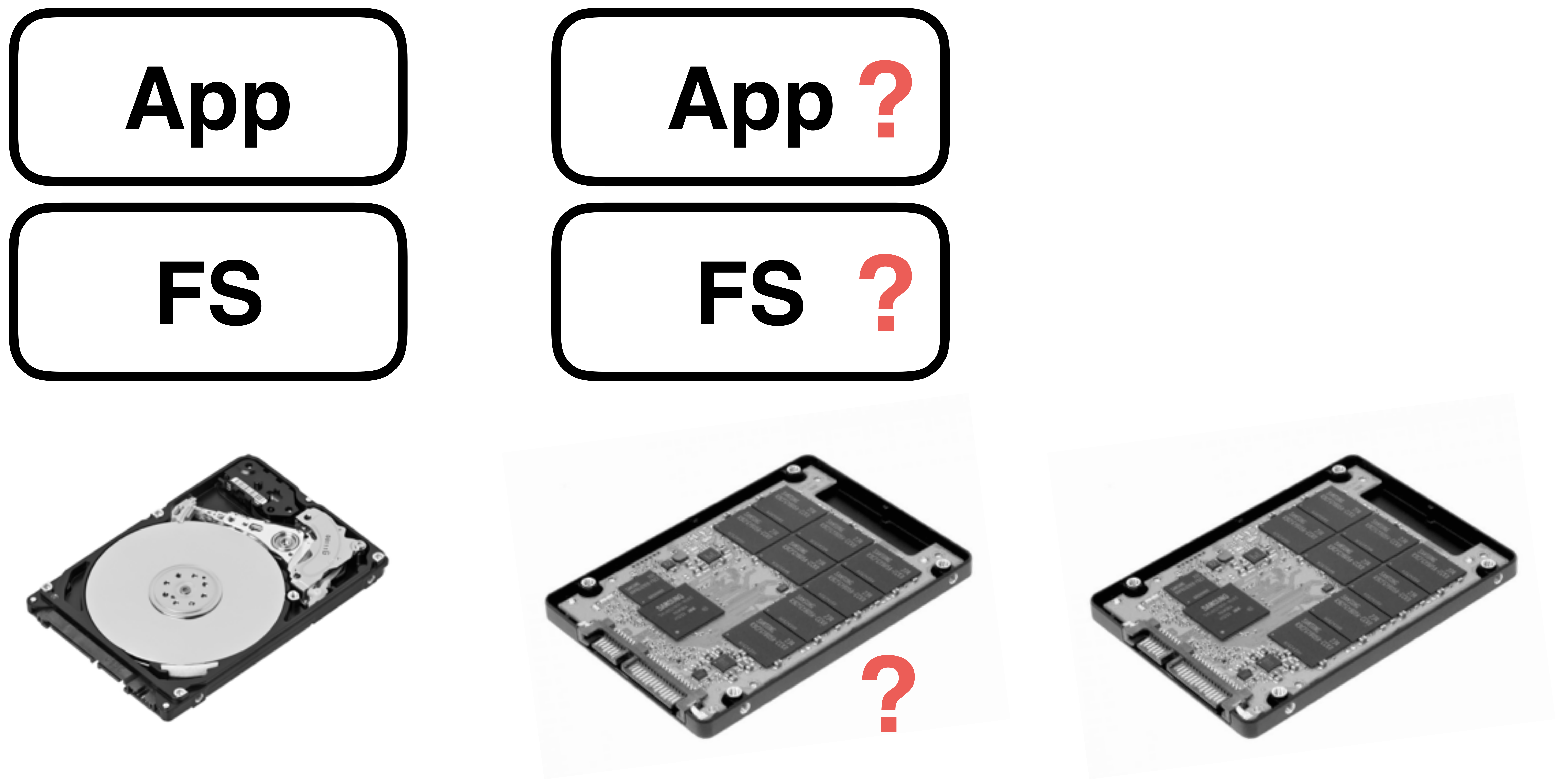
Storage stack is shifting from the HDD era to the SSD era



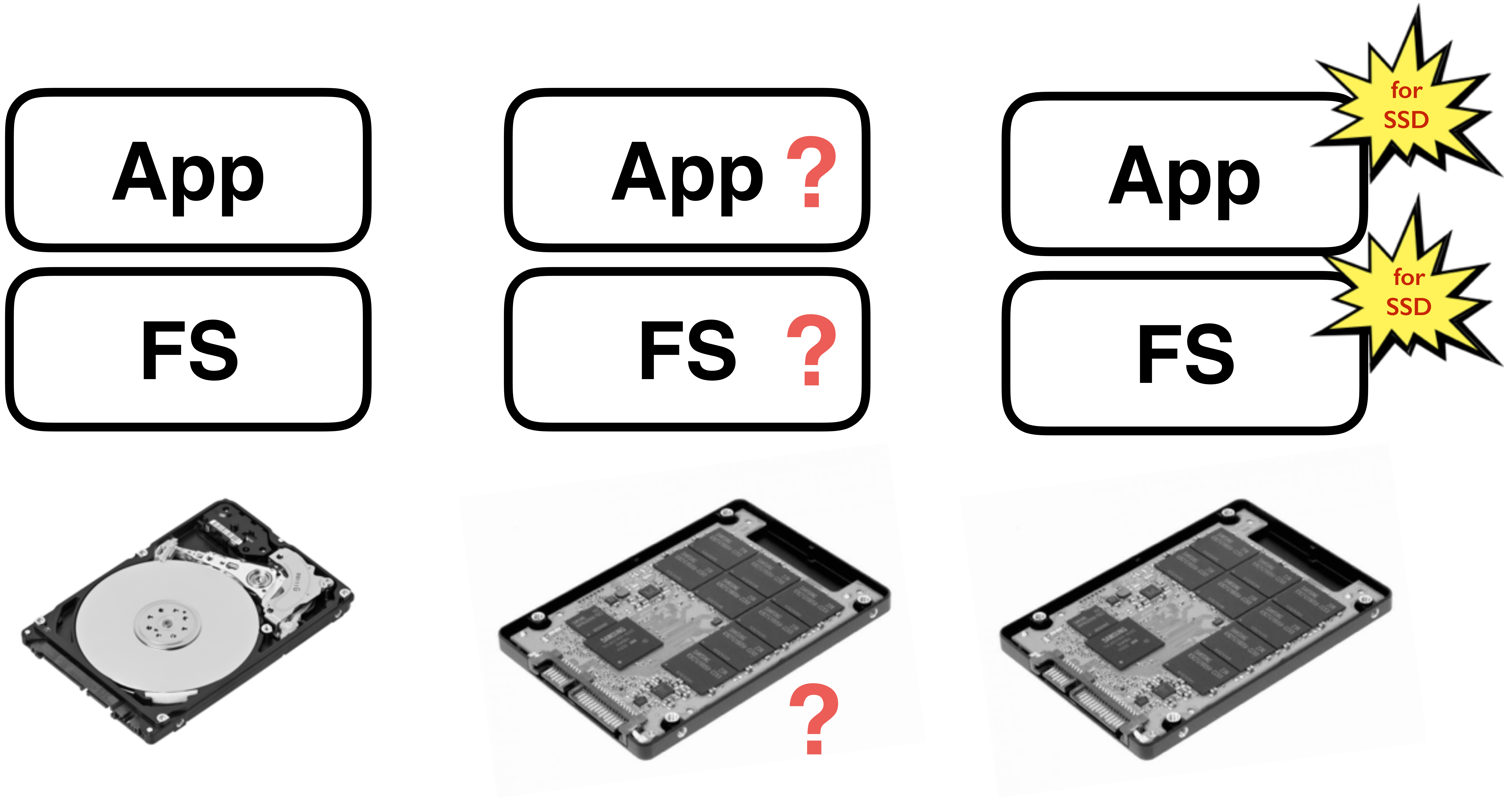
Storage stack is shifting from the HDD era to the SSD era



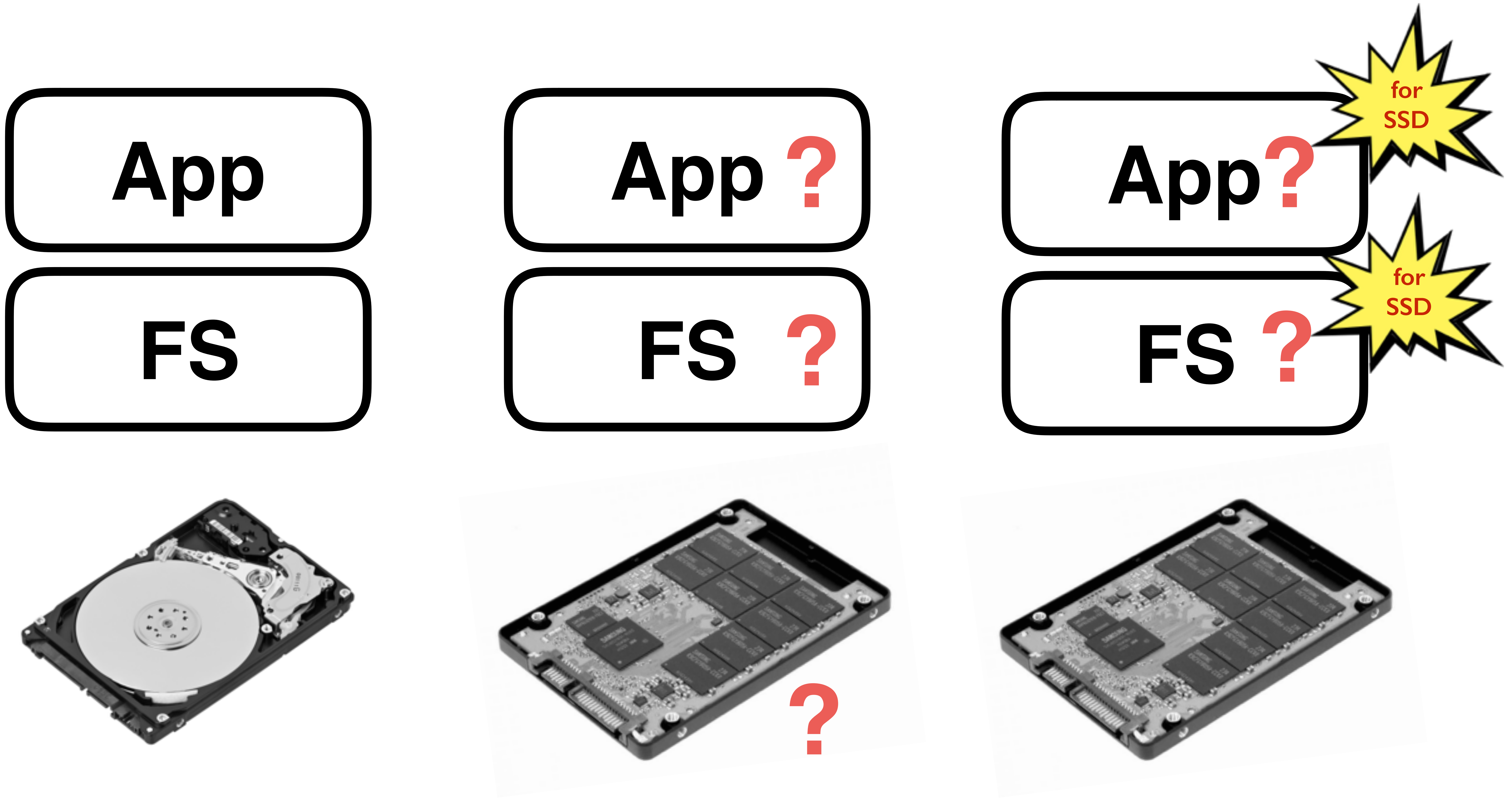
Storage stack is shifting from the HDD era to the SSD era



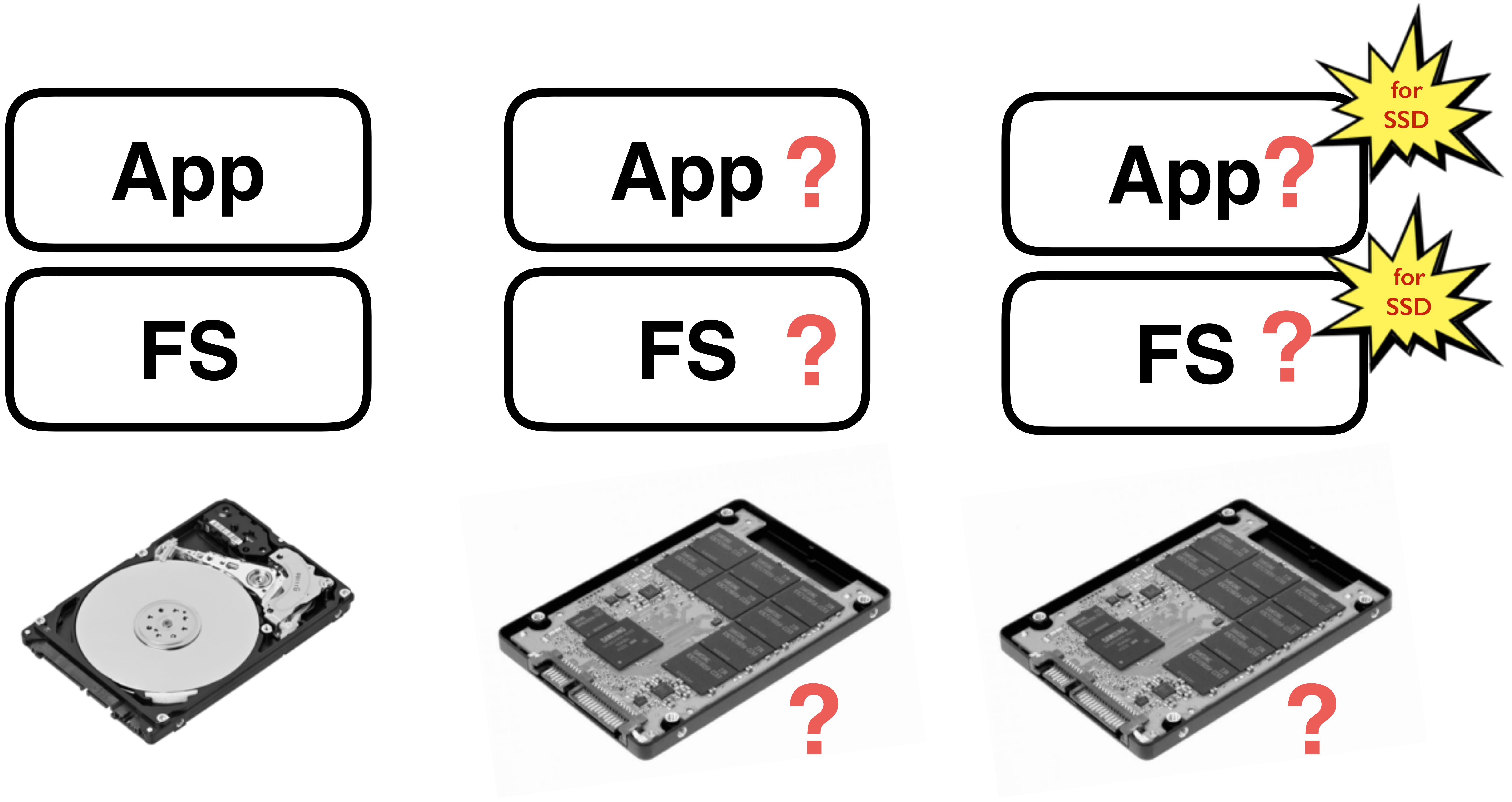
Storage stack is shifting from the HDD era to the SSD era



Storage stack is shifting from the HDD era to the SSD era



Storage stack is shifting from the HDD era to the SSD era



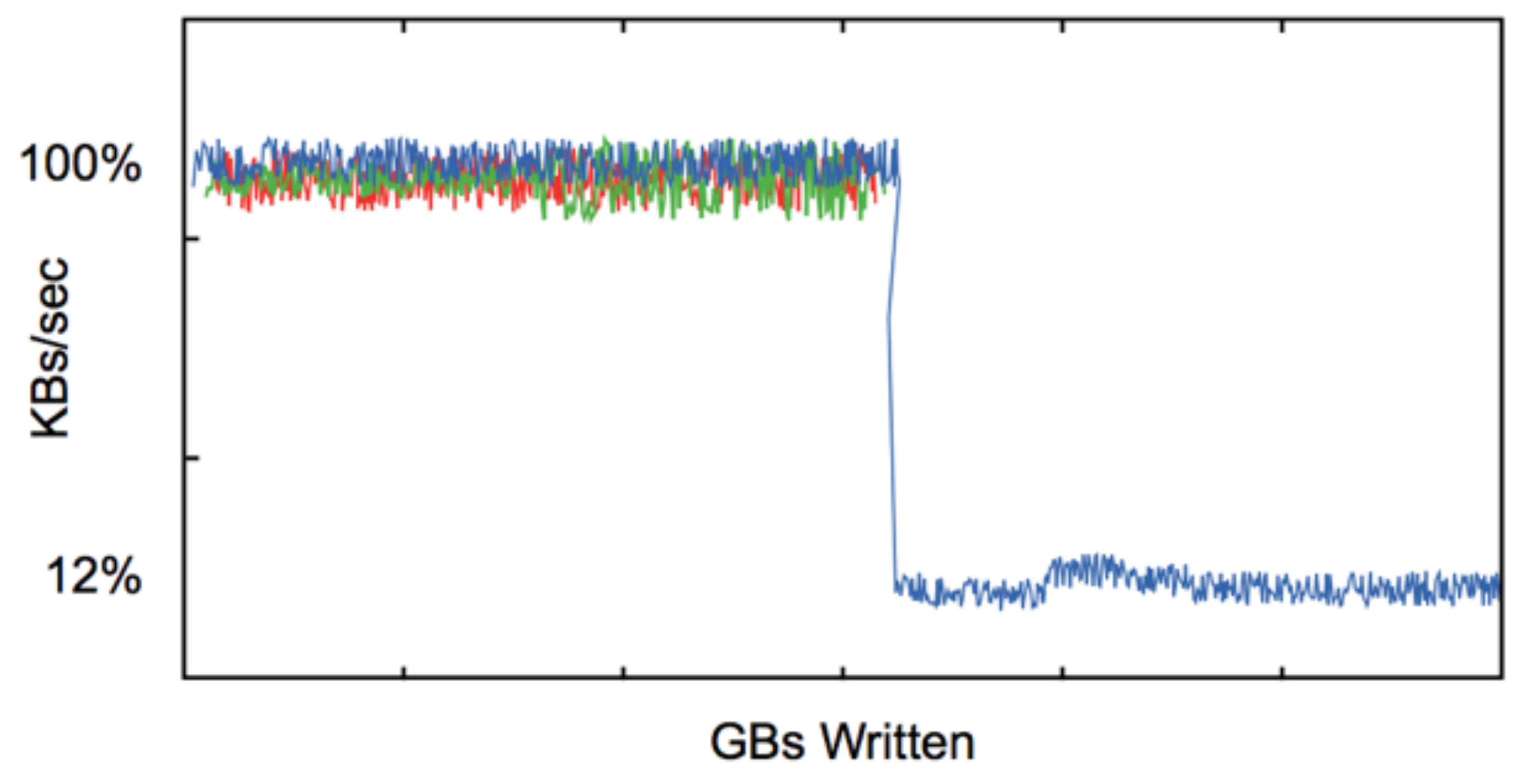
The consequences of misusing SSDs

http://crestingwave.com/sites/default/files/collateral/velobit_whitepaper_ssdperformancetips.pdf

S. Boboila and P. Desnoyers. Write Endurance in Flash Drives: Measurements and Analysis. In Proceedings of the 8th USENIX Symposium on File and Storage Technologies (FAST '10), San Jose, California, February 2010

The consequences of misusing SSDs

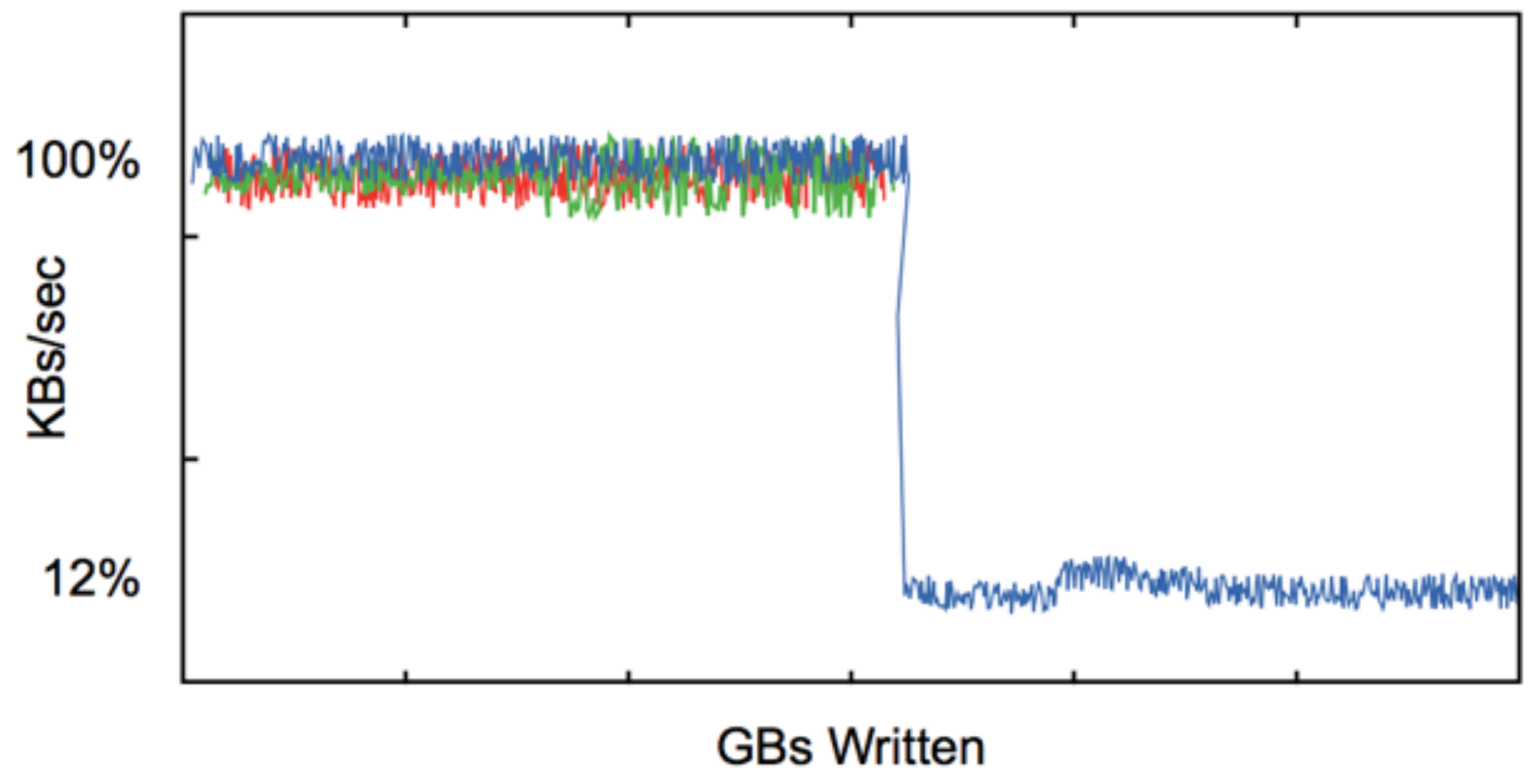
Performance degradation



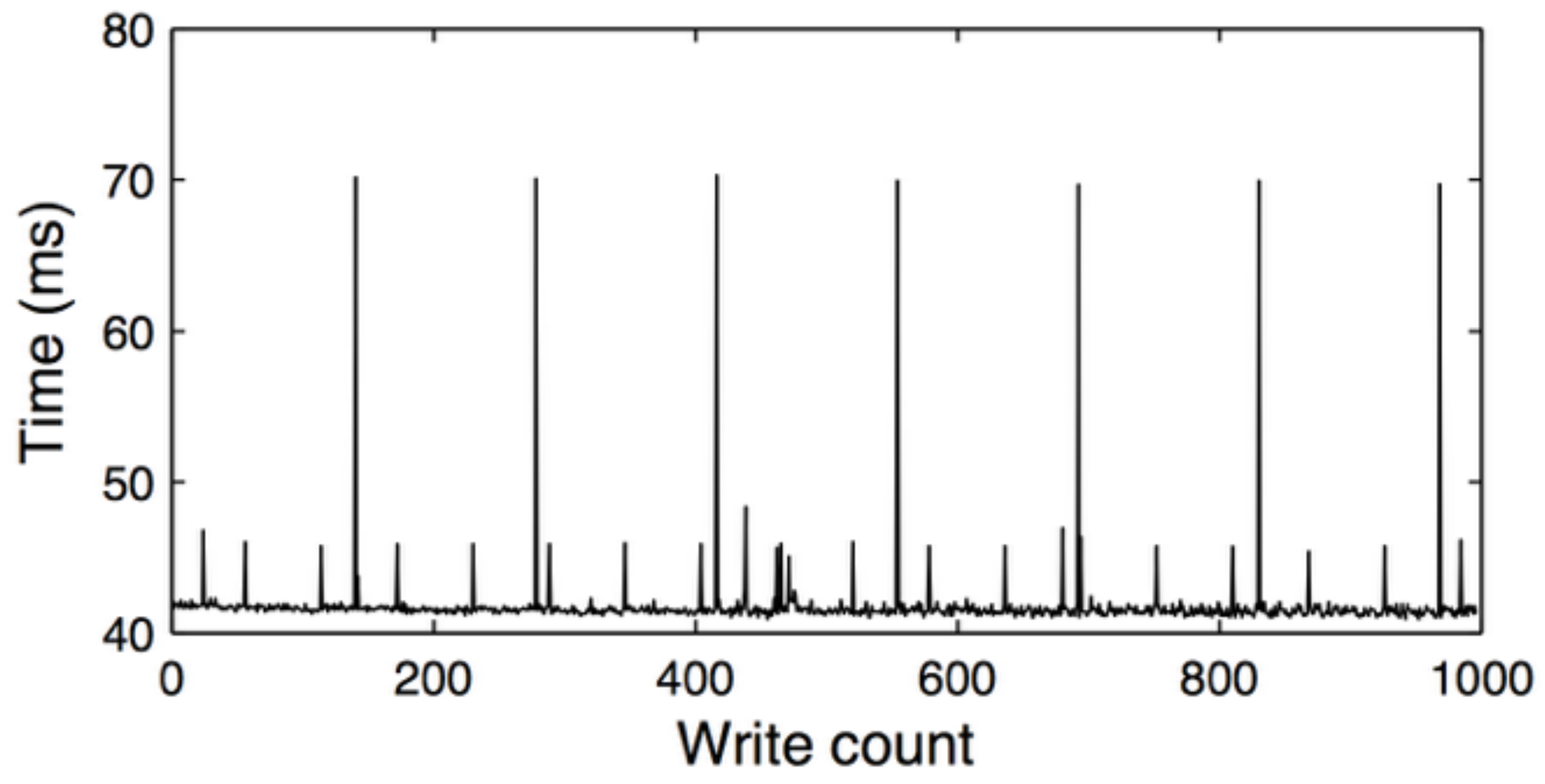
http://crestingwave.com/sites/default/files/collateral/velobit_whitepaper_ssdperformancetips.pdf
S. Boboila and P. Desnoyers. Write Endurance in Flash Drives: Measurements and Analysis. In Proceedings of the 8th USENIX Symposium on File and Storage Technologies (FAST '10), San Jose, California, February 2010

The consequences of misusing SSDs

Performance degradation

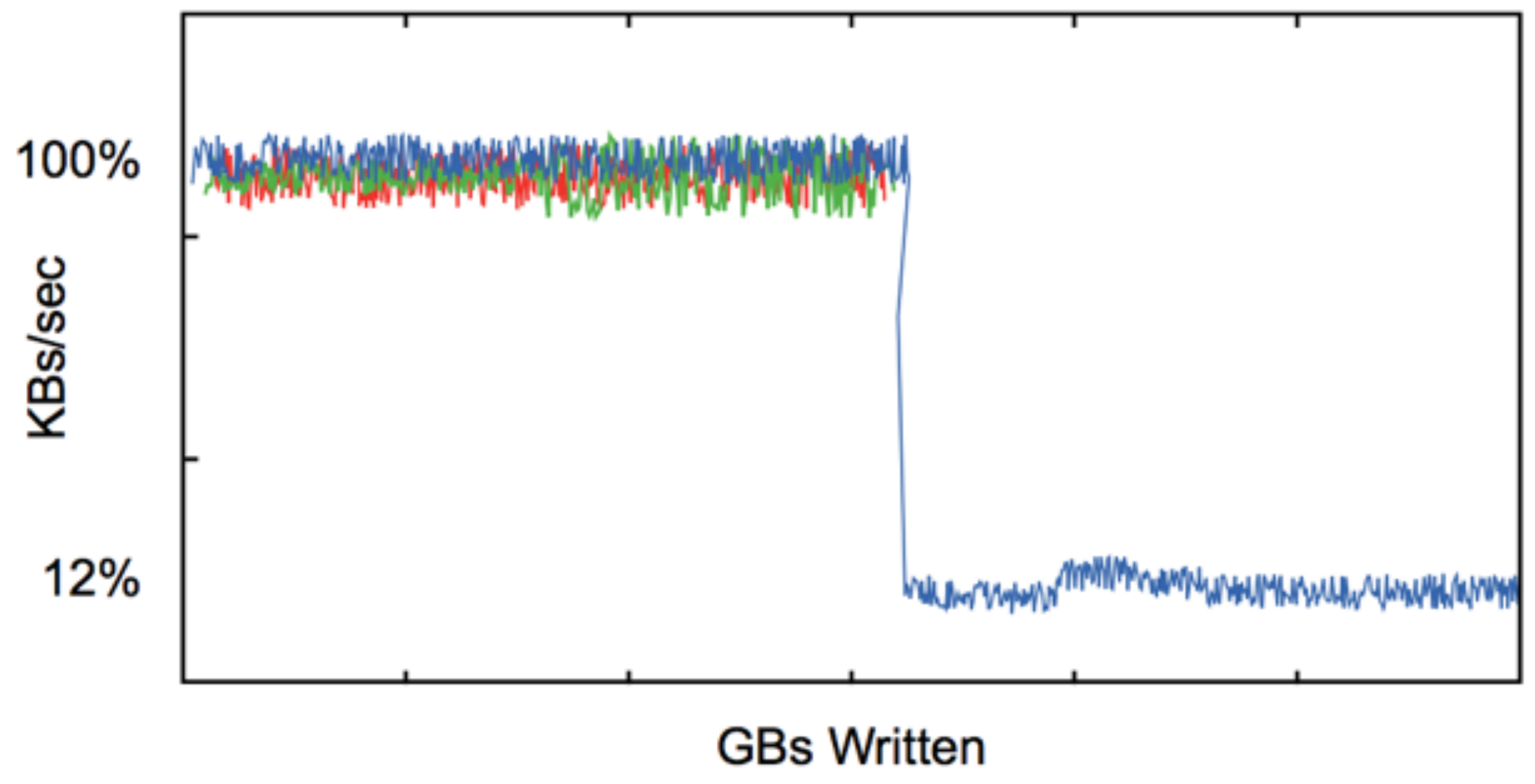


Performance fluctuation

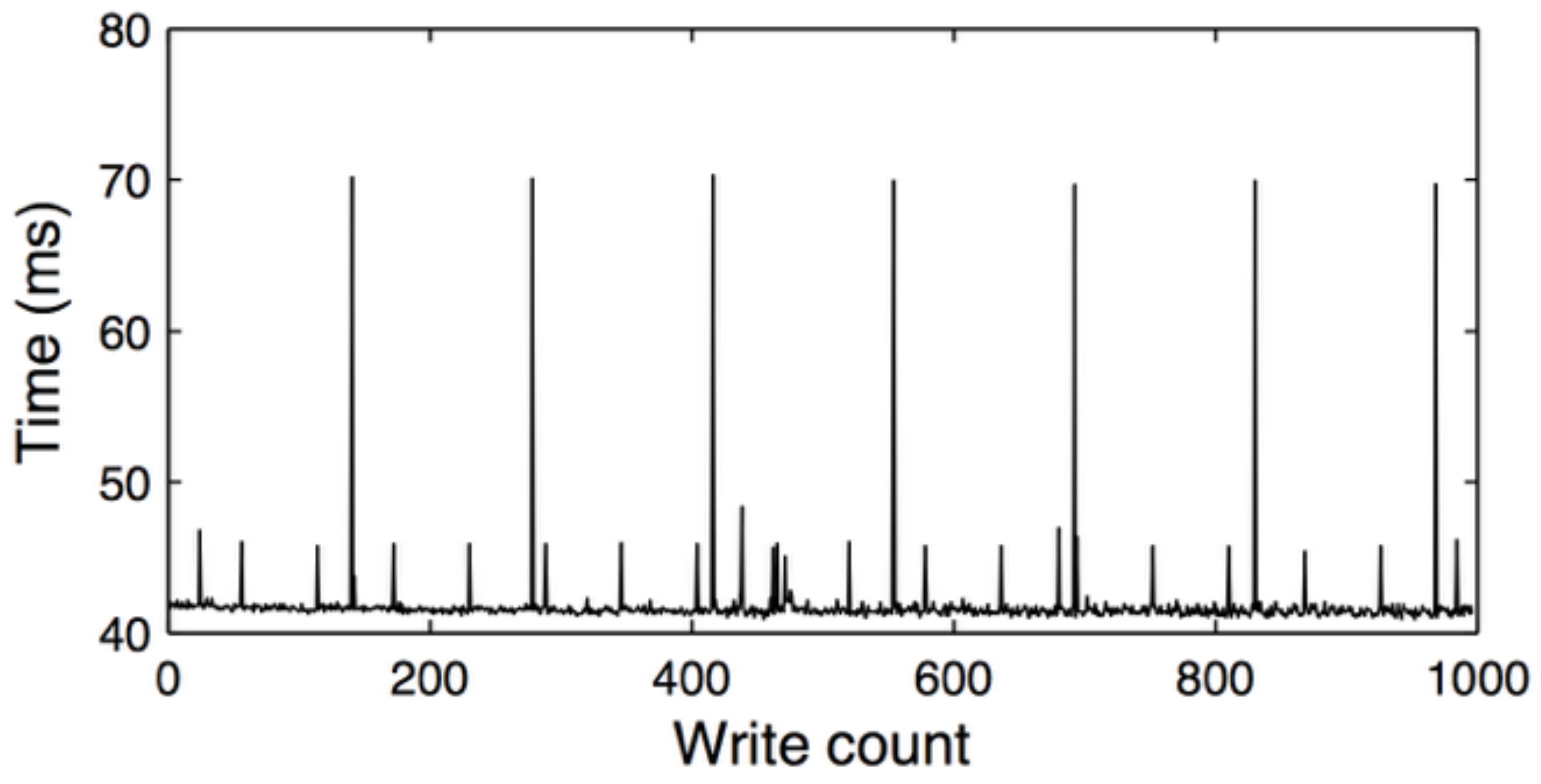


The consequences of misusing SSDs

Performance degradation



Performance fluctuation



Early end of device life

http://crestingwave.com/sites/default/files/collateral/velobit_whitepaper_ssdperformancetips.pdf
S. Boboila and P. Desnoyers. Write Endurance in Flash Drives: Measurements and Analysis. In Proceedings of the 8th USENIX Symposium on File and Storage Technologies (FAST '10), San Jose, California, February 2010

What is the right way to achieve high performance on SSDs?

What is the right way to achieve high performance on SSDs?

Block Device Interface: ***read(range), write(range), discard(range)***

What is the right way to achieve high performance on SSDs?

Block Device Interface: *read(range), write(range), discard(range)*

Unwritten Contract of HDDs

- **Sequential accesses are best**
- **Nearby accesses are more efficient than farther ones**

MEMS-based storage devices and standard disk interfaces: A square peg in a round hole?
Steven W. Schlosser, Gregory R. Ganger
FAST'04

What is the right way to achieve high performance on SSDs?

Block Device Interface: ***read(range), write(range), discard(range)***

Unwritten Contract of HDDs

- **Sequential accesses are best**
- **Nearby accesses are more efficient than farther ones**

MEMS-based storage devices and standard disk interfaces: A square peg in a round hole?
Steven W. Schlosser, Gregory R. Ganger
FAST'04

Unwritten Contract of SSDs

?

What is the right way to achieve high performance on SSDs?

What is the right way to achieve high performance on SSDs?

- Existing studies

What is the right way to achieve high performance on SSDs?

- Existing studies
- Experience of implementing a detailed SSD simulator

What is the right way to achieve high performance on SSDs?

- Existing studies
- Experience of implementing a detailed SSD simulator
- Analysis of experiments

What is the right way to achieve high performance on SSDs?

- Existing studies
- Experience of implementing a detailed SSD simulator
- Analysis of experiments

$\Downarrow$

The Unwritten Contract of SSDs

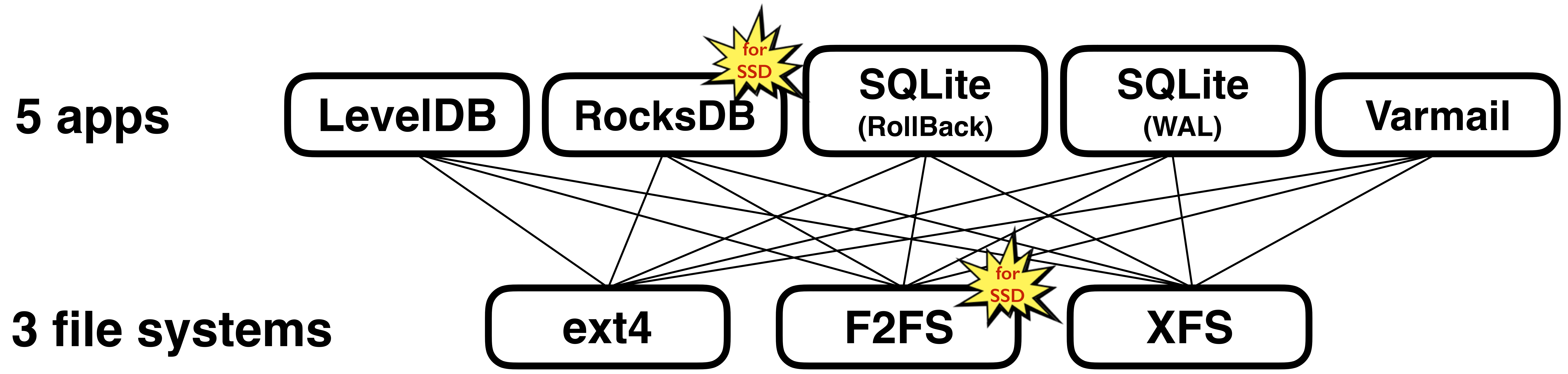
Do the current apps/FSes comply with the unwritten contract of SSDs?

Do the current apps/FSes comply with the unwritten contract of SSDs?

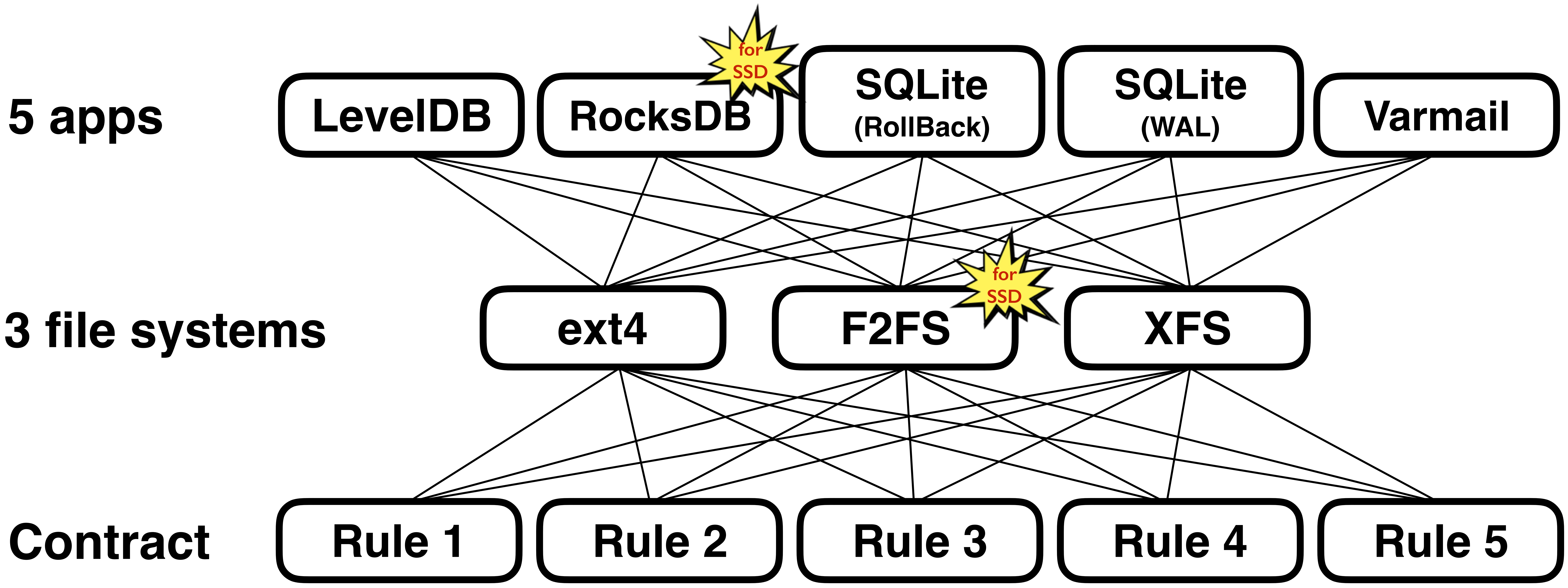
5 apps

LevelDB | RocksDB (for SSD) | SQLite (RollBack) | SQLite (WAL) | Varmail

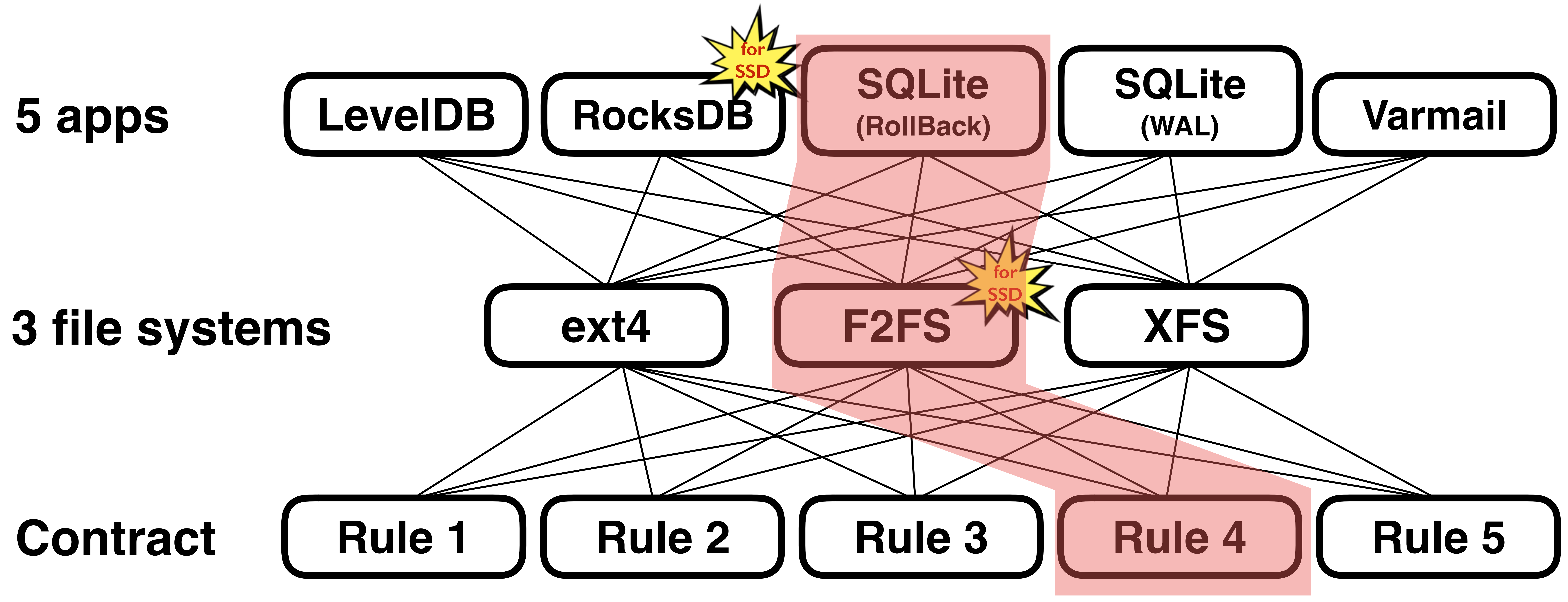
Do the current apps/FSes comply with the unwritten contract of SSDs?



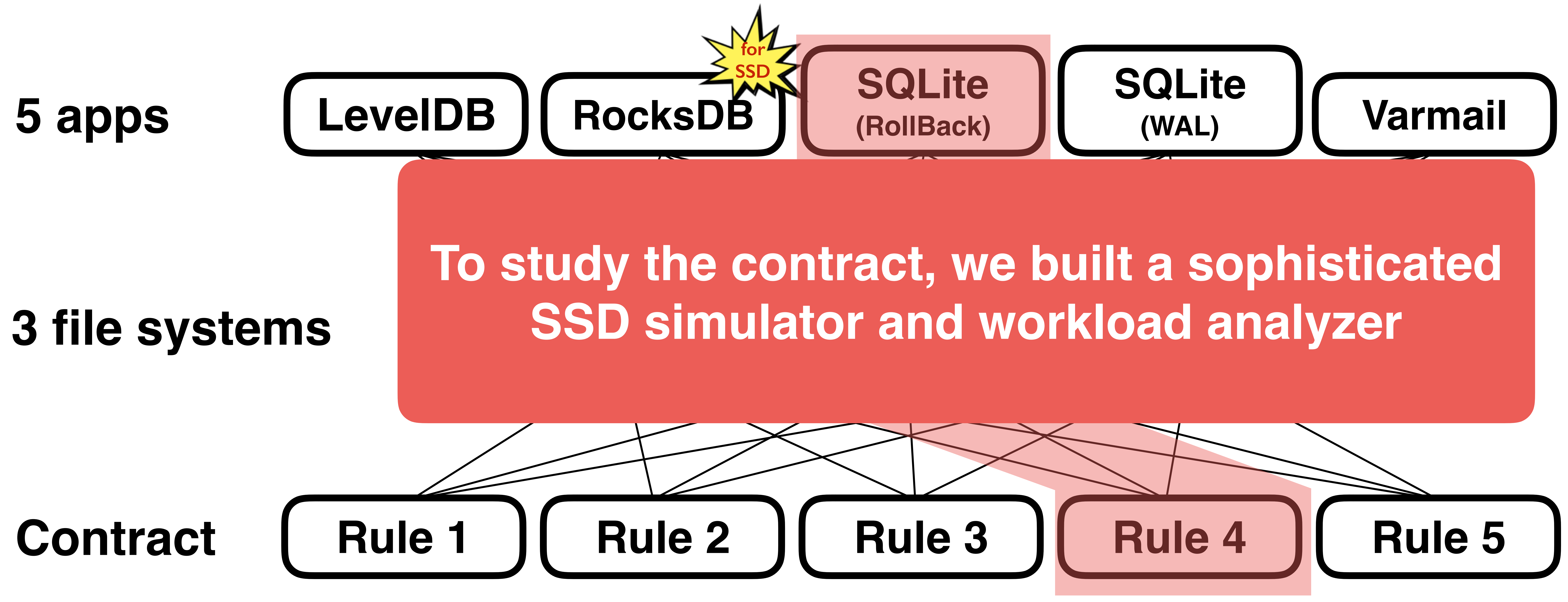
Do the current apps/FSes comply with the unwritten contract of SSDs?



Do the current apps/FSes comply with the unwritten contract of SSDs?



Do the current apps/FSes comply with the unwritten contract of SSDs?



To study the contract, we built a sophisticated SSD simulator and workload analyzer

In the paper

In the paper

We made 24 detailed observations

In the paper

We made 24 detailed observations

We learned several high-level lessons

Outline

Outline

Overview

SSD Unwritten Contract

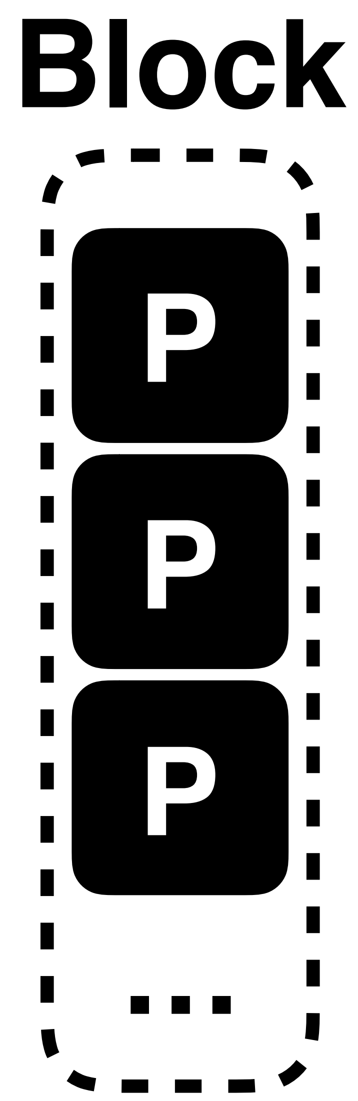
Violations of the Unwritten Contract

Conclusions

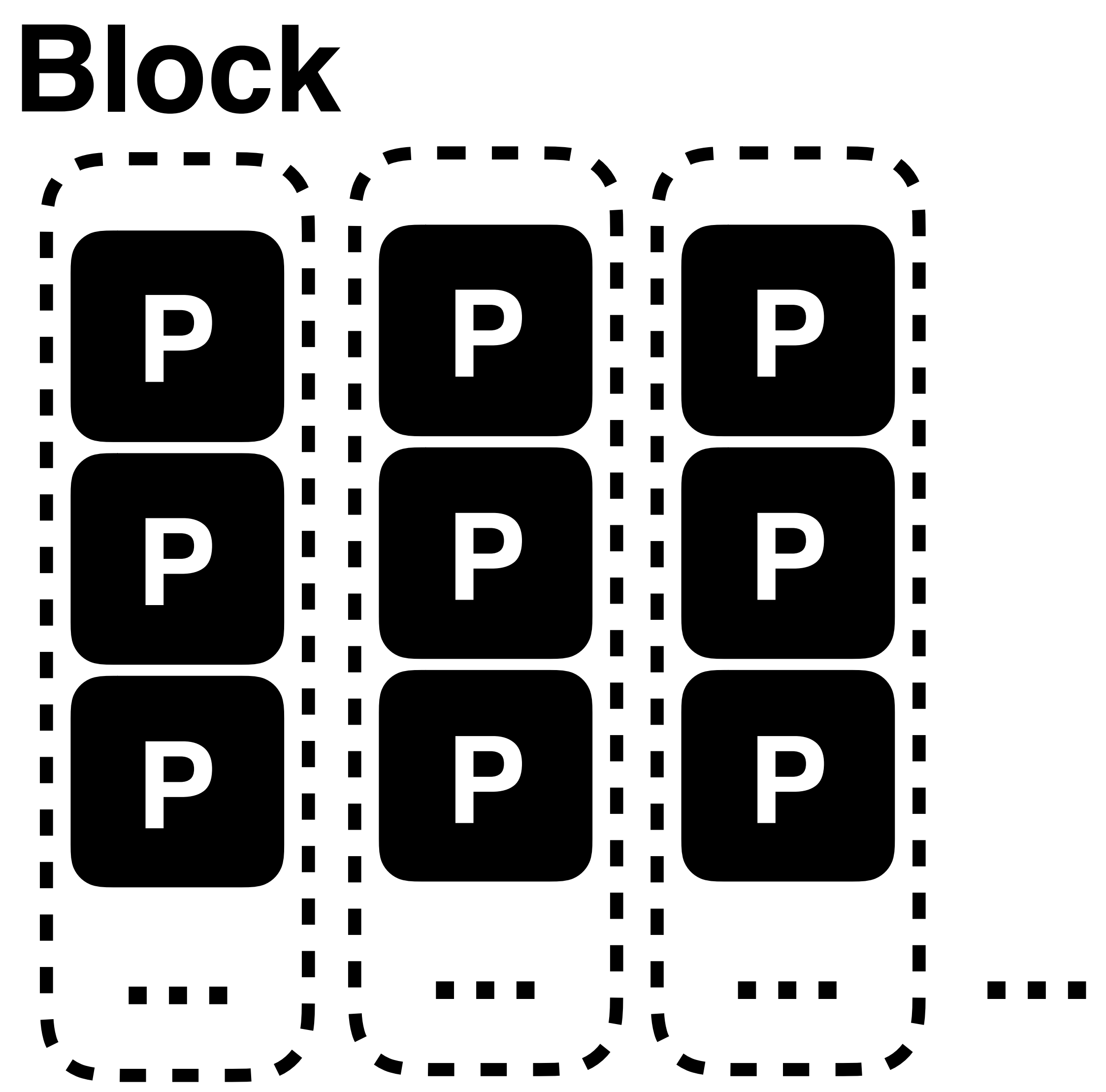
SSD Background

SSD Background

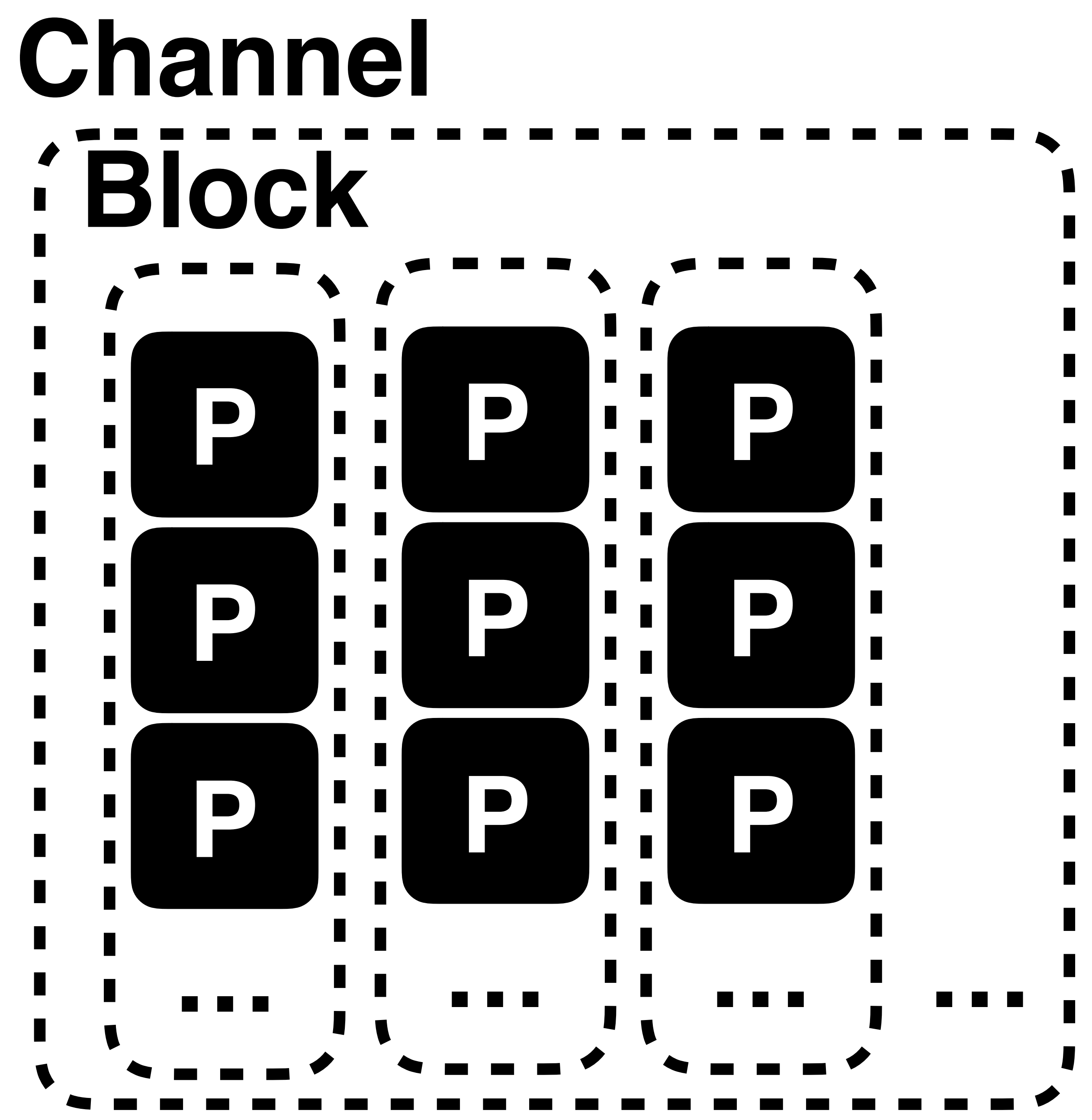
SSD Background



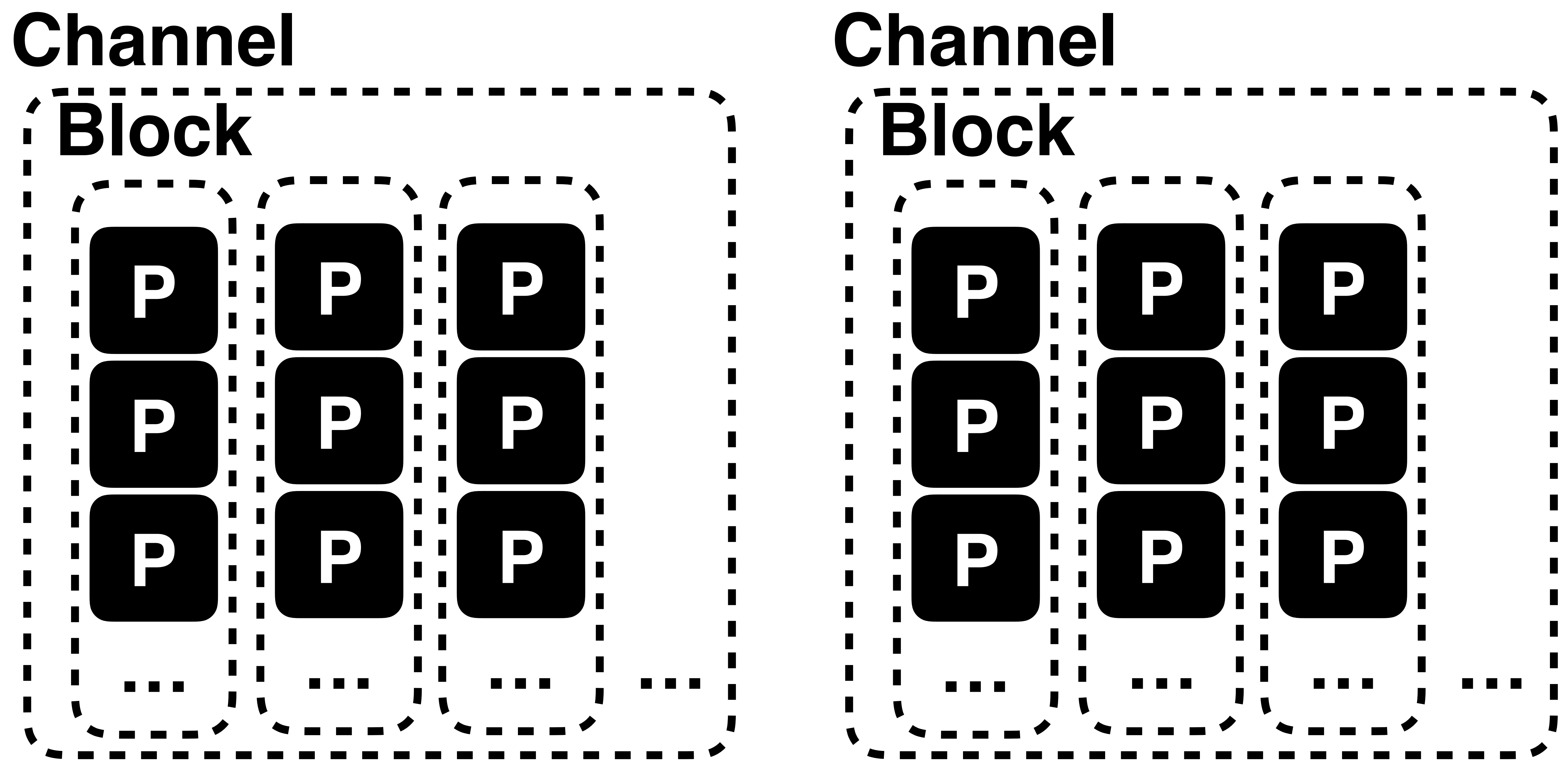
SSD Background



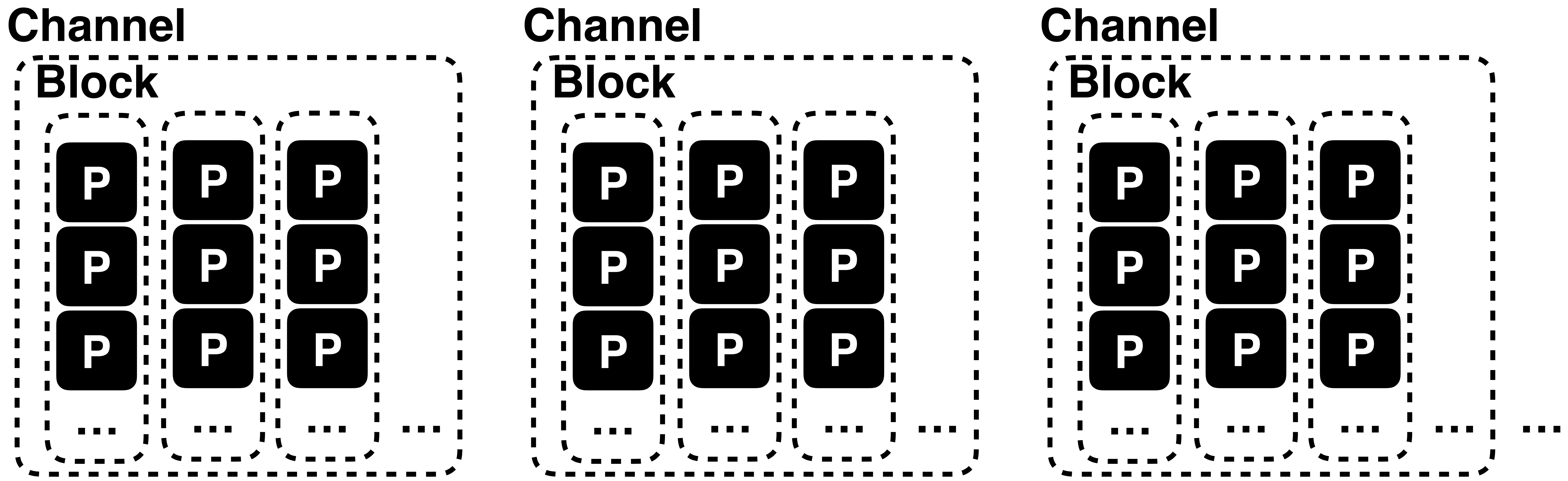
SSD Background



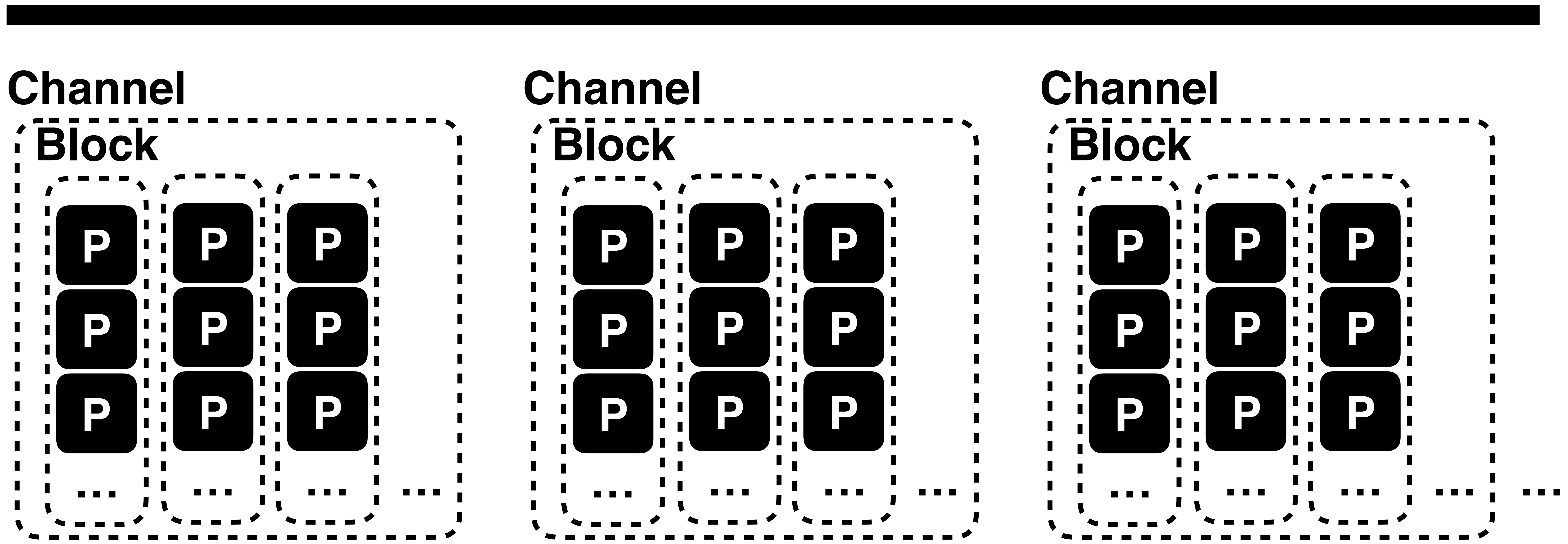
SSD Background



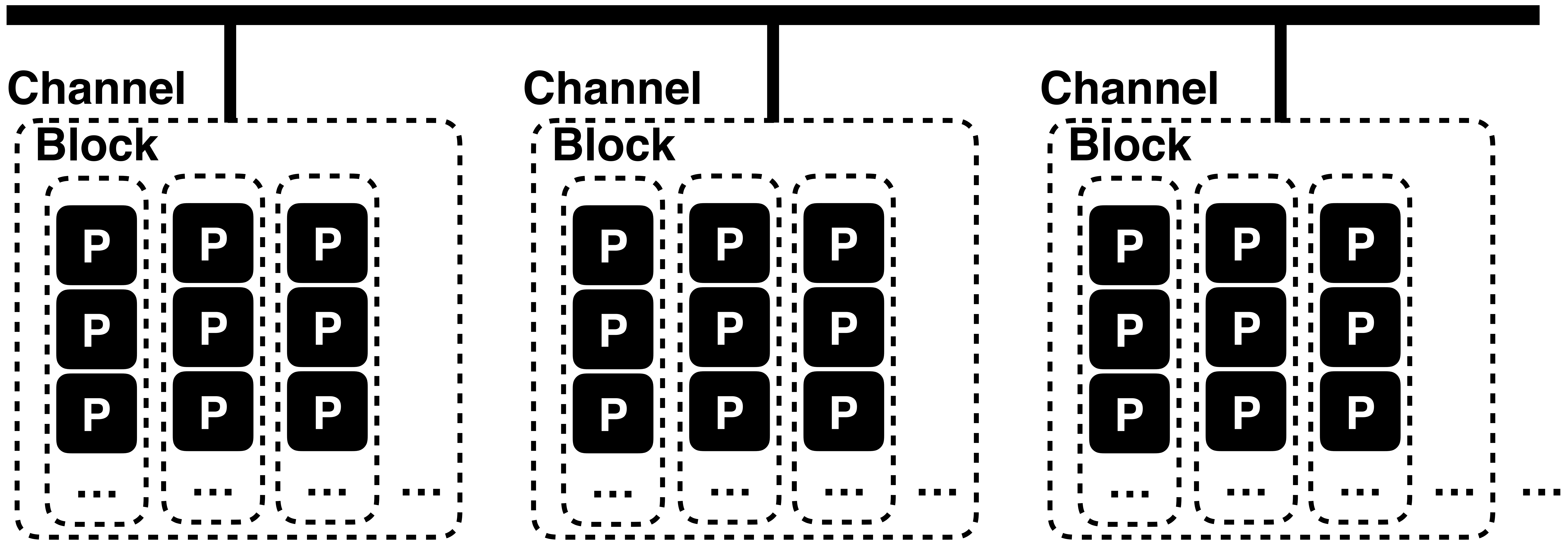
SSD Background



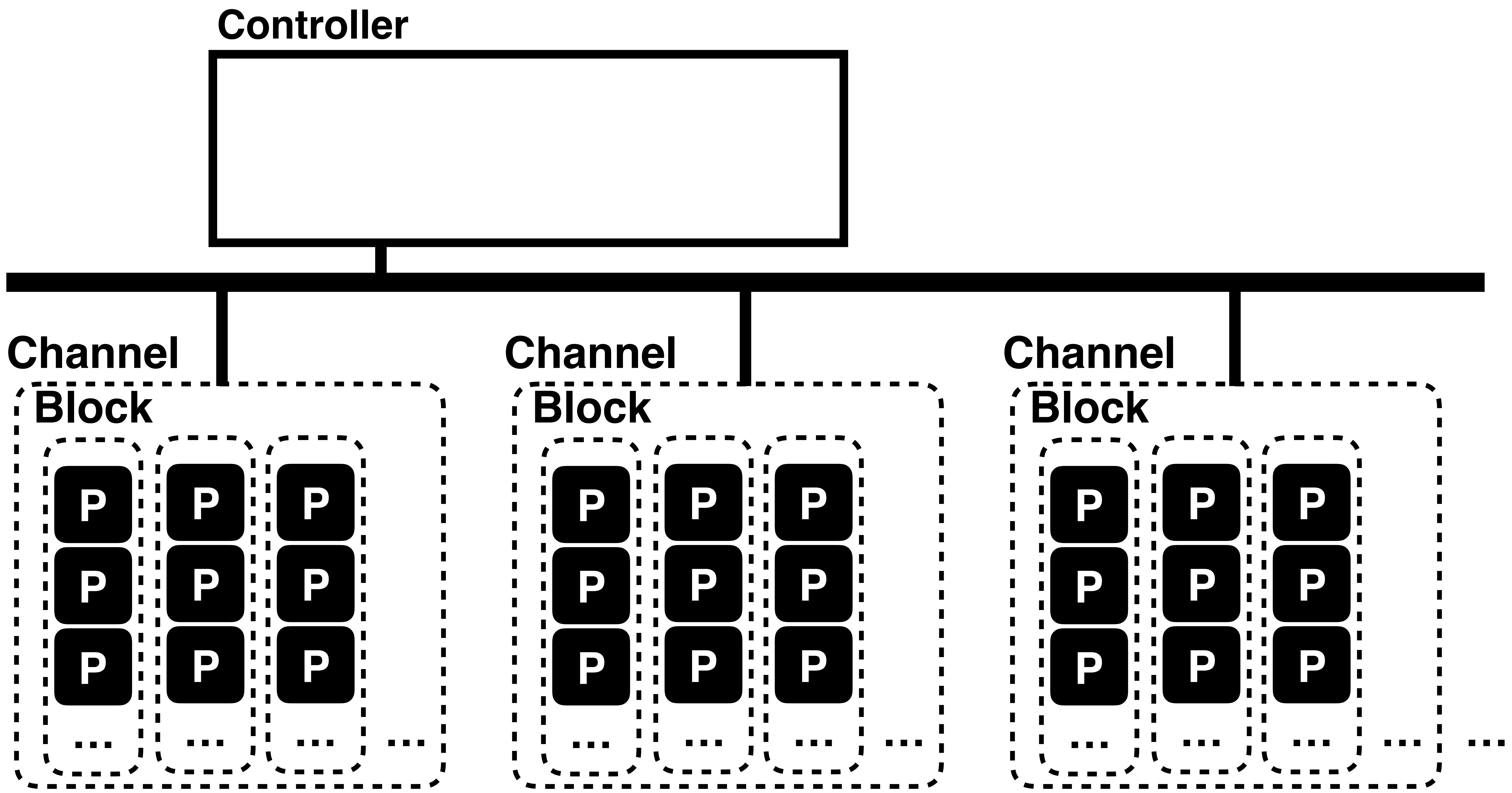
SSD Background



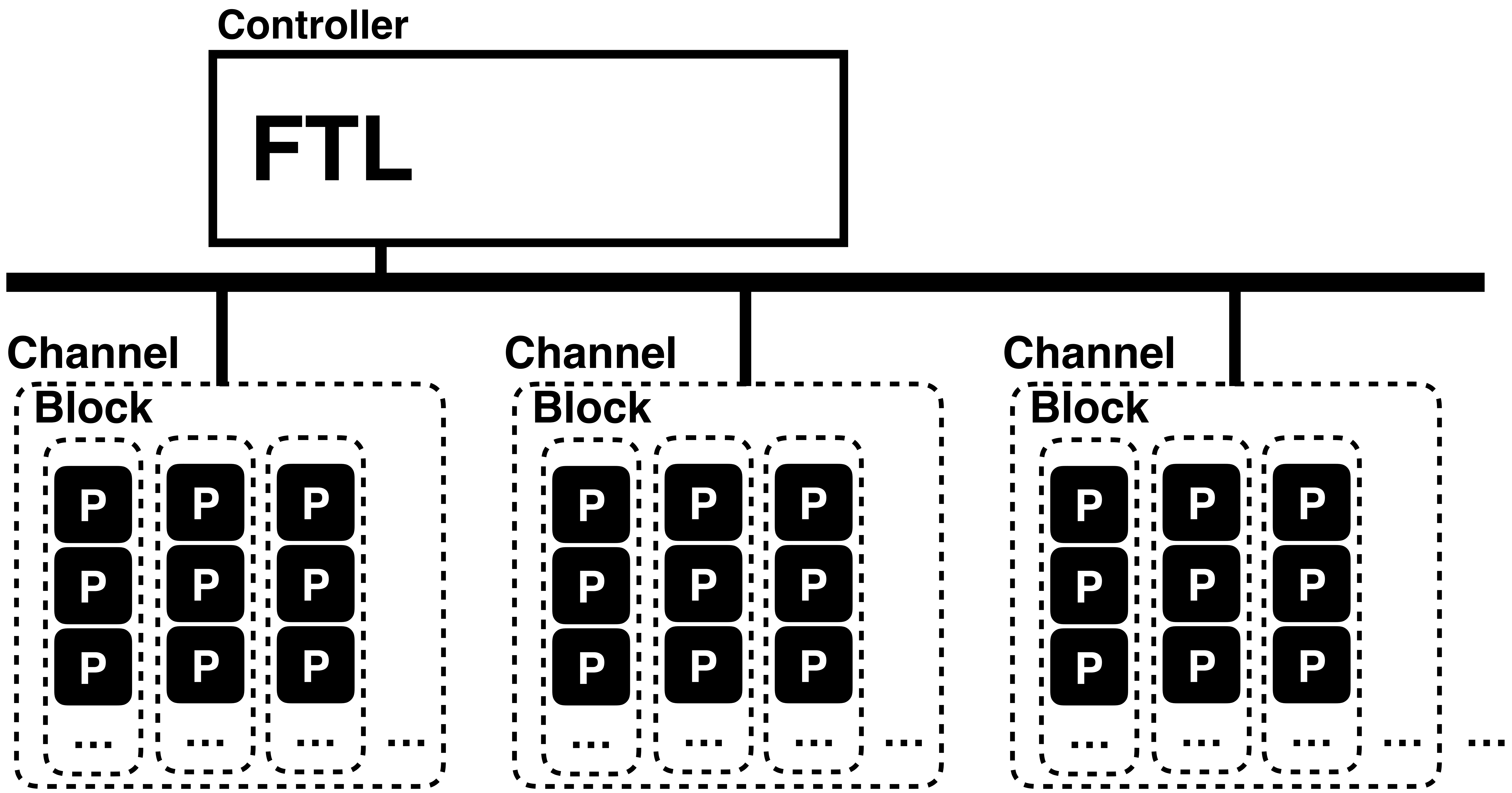
SSD Background



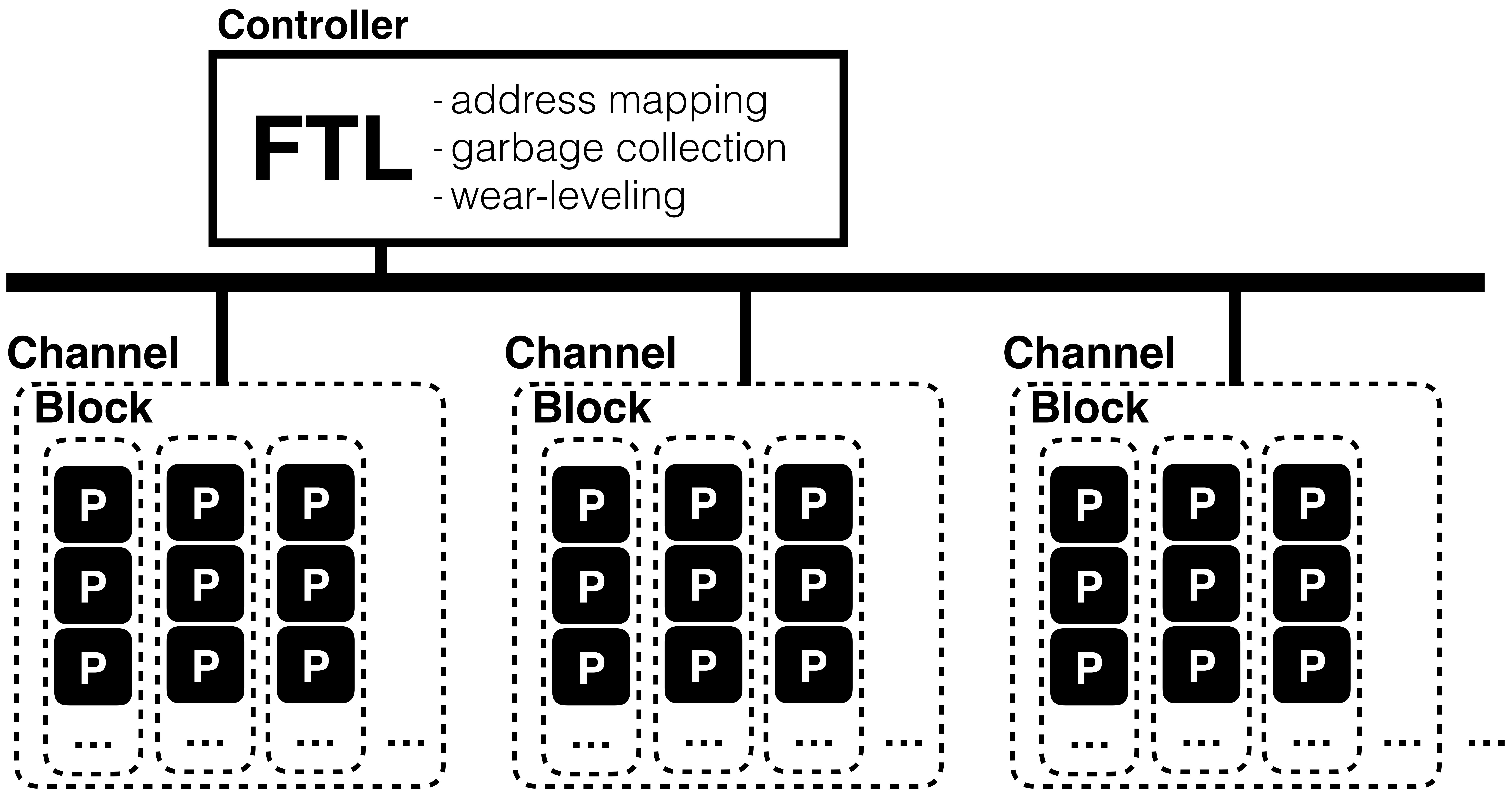
SSD Background



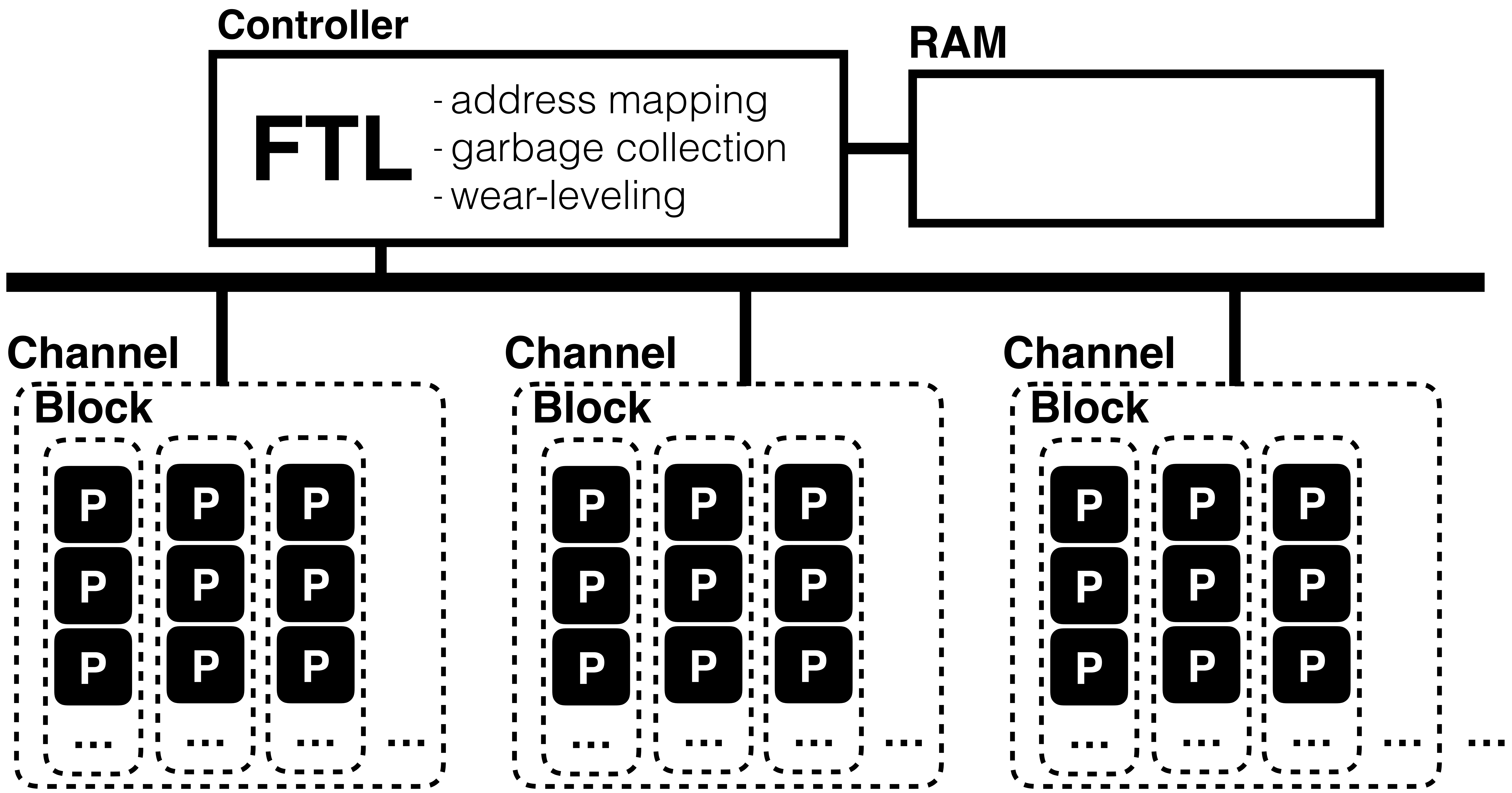
SSD Background



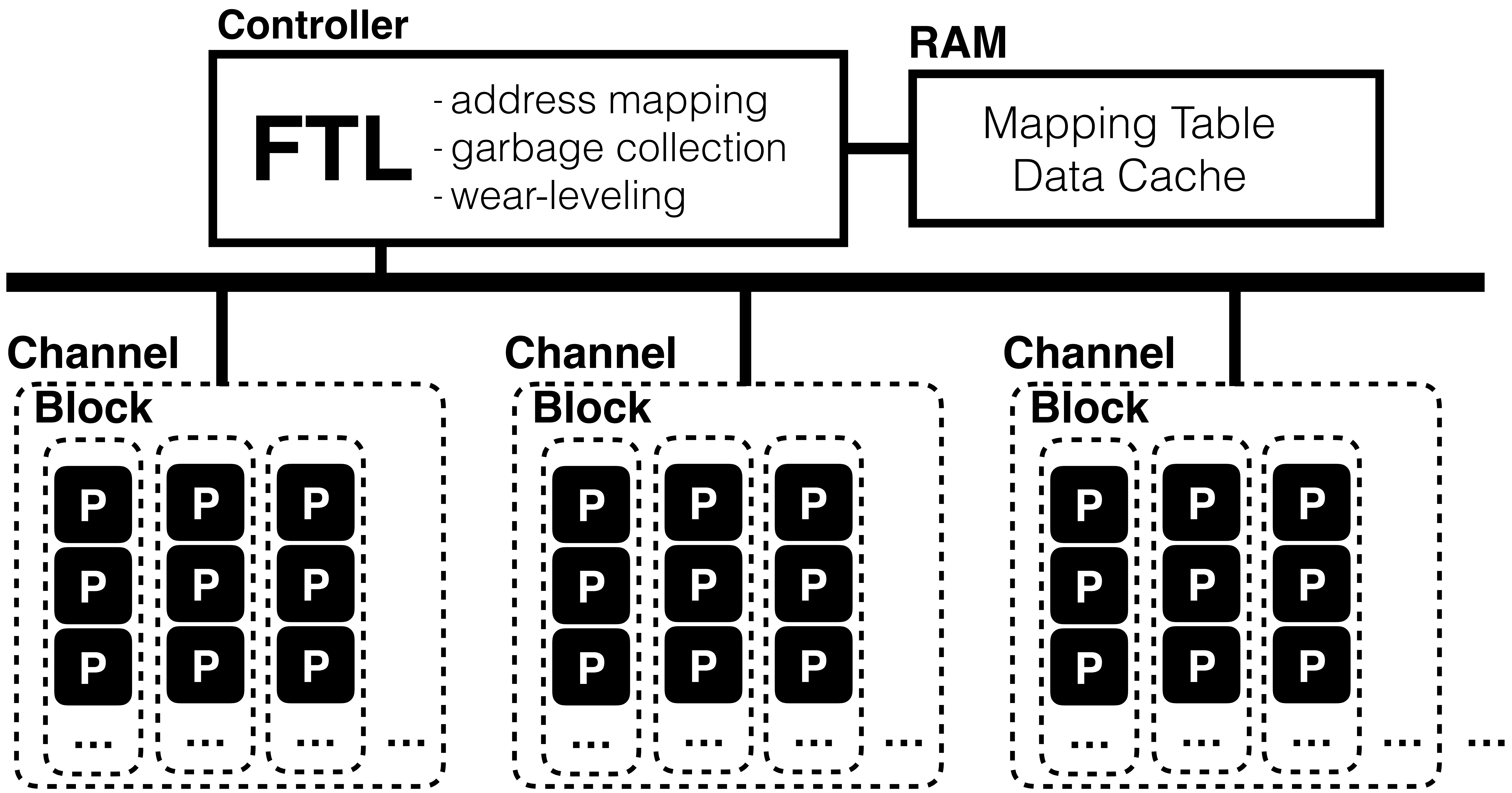
SSD Background



SSD Background



SSD Background



Rules of the Unwritten Contract

#1 Request Scale

#2 Locality

#3 Aligned Sequentiality

#4 Grouping by Death Time

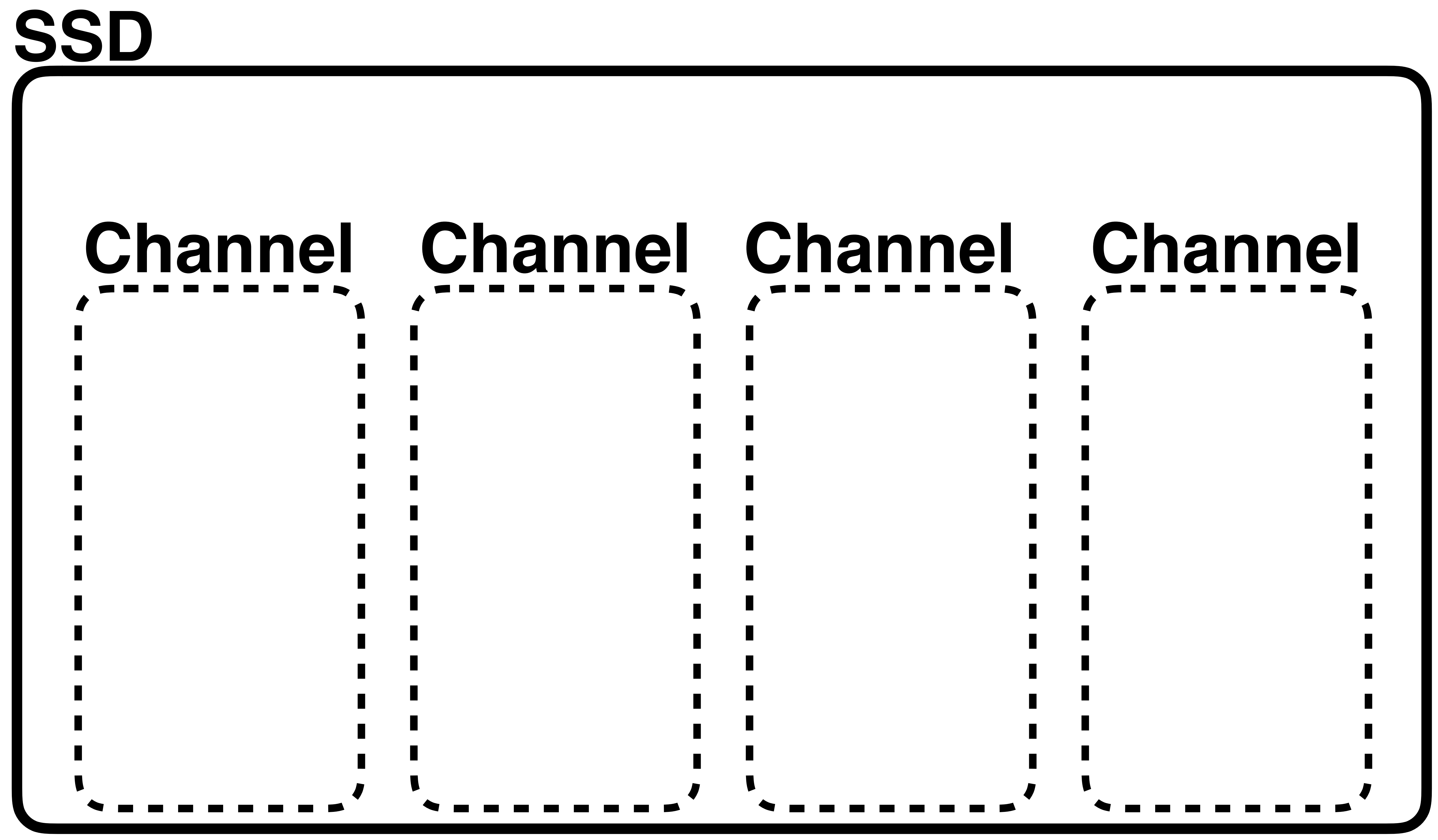
#5 Uniform Data Lifetime

Rule #1: Request Scale

SSD clients should issue **large** data requests or **multiple** outstanding data requests.

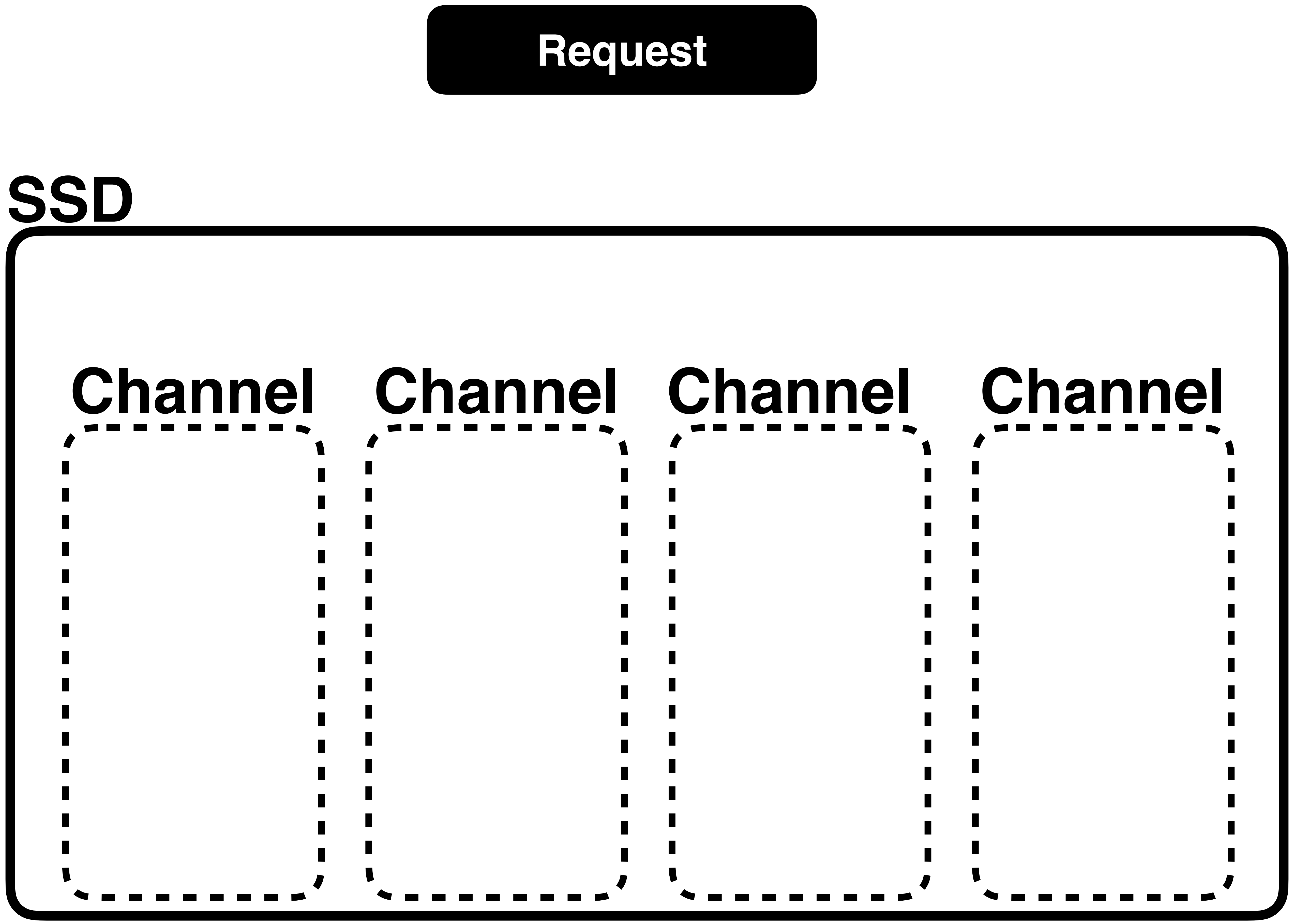
Rule #1: Request Scale

SSD clients should issue **large** data requests or **multiple** outstanding data requests.



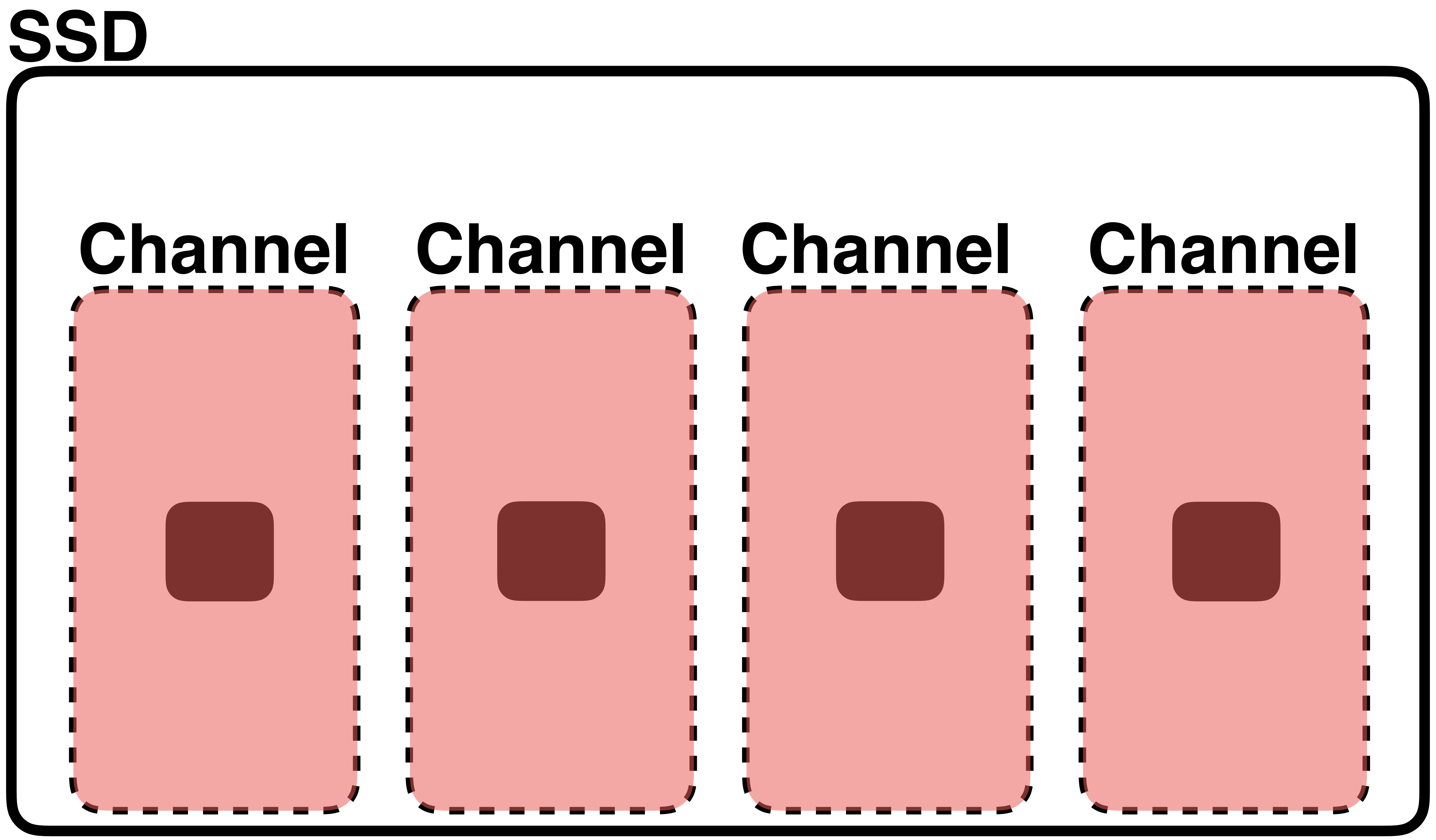
Rule #1: Request Scale

SSD clients should issue **large** data requests or **multiple** outstanding data requests.



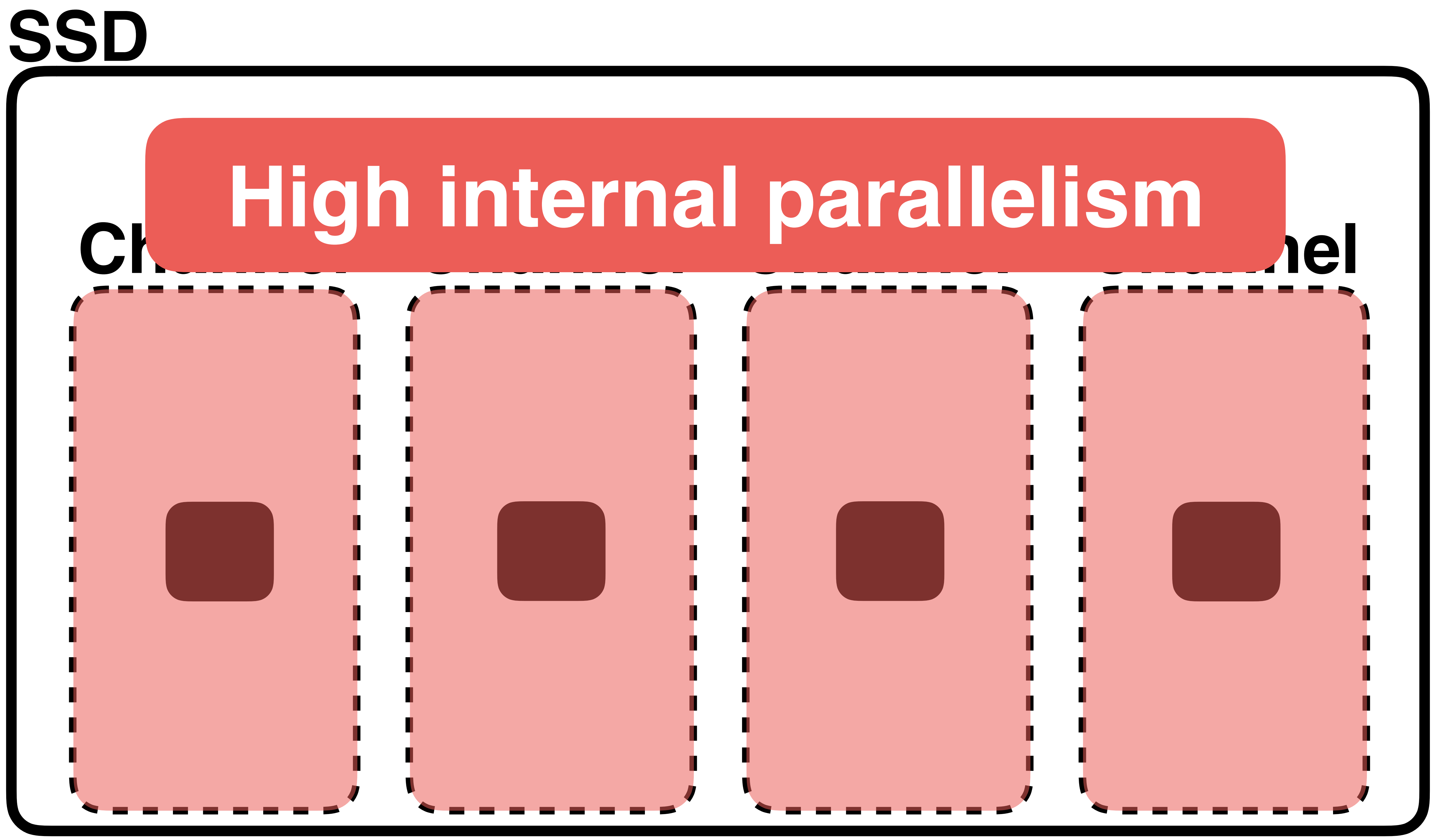
Rule #1: Request Scale

SSD clients should issue **large** data requests or **multiple** outstanding data requests.



Rule #1: Request Scale

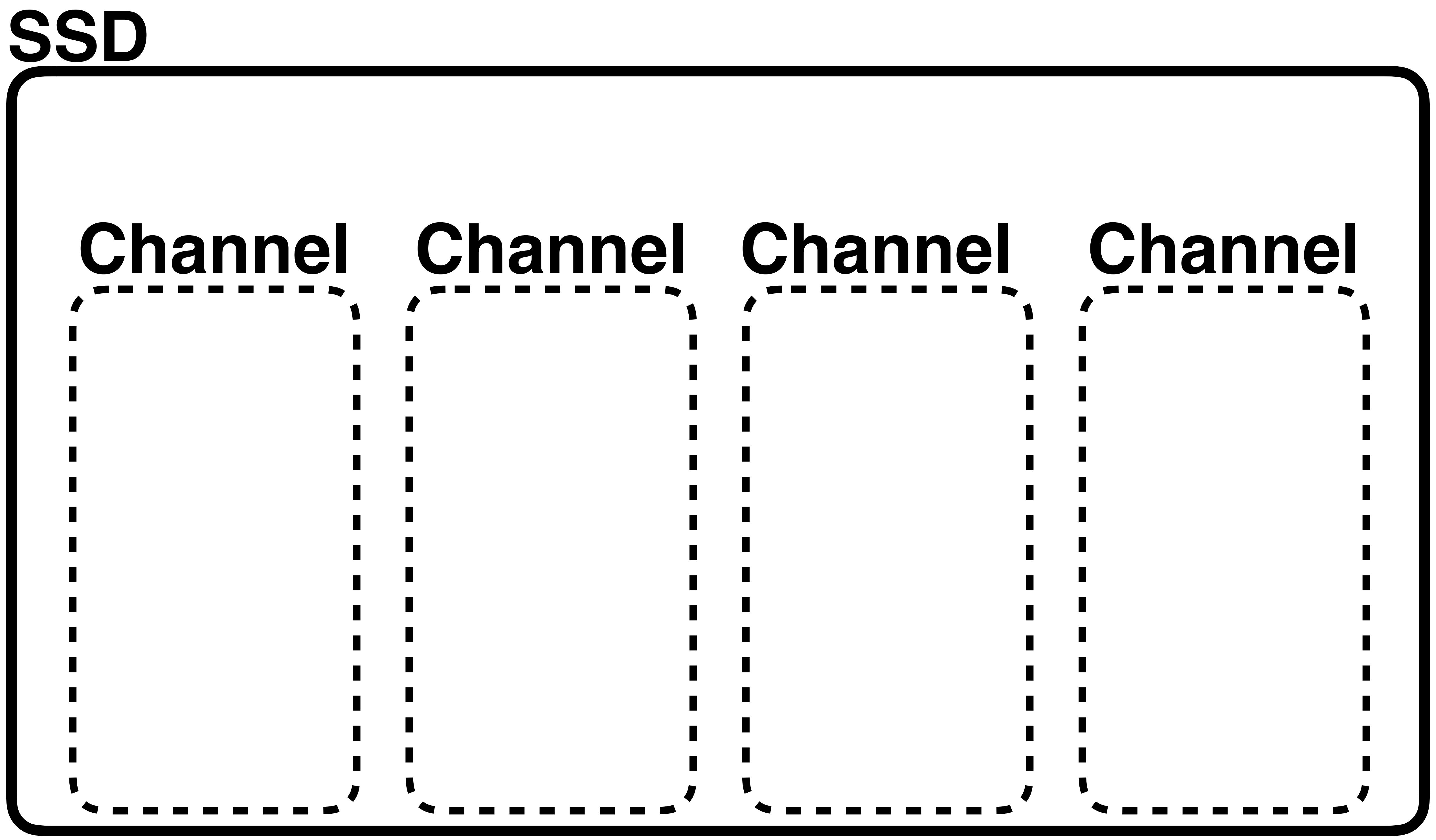
SSD clients should issue **large** data requests or **multiple** outstanding data requests.



Rule #1: Request Scale

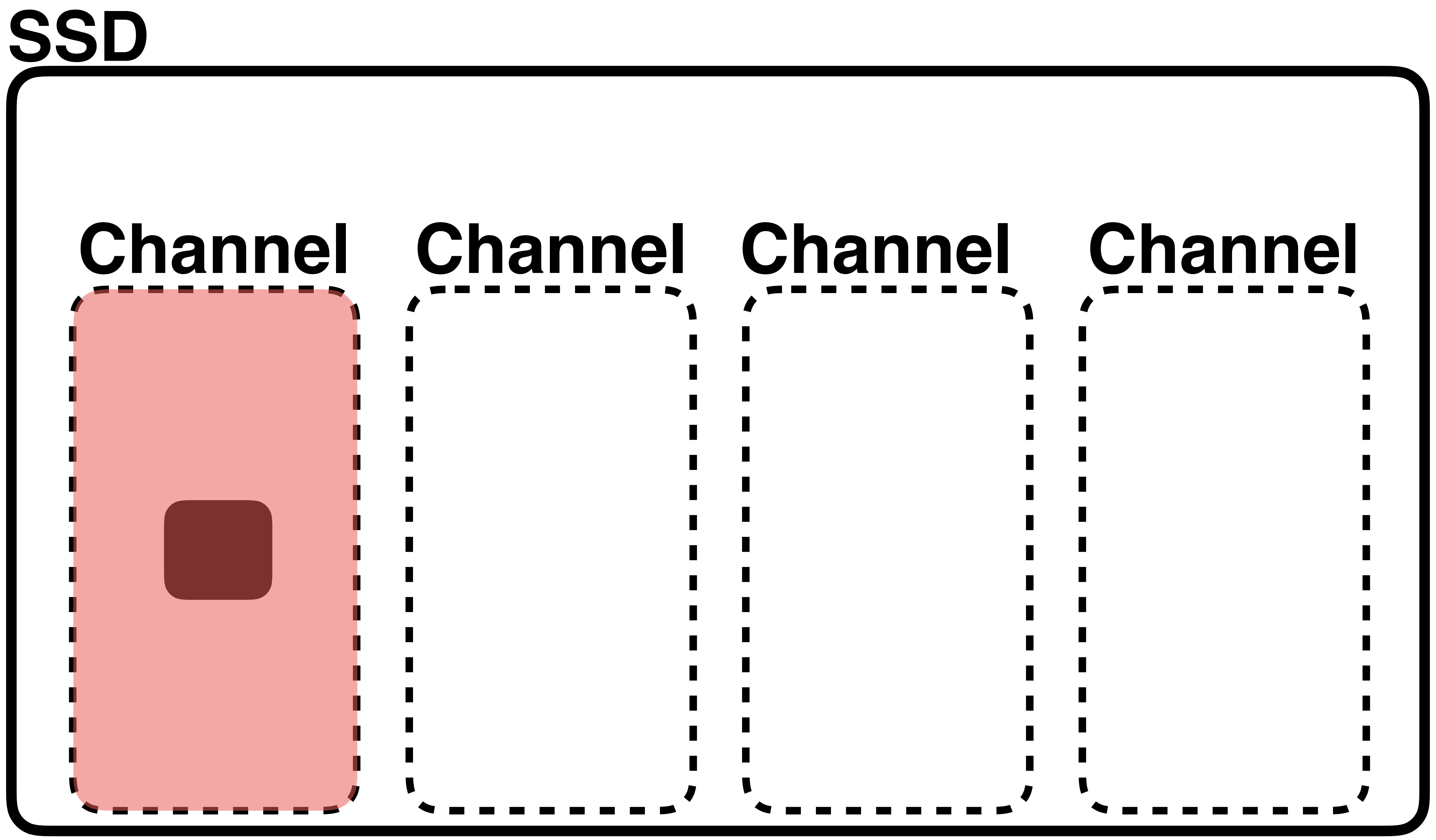
Violation

Rule #1: Request Scale
Violation

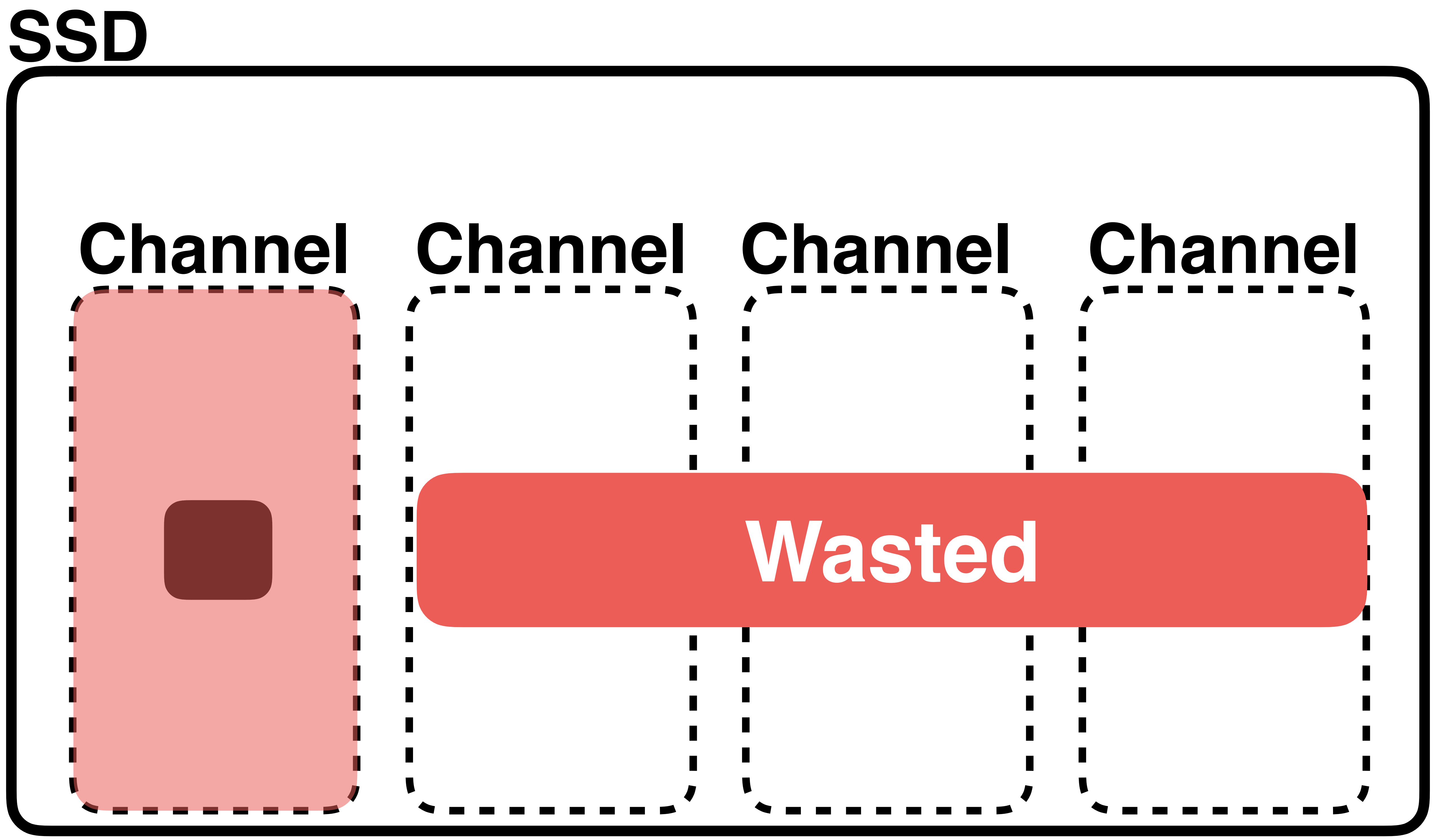


Rule #1: Request Scale

Violation



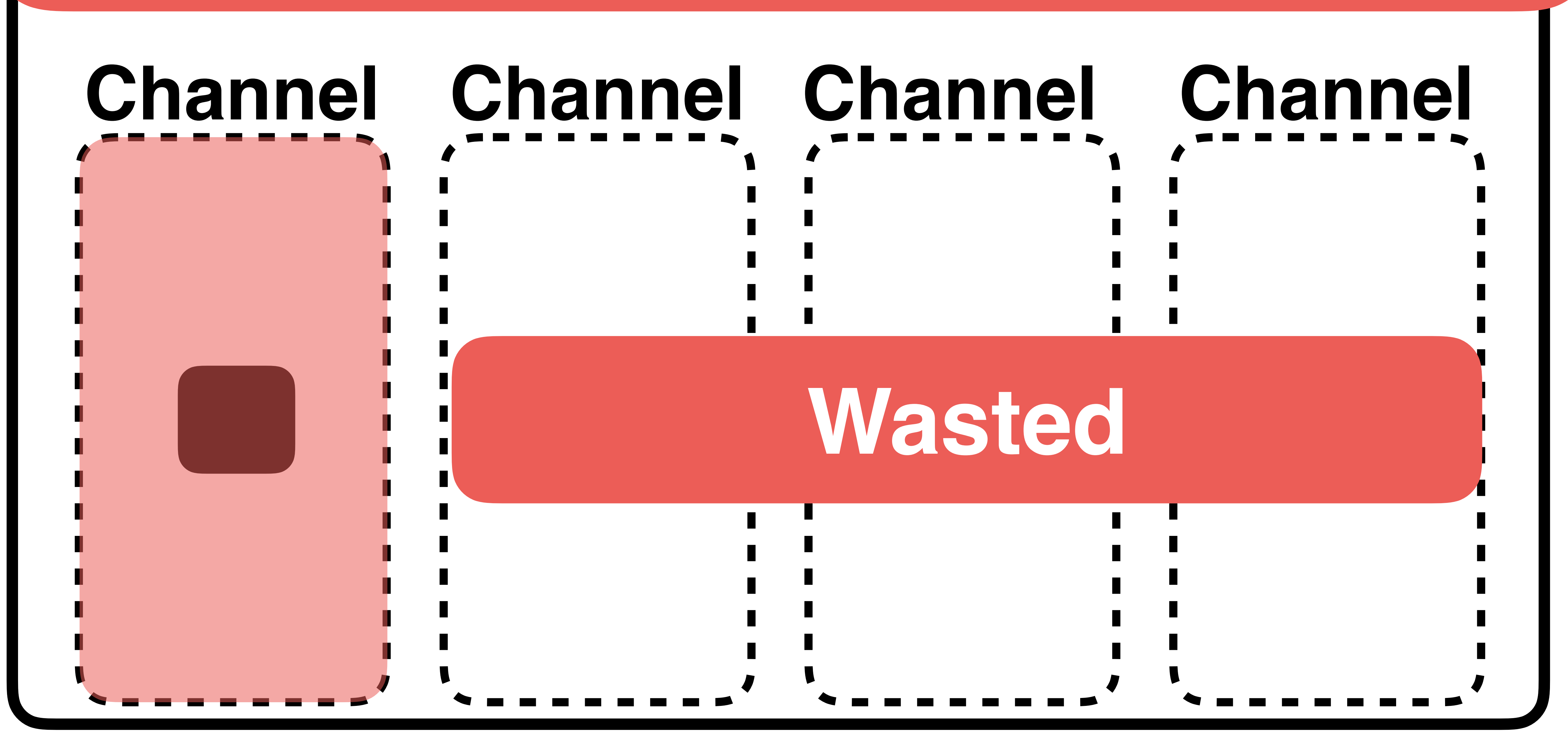
Rule #1: Request Scale
Violation



Rule #1: Request Scale
Violation

If you violate the rule:

- Low internal parallelism



Rule #1: Request Scale
Violation

If you violate the rule:

- Low internal parallelism

Performance impact:

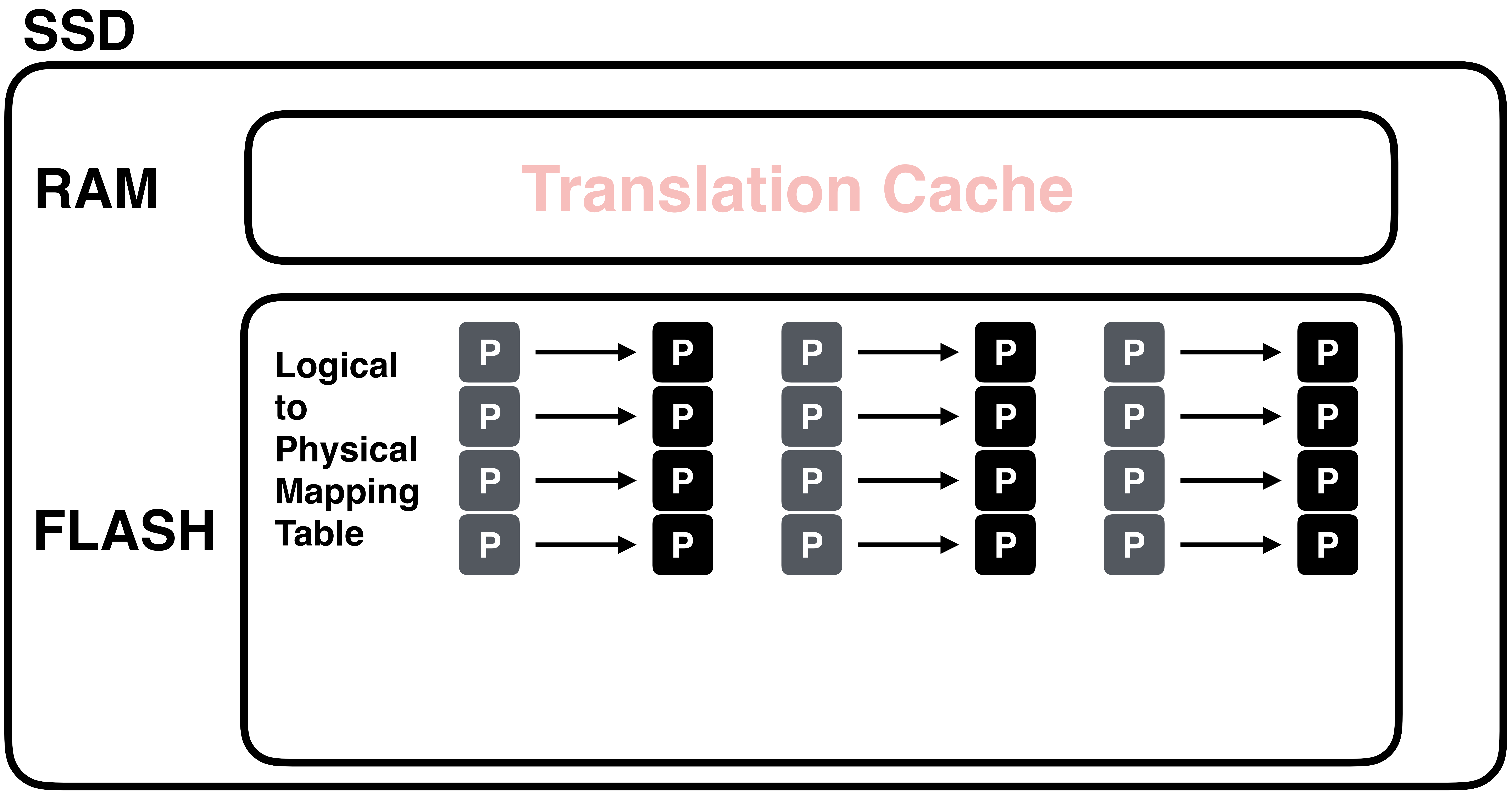
18x read bandwidth

10x write bandwidth

F. Chen, R. Lee, and X. Zhang. Essential Roles of Exploit- ing Internal Parallelism of Flash Memory Based Solid State Drives in High-speed Data Processing. In Proceedings of the 17th International Symposium on High Performance Com- puter Architecture (HPCA-11), pages 266–277, San Antonio, Texas, February 2011.

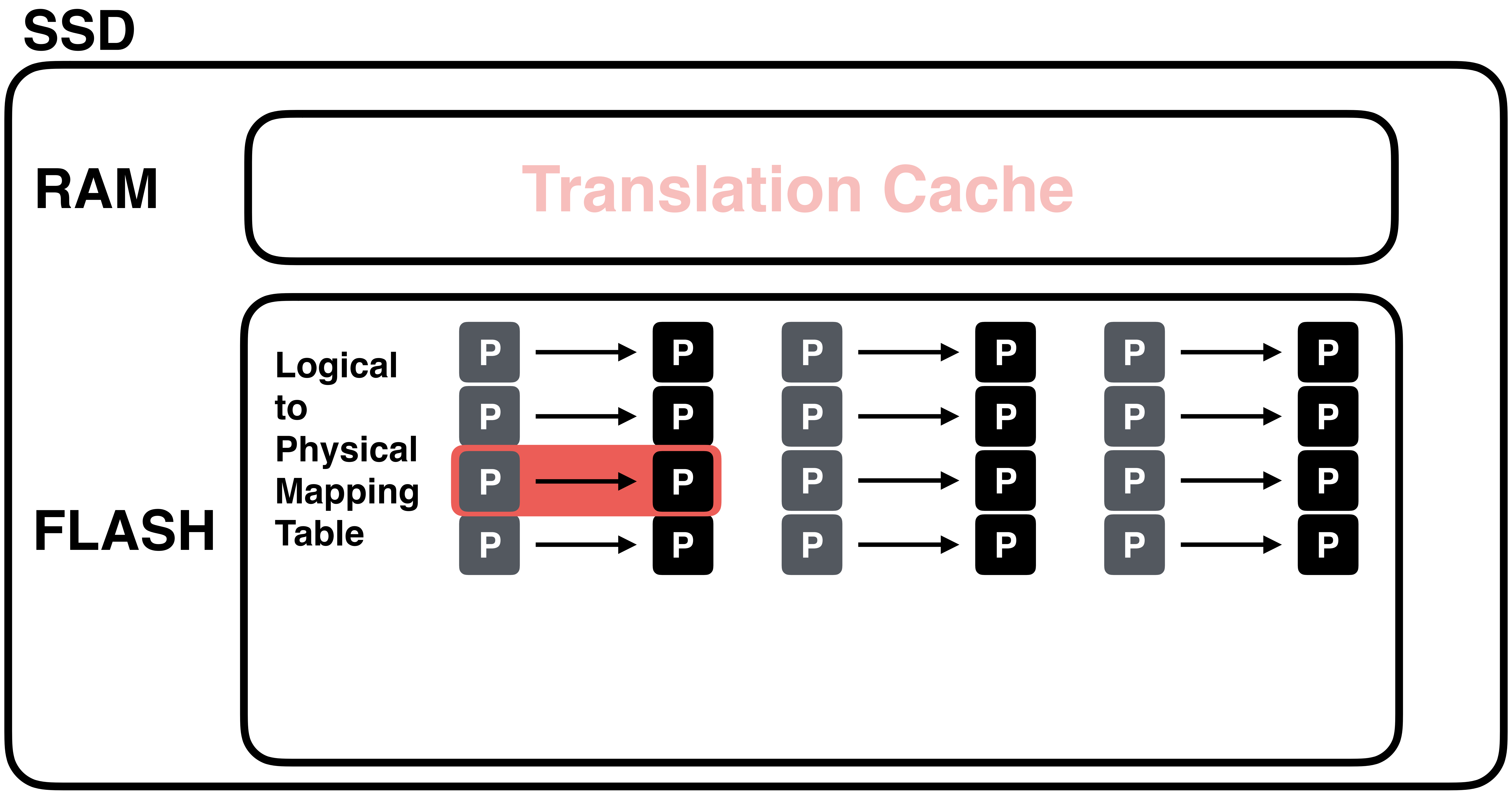
Rule 2: Locality

SSD clients should access with locality



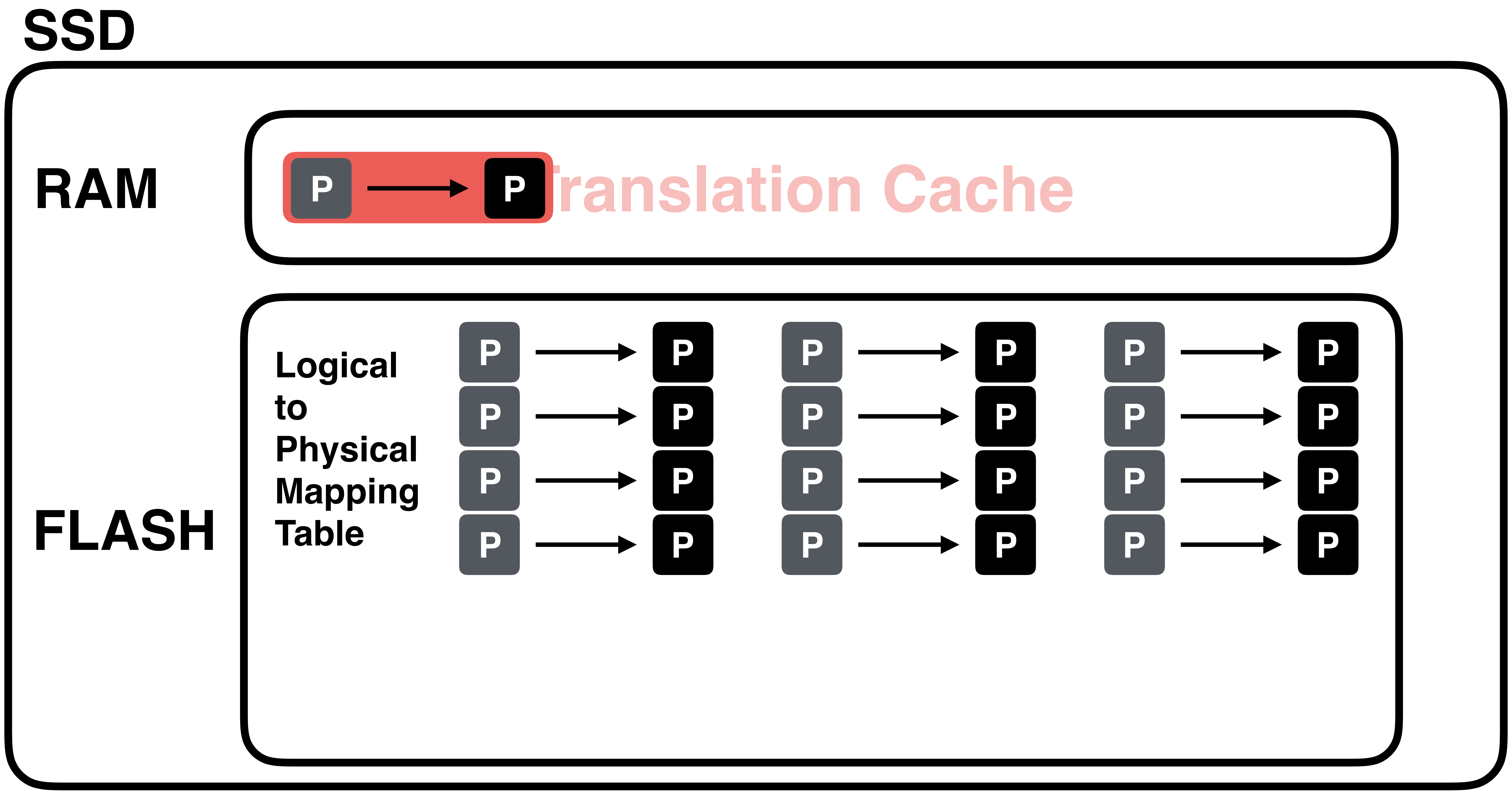
Rule 2: Locality

SSD clients should access with locality



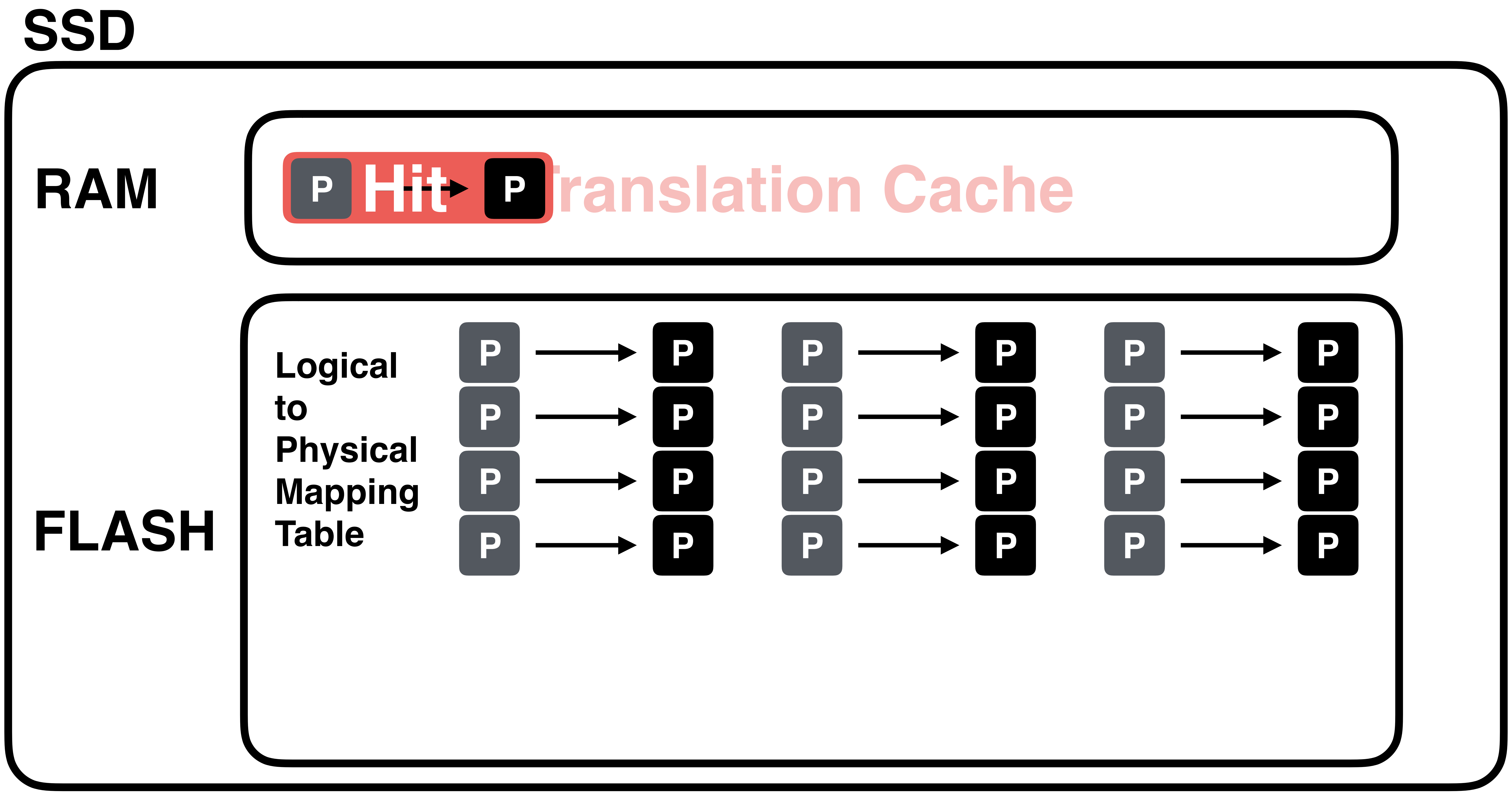
Rule 2: Locality

SSD clients should access with locality



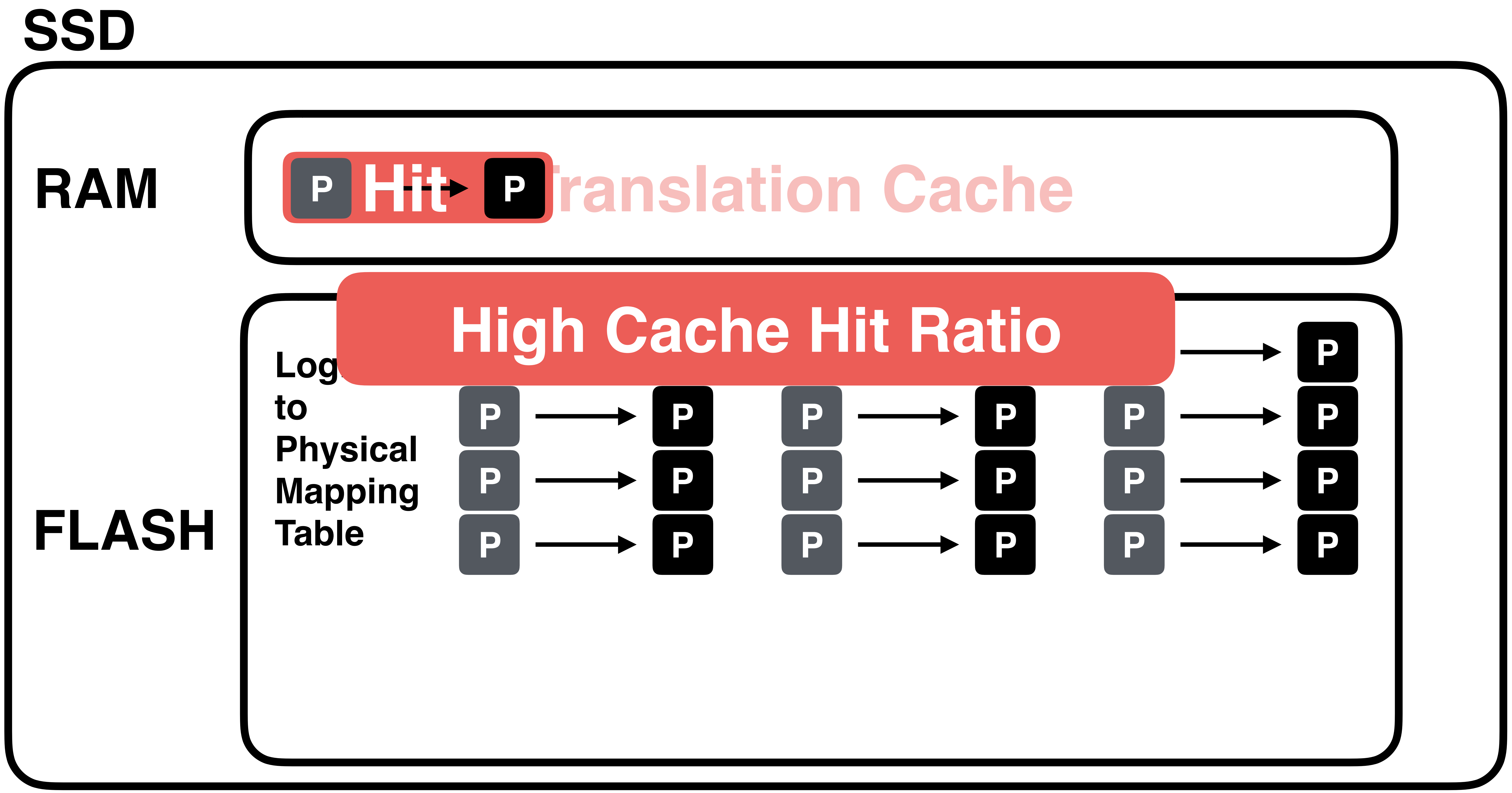
Rule 2: Locality

SSD clients should access with locality



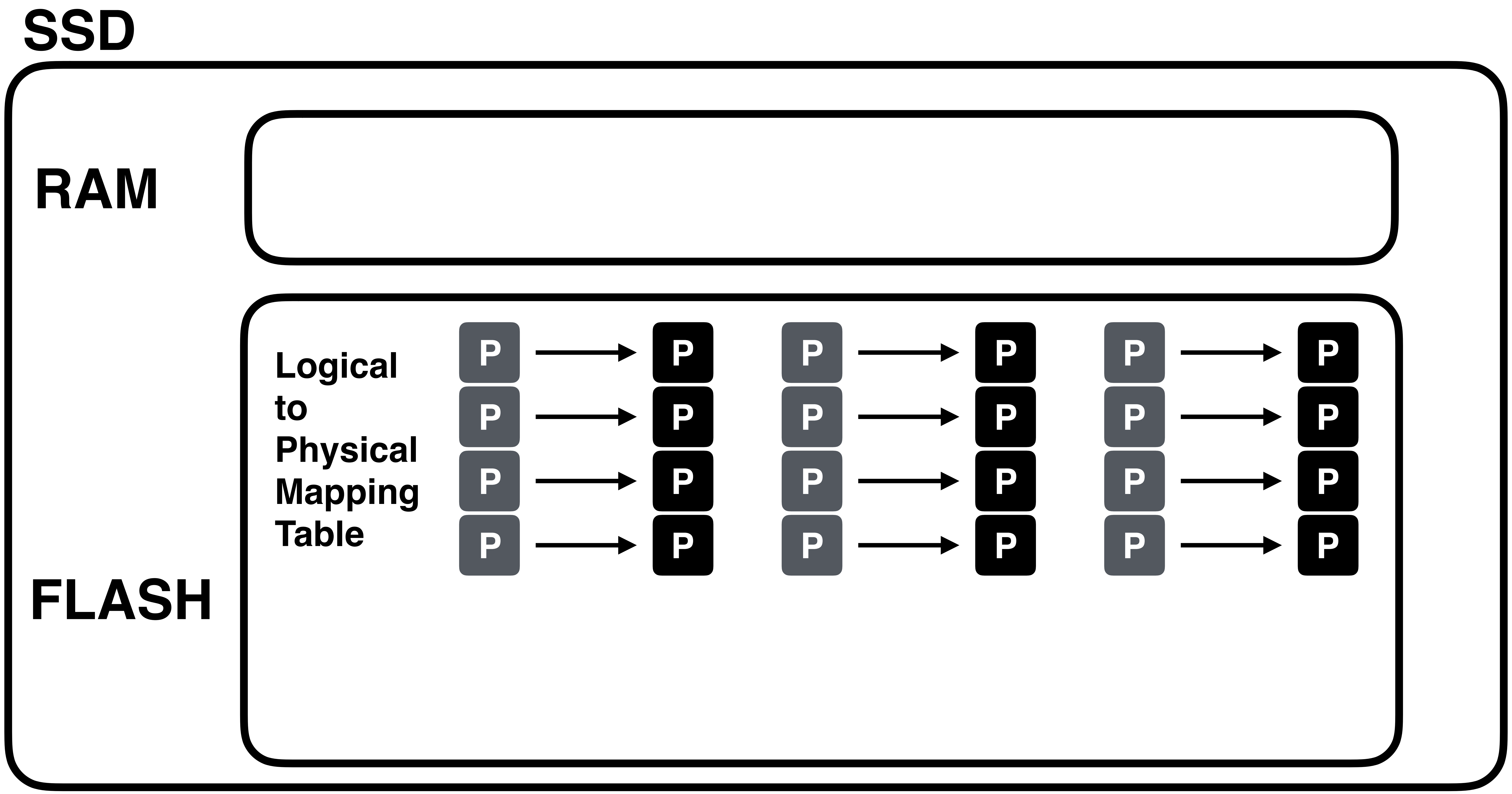
Rule 2: Locality

SSD clients should access with locality



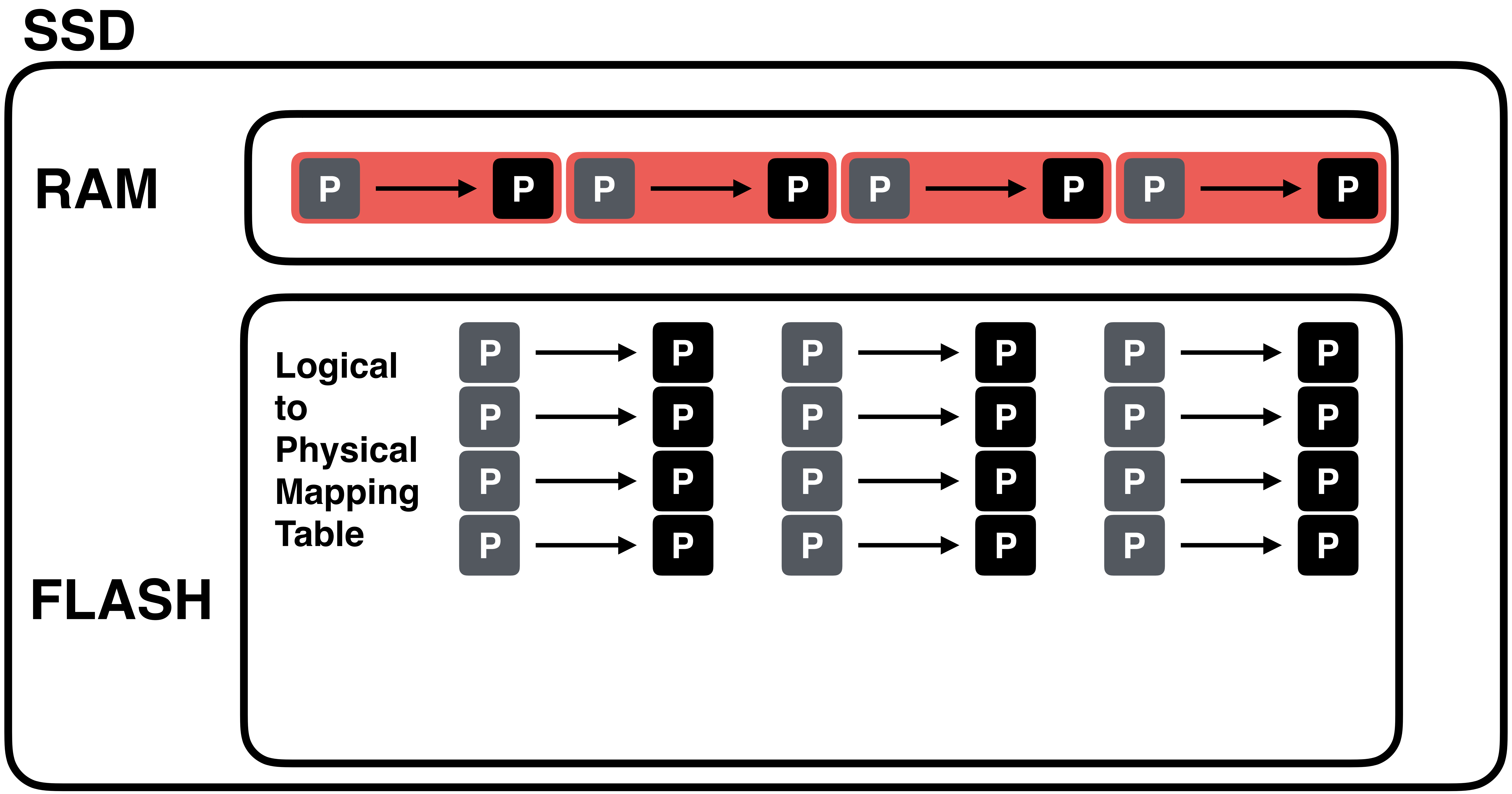
Rule 2: Locality Violation

SSD clients should access with locality



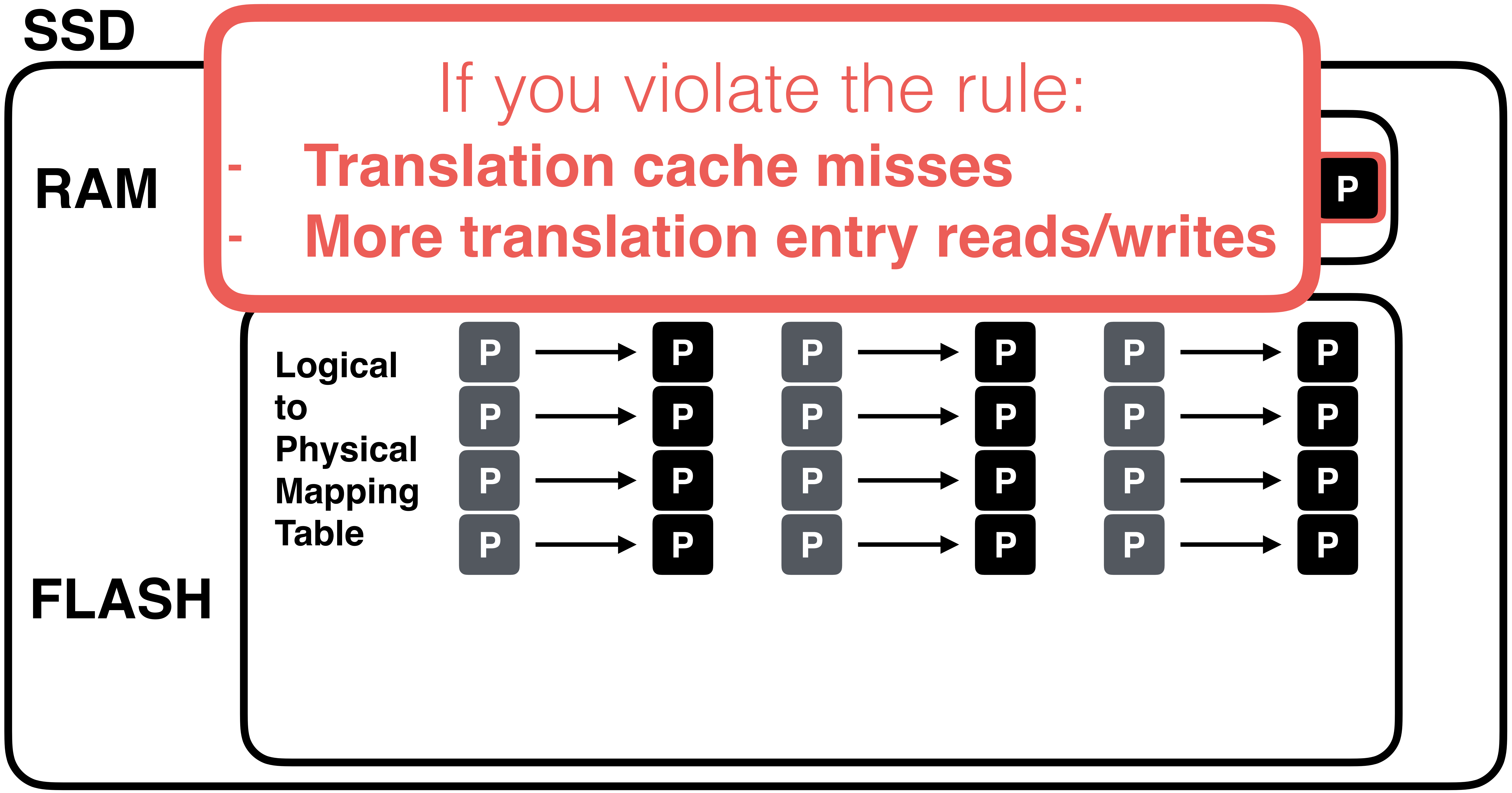
Rule 2: Locality Violation

SSD clients should access with locality



Rule 2: Locality Violation

SSD clients should access with locality

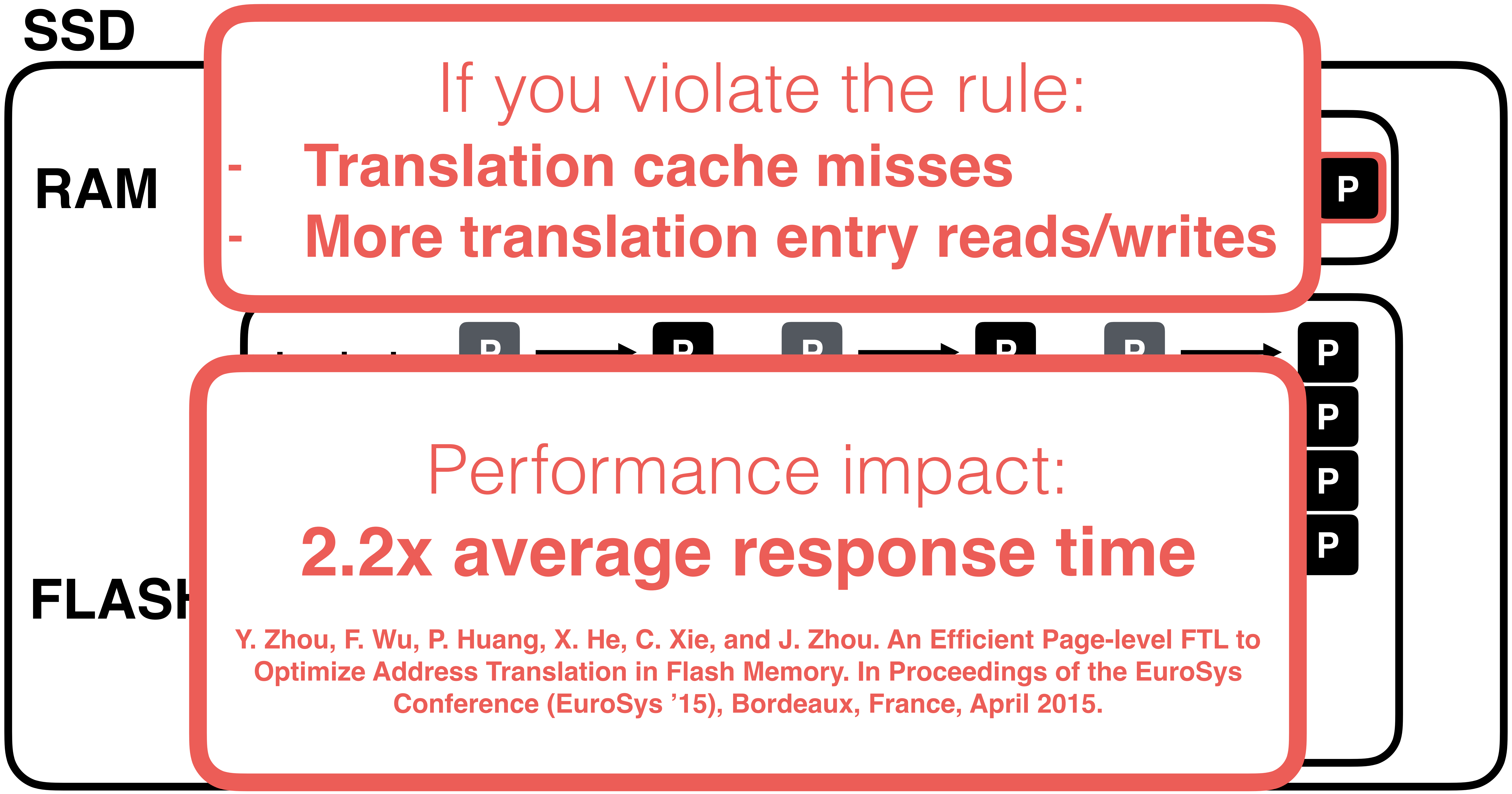


If you violate the rule:

- Translation cache misses
- More translation entry reads/writes

Rule 2: Locality Violation

SSD clients should access with locality



If you violate the rule:

- **Translation cache misses**
- **More translation entry reads/writes**

Performance impact:

2.2x average response time

Y. Zhou, F. Wu, P. Huang, X. He, C. Xie, and J. Zhou. An Efficient Page-level FTL to Optimize Address Translation in Flash Memory. In Proceedings of the EuroSys Conference (EuroSys '15), Bordeaux, France, April 2015.

Rule 3: Aligned Sequentiality

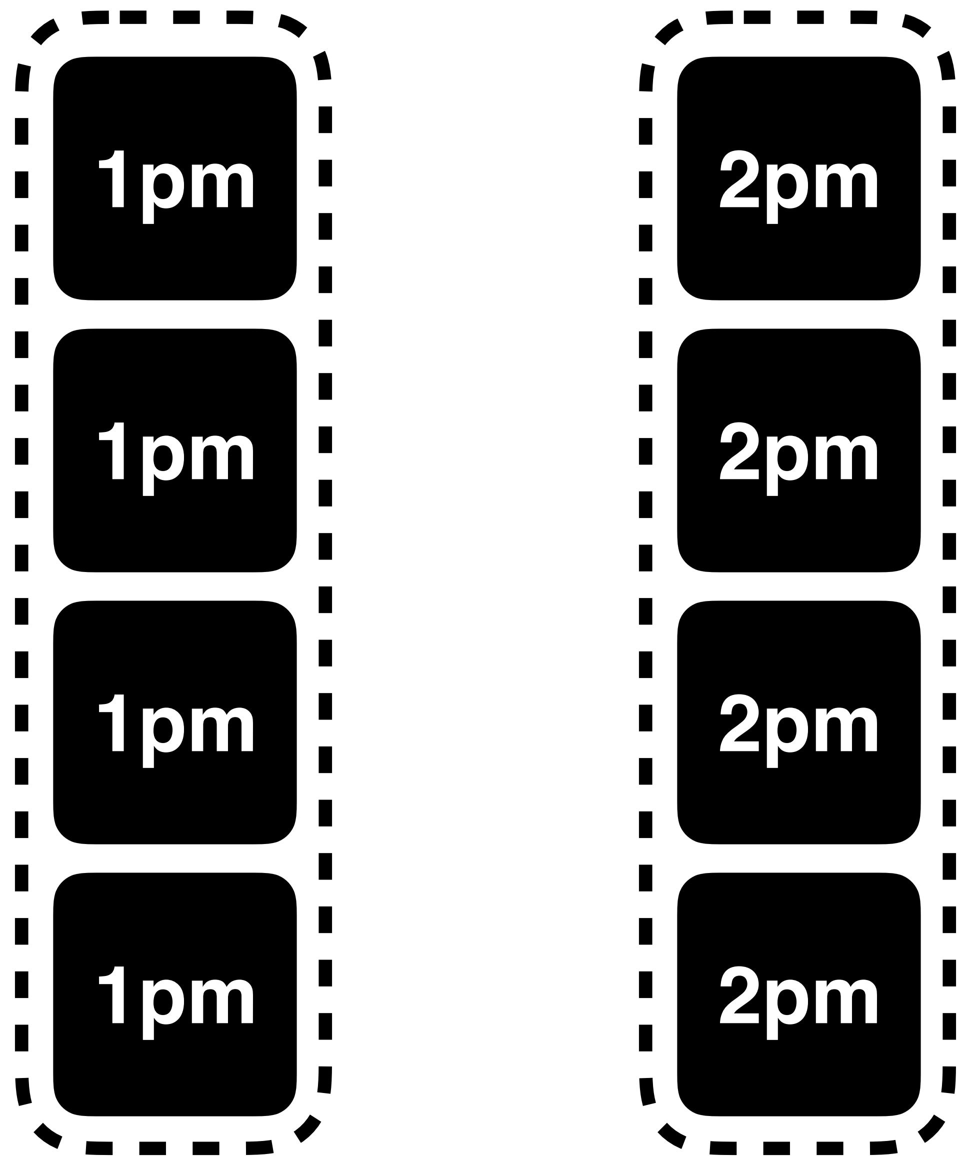
Details in the paper

Rule 4: Grouping By Death Time

Data with similar death times should be placed in the same block.

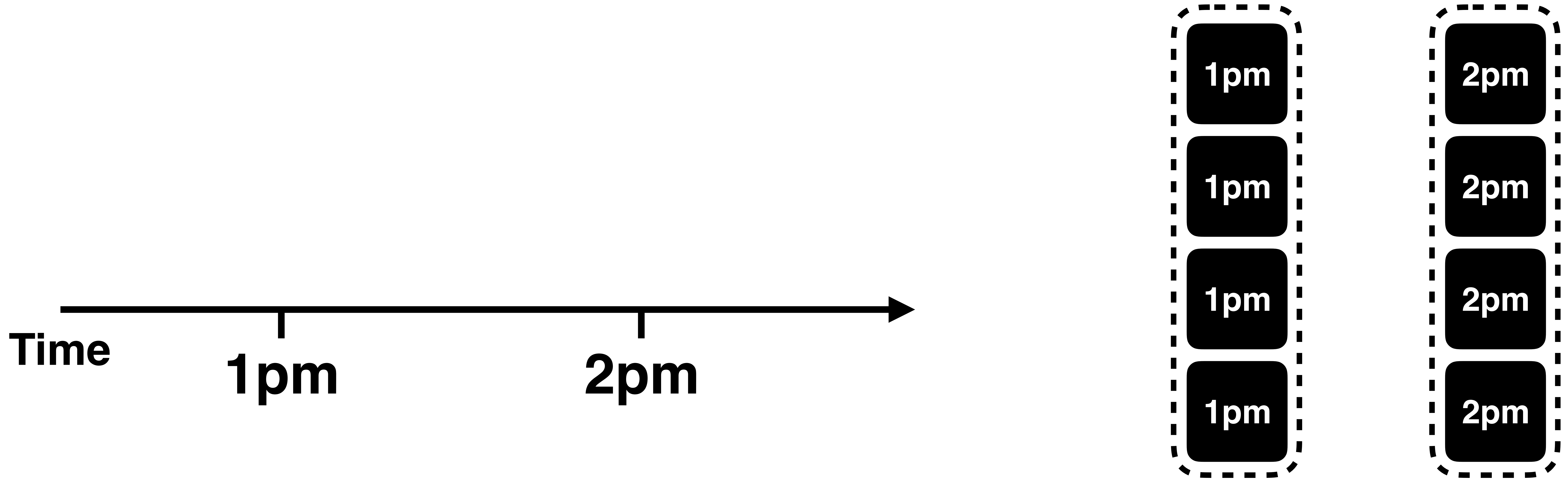
Rule 4: Grouping By Death Time

Data with similar death times should be placed in the same block.



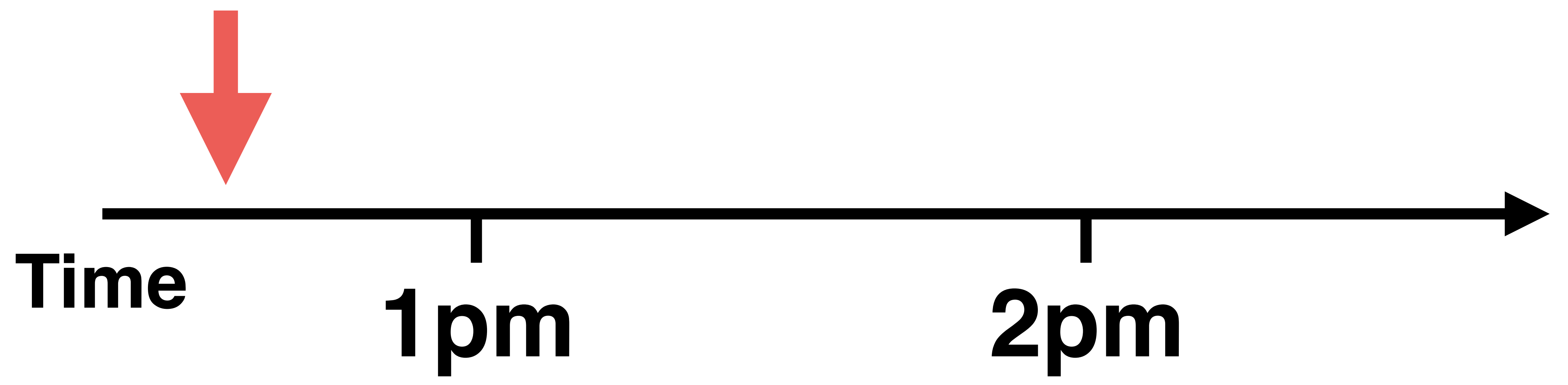
Rule 4: Grouping By Death Time

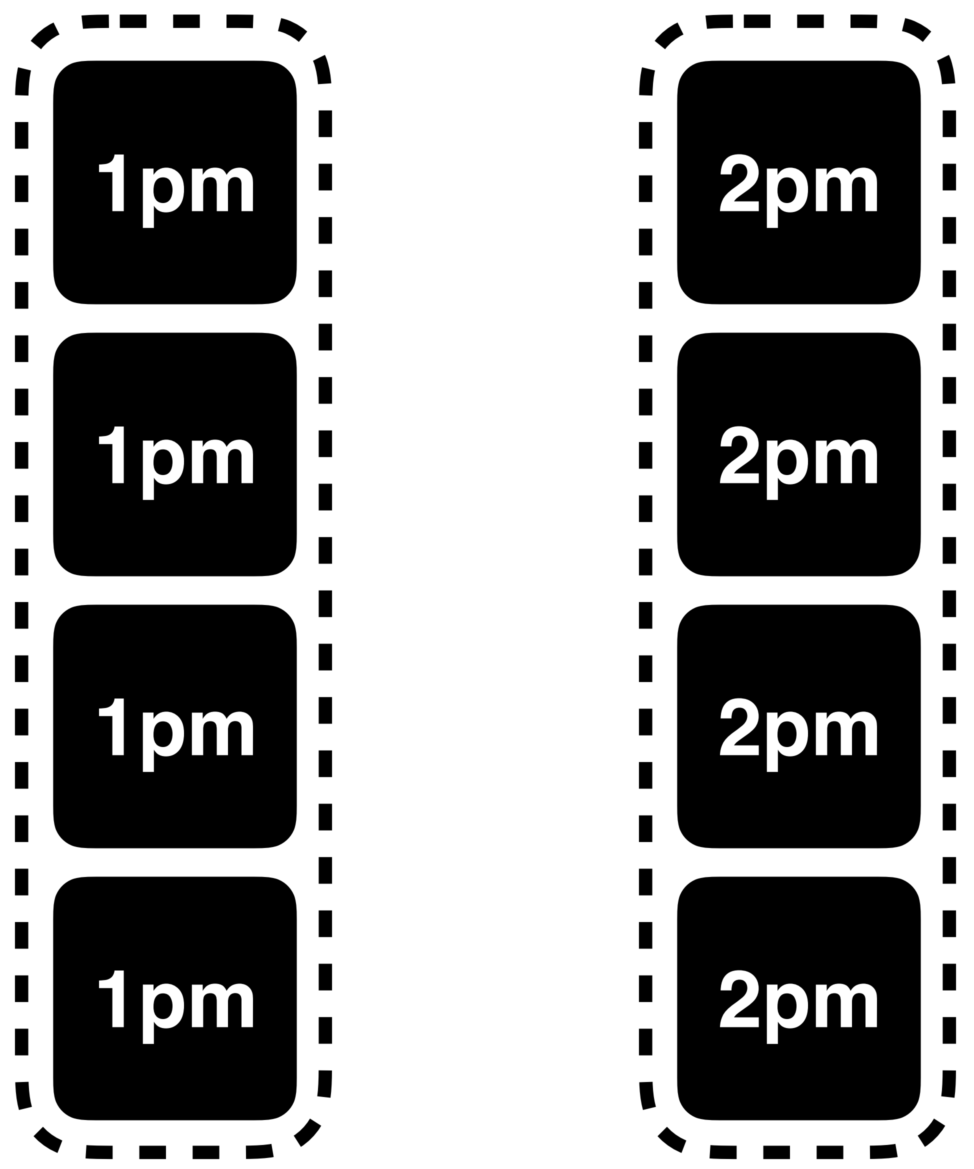
Data with similar death times should be placed in the same block.



Rule 4: Grouping By Death Time

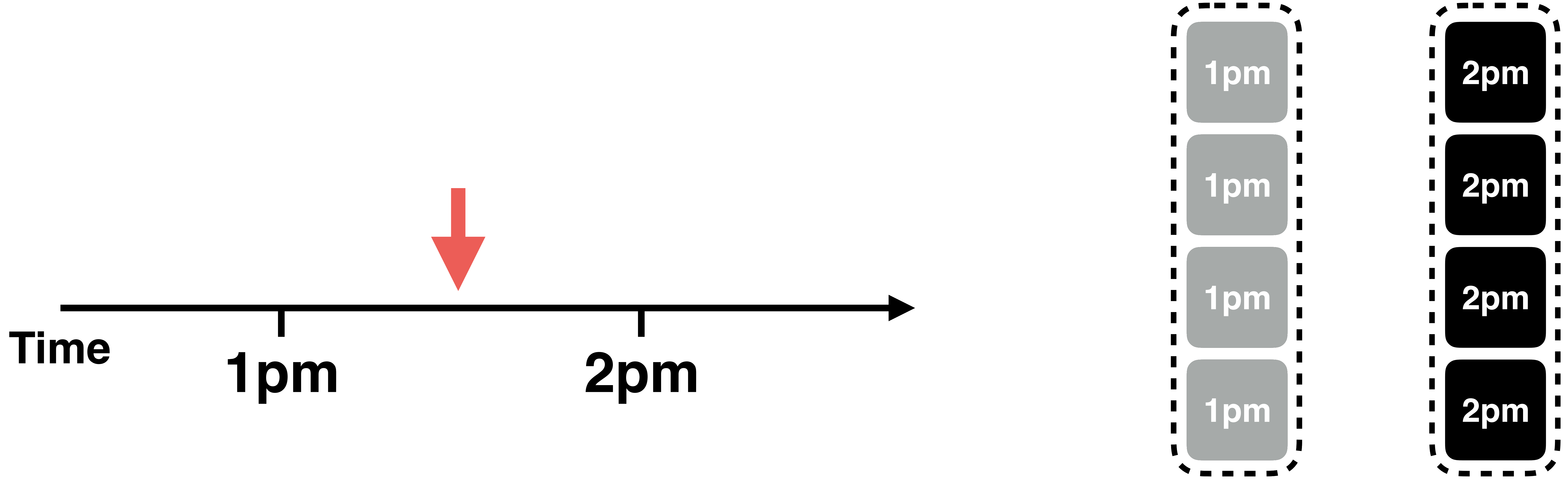
Data with similar death times should be placed in the same block.





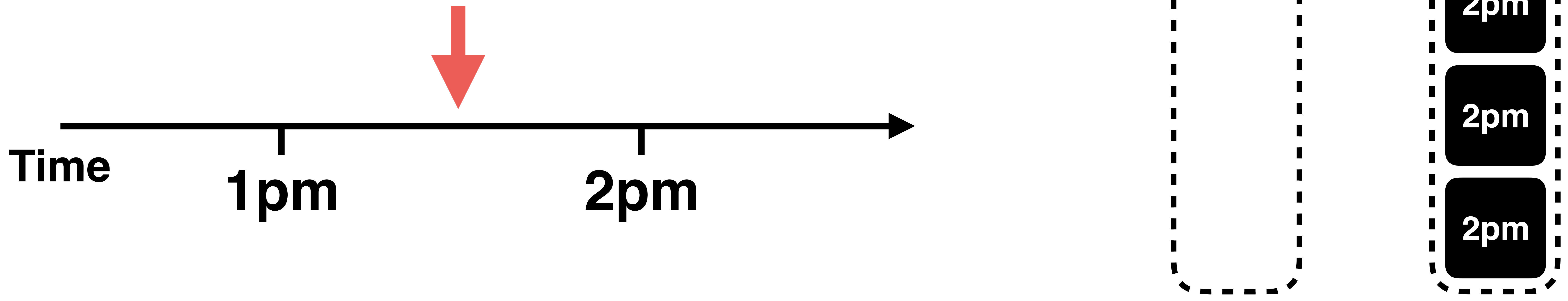
Rule 4: Grouping By Death Time

Data with similar death times should be placed in the same block.



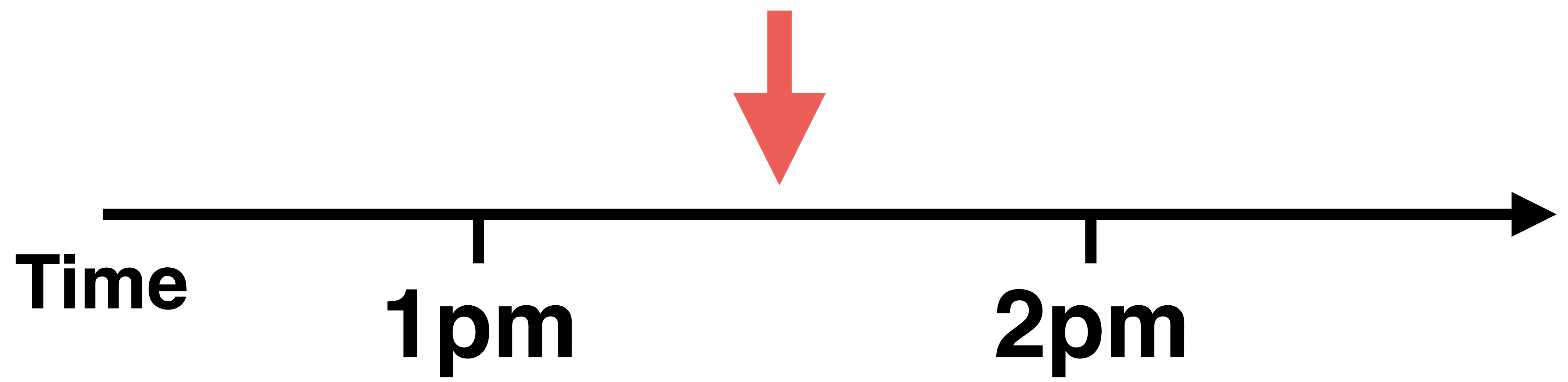
Rule 4: Grouping By Death Time

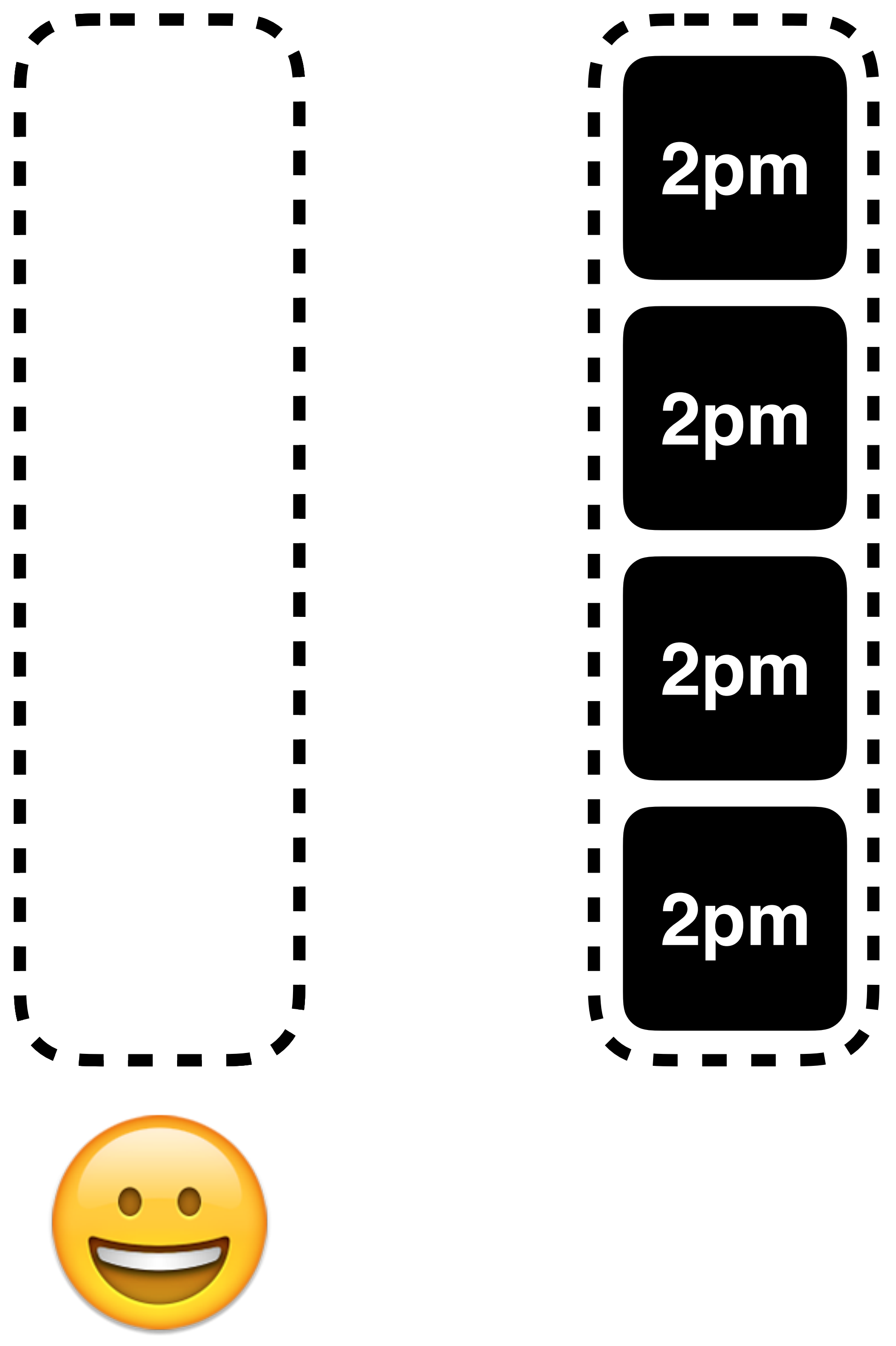
Data with similar death times should be placed in the same block.



Rule 4: Grouping By Death Time

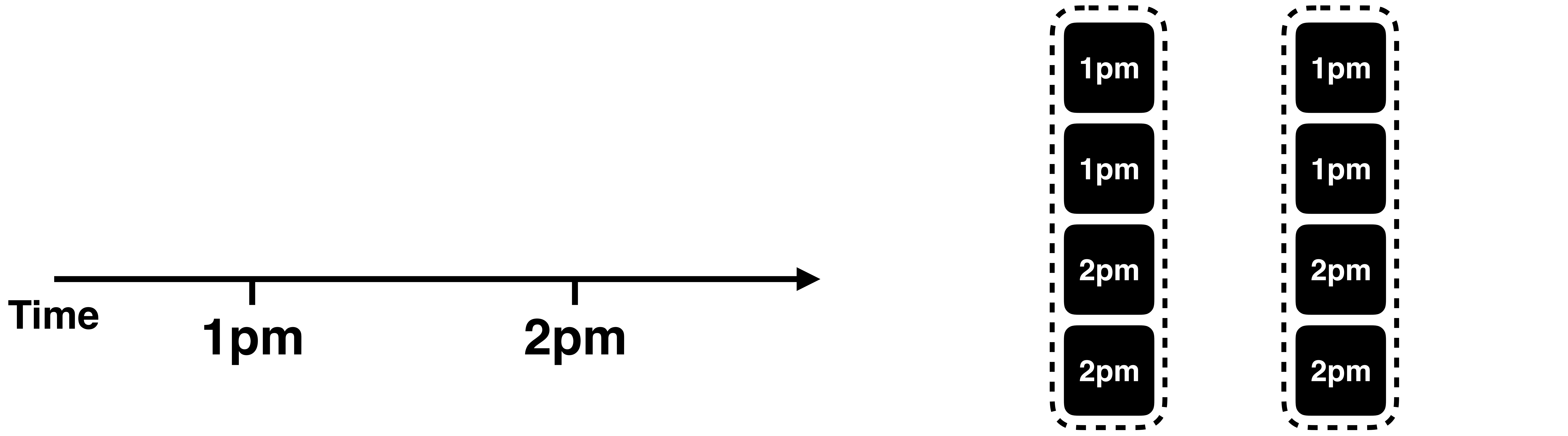
Data with similar death times should be placed in the same block.





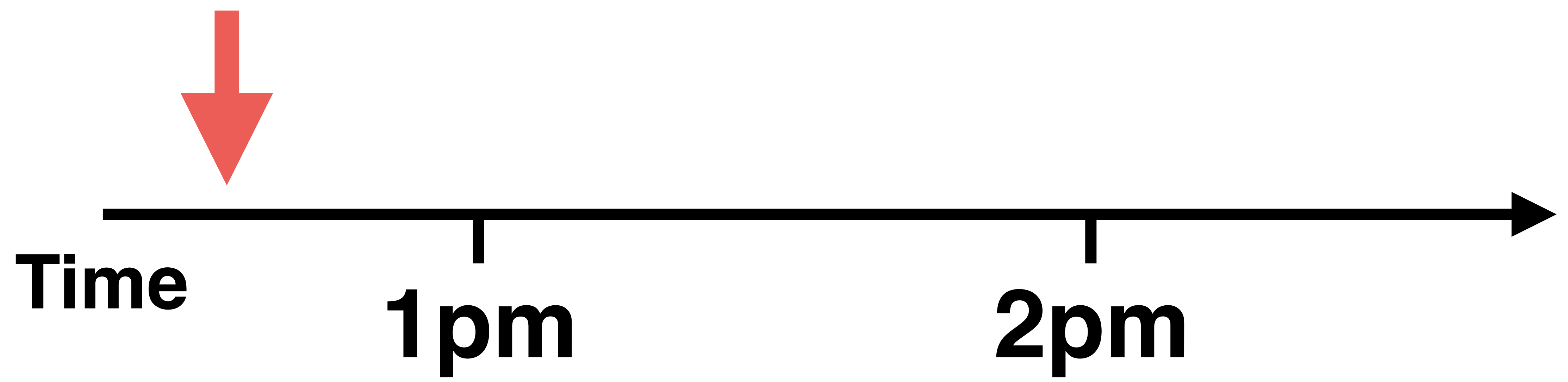
Rule 4: Grouping By Death Time

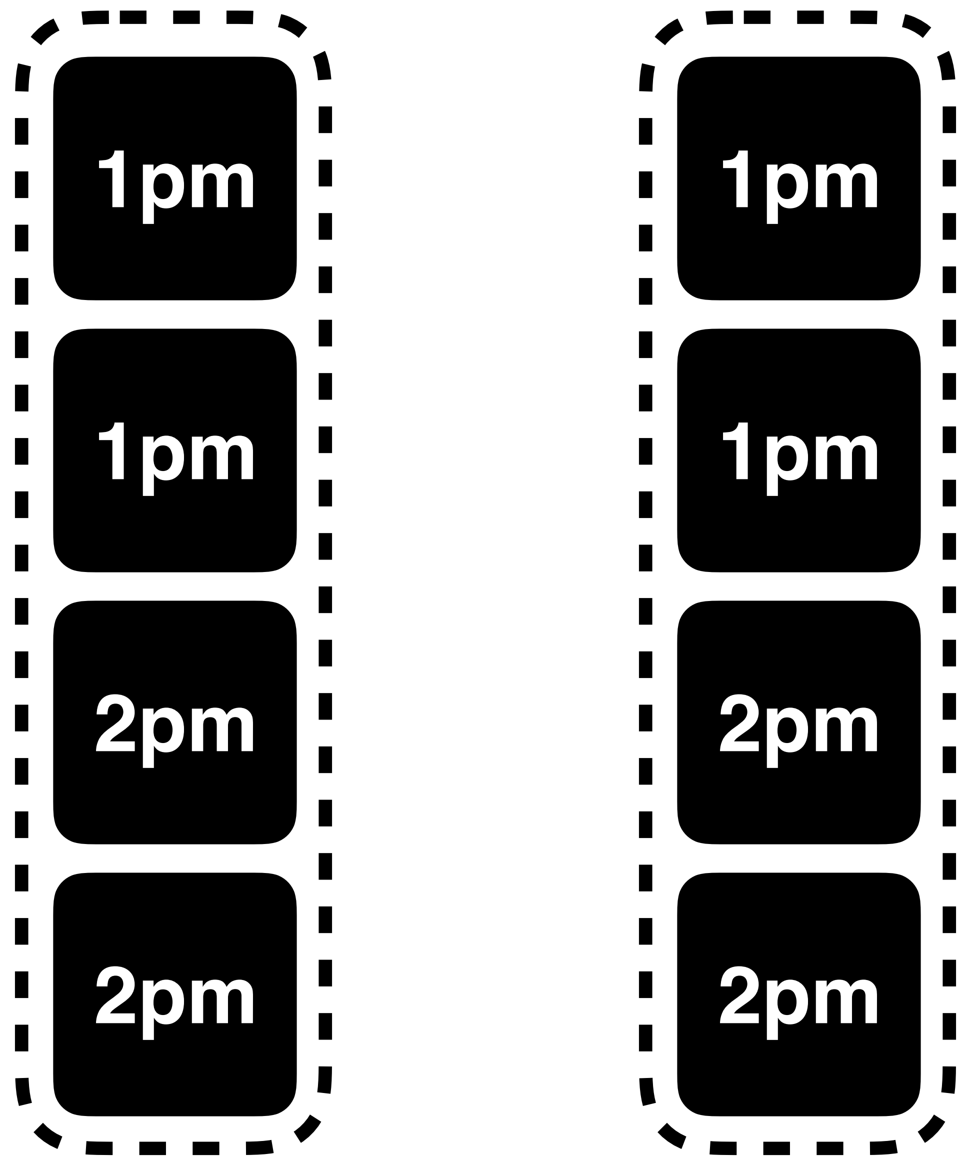
Violation



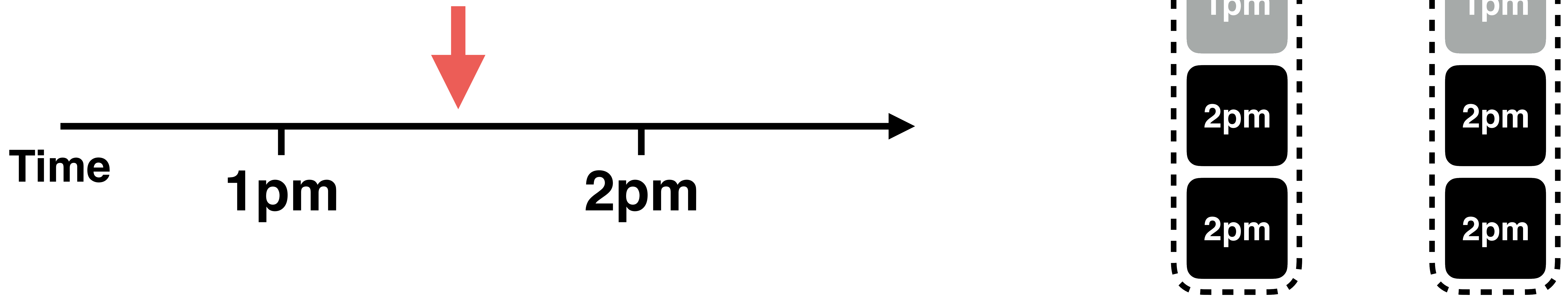
Rule 4: Grouping By Death Time

Violation



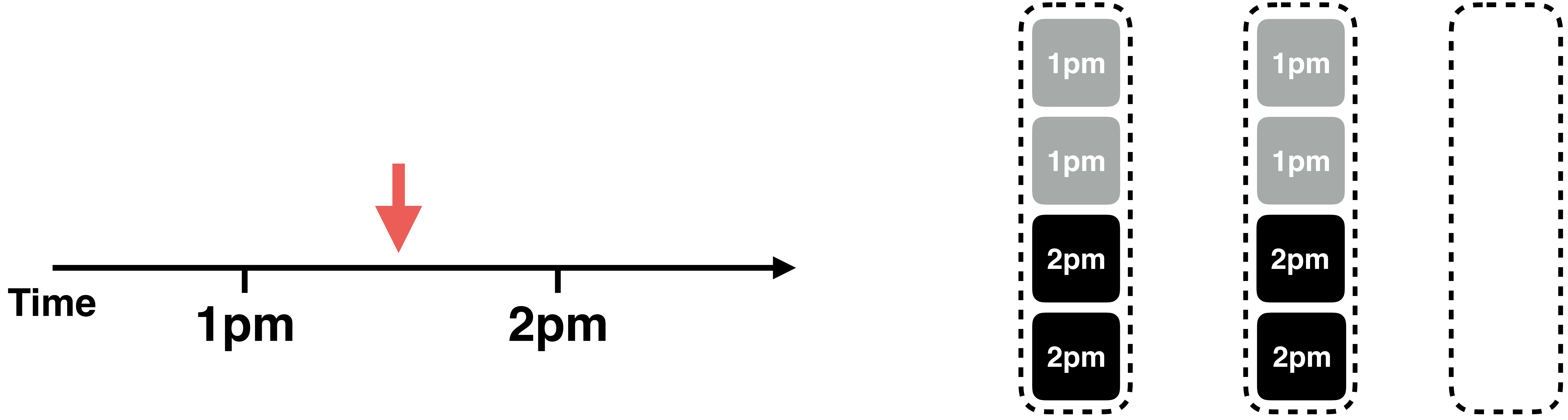


Rule 4: Grouping By Death Time
Violation



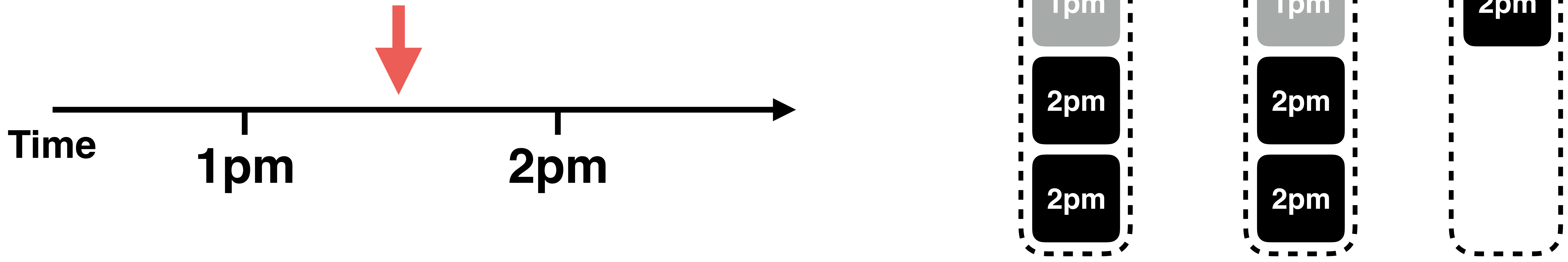
Rule 4: Grouping By Death Time

Violation



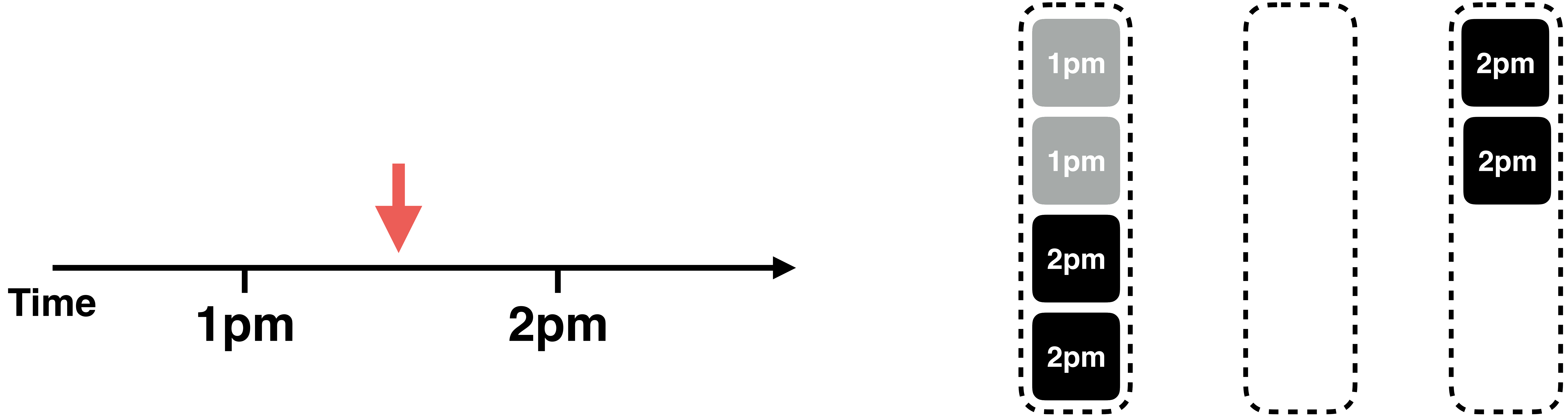
Rule 4: Grouping By Death Time

Violation

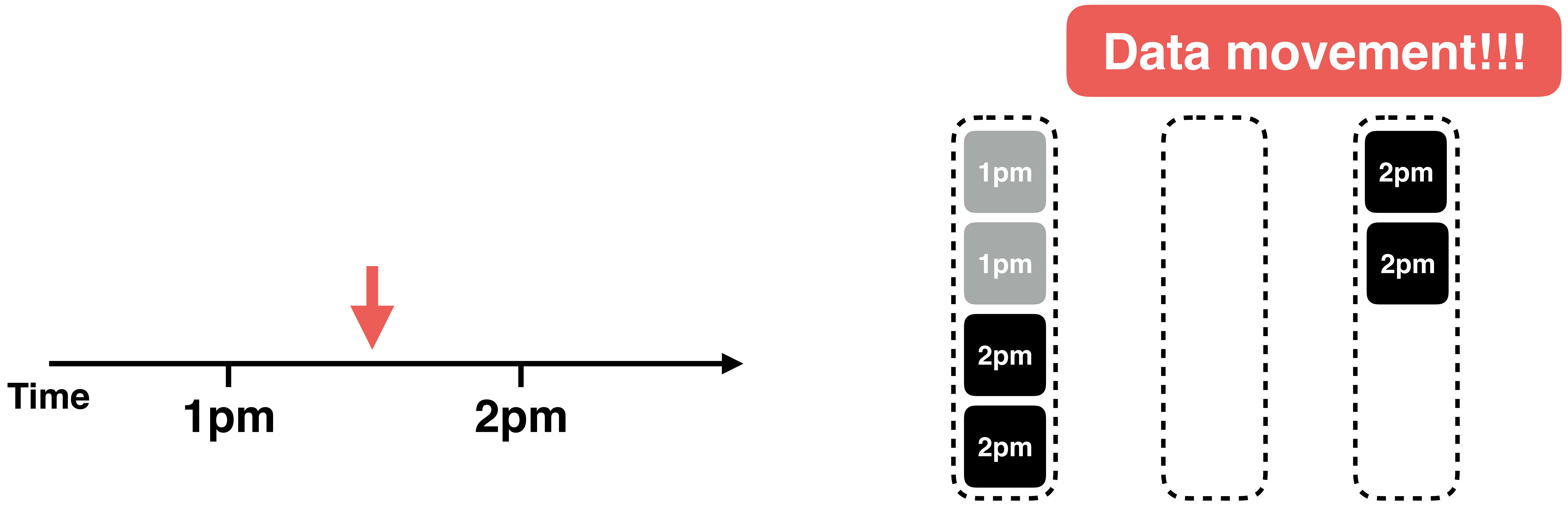


Rule 4: Grouping By Death Time

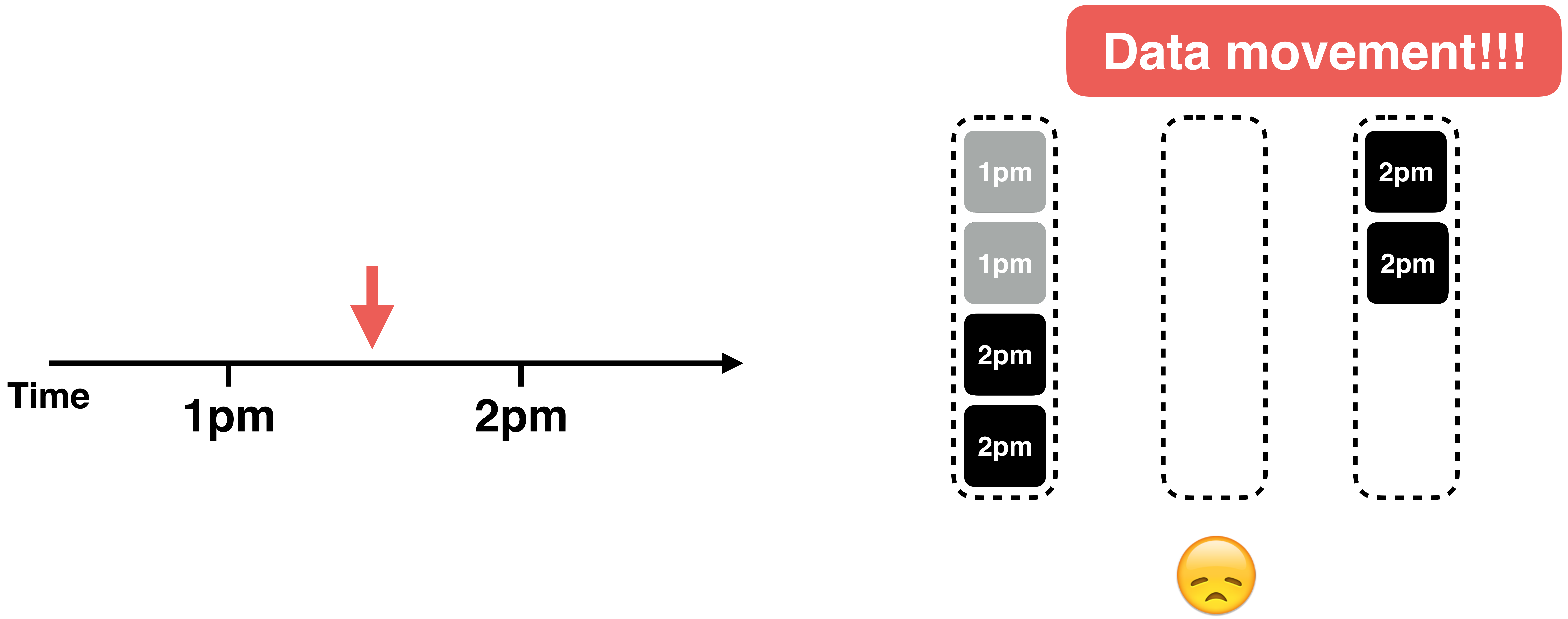
Violation



Rule 4: Grouping By Death Time
Violation



Rule 4: Grouping By Death Time
Violation

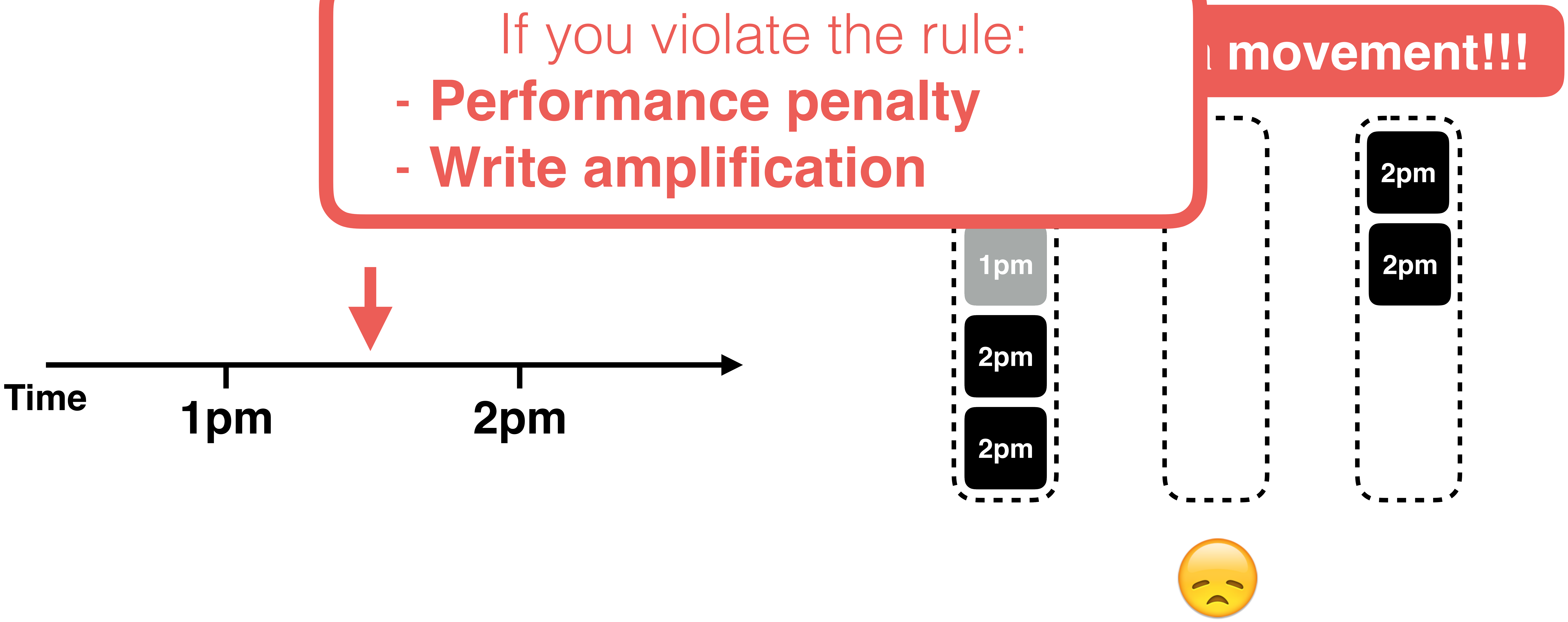


Rule 4: Grouping By Death Time

Violation

If you violate the rule:

- **Performance penalty**
- **Write amplification**



Rule 4: Grouping By Death Time

Violation

movement!!!

If you violate the rule:

- **Performance penalty**
- **Write amplification**

Performance impact:

4.8x write bandwidth

1.6x throughput

1.8x block erasure count

C. Lee, D. Sim, J.-Y. Hwang, and S. Cho. F2FS: A New File System for Flash Storage. In Proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST '15), Santa Clara, California, February 2015.

J.-U. Kang, J. Hyun, H. Maeng, and S. Cho. The Multi- streamed Solid-State Drive. In 6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '14), Philadelphia, PA, June 2014.

. Y. Cheng, F. Douglis, P. Shilane, G. Wallace, P. Desnoyers, and K. Li. Erasing Belady's Limitations: In Search of Flash Cache Offline Optimality. In 2016 USENIX Annual Technical Conference (USENIX ATC 16), pages 379–392, Denver, CO, 2016. USENIX Association.

Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime

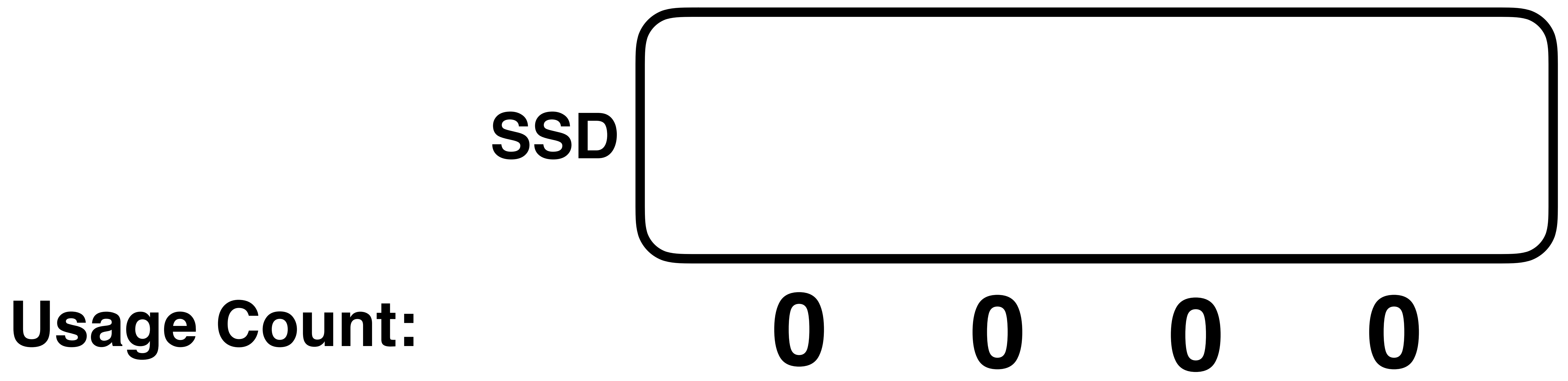
1 Day

Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

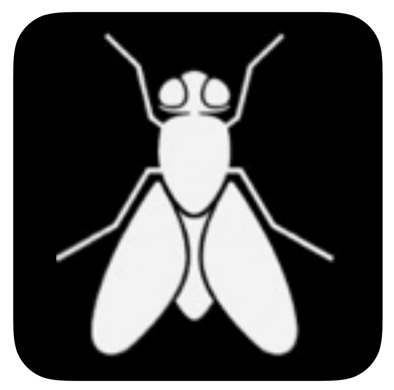
Lifetime

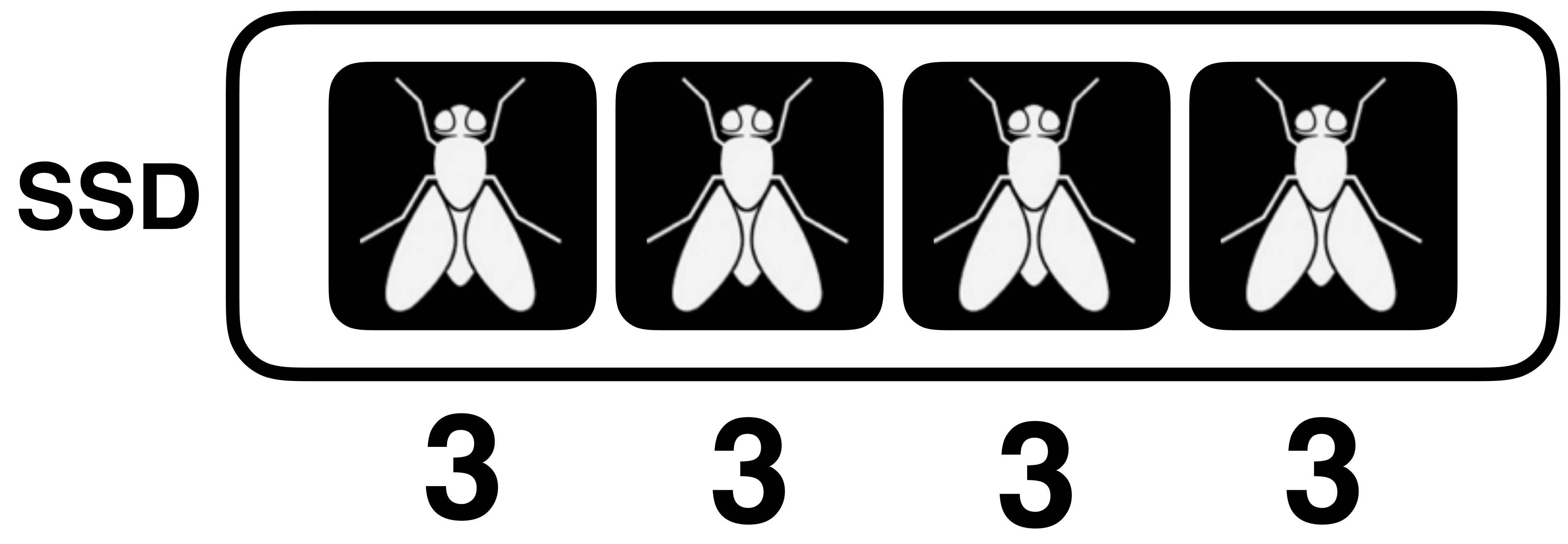
1 Day



Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

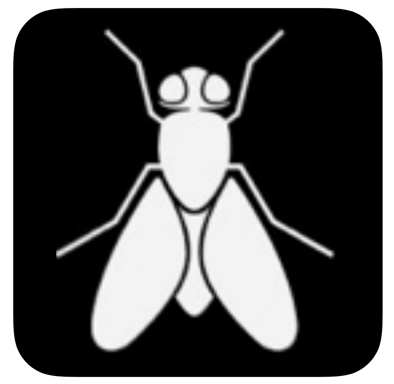




Rule 5: Uniform Data Lifetime

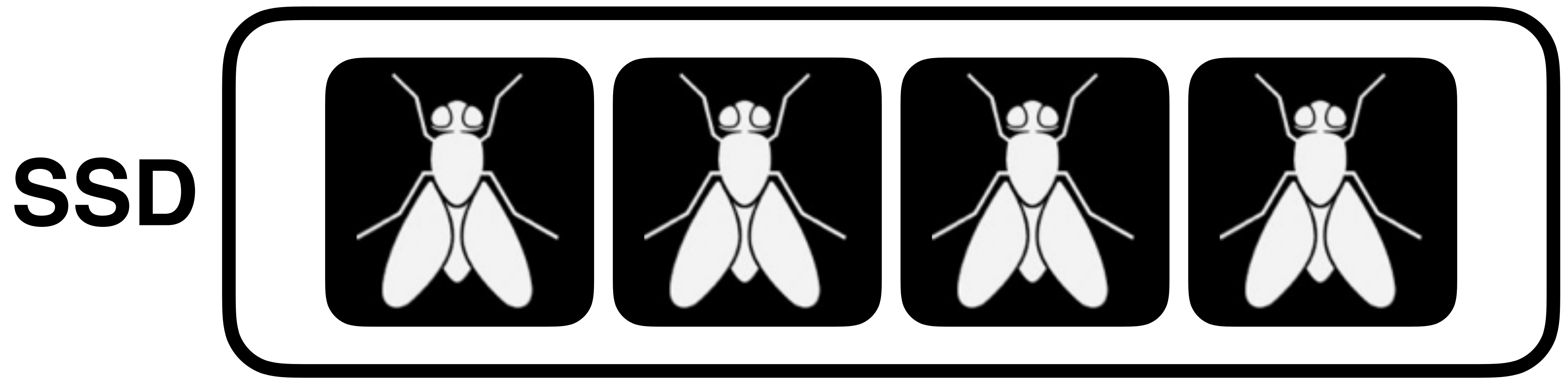
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

No wear-leveling needed



Usage Count: **3 3 3 3**

Rule 5: Uniform Data Lifetime

Violation

Lifetime

1 Day

1000 Years

SSD

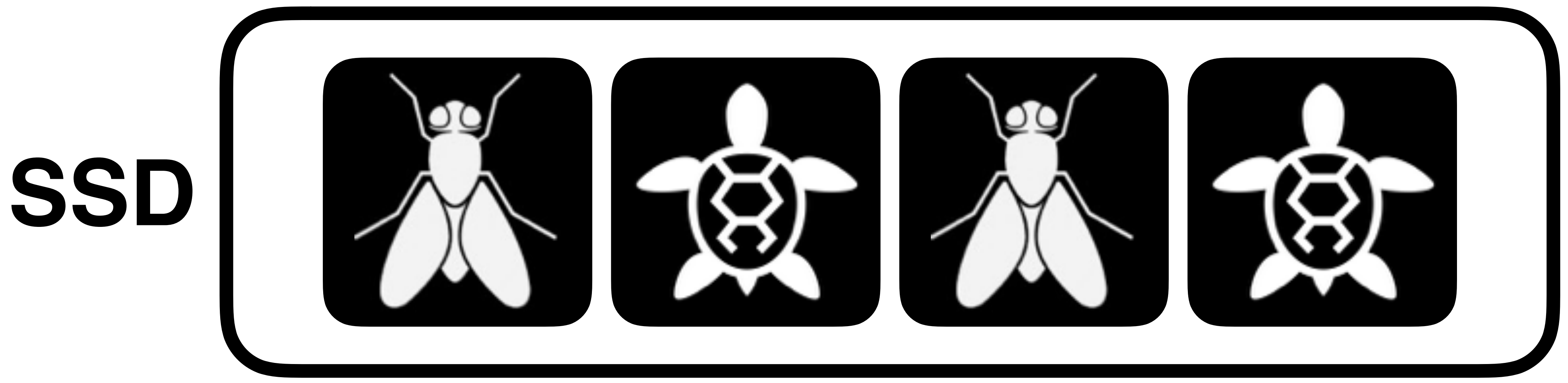
Usage Count: 0 0 0 0

Rule 5: Uniform Data Lifetime
Violation

Lifetime

1 Day

1000 Years



Usage Count: 365*1000 1 365*1000 1

Rule 5: Uniform Data Lifetime

Violation

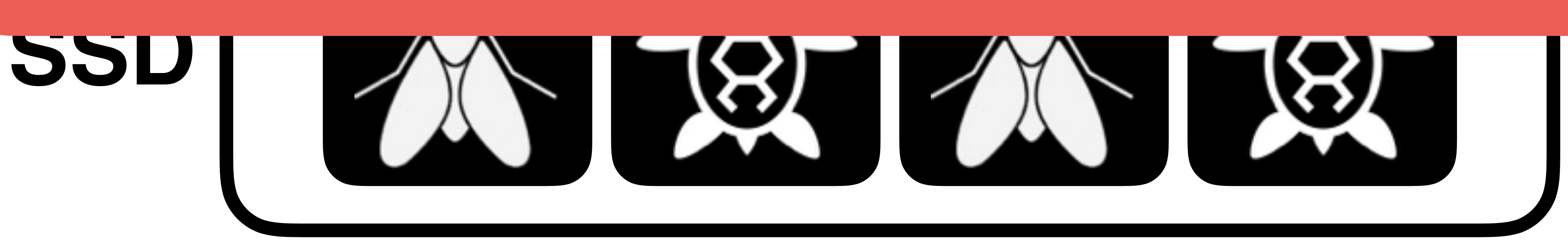
Lifetime

1 Day

1000 Year

Some blocks wear out sooner

Frequent wear-leveling needed!!!



Usage Count: 365*1000 1 365*1000 1

Rule 5: Uniform Data Lifetime

Violation

If you violate the rule:

- **Performance penalty**
- **Write amplification**

Some blocks wear out sooner

Frequent wear-leveling needed!!!

Lifetime

1 Day

1000 Year

Usage Count: 365*1000 1 365*1000 1

Rule 5: Uniform Data Lifetime

Violation

If you violate the rule:

- **Performance penalty**
- **Write amplification**

Lifetime

1 Day

1000 Year

Performance impact:

1.6x write latency

S. Boboila and P. Desnoyers. Write Endurance in Flash Drives: Measurements and Analysis. In Proceedings of the 8th USENIX Symposium on File and Storage Technologies (FAST '10), San Jose, California, February 2010.

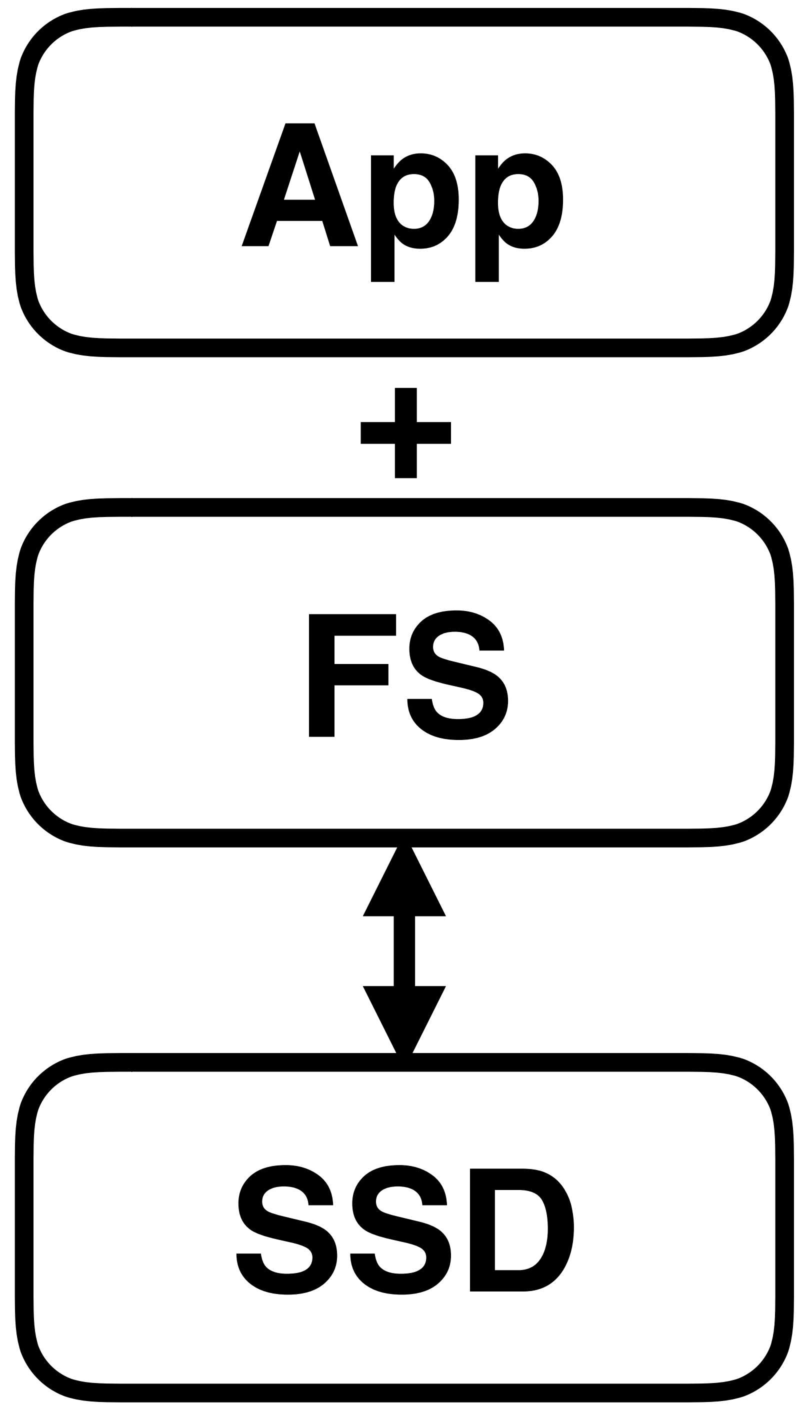
Usage Count:

Outline

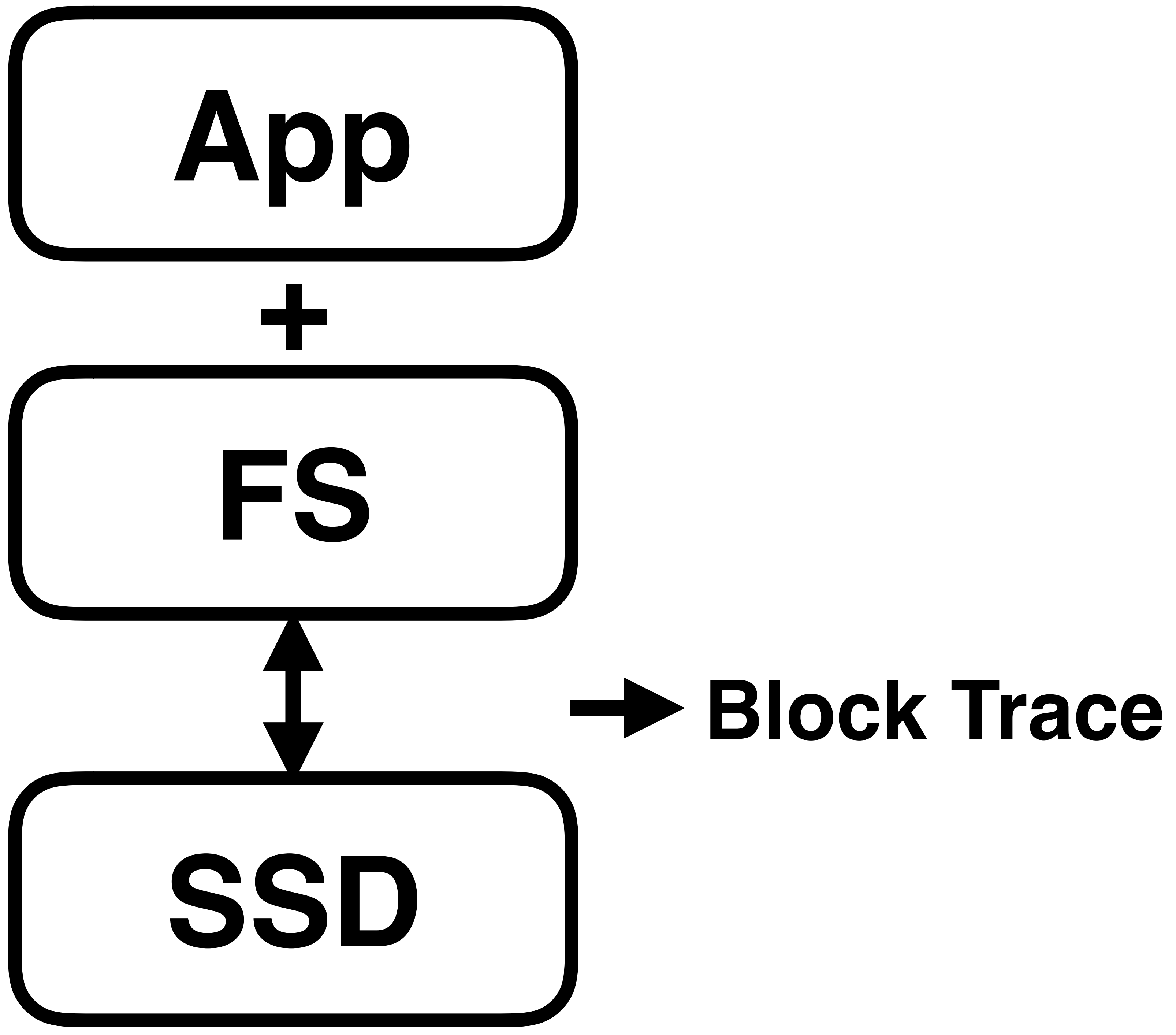
Do applications/file systems comply with the unwritten contract?

We conduct vertical analysis to find violations of SSD contract

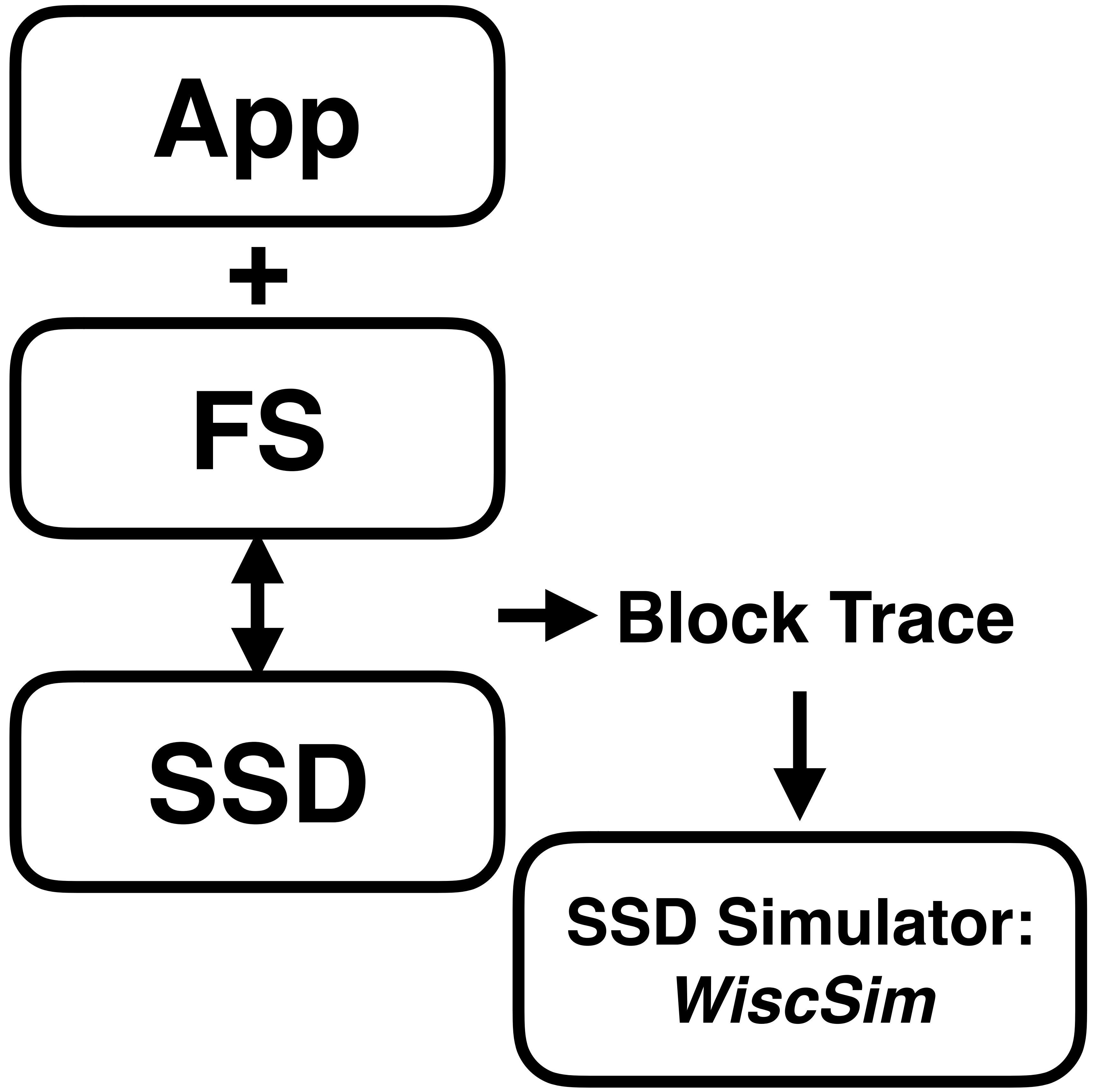
We conduct vertical analysis to find violations of SSD contract



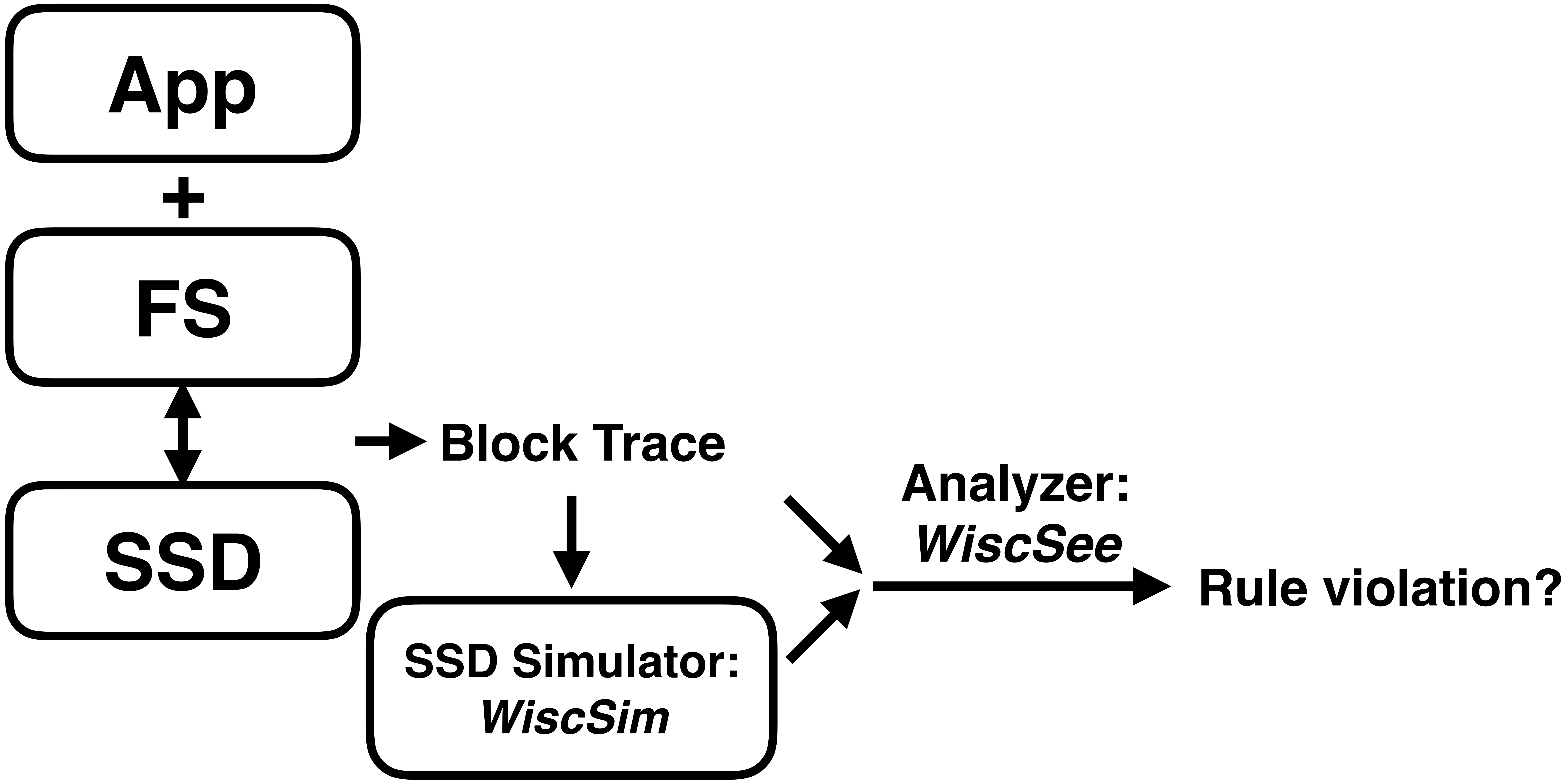
We conduct vertical analysis to find violations of SSD contract



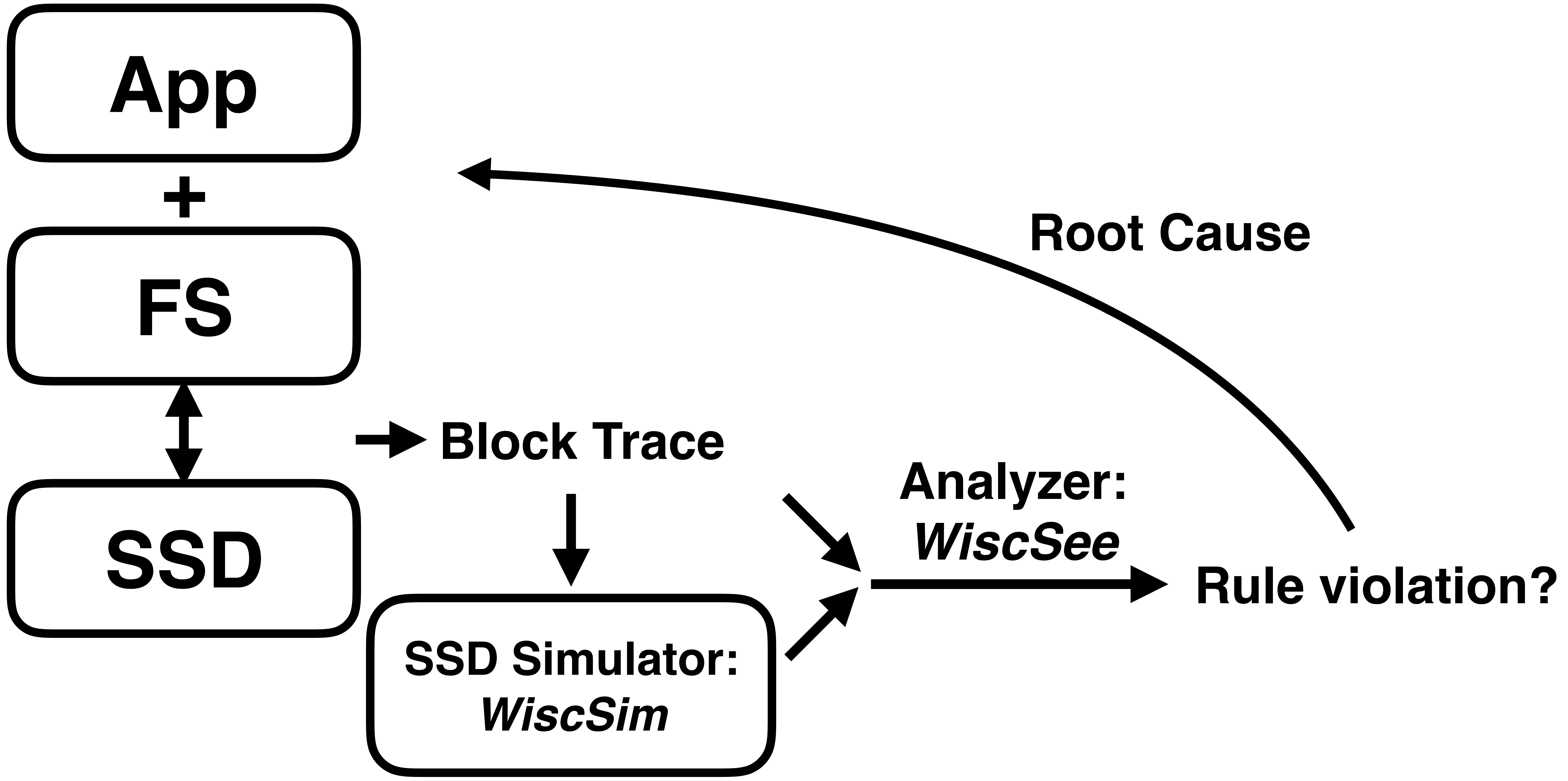
We conduct vertical analysis to find violations of SSD contract



We conduct vertical analysis to find violations of SSD contract



We conduct vertical analysis to find violations of SSD contract



2 of Our 24 Observations

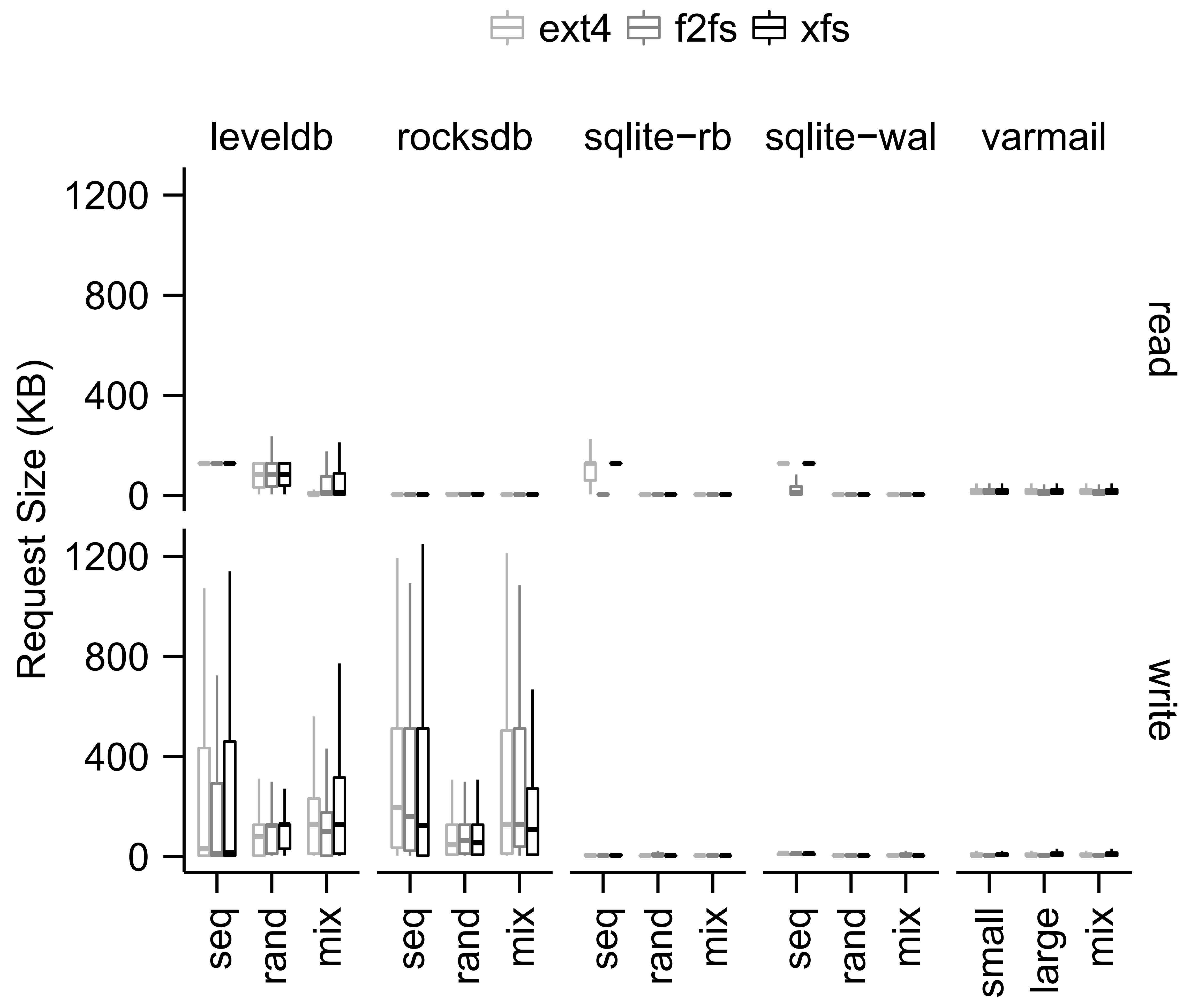
1. **Linux page cache limits request scale**
2. **F2FS incurs more GC overhead than traditional file systems**

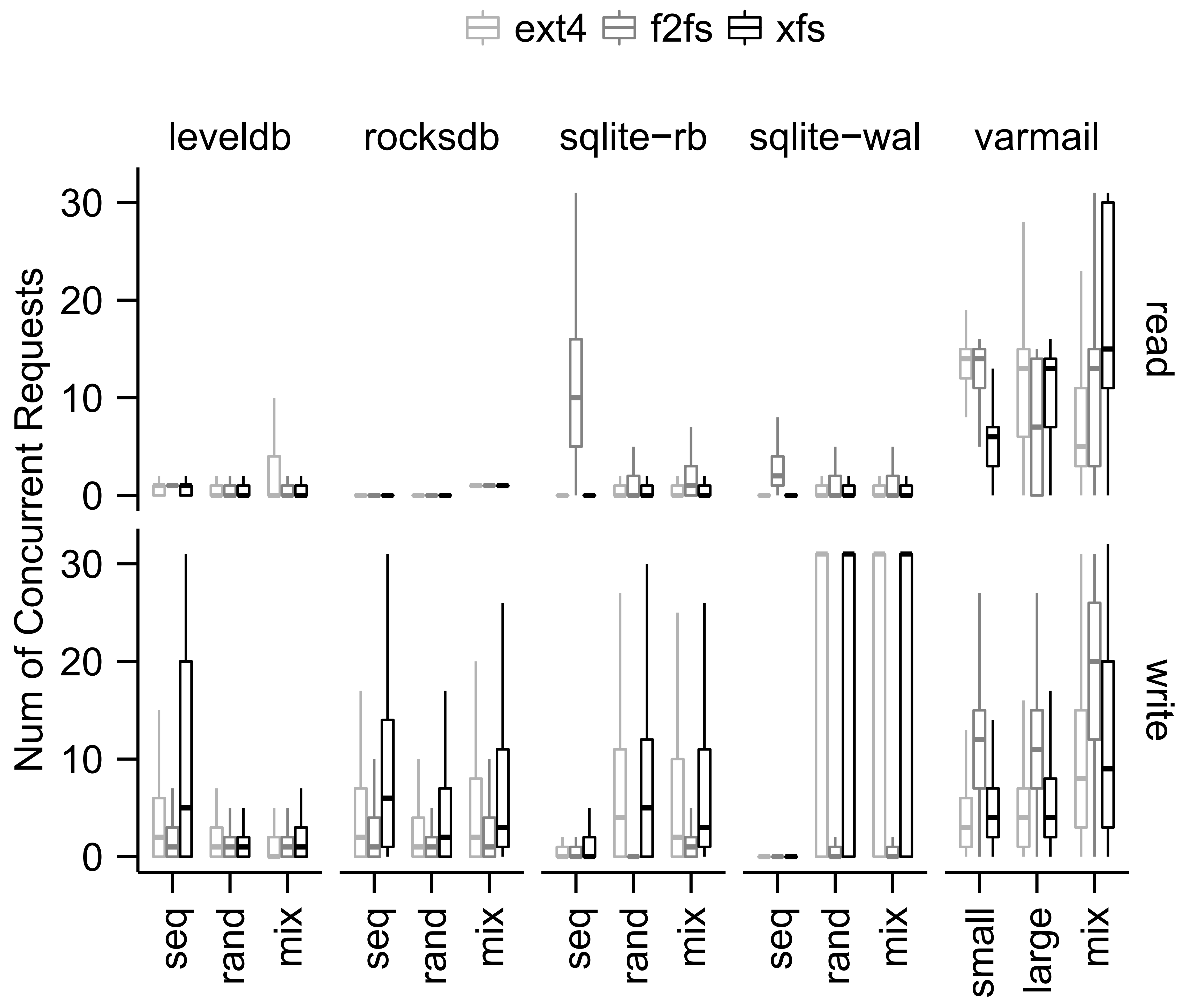
2 of Our 24 Observations

1. Linux page cache limits request scale

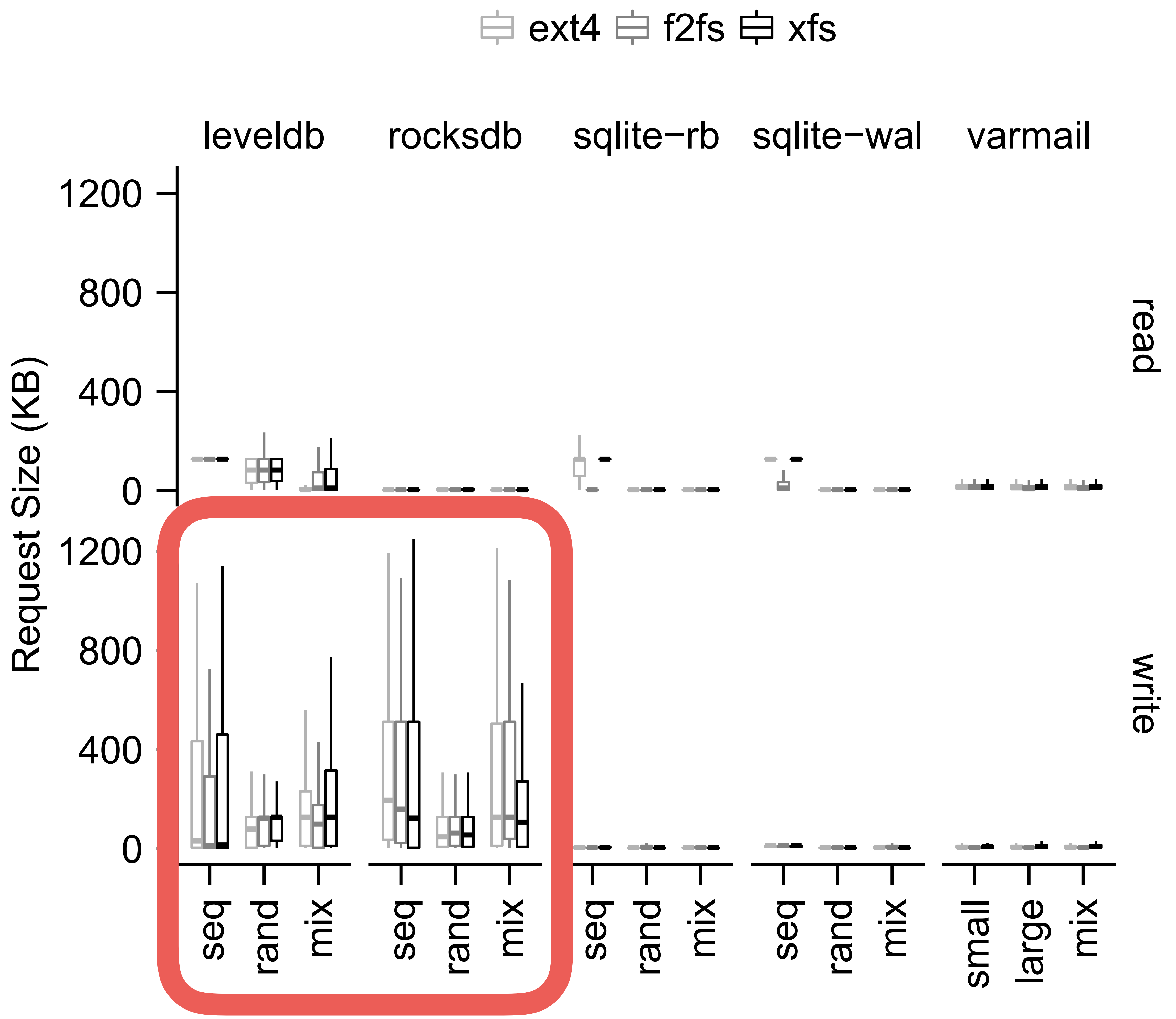
2. F2FS incurs more GC overhead than traditional file systems

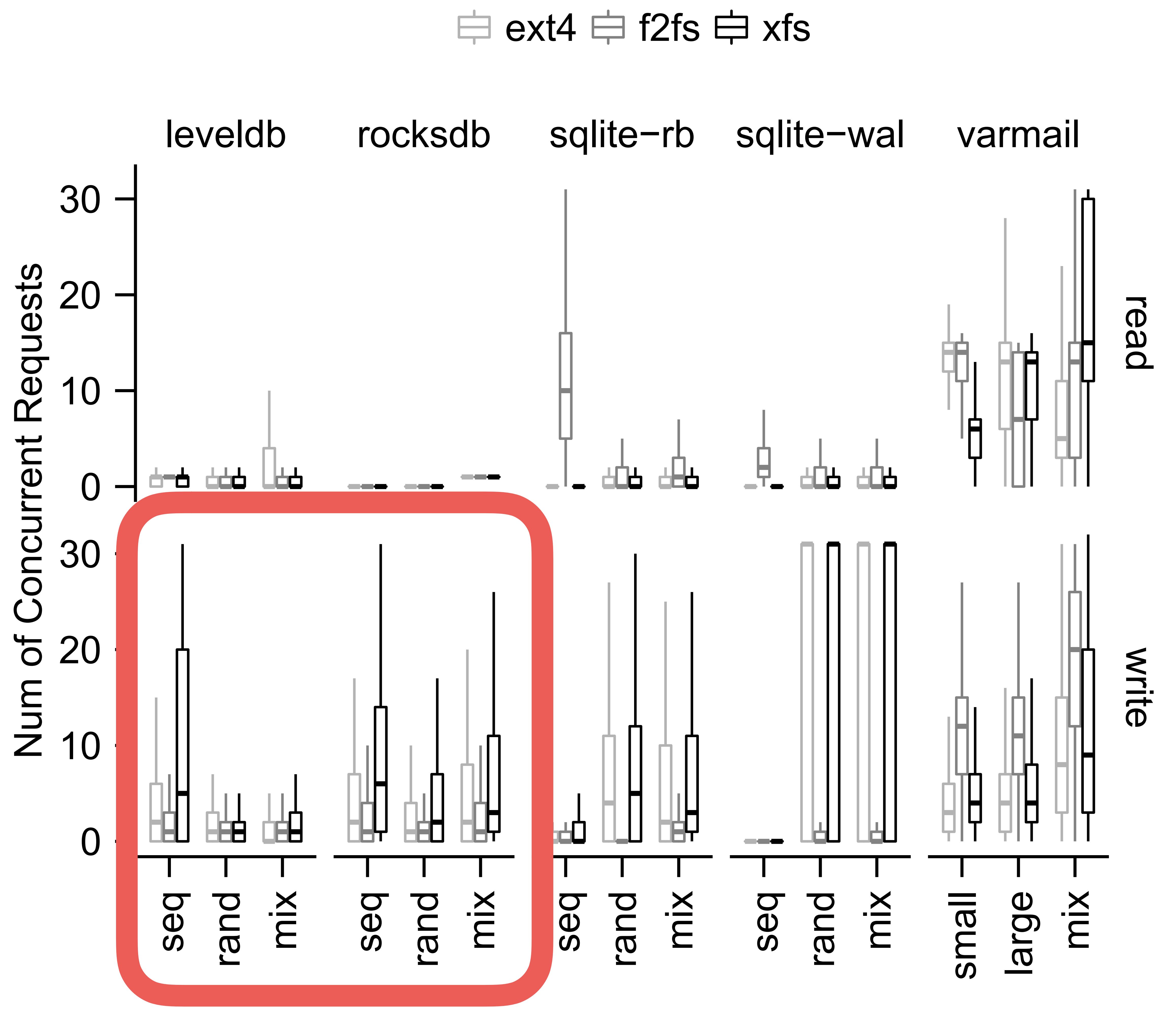
We evaluate request scale by request size and number of concurrent requests



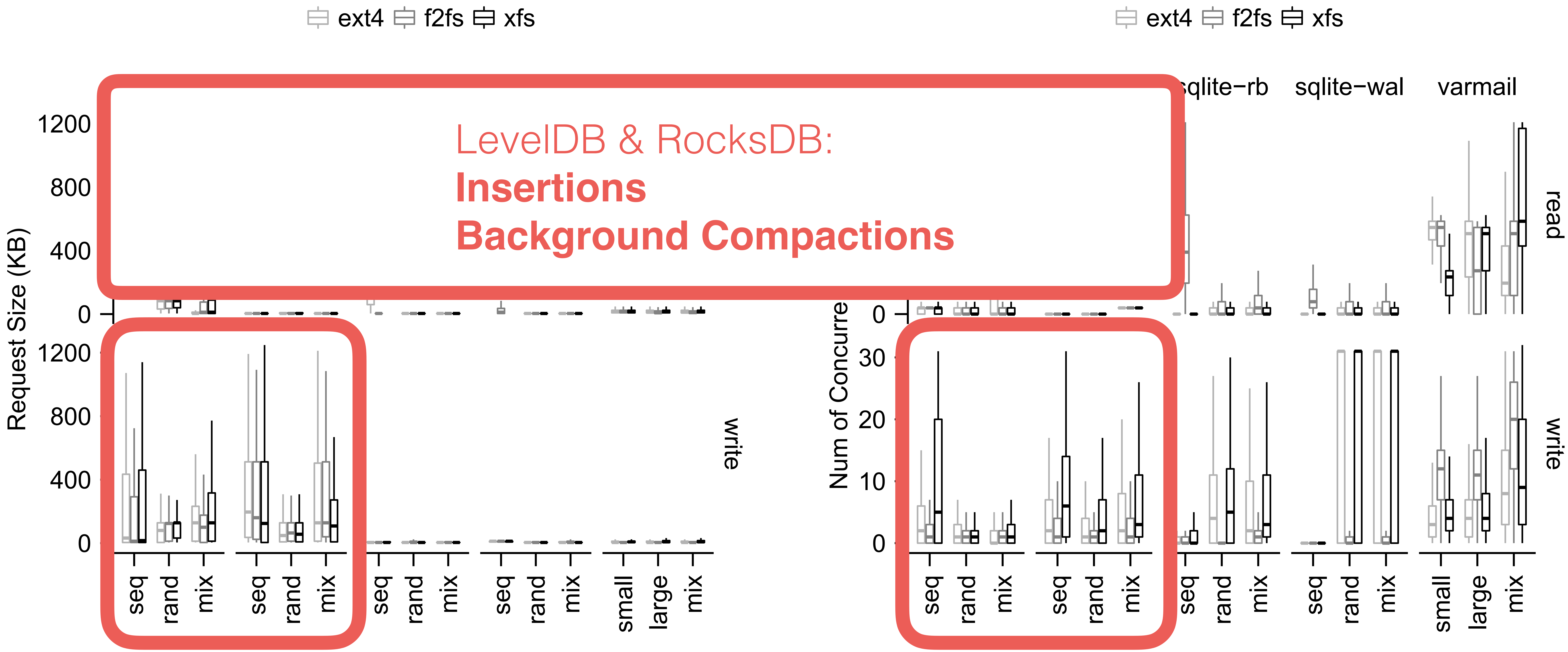


We evaluate request scale by request size and number of concurrent requests



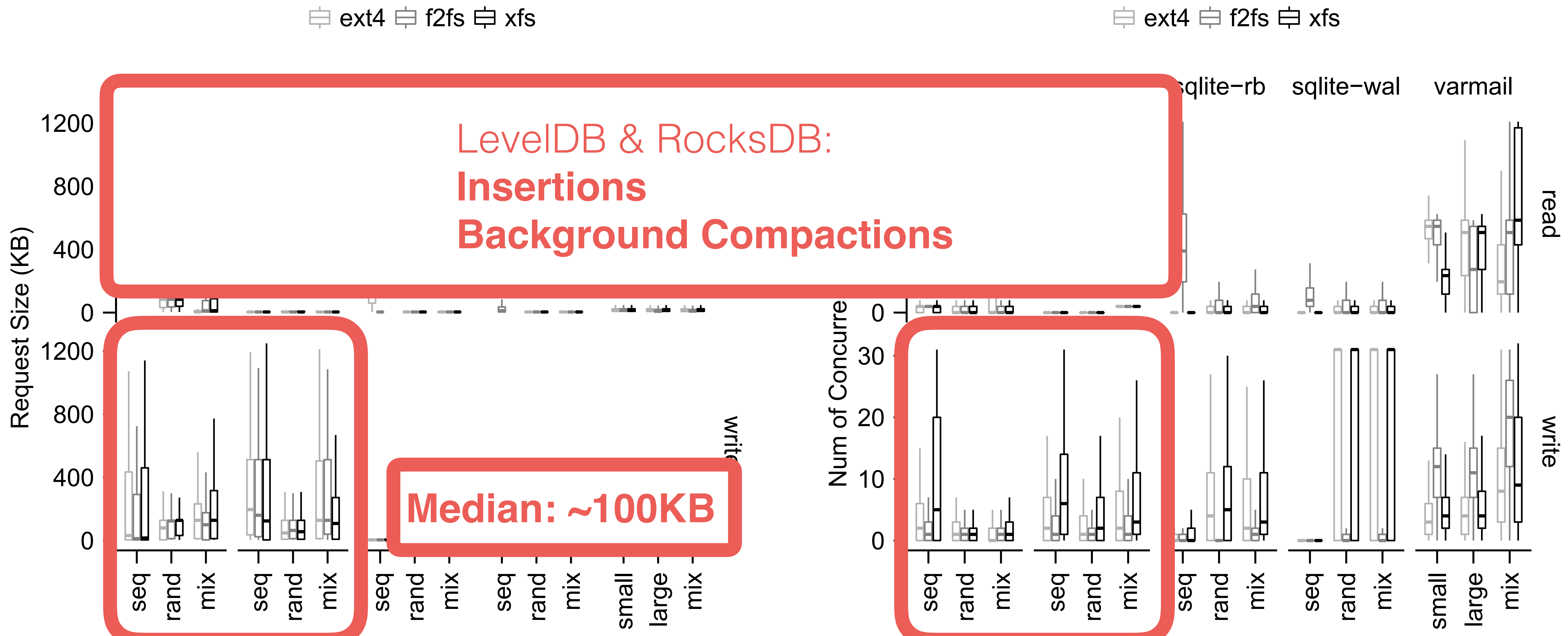


We evaluate request scale by request size and number of concurrent requests



LevelDB & RocksDB:
Insertions
Background Compactions

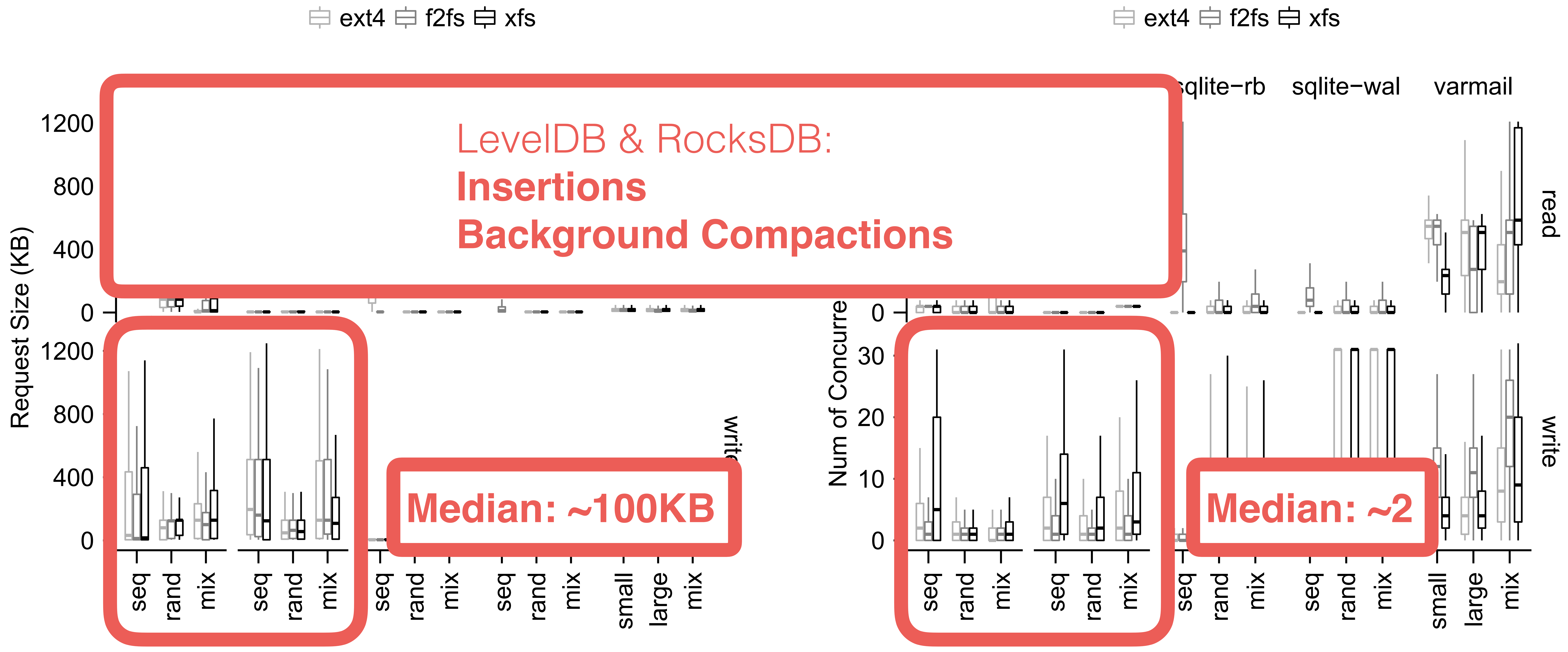
We evaluate request scale by request size and number of concurrent requests



LevelDB & RocksDB:
Insertions
Background Compactions

Median: ~100KB

We evaluate request scale by request size and number of concurrent requests



LevelDB & RocksDB:
Insertions
Background Compactions

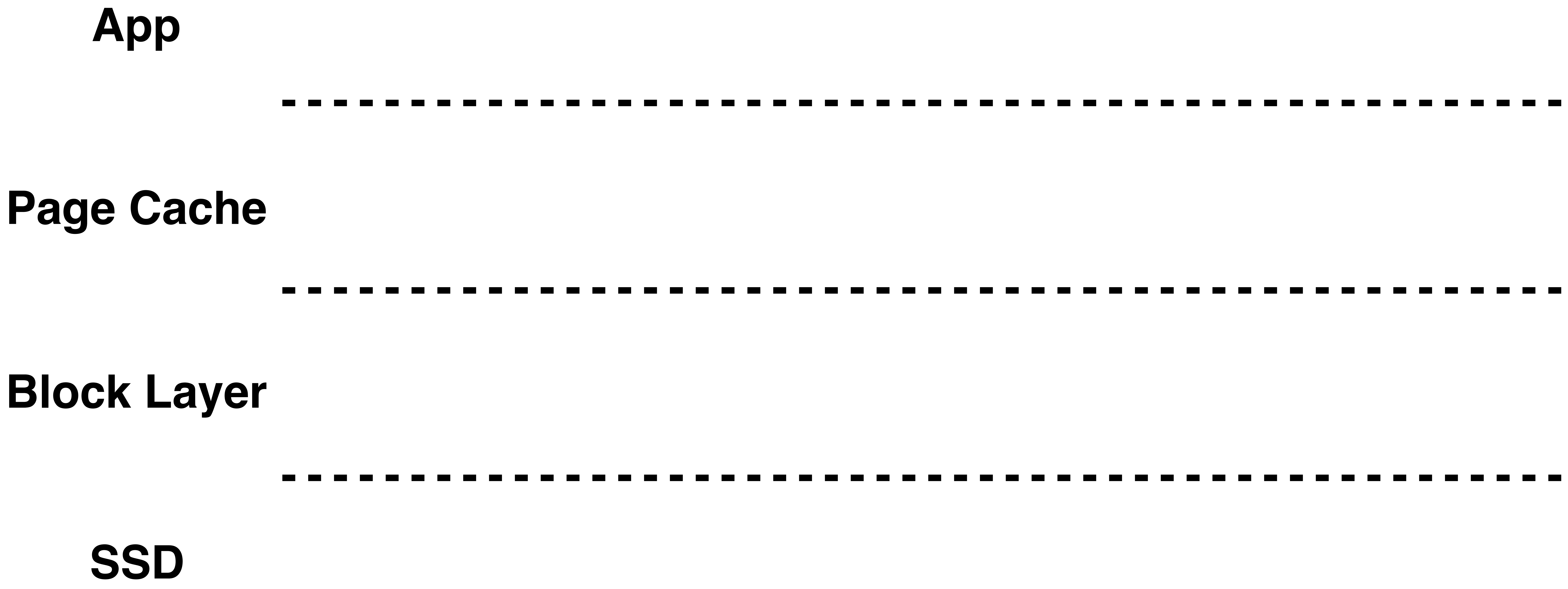
Median: ~100KB

Median: ~2

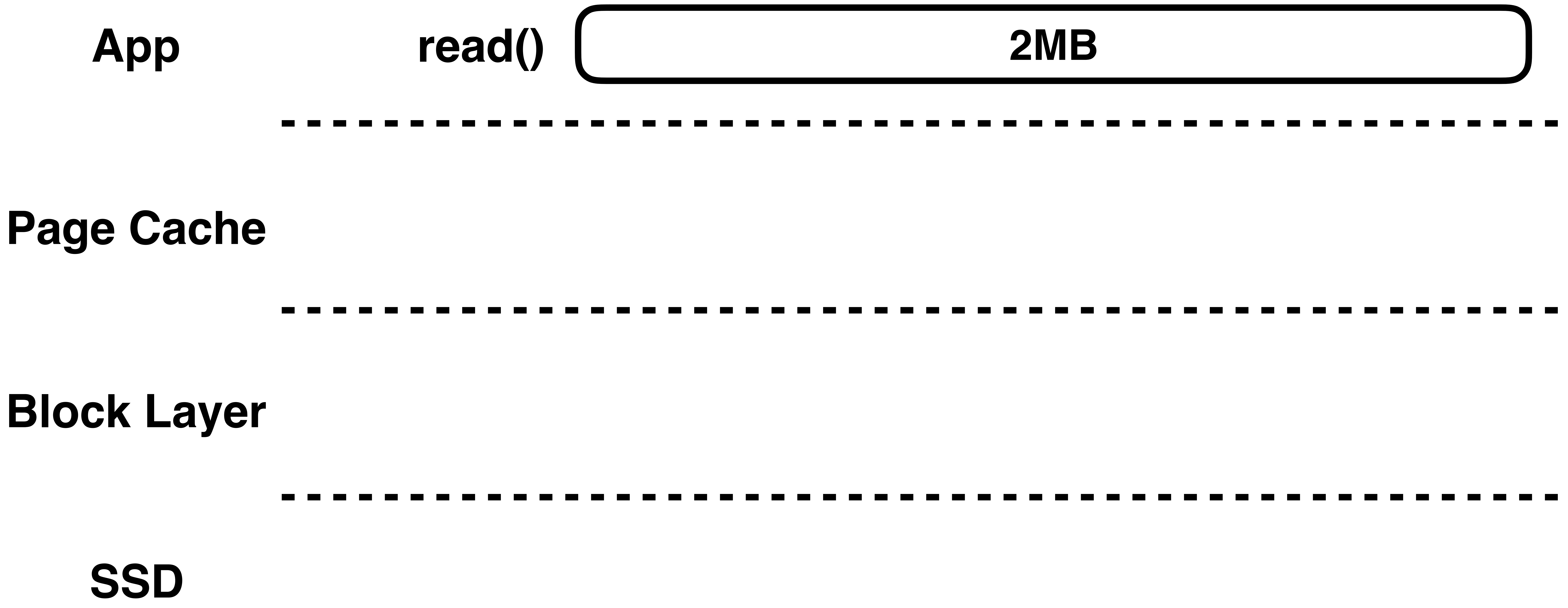
LevelDB and RocksDB can access files in large sizes.

Why was the request scale low?

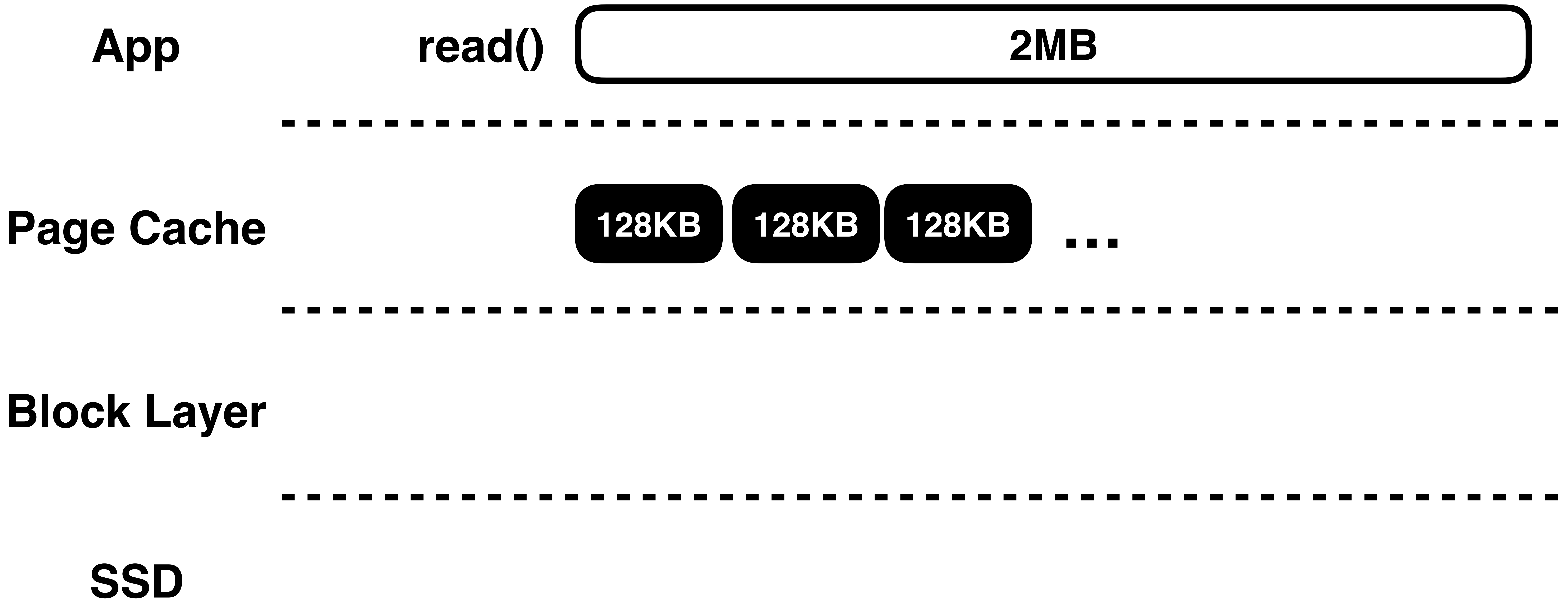
Buffered *read()*: Page cache implementation splits and serializes user requests



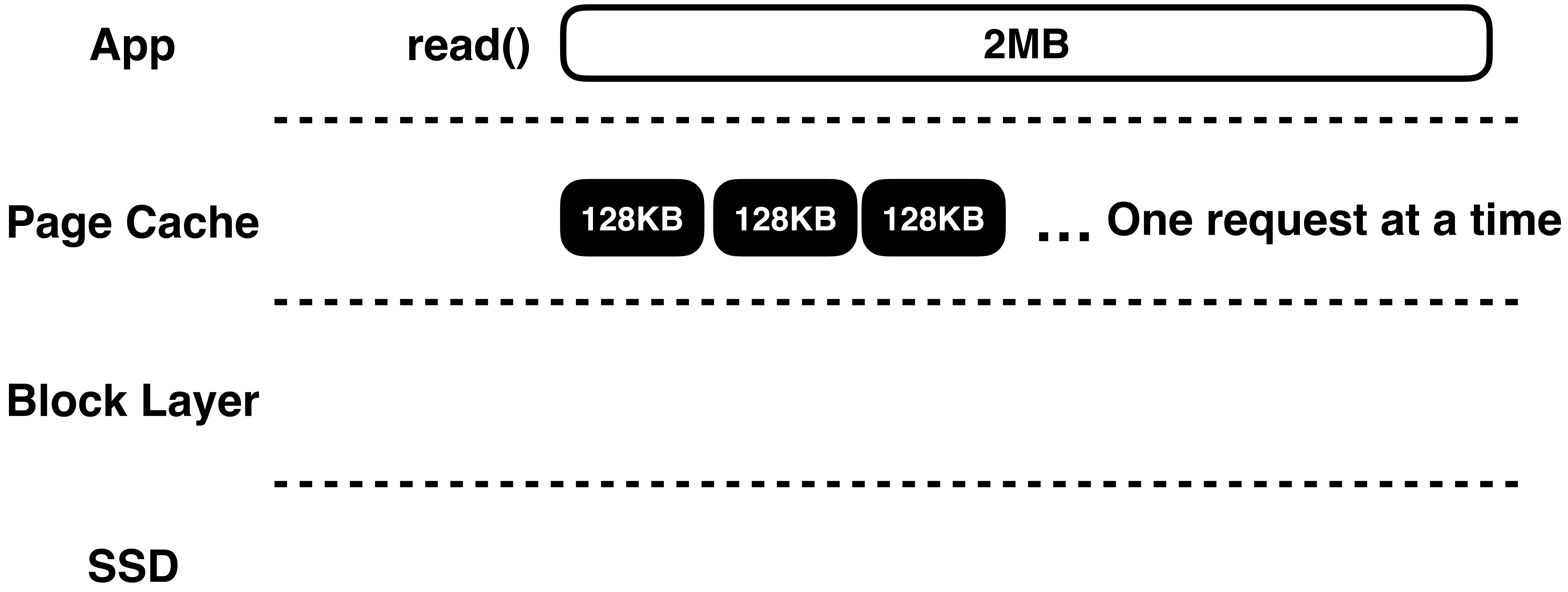
Buffered *read()*: Page cache implementation splits and serializes user requests



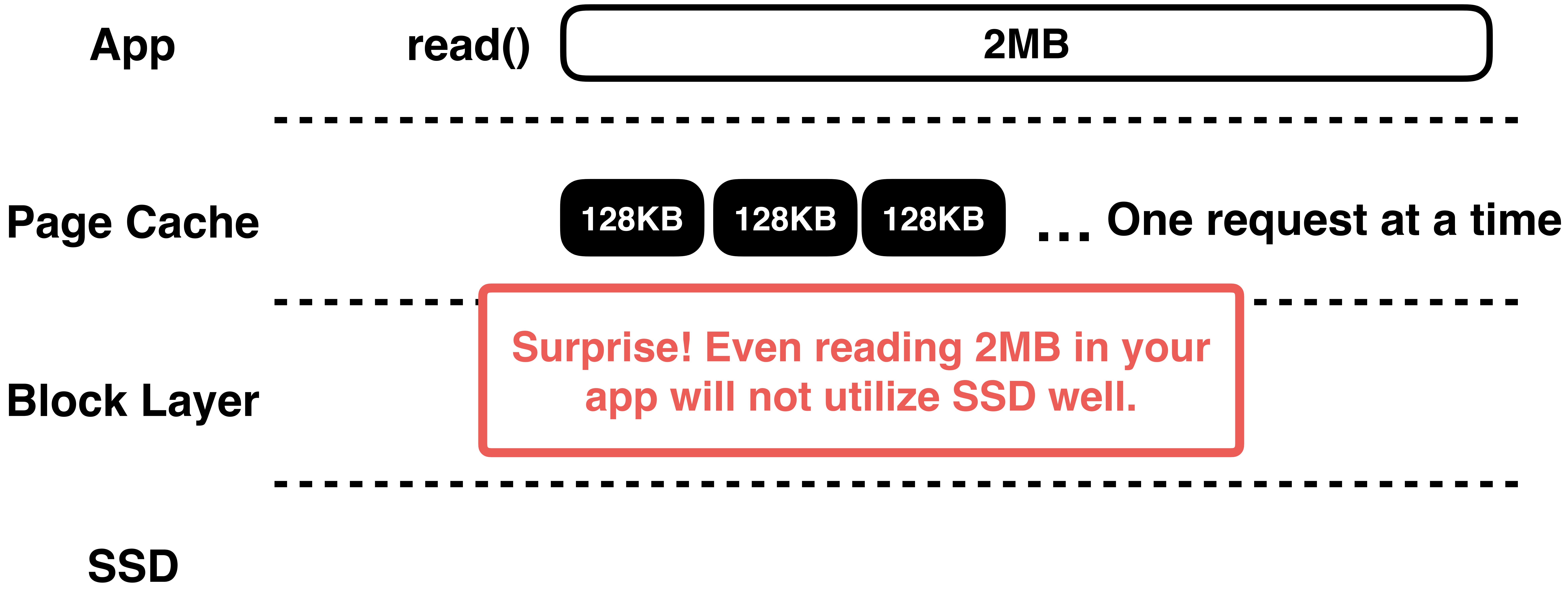
Buffered *read()*: Page cache implementation splits and serializes user requests



Buffered *read()*: Page cache implementation splits and serializes user requests



Buffered *read()*: Page cache implementation splits and serializes user requests



Surprise! Even reading 2MB in your app will not utilize SSD well.

Cause of Violation

Large reads are throttled by small prefetching (readahead).

2 of Our 24 Observations

➡ **1. Linux page cache limits request scale**

2. F2FS incurs more GC overhead than traditional file systems

2 of Our 24 Observations

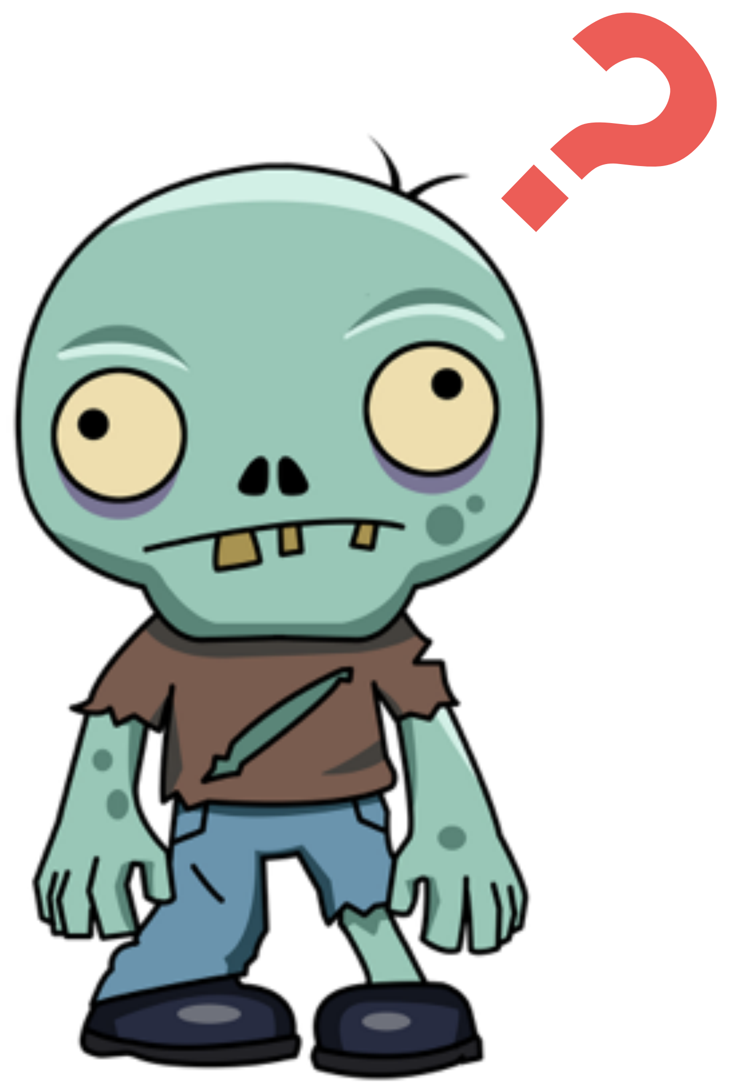
1. Linux page cache limits request scale

➡ 2. F2FS incurs more GC overhead than traditional file systems

We study GC (rule #3: grouping by death time) by zombie curves

We study GC (rule #3: grouping by death time) by zombie curves

What’s a zombie curve?



We study GC (rule #3: grouping by death time) by zombie curves

What’s a zombie curve?

We study GC (rule #3: grouping by death time) by zombie curves

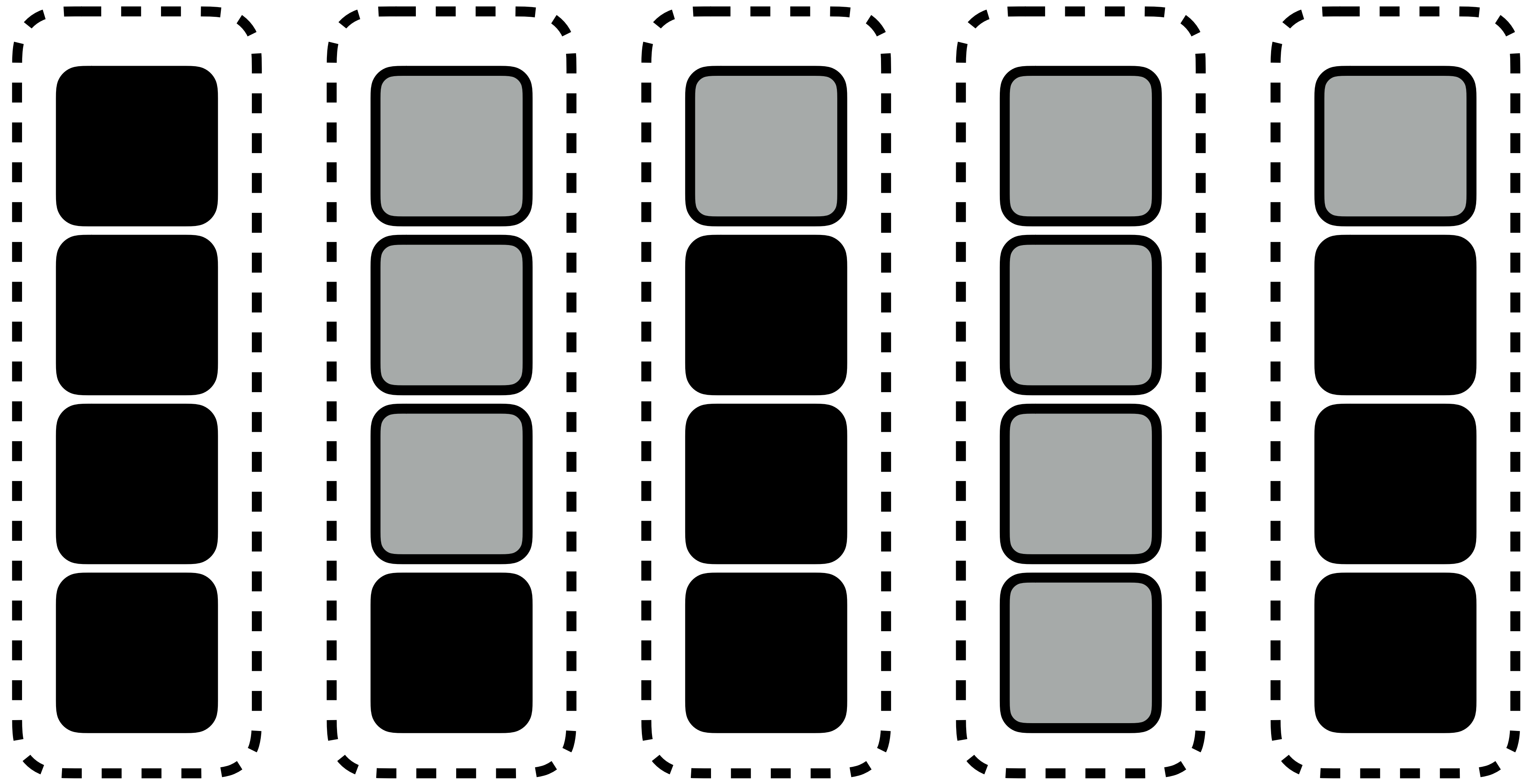
What’s a zombie curve?

Run workloads with **infinite** space over-provisioning

We study GC (rule #3: grouping by death time) by zombie curves

What's a zombie curve?

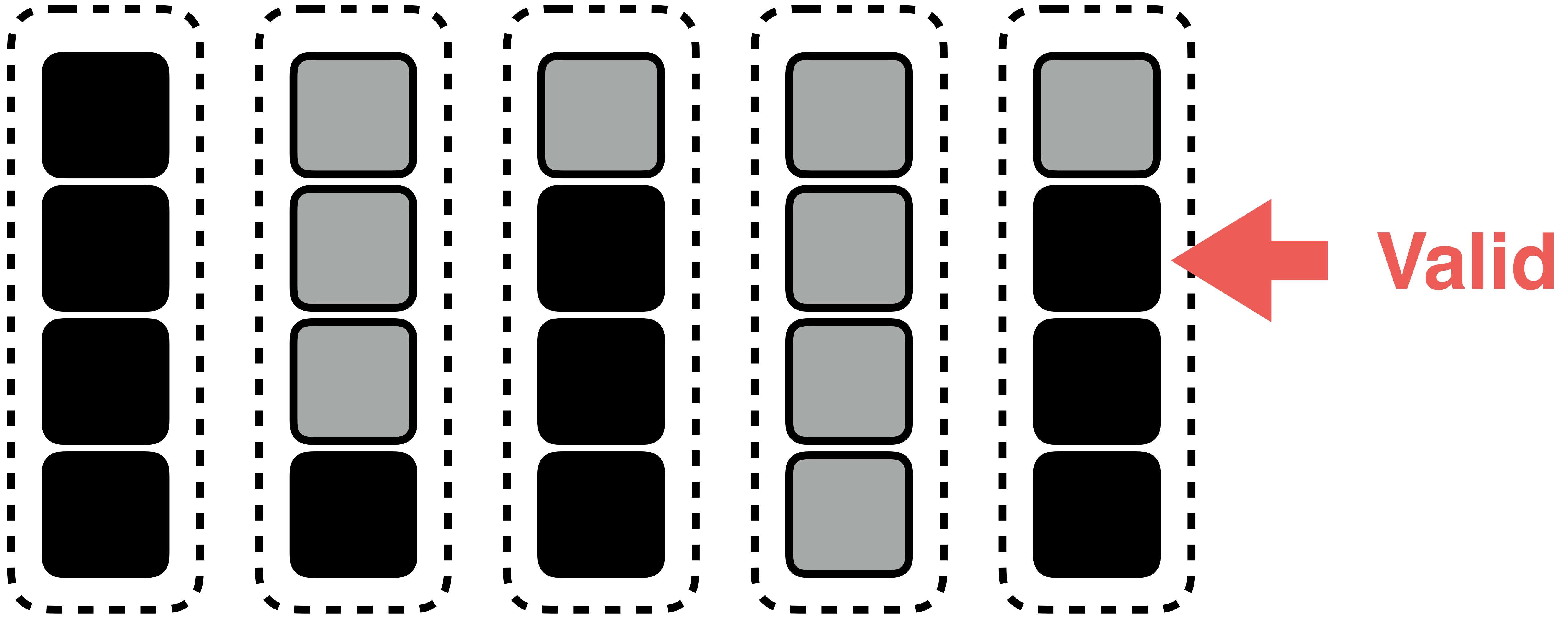
Run workloads with **infinite** space over-provisioning



We study GC (rule #3: grouping by death time) by zombie curves

What's a zombie curve?

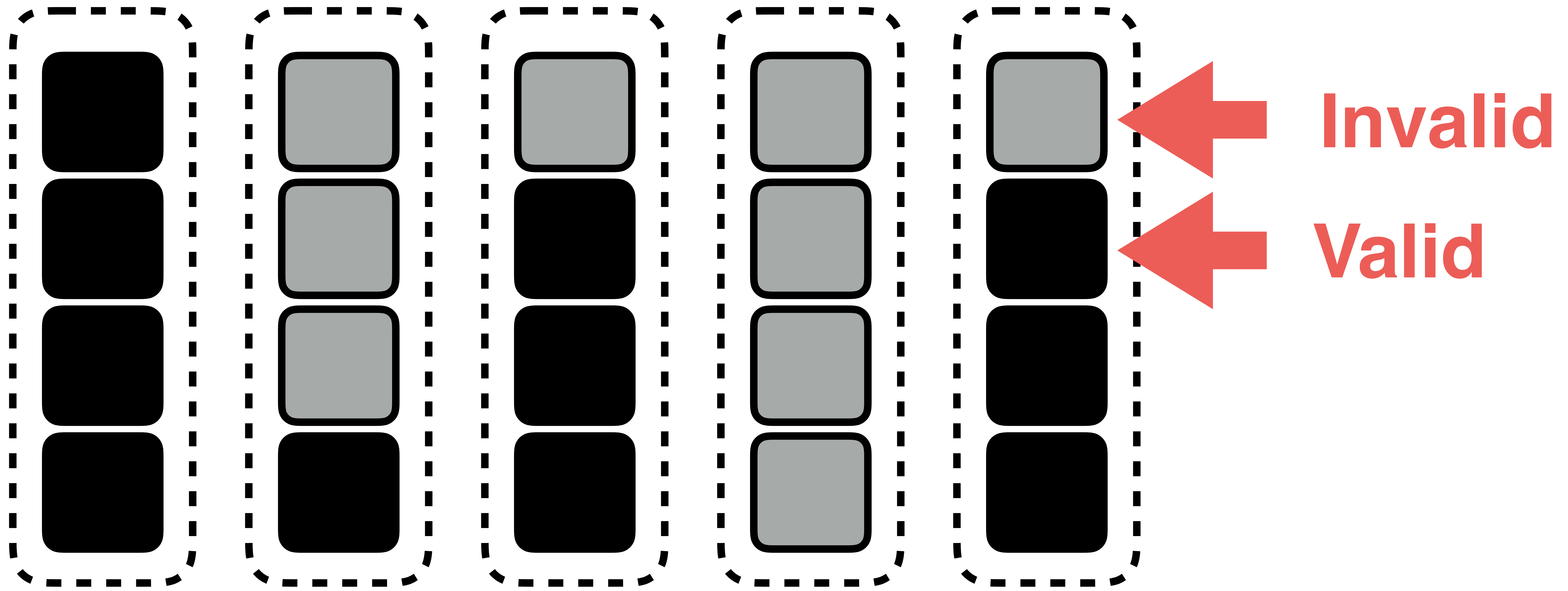
Run workloads with **infinite** space over-provisioning



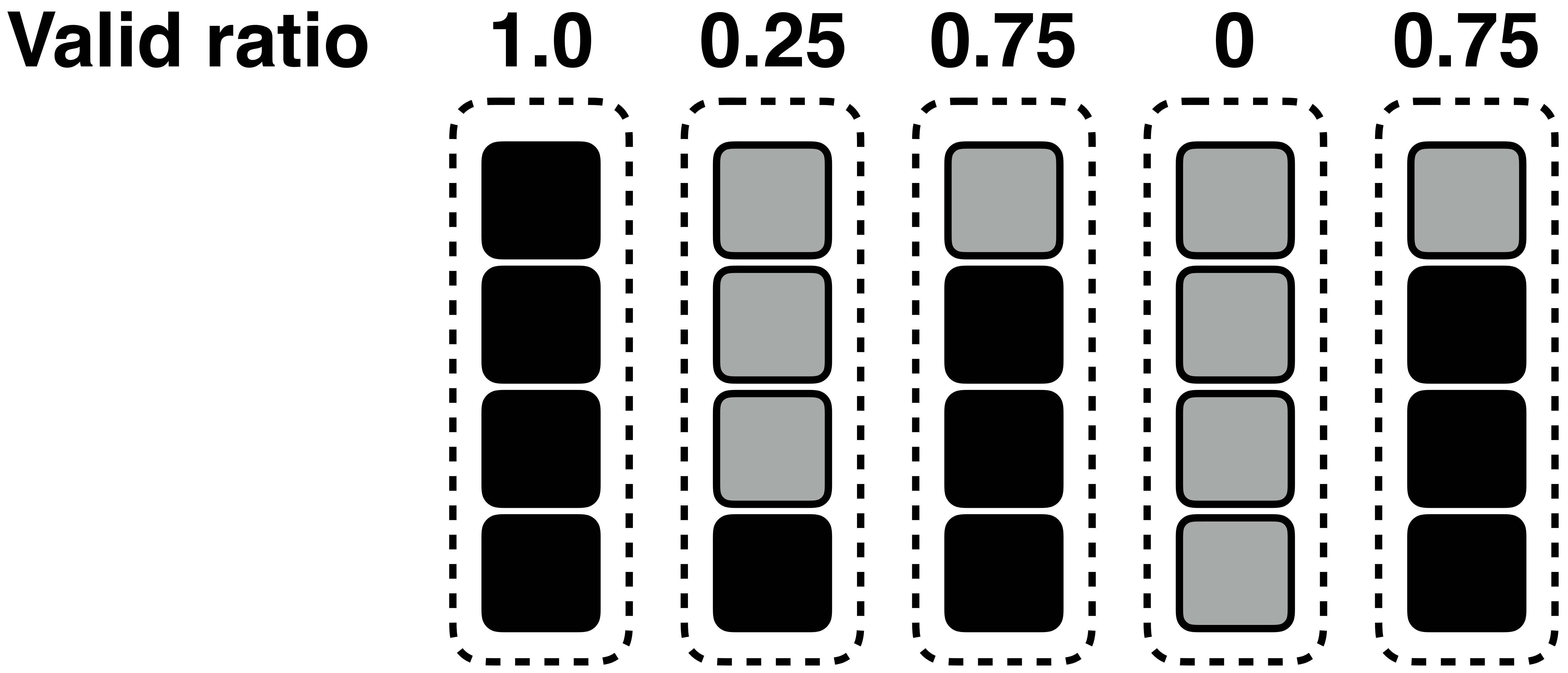
We study GC (rule #3: grouping by death time) by zombie curves

What's a zombie curve?

Run workloads with **infinite** space over-provisioning

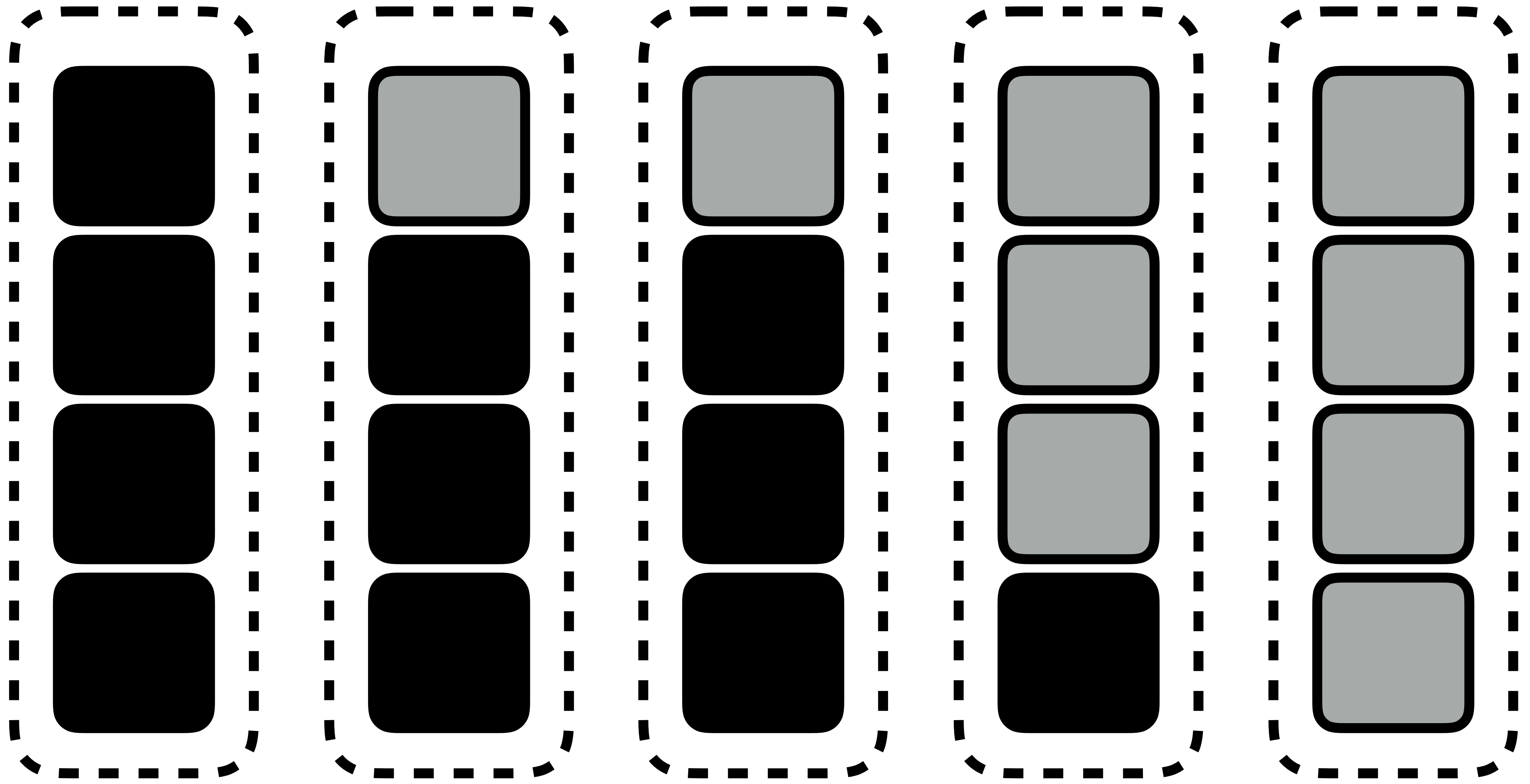


We study GC (rule #3: grouping by death time) by zombie curves

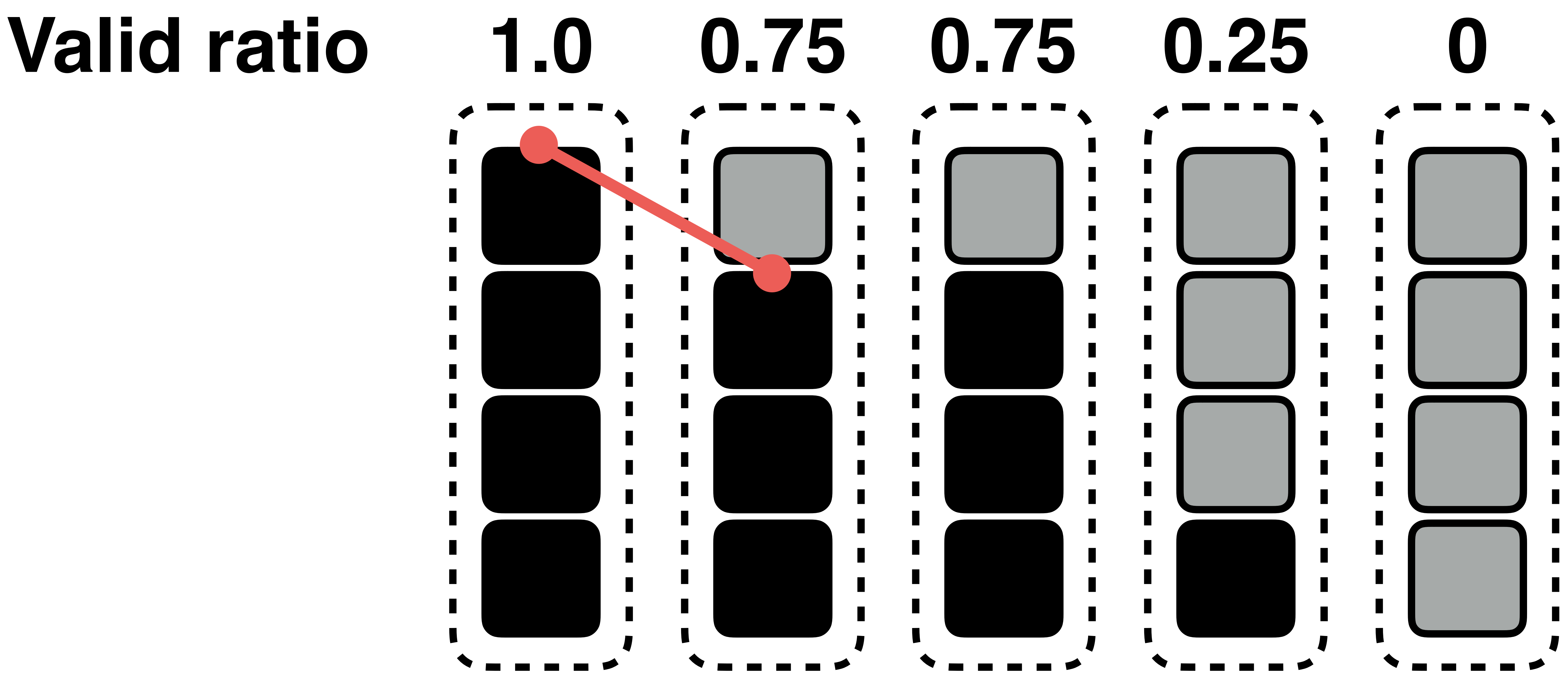


We study GC (rule #3: grouping by death time) by zombie curves

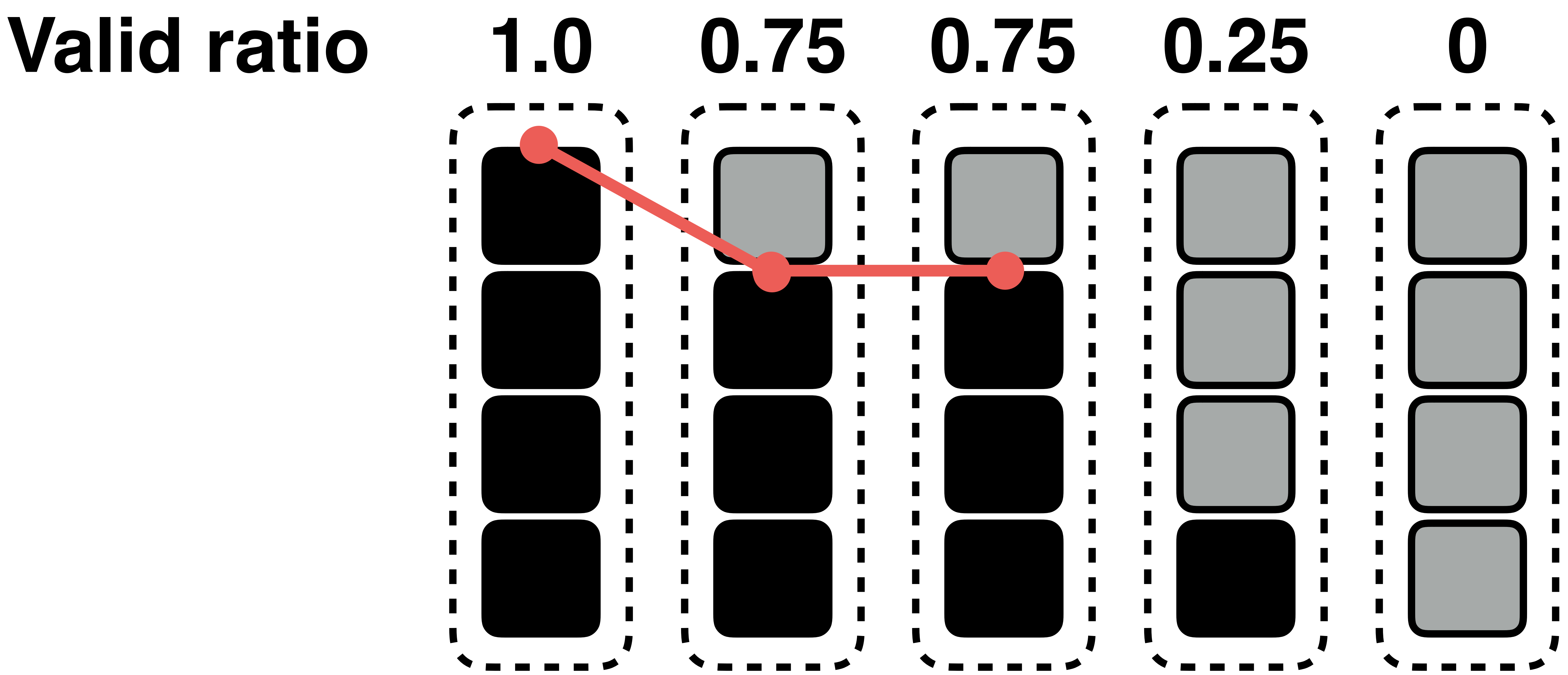
Valid ratio 1.0 0.75 0.75 0.25 0



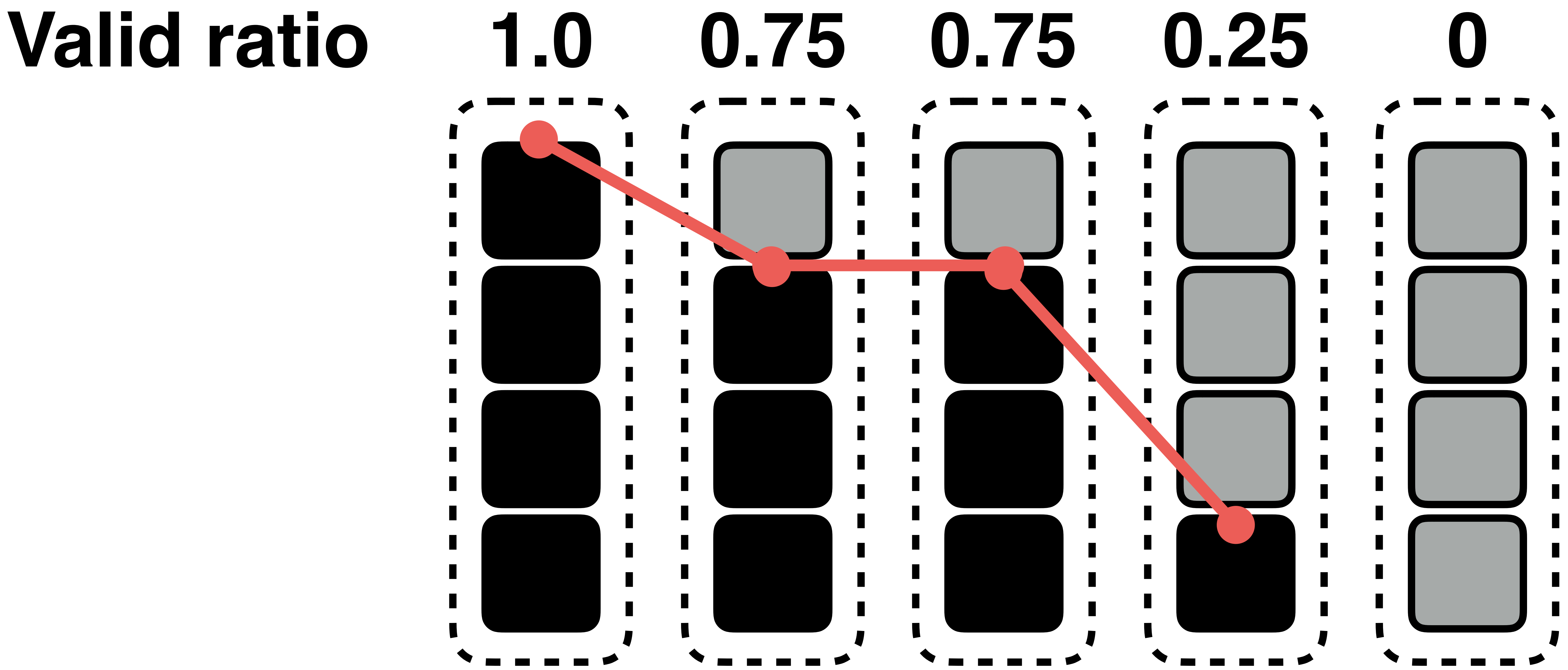
We study GC (rule #3: grouping by death time) by zombie curves



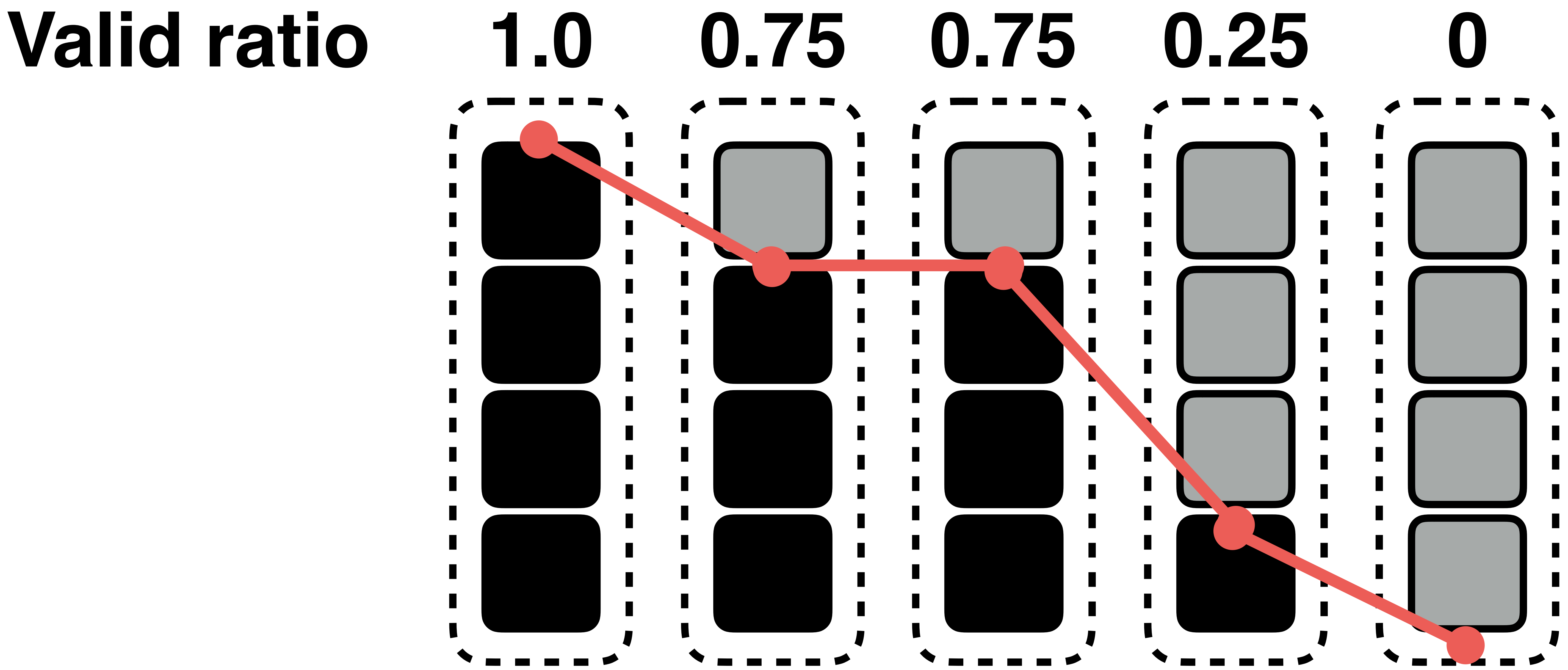
We study GC (rule #3: grouping by death time) by zombie curves



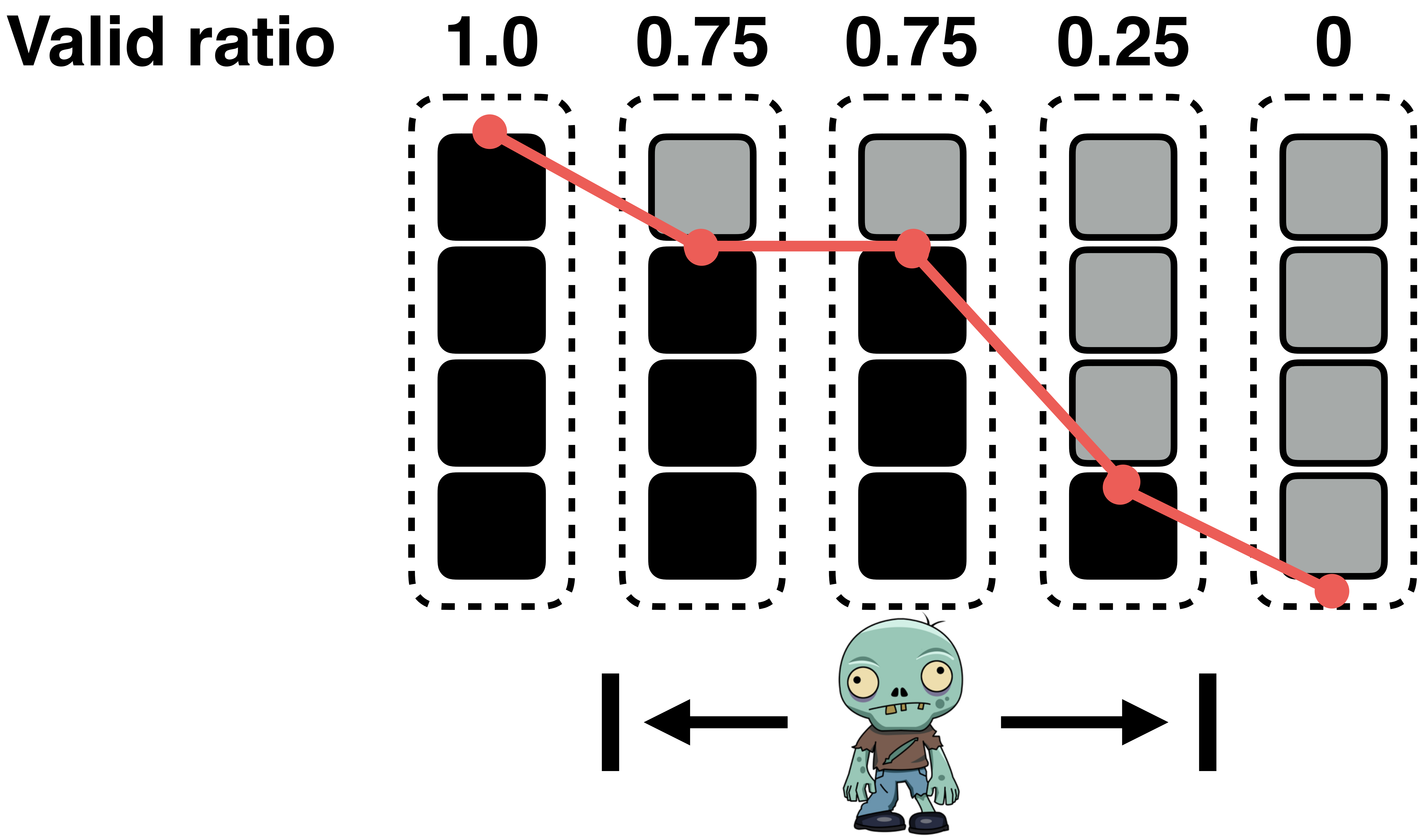
We study GC (rule #3: grouping by death time) by zombie curves



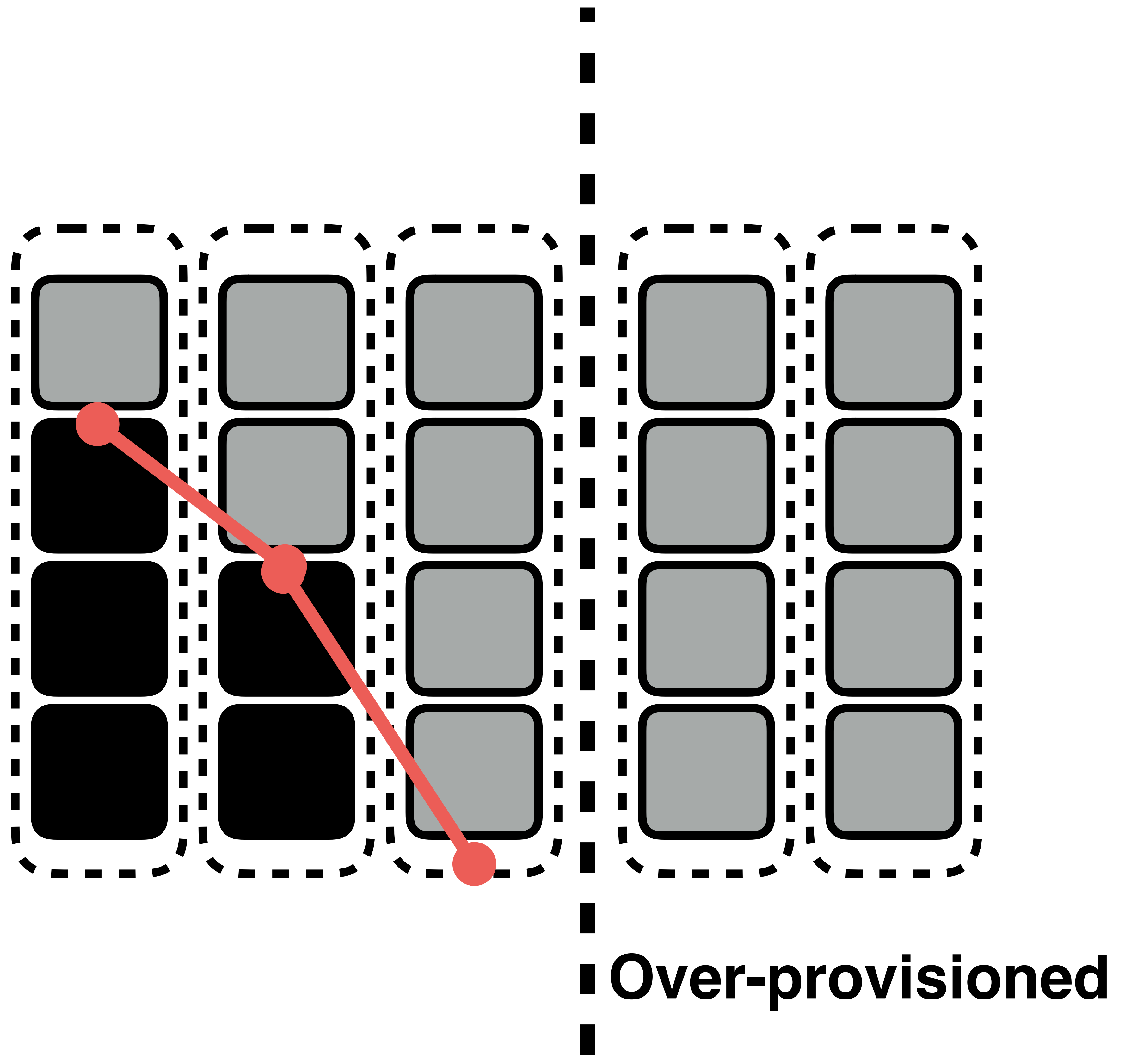
We study GC (rule #3: grouping by death time) by zombie curves

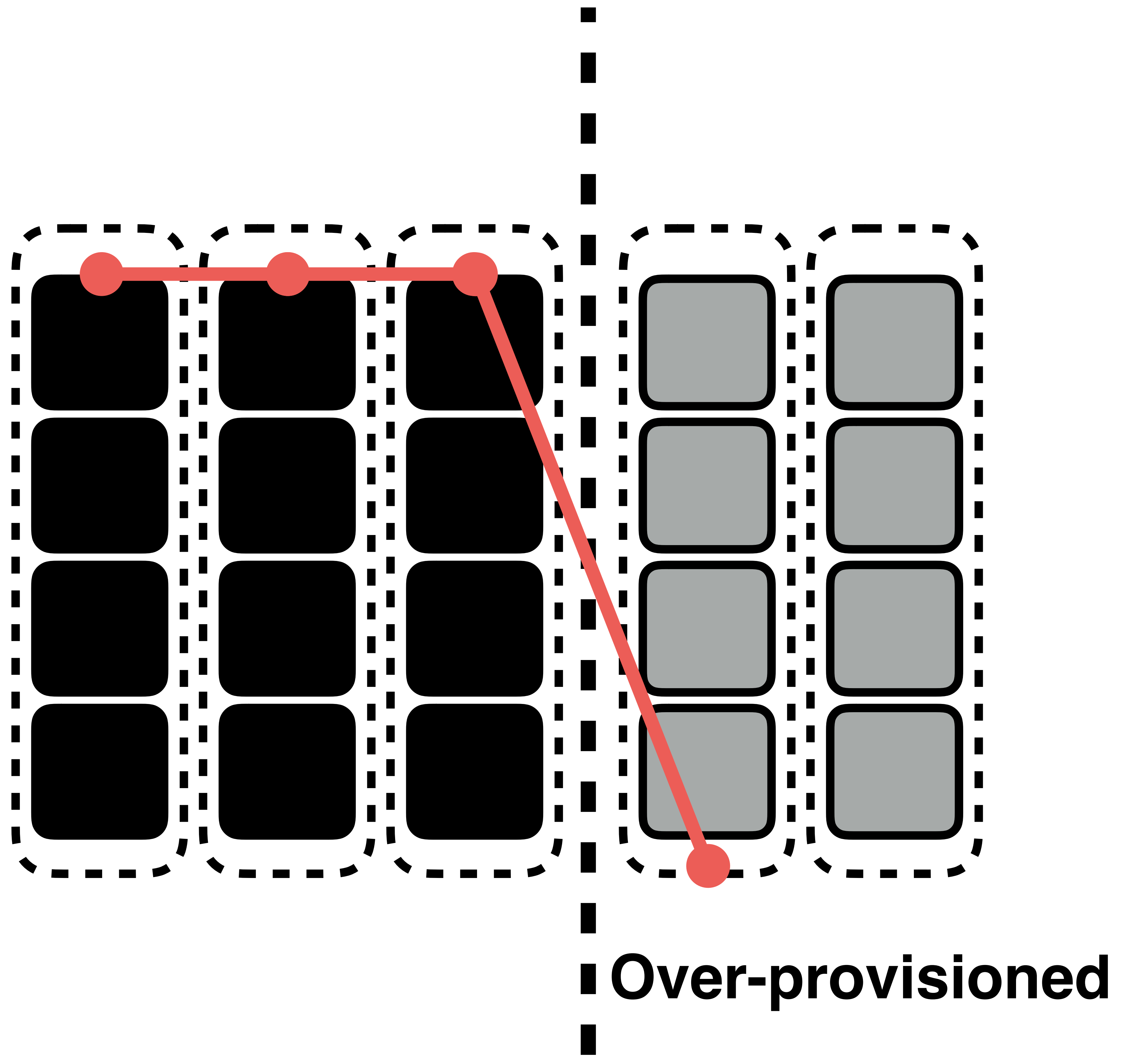


We study GC (rule #3: grouping by death time) by zombie curves

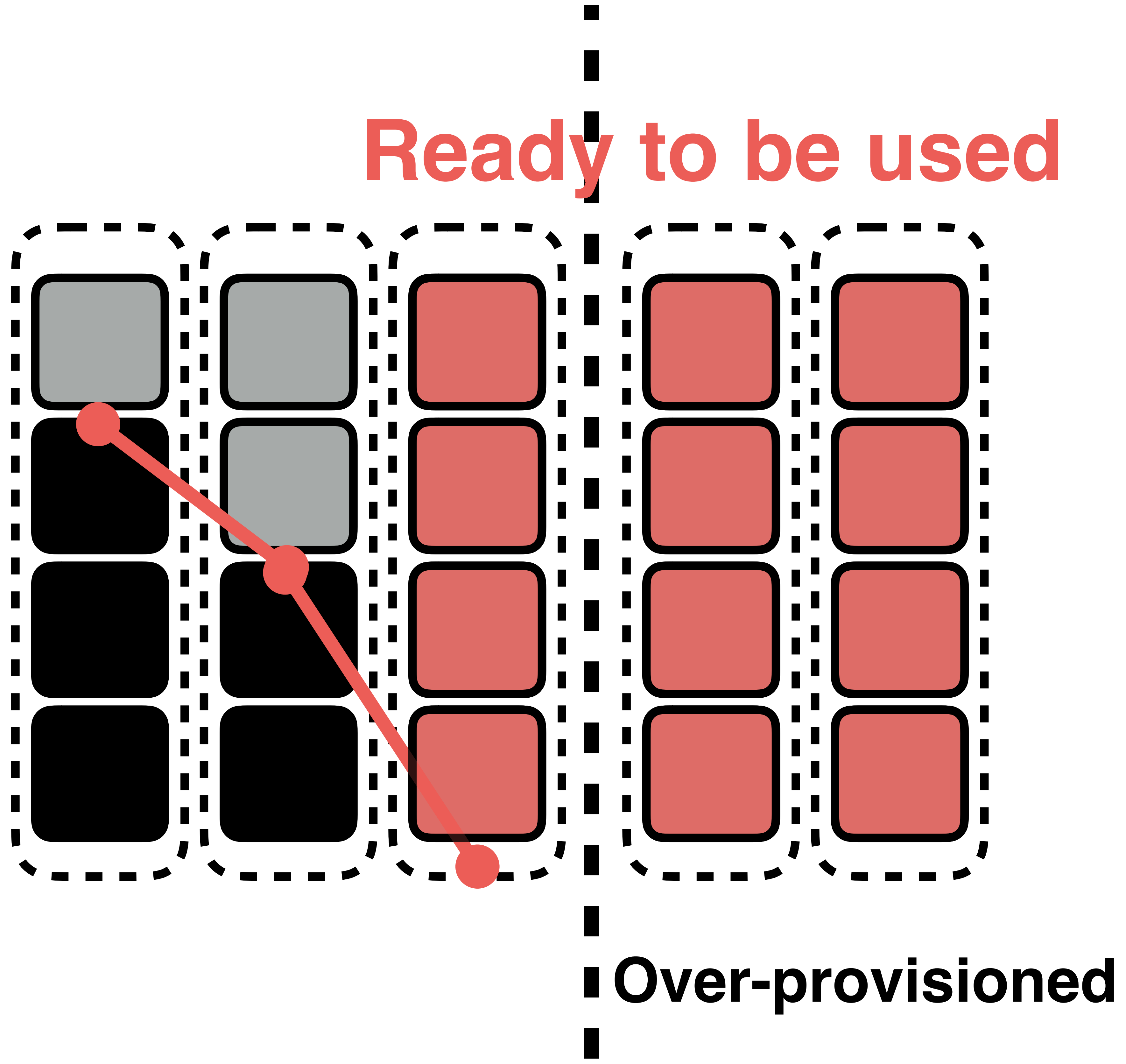


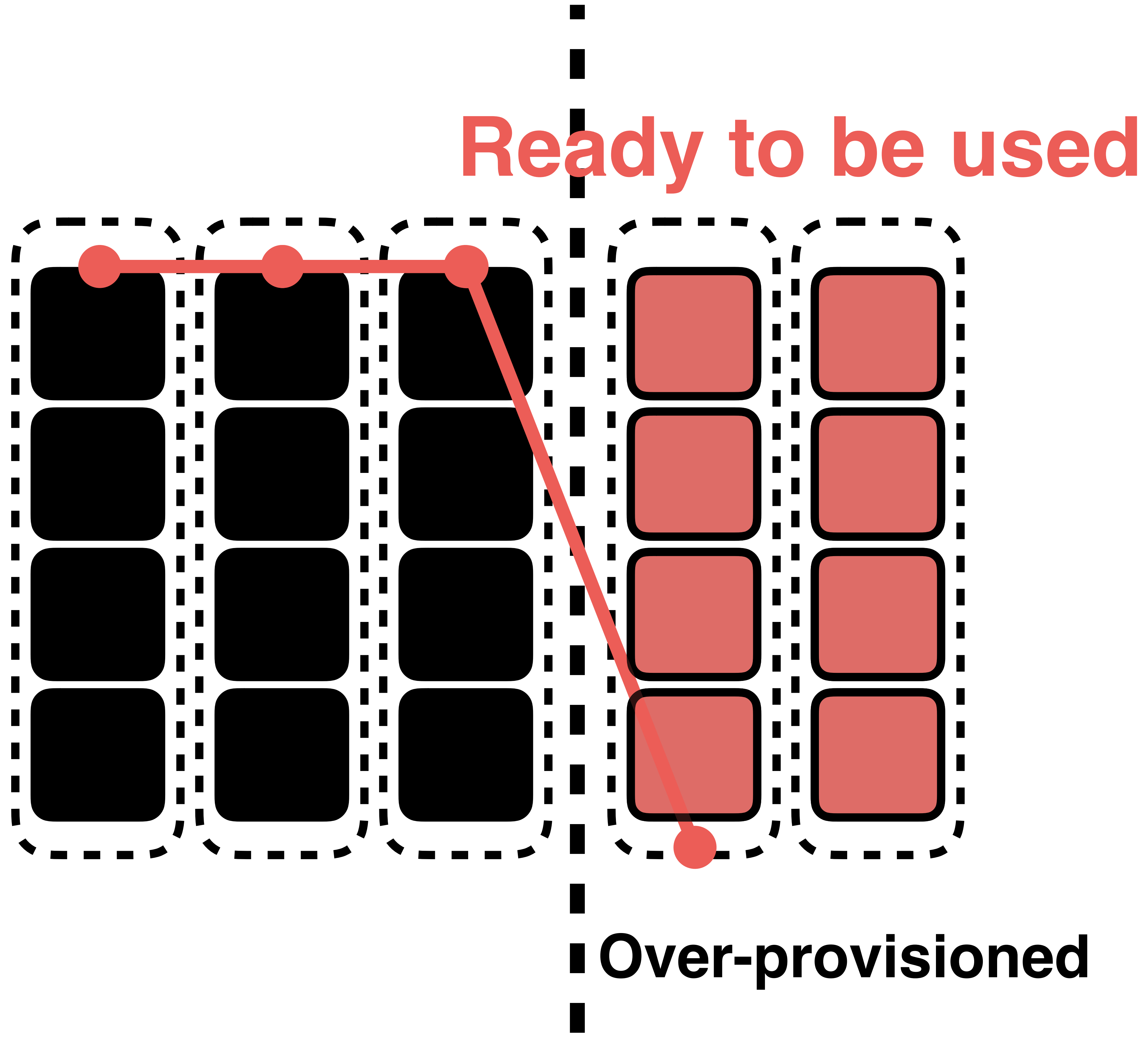
What's a good zombie curve?



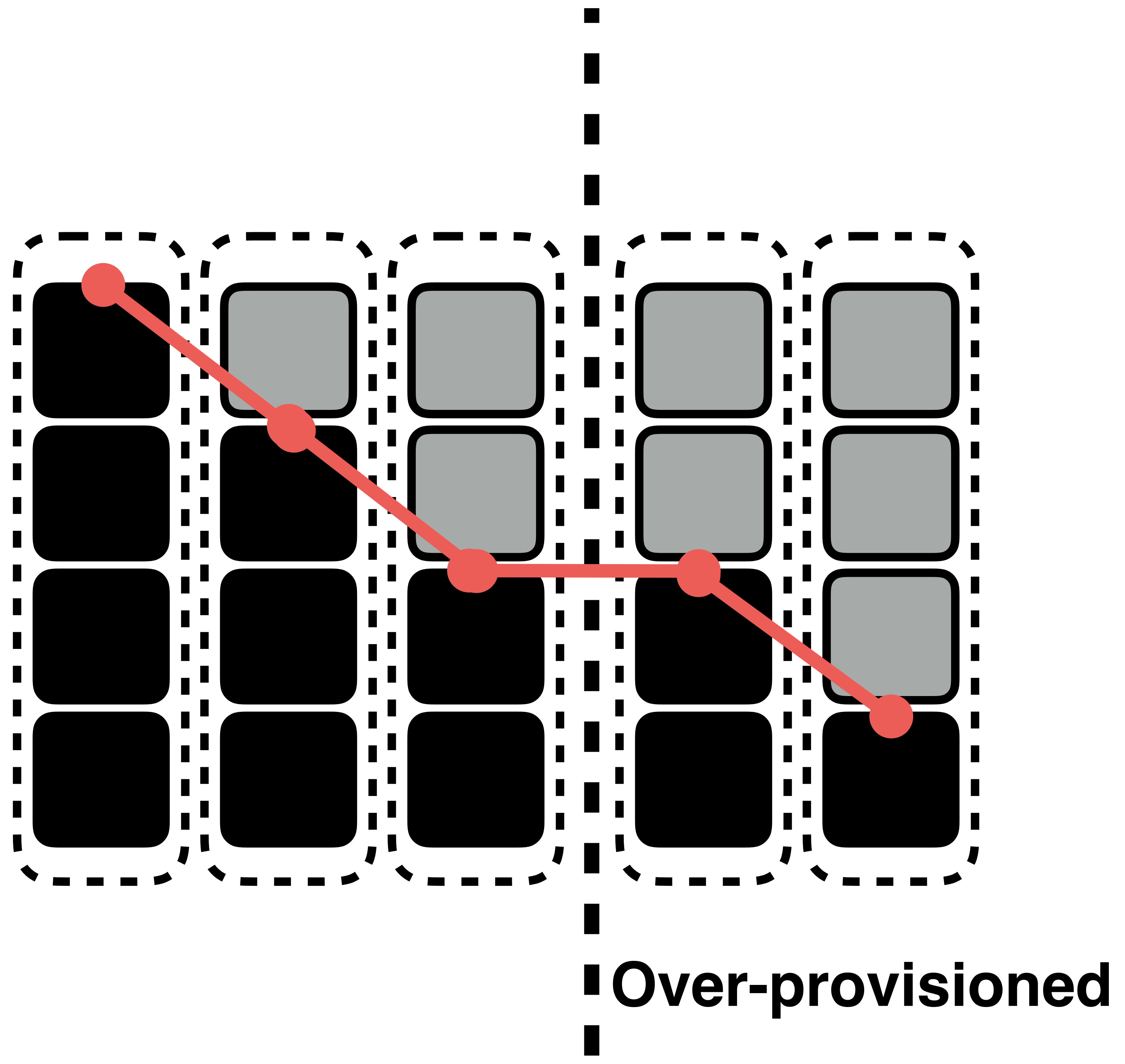


What’s a good zombie curve?

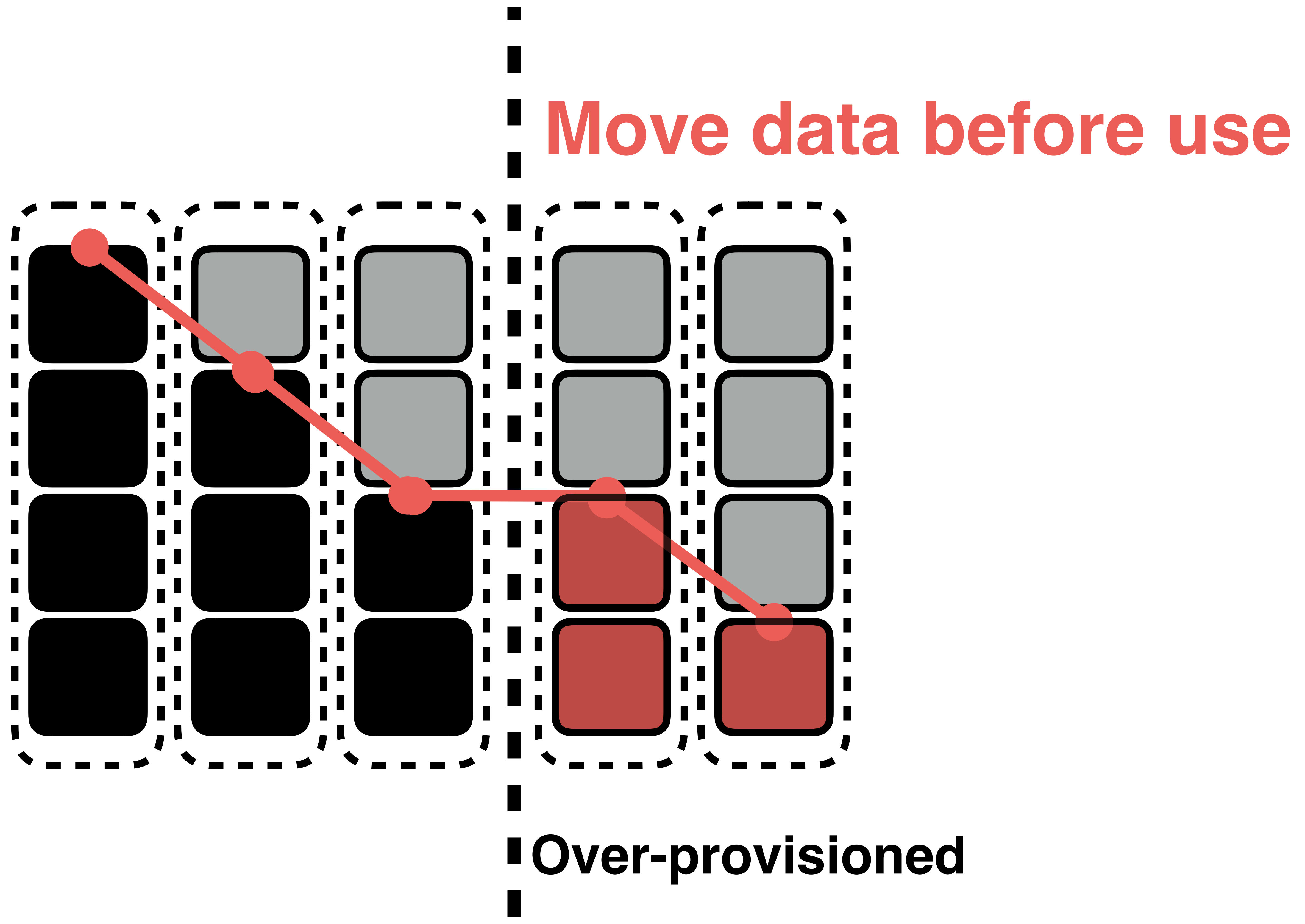




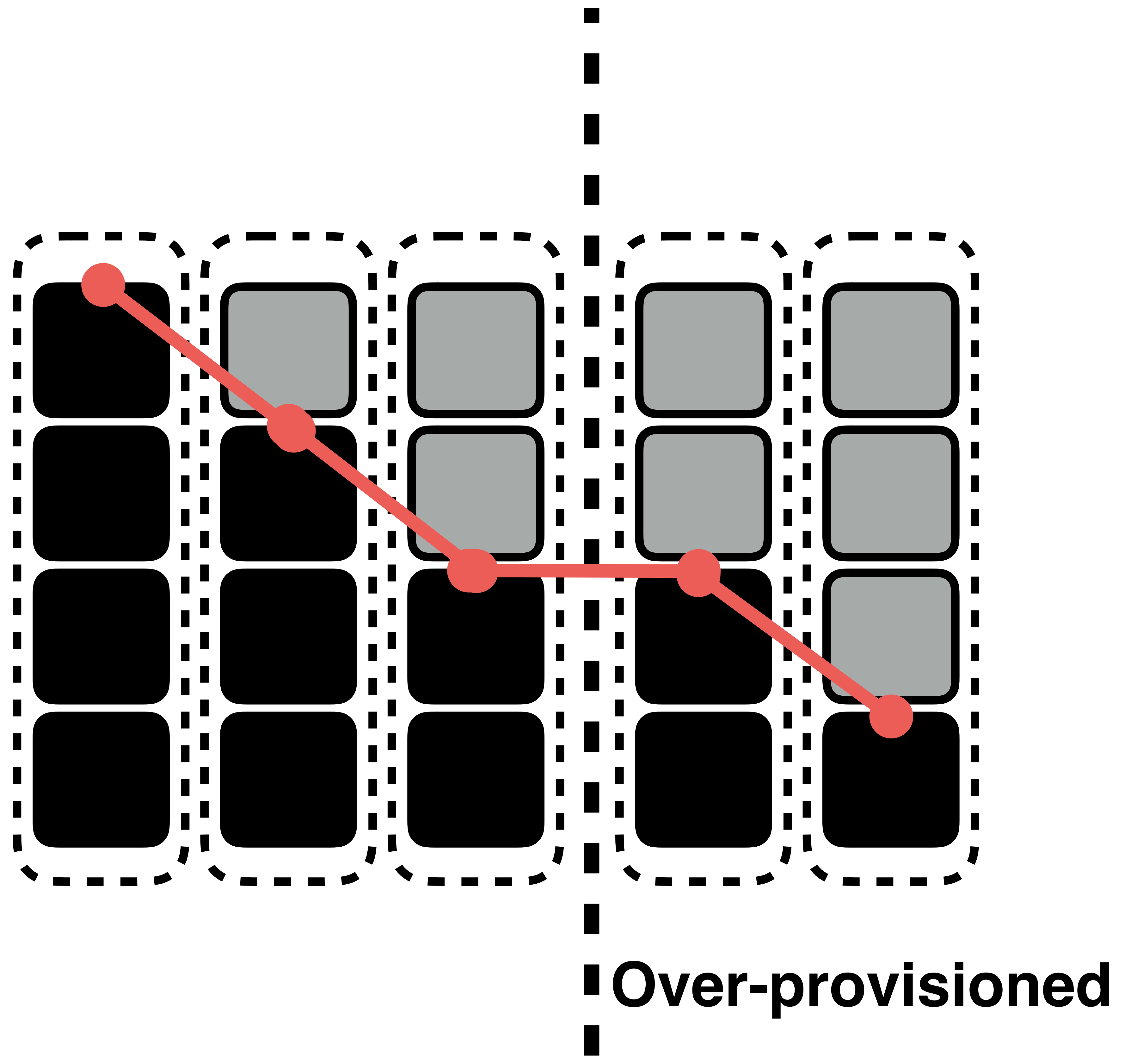
What's a bad zombie curve?



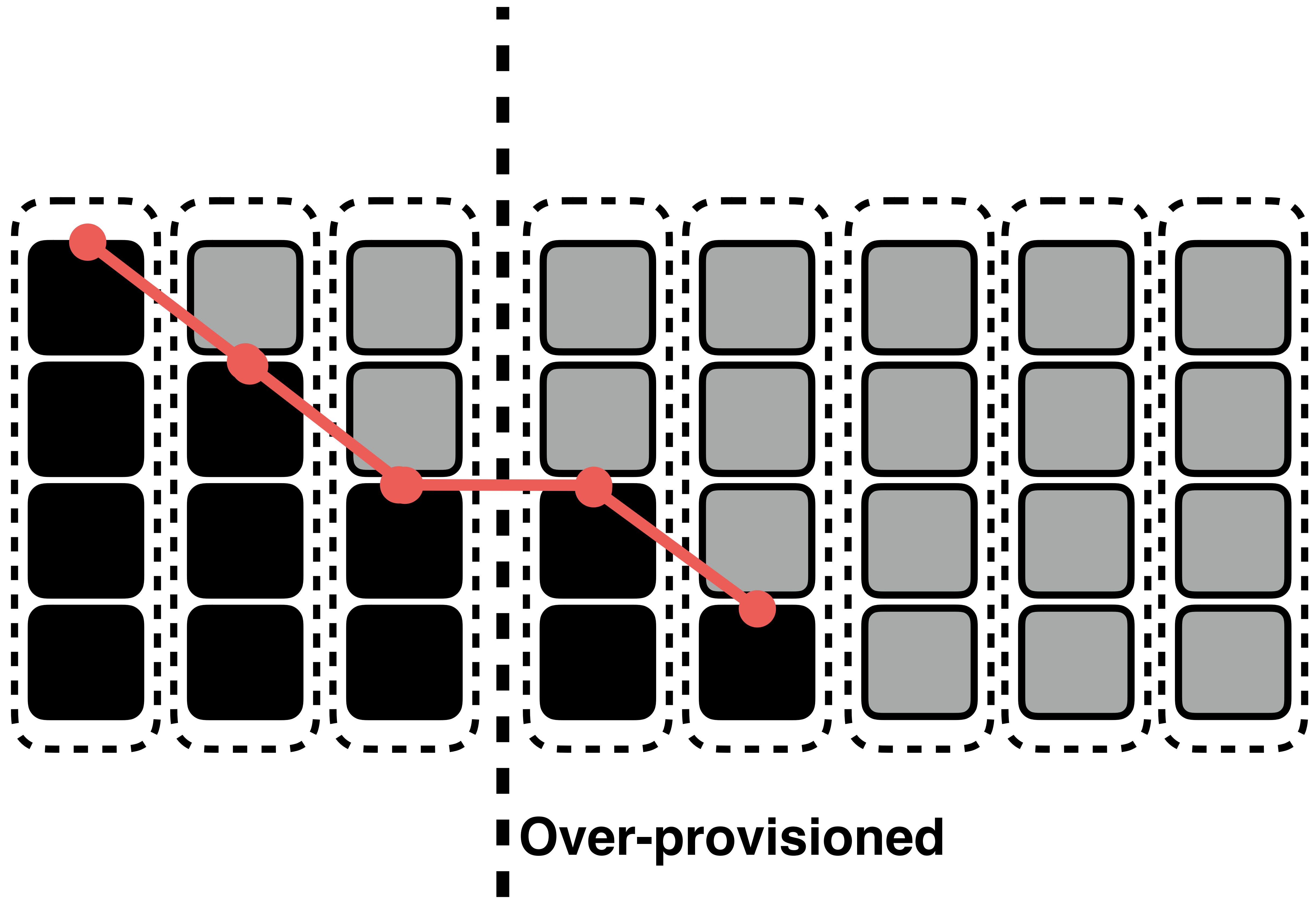
What's a bad zombie curve?



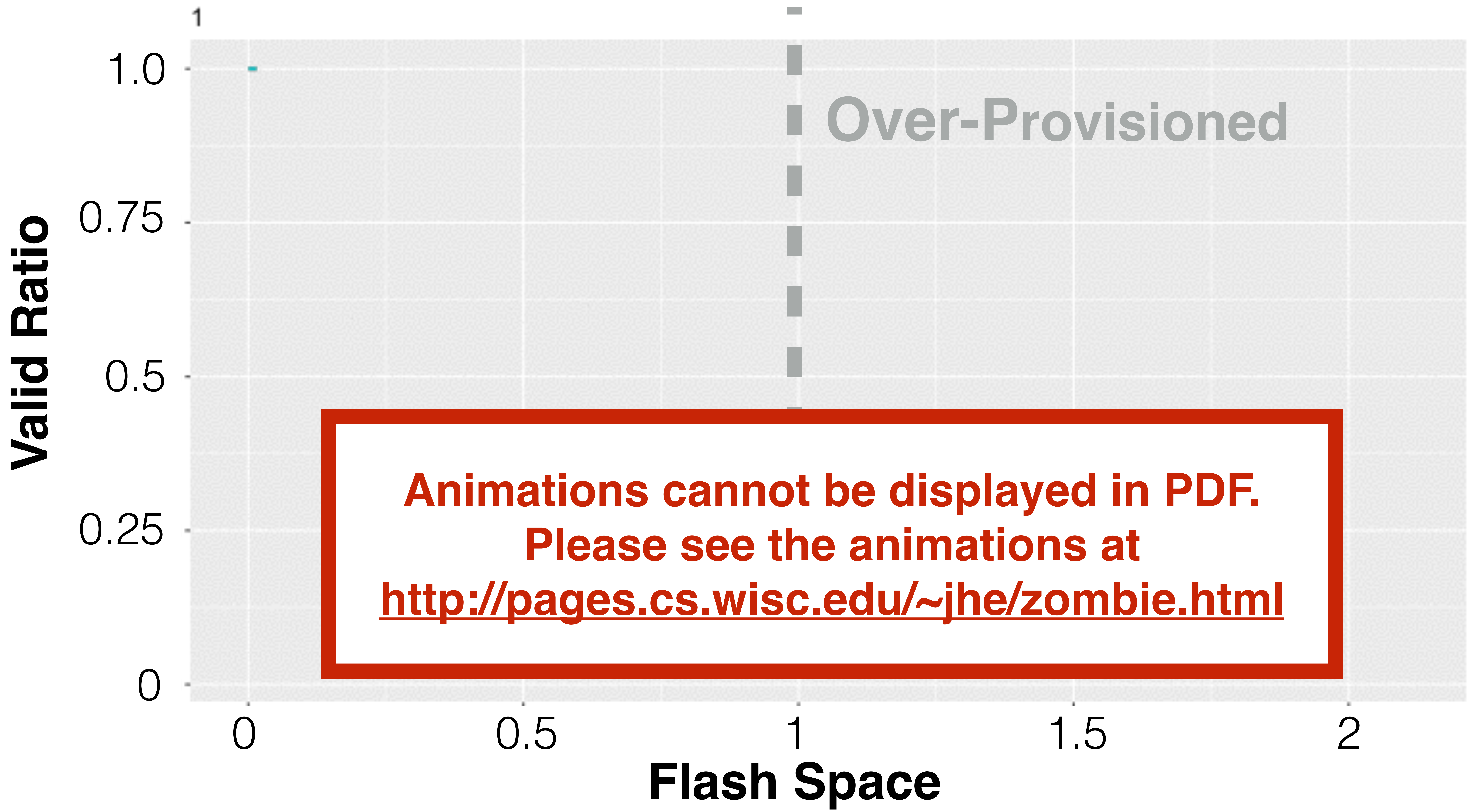
BTW, zombie curve helps you choose over-provisioning ratio



BTW, zombie curve helps you choose over-provisioning ratio

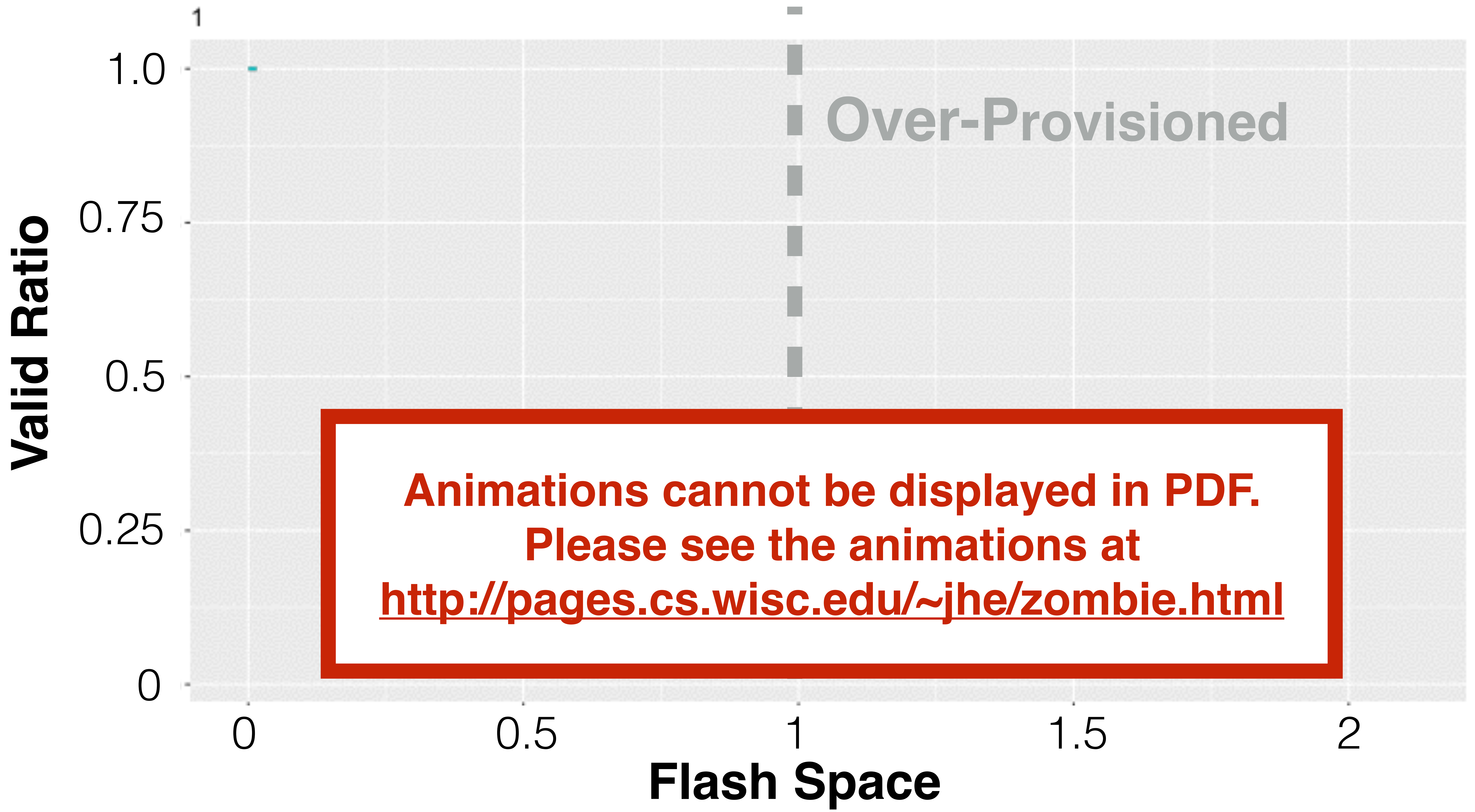


F2FS incurs a worse zombie curve (higher GC overhead) than ext4 for SQLite



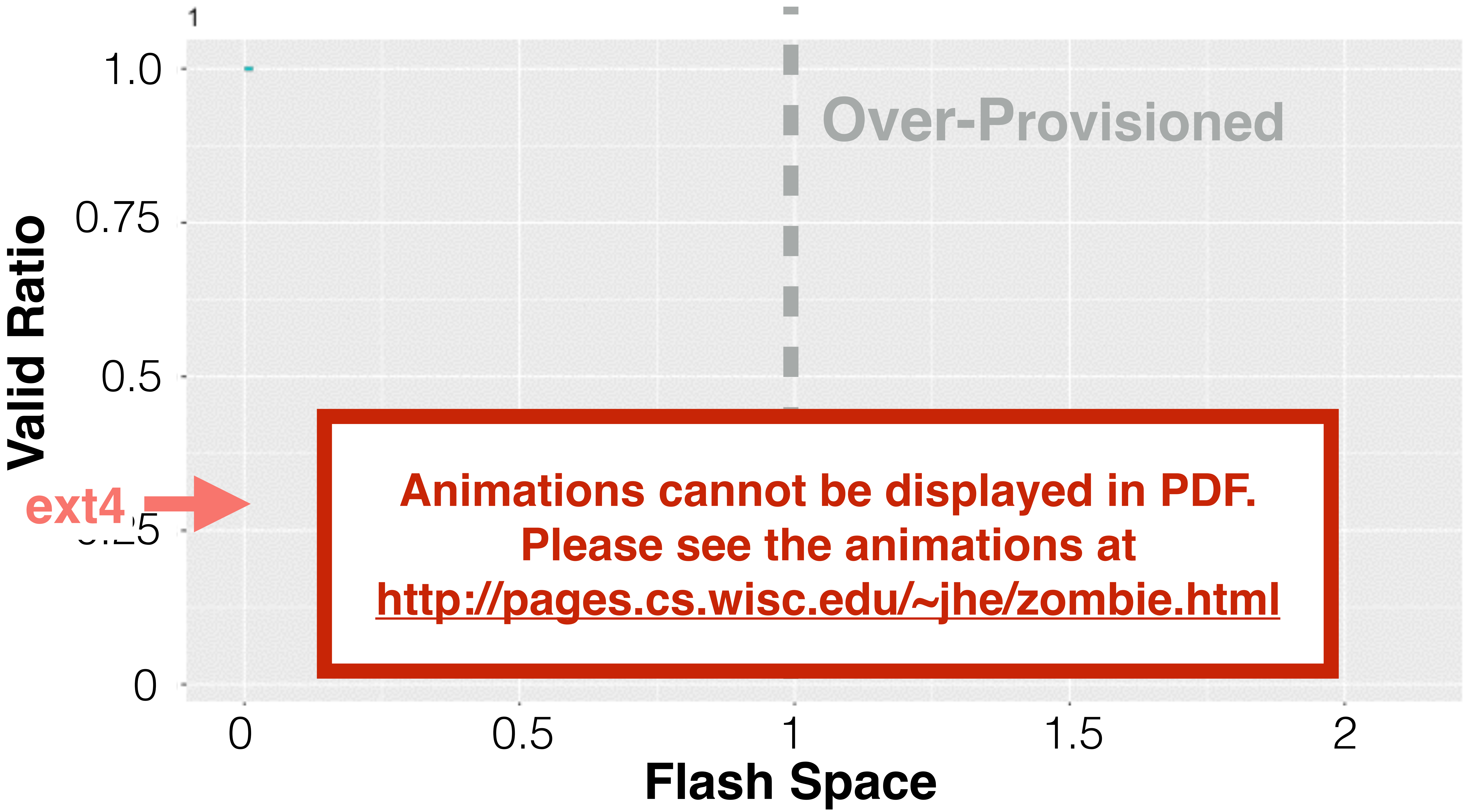
Animations cannot be displayed in PDF. Please see the animations at http://pages.cs.wisc.edu/~jhe/zombie.html

F2FS incurs a worse zombie curve (higher GC overhead) than ext4 for SQLite



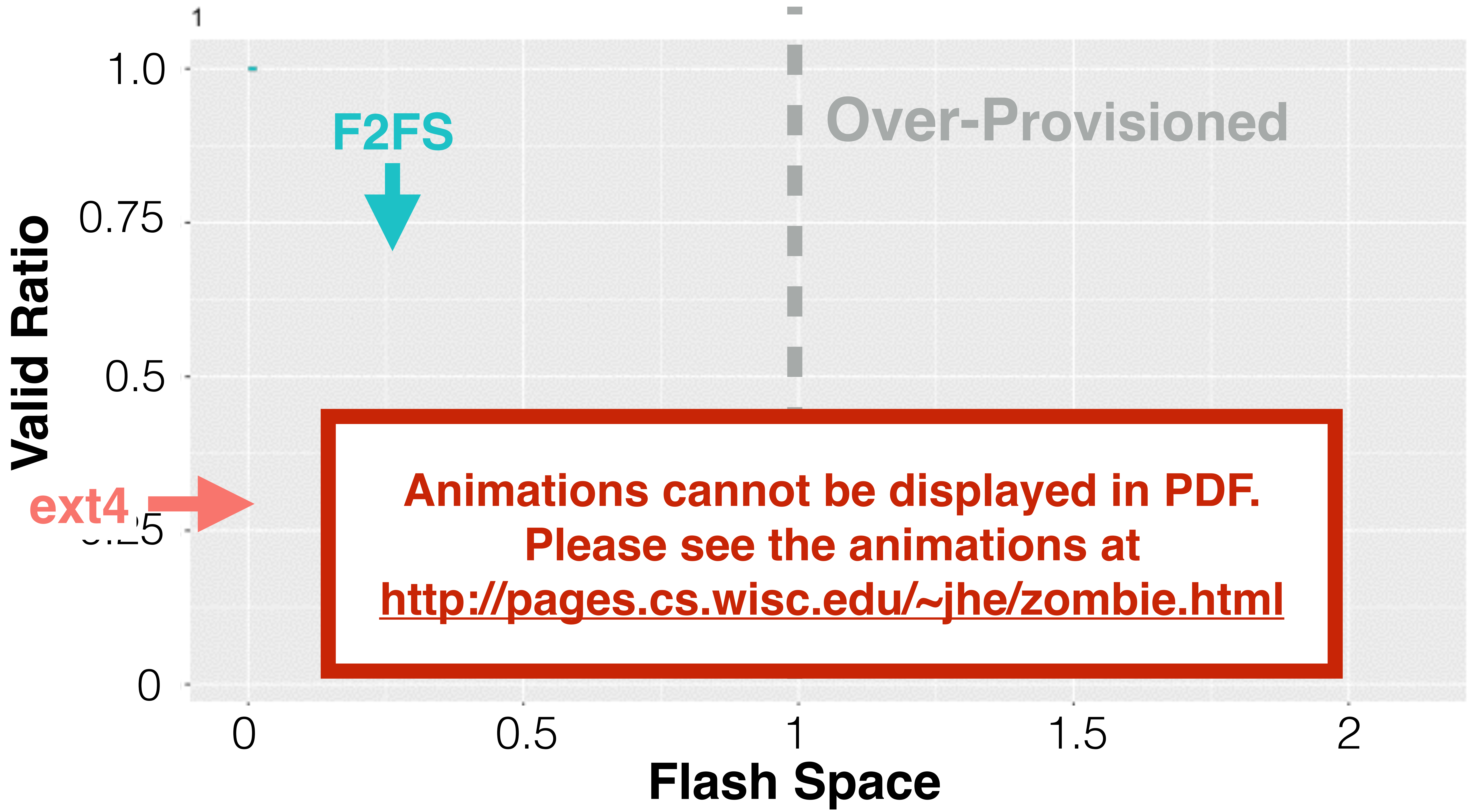
Animations cannot be displayed in PDF. Please see the animations at http://pages.cs.wisc.edu/~jhe/zombie.html

F2FS incurs a worse zombie curve (higher GC overhead) than ext4 for SQLite



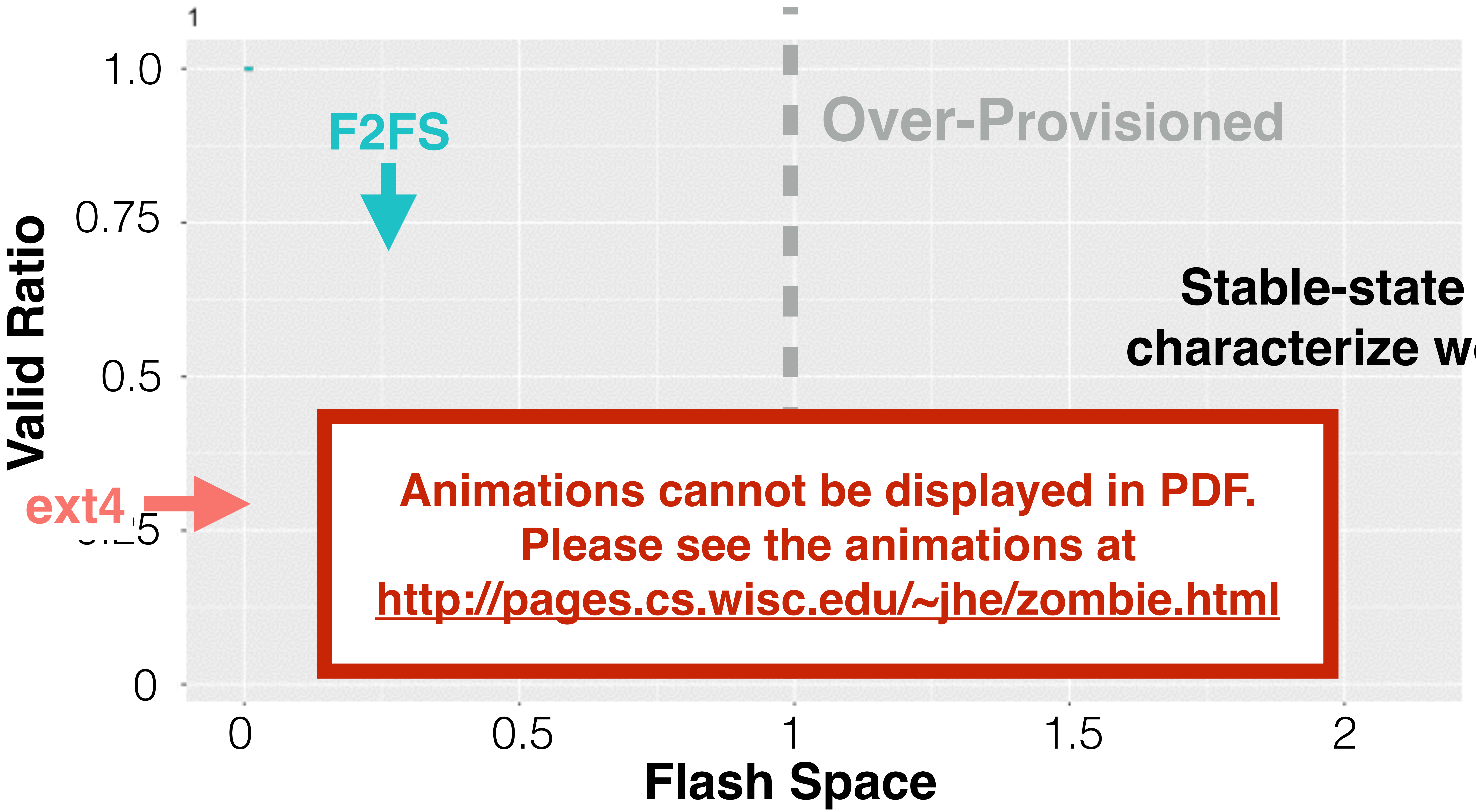
Animations cannot be displayed in PDF. Please see the animations at http://pages.cs.wisc.edu/~jhe/zombie.html

F2FS incurs a worse zombie curve (higher GC overhead) than ext4 for SQLite



Animations cannot be displayed in PDF. Please see the animations at http://pages.cs.wisc.edu/~jhe/zombie.html

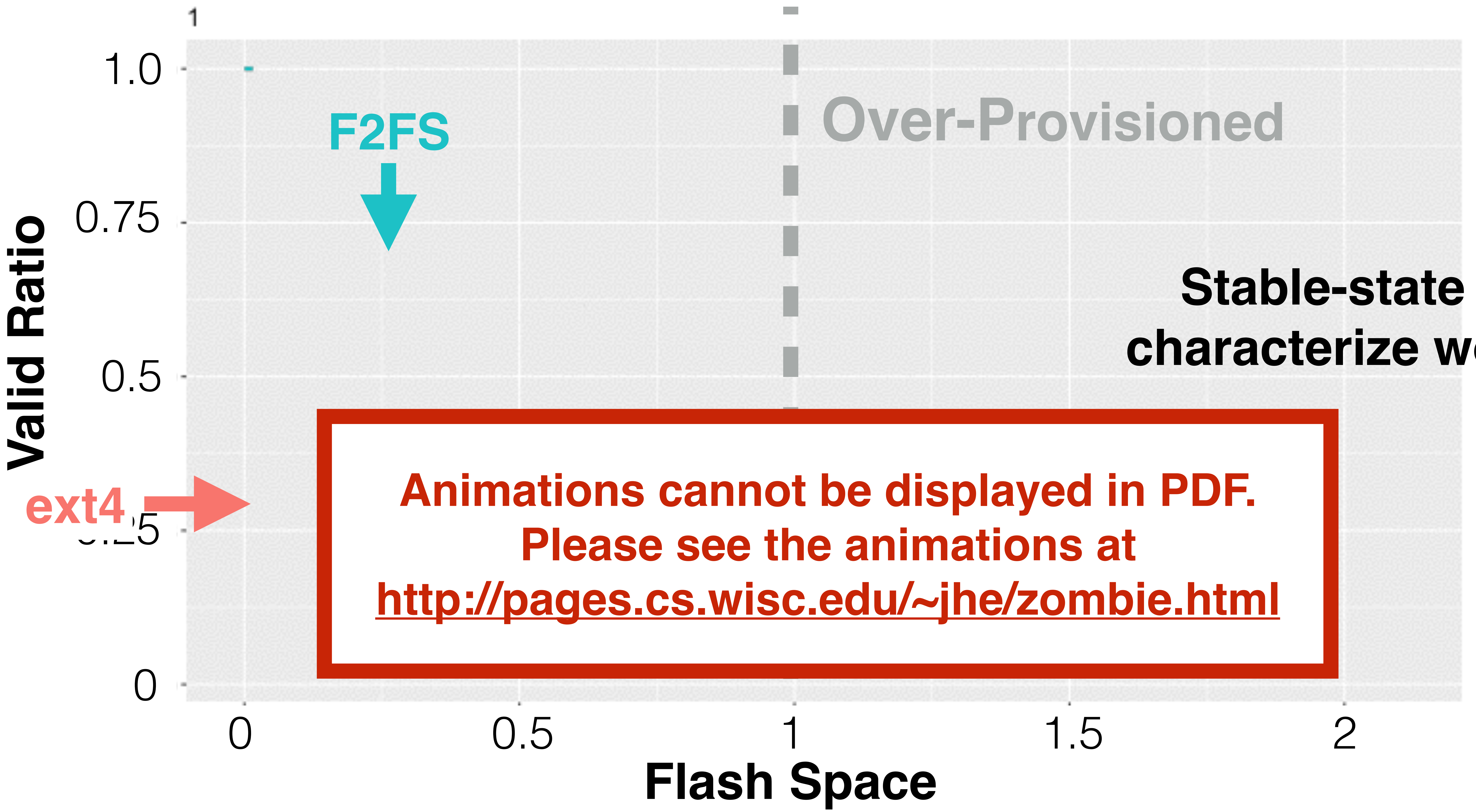
F2FS incurs a worse zombie curve (higher GC overhead) than ext4 for SQLite



Stable-state curves characterize workloads.

Animations cannot be displayed in PDF.
Please see the animations at
http://pages.cs.wisc.edu/~jhe/zombie.html

F2FS incurs a worse zombie curve (higher GC overhead) than ext4 for SQLite



Stable-state curves characterize workloads.

Animations cannot be displayed in PDF. Please see the animations at http://pages.cs.wisc.edu/~jhe/zombie.html

Why did F2FS incur a worse zombie curve (GC overhead)?

Why did F2FS incur a worse zombie curve (GC overhead)?

- SQLite fragmented F2FS

Why did F2FS incur a worse zombie curve (GC overhead)?

- SQLite fragmented F2FS
- F2FS did not discard data that was deleted by SQLite

Why did F2FS incur a worse zombie curve (GC overhead)?

- SQLite fragmented F2FS
- F2FS did not discard data that was deleted by SQLite
- F2FS was not able to stay log-structured for SQLite’s I/O pattern

More Observations

More Observations

Legacy file system allocation policies break locality

More Observations

Legacy file system allocation policies break locality

Application log structuring does not reduce GC

More Observations

Legacy file system allocation policies break locality

Application log structuring does not reduce GC

24 observations in the paper

Lessons Learned

Lessons Learned

The SSD contract is multi-dimensional

- Optimizing for one dimension is not enough
- We need more sophisticated tools to analyze workloads

Lessons Learned

The SSD contract is multi-dimensional

- Optimizing for one dimension is not enough
- We need more sophisticated tools to analyze workloads

Although not perfect, traditional file systems perform surprisingly well upon SSDs

Lessons Learned

The SSD contract is multi-dimensional

- Optimizing for one dimension is not enough
- We need more sophisticated tools to analyze workloads

Although not perfect, traditional file systems perform surprisingly well upon SSDs

Myths spread if the unwritten contract is not clarified

- “Random writes increase GC overhead”

Conclusions

Conclusions

Understanding the unwritten contract is crucial for designing high performance application and file systems

Conclusions

Understanding the unwritten contract is crucial for designing high performance application and file systems

System designing demands more vertical analysis

Conclusions

Understanding the unwritten contract is crucial for designing high performance application and file systems

System designing demands more vertical analysis

WiscSee (analyzer) and WiscSim (SSD simulator) are available at:
http://research.cs.wisc.edu/adsl/Software/wiscsee