The Unwritten Contract of Solid State Drives

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

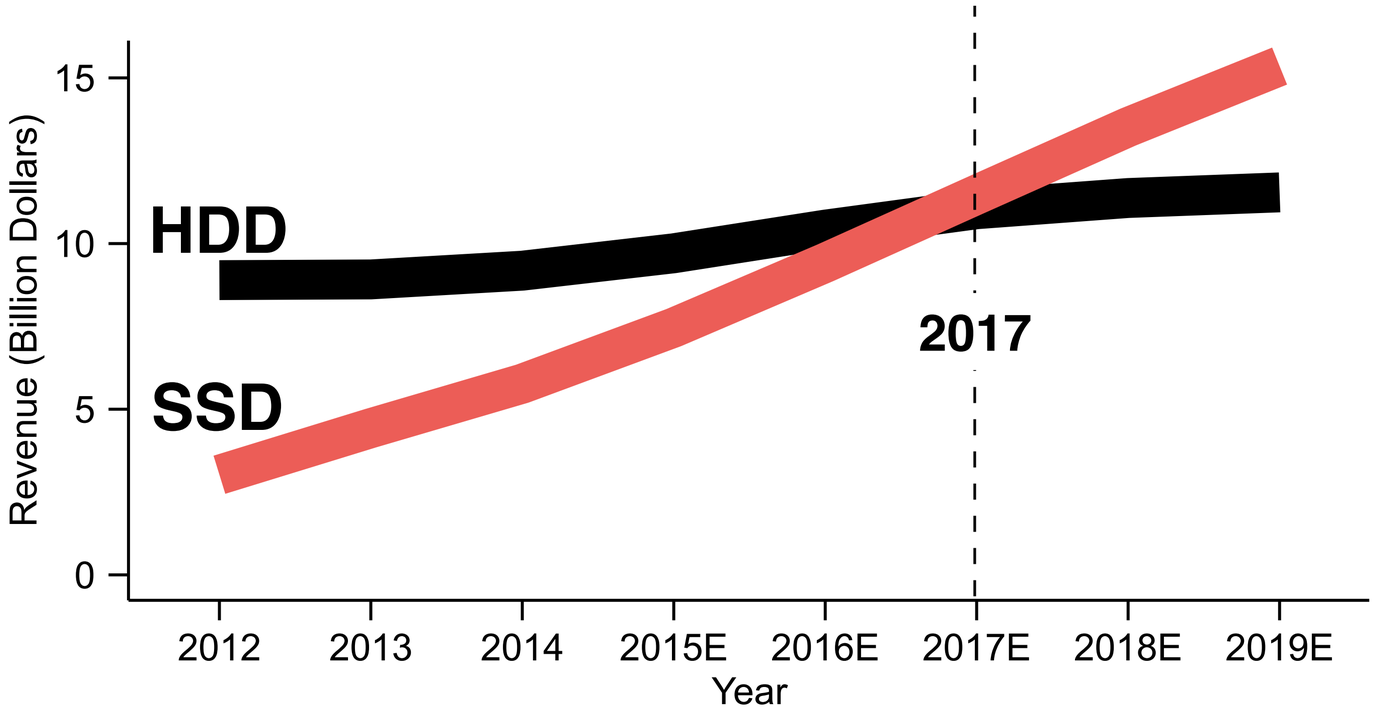
Department of Computer Sciences, University of Wisconsin - Madison

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

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

App

FS

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

App

FS

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

App

FS

App

FS

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

App

FS

App ?

FS

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

App

FS

App ?

FS ?

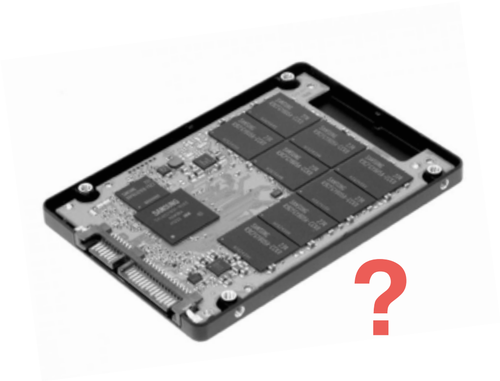
Storage stack is shifting from the HDD era to the SSD era

App

FS

App ?

FS ?



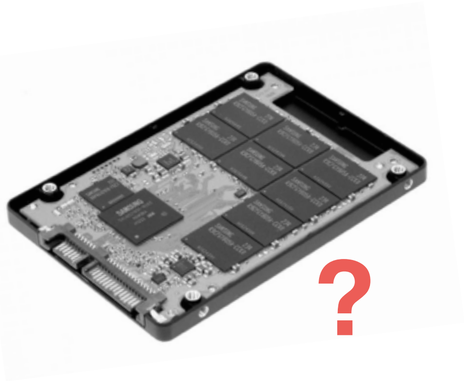
Storage stack is shifting from the HDD era to the SSD era

App

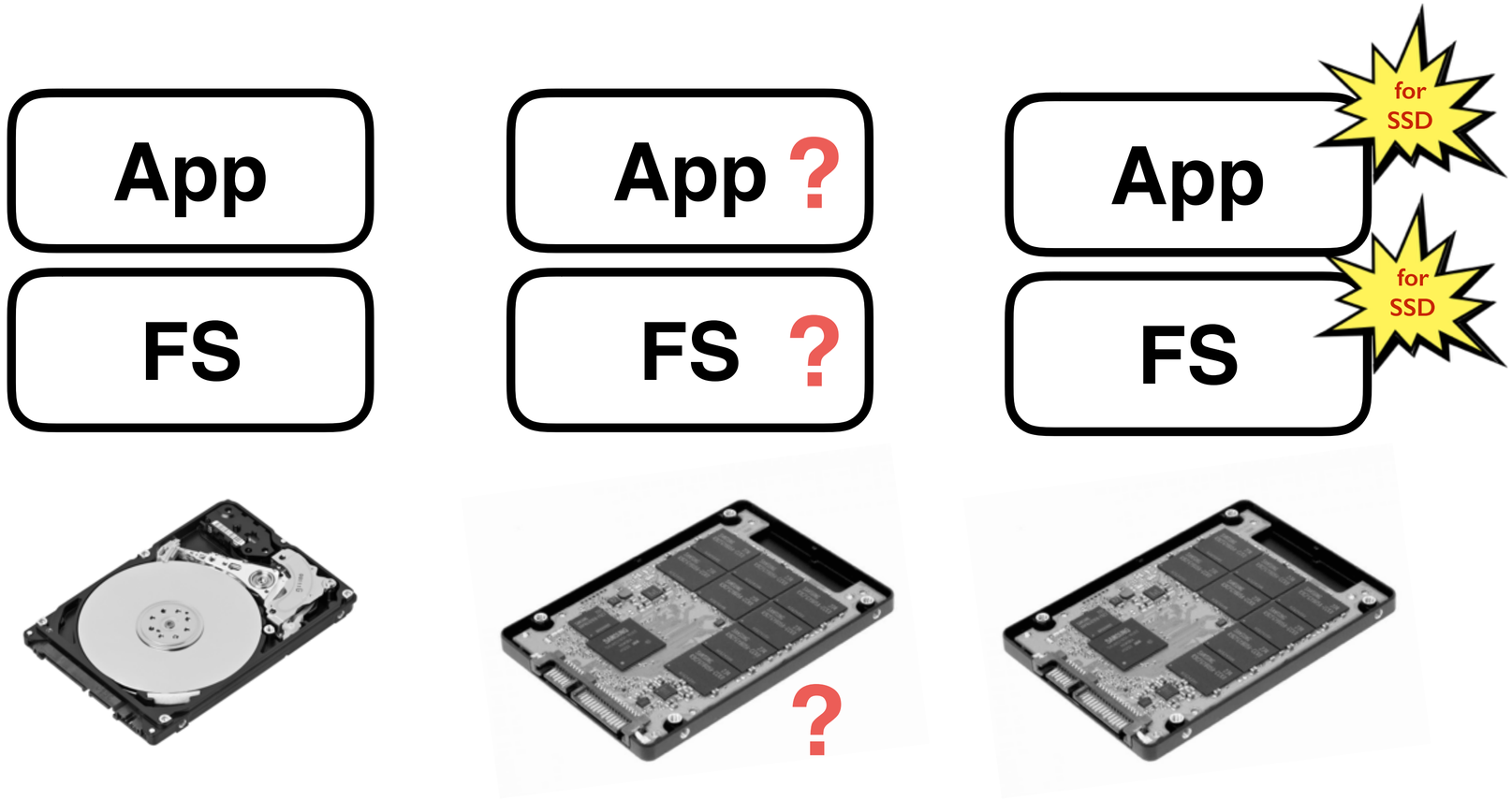
FS

App ?

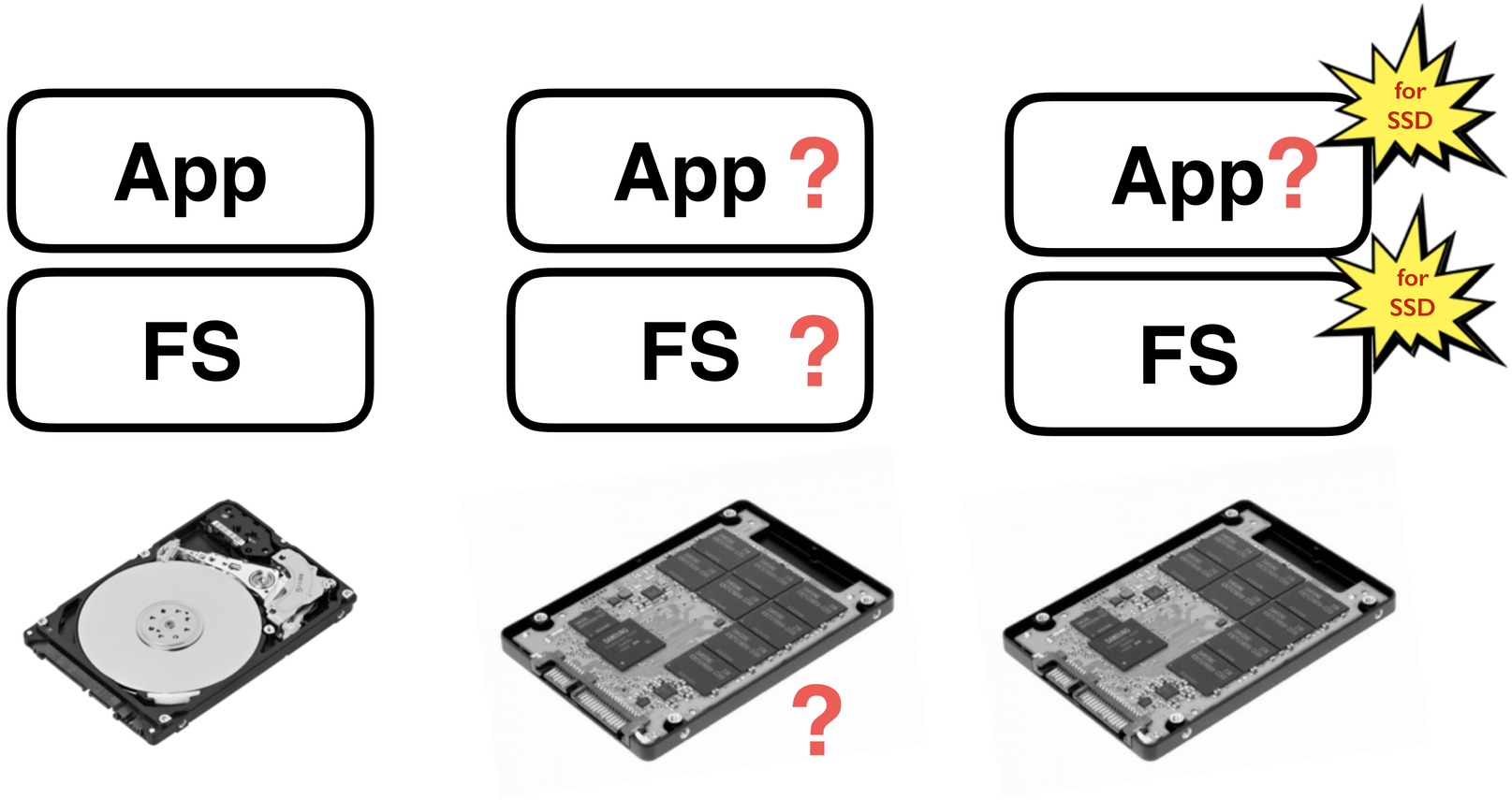
FS ?



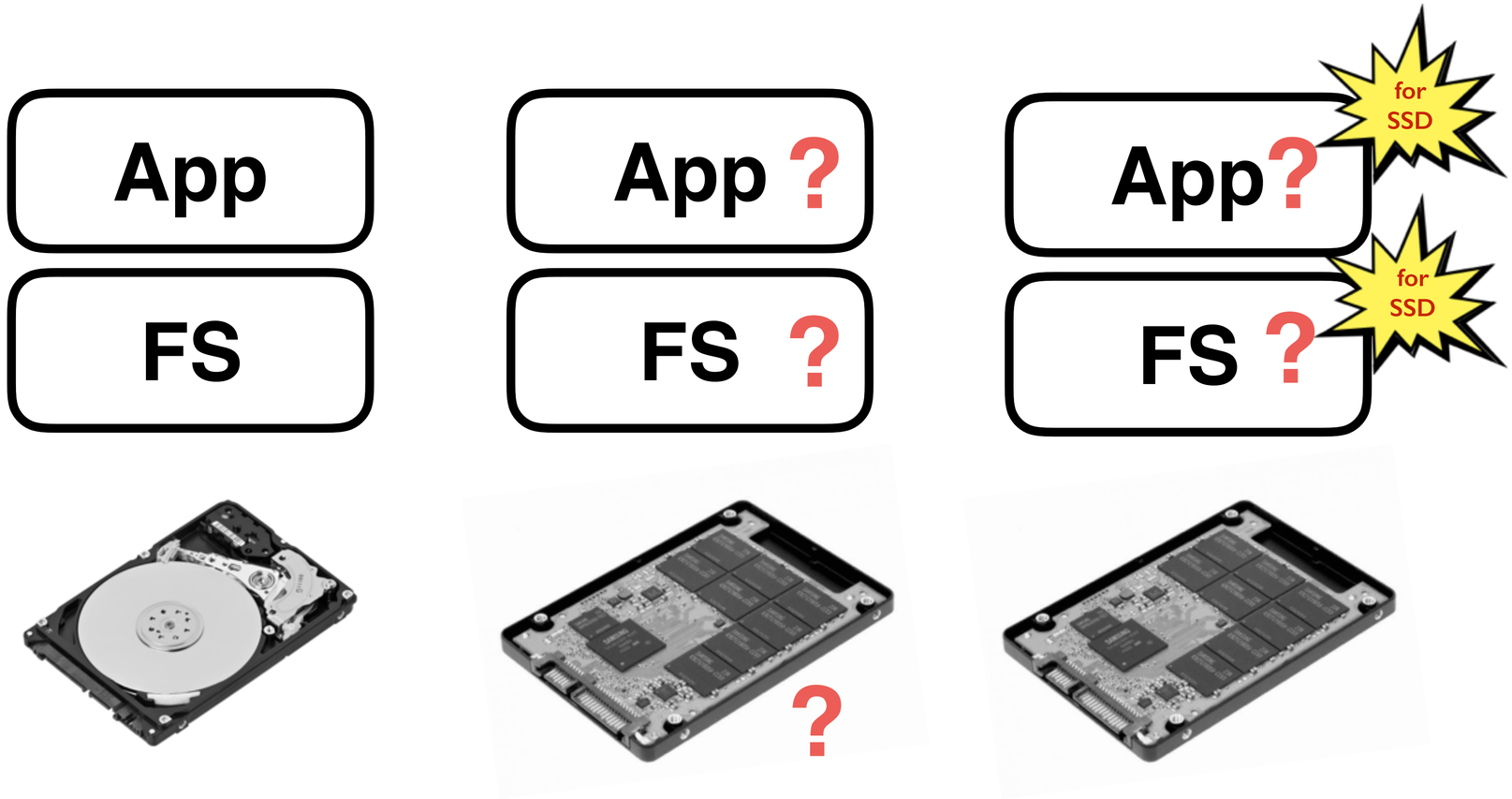
Storage stack is shifting from the HDD era to the SSD era



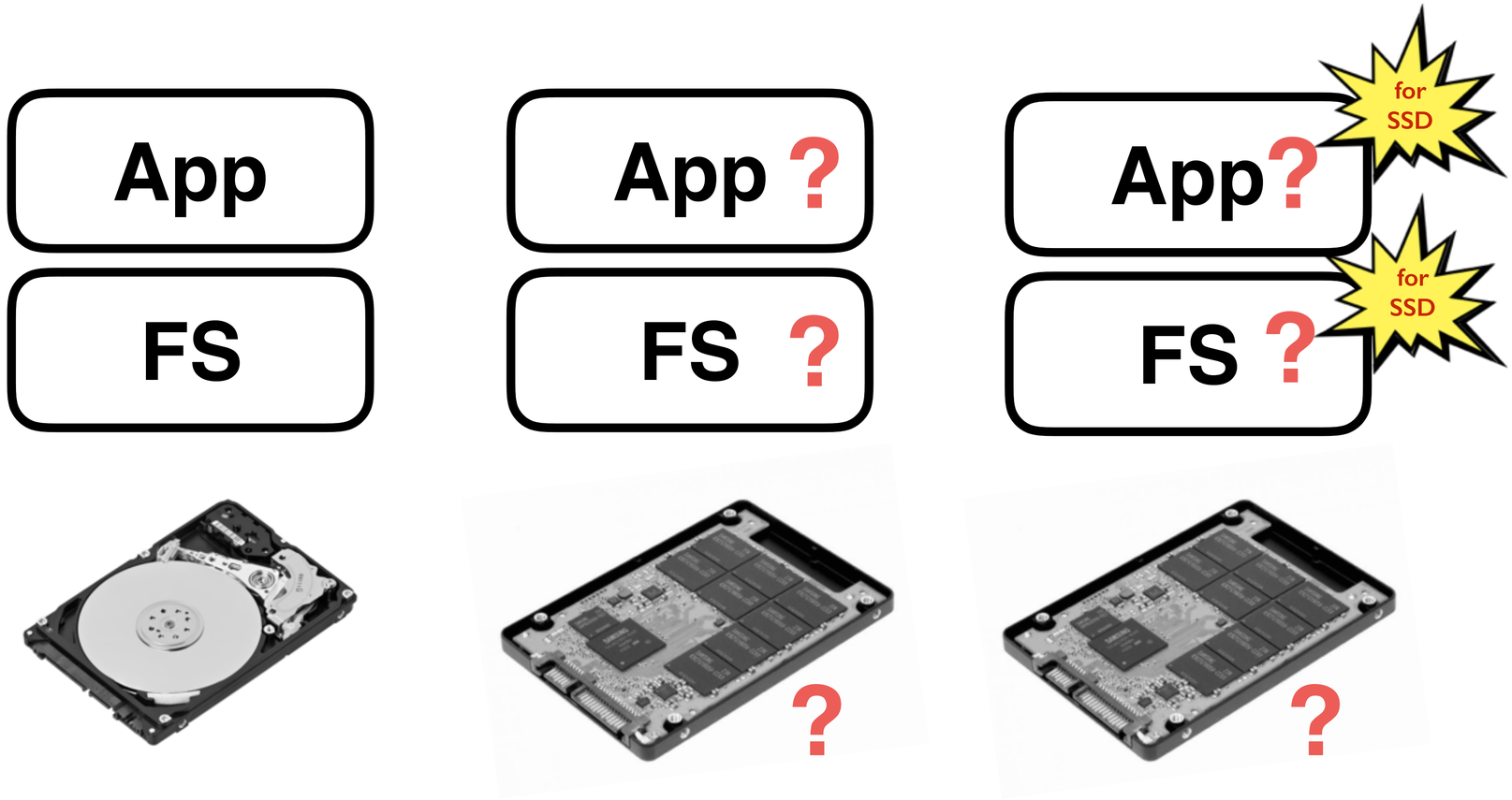
Storage stack is shifting from the HDD era to the SSD era



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



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

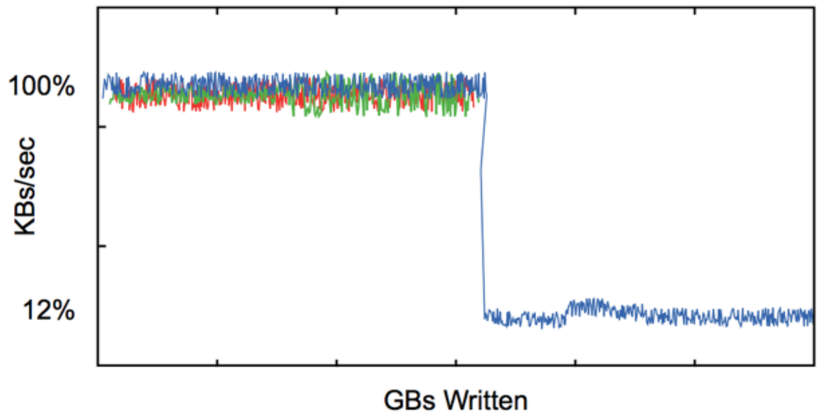


The consequences of misuses

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 misuses

Performance degradation

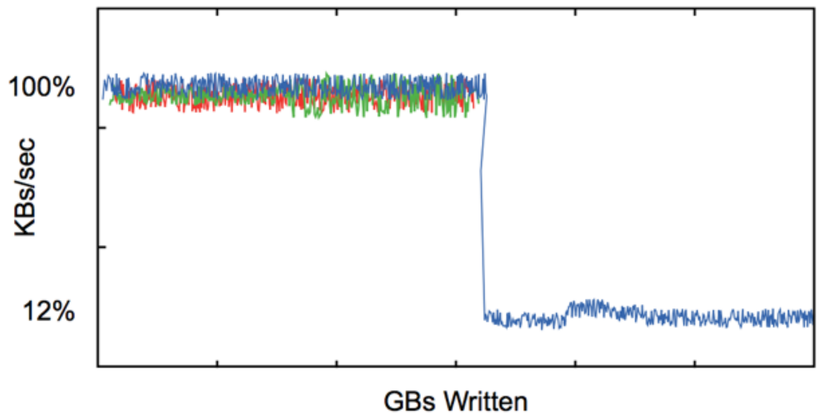


http://crestingwave.com/sites/default/files/collateral/velobit_whitepaper_ssdperformancetips.pdf

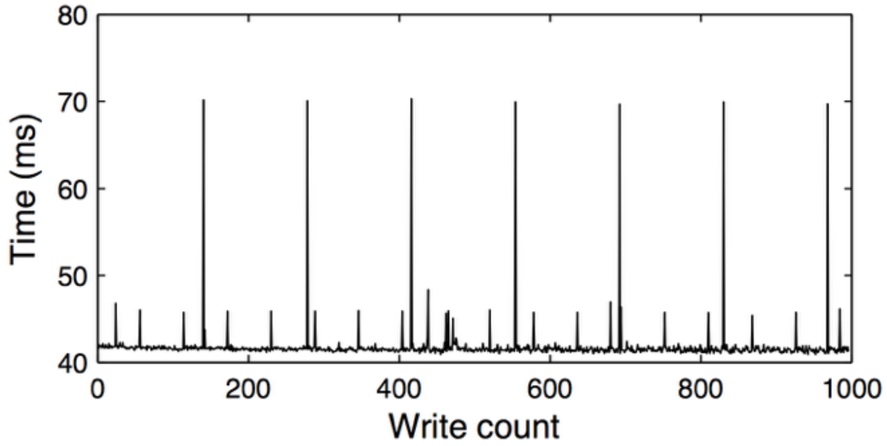
S. Boboila and P. Desnoyers. Write Endurance in Flash Drives: Measurements and Analysis. In Proceedings of the 8th USENIX Symposium on File and Storage Technologies (FAST '10), San Jose, California, February 2010

The consequences of misuses

Performance degradation



Performance fluctuation

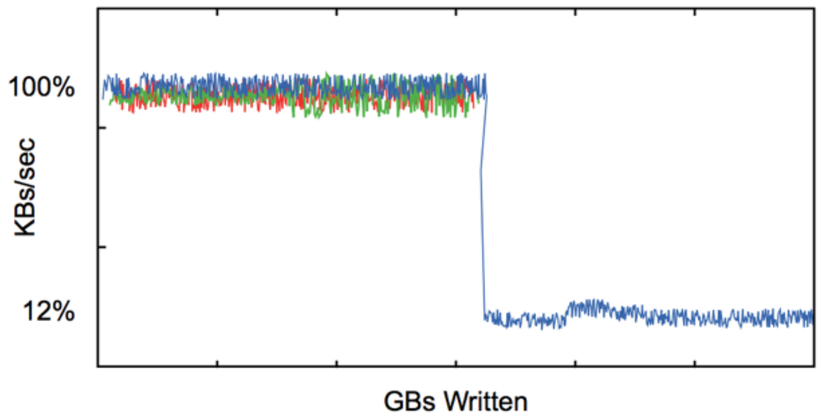


http://crestingwave.com/sites/default/files/collateral/velobit_whitepaper_ssdperformancetips.pdf

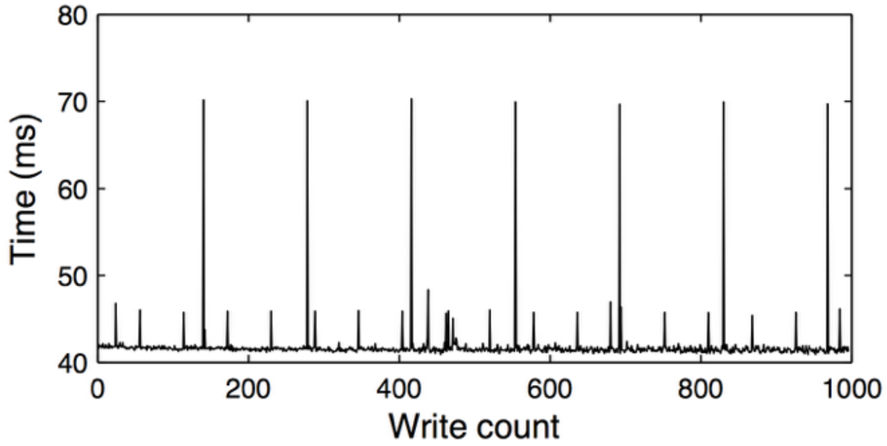
S. Boboila and P. Desnoyers. Write Endurance in Flash Drives: Measurements and Analysis. In Proceedings of the 8th USENIX Symposium on File and Storage Technologies (FAST '10), San Jose, California, February 2010

The consequences of misuses

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?

The **written** contract

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

The **written** contract

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

The **written** contract

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

The **written** contract

Example: SATA read command format

Register	7	6	5	4	3	2	1	0
Features(7:0)	Sector Count 7:0							
Features(15:8)	Sector Count 15:8							
Count(7:0)	TAG					Reserved		
Count(15:8)	PRIO(1:0)		Reserved					
LBA(7:0)	LBA 7:0							
LBA(31:24)	LBA 31:24							
LBA(15:8)	LBA 15:8							
LBA(39:32)	LBA 39:32							
LBA(23:16	LBA 23:16							
LBA(47:40)	LBA 47:40							
ICC	ICC(7:0)							
Device	FUA	1	Res	0	Reserved			
Command	60h							

Figure 205 – READ FPDMA QUEUED command definition

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

The **written** contract

Example: SATA read command format

Register	7	6	5	4	3	2	1	0
Features(7:0)	Sector Count 7:0							
Features(15:8)	Sector Count 15:8							
Count(7:0)	TAG					Reserved		
Count(15:8)	PRIO(1:0)		Reserved					
LBA(7:0)	LBA 7:0							
LBA(31:24)	LBA 31:24							
LBA(15:8)	LBA 15:8							
LBA(39:32)	LBA 39:32							
LBA(23:16	LBA 23:16							
LBA(47:40)	LBA 47:40							
ICC	ICC(7:0)							
Device	FUA	1	Res	0	Reserved			
Command	60h							

Figure 205 – READ FPDMA QUEUED command definition

The **unwritten** contract

?

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

The **written** contract

Example: SATA read command format

Register	7	6	5	4	3	2	1	0
Features(7:0)	Sector Count 7:0							
Features(15:8)	Sector Count 15:8							
Count(7:0)	TAG					Reserved		
Count(15:8)	PRIO(1:0)		Reserved					
LBA(7:0)	LBA 7:0							
LBA(31:24)	LBA 31:24							
LBA(15:8)	LBA 15:8							
LBA(39:32)	LBA 39:32							
LBA(23:16	LBA 23:16							
LBA(47:40)	LBA 47:40							
ICC	ICC(7:0)							
Device	FUA	1	Res	0	Reserved			
Command	60h							

Figure 205 – READ FPDMA QUEUED command definition

The **unwritten** contract

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

The **written** contract

Example: SATA read command format

Register	7	6	5	4	3	2	1	0
Features(7:0)	Sector Count 7:0							
Features(15:8)	Sector Count 15:8							
Count(7:0)	TAG					Reserved		
Count(15:8)	PRIO(1:0)		Reserved					
LBA(7:0)	LBA 7:0							
LBA(31:24)	LBA 31:24							
LBA(15:8)	LBA 15:8							
LBA(39:32)	LBA 39:32							
LBA(23:16	LBA 23:16							
LBA(47:40)	LBA 47:40							
ICC	ICC(7:0)							
Device	FUA	1	Res	0	Reserved			
Command	60h							

Figure 205 – READ FPDMA QUEUED command definition

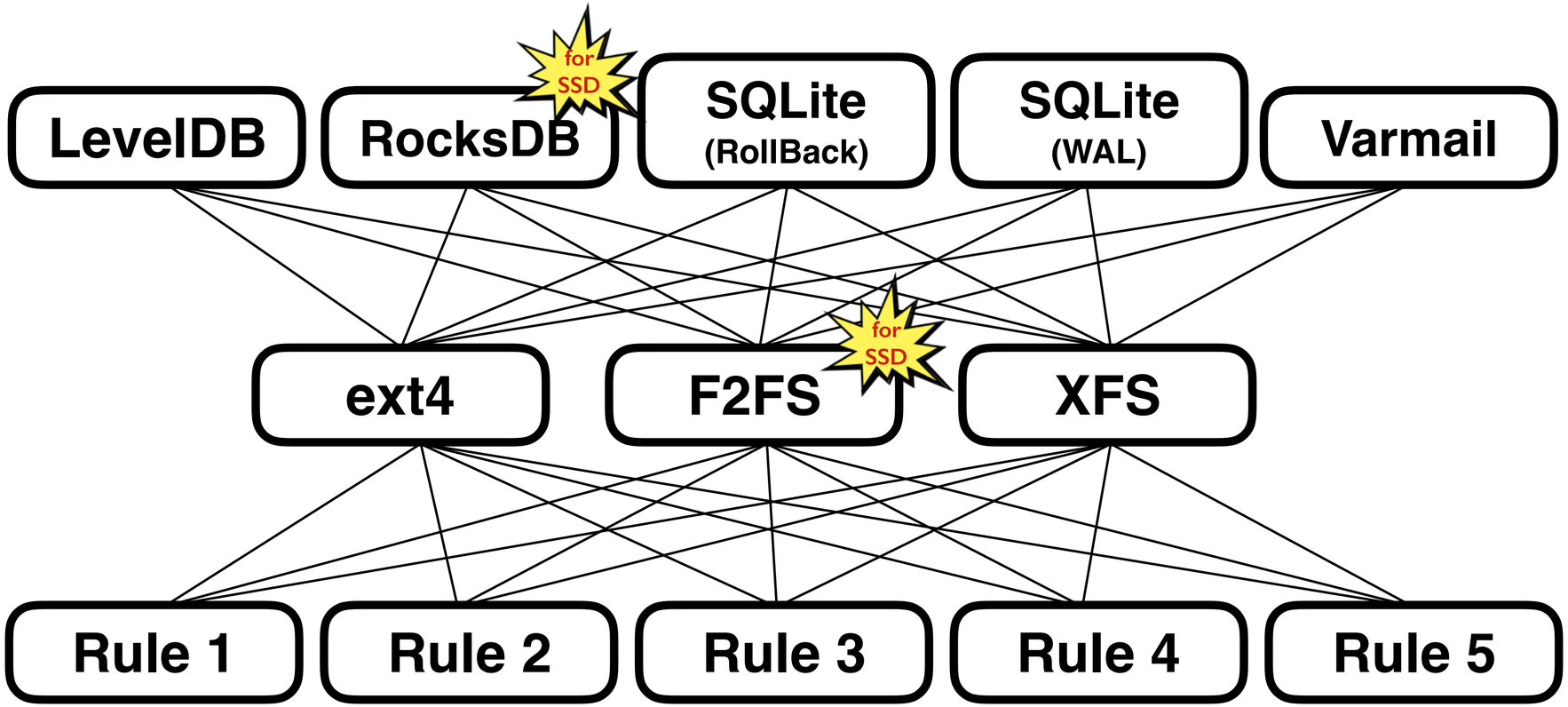
The **unwritten** contract

Rules

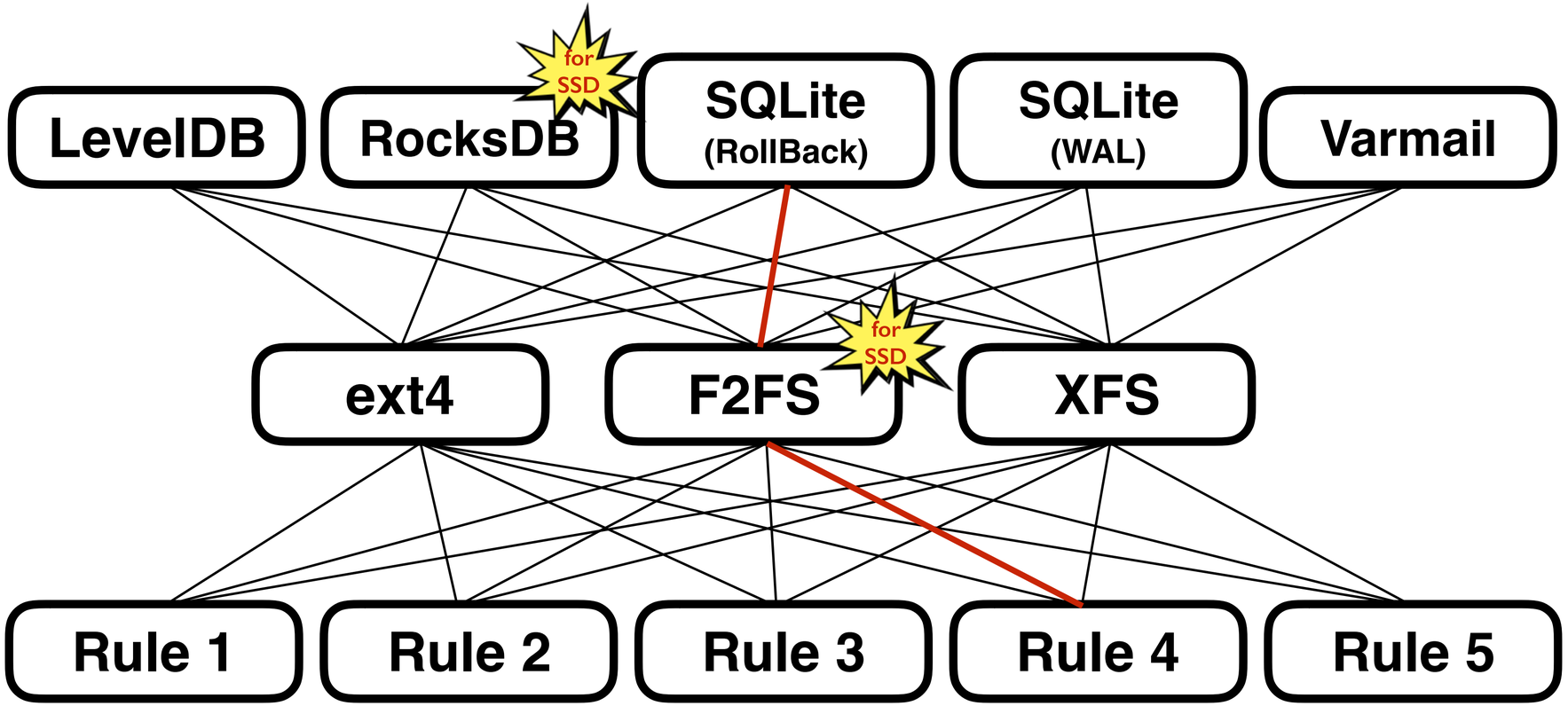
1. **Request Scale**
2. **Locality**
3. **Aligned Sequentiality**
4. **Grouping by Death Time**
5. **Uniform Data Lifetime**

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?



We made 24 observations, here are the 3 in this talk

- **Implementation of Linux limits performance**
- **F2FS incurs more GC overhead than ext4/XFS**
- **Application log structuring does not reduce GC**

Outline

- **Overview**
- SSD Unwritten Contract
- Violations of Unwritten Contract
  - Method
  - Observations
- Lessons Learned
- Conclusions

Outline

- Overview
- **SSD Unwritten Contract**
- Violations of Unwritten Contract
  - Method
  - Observations
- Lessons Learned
- Conclusions

We summarize the unwritten contract from existing studies

Contract Rules

#1 Request Scale

#2 Locality

#3 Aligned Sequentiality

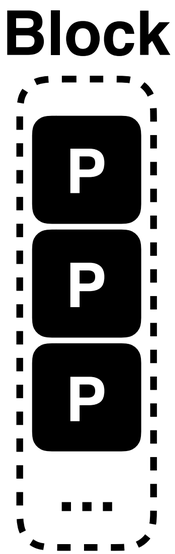
#4 Grouping by Death Time

#5 Uniform Data Lifetime

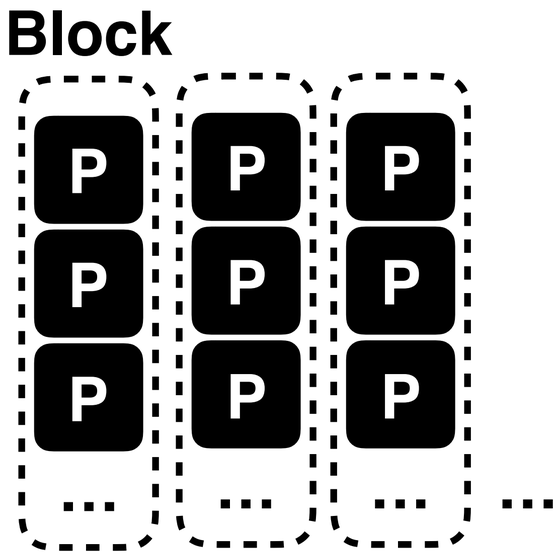
SSD Background

SSD Background

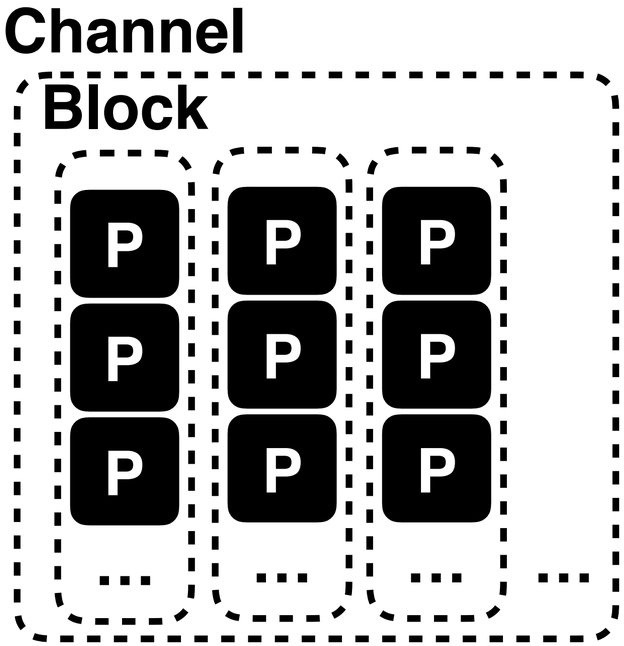
SSD Background



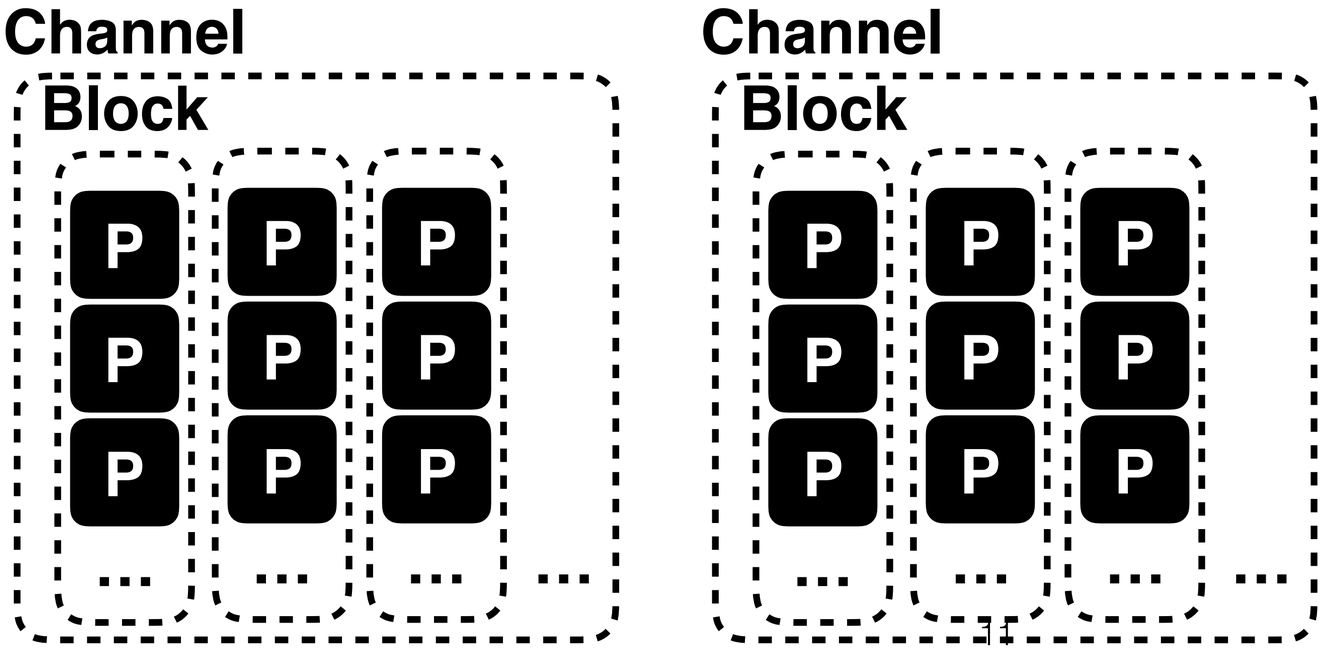
SSD Background



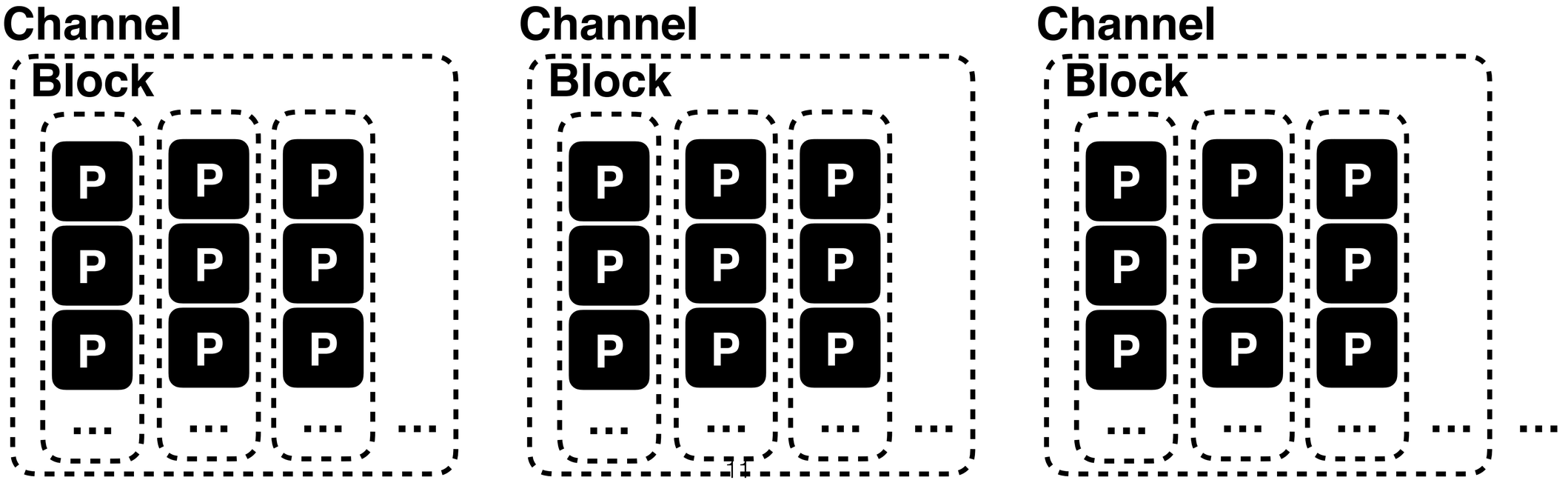
SSD Background



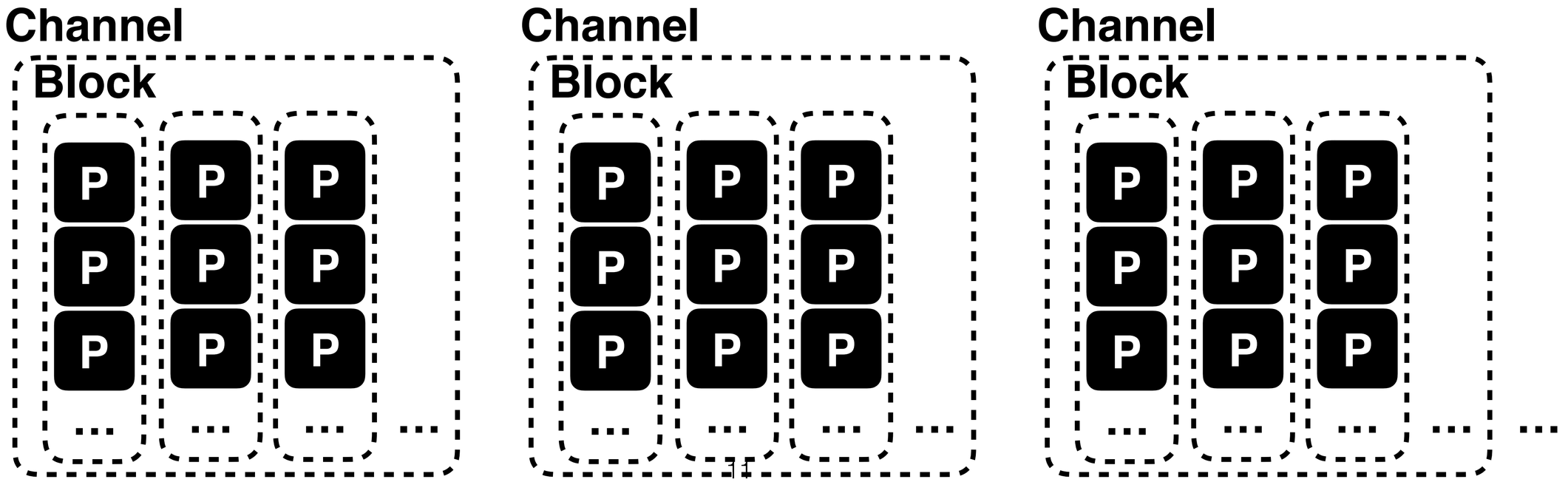
SSD Background



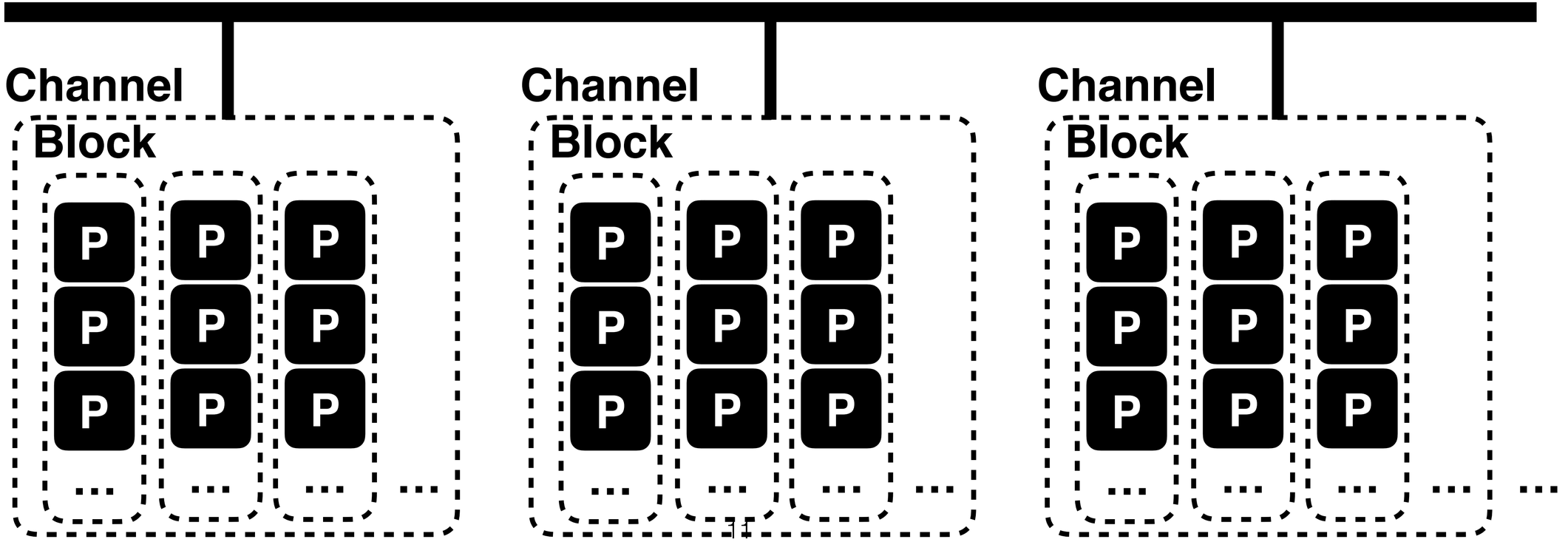
SSD Background



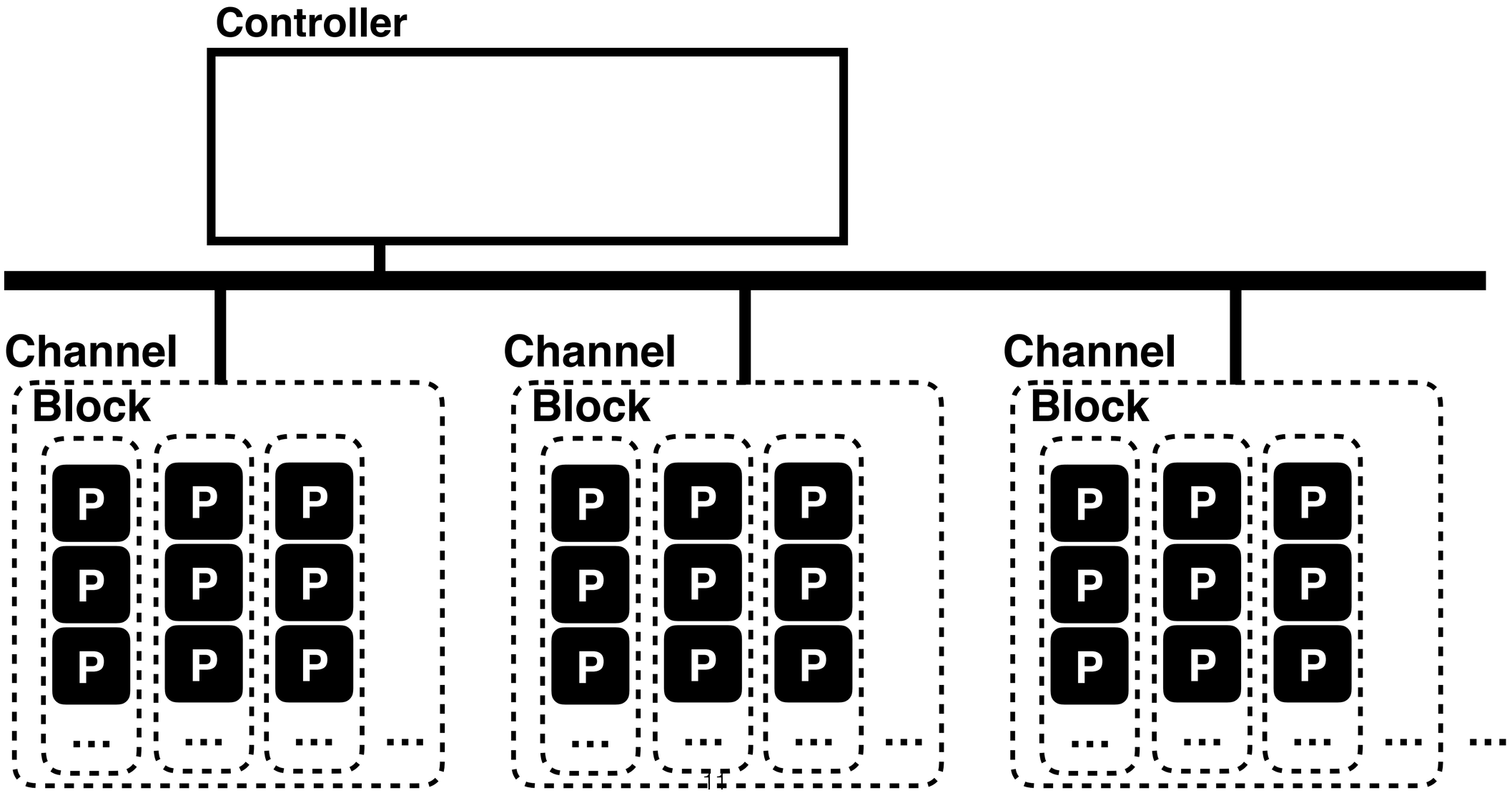
SSD Background



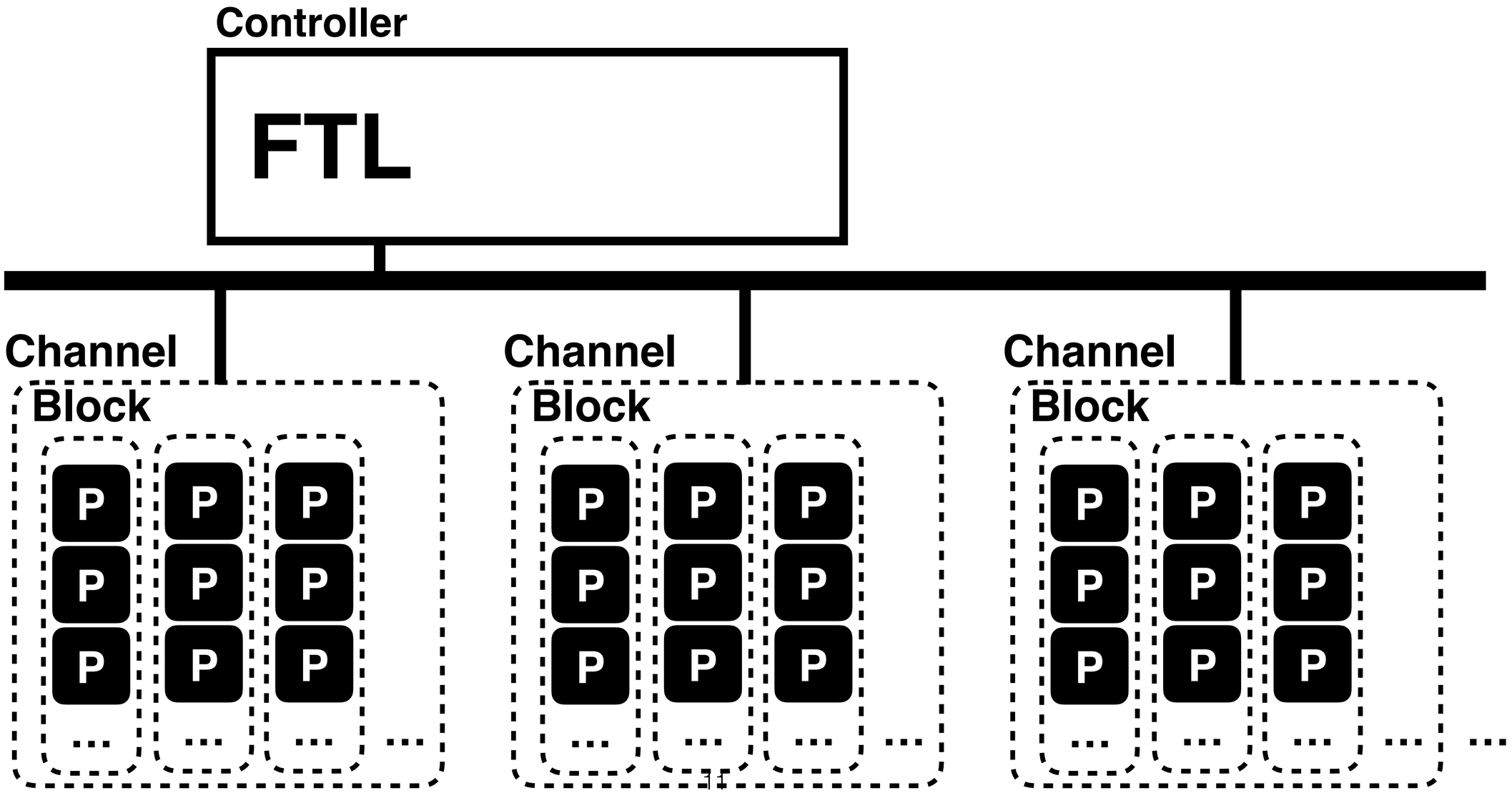
SSD Background



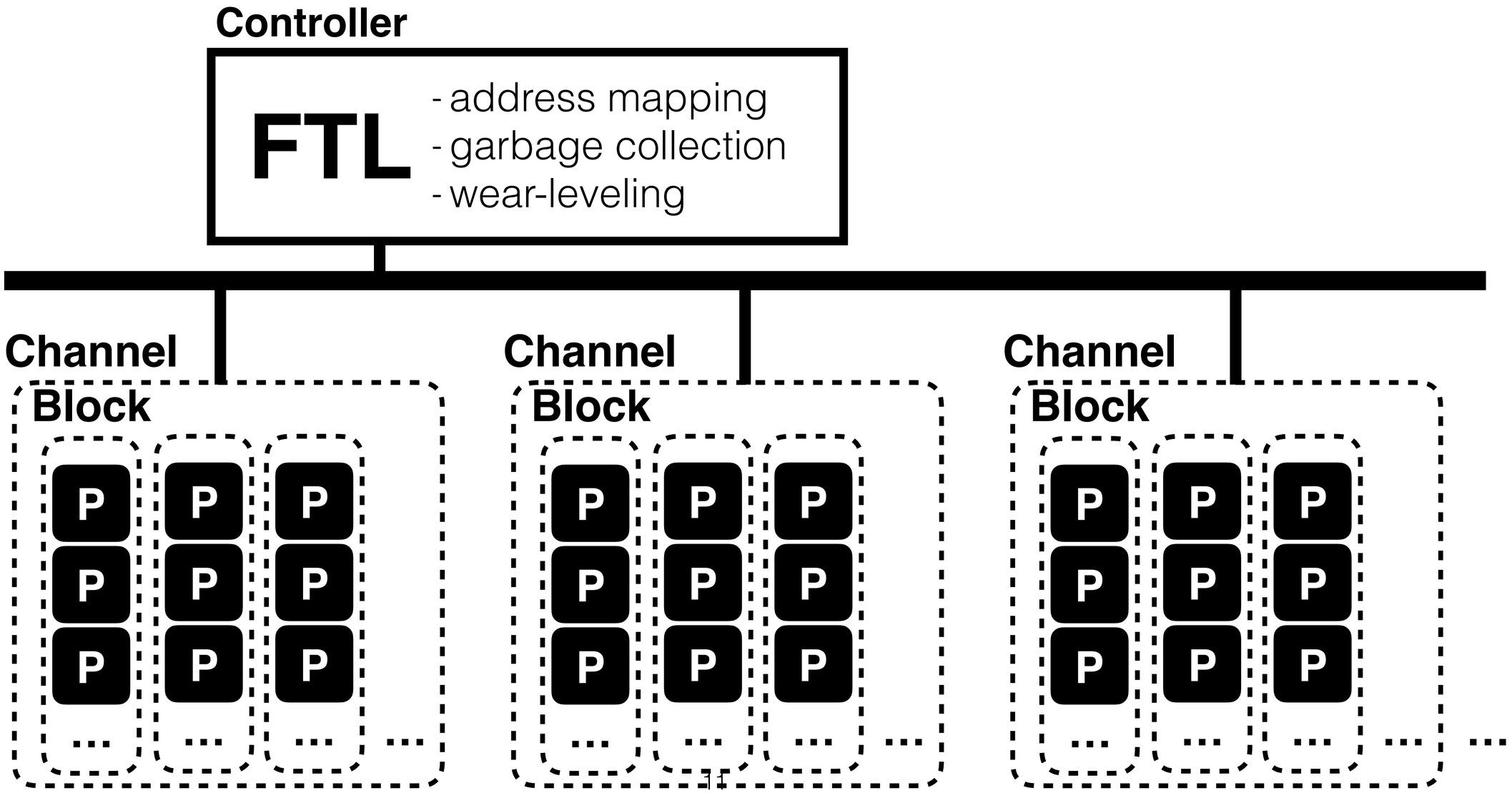
SSD Background



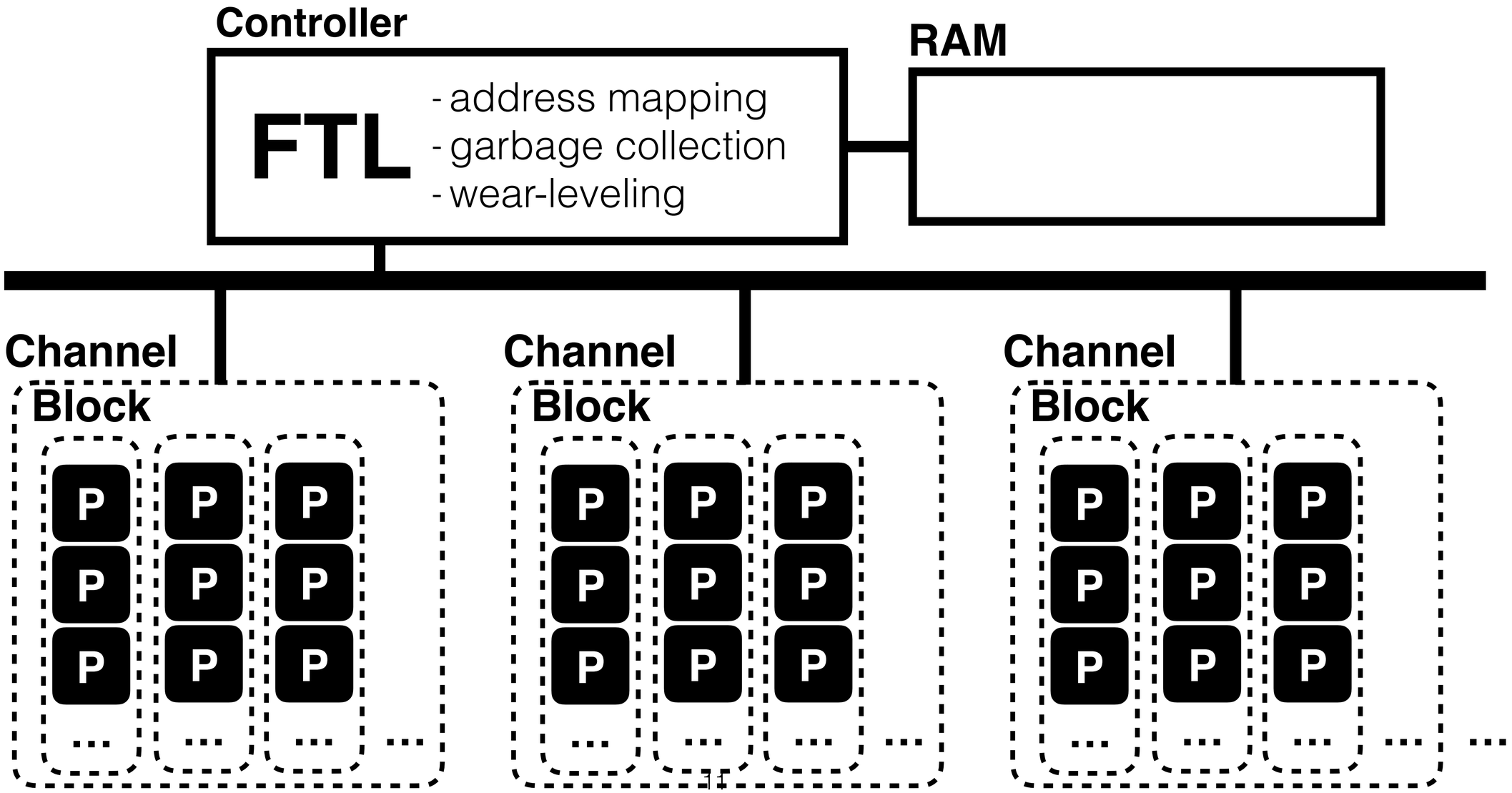
SSD Background



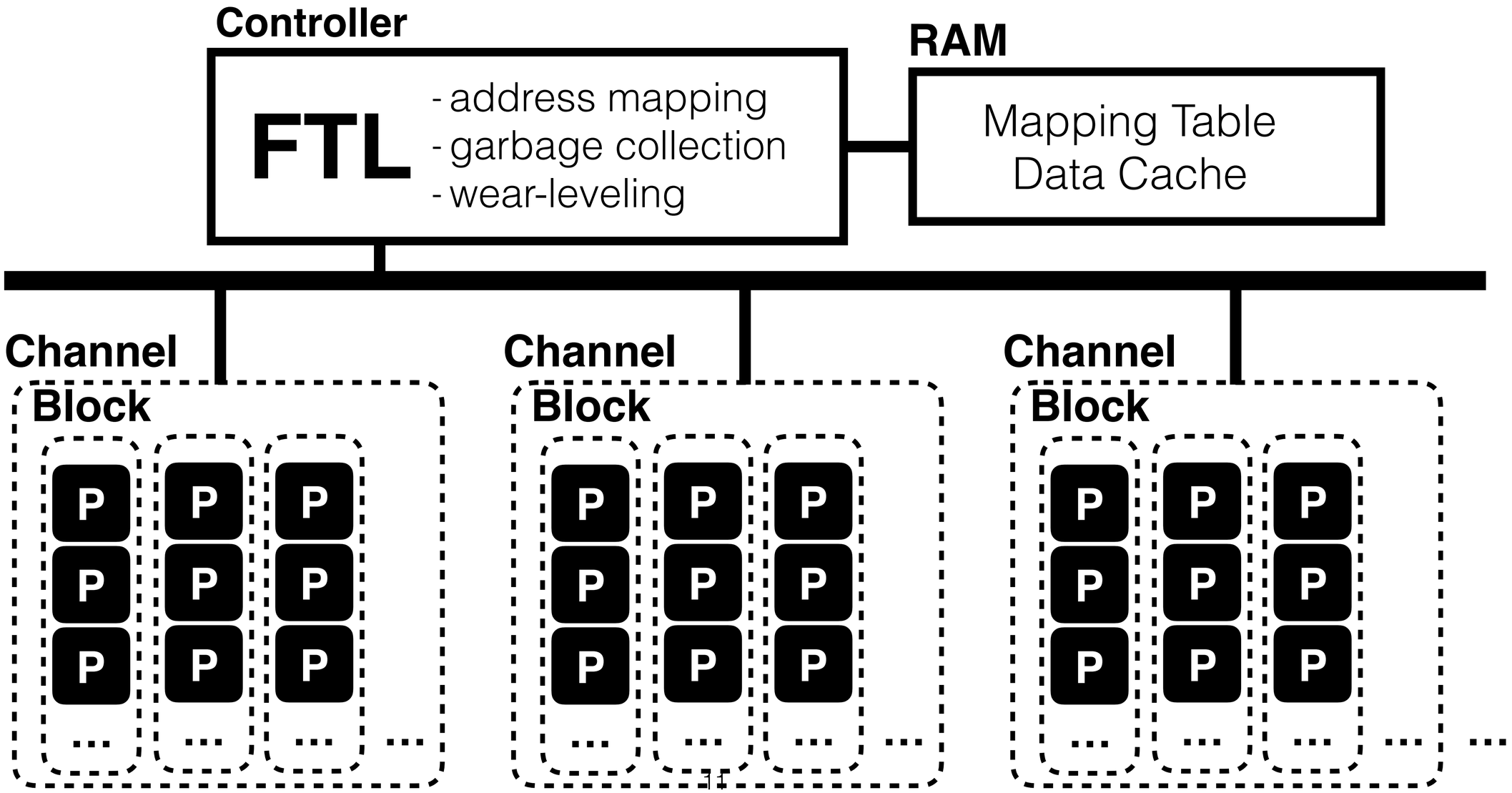
SSD Background



SSD Background



SSD Background

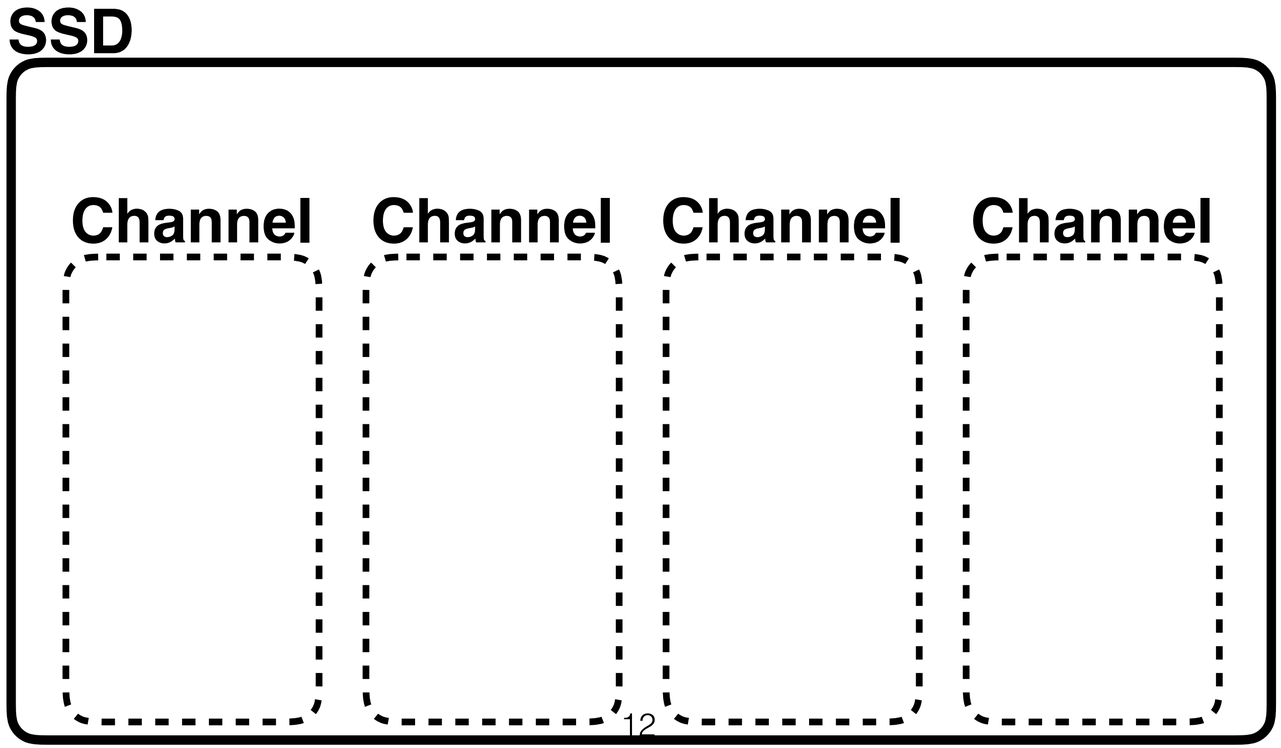


Rule #1: Request Scale

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

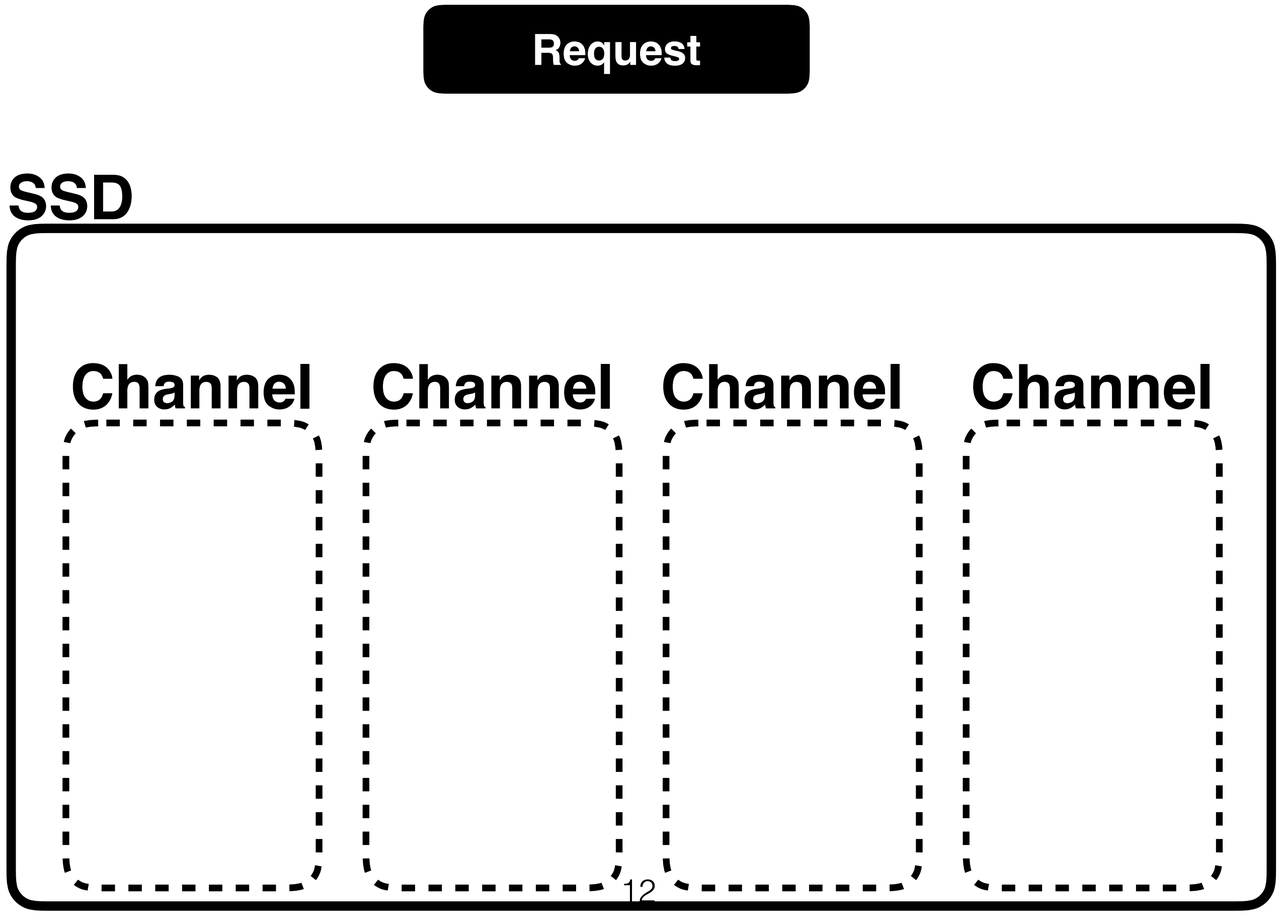
Rule #1: Request Scale

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



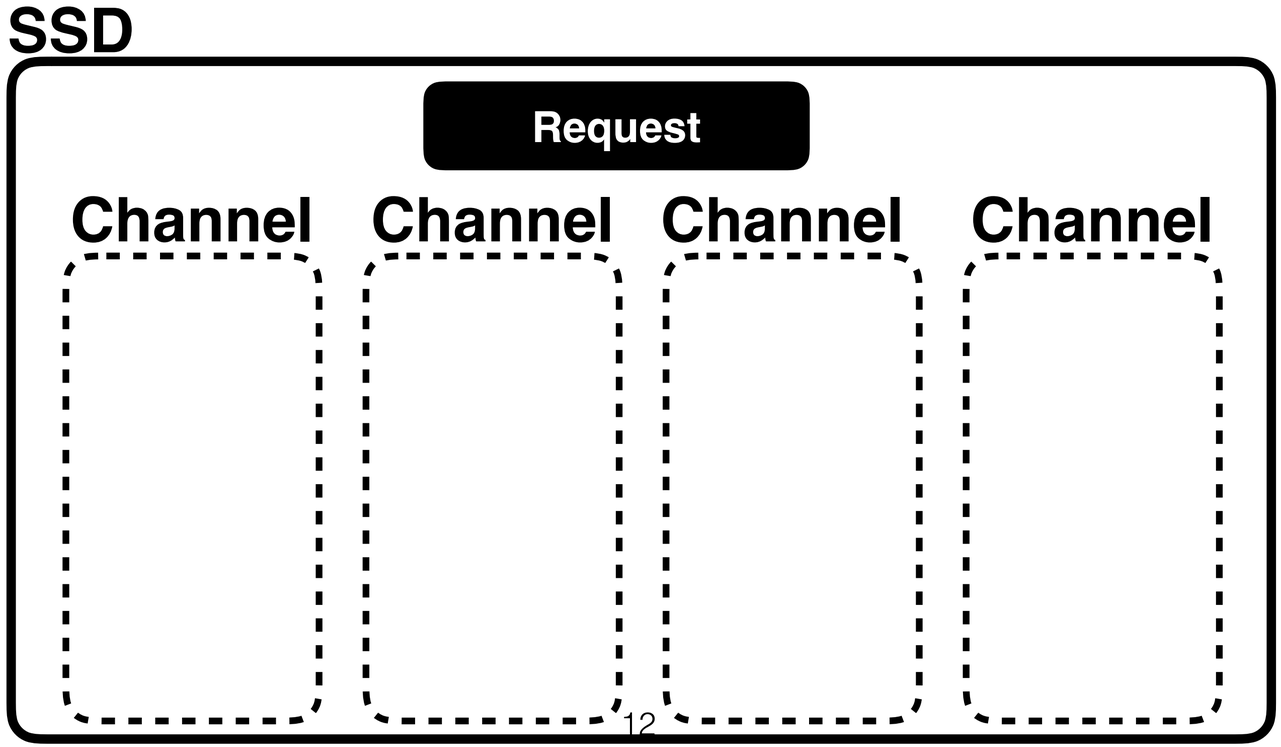
Rule #1: Request Scale

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



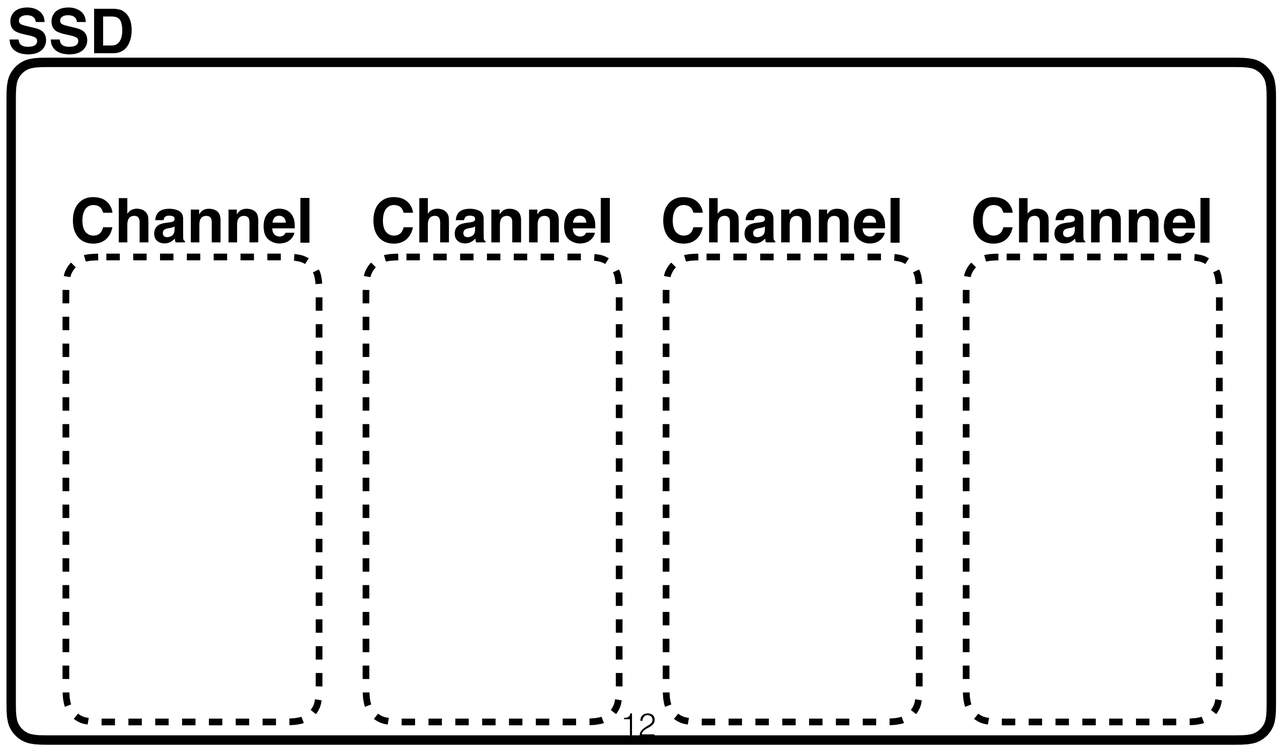
Rule #1: Request Scale

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



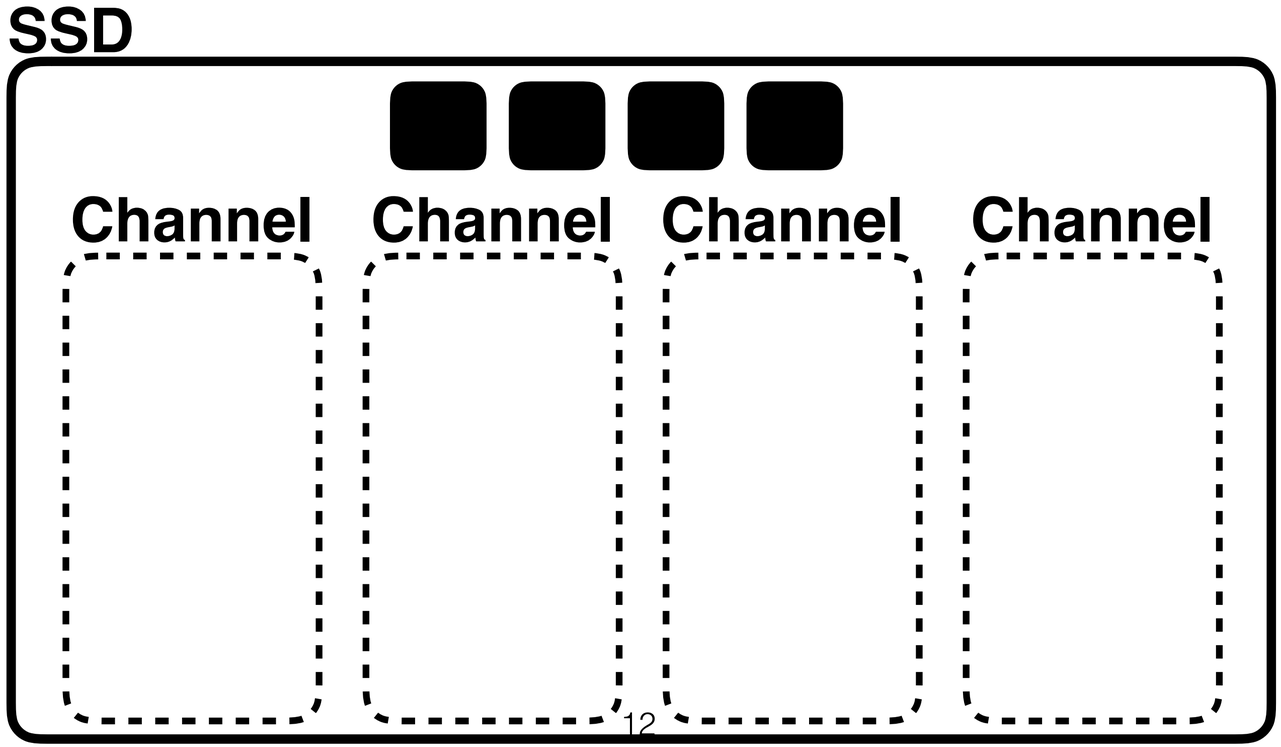
Rule #1: Request Scale

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



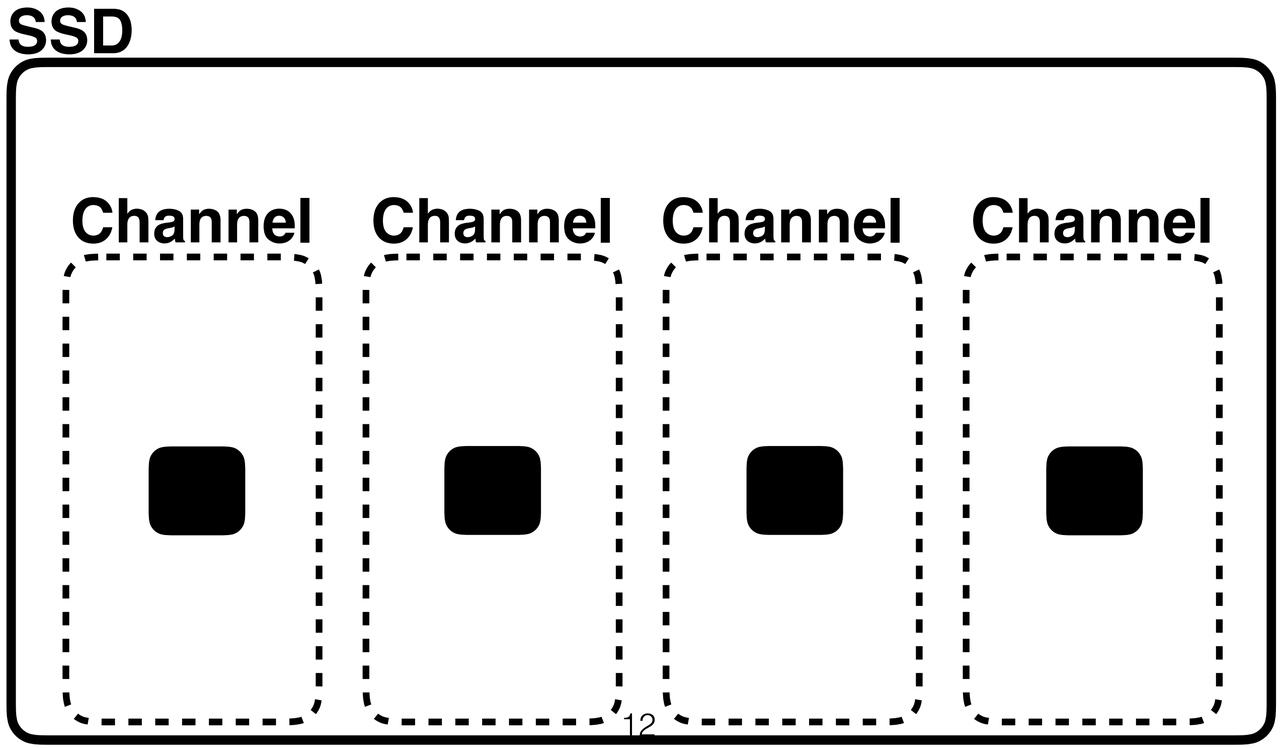
Rule #1: Request Scale

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



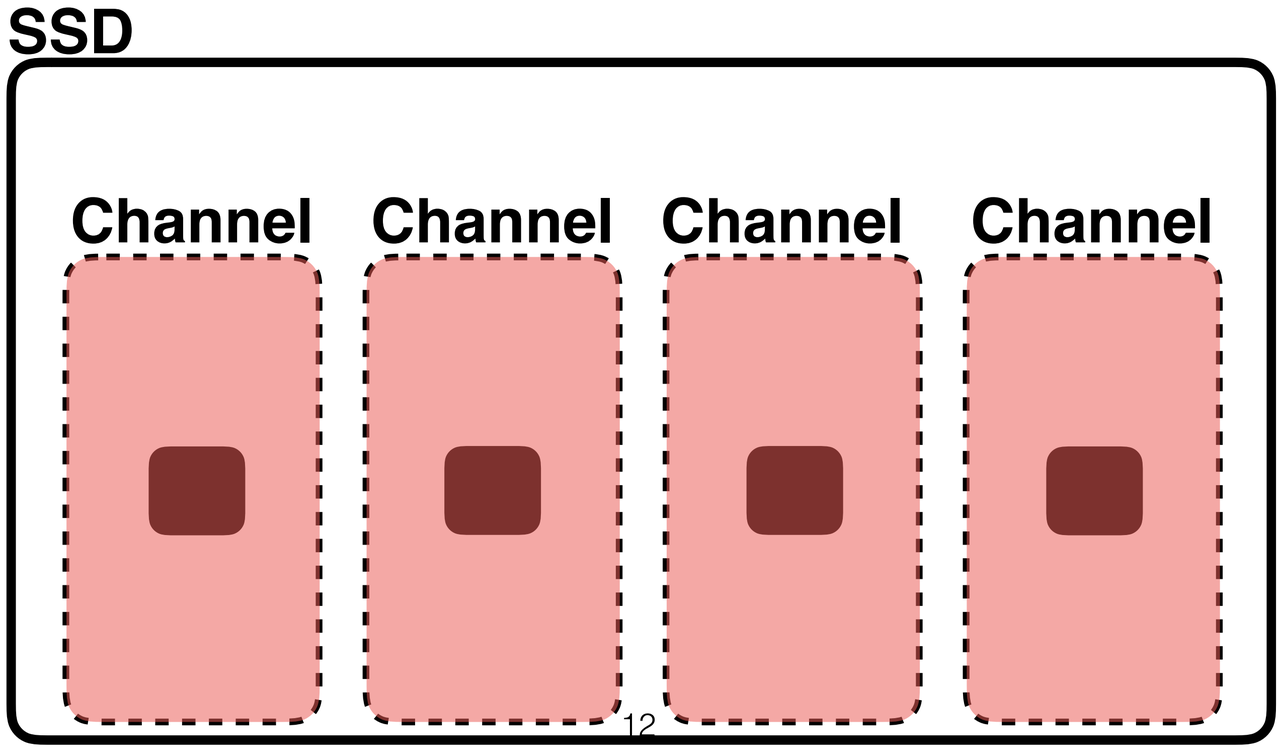
Rule #1: Request Scale

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



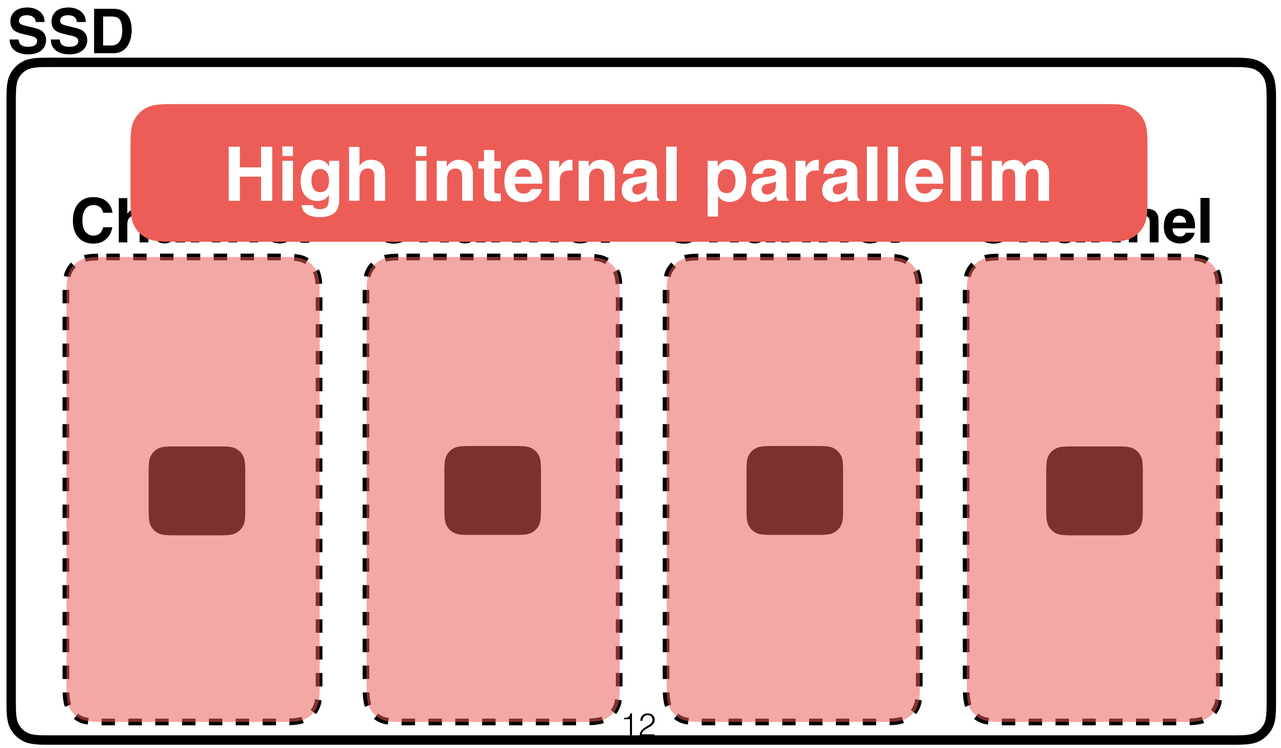
Rule #1: Request Scale

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.

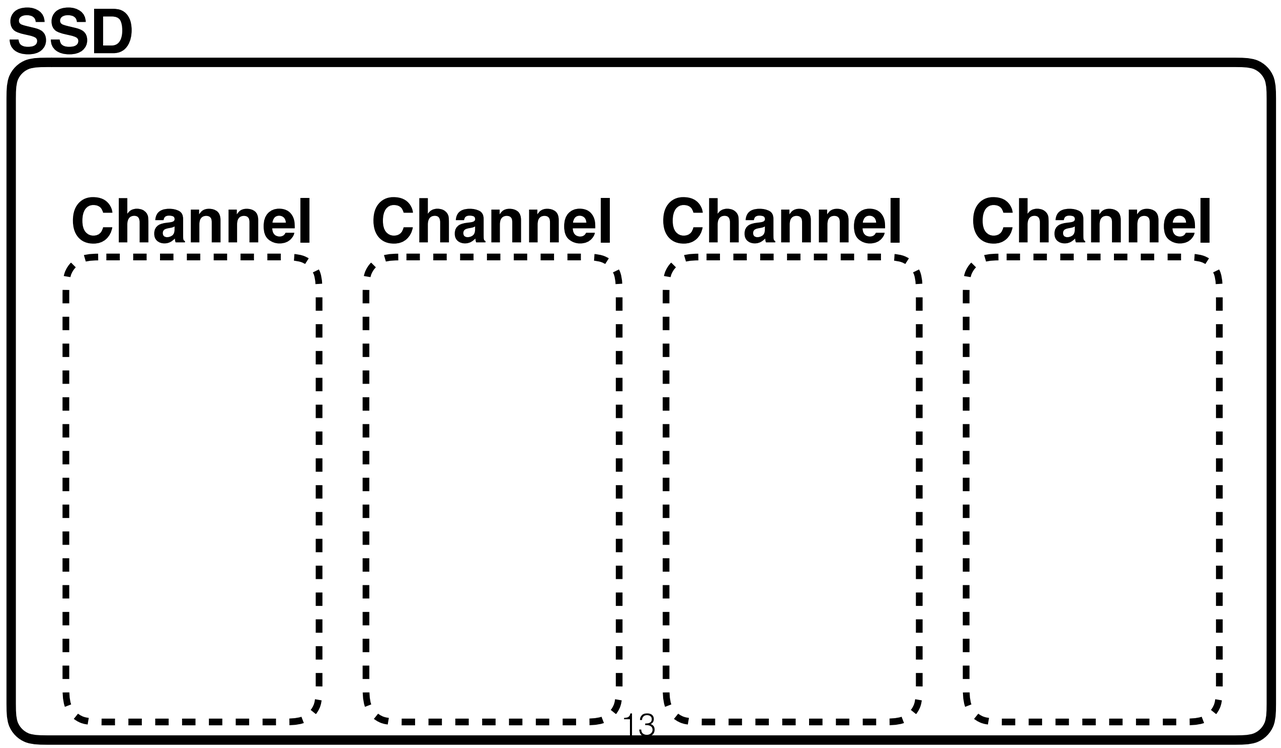


Rule #1: Request Scale

Violation

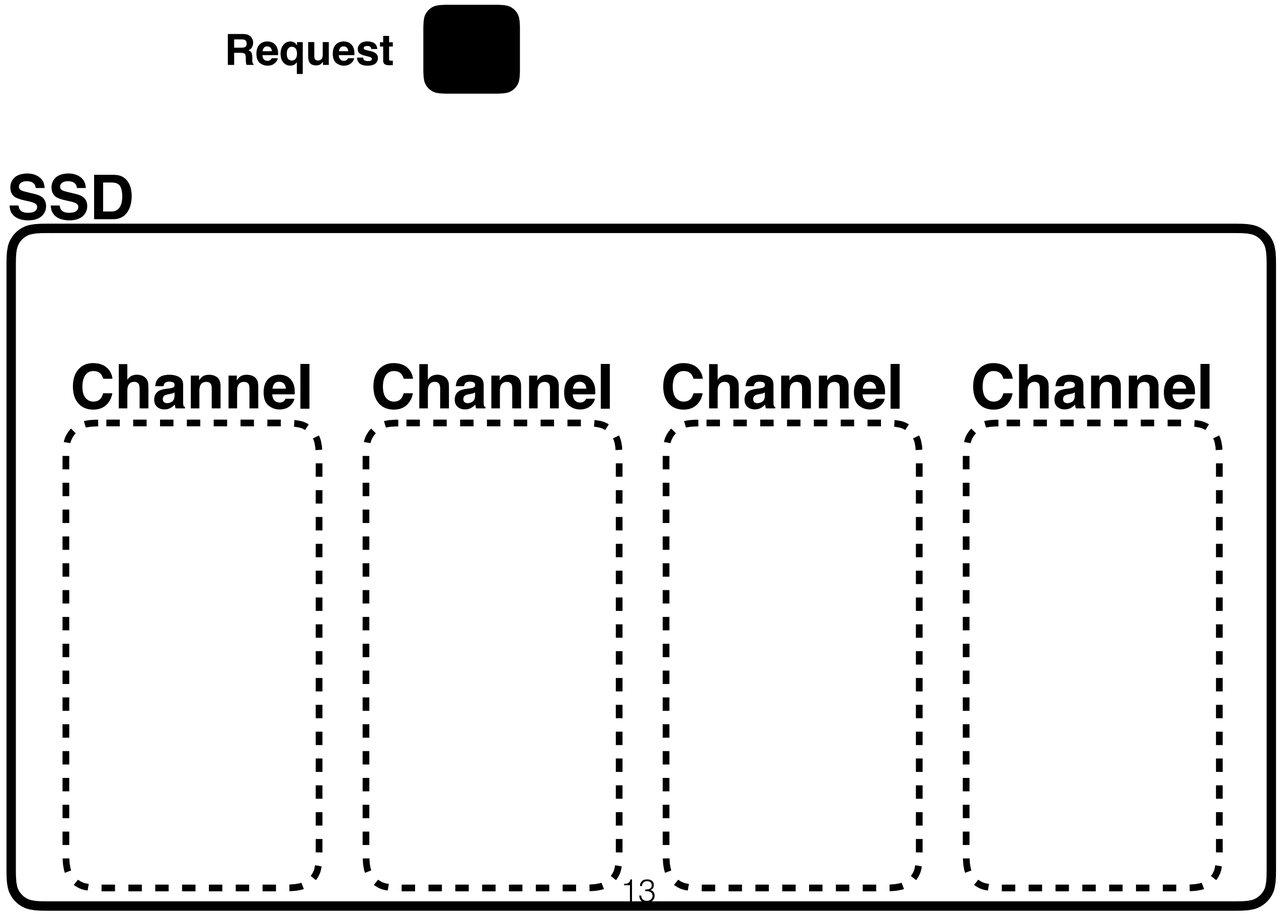
Rule #1: Request Scale

Violation



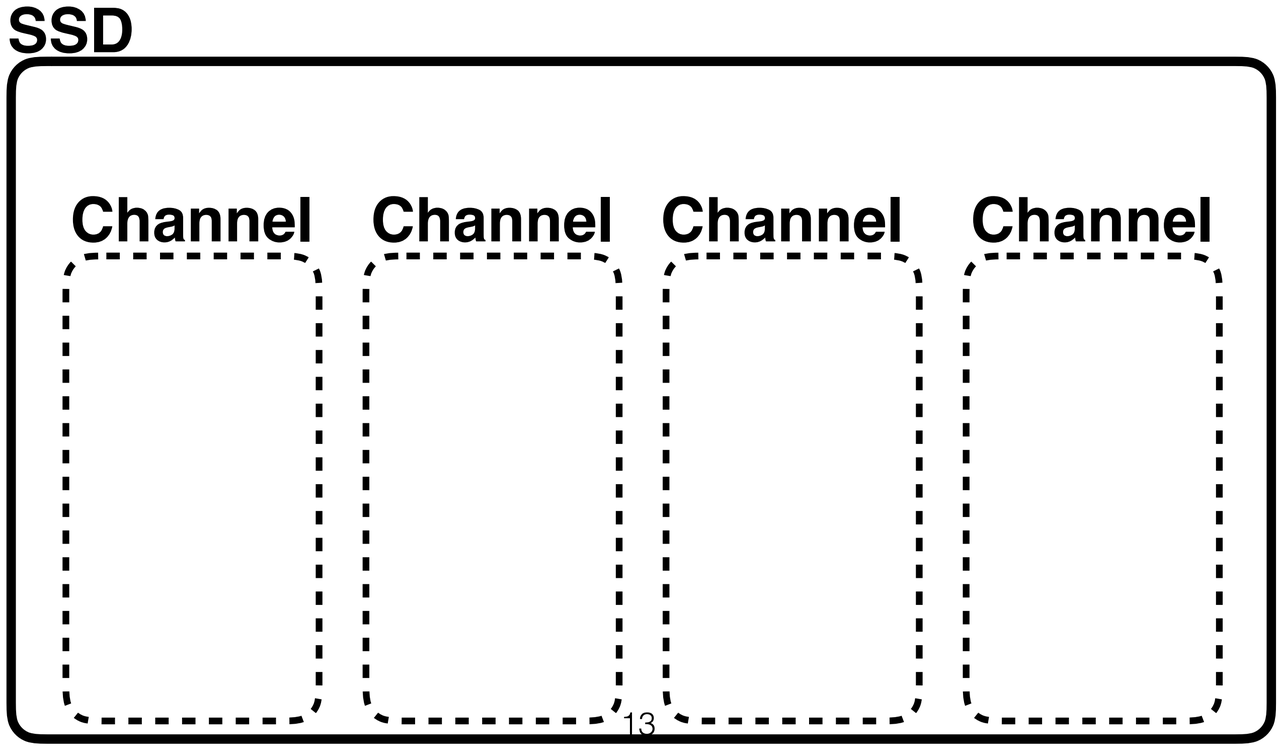
Rule #1: Request Scale

Violation



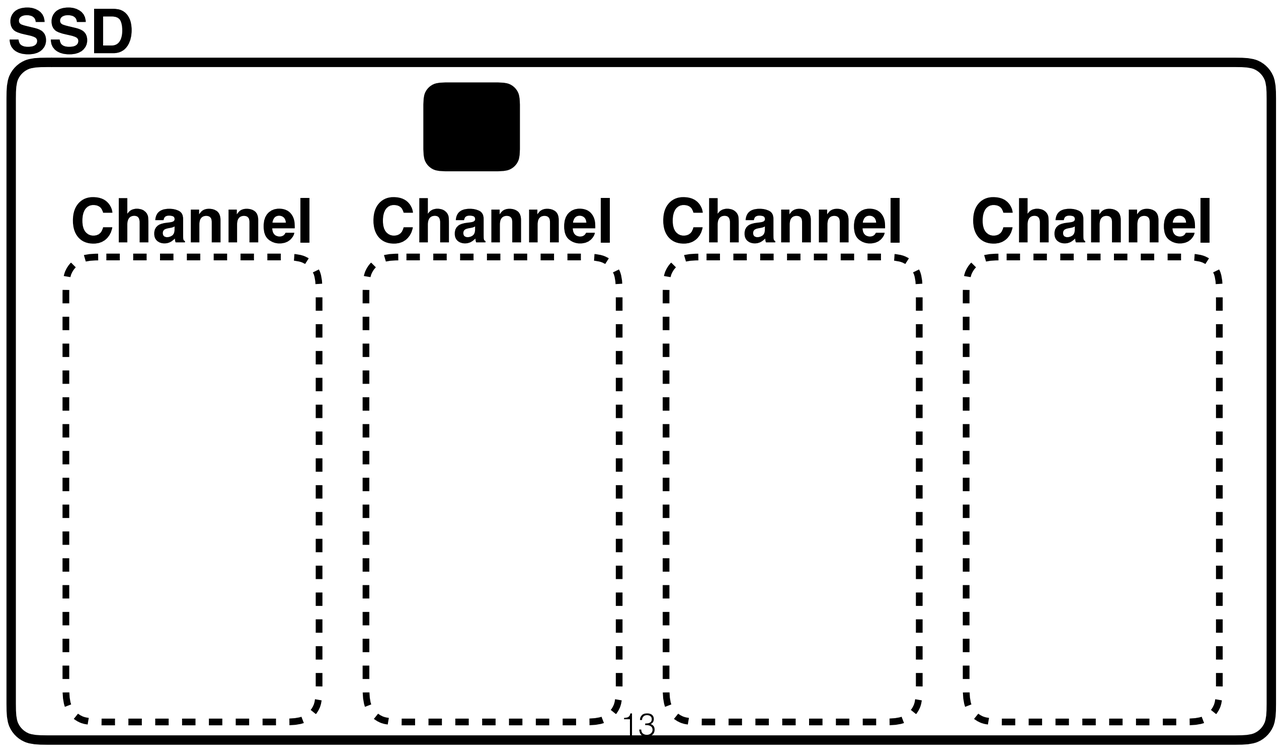
Rule #1: Request Scale

Violation



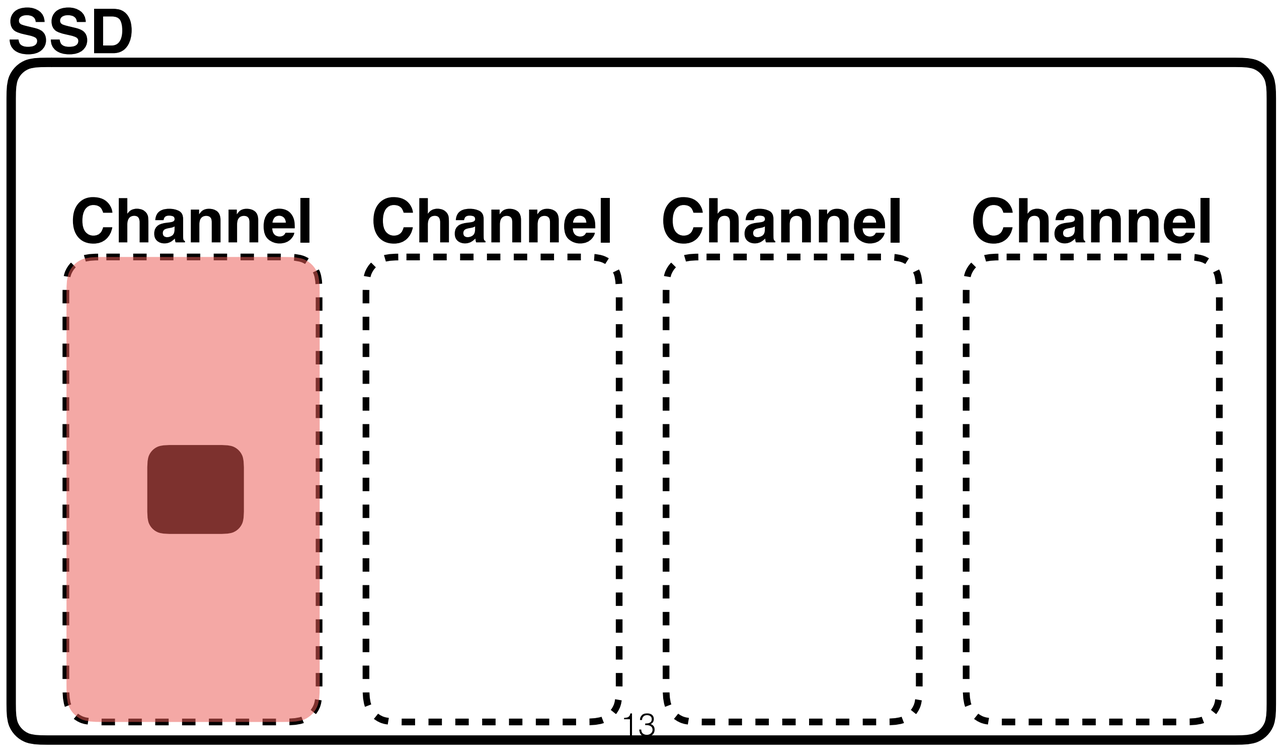
Rule #1: Request Scale

Violation



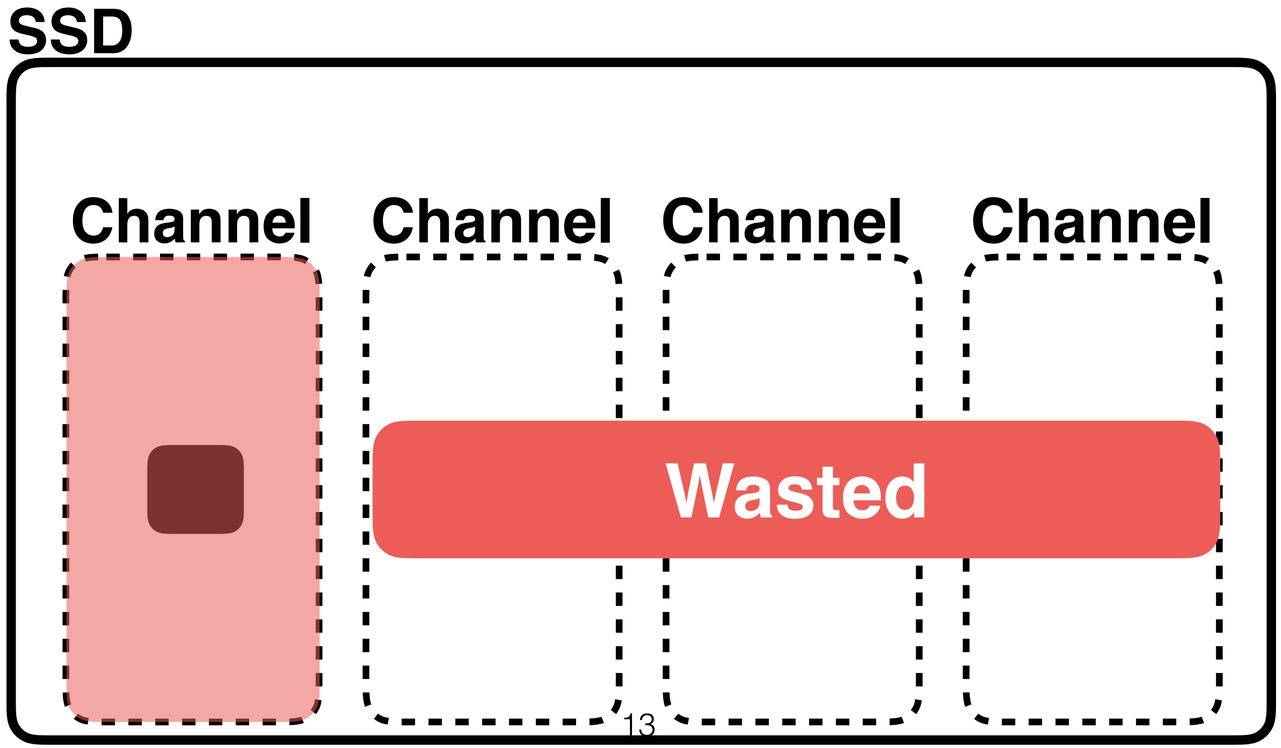
Rule #1: Request Scale

Violation



Rule #1: Request Scale

Violation

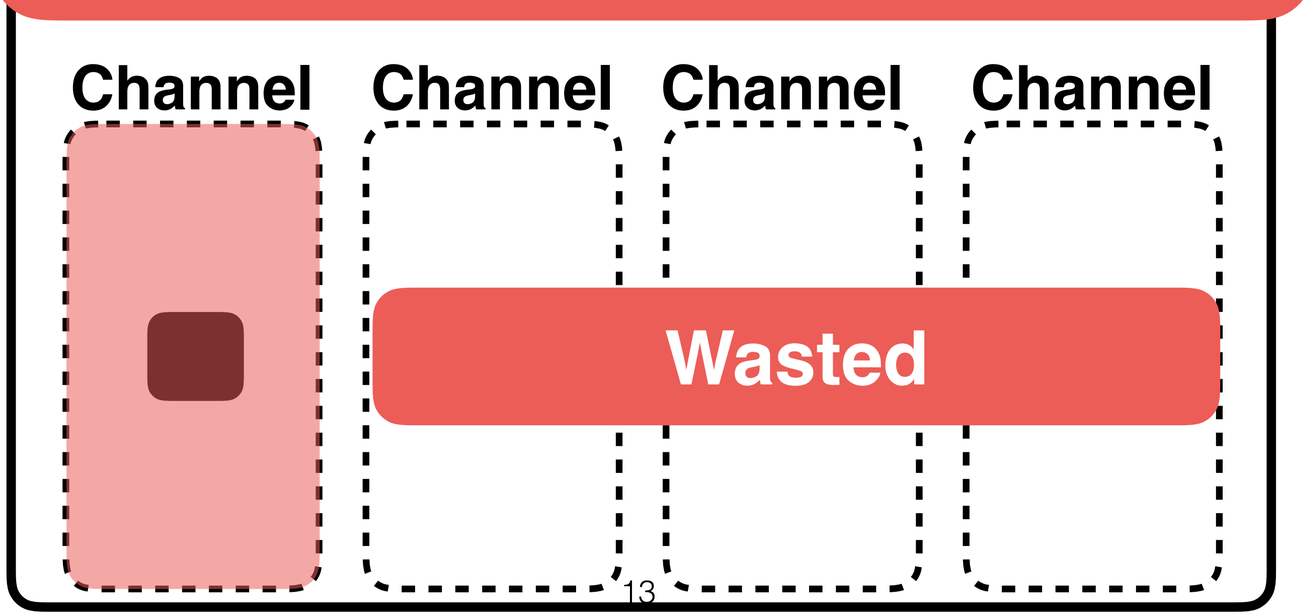


Rule #1: Request Scale

Violation

If you violate the rule:

- Low internal parallelism



Rule #1: Request Scale

Violation

If you violate the rule:

- Low internal parallelism

Performance impact:

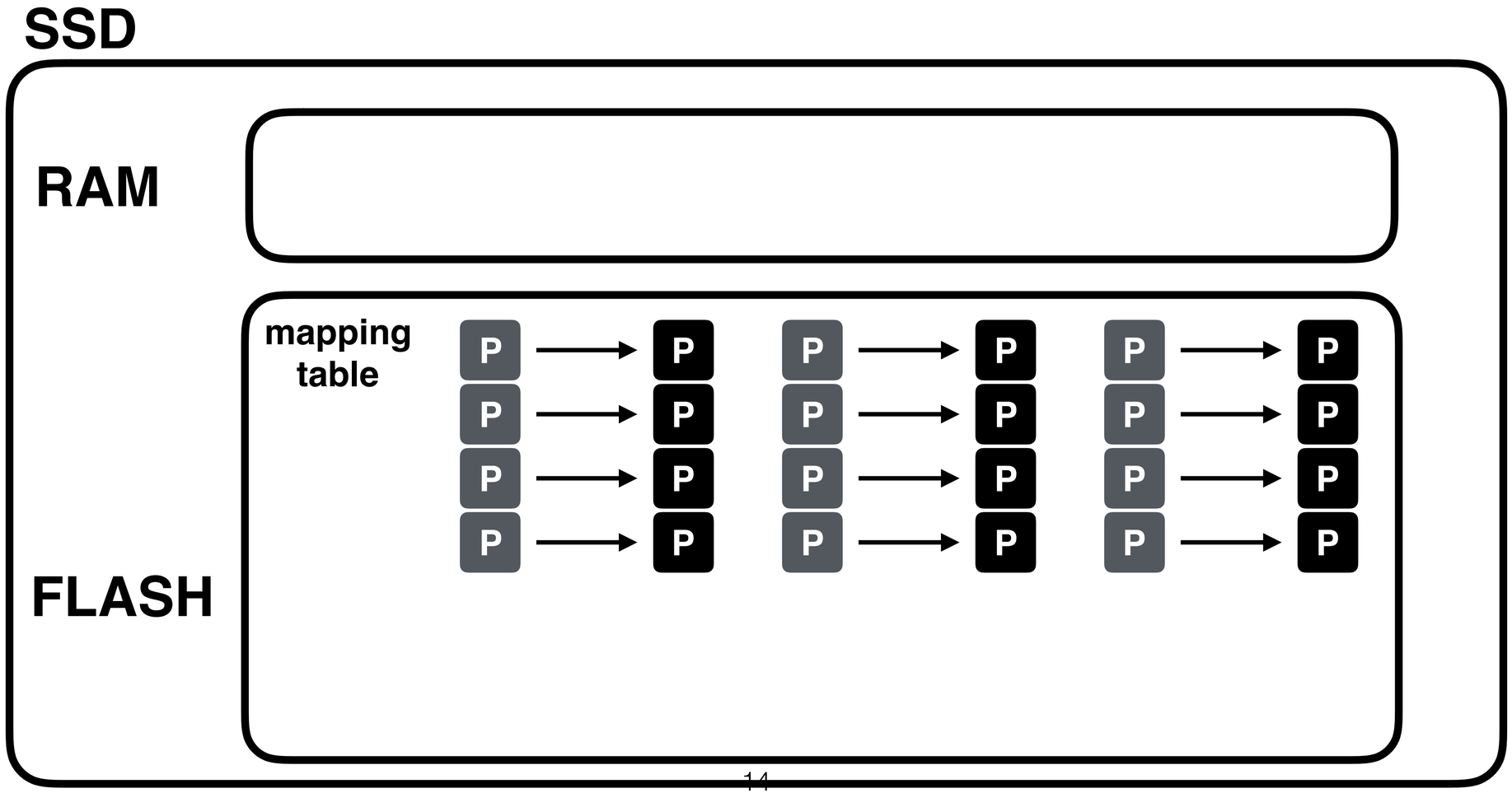
18x read bandwidth

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.

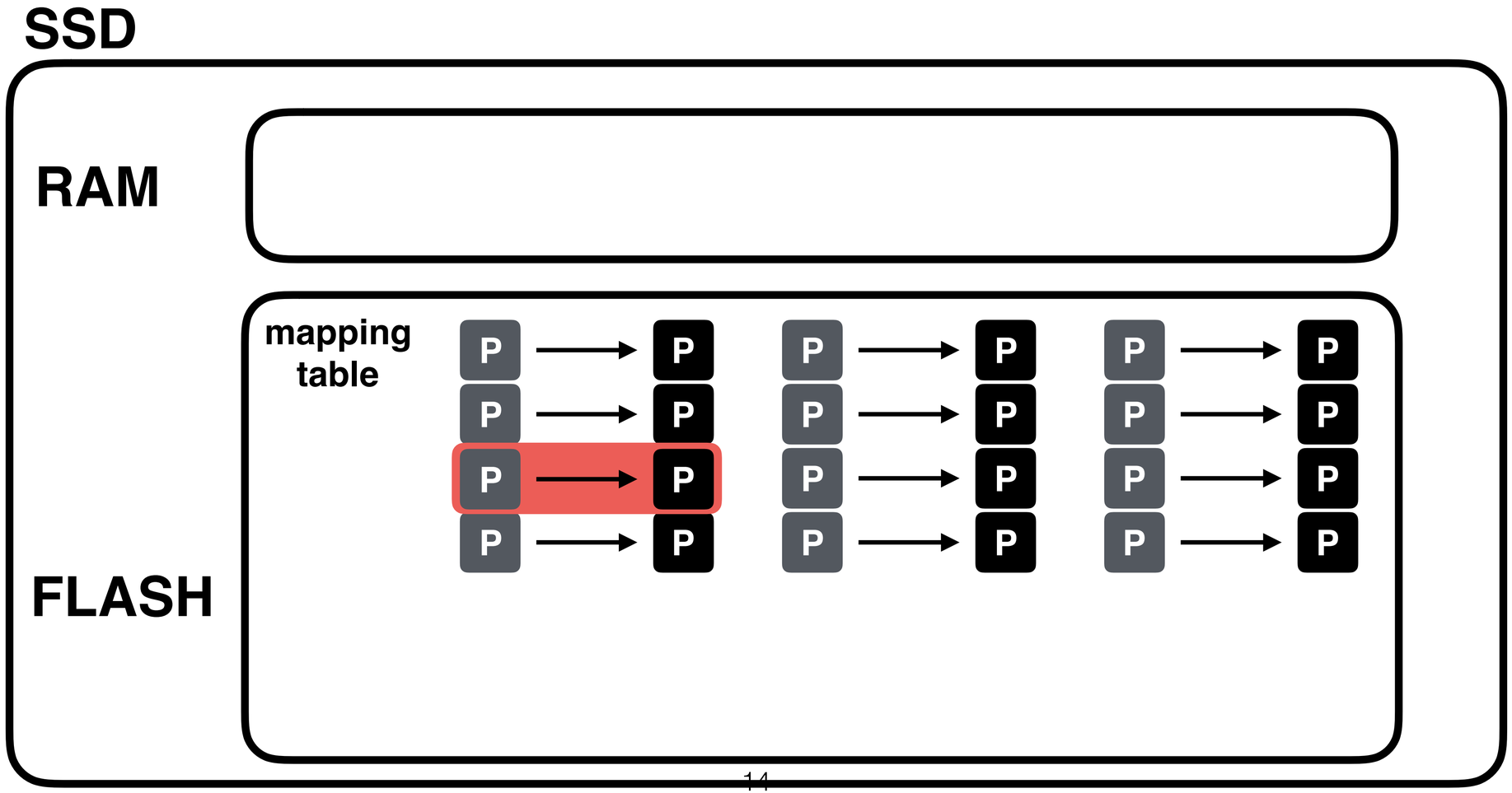
Rule 2: Locality

SSD clients should access with locality



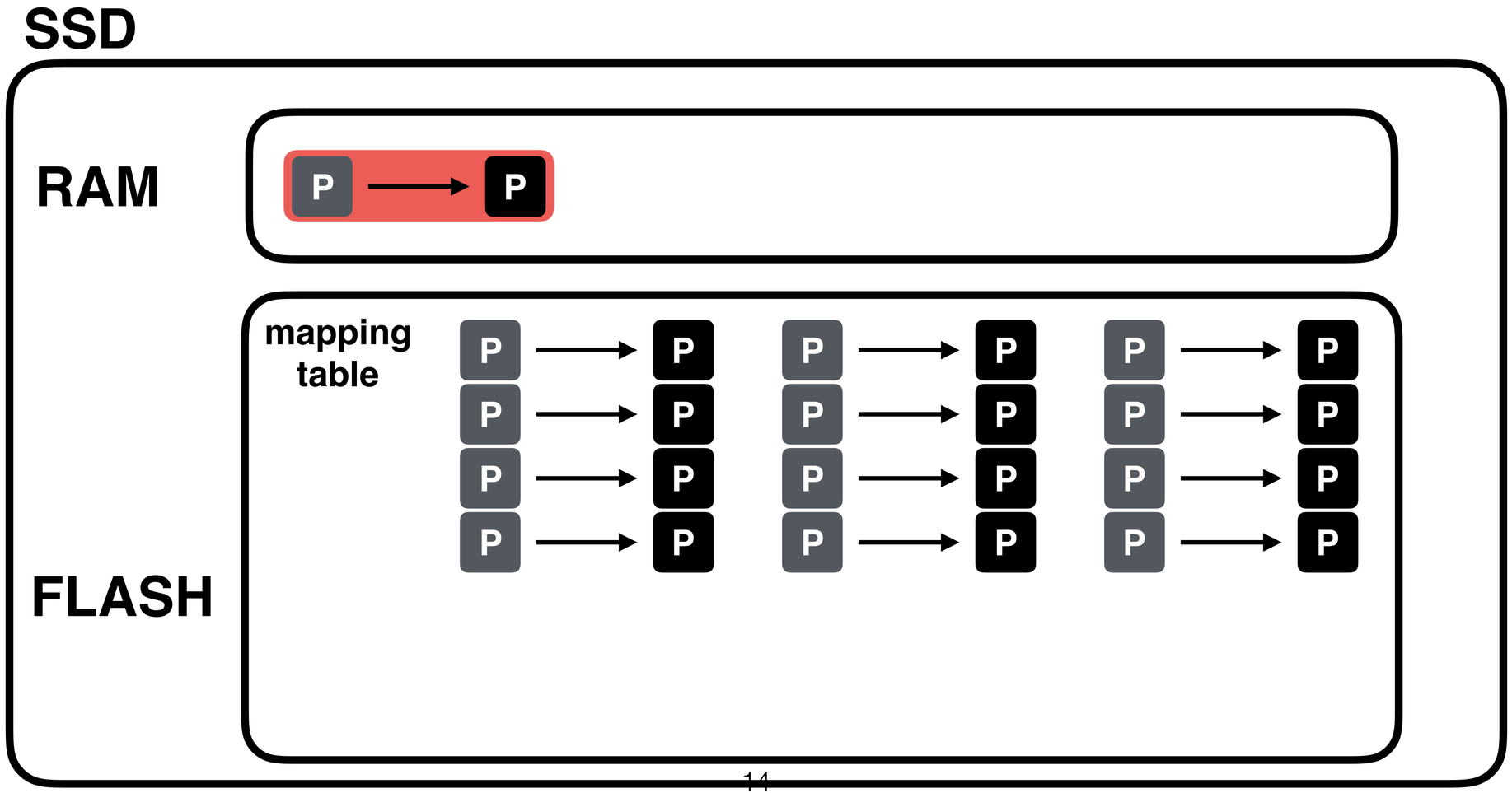
Rule 2: Locality

SSD clients should access with locality



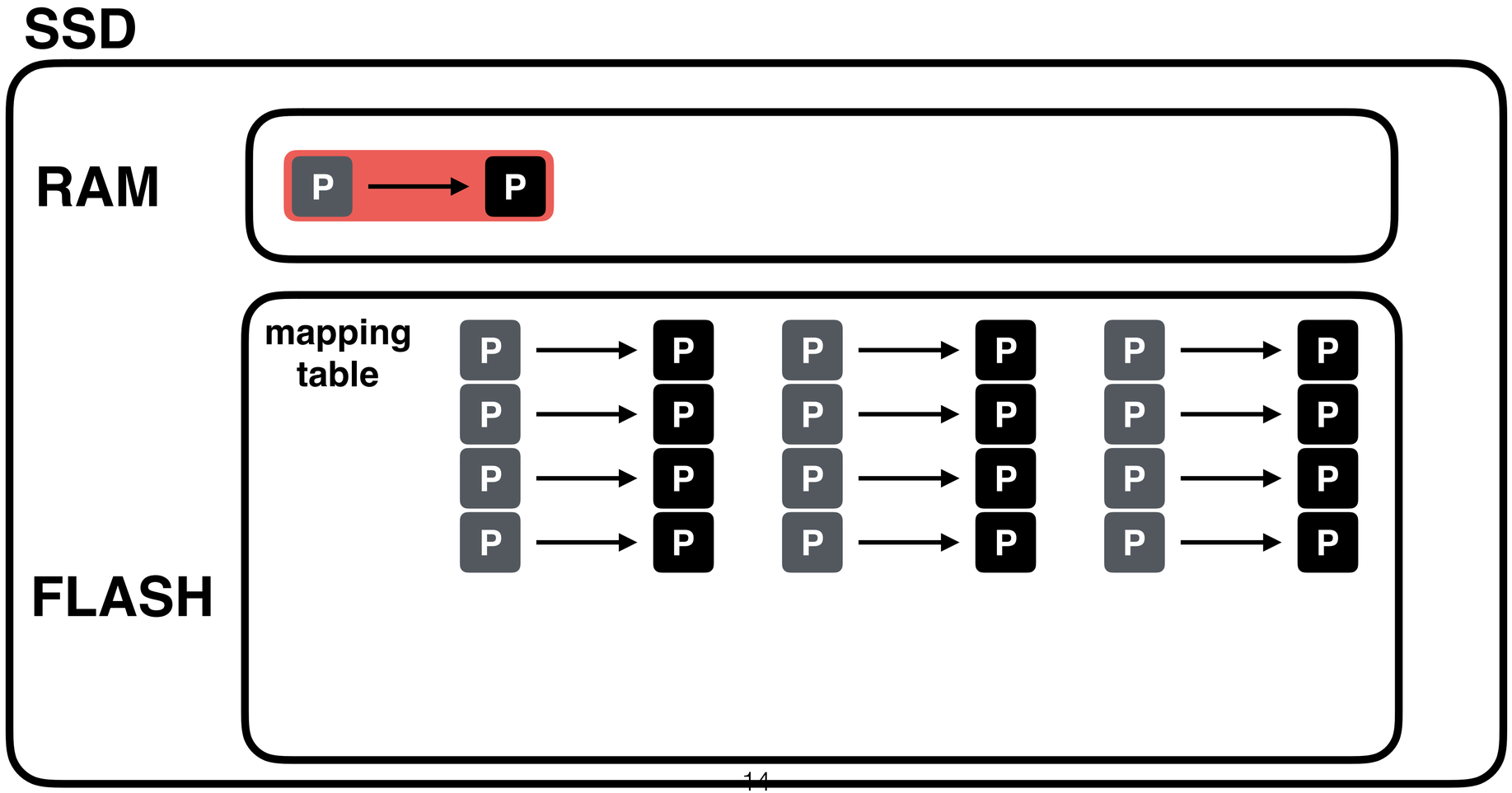
Rule 2: Locality

SSD clients should access with locality



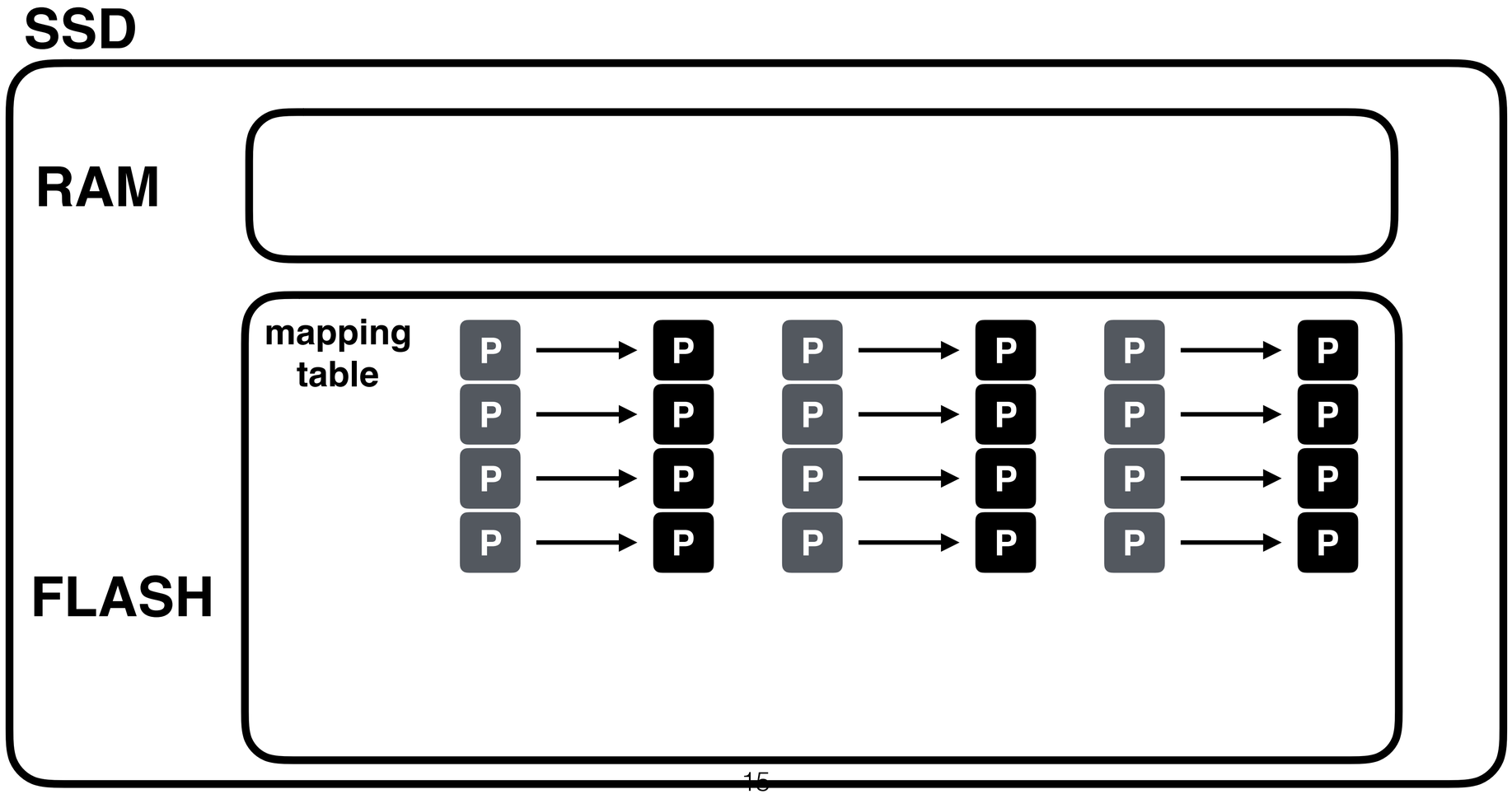
Rule 2: Locality

SSD clients should access with locality



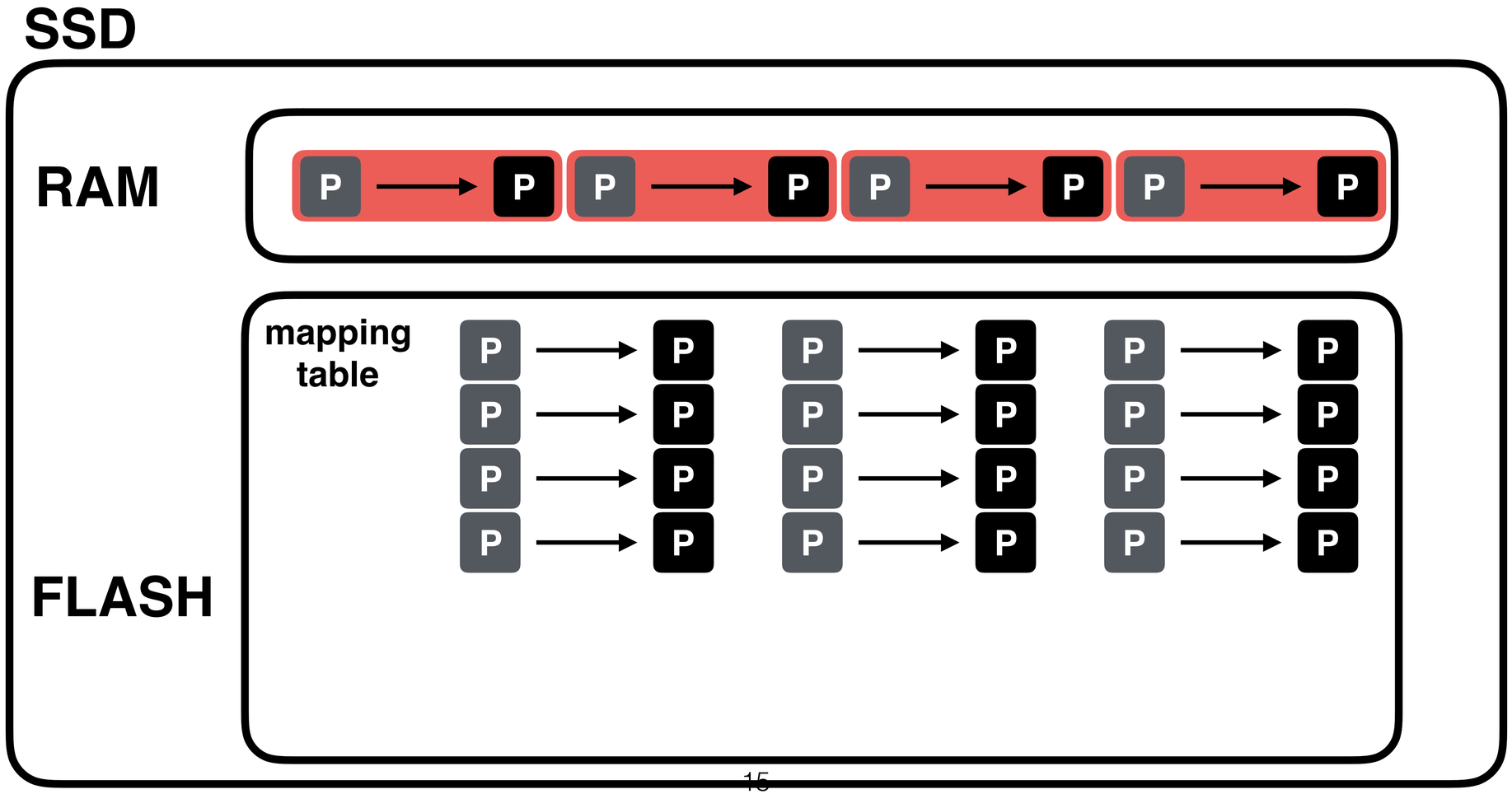
Rule 2: Locality Violation

SSD clients should access with locality



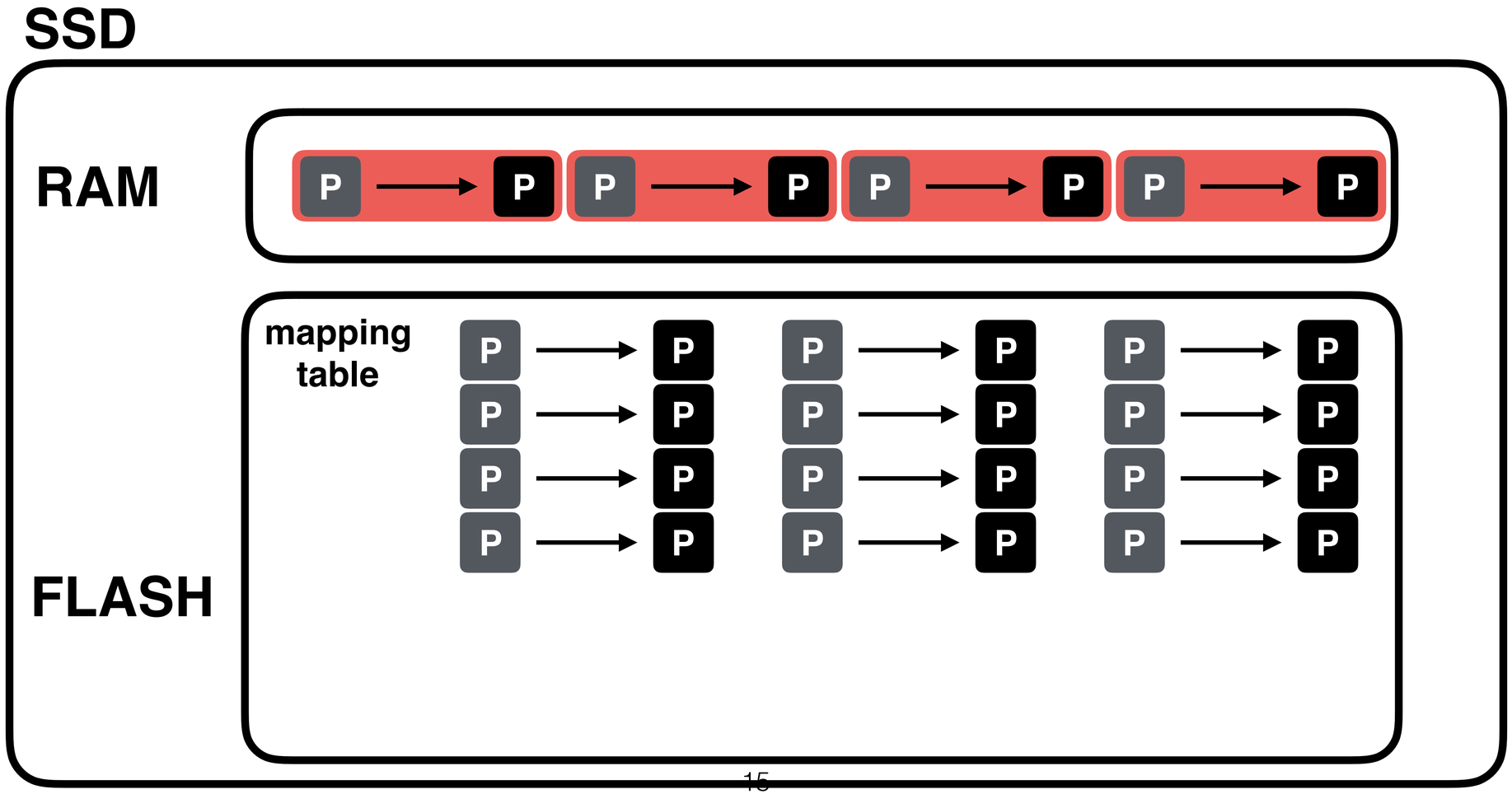
Rule 2: Locality Violation

SSD clients should access with locality



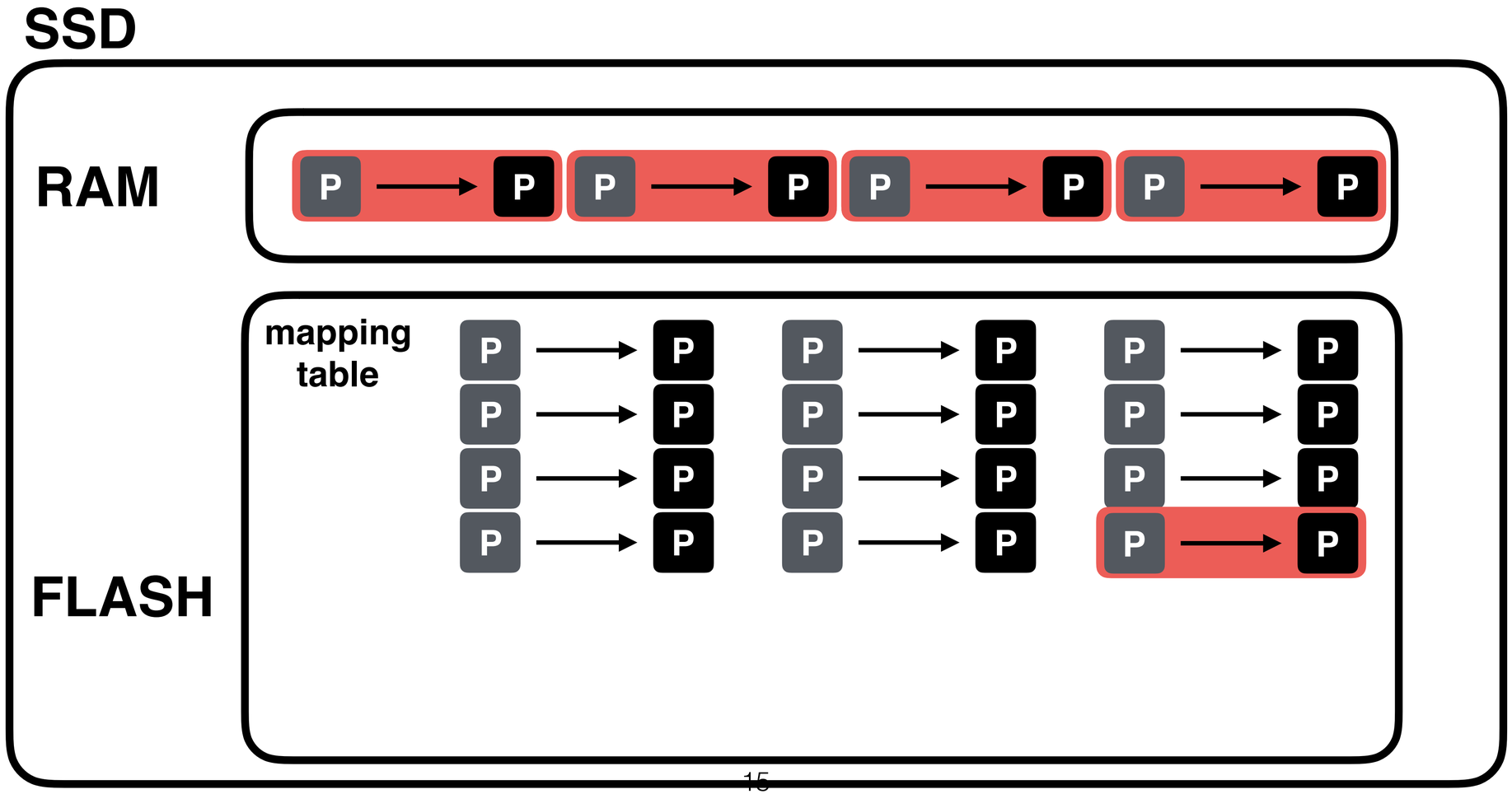
Rule 2: Locality Violation

SSD clients should access with locality



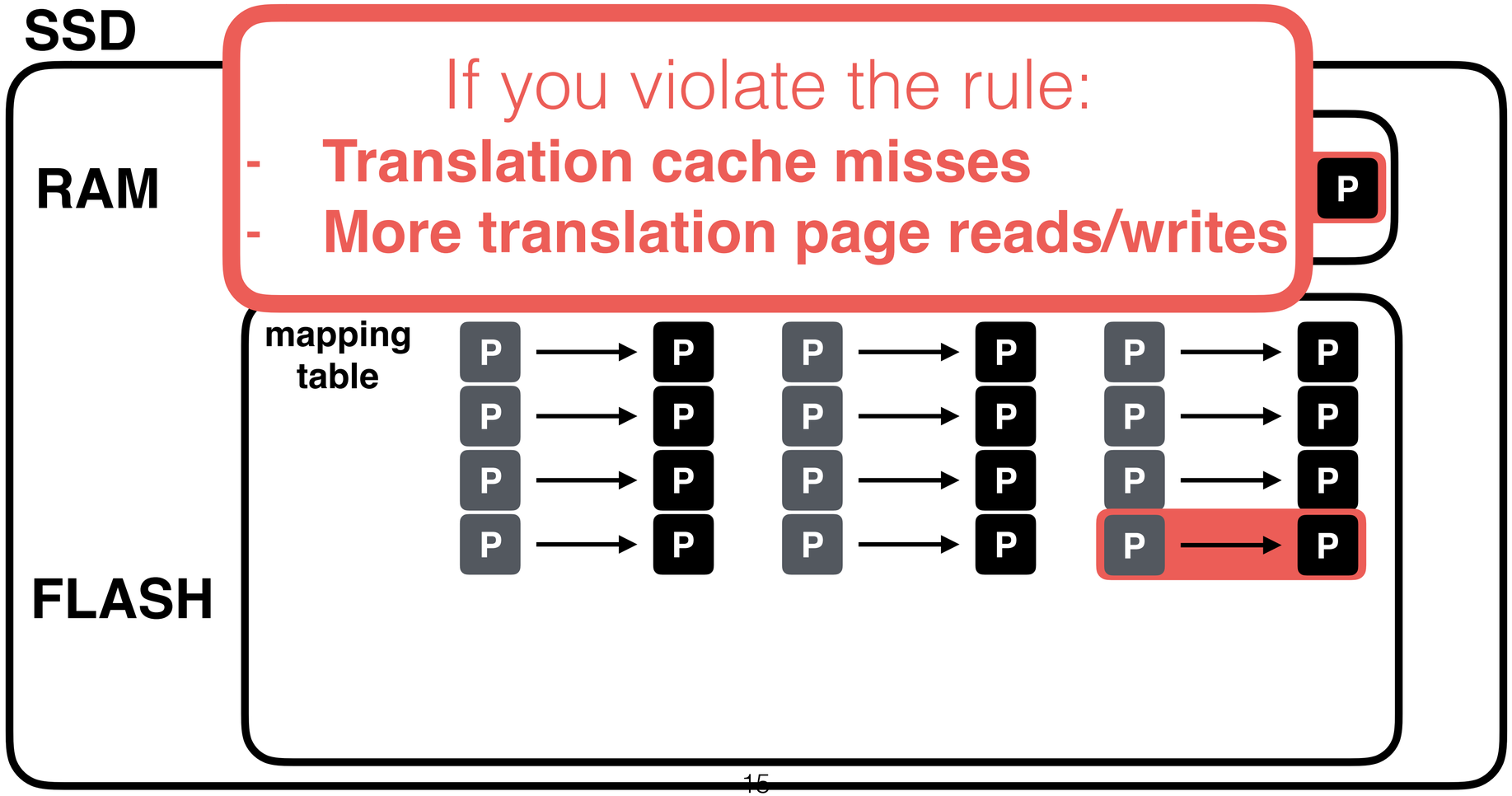
Rule 2: Locality Violation

SSD clients should access with locality



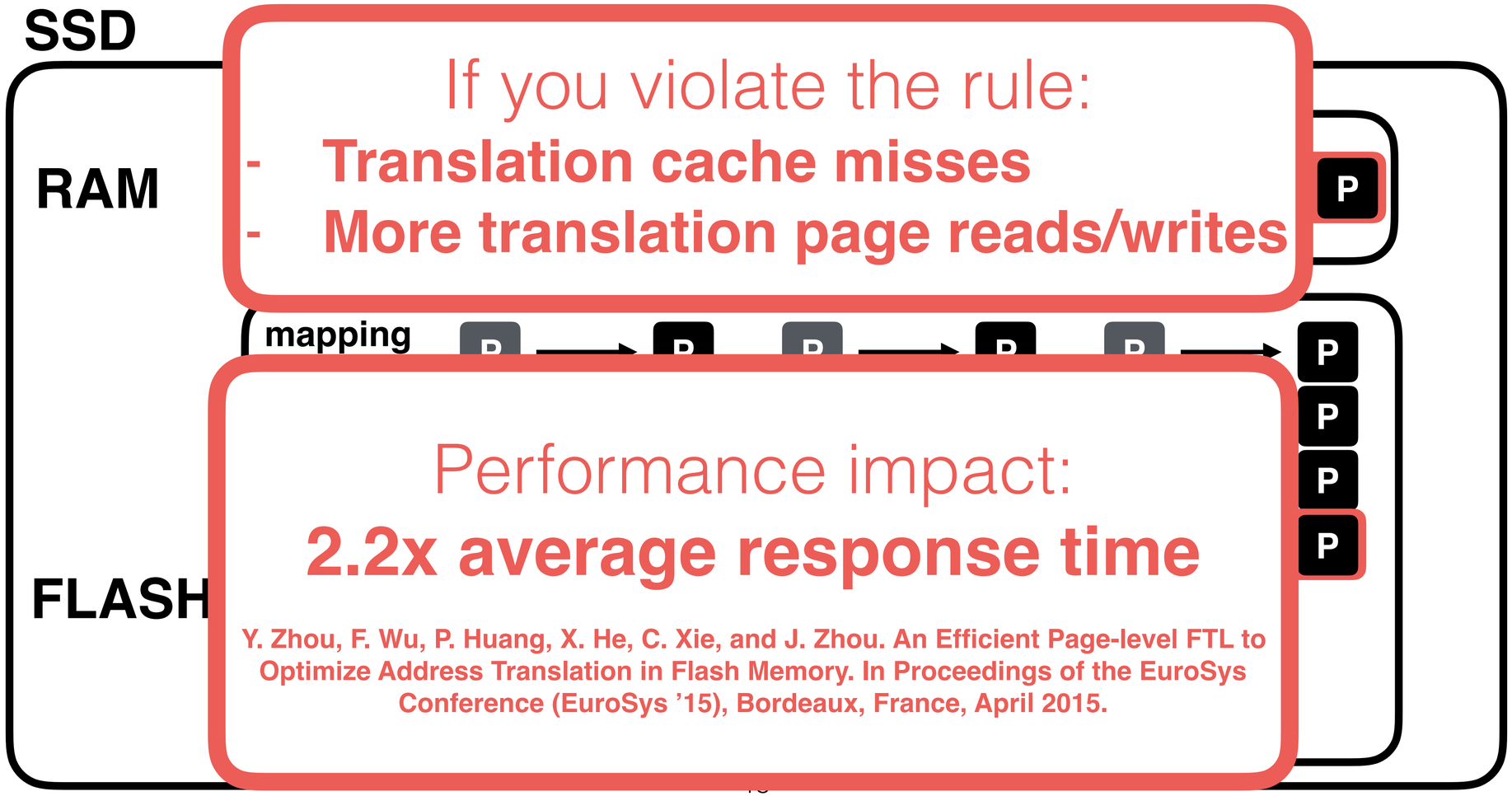
Rule 2: Locality Violation

SSD clients should access with locality



Rule 2: Locality Violation

SSD clients should access with locality



Rule 3: Aligned Sequentiality

1. Start writing at block boundary

2. Write sequentially

RAM

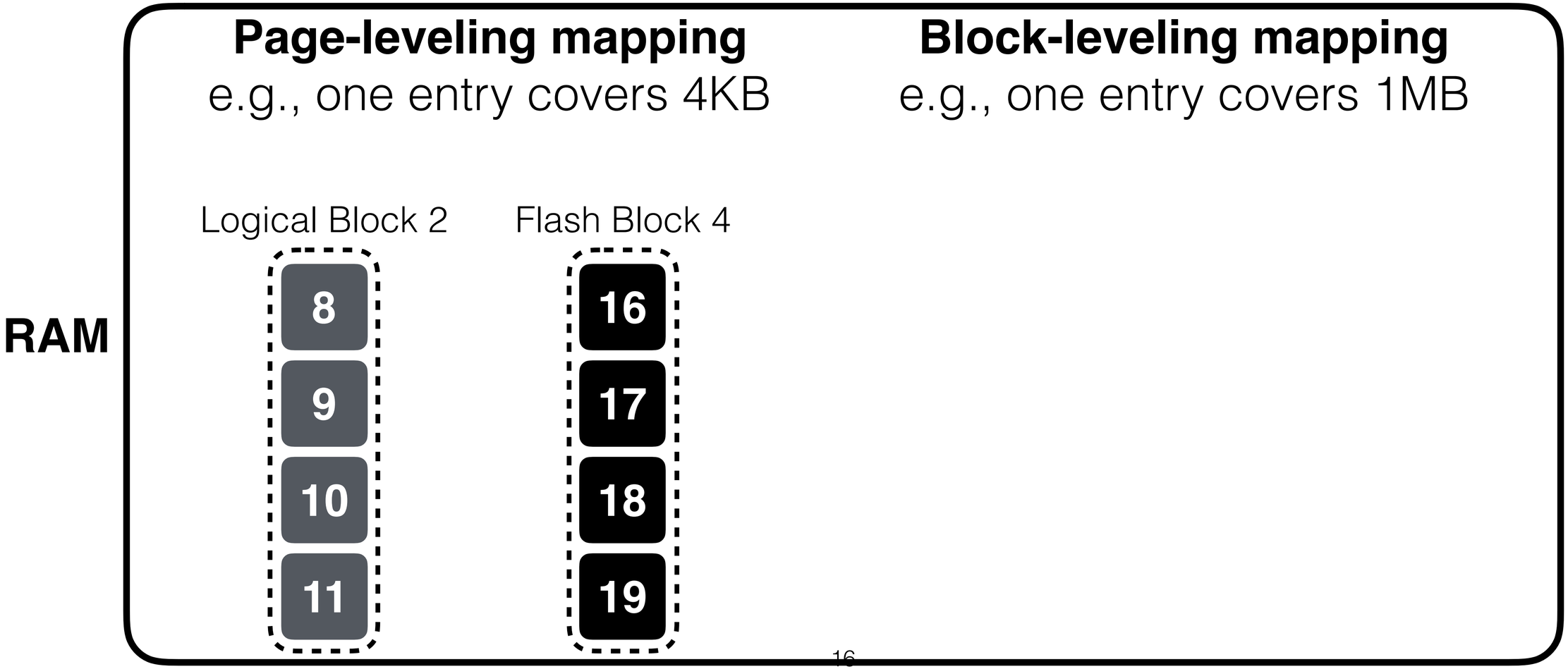
Page-leveling mapping
e.g., one entry covers 4KB

Block-leveling mapping
e.g., one entry covers 1MB

Rule 3: Aligned Sequentiality

1. Start writing at block boundary

2. Write sequentially



Rule 3: Aligned Sequentiality

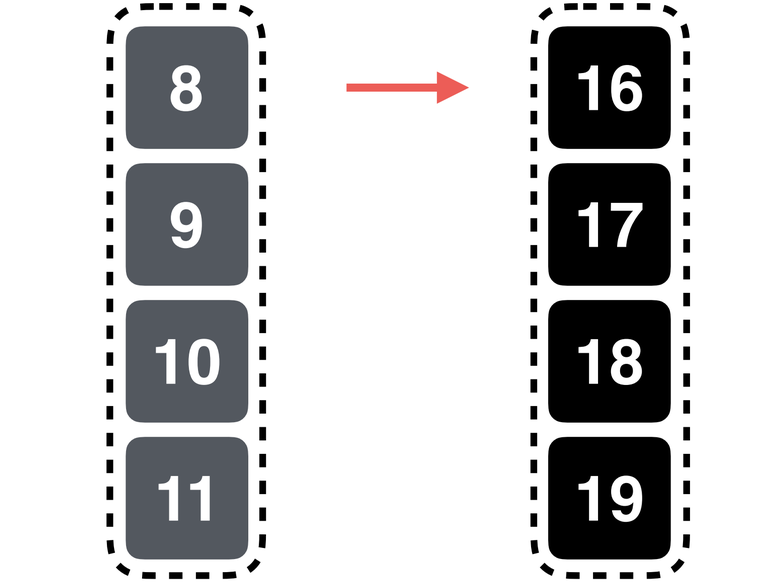
1. Start writing at block boundary

2. Write sequentially

RAM

Page-leveling mapping
e.g., one entry covers 4KB

Block-leveling mapping
e.g., one entry covers 1MB



Rule 3: Aligned Sequentiality

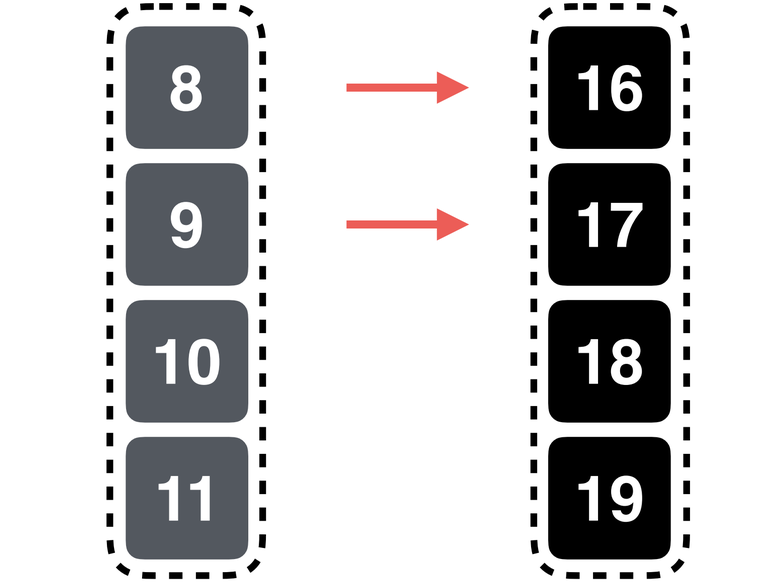
1. Start writing at block boundary

2. Write sequentially

RAM

Page-leveling mapping
e.g., one entry covers 4KB

Block-leveling mapping
e.g., one entry covers 1MB



Rule 3: Aligned Sequentiality

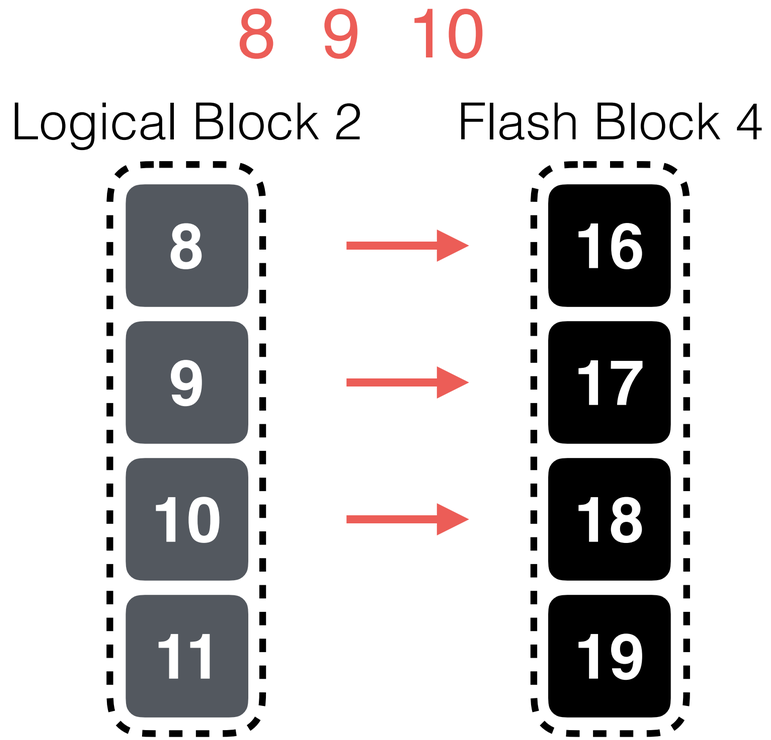
1. Start writing at block boundary

2. Write sequentially

RAM

Page-leveling mapping

e.g., one entry covers 4KB



Block-leveling mapping

e.g., one entry covers 1MB

Rule 3: Aligned Sequentiality

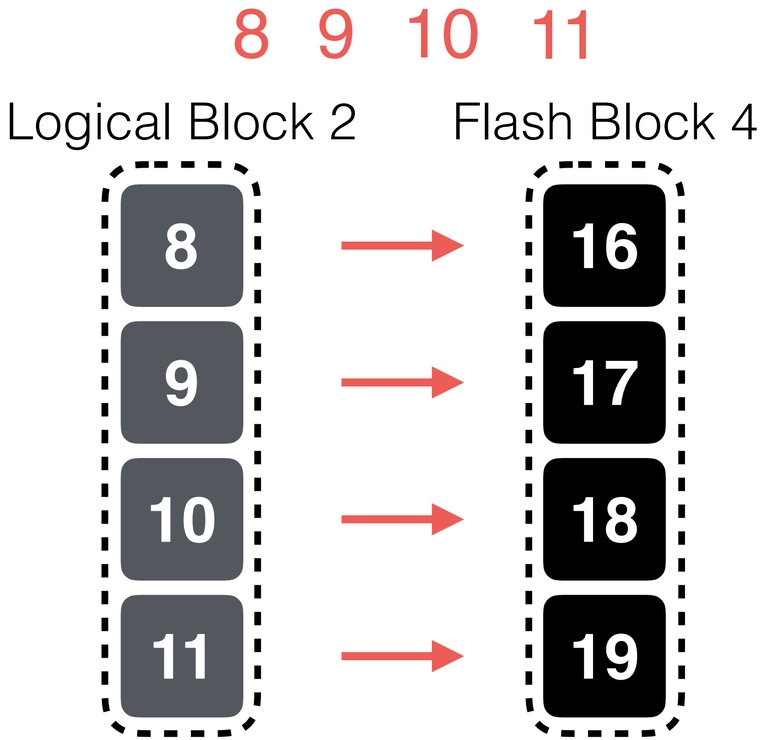
1. Start writing at block boundary

2. Write sequentially

RAM

Page-leveling mapping

e.g., one entry covers 4KB



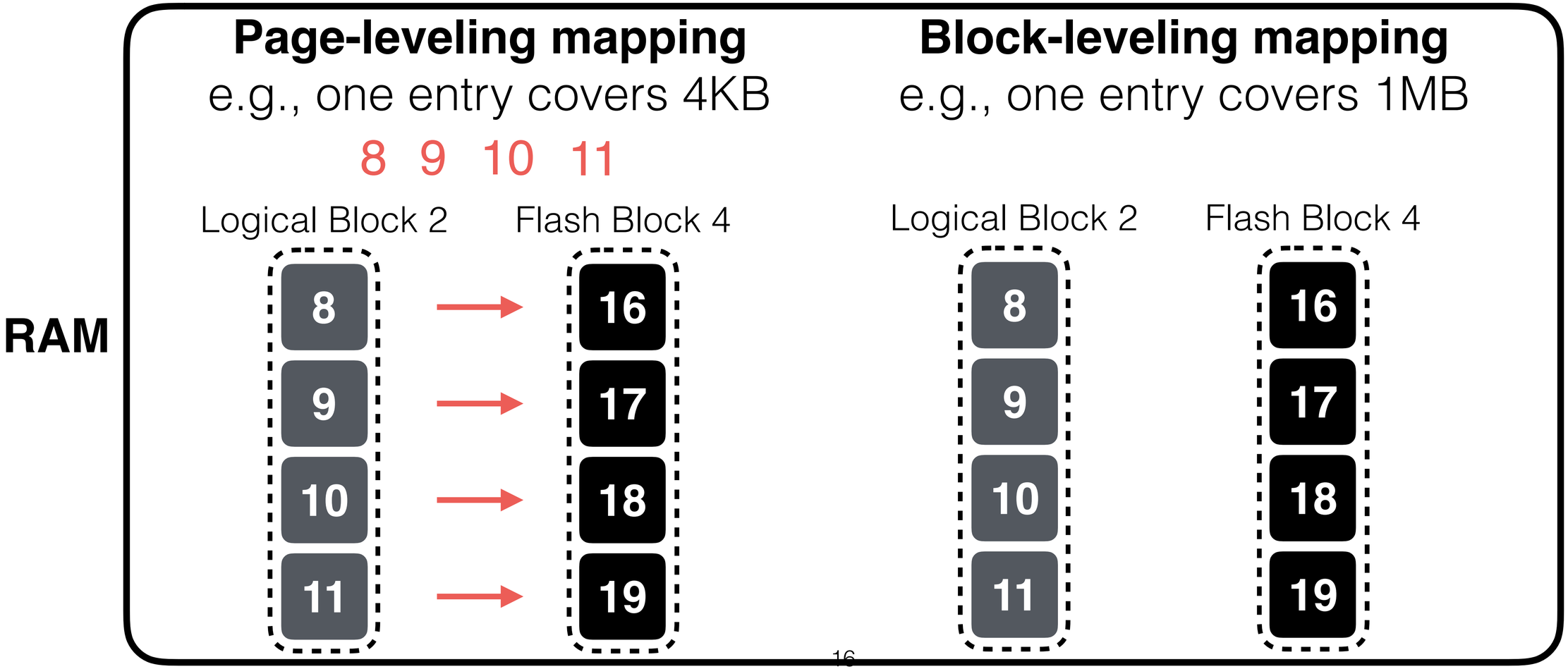
Block-leveling mapping

e.g., one entry covers 1MB

Rule 3: Aligned Sequentiality

1. Start writing at block boundary

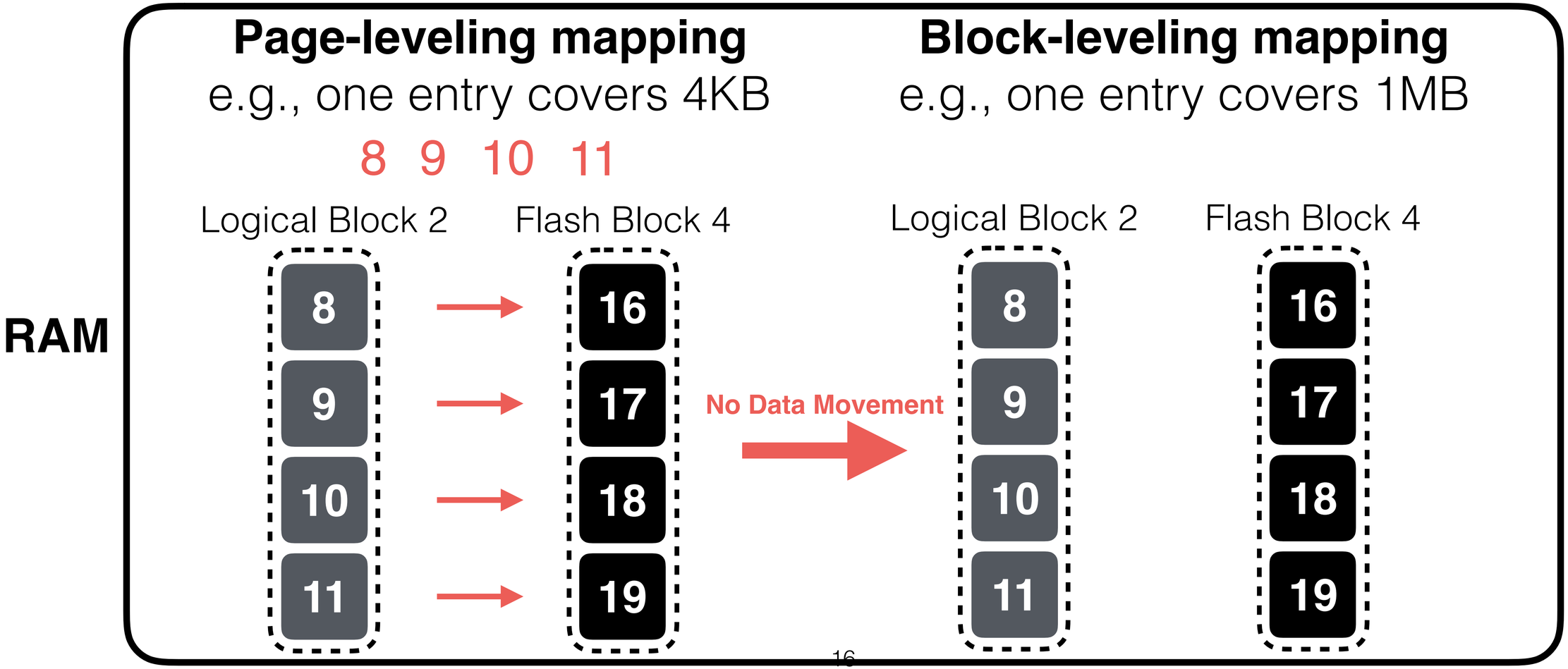
2. Write sequentially



Rule 3: Aligned Sequentiality

1. Start writing at block boundary

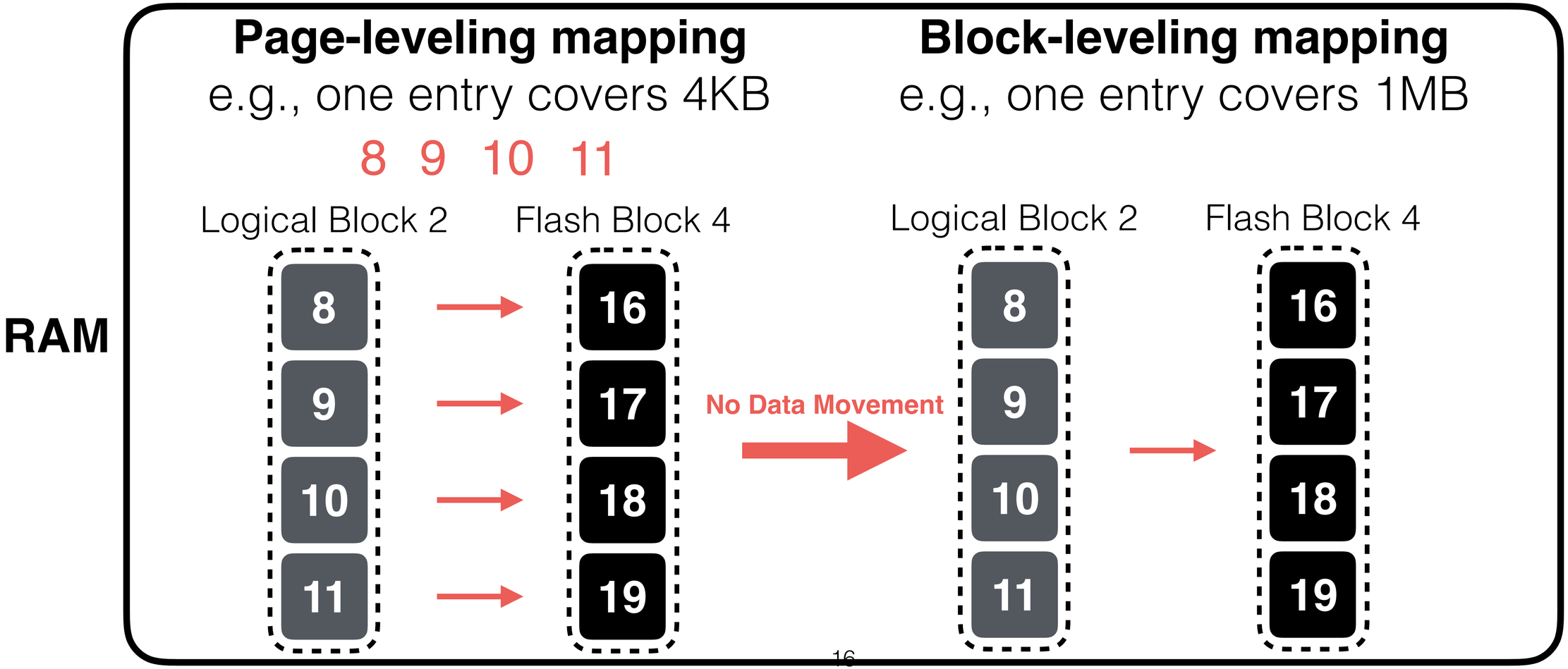
2. Write sequentially



Rule 3: Aligned Sequentiality

1. Start writing at block boundary

2. Write sequentially



Rule 3: Aligned Sequentiality

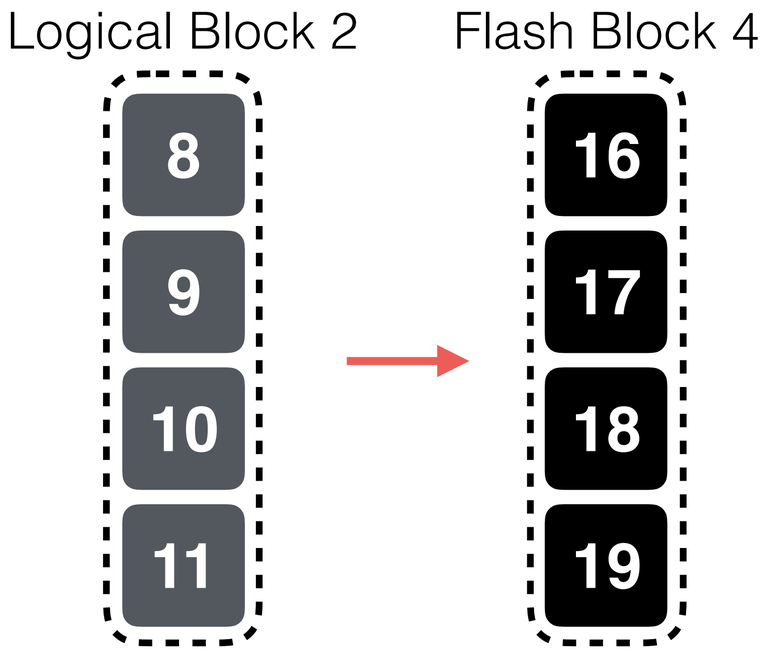
1. Start writing at block boundary

2. Write sequentially

RAM

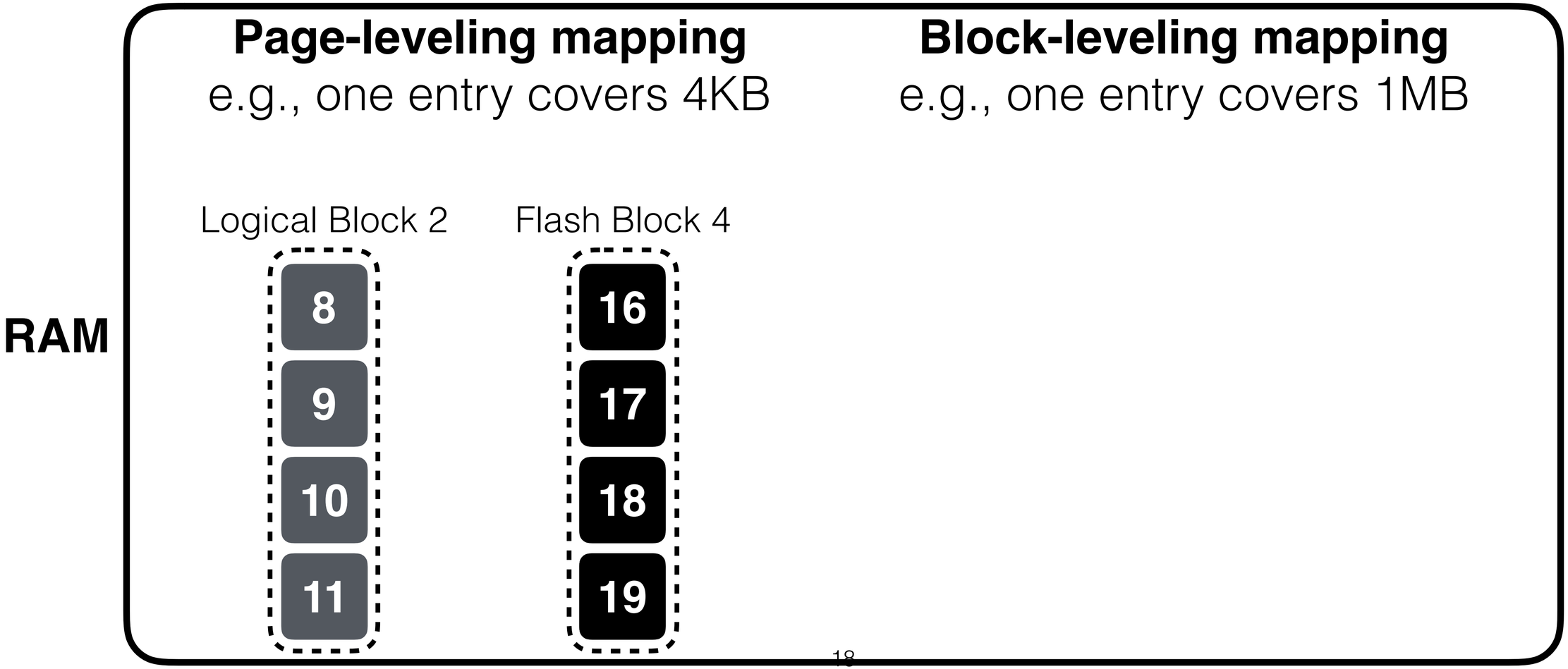
Page-leveling mapping
e.g., one entry covers 4KB

Block-leveling mapping
e.g., one entry covers 1MB



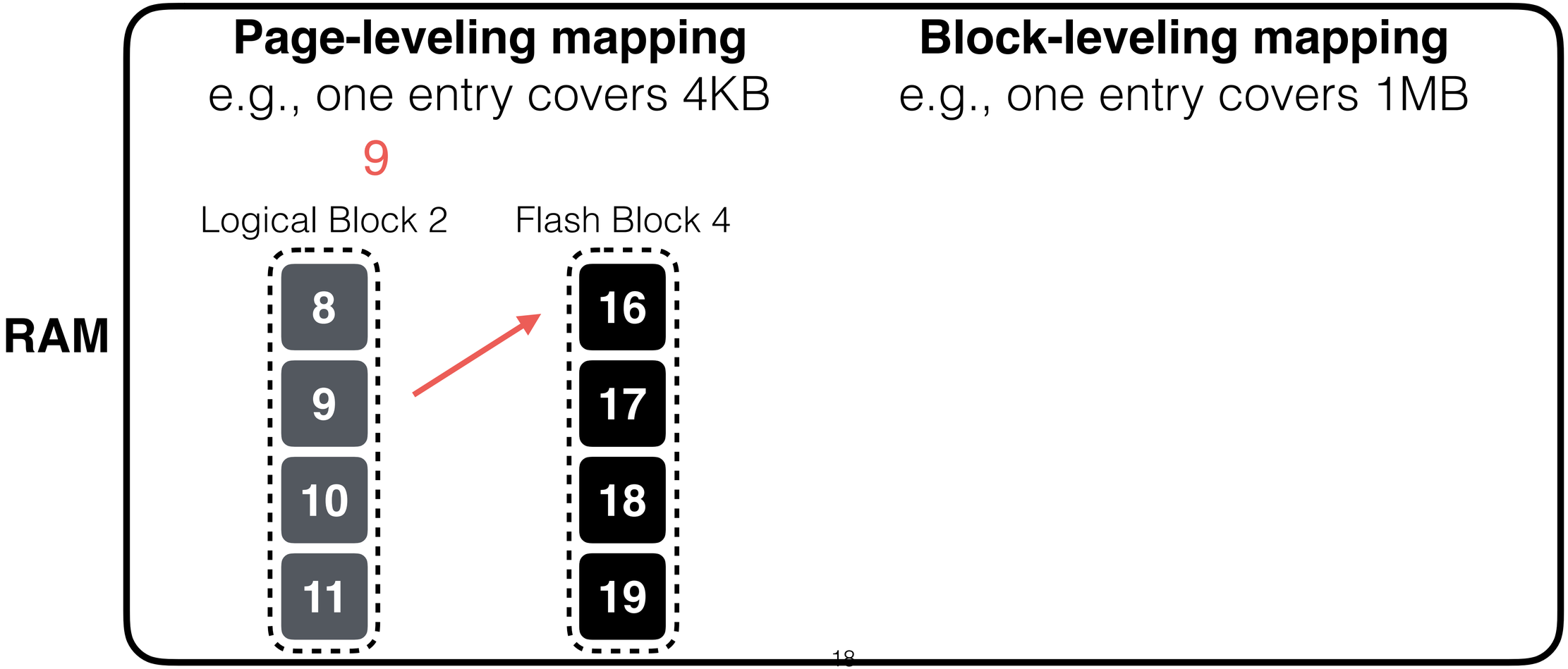
Rule 3: Aligned Sequentiality

Violation



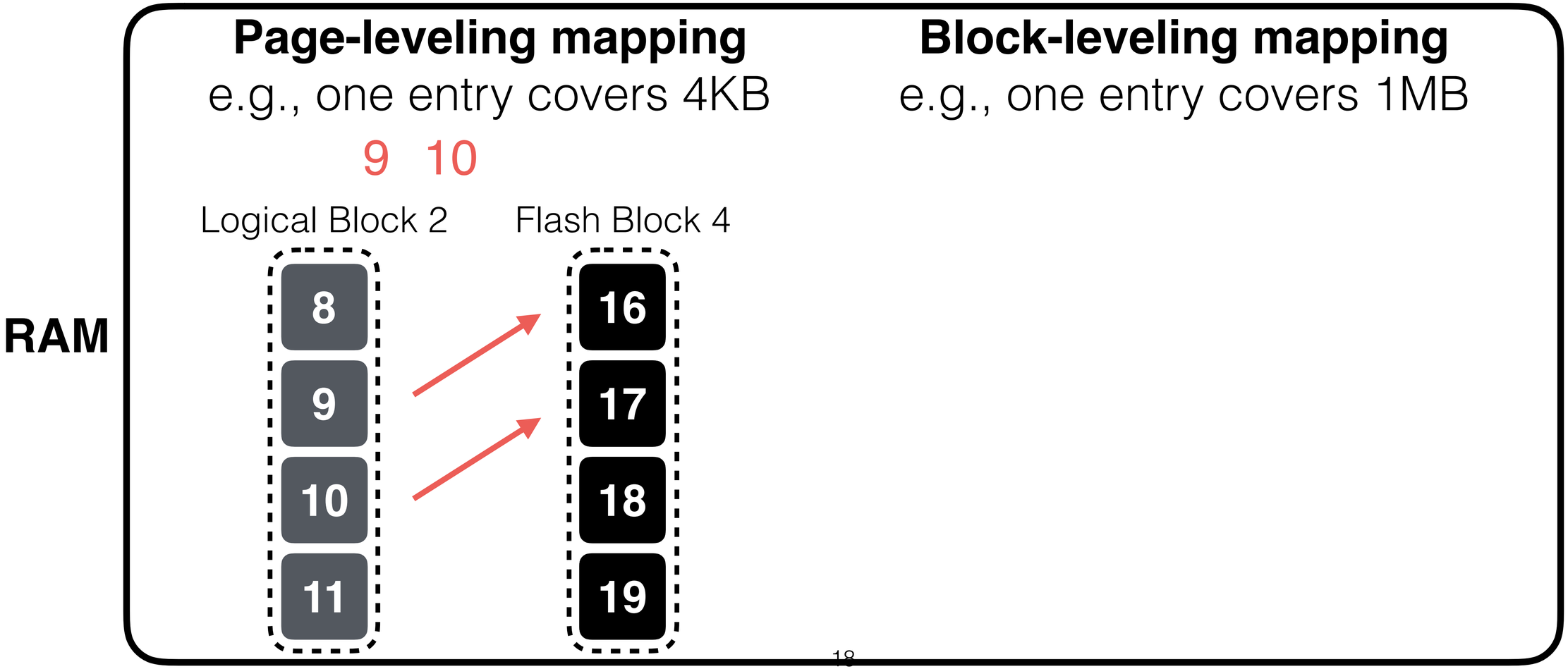
Rule 3: Aligned Sequentiality

Violation



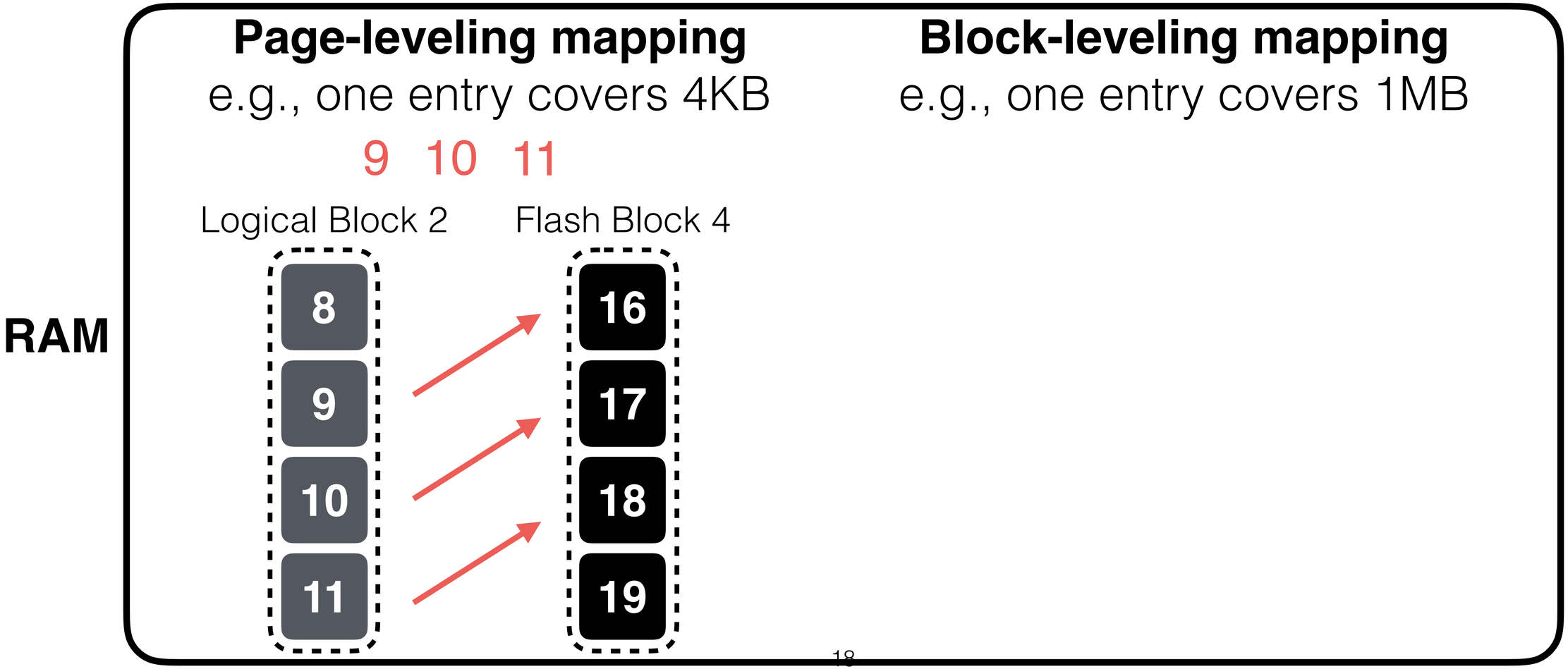
Rule 3: Aligned Sequentiality

Violation



Rule 3: Aligned Sequentiality

Violation



Rule 3: Aligned Sequentiality

Violation

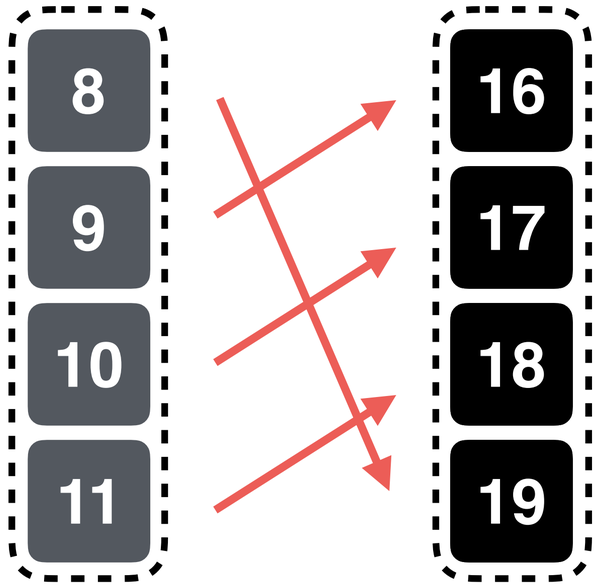
RAM

Page-leveling mapping

e.g., one entry covers 4KB

9 10 11 8

Logical Block 2 Flash Block 4

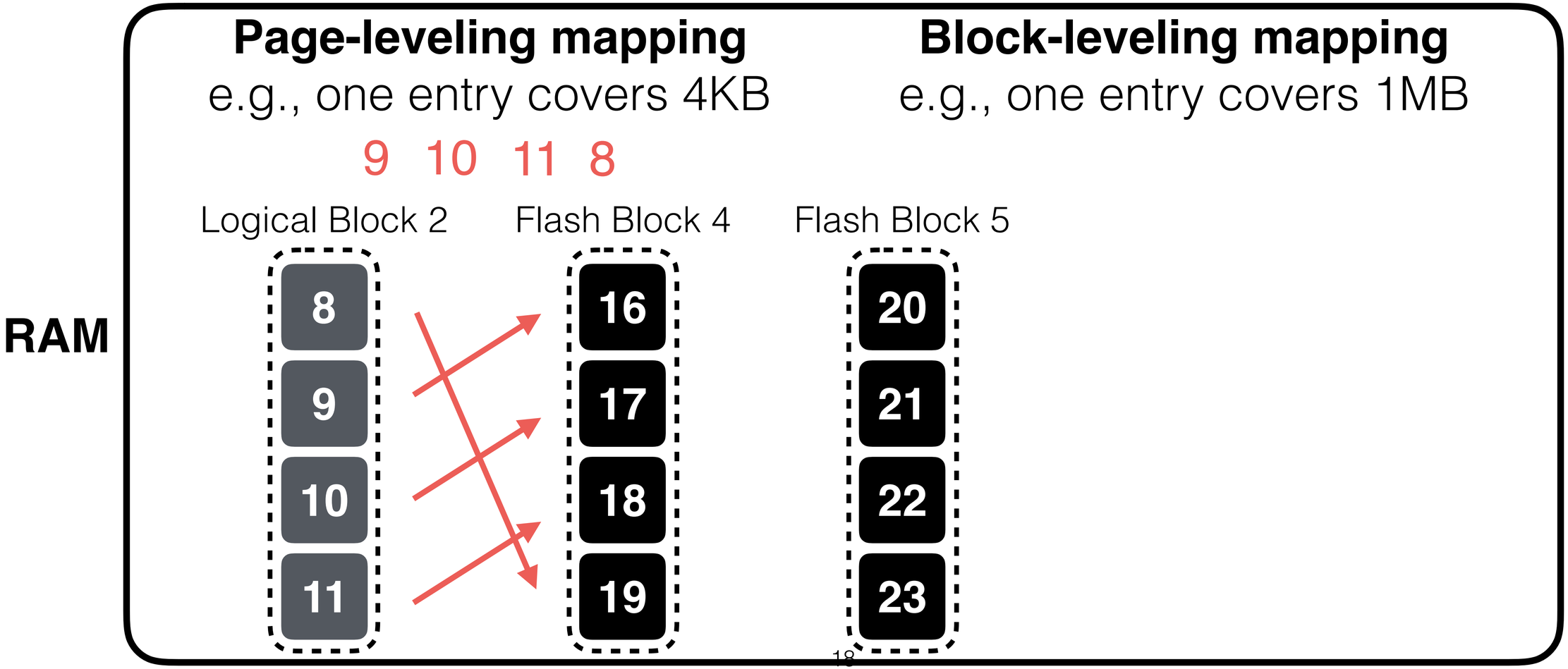


Block-leveling mapping

e.g., one entry covers 1MB

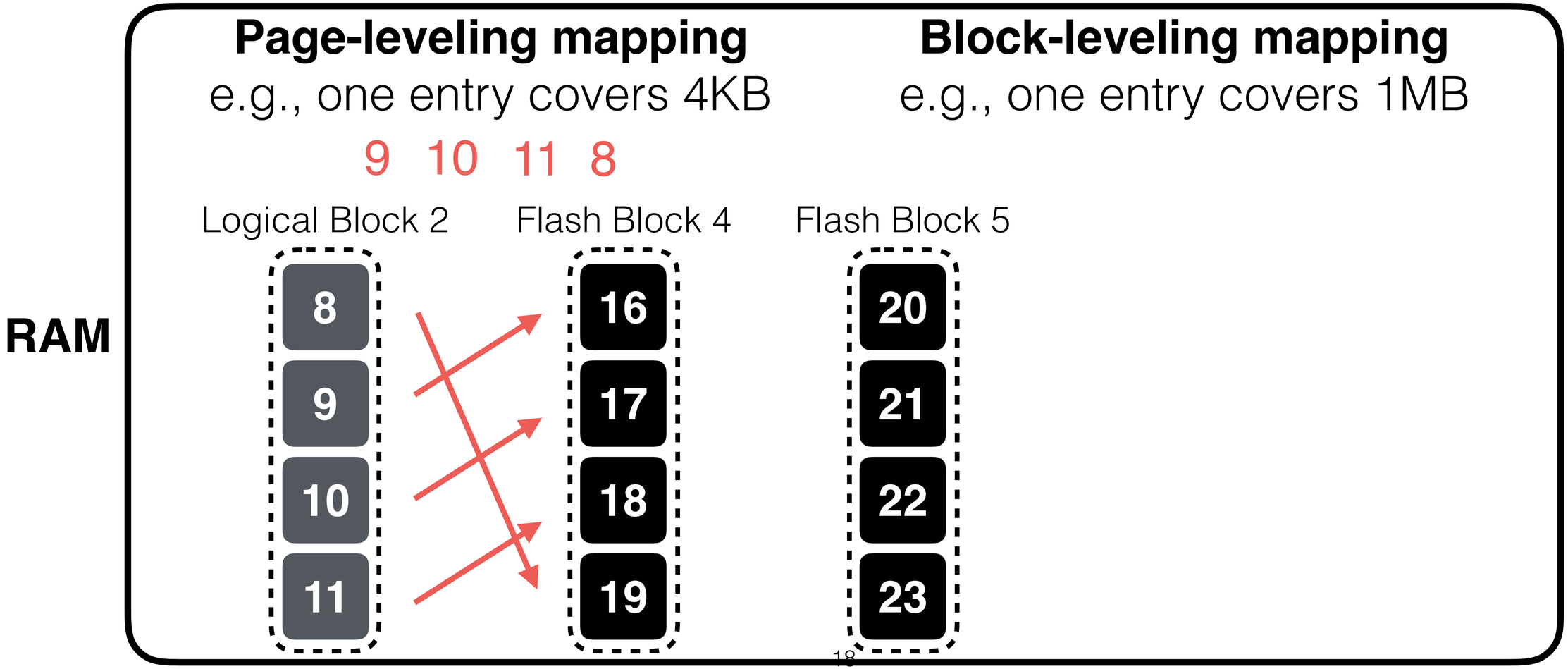
Rule 3: Aligned Sequentiality

Violation



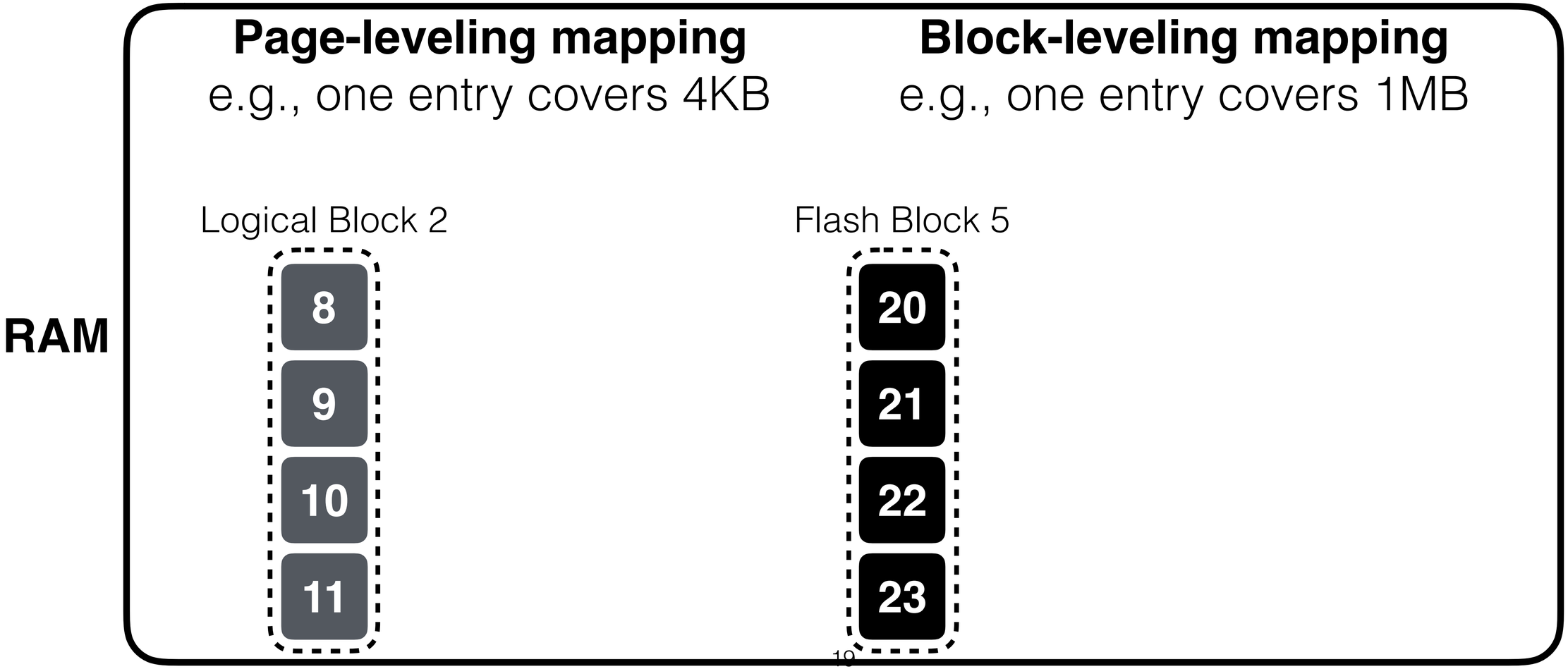
Rule 3: Aligned Sequentiality

Violation



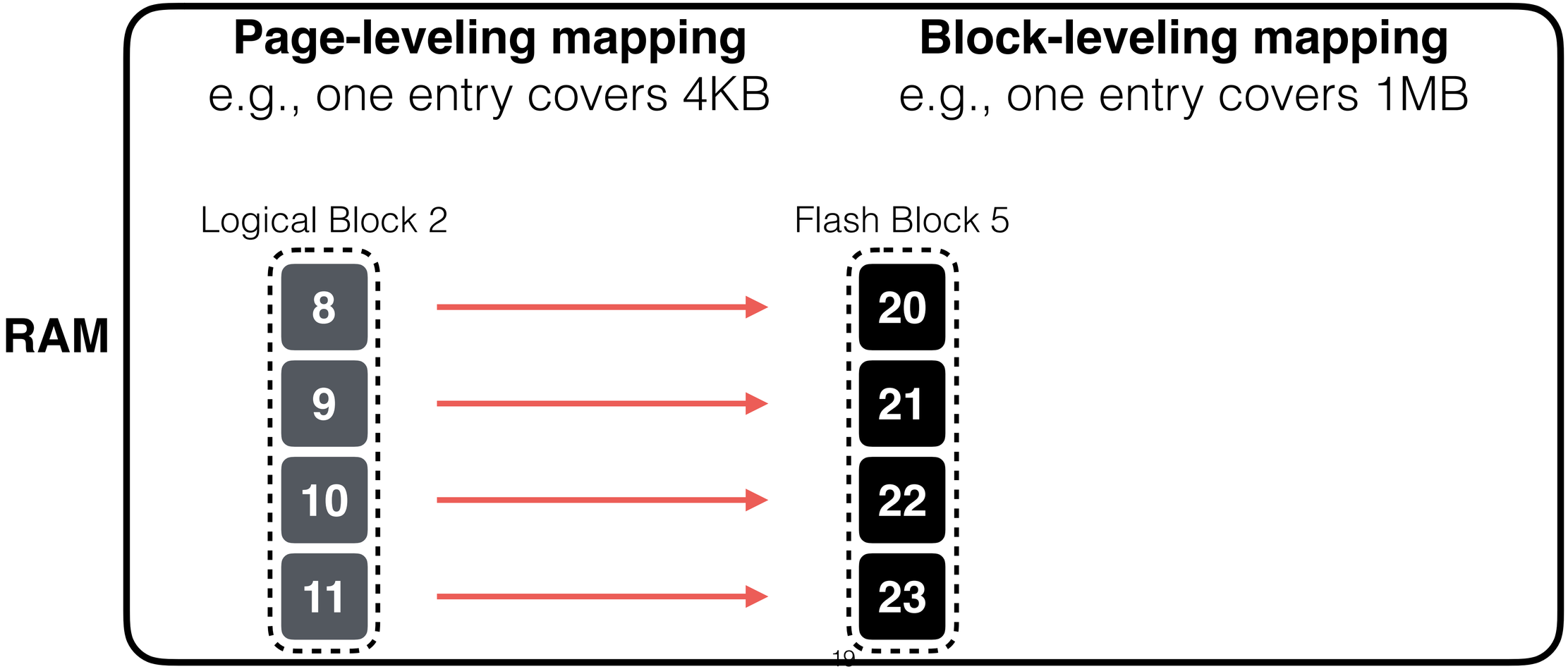
Rule 3: Aligned Sequentiality

Violation



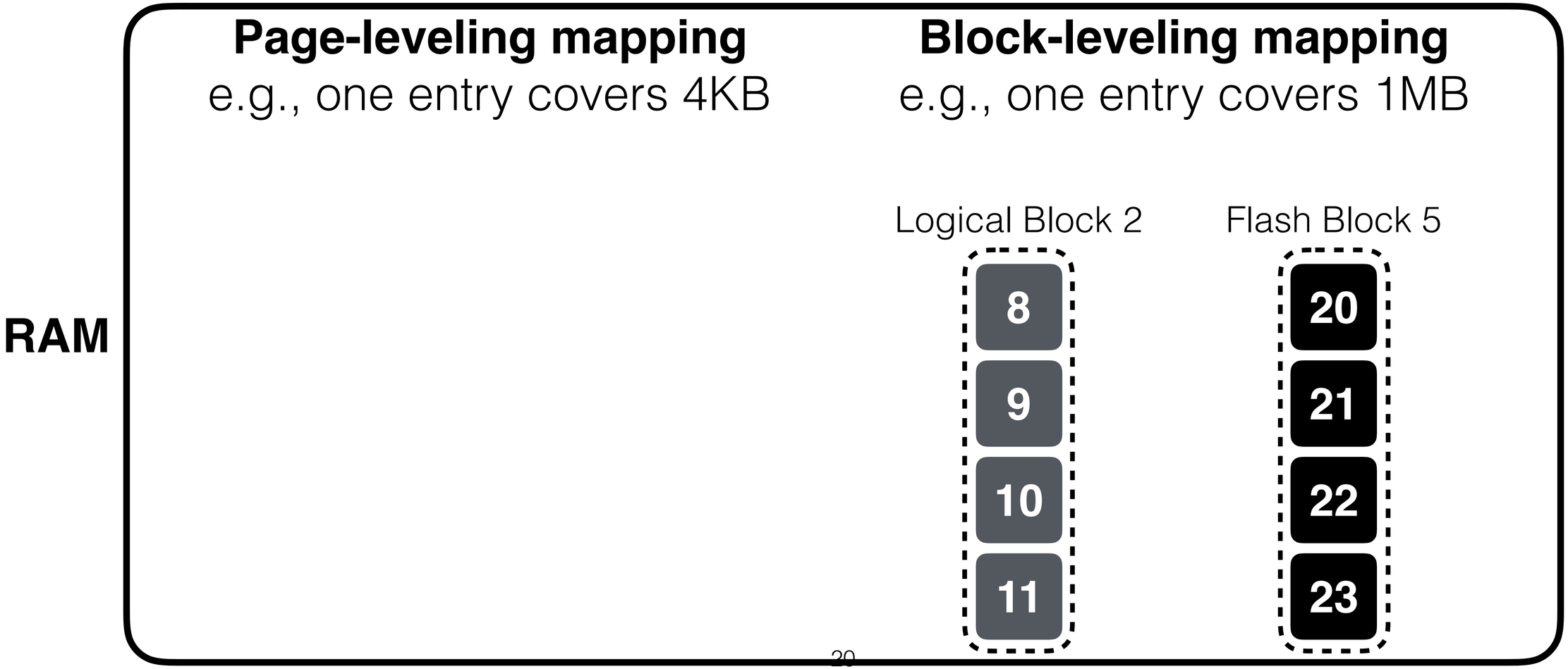
Rule 3: Aligned Sequentiality

Violation



Rule 3: Aligned Sequentiality

Violation



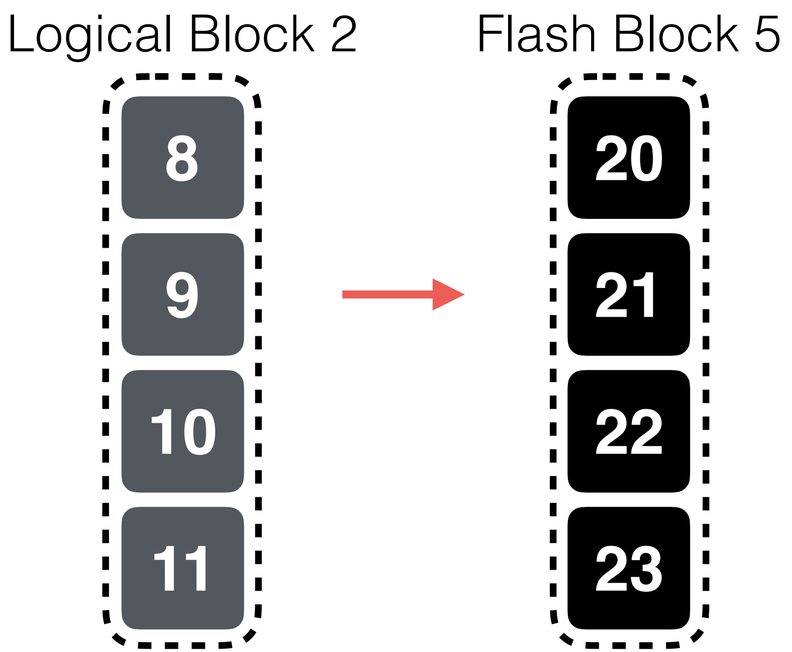
Rule 3: Aligned Sequentiality

Violation

RAM

Page-leveling mapping
e.g., one entry covers 4KB

Block-leveling mapping
e.g., one entry covers 1MB



Rule 3: Aligned Sequentiality

Violation

If you violate the rule:

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

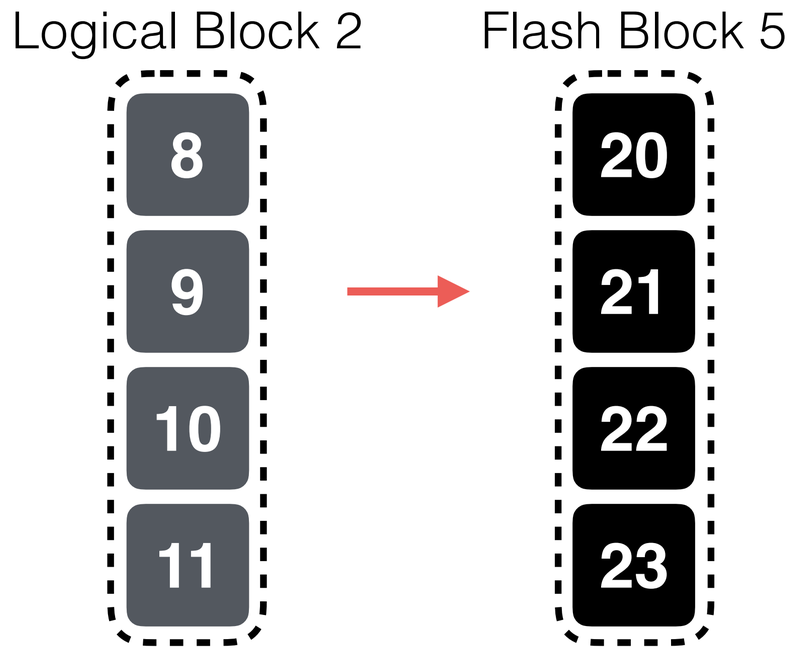
Page-l

e.g., one

apping

ers 1MB

RAM



Rule 3: Aligned Sequentiality

Violation

If you violate the rule:

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

Page-l

e.g., one

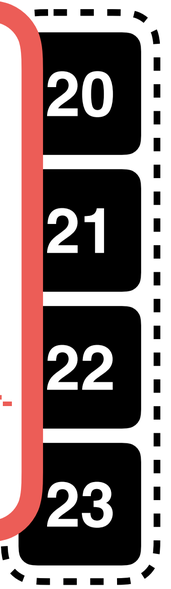
apping

ers 1MB

RAM

Logical Block 2

Flash Block 5



Performance impact:

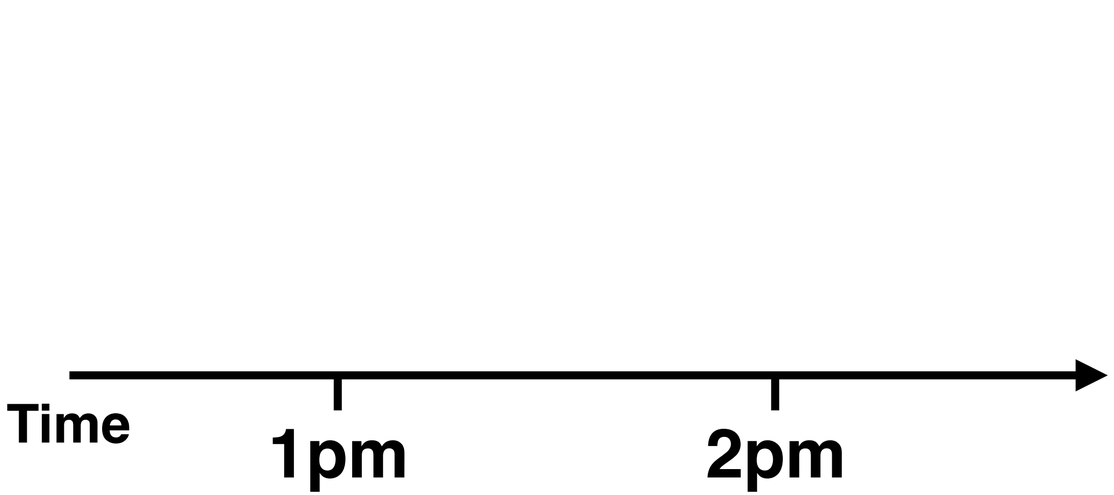
2.5x execution time

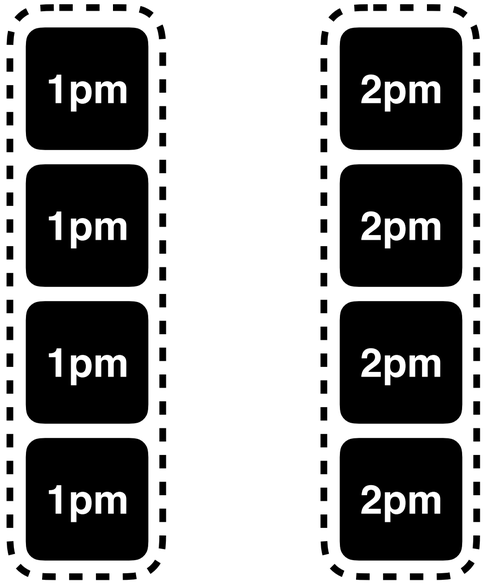
2.4x block erasure count

S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, and H.- J. Song. A Log Buffer-based Flash Translation Layer Using Fully-associative Sector Translation. ACM Trans. Embed. Comput. Syst., 6(3), July 2007.

Rule 4: Grouping By Death Time

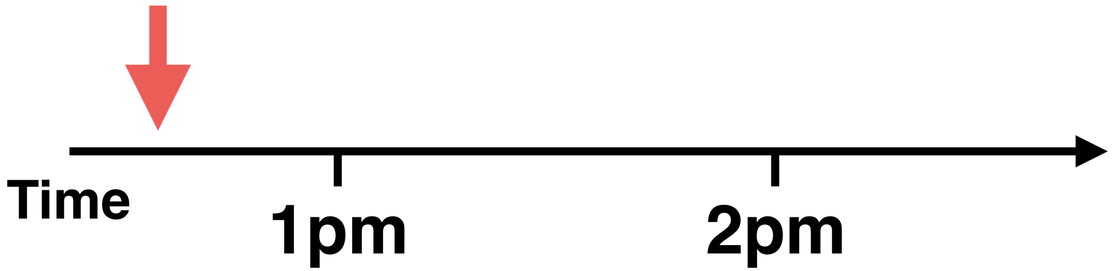
Data with similar death times should be placed in the same block.

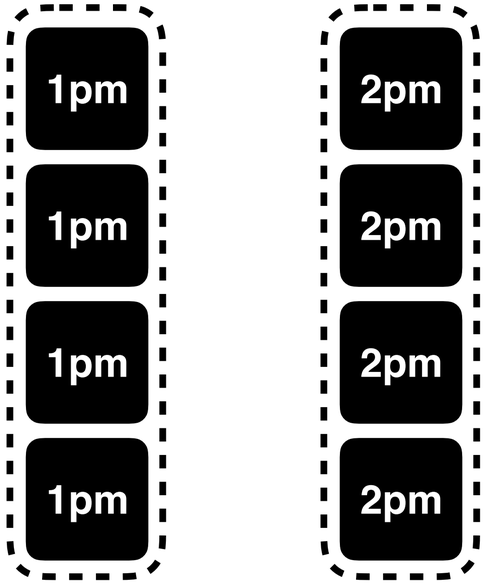




Rule 4: Grouping By Death Time

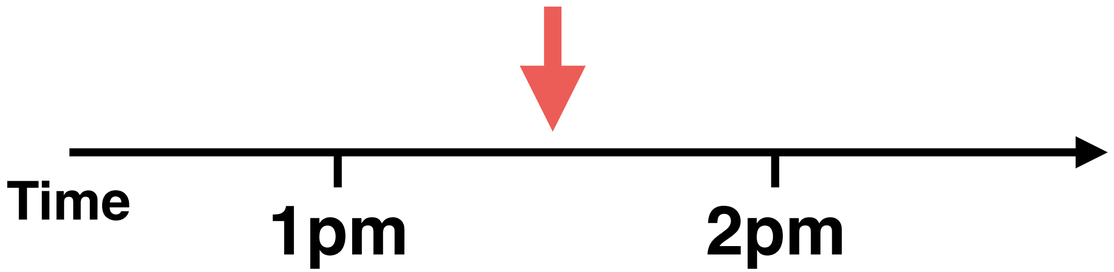
Data with similar death times should be placed in the same block.

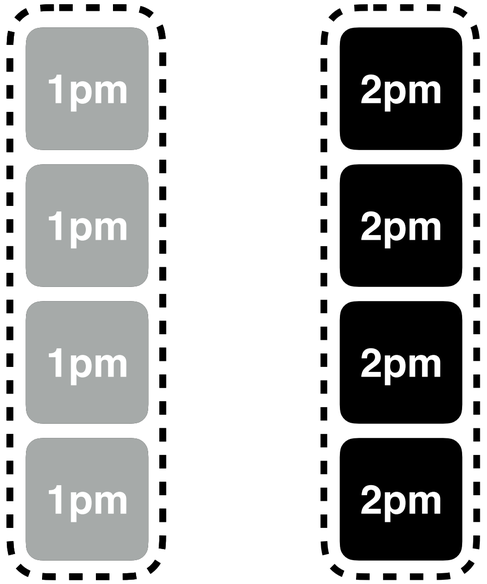




Rule 4: Grouping By Death Time

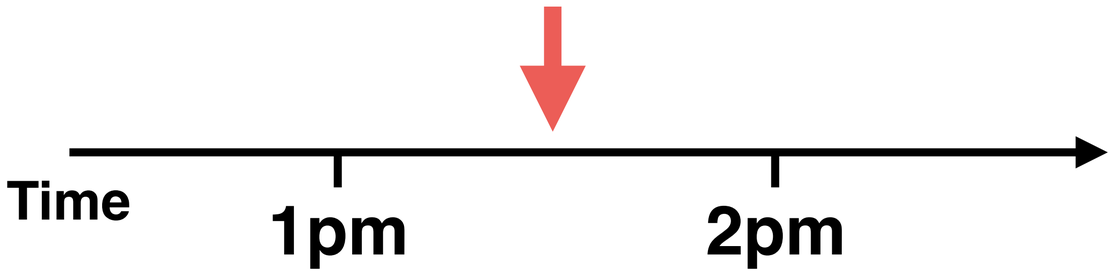
Data with similar death times should be placed in the same block.

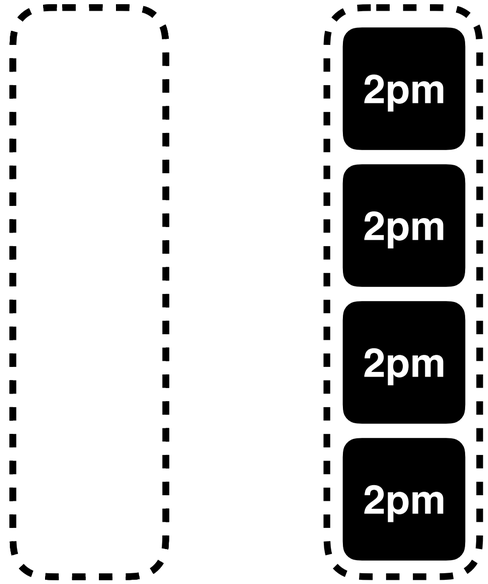




Rule 4: Grouping By Death Time

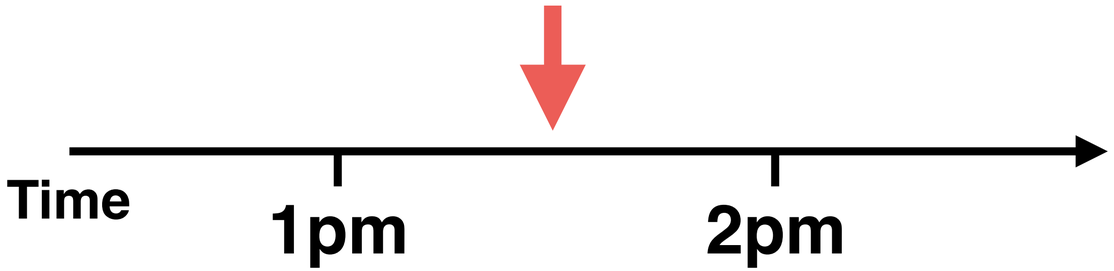
Data with similar death times should be placed in the same block.

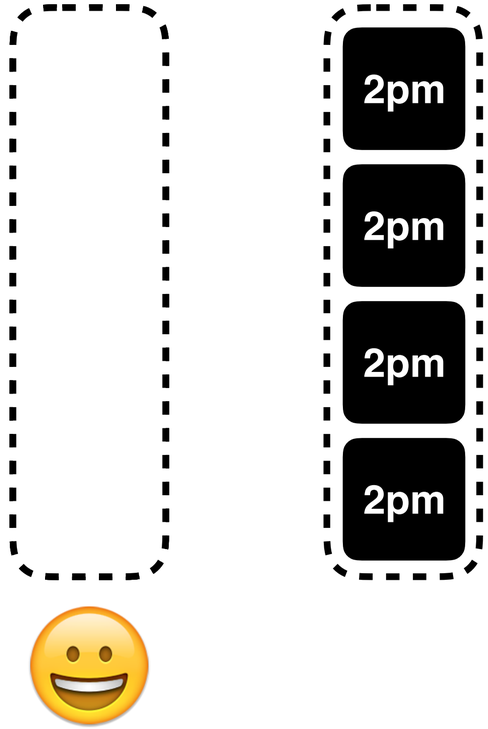




Rule 4: Grouping By Death Time

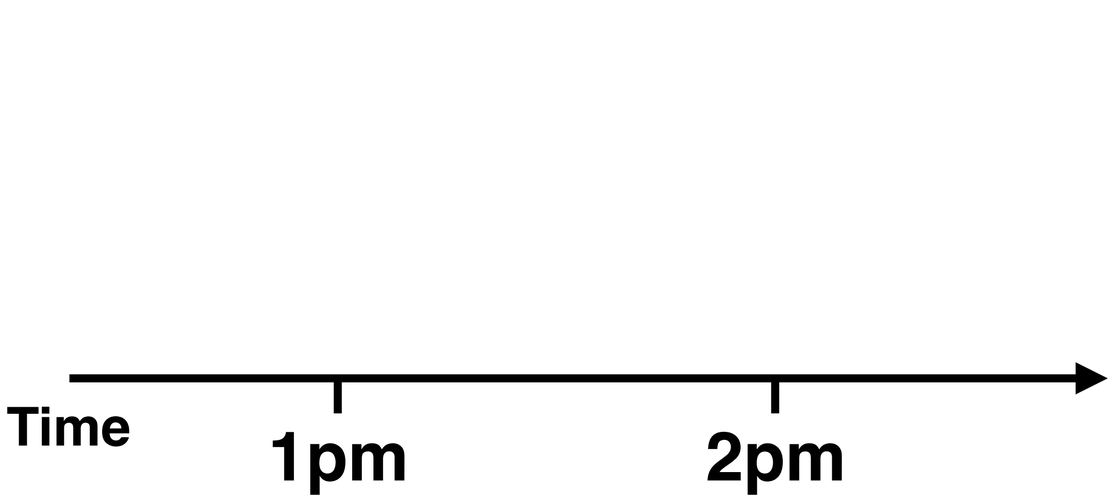
Data with similar death times should be placed in the same block.

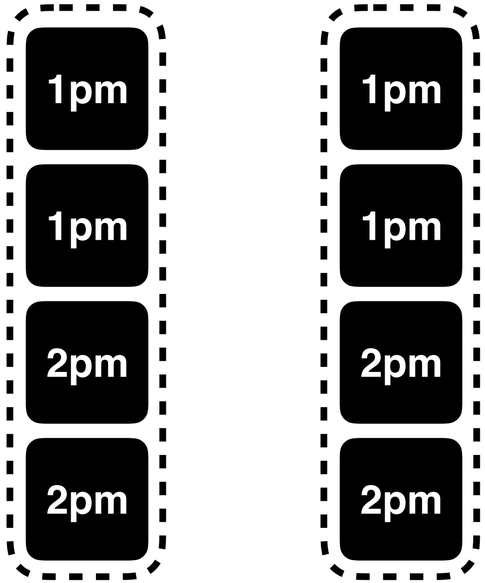




Rule 4: Grouping By Death Time

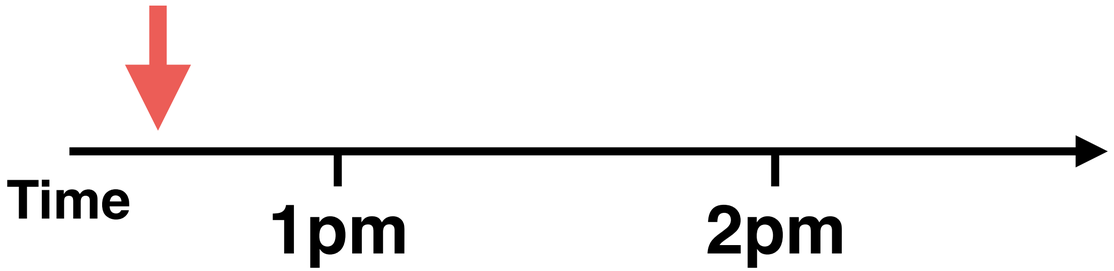
Violation

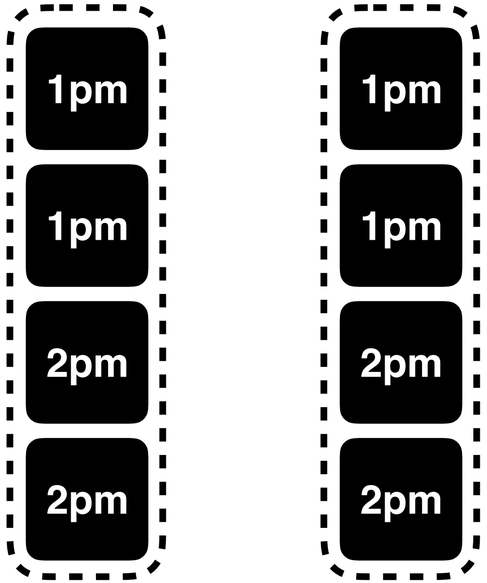




Rule 4: Grouping By Death Time

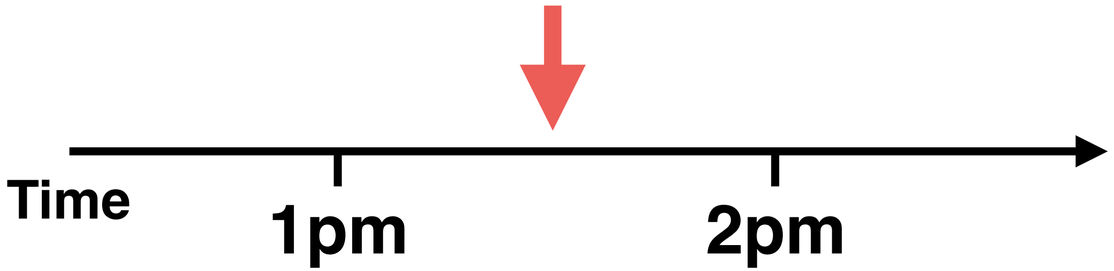
Violation

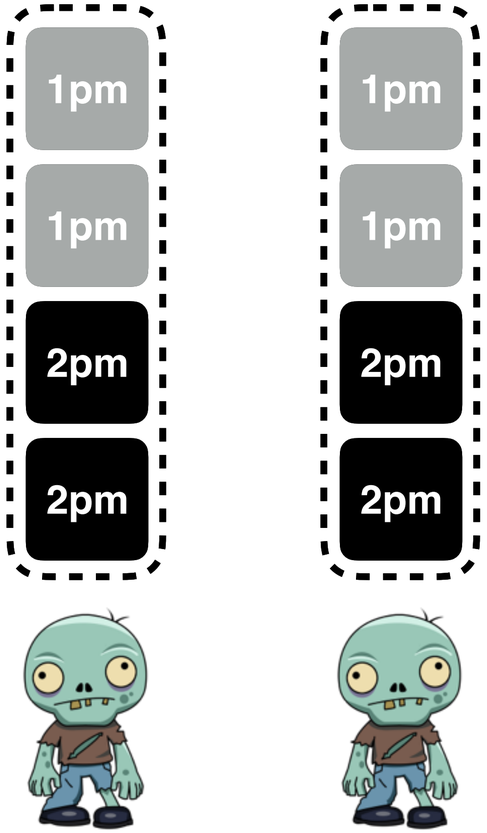




Rule 4: Grouping By Death Time

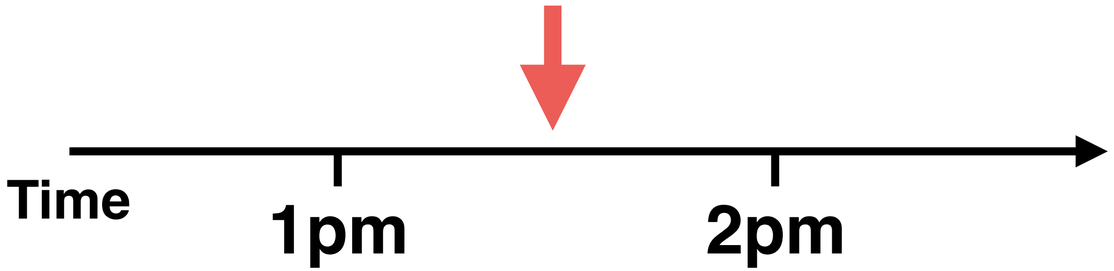
Violation

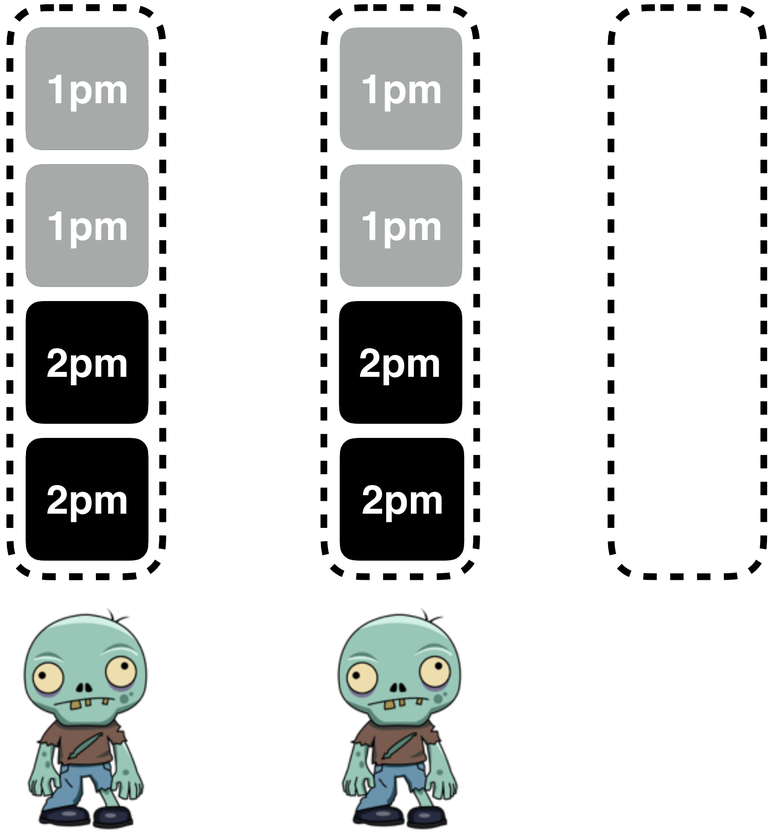




Rule 4: Grouping By Death Time

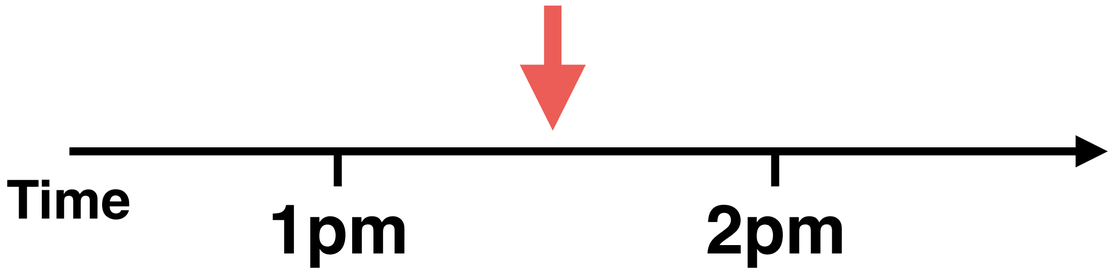
Violation

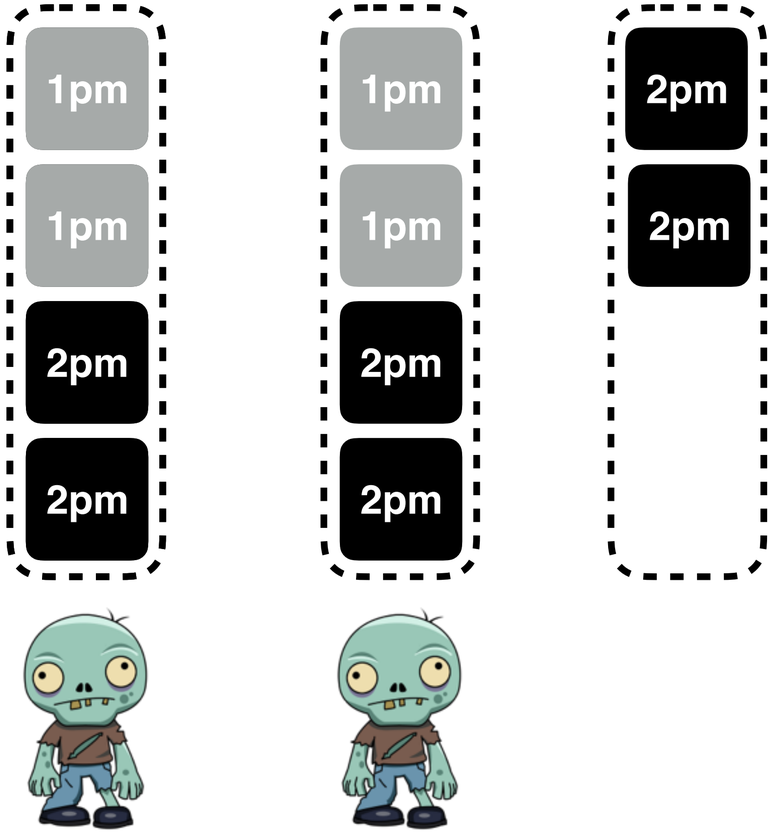




Rule 4: Grouping By Death Time

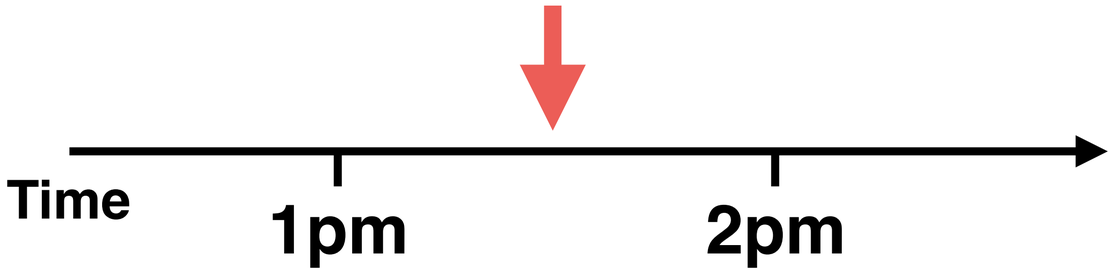
Violation

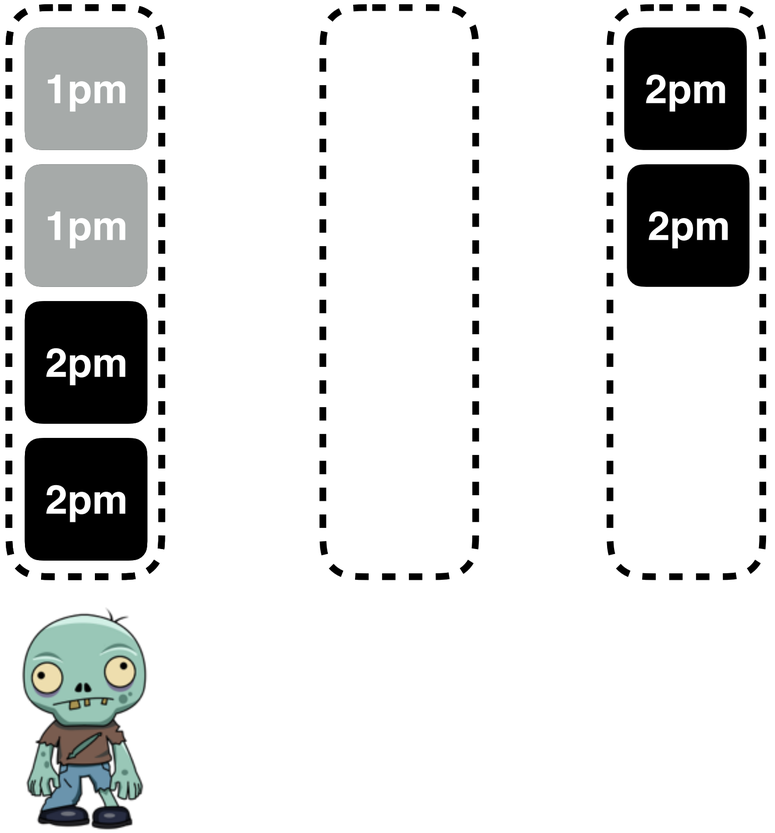




Rule 4: Grouping By Death Time

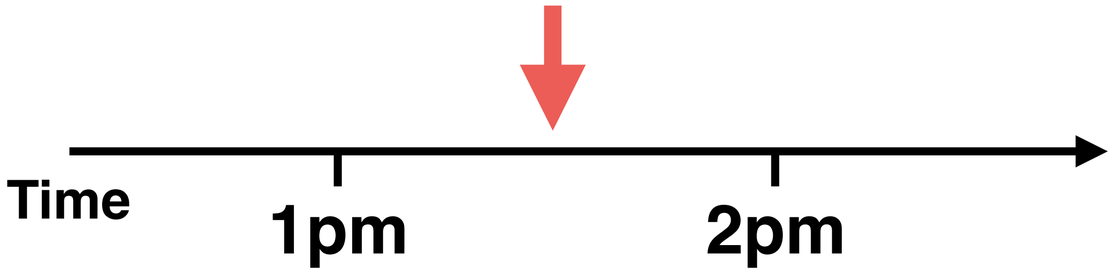
Violation



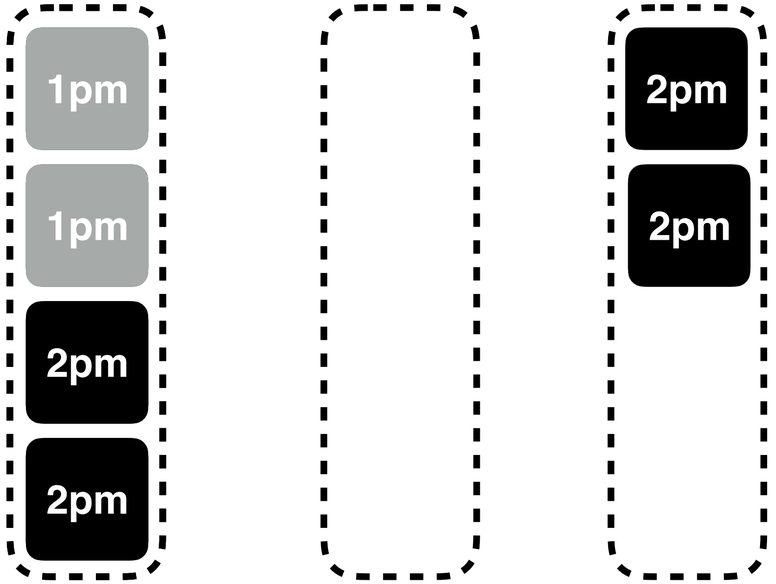


Rule 4: Grouping By Death Time

Violation

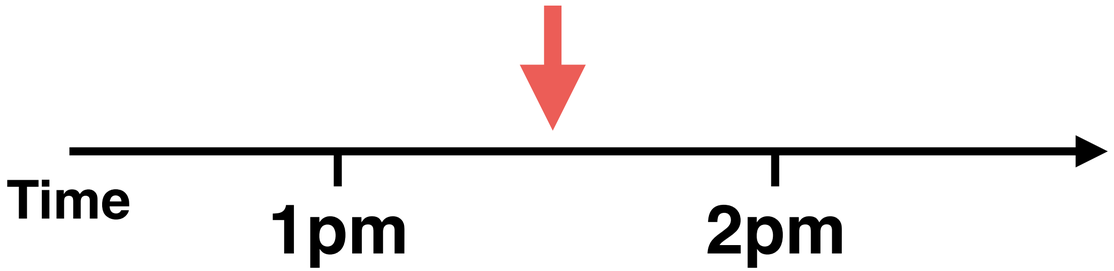


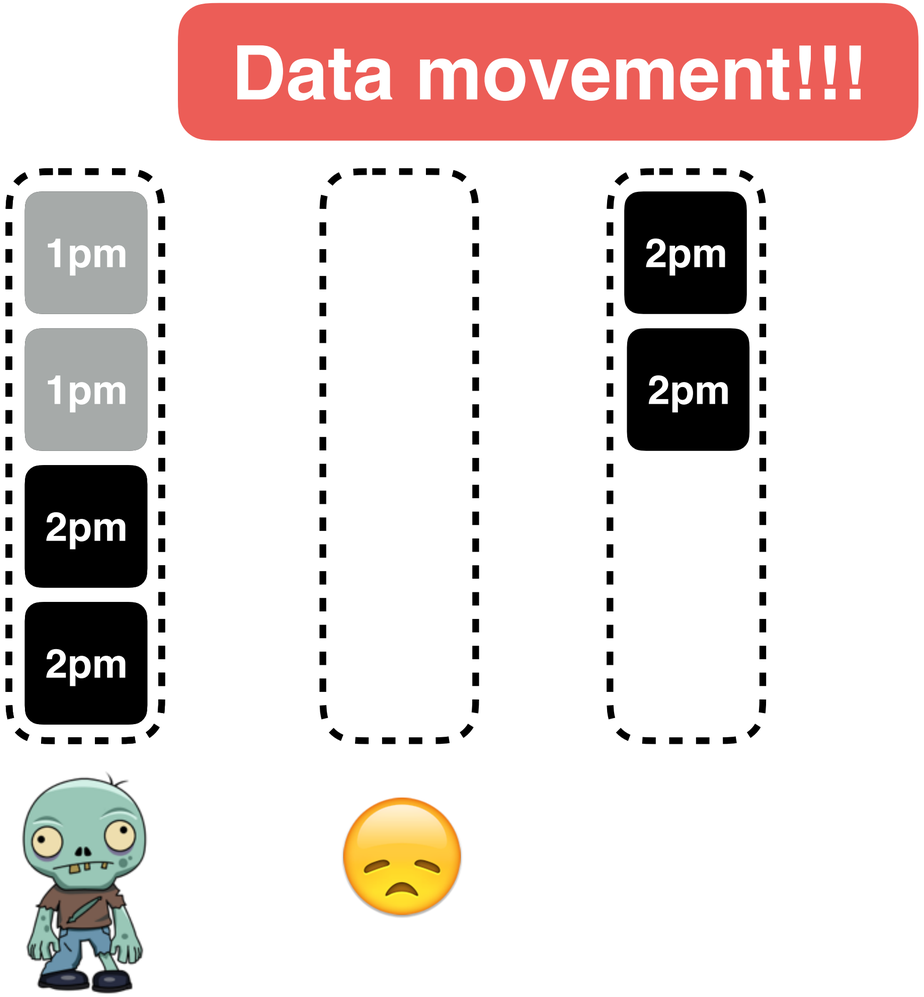
Data movement!!!



Rule 4: Grouping By Death Time

Violation



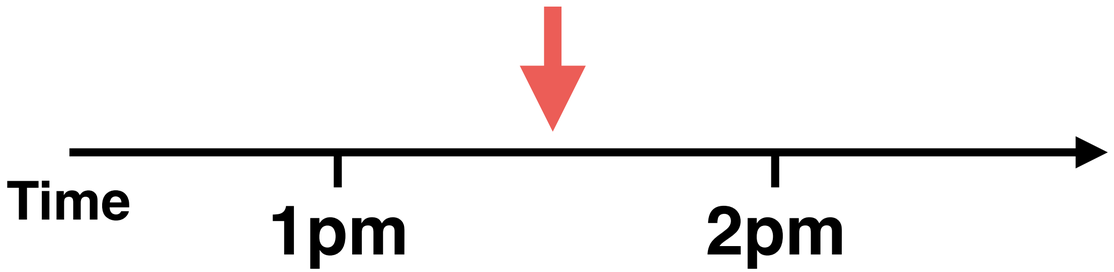


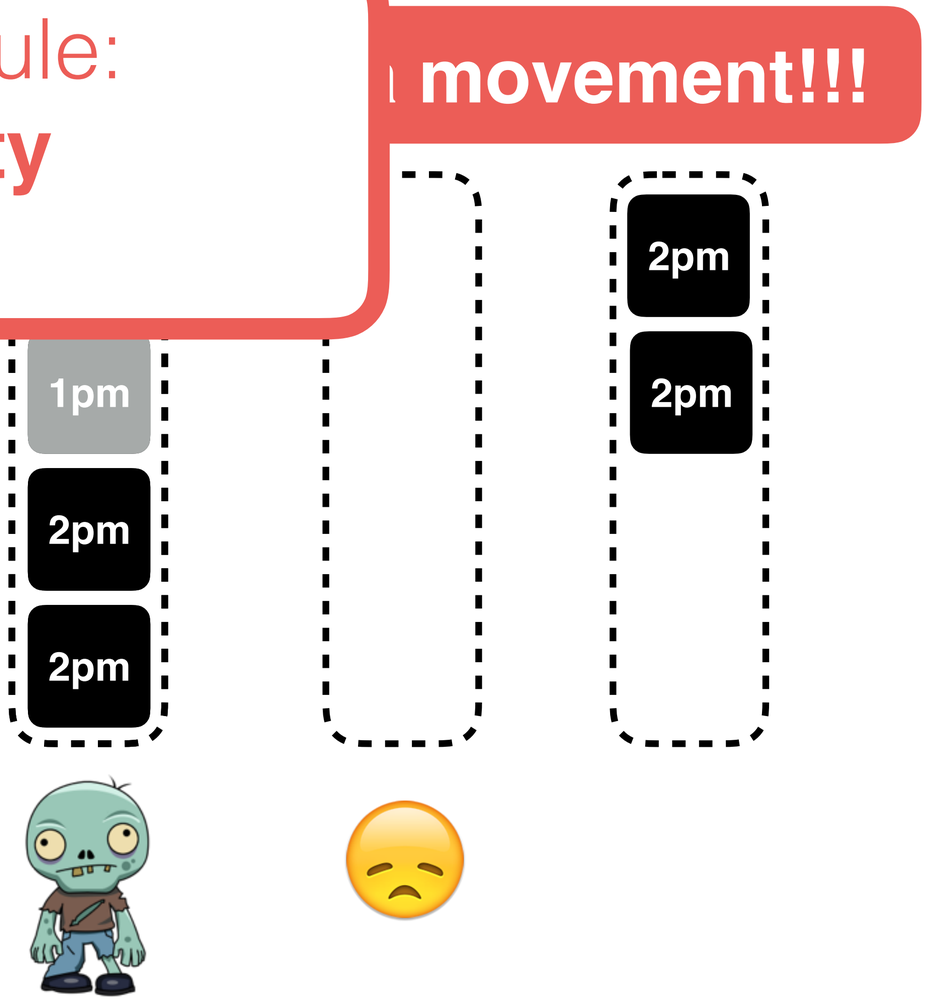
Rule 4: Grouping By Death Time

Violation

If you violate the rule:

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





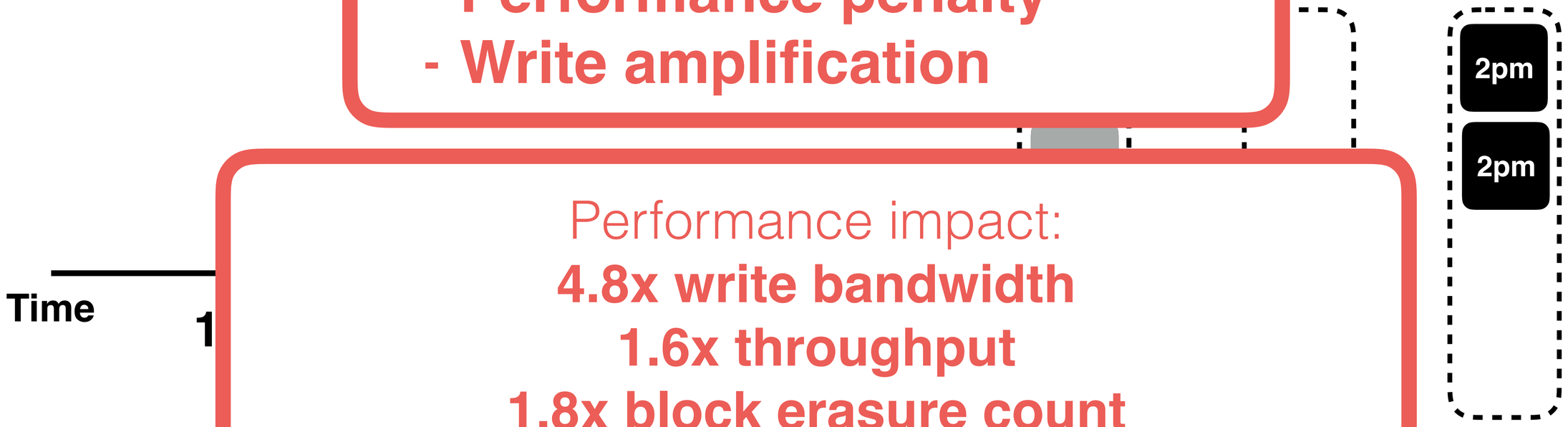
Rule 4: Grouping By Death Time

Violation

If you violate the rule:

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

movement!!!



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.

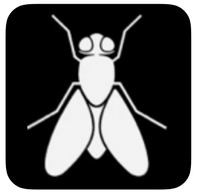
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime

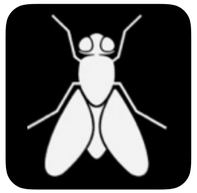


1 Day

Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

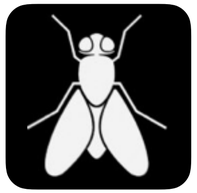
SSD

Program Count: **0 0 0 0**

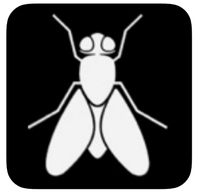
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day



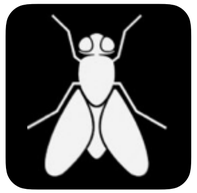
SSD

Program Count: **0 0 0 0**

Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

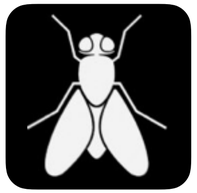
SSD

Program Count: **0 0 0 0**

Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

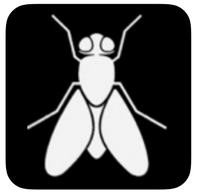
SSD

Program Count: **0** **0** **0**

Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

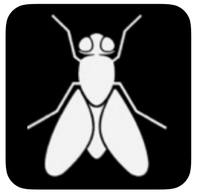
SSD

Program Count: **1 0 0 0**

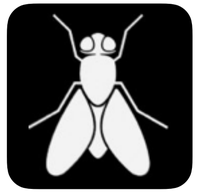
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day



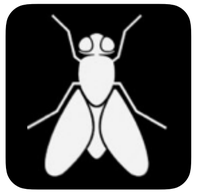
SSD

Program Count: **1 0 0 0**

Rule 5: Uniform Data Lifetime

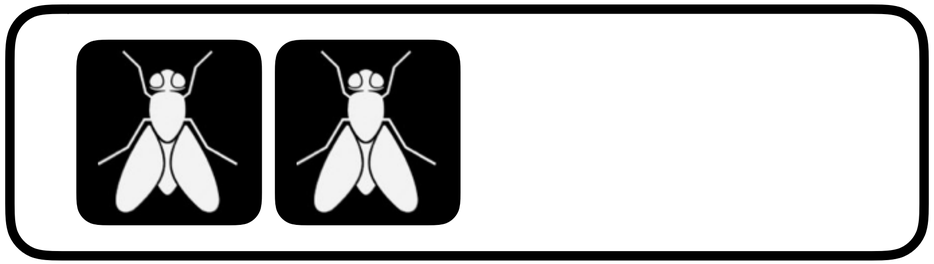
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

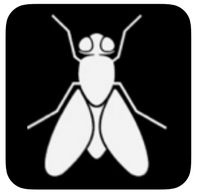


Program Count: **1 0 0 0**

Rule 5: Uniform Data Lifetime

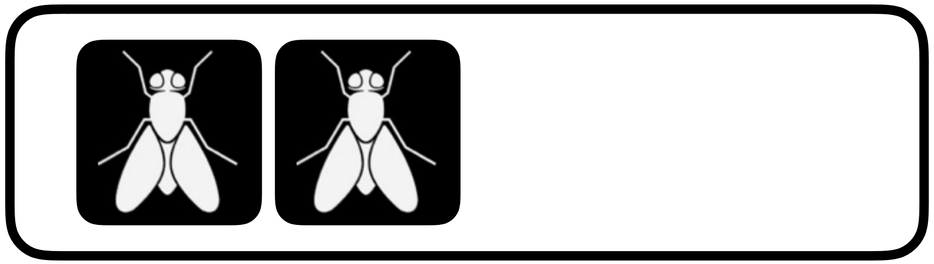
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

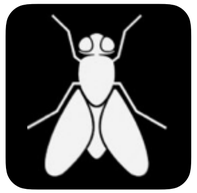


Program Count: **1** **0** **0**

Rule 5: Uniform Data Lifetime

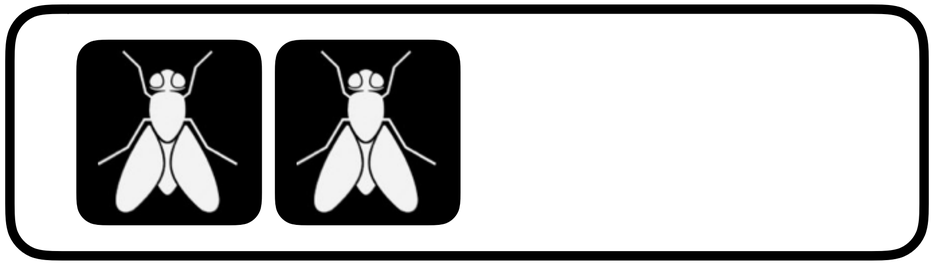
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

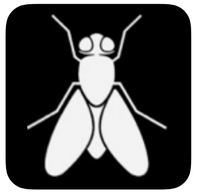


Program Count: **1 1 0 0**

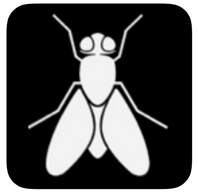
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

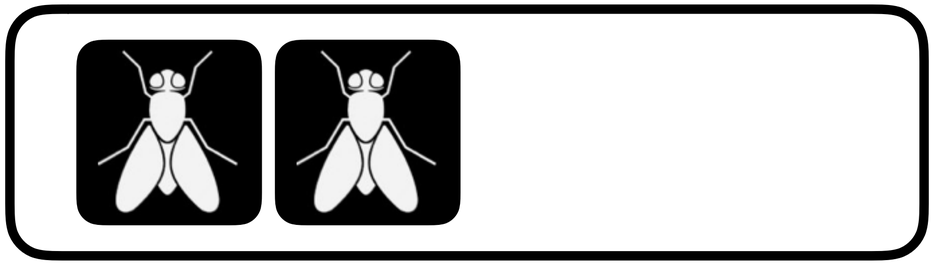
Lifetime



1 Day



SSD

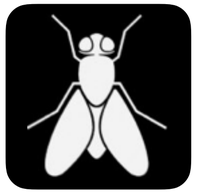


Program Count: **1 1 0 0**

Rule 5: Uniform Data Lifetime

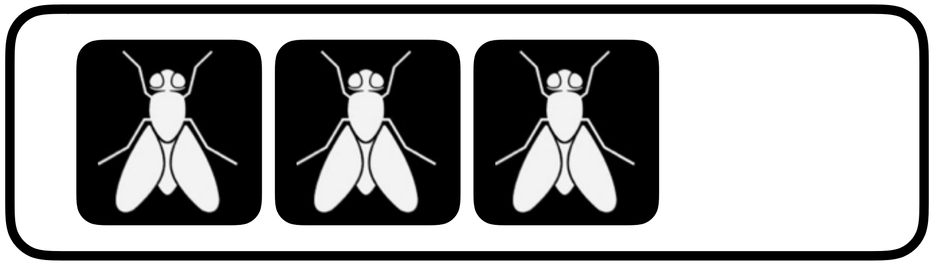
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

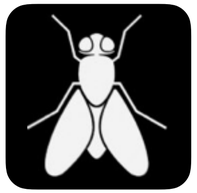


Program Count: **1 1 0 0**

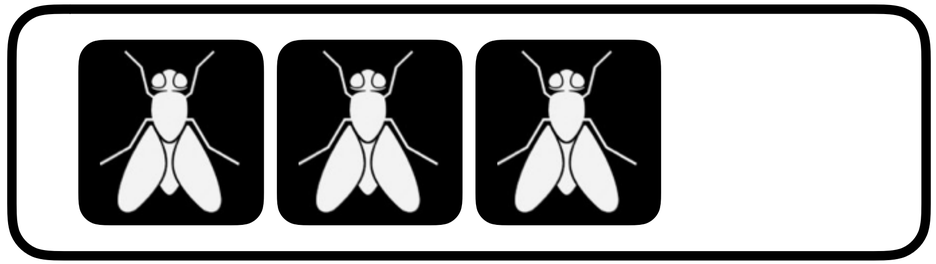
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



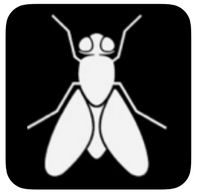
1 Day



Rule 5: Uniform Data Lifetime

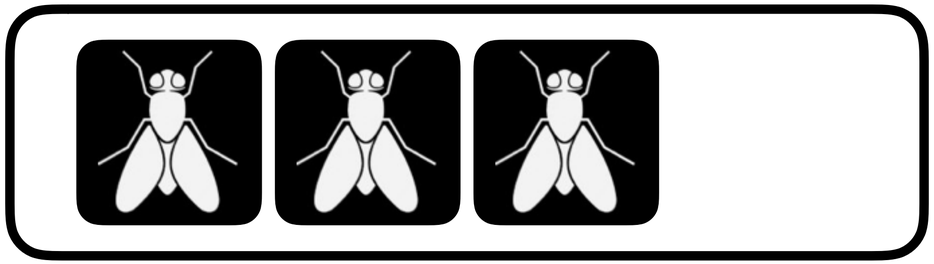
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

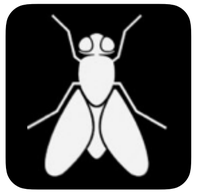


Program Count: **1 1 1 0**

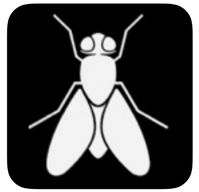
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

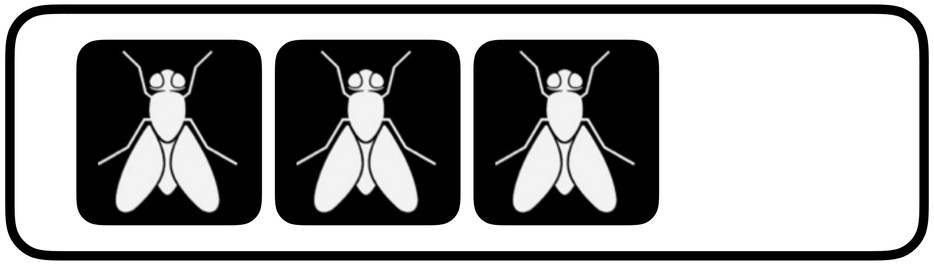
Lifetime



1 Day



SSD

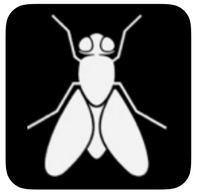


Program Count: **1 1 1 0**

Rule 5: Uniform Data Lifetime

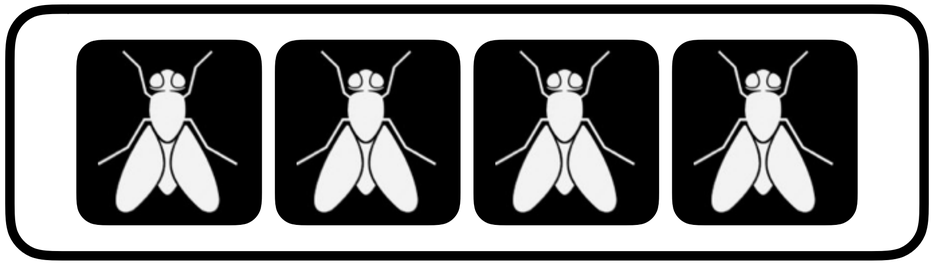
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

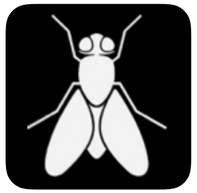


Program Count: **1 1 1 0**

Rule 5: Uniform Data Lifetime

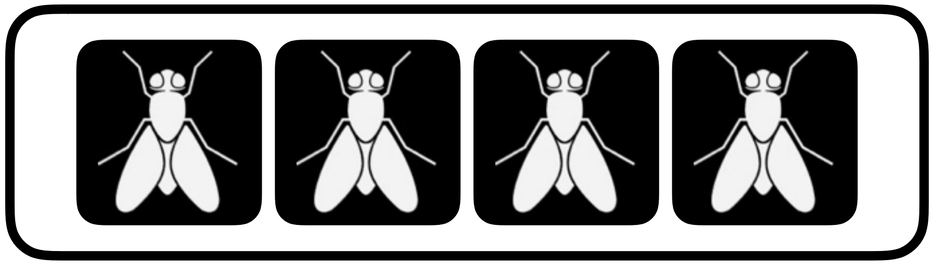
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

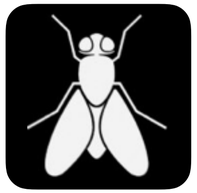


Program Count: **1** **1** **1**

Rule 5: Uniform Data Lifetime

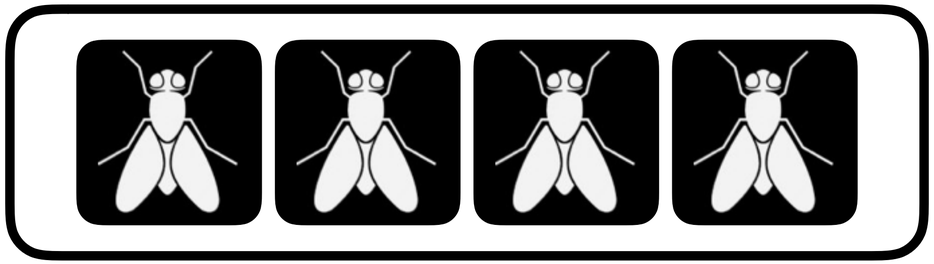
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

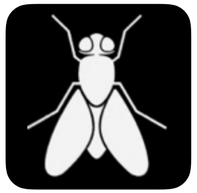


Program Count: **1 1 1 1**

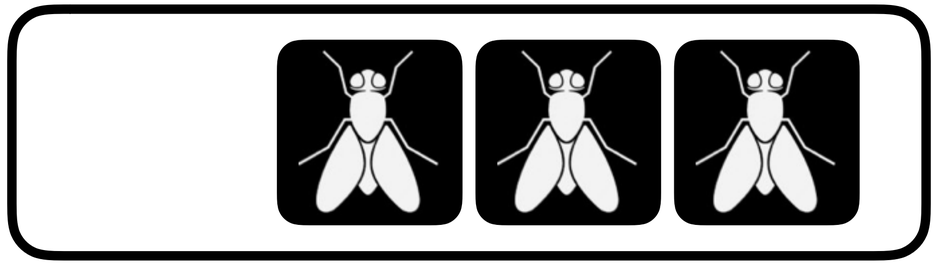
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



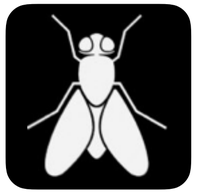
1 Day



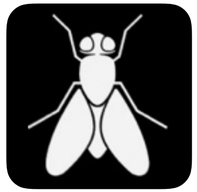
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

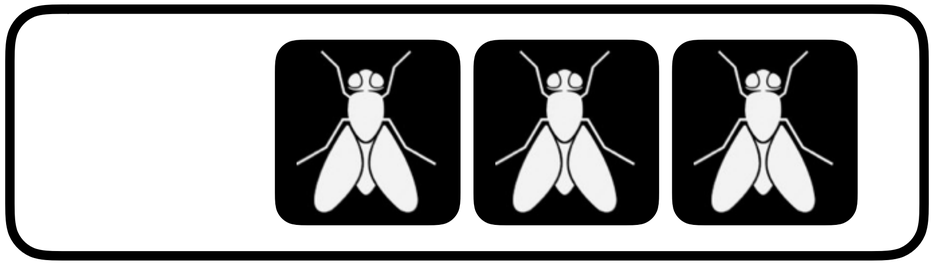
Lifetime



1 Day



SSD

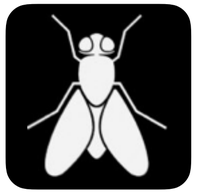


Program Count: 1 1 1 1

Rule 5: Uniform Data Lifetime

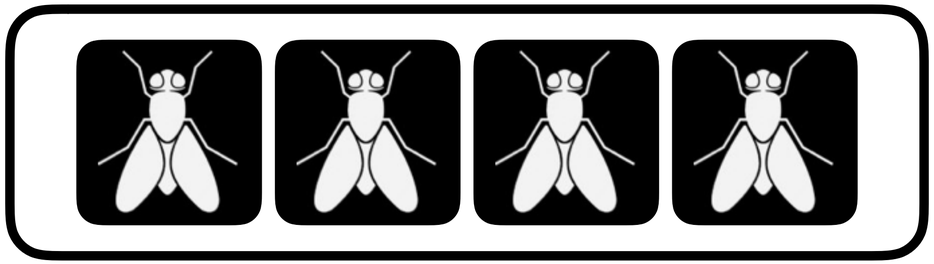
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

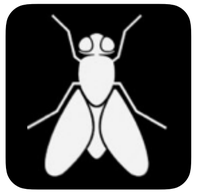


Program Count: **1 1 1 1**

Rule 5: Uniform Data Lifetime

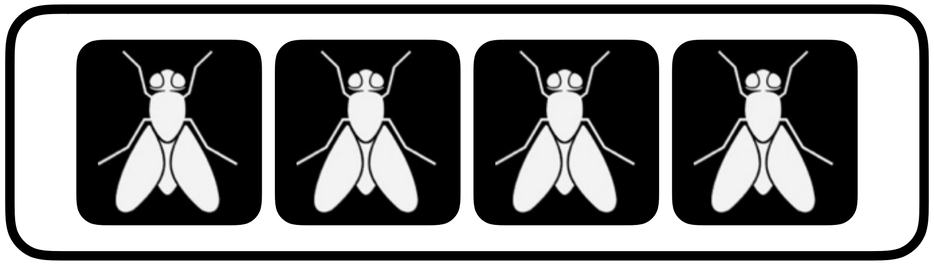
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

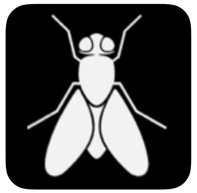


Program Count: **1 1 1**

Rule 5: Uniform Data Lifetime

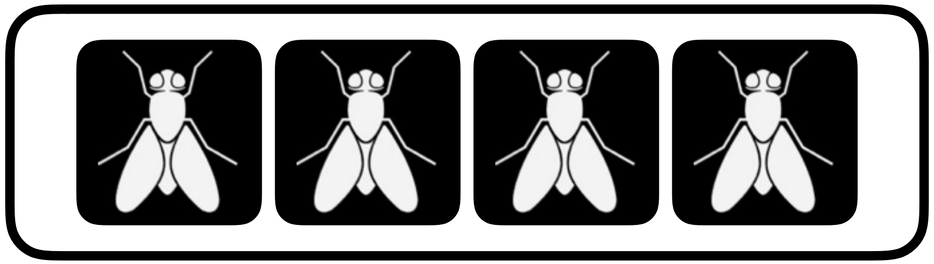
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

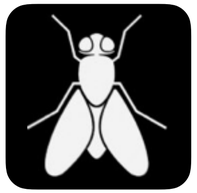


Program Count: **2 1 1 1**

Rule 5: Uniform Data Lifetime

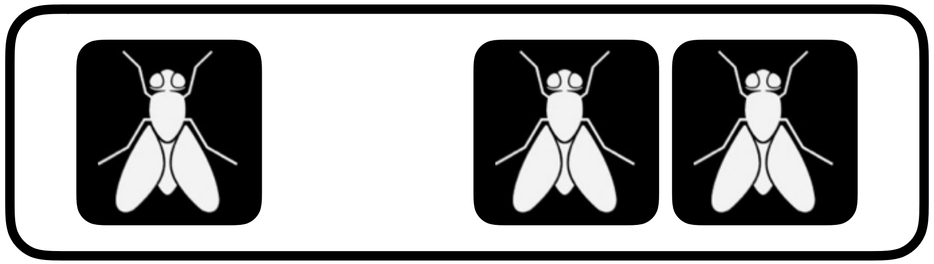
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

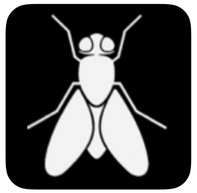


Program Count: **2** **1** **1** **1**

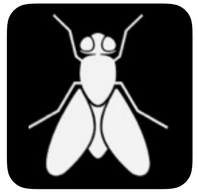
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

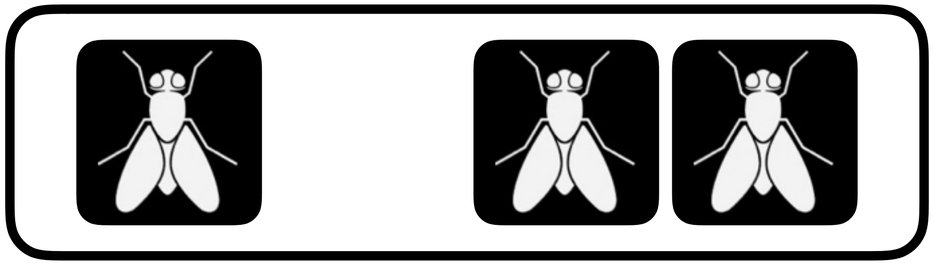
Lifetime



1 Day



SSD

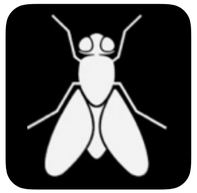


Program Count: **2** **1** **1** **1**

Rule 5: Uniform Data Lifetime

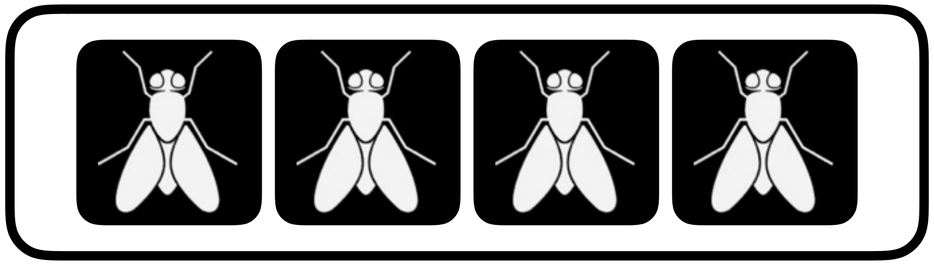
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

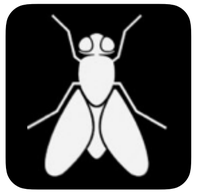


Program Count: **2** **1** **1** **1**

Rule 5: Uniform Data Lifetime

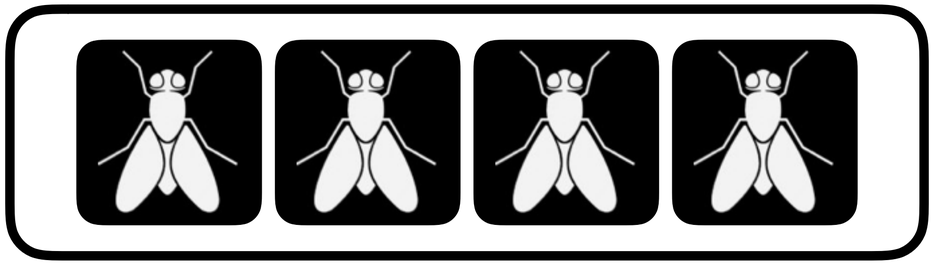
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

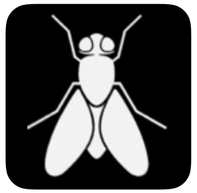


Program Count: **2** **1** **1**

Rule 5: Uniform Data Lifetime

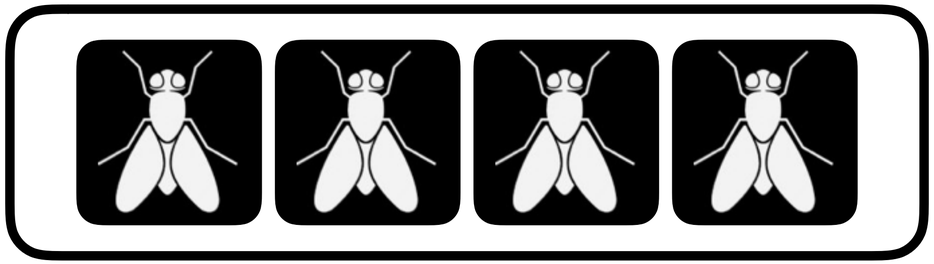
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

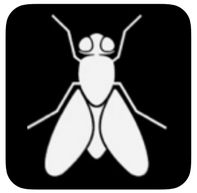


Program Count: **2 2 1 1**

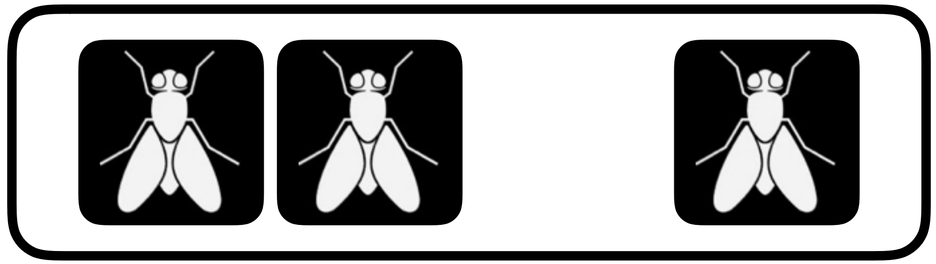
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



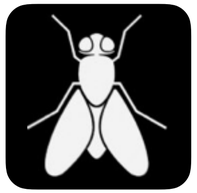
1 Day



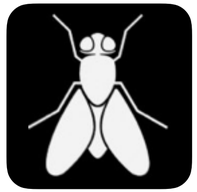
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

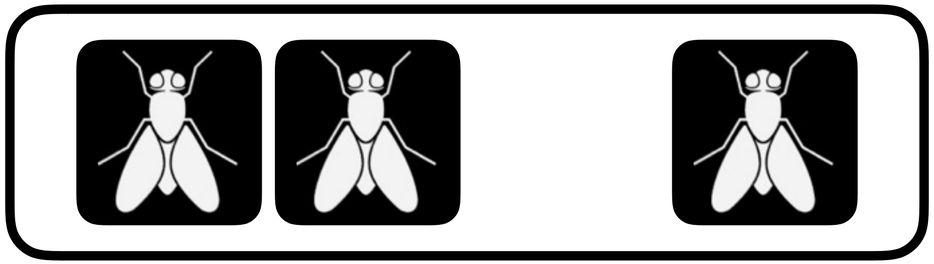
Lifetime



1 Day



SSD

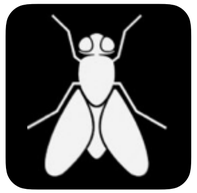


Program Count: **2 2 1 1**

Rule 5: Uniform Data Lifetime

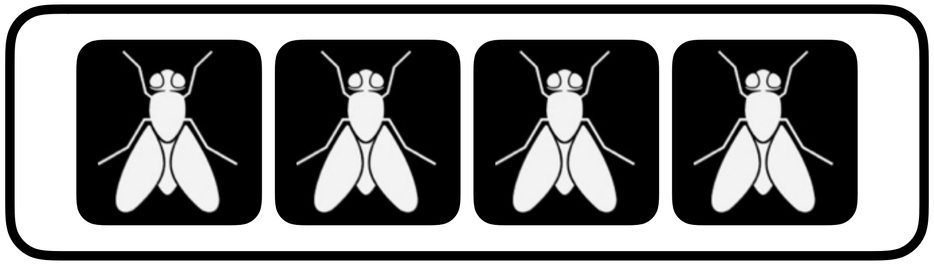
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

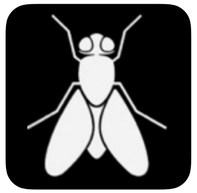


Program Count: **2 2 1 1**

Rule 5: Uniform Data Lifetime

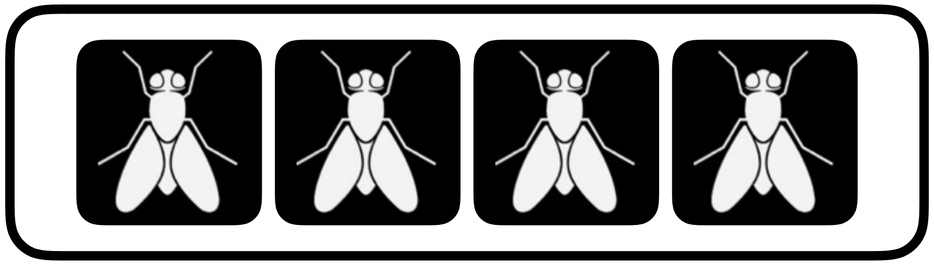
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

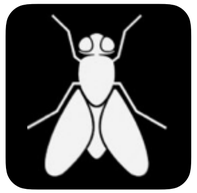


Program Count: **2** **2** **1**

Rule 5: Uniform Data Lifetime

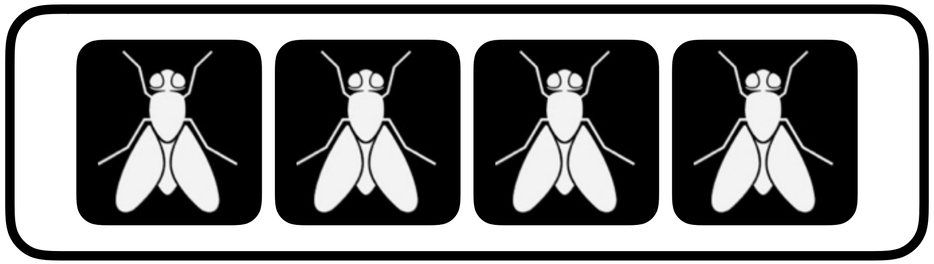
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

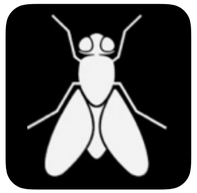


Program Count: **2 2 2 1**

Rule 5: Uniform Data Lifetime

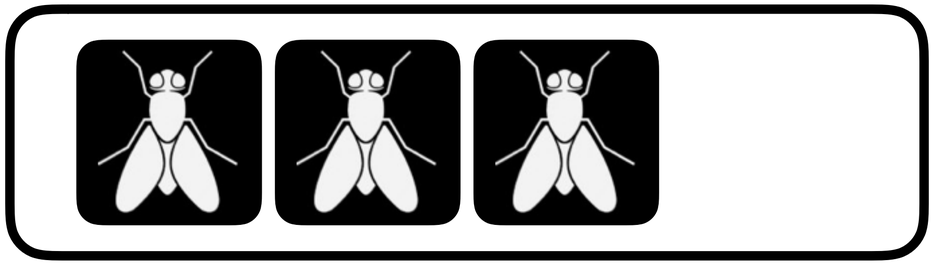
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

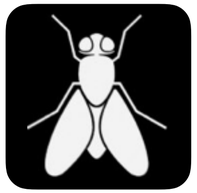


Program Count: **2 2 2 1**

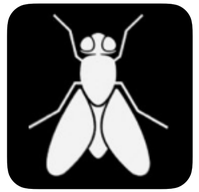
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

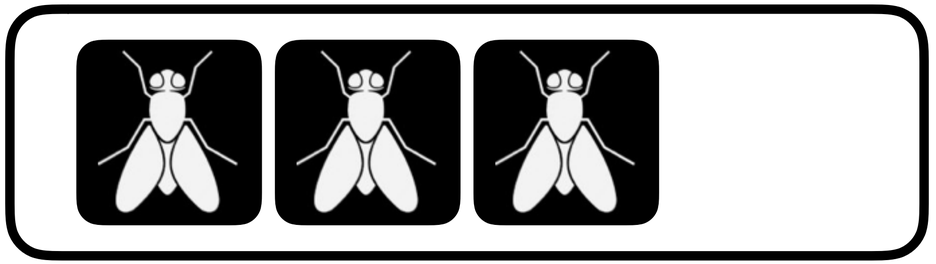
Lifetime



1 Day



SSD

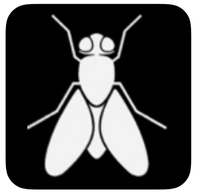


Program Count: **2 2 2 1**

Rule 5: Uniform Data Lifetime

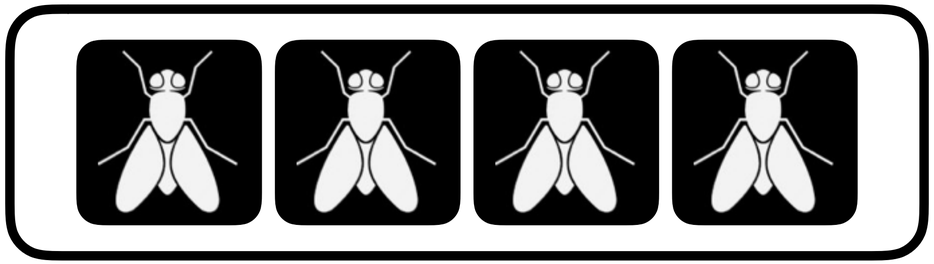
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

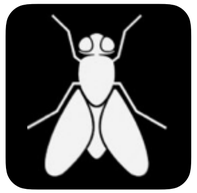


Program Count: **2 2 2 1**

Rule 5: Uniform Data Lifetime

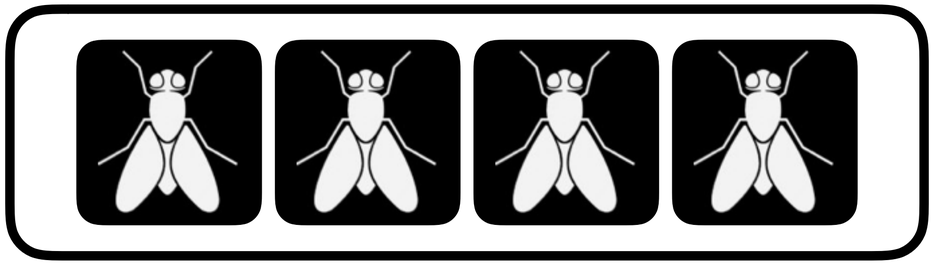
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

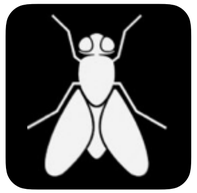


Program Count: **2** **2** **2**

Rule 5: Uniform Data Lifetime

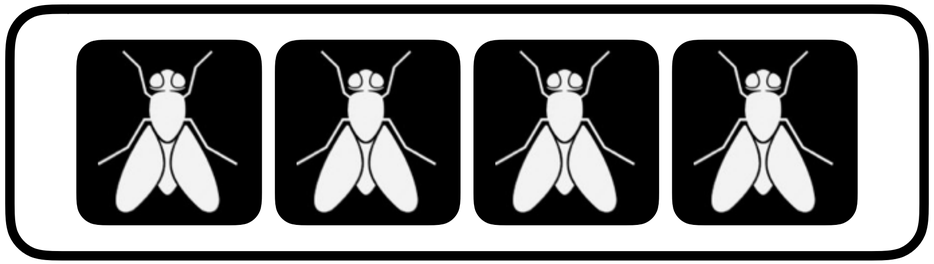
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

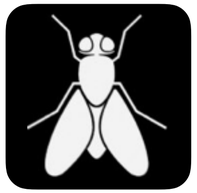


Program Count: **2** **2** **2** **2**

Rule 5: Uniform Data Lifetime

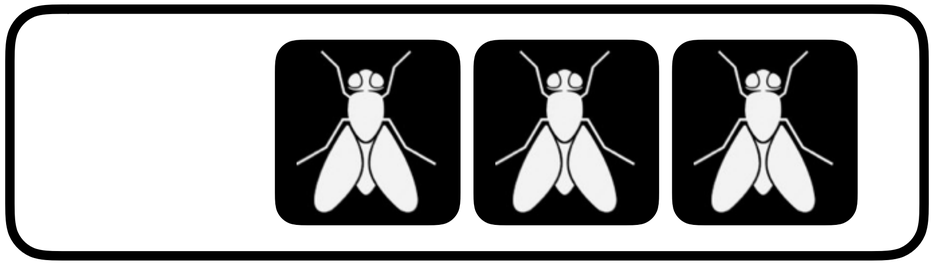
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

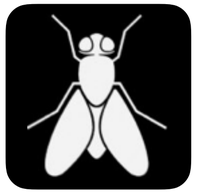


Program Count: **2** **2** **2** **2**

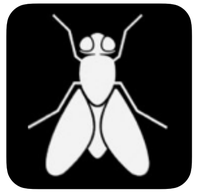
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

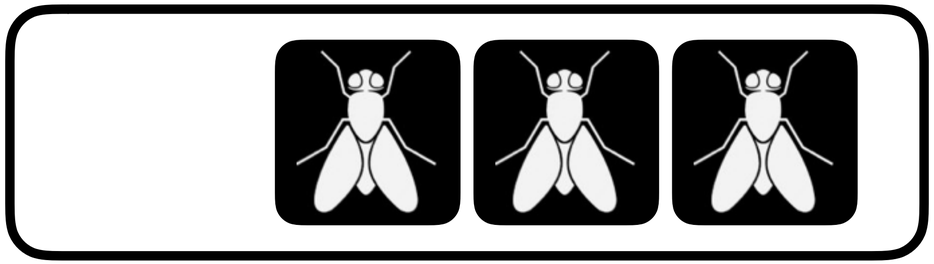
Lifetime



1 Day



SSD

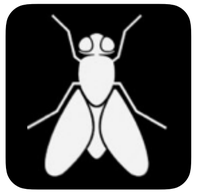


Program Count: **2** **2** **2** **2**

Rule 5: Uniform Data Lifetime

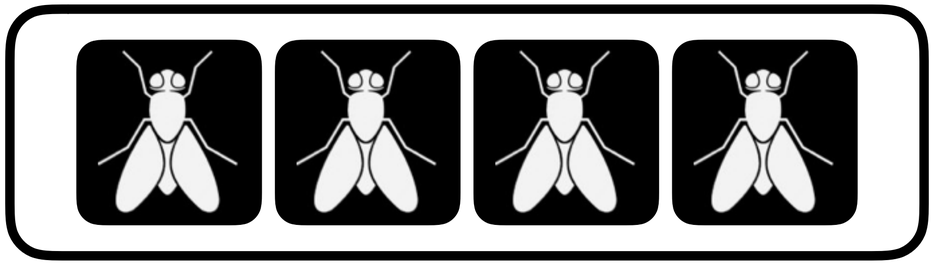
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

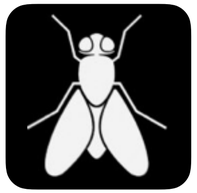


Program Count: **2** **2** **2** **2**

Rule 5: Uniform Data Lifetime

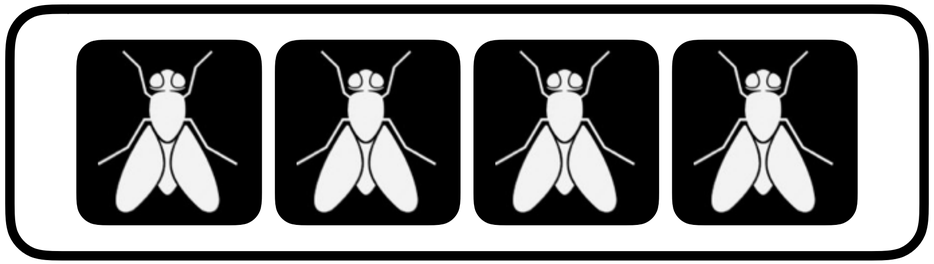
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

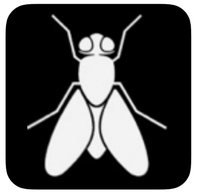


Program Count: **2** **2** **2**

Rule 5: Uniform Data Lifetime

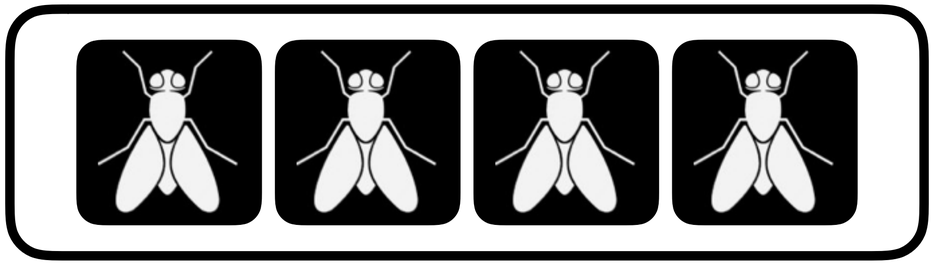
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

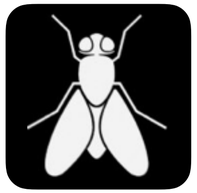


Program Count: **3** **2** **2** **2**

Rule 5: Uniform Data Lifetime

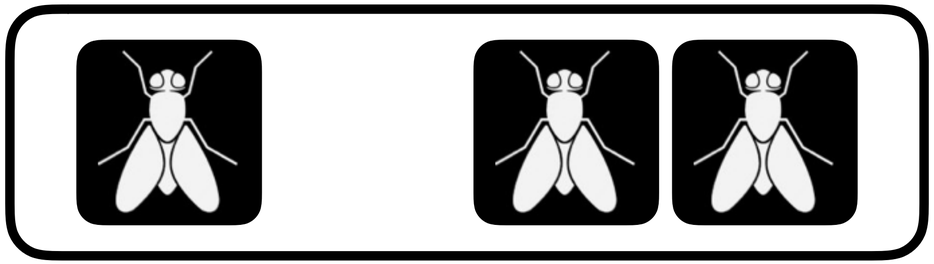
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

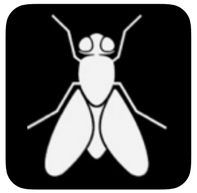


Program Count: **3** **2** **2** **2**

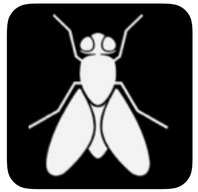
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

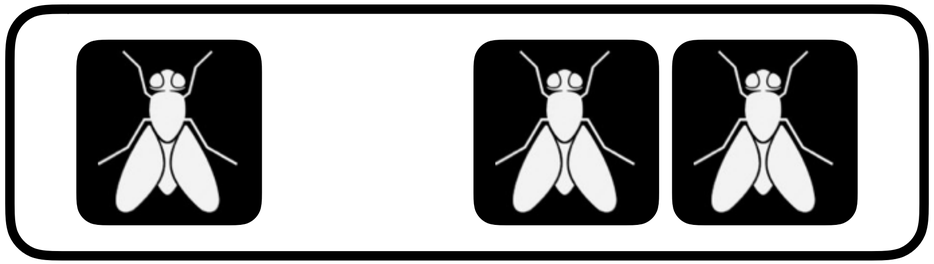
Lifetime



1 Day



SSD

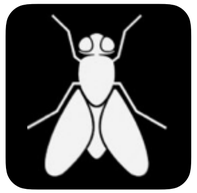


Program Count: **3** **2** **2** **2**

Rule 5: Uniform Data Lifetime

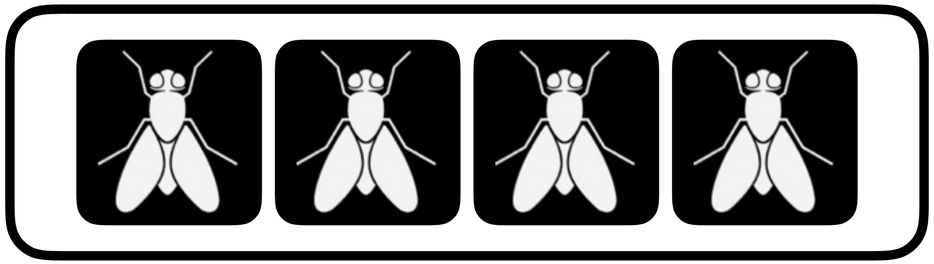
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

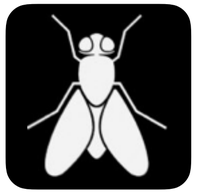


Program Count: **3** **2** **2** **2**

Rule 5: Uniform Data Lifetime

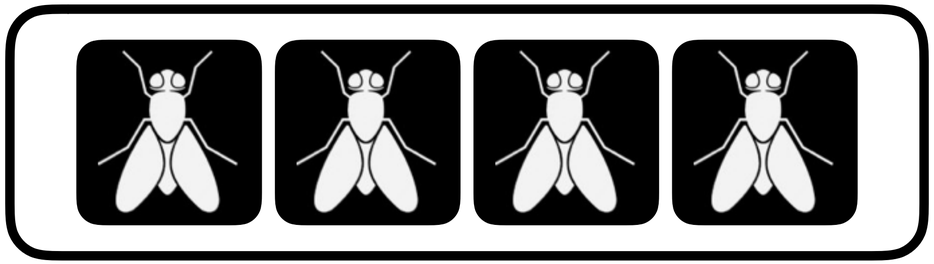
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

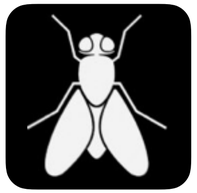


Program Count: **3** **2** **2**

Rule 5: Uniform Data Lifetime

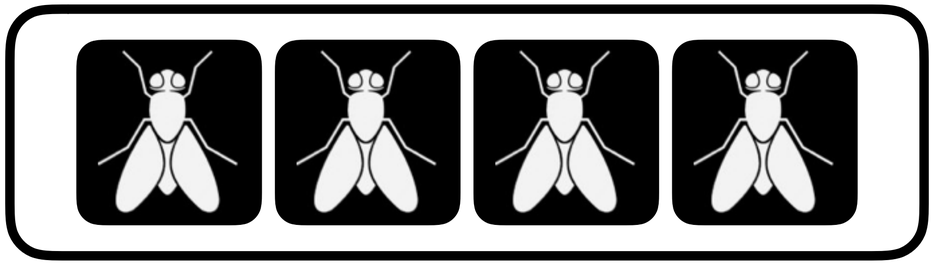
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

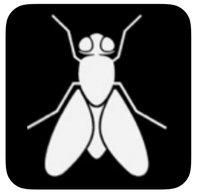


Program Count: **3** **3** **2** **2**

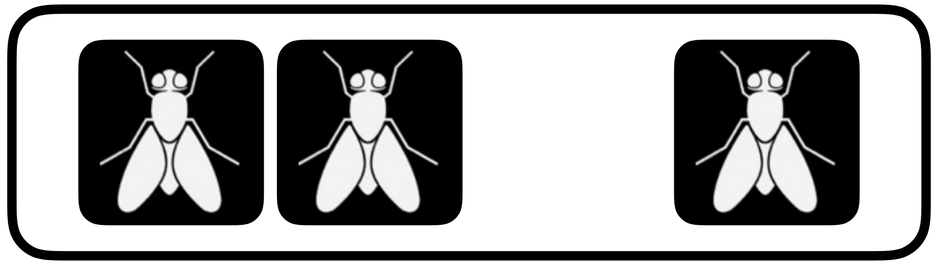
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

Lifetime



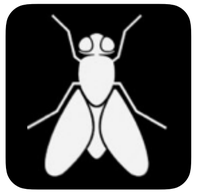
1 Day



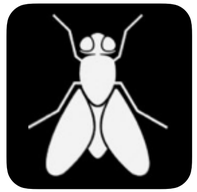
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

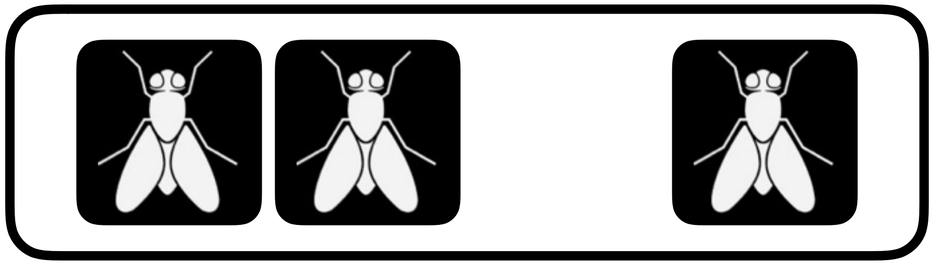
Lifetime



1 Day



SSD

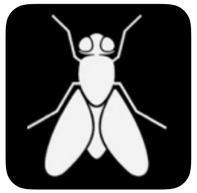


Program Count: 3 3 2 2

Rule 5: Uniform Data Lifetime

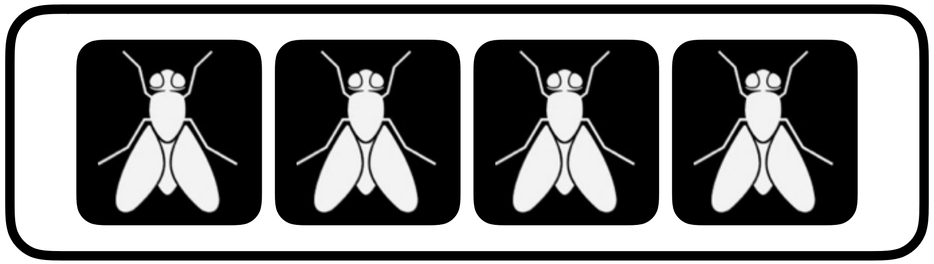
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

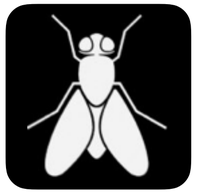


Program Count: **3** **3** **2** **2**

Rule 5: Uniform Data Lifetime

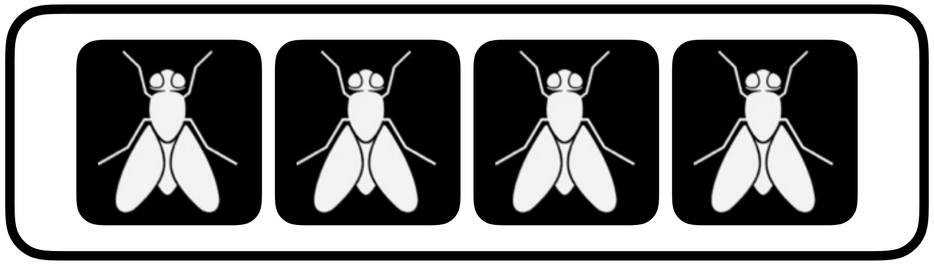
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

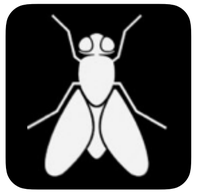


Program Count: **3** **3** **2**

Rule 5: Uniform Data Lifetime

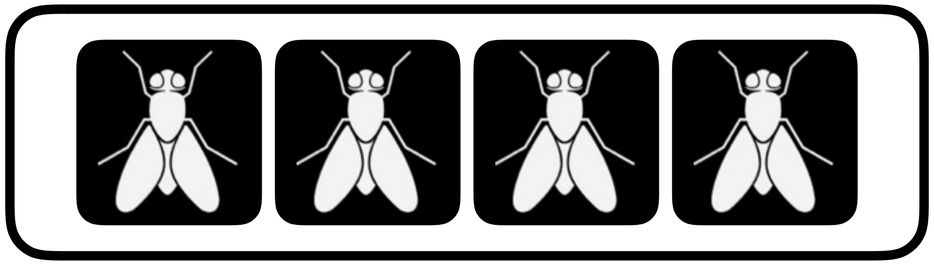
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

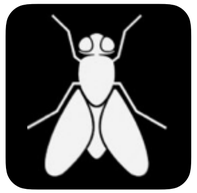


Program Count: **3** **3** **3** **2**

Rule 5: Uniform Data Lifetime

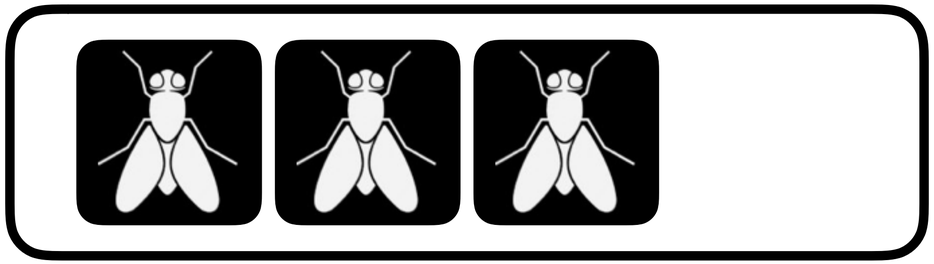
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

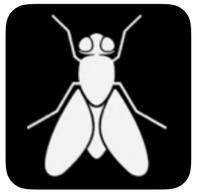


Program Count: **3** **3** **3** **2**

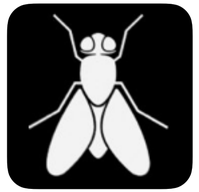
Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

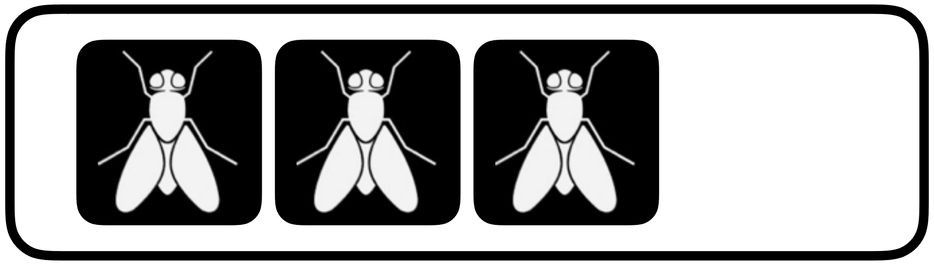
Lifetime



1 Day



SSD

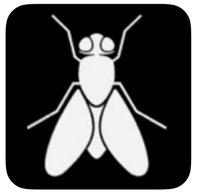


Program Count: **3** **3** **3** **2**

Rule 5: Uniform Data Lifetime

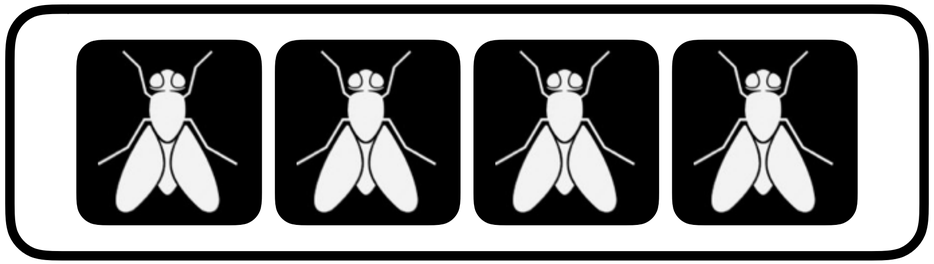
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

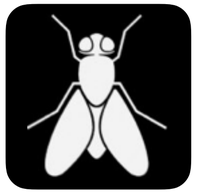


Program Count: **3** **3** **3** **2**

Rule 5: Uniform Data Lifetime

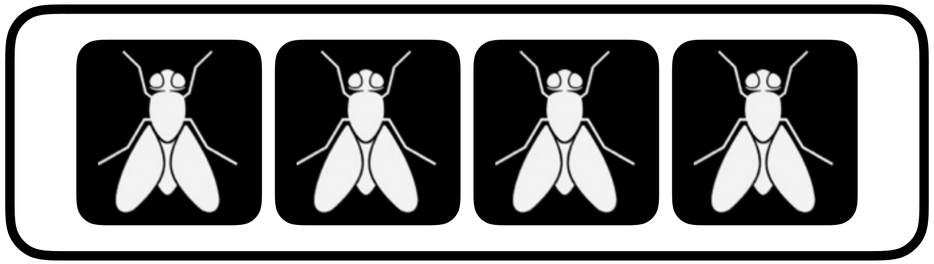
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

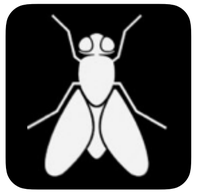


Program Count: **3** **3** **3**

Rule 5: Uniform Data Lifetime

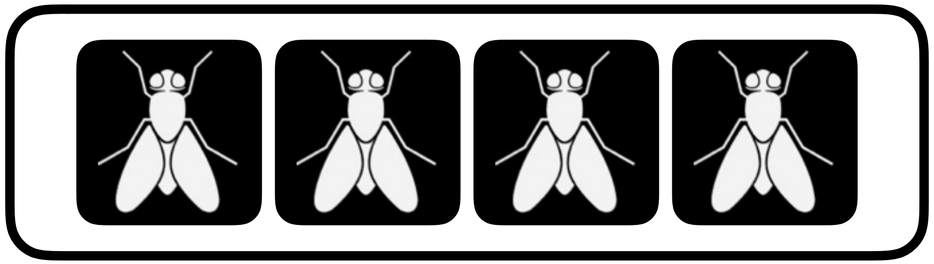
Clients of SSDs should create data with similar lifetimes

Lifetime



1 Day

SSD

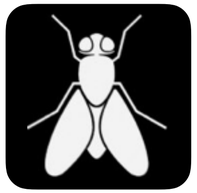


Program Count: **3** **3** **3** **3**

Rule 5: Uniform Data Lifetime

Clients of SSDs should create data with similar lifetimes

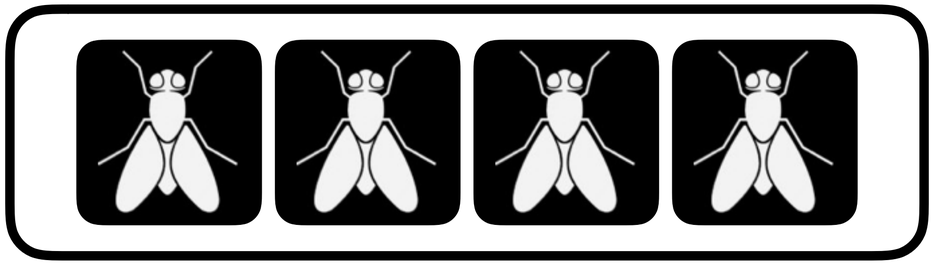
Lifetime



1 Day

No wear-leveling needed

SSD

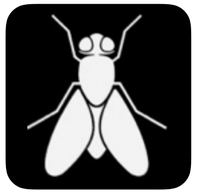


Program Count: **3** **3** **3** **3**

Rule 5: Uniform Data Lifetime

Violation

Lifetime



1 Day

1000 Years

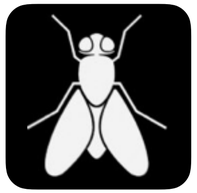
SSD

Program Count: **0** **0** **0** **0**

Rule 5: Uniform Data Lifetime

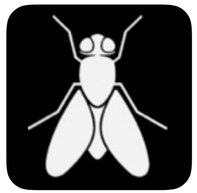
Violation

Lifetime



1 Day

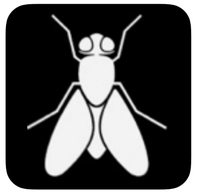
1000 Years



Rule 5: Uniform Data Lifetime

Violation

Lifetime



1 Day

1000 Years

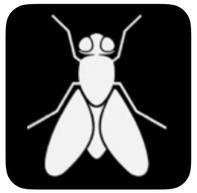
SSD

Program Count: **0** **0** **0** **0**

Rule 5: Uniform Data Lifetime

Violation

Lifetime



1 Day

1000 Years

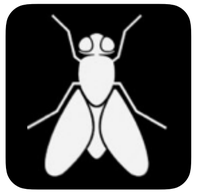
SSD

Program Count: **0** **0** **0**

Rule 5: Uniform Data Lifetime

Violation

Lifetime



1 Day

1000 Years

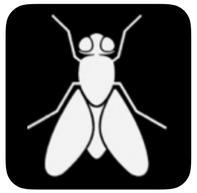
SSD

Program Count: **1 0 0 0**

Rule 5: Uniform Data Lifetime

Violation

Lifetime



1 Day

1000 Years

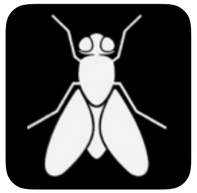
SSD

Program Count: 1 0 0 0

Rule 5: Uniform Data Lifetime

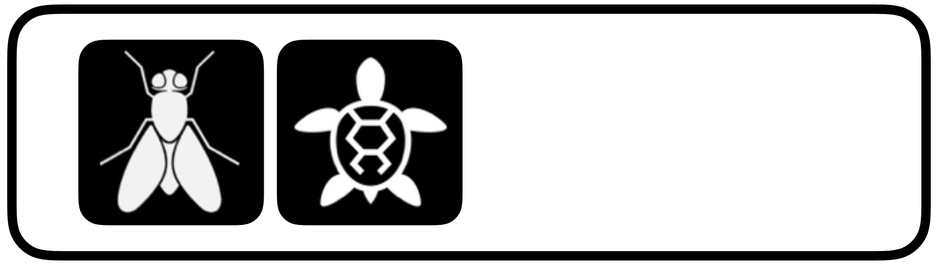
Violation

Lifetime



1 Day

1000 Years

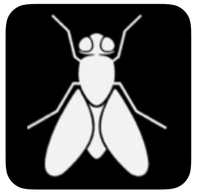


Program Count: 1 0 0 0

Rule 5: Uniform Data Lifetime

Violation

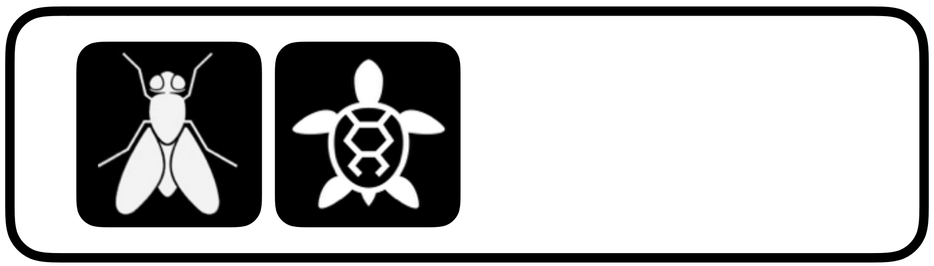
Lifetime



1 Day

1000 Years

SSD

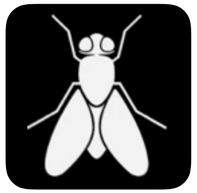


Program Count: **1** **0** **0**

Rule 5: Uniform Data Lifetime

Violation

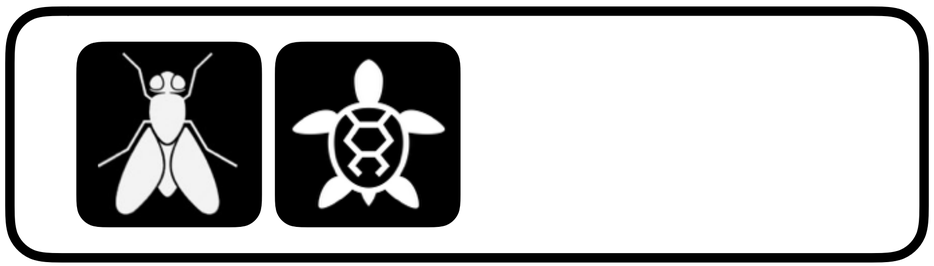
Lifetime



1 Day

1000 Years

SSD

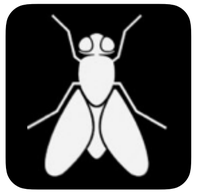


Program Count: 1 1 0 0

Rule 5: Uniform Data Lifetime

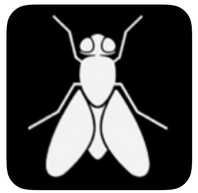
Violation

Lifetime

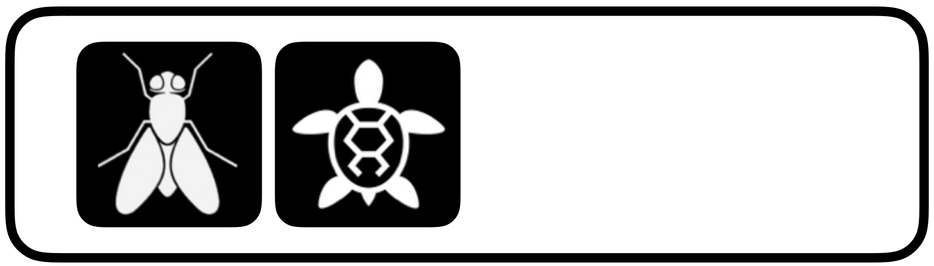


1 Day

1000 Years



SSD

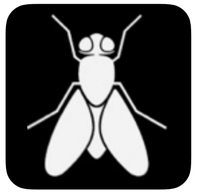


Program Count: 1 1 0 0

Rule 5: Uniform Data Lifetime

Violation

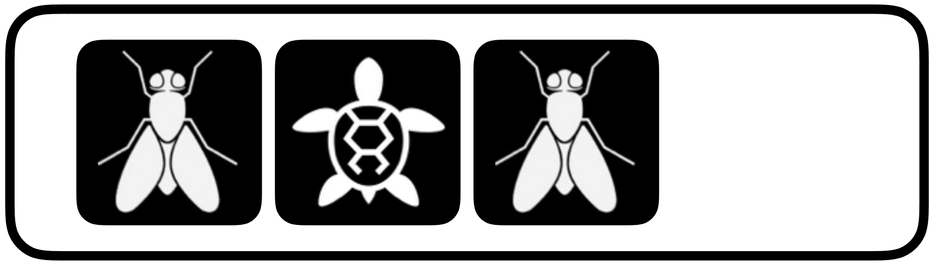
Lifetime



1 Day

1000 Years

SSD

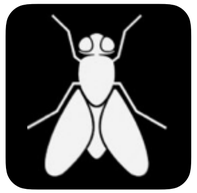


Program Count: 1 1 0 0

Rule 5: Uniform Data Lifetime

Violation

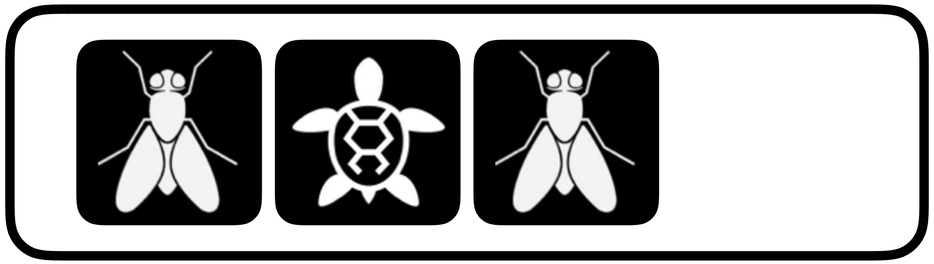
Lifetime



1 Day

1000 Years

SSD

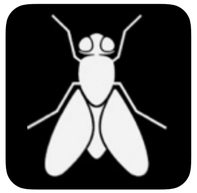


Program Count: **1** **1** **0**

Rule 5: Uniform Data Lifetime

Violation

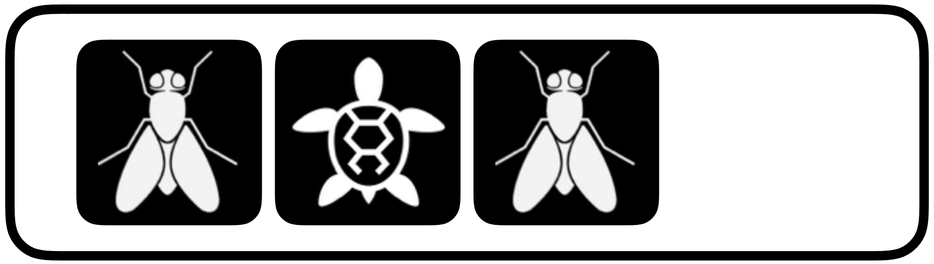
Lifetime



1 Day

1000 Years

SSD

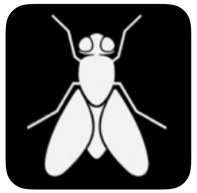


Program Count: **1 1 1 0**

Rule 5: Uniform Data Lifetime

Violation

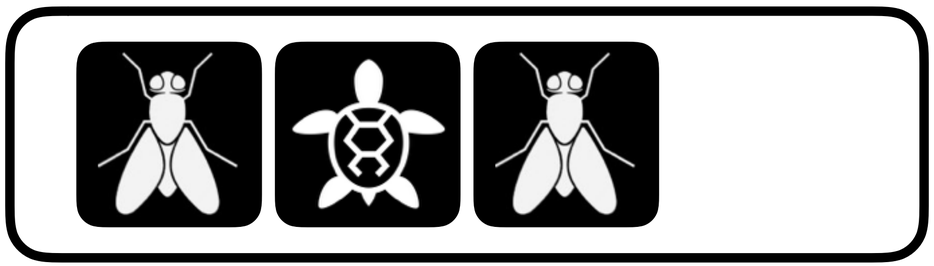
Lifetime



1 Day

1000 Years

SSD

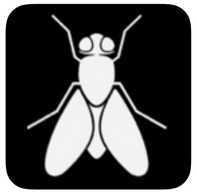


Program Count: 1 1 1 0

Rule 5: Uniform Data Lifetime

Violation

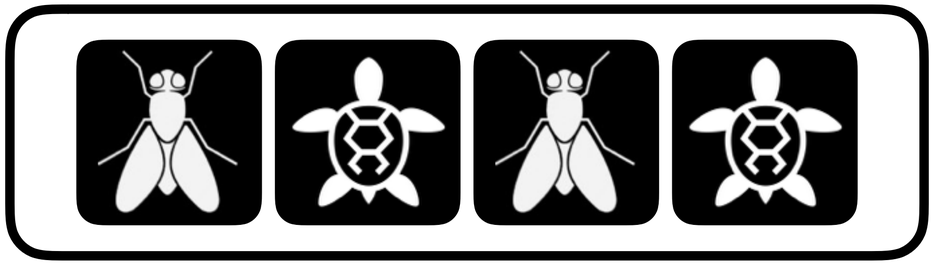
Lifetime



1 Day

1000 Years

SSD

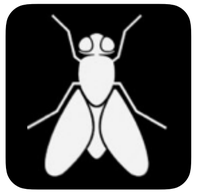


Program Count: **1 1 1 0**

Rule 5: Uniform Data Lifetime

Violation

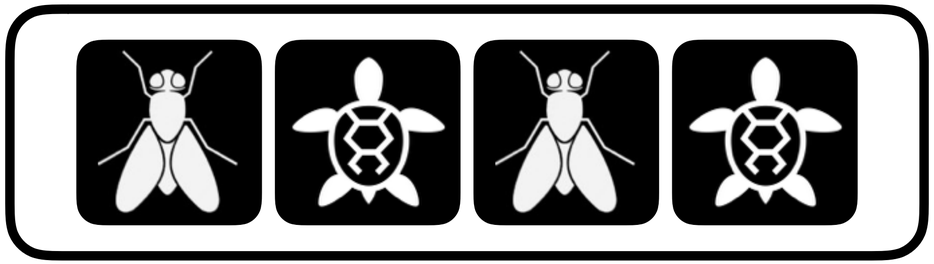
Lifetime



1 Day

1000 Years

SSD

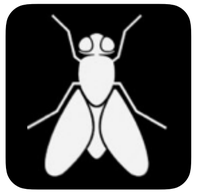


Program Count: 1 1 1

Rule 5: Uniform Data Lifetime

Violation

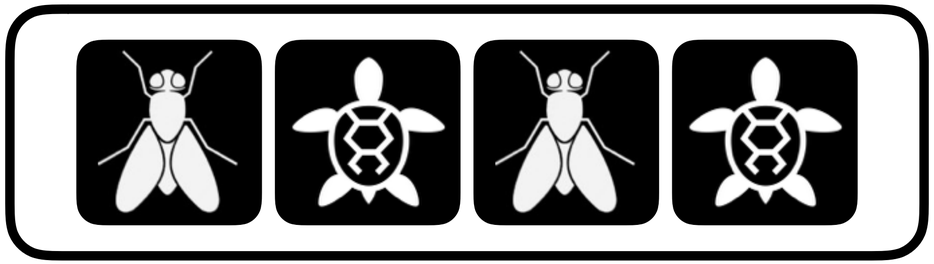
Lifetime



1 Day

1000 Years

SSD

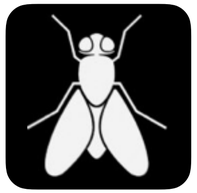


Program Count: 1 1 1 1

Rule 5: Uniform Data Lifetime

Violation

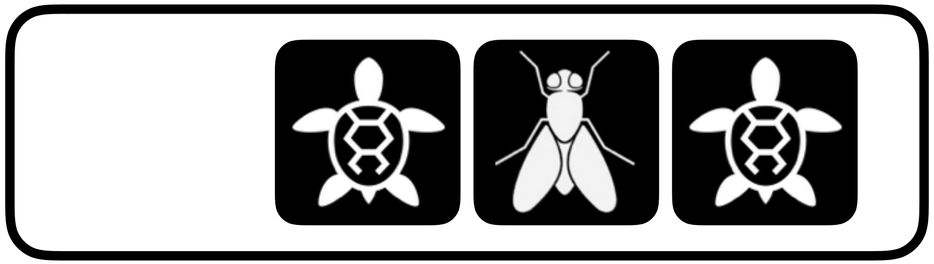
Lifetime



1 Day

1000 Years

SSD

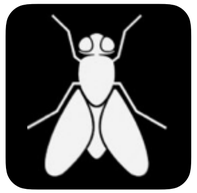


Program Count: 1 1 1 1

Rule 5: Uniform Data Lifetime

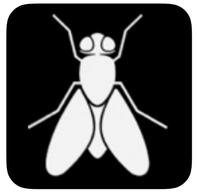
Violation

Lifetime

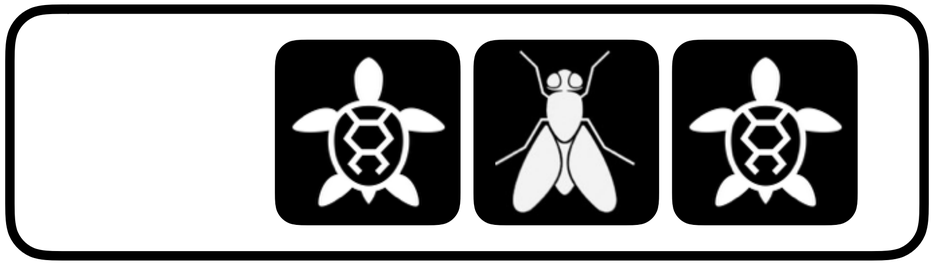


1 Day

1000 Years



SSD

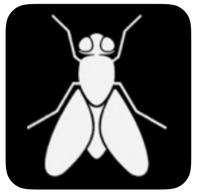


Program Count: 1 1 1 1

Rule 5: Uniform Data Lifetime

Violation

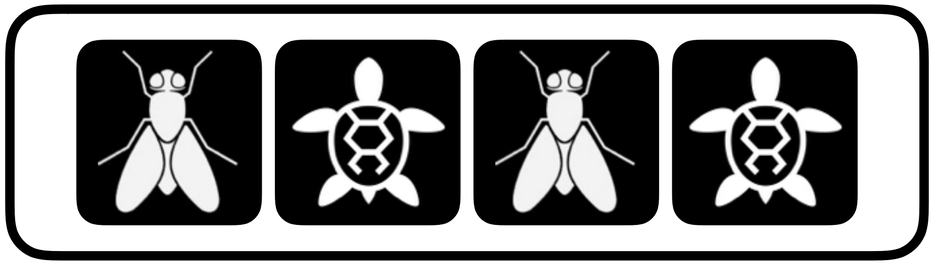
Lifetime



1 Day

1000 Years

SSD

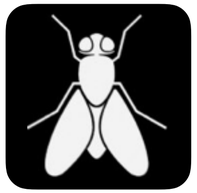


Program Count: 1 1 1 1

Rule 5: Uniform Data Lifetime

Violation

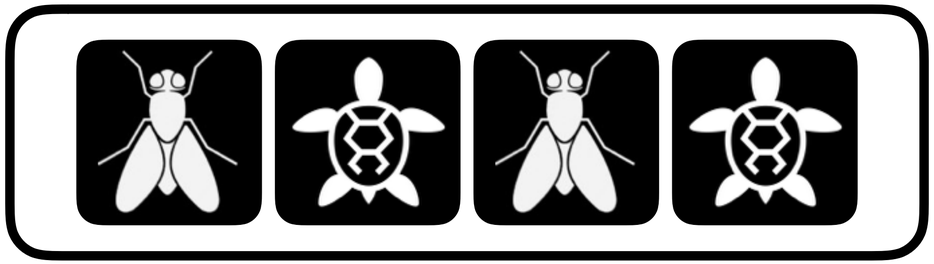
Lifetime



1 Day

1000 Years

SSD

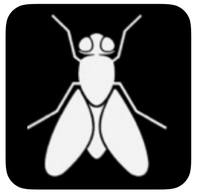


Program Count: 1 1 1

Rule 5: Uniform Data Lifetime

Violation

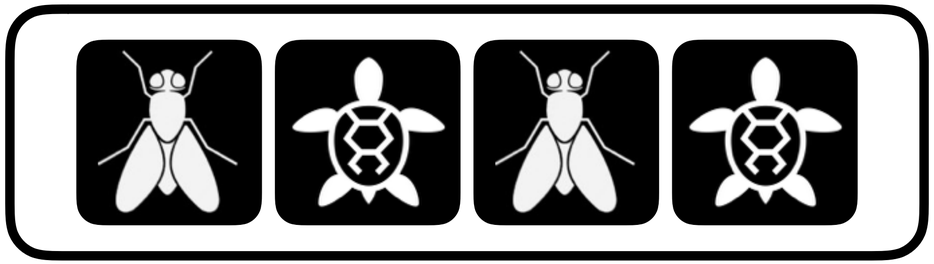
Lifetime



1 Day

1000 Years

SSD

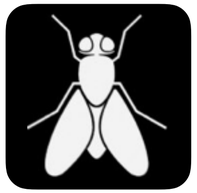


Program Count: 2 1 1 1

Rule 5: Uniform Data Lifetime

Violation

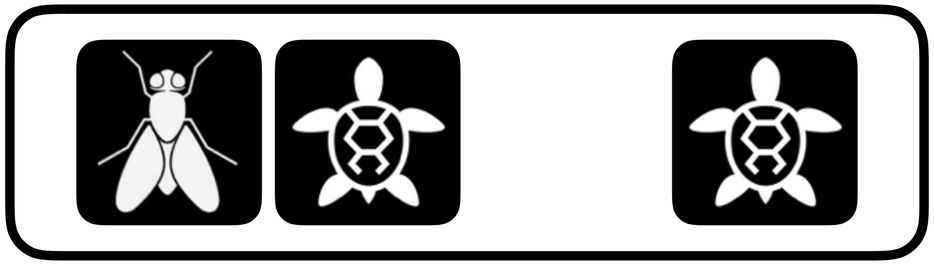
Lifetime



1 Day

1000 Years

SSD

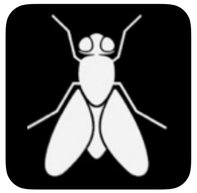


Program Count: 2 1 1 1

Rule 5: Uniform Data Lifetime

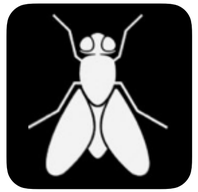
Violation

Lifetime

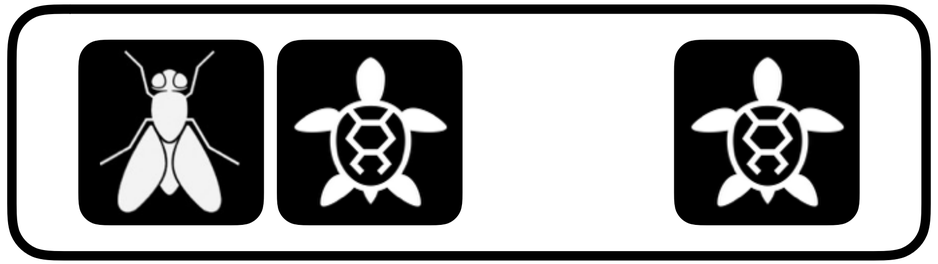


1 Day

1000 Years



SSD

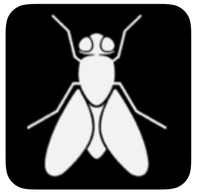


Program Count: 2 1 1 1

Rule 5: Uniform Data Lifetime

Violation

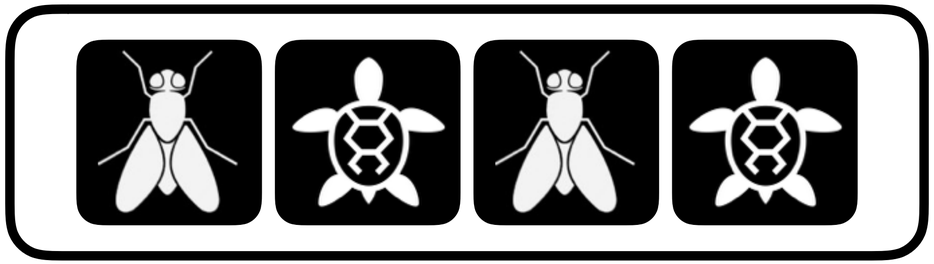
Lifetime



1 Day

1000 Years

SSD

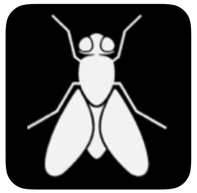


Program Count: 2 1 1 1

Rule 5: Uniform Data Lifetime

Violation

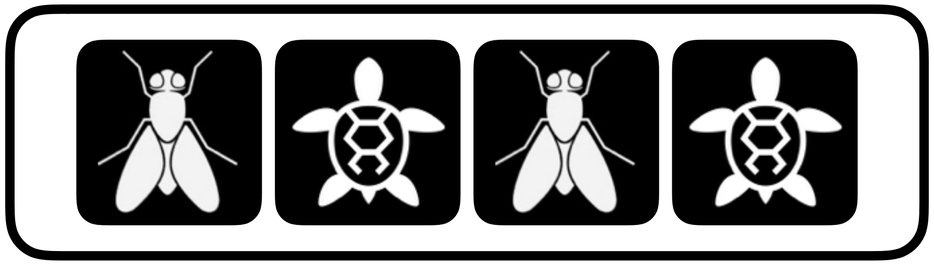
Lifetime



1 Day

1000 Years

SSD

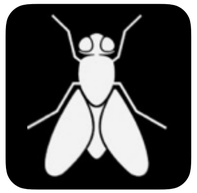


Program Count: 2 1 1

Rule 5: Uniform Data Lifetime

Violation

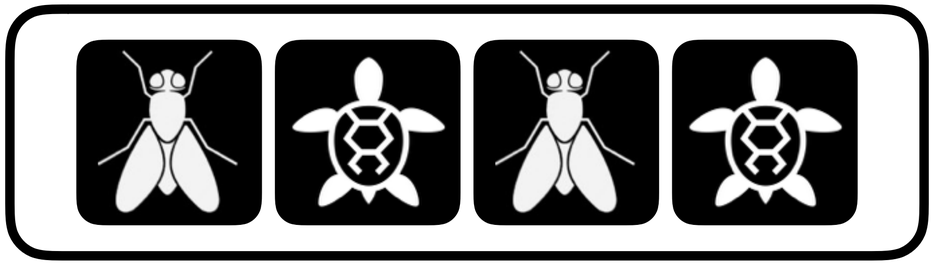
Lifetime



1 Day

1000 Years

SSD

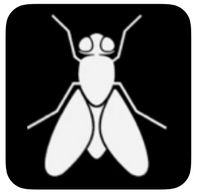


Program Count: 2 1 2 1

Rule 5: Uniform Data Lifetime

Violation

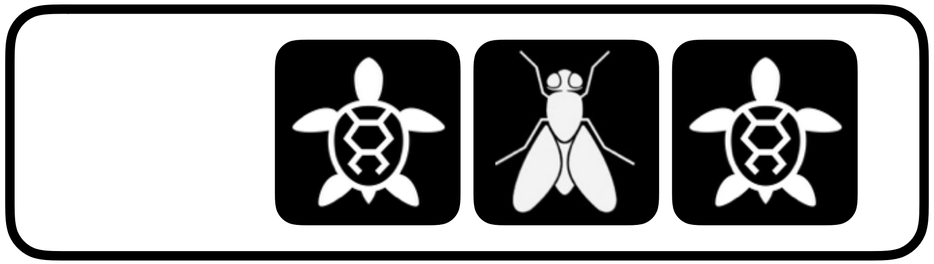
Lifetime



1 Day

1000 Years

SSD

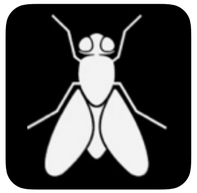


Program Count: 2 1 2 1

Rule 5: Uniform Data Lifetime

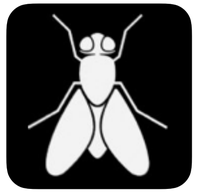
Violation

Lifetime

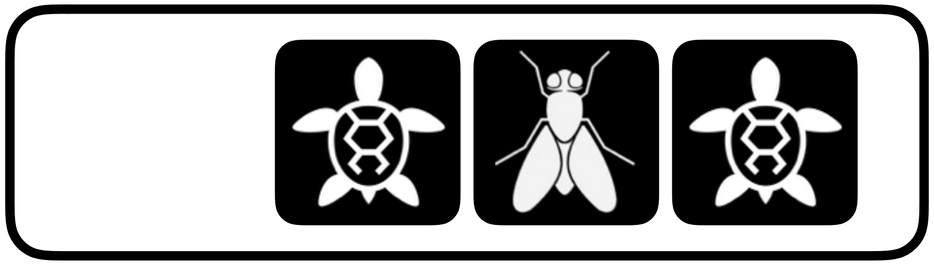


1 Day

1000 Years



SSD

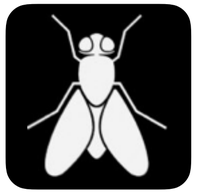


Program Count: 2 1 2 1

Rule 5: Uniform Data Lifetime

Violation

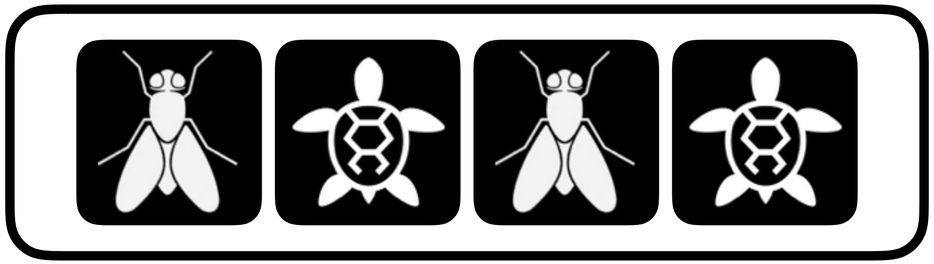
Lifetime



1 Day

1000 Years

SSD

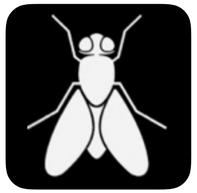


Program Count: 2 1 2 1

Rule 5: Uniform Data Lifetime

Violation

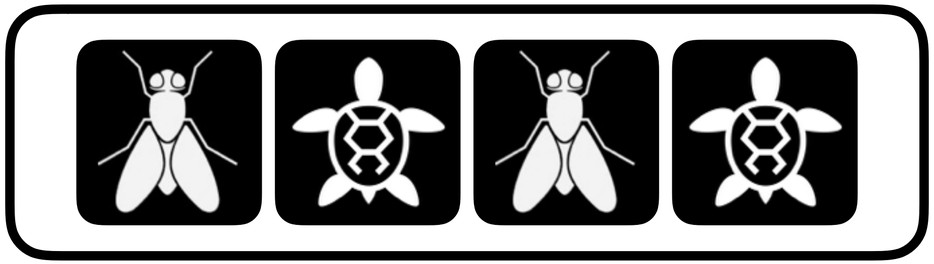
Lifetime



1 Day

1000 Years

SSD

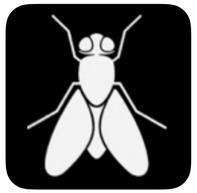


Program Count: 1 2 1

Rule 5: Uniform Data Lifetime

Violation

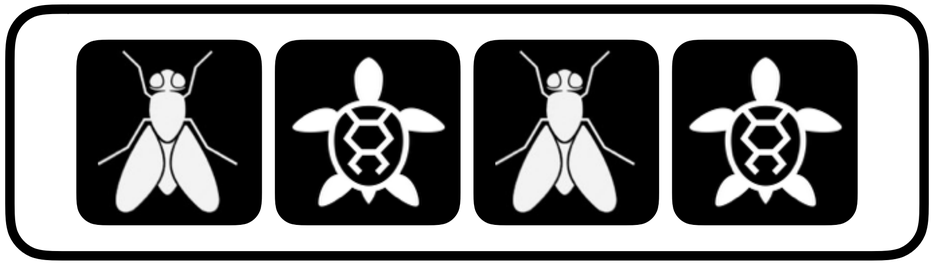
Lifetime



1 Day

1000 Years

SSD

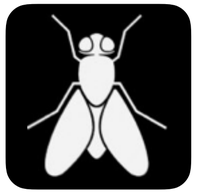


Program Count: 3 1 2 1

Rule 5: Uniform Data Lifetime

Violation

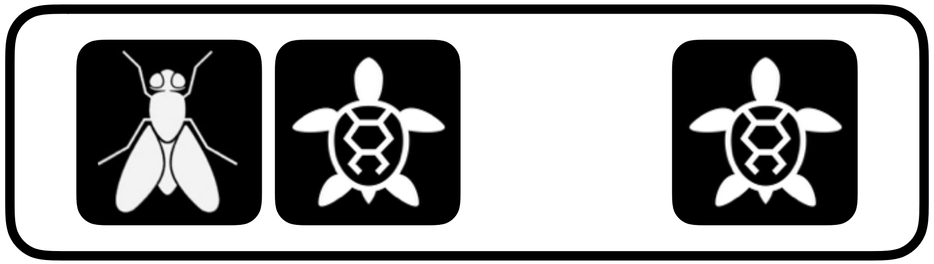
Lifetime



1 Day

1000 Years

SSD

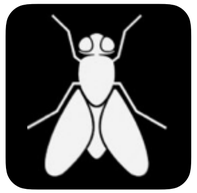


Program Count: 3 1 2 1

Rule 5: Uniform Data Lifetime

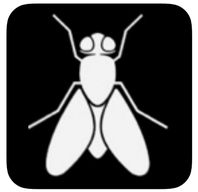
Violation

Lifetime

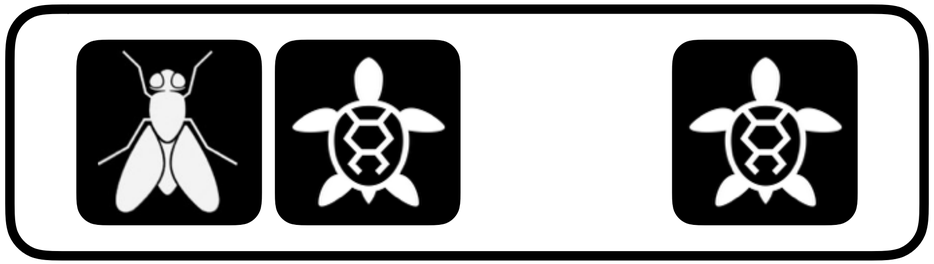


1 Day

1000 Years



SSD

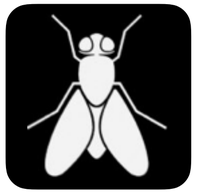


Program Count: 3 1 2 1

Rule 5: Uniform Data Lifetime

Violation

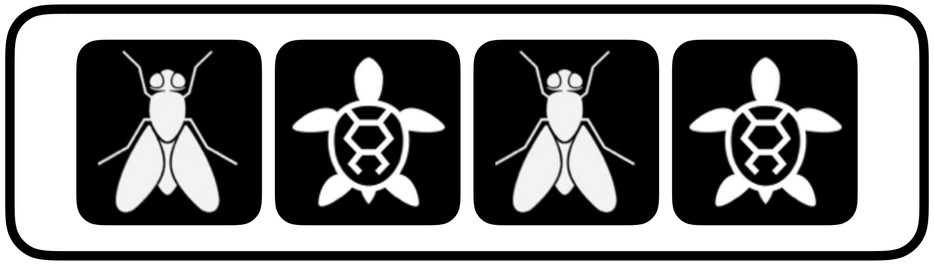
Lifetime



1 Day

1000 Years

SSD

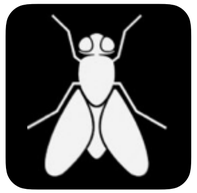


Program Count: 3 1 2 1

Rule 5: Uniform Data Lifetime

Violation

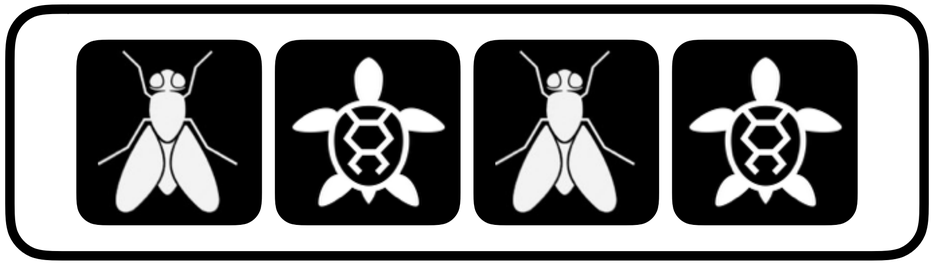
Lifetime



1 Day

1000 Years

SSD

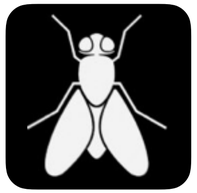


Program Count: 3 1 1

Rule 5: Uniform Data Lifetime

Violation

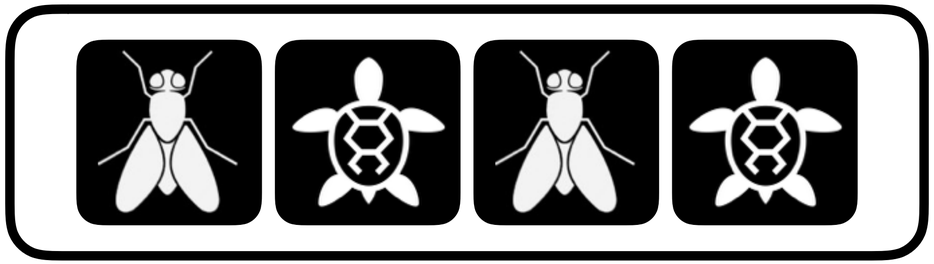
Lifetime



1 Day

1000 Years

SSD

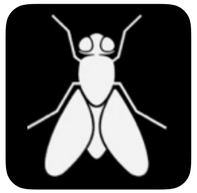


Program Count: 3 1 3 1

Rule 5: Uniform Data Lifetime

Violation

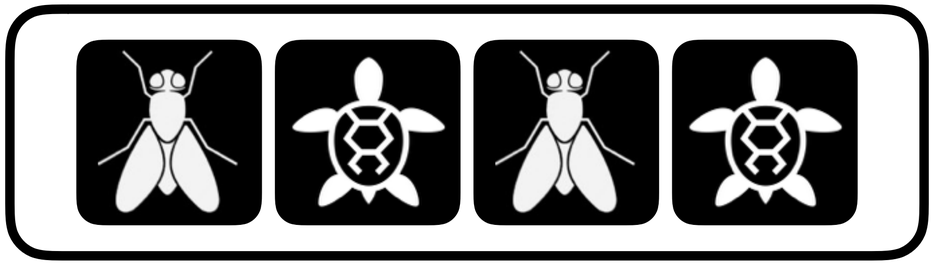
Lifetime



1 Day

1000 Years

SSD

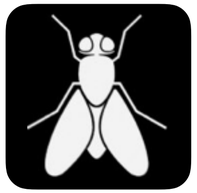


Program Count: 3 1 3 1

Rule 5: Uniform Data Lifetime

Violation

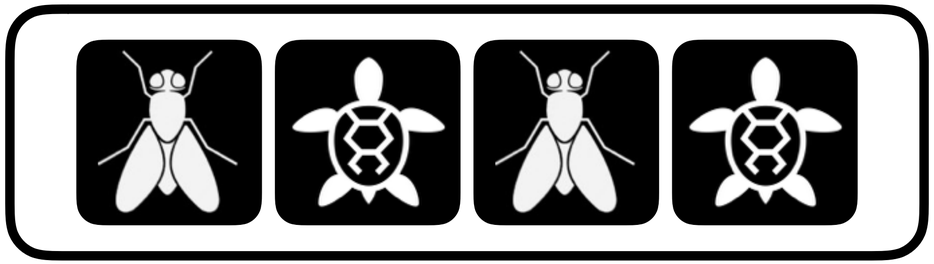
Lifetime



1 Day

1000 Years

SSD

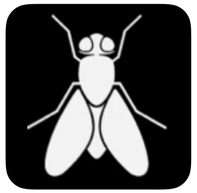


Program Count: 365*1000 1 365*1000 1

Rule 5: Uniform Data Lifetime

Violation

Lifetime

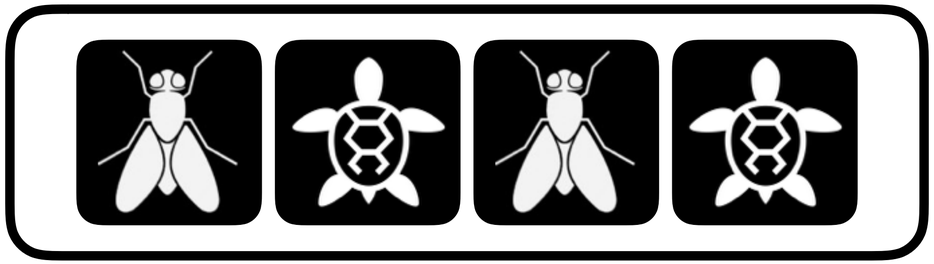


1 Day

1000 Years

Frequent wear-leveling needed!!!

SSD

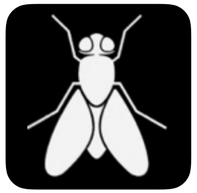


Program Count: 365*1000 1 365*1000 1

Rule 5: Uniform Data Lifetime

Violation

Lifetime



1 Day

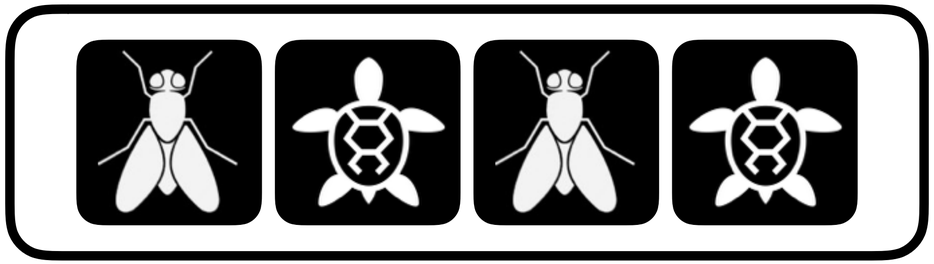
1000 Years

If you violate the rule:

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

Frequent wear-leveling needed!!!

SSD

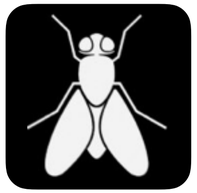


Program Count: **365*1000** **1** **365*1000** **1**

Rule 5: Uniform Data Lifetime

Violation

Lifetime



1 Day

1000 Year

If you violate the rule:

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

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.

Program Count:

A FTL often has a subset of rules

A FTL often has a subset of rules

Rules

#1 Request Scale

#2 Locality

#3 Aligned Sequentiality

#4 Grouping by Death Time

#5 Uniform Data Lifetime

A FTL often has a subset of rules

Page-level Mapping FTL

Rules

#1 Request Scale

#2 Locality

#3 Aligned Sequentiality

#4 Grouping by Death Time

#5 Uniform Data Lifetime

A FTL often has a subset of rules

Rules	Page-level Mapping FTL
	Immediate
#1 Request Scale	**x**
#2 Locality	**x**
#3 Aligned Sequentiality	
#4 Grouping by Death Time	
#5 Uniform Data Lifetime	

A FTL often has a subset of rules

	Page-level Mapping FTL	
Rules	**Immediate**	**Sustainable**
#1 Request Scale	**x**	**x**
#2 Locality	**x**	**x**
#3 Aligned Sequentiality		
#4 Grouping by Death Time		**x**
#5 Uniform Data Lifetime		**x**

A FTL often has a subset of rules

	Page-level Mapping FTL		Hybrid Mapping FTL
Rules	**Immediate**	**Sustainable**	
#1 Request Scale	**x**	**x**	
#2 Locality	**x**	**x**	
#3 Aligned Sequentiality			
#4 Grouping by Death Time		**x**	
#5 Uniform Data Lifetime		**x**	

A FTL often has a subset of rules

	Page-level Mapping FTL		Hybrid Mapping FTL
Rules	**Immediate**	**Sustainable**	**Immediate**
#1 Request Scale	**x**	**x**	**x**
#2 Locality	**x**	**x**	
#3 Aligned Sequentiality			
#4 Grouping by Death Time		**x**	
#5 Uniform Data Lifetime		**x**	

A FTL often has a subset of rules

	Page-level Mapping FTL		Hybrid Mapping FTL	
Rules	**Immediate**	**Sustainable**	**Immediate**	**Sustainable**
#1 Request Scale	**x**	**x**	**x**	**x**
#2 Locality	**x**	**x**		
#3 Aligned Sequentiality				**x**
#4 Grouping by Death Time		**x**		
#5 Uniform Data Lifetime		**x**		**x**

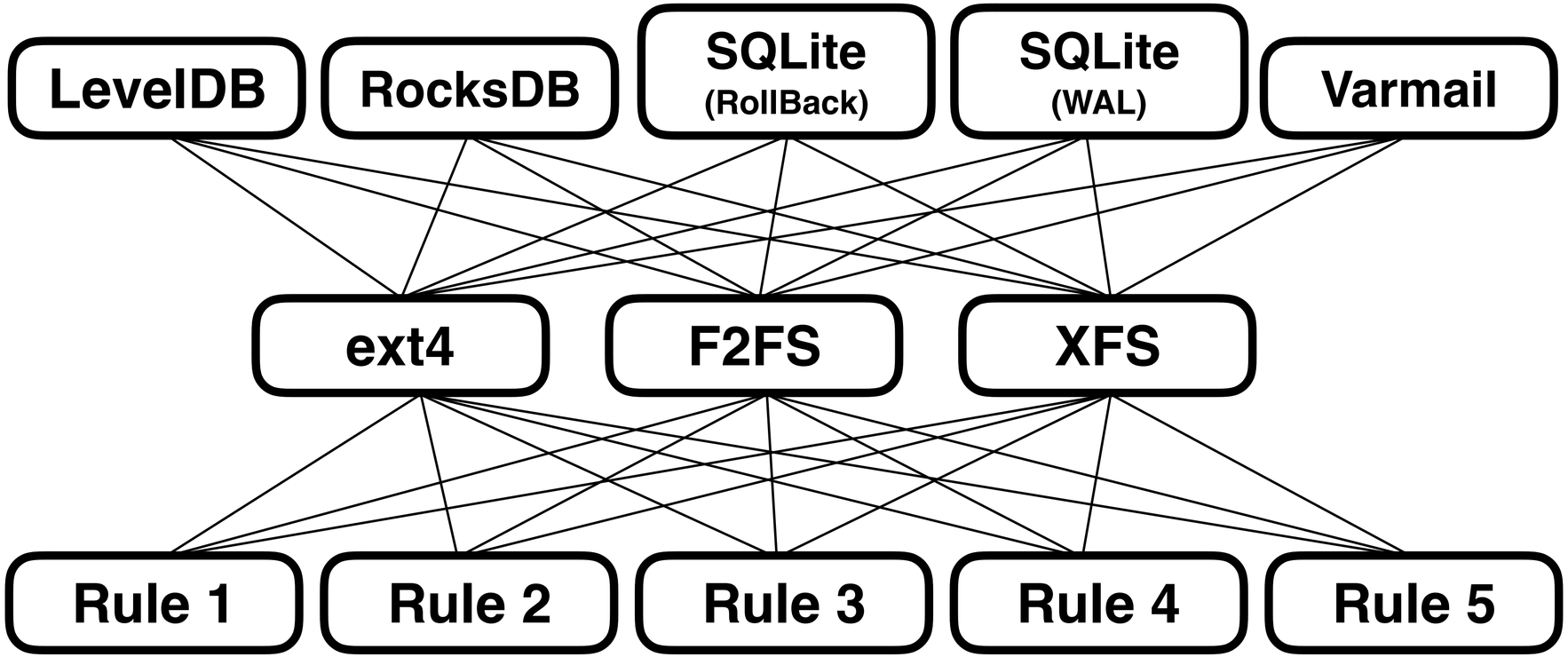
Do applications/file systems comply with the unwritten contract?

Outline

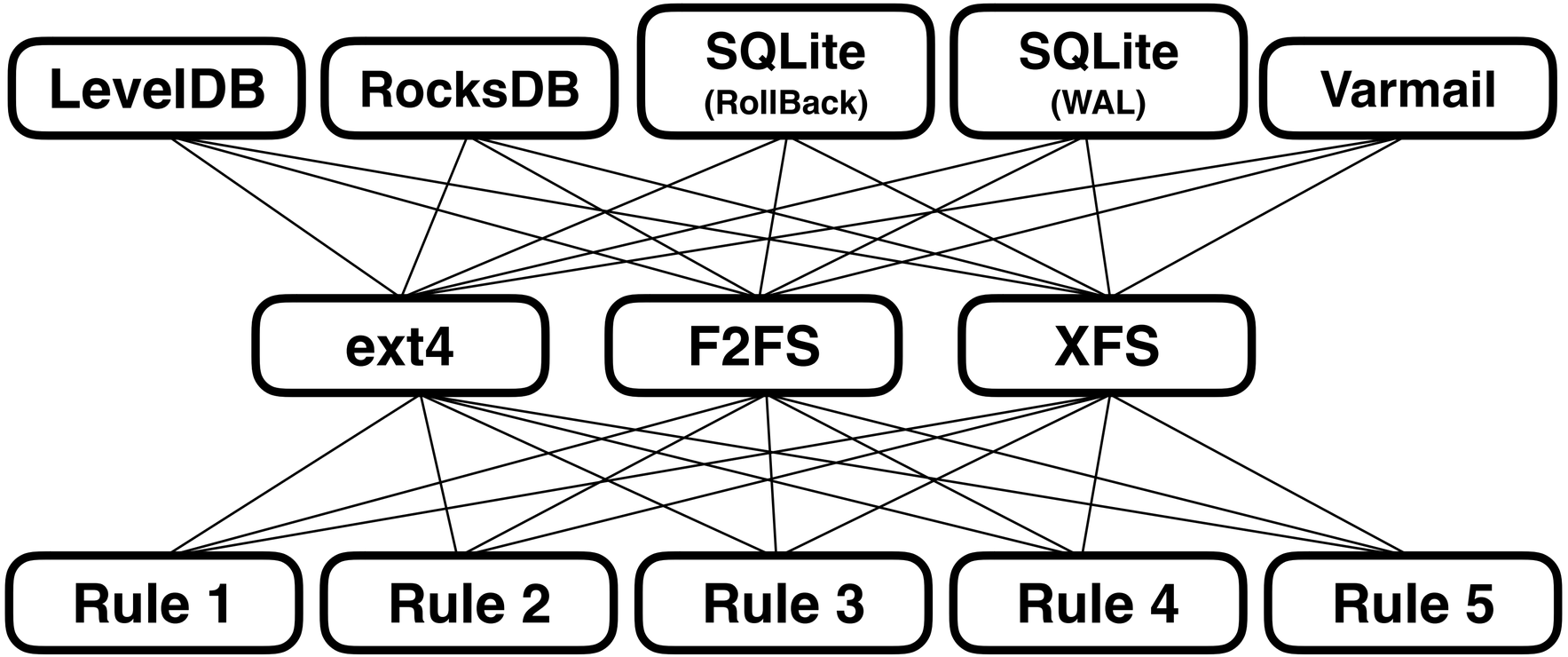
- Overview
- SSD Unwritten Contract
- **Violations of Unwritten Contract**
  - Method
  - Observations
- Lessons Learned
- Conclusions

Outline

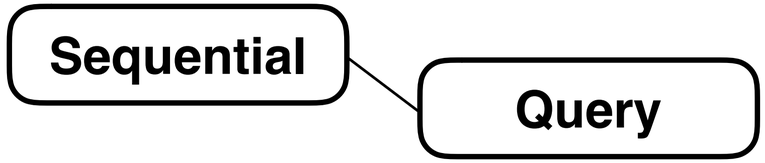
- Overview
- SSD Unwritten Contract
- Violations of Unwritten Contract
  - **Method**
  - Observations
- Lessons Learned
- Conclusions

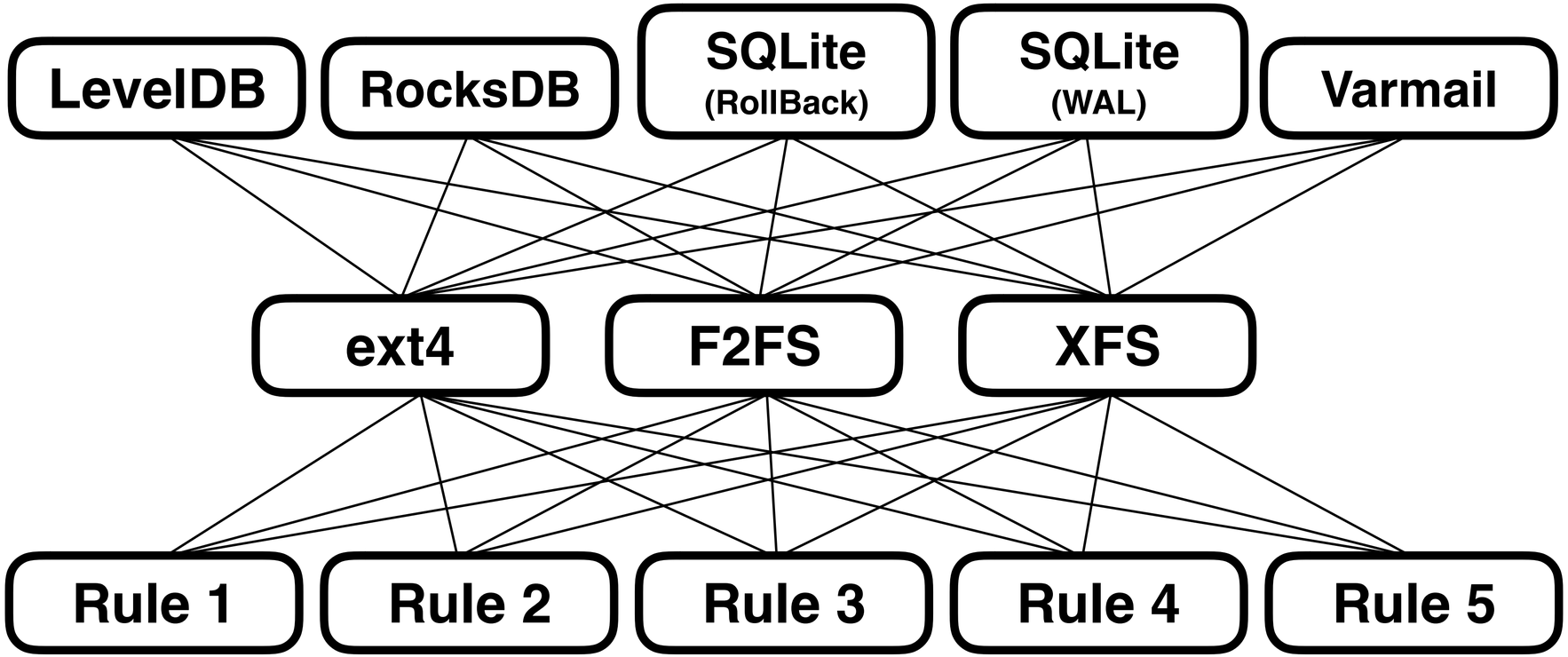

LevelDB
RocksDB
SQLite
(RollBack)
SQLite
(WAL)
Varmail
ext4
F2FS
XFS
Rule 1
Rule 2
Rule 3
Rule 4
Rule 5

Database Operations:

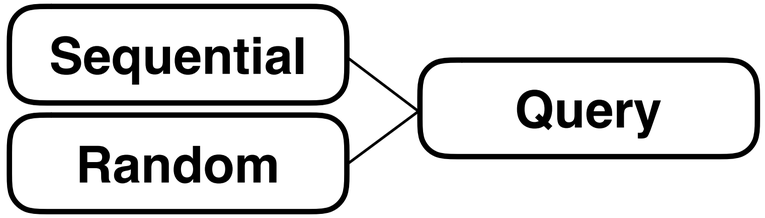


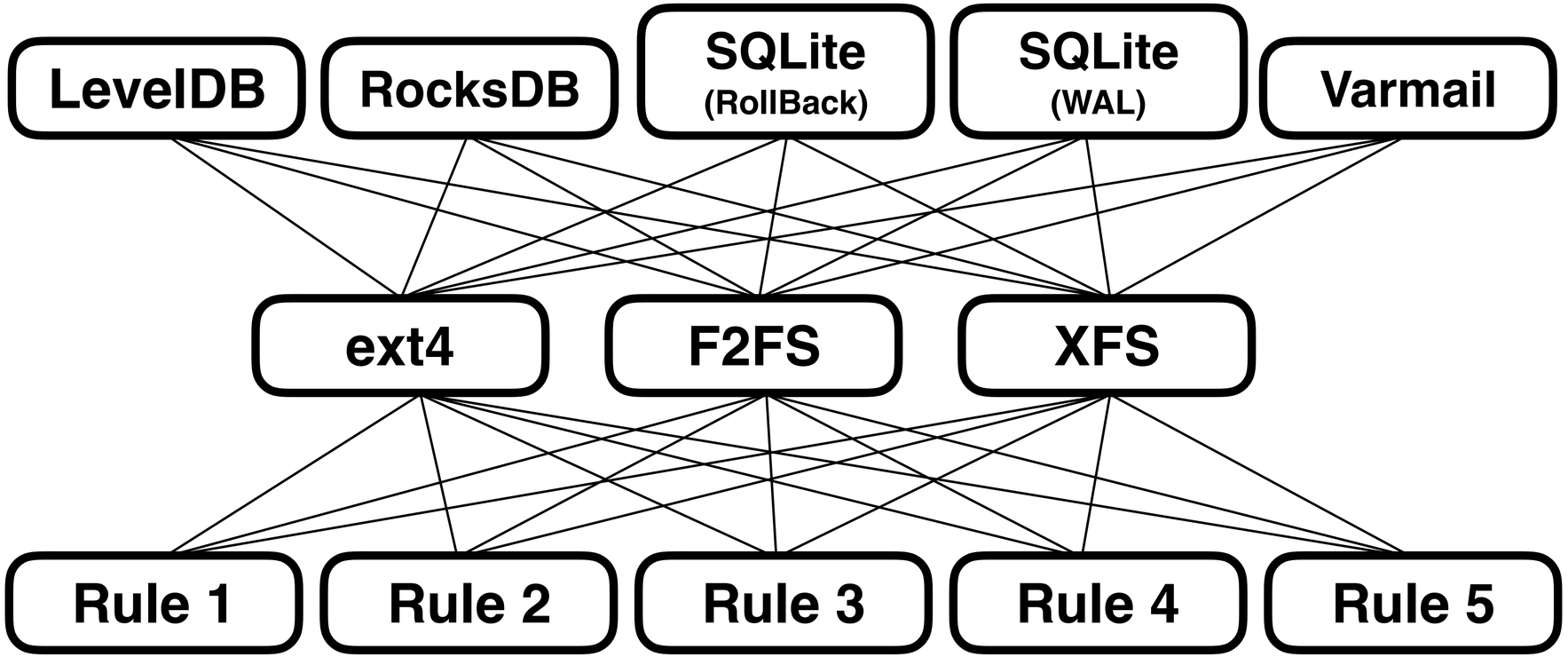
Database Operations:



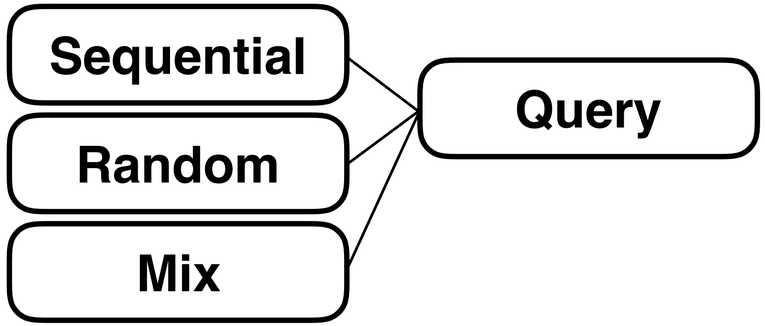


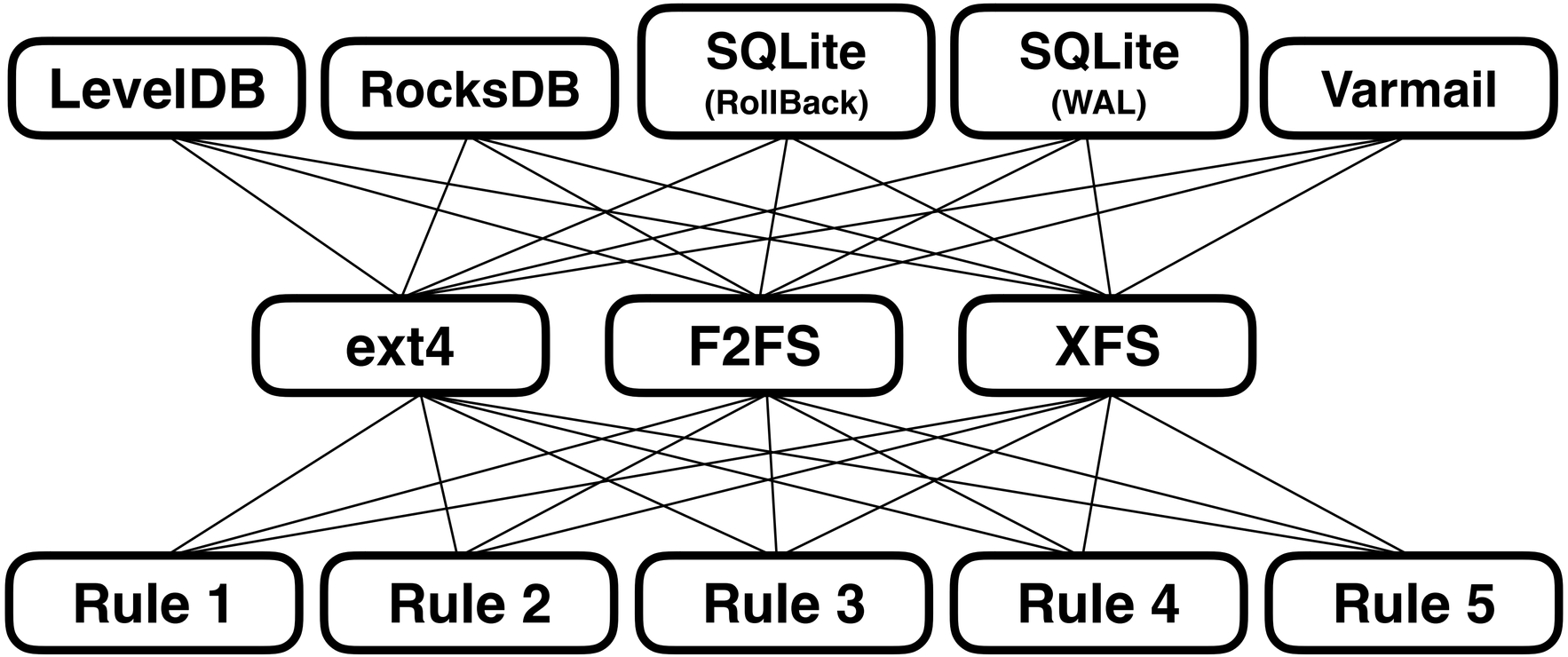
Database Operations:



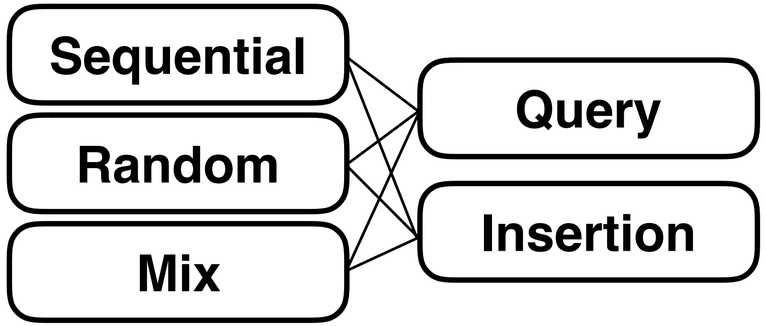


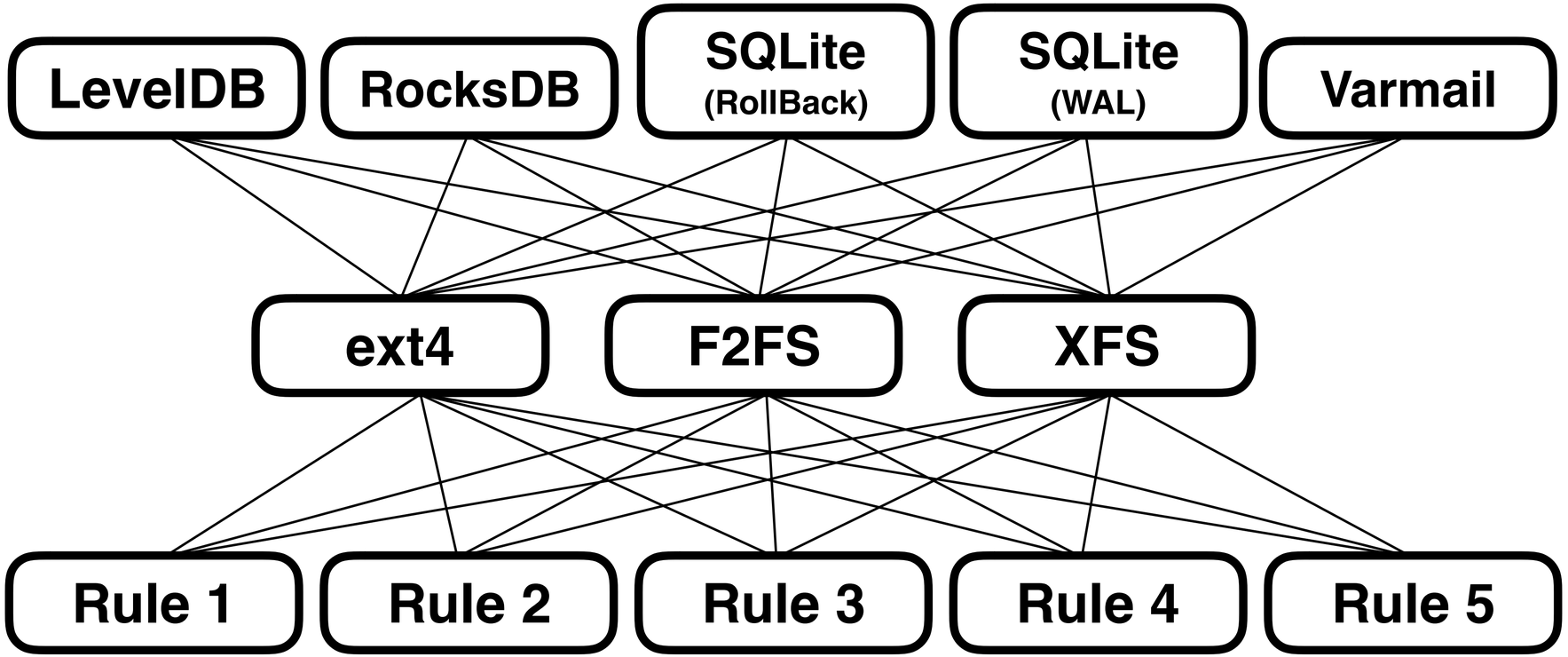
Database Operations:



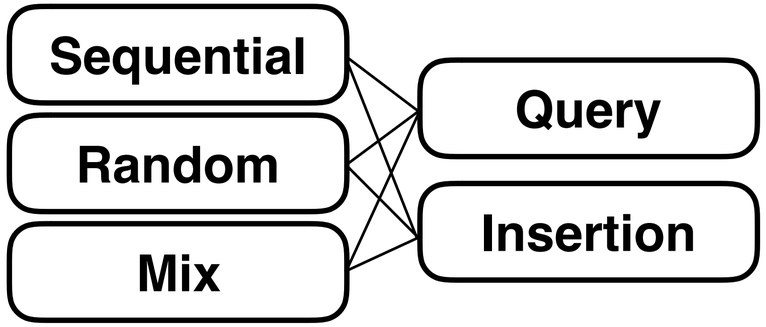


Database Operations:

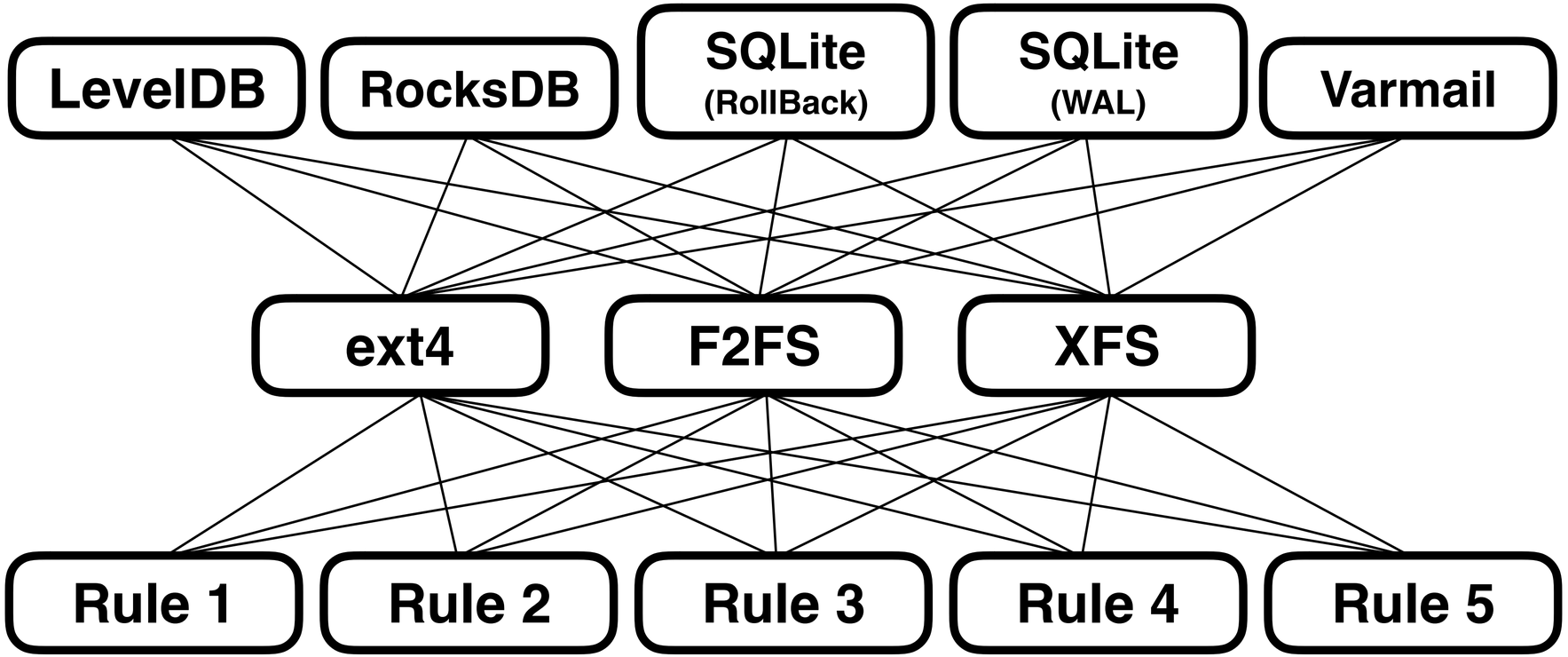




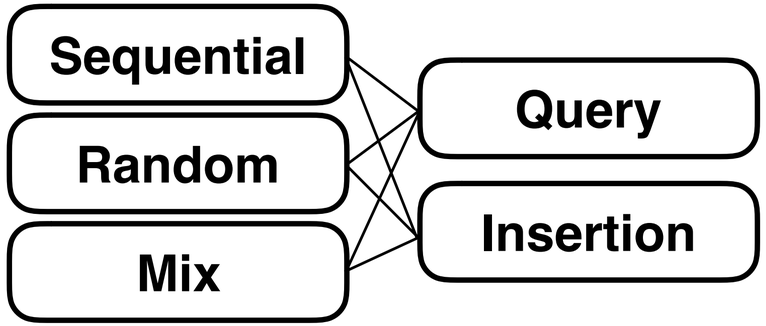
Database Operations:



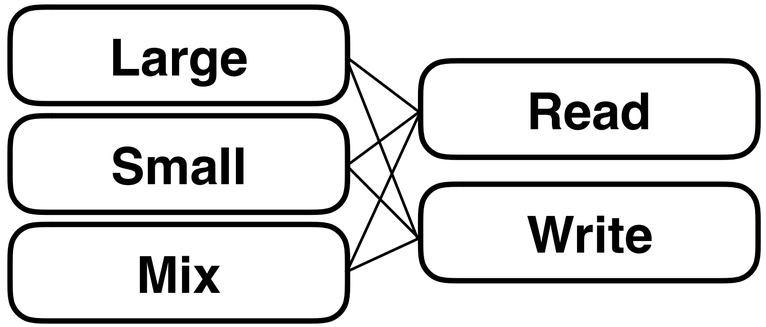
Varmail Operations:

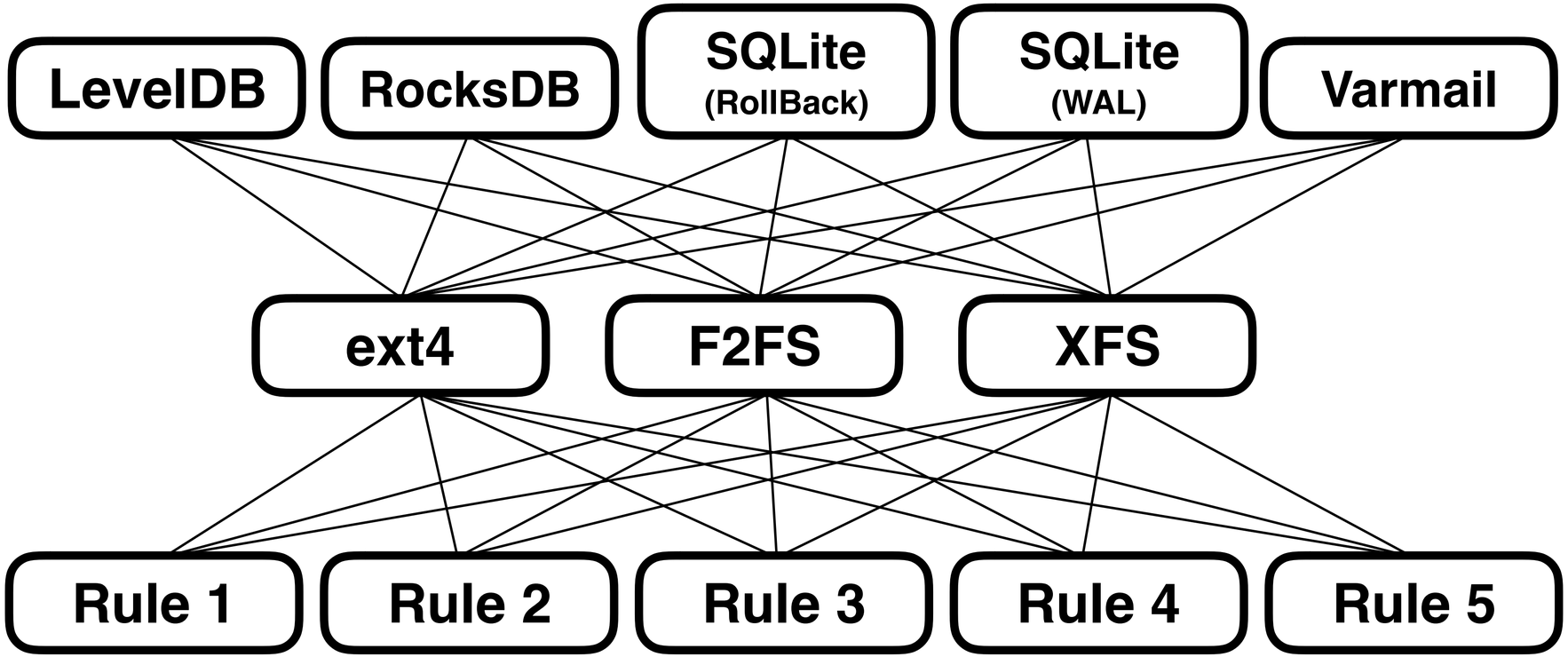


Database Operations:

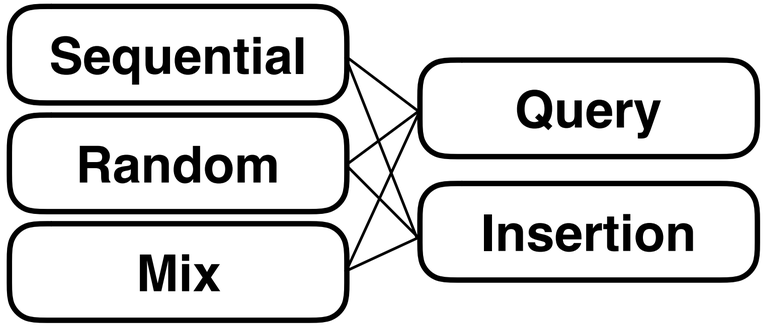


Varmail Operations:

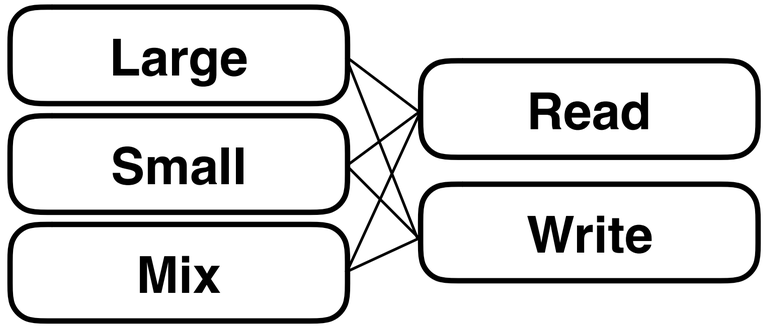


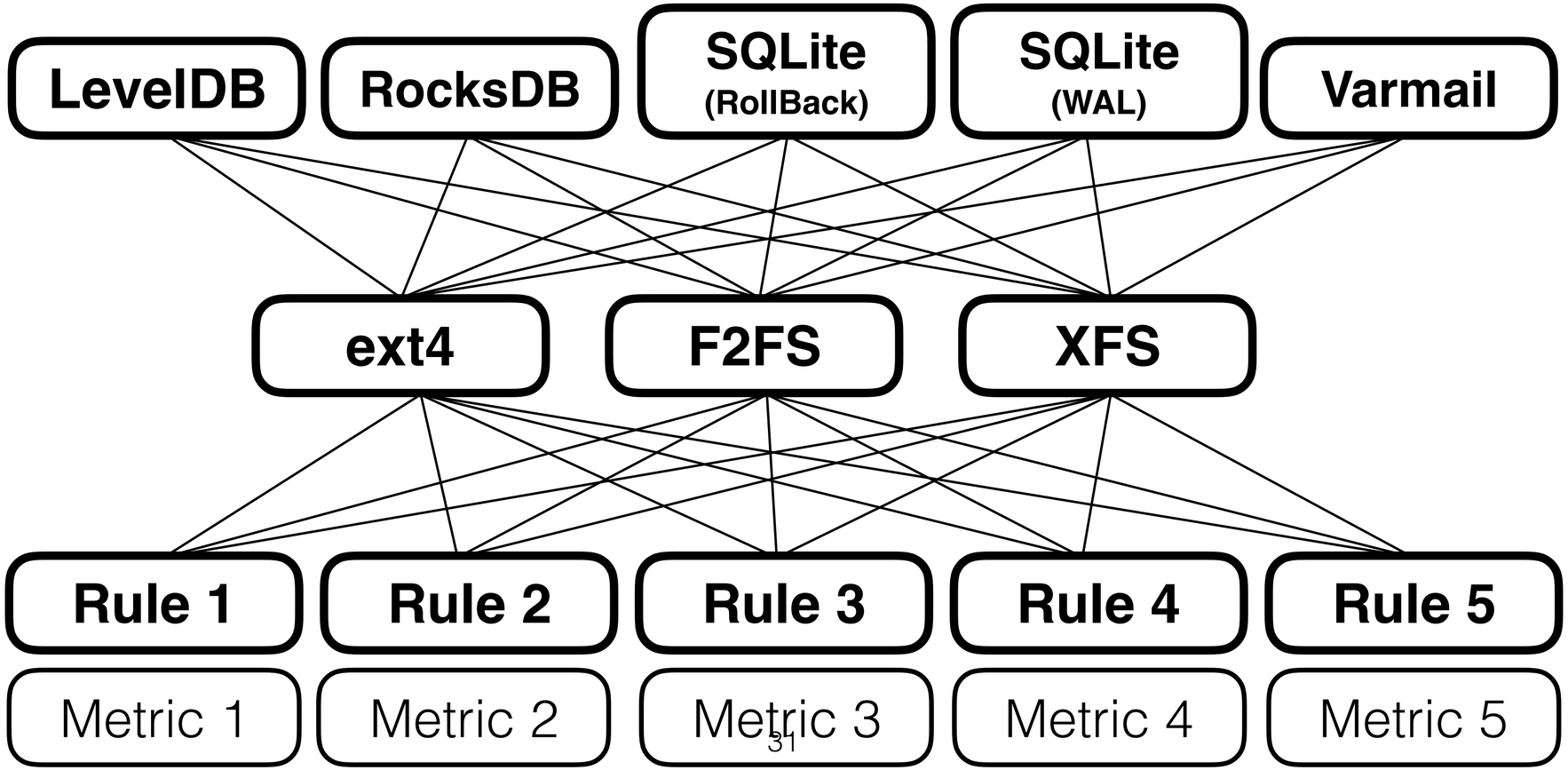


Database Operations:



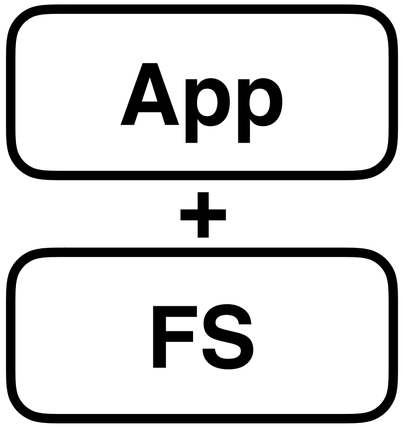
Varmail Operations:



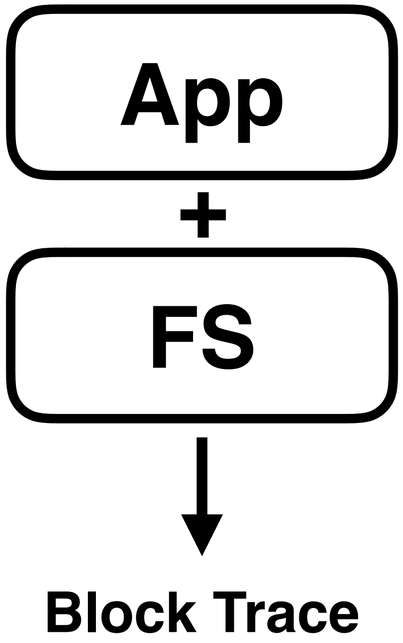


We conduct vertical analysis to find violations of SSD contract.

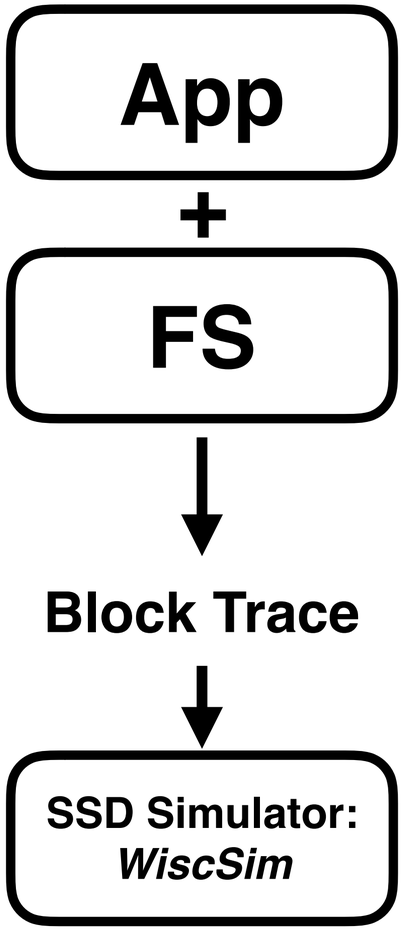
We conduct vertical analysis to find violations of SSD contract.



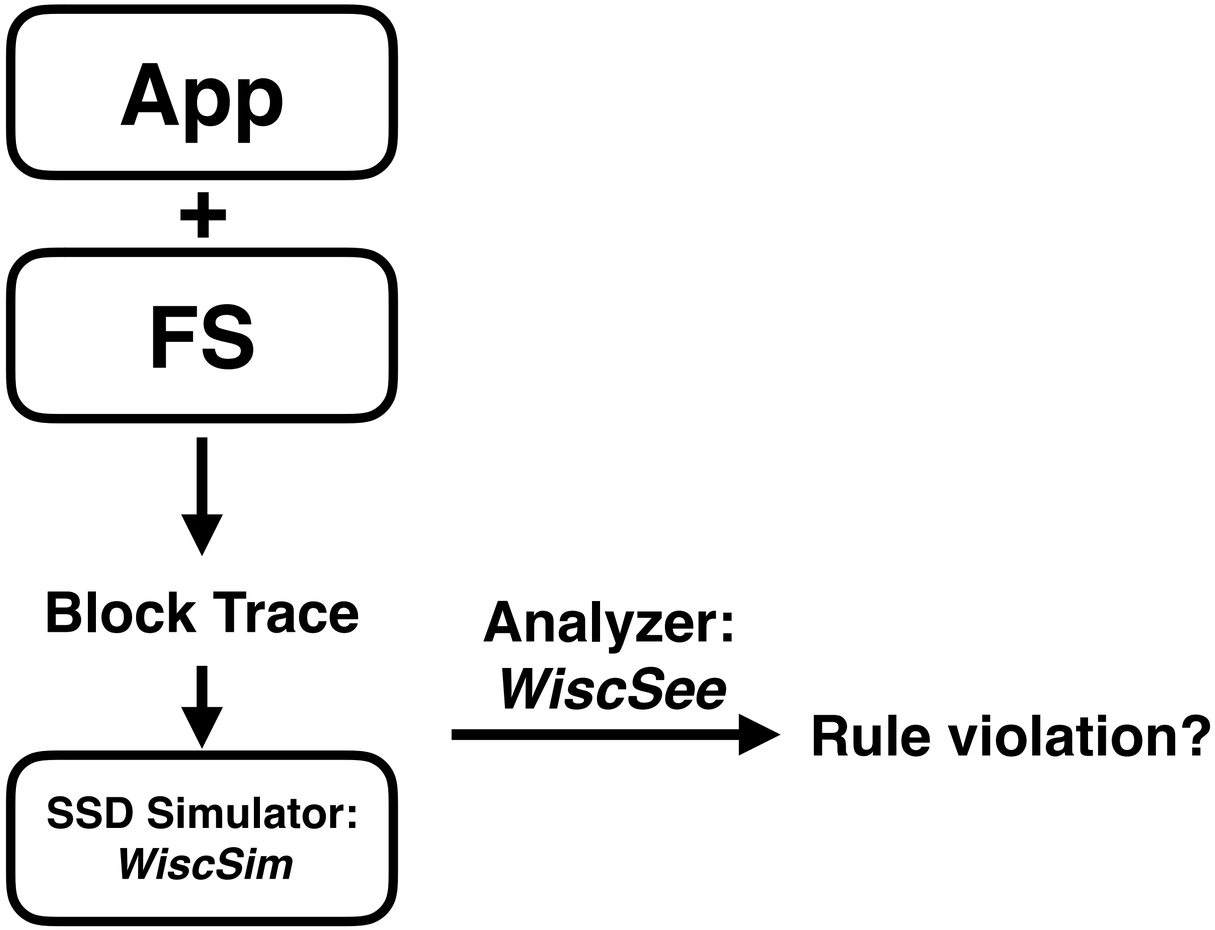
We conduct vertical analysis to find violations of SSD contract.



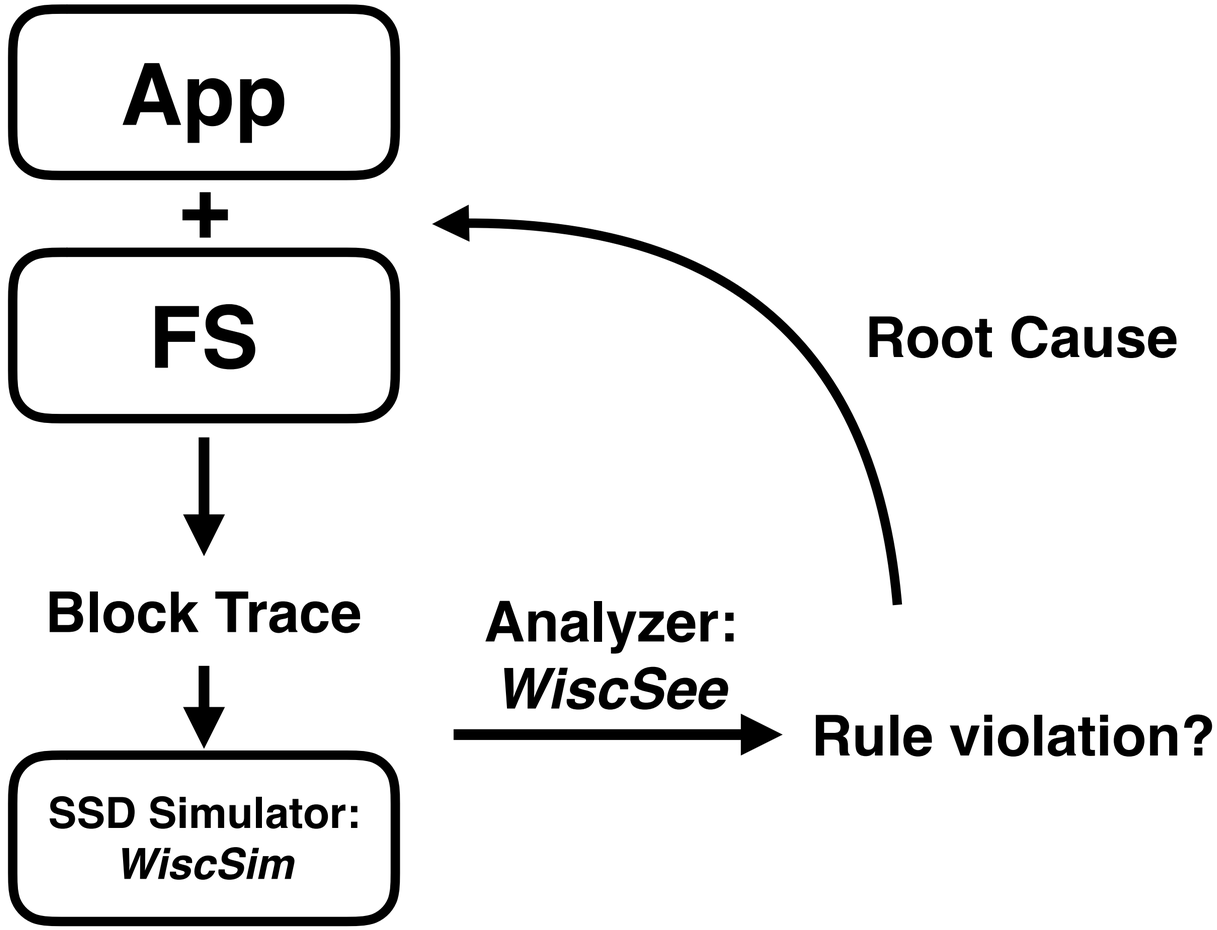
We conduct vertical analysis to find violations of SSD contract.



We conduct vertical analysis to find violations of SSD contract.



We conduct vertical analysis to find violations of SSD contract.



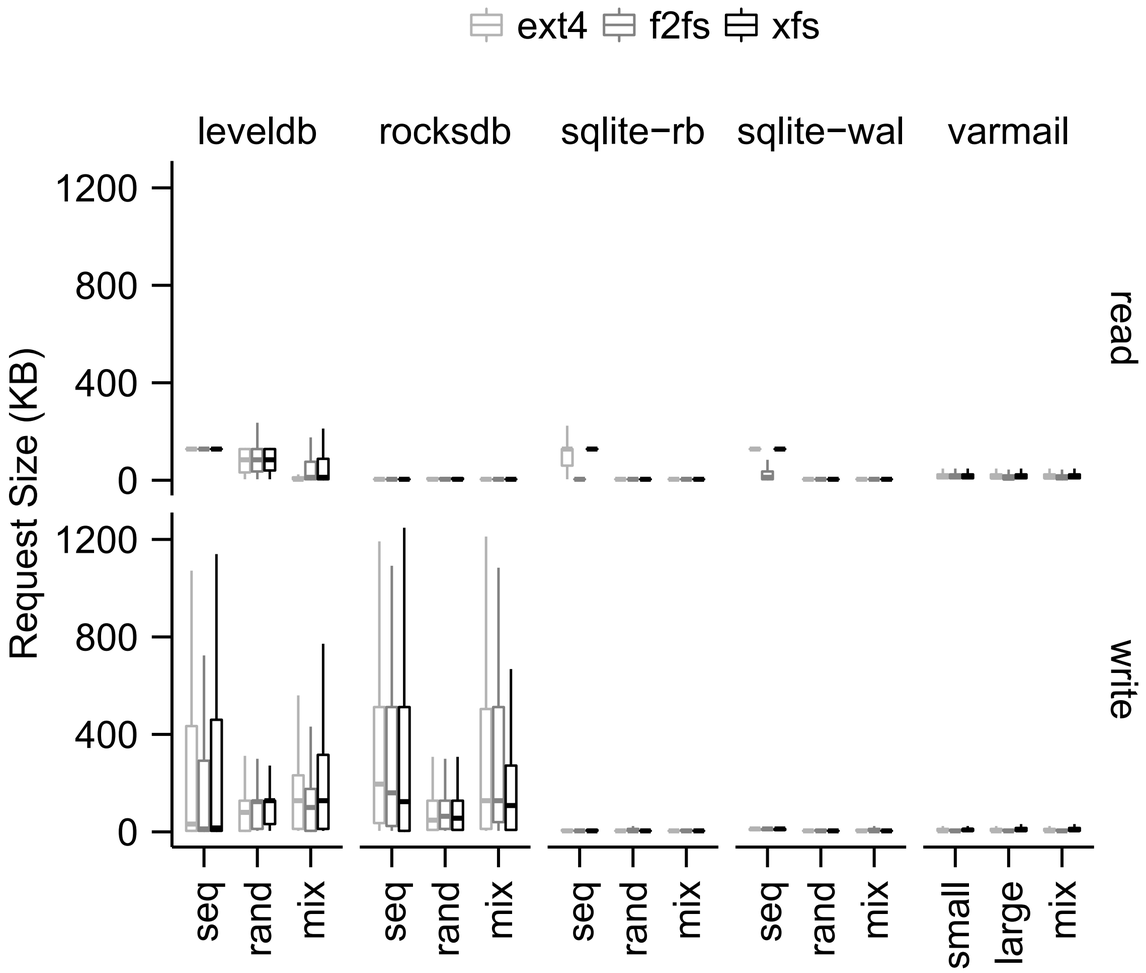
Outline

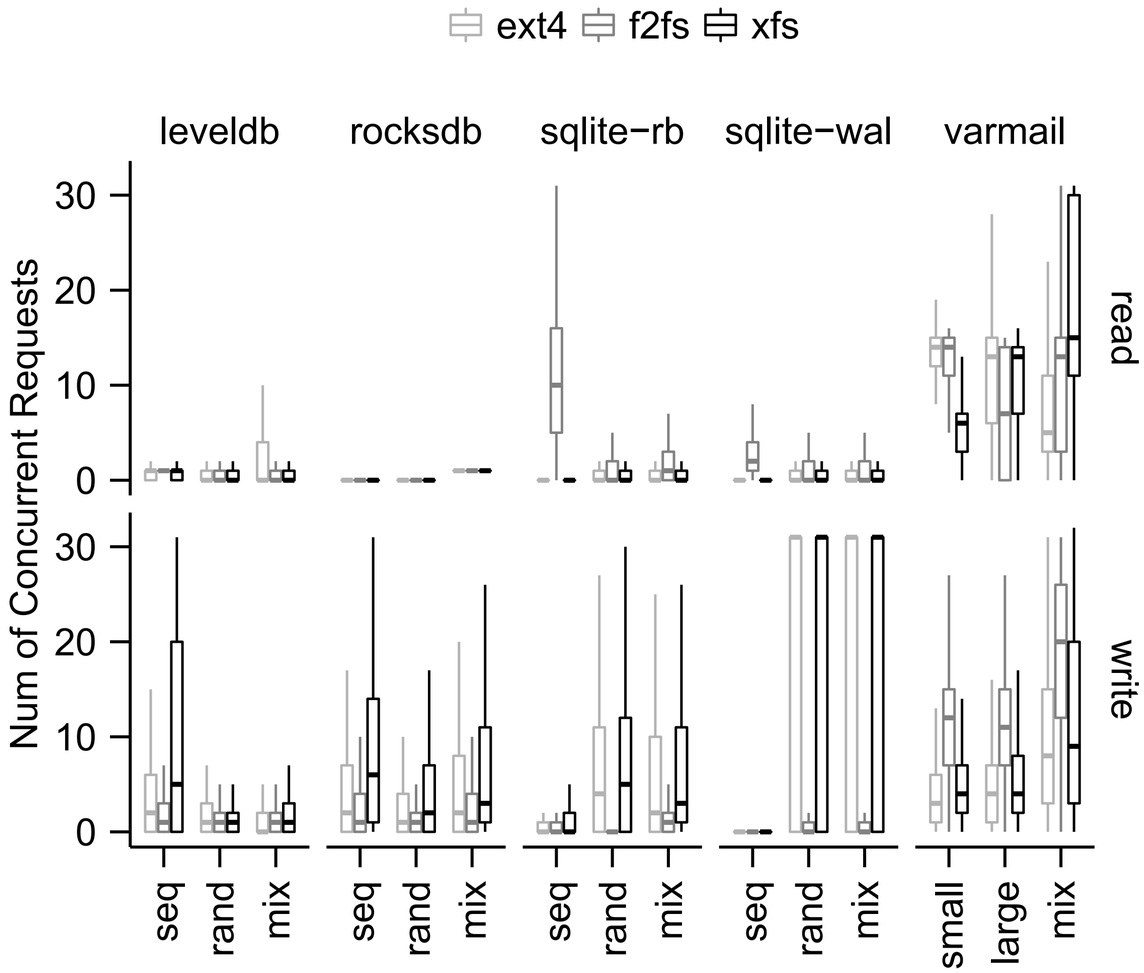
- Overview
- SSD Unwritten Contract
- Violations of Unwritten Contract
  - Method
  - **Observations**
- Lessons Learned
- Conclusions

Observations

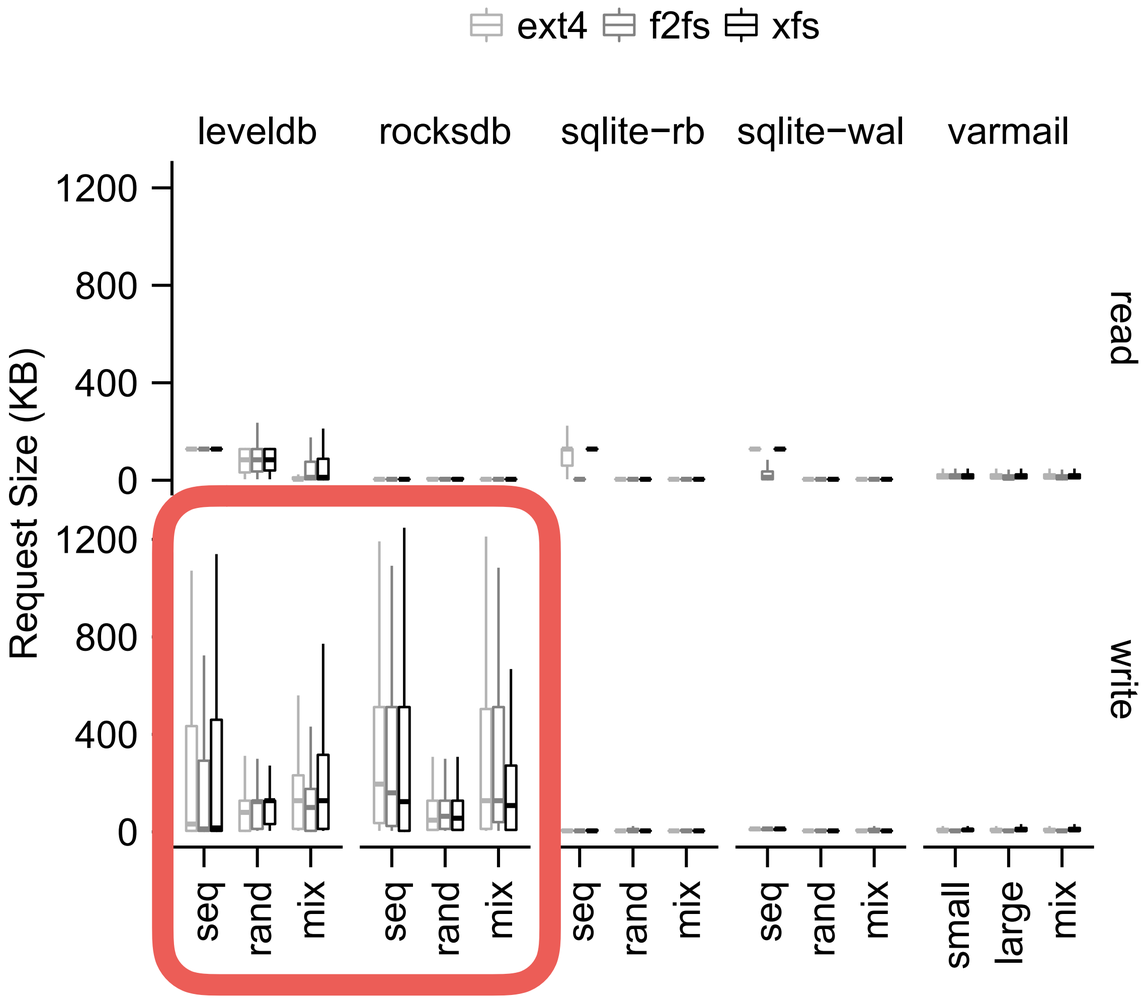
- Rule 1: Request Scale
- Implementation of Linux limits performance
- Rule 3: Grouping by Death Time
  - F2FS incurs more GC overhead than ext4/XFS
  - Application log structuring does not reduce GC

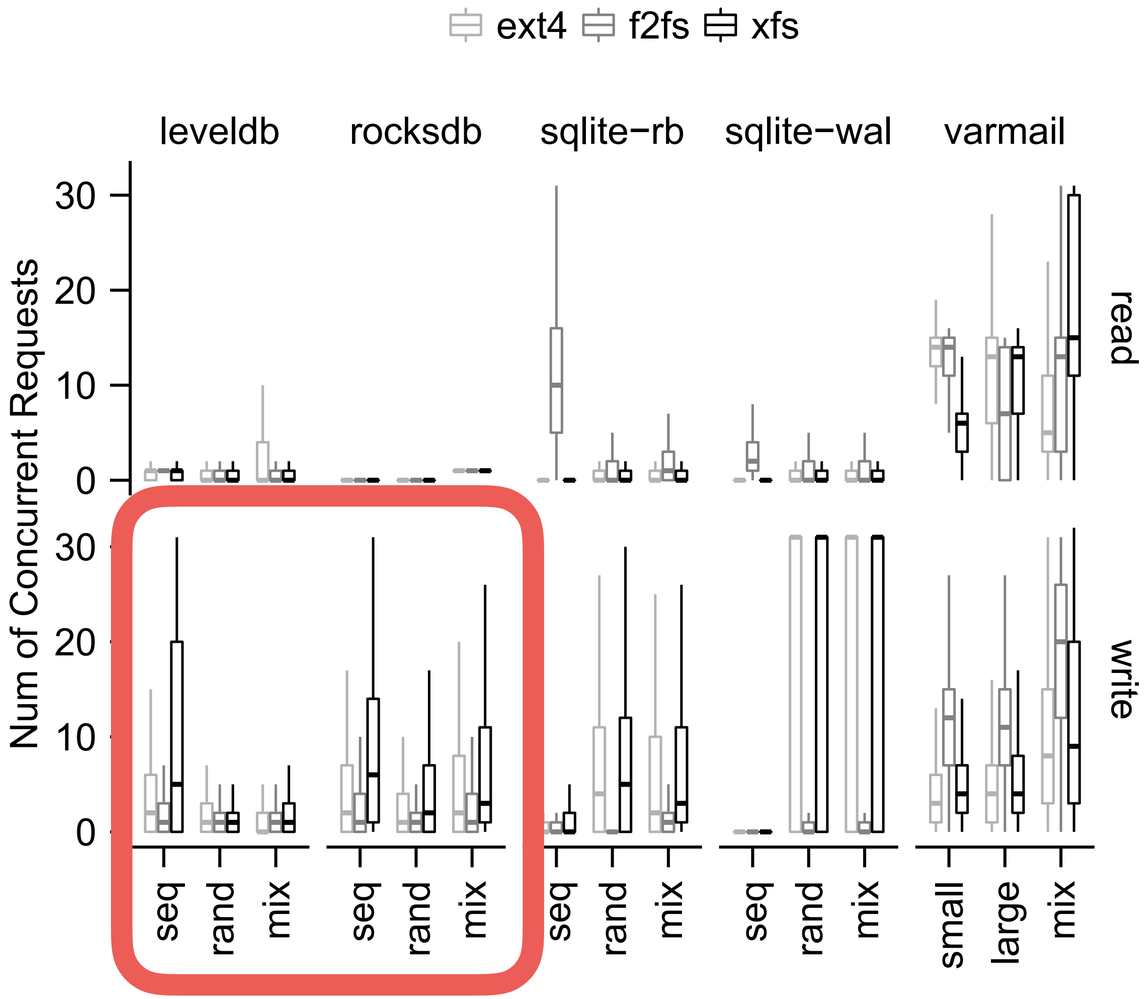
We evaluate request scale by request size and number of concurrent requests



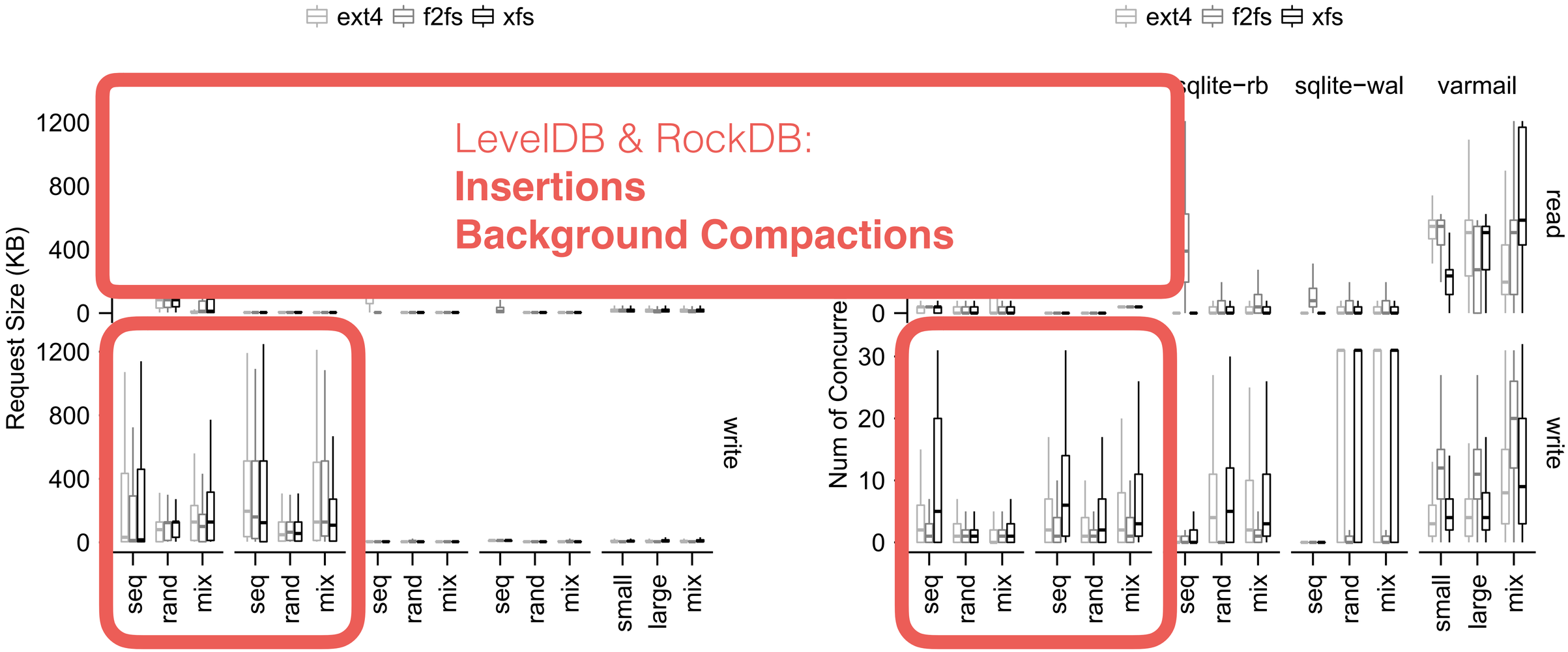


We evaluate request scale by request size and number of concurrent requests

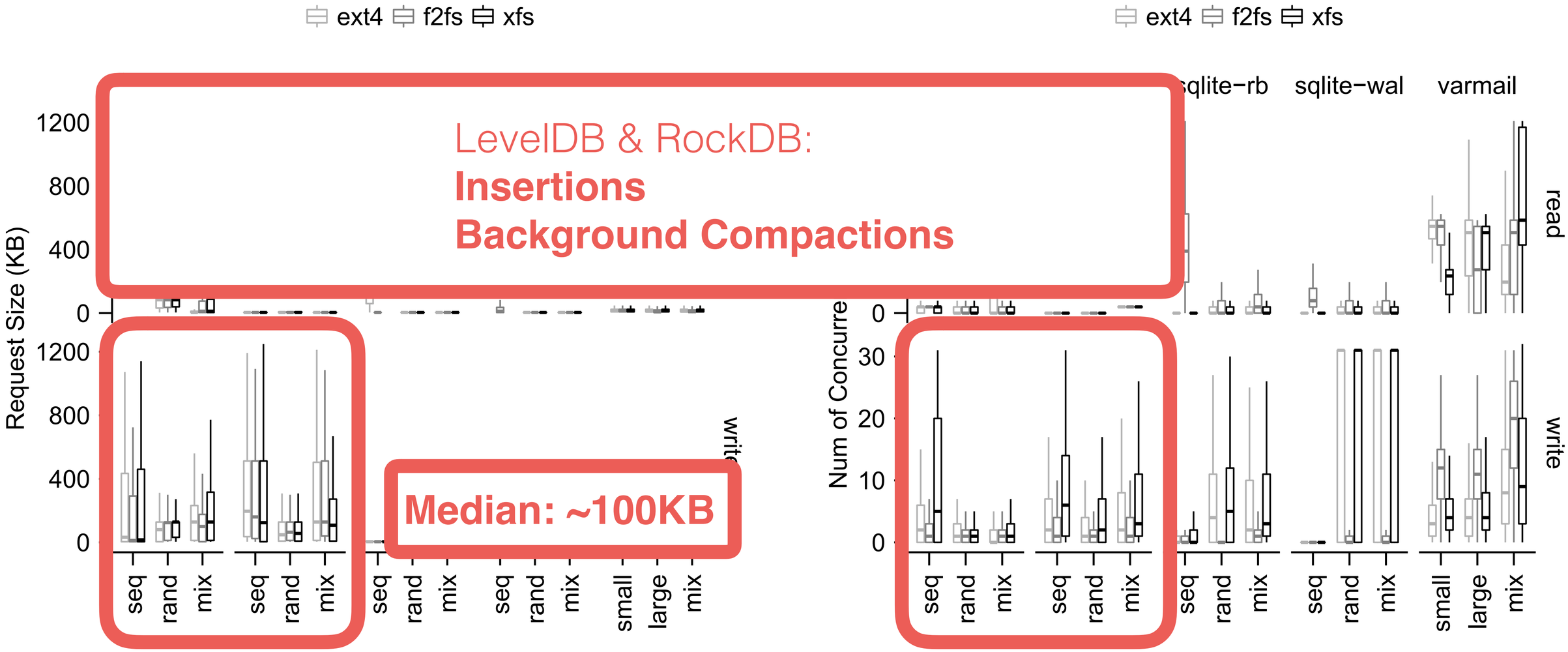




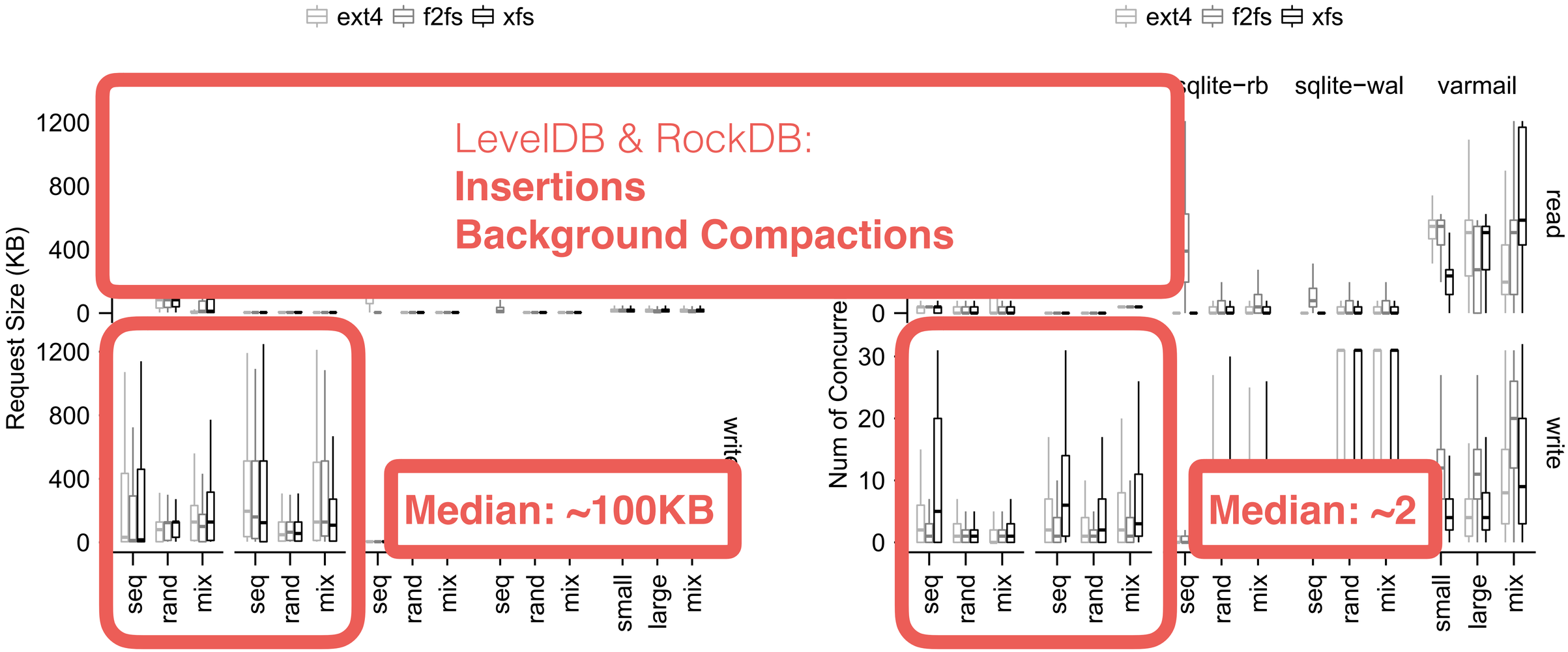
We evaluate request scale by request size and number of concurrent requests



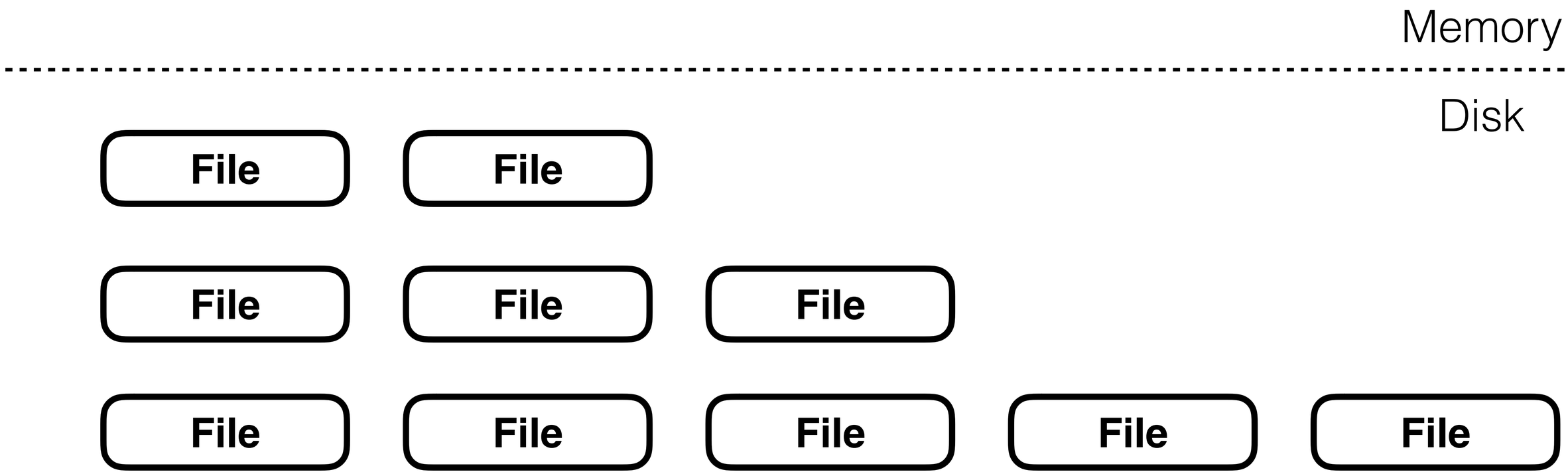
We evaluate request scale by request size and number of concurrent requests



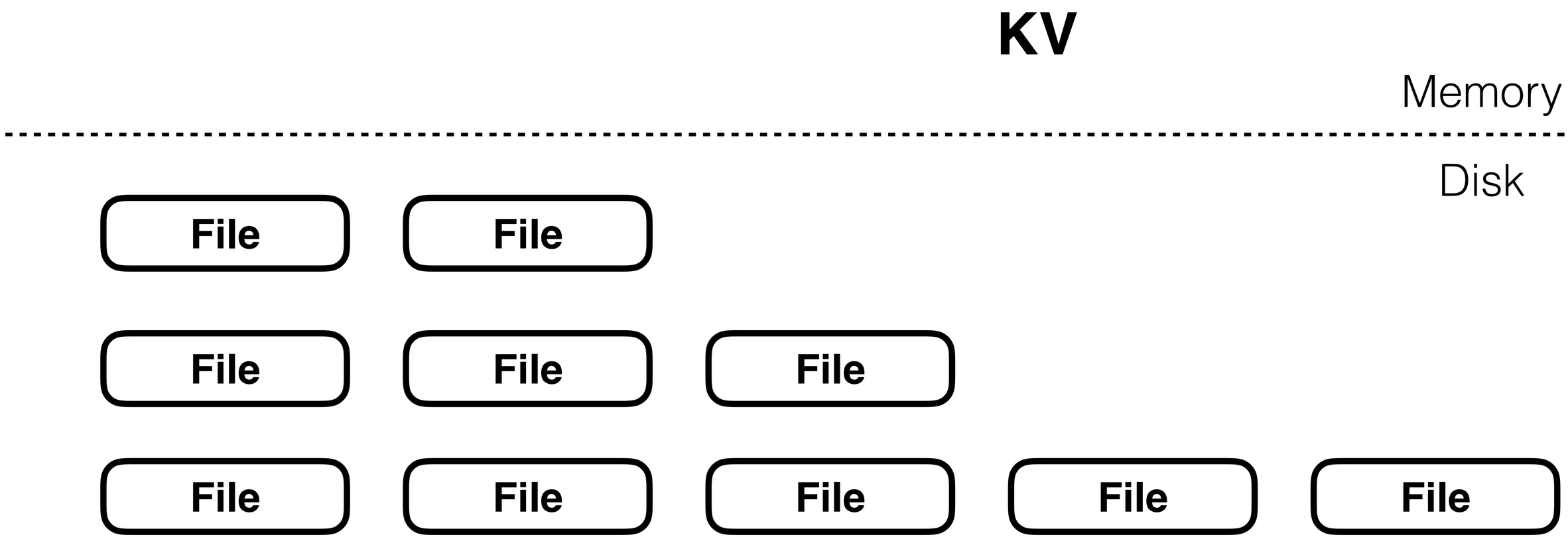
We evaluate request scale by request size and number of concurrent requests



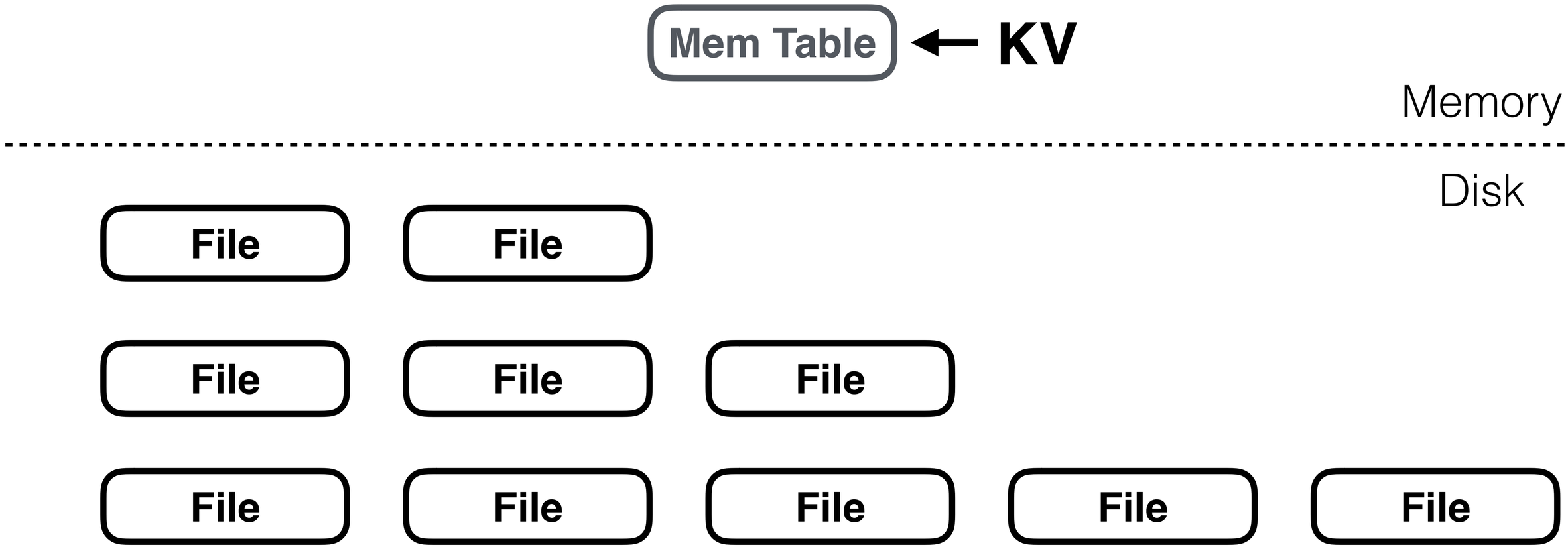
LevelDB & RocksDB Background



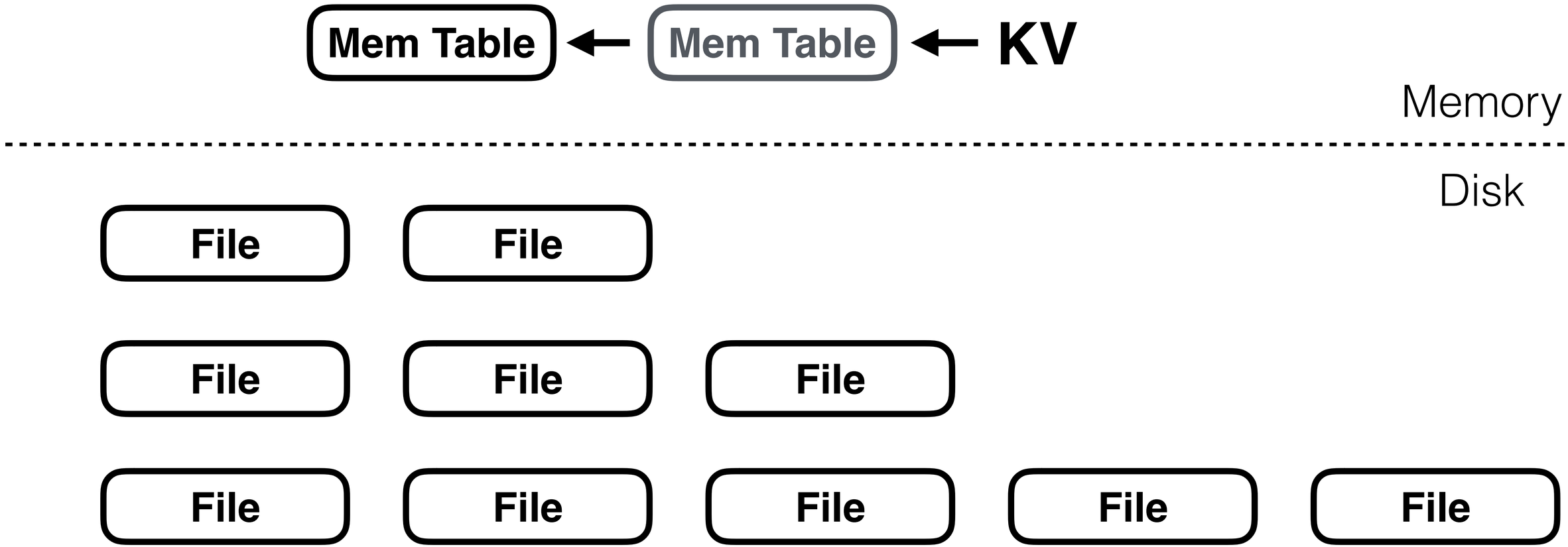
LevelDB & RocksDB Background



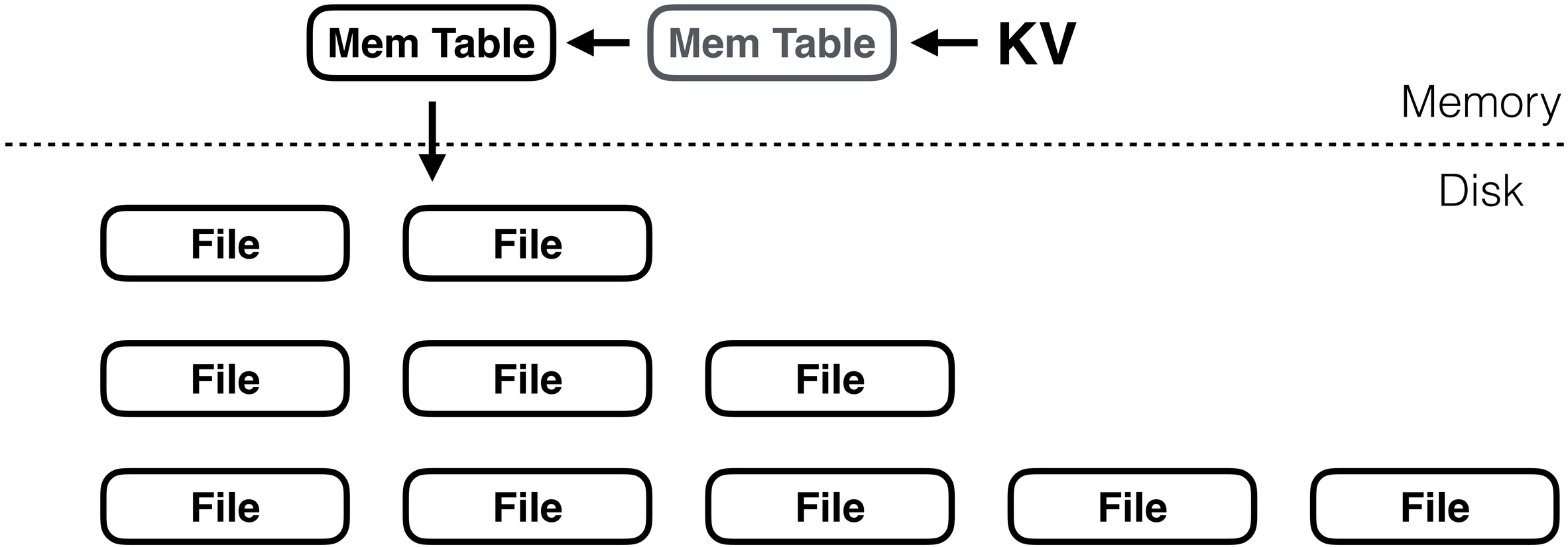
LevelDB & RocksDB Background



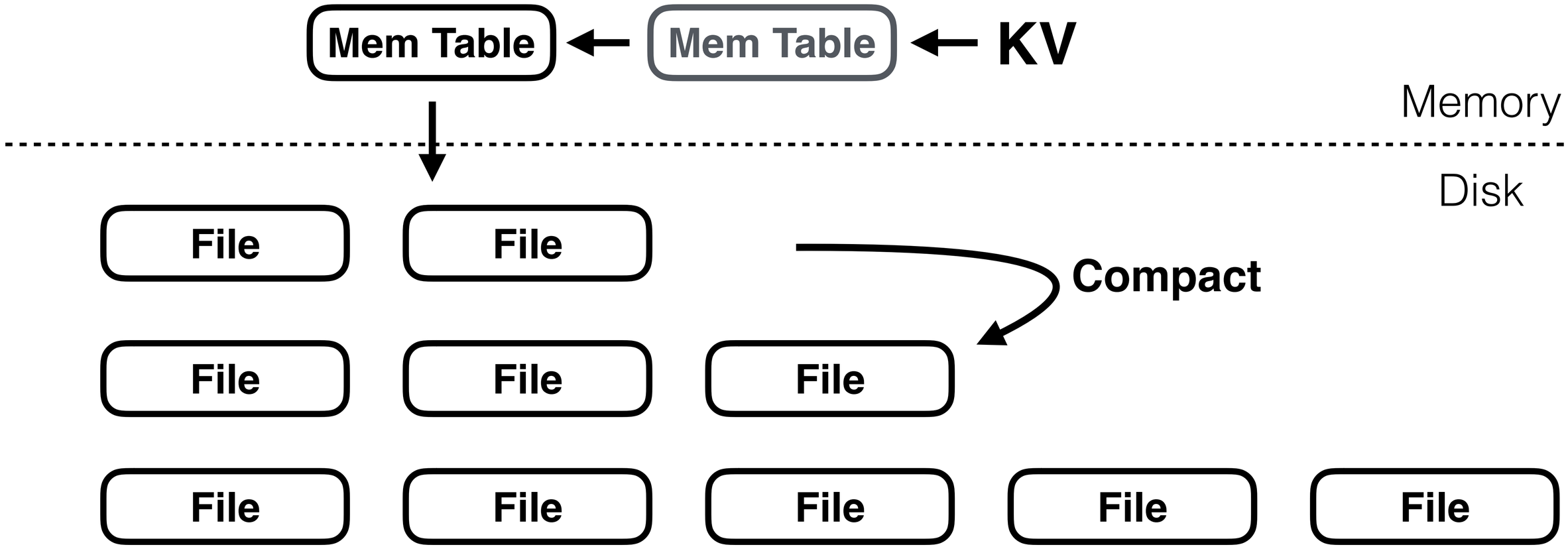
LevelDB & RocksDB Background



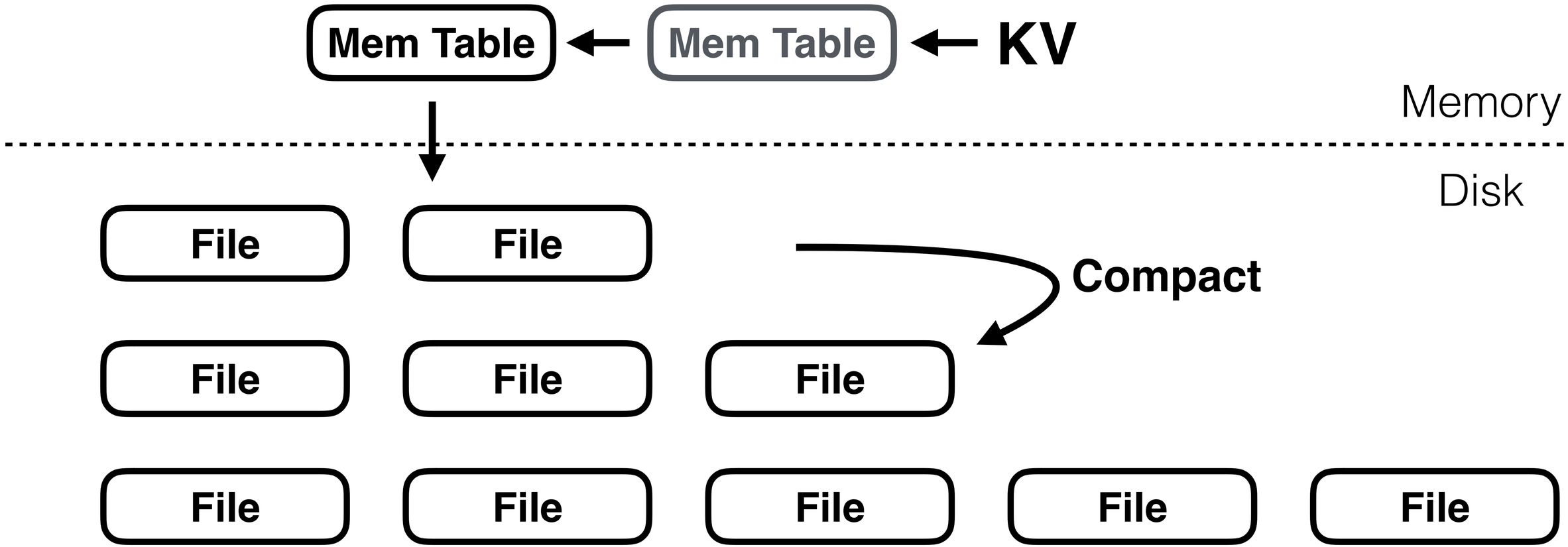
LevelDB & RocksDB Background



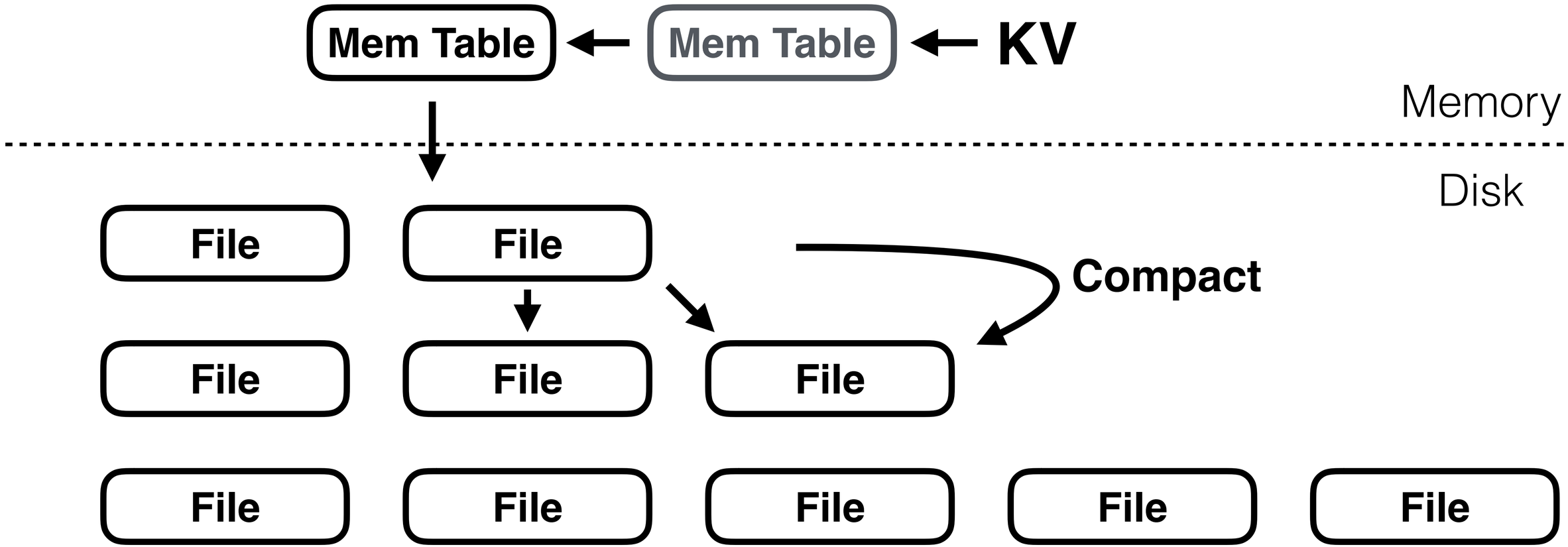
LevelDB & RocksDB Background



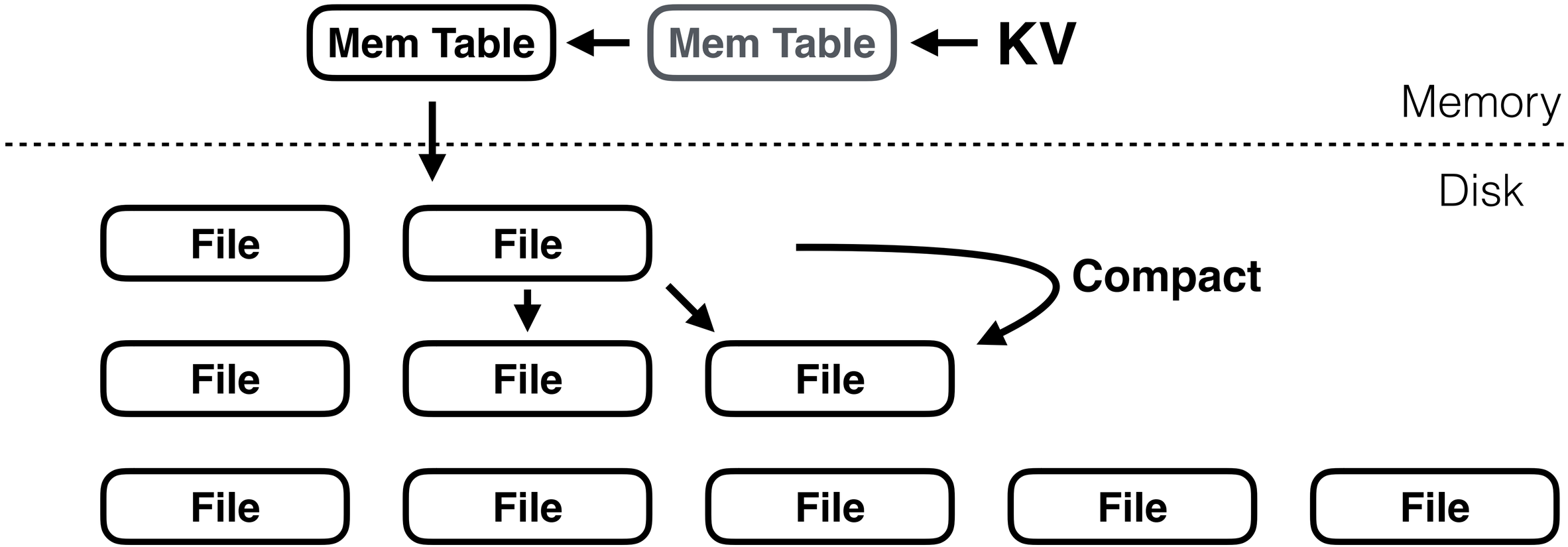
LevelDB & RocksDB Background



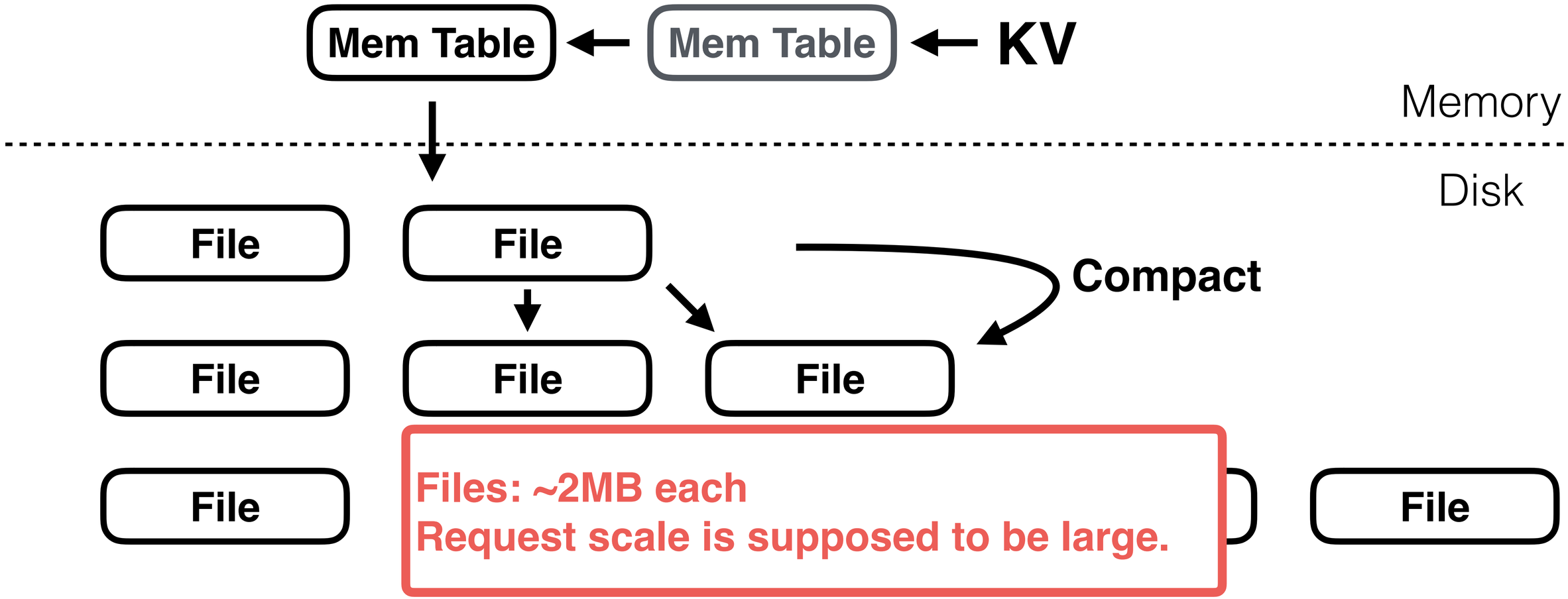
LevelDB & RocksDB Background



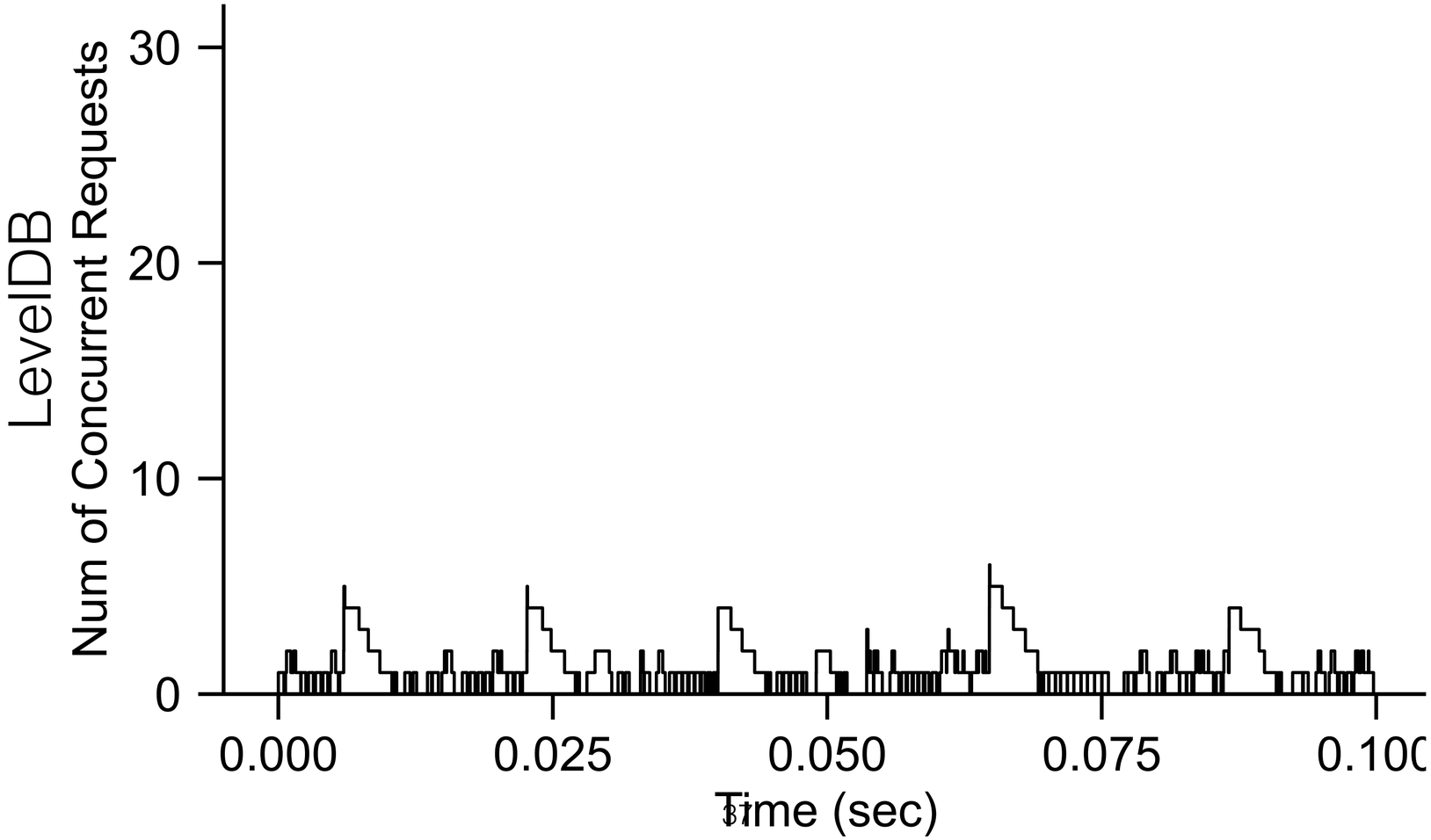
LevelDB & RocksDB Background



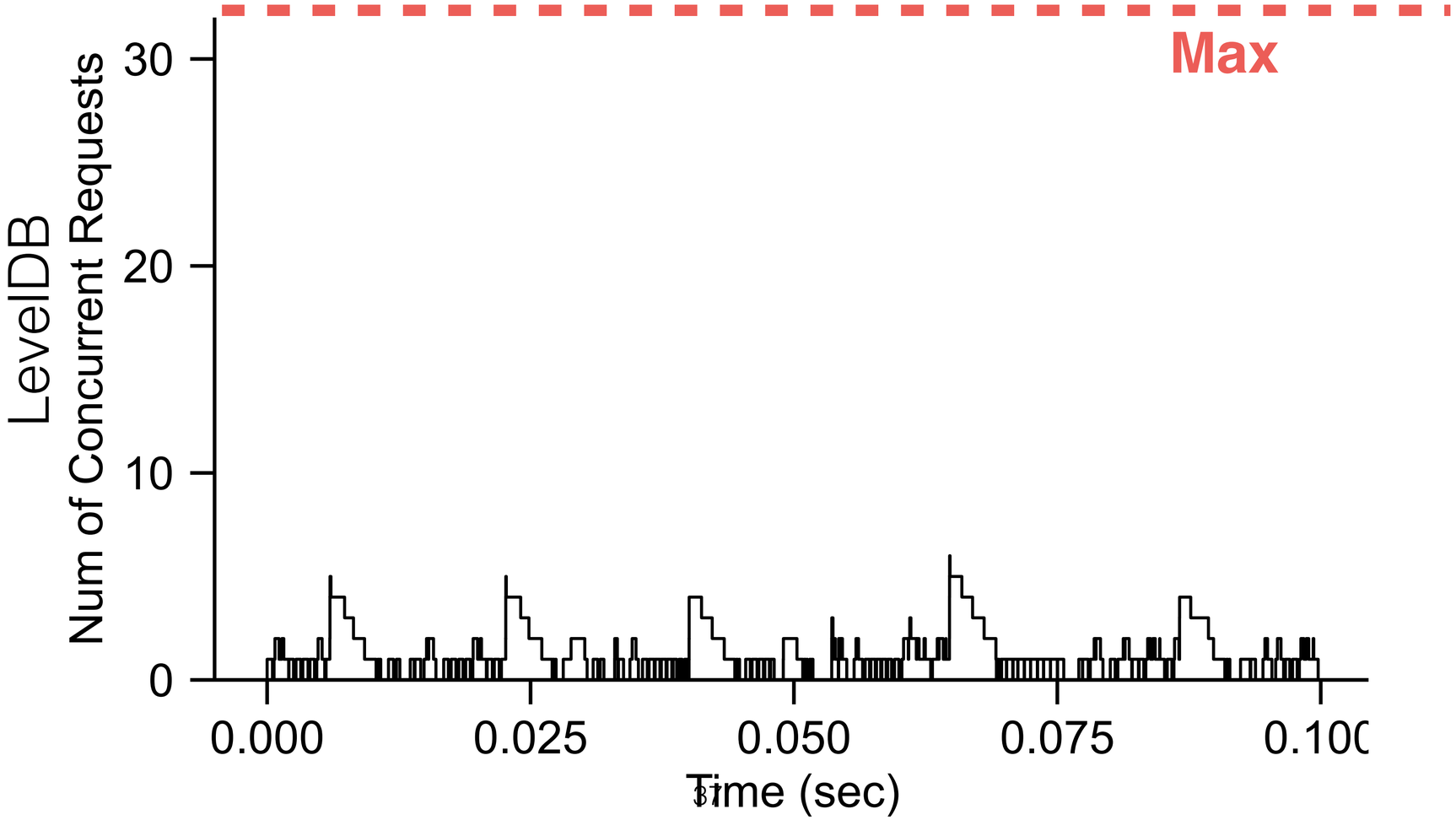
LevelDB & RocksDB Background



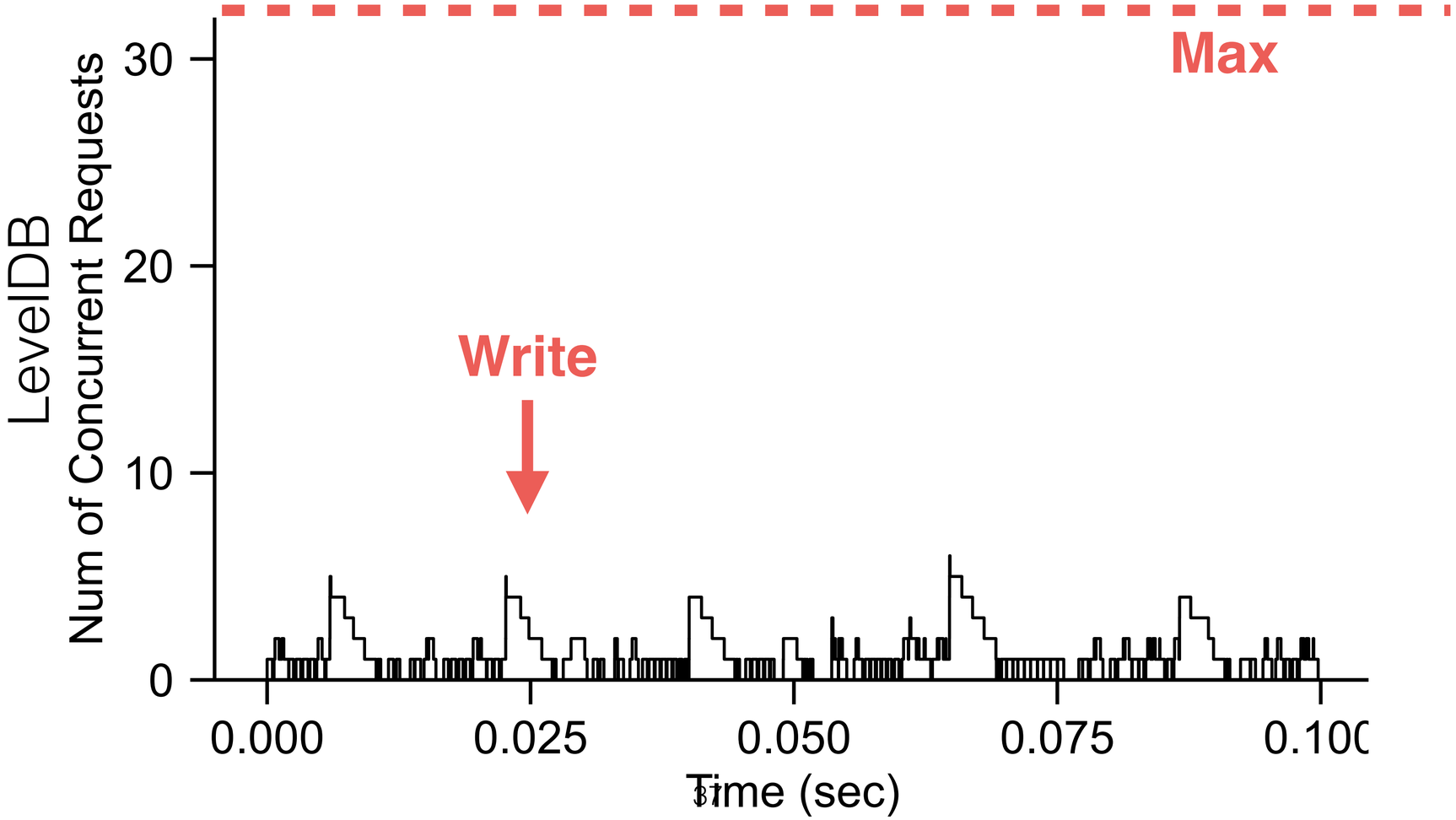
Num of concurrent requests is low during compaction



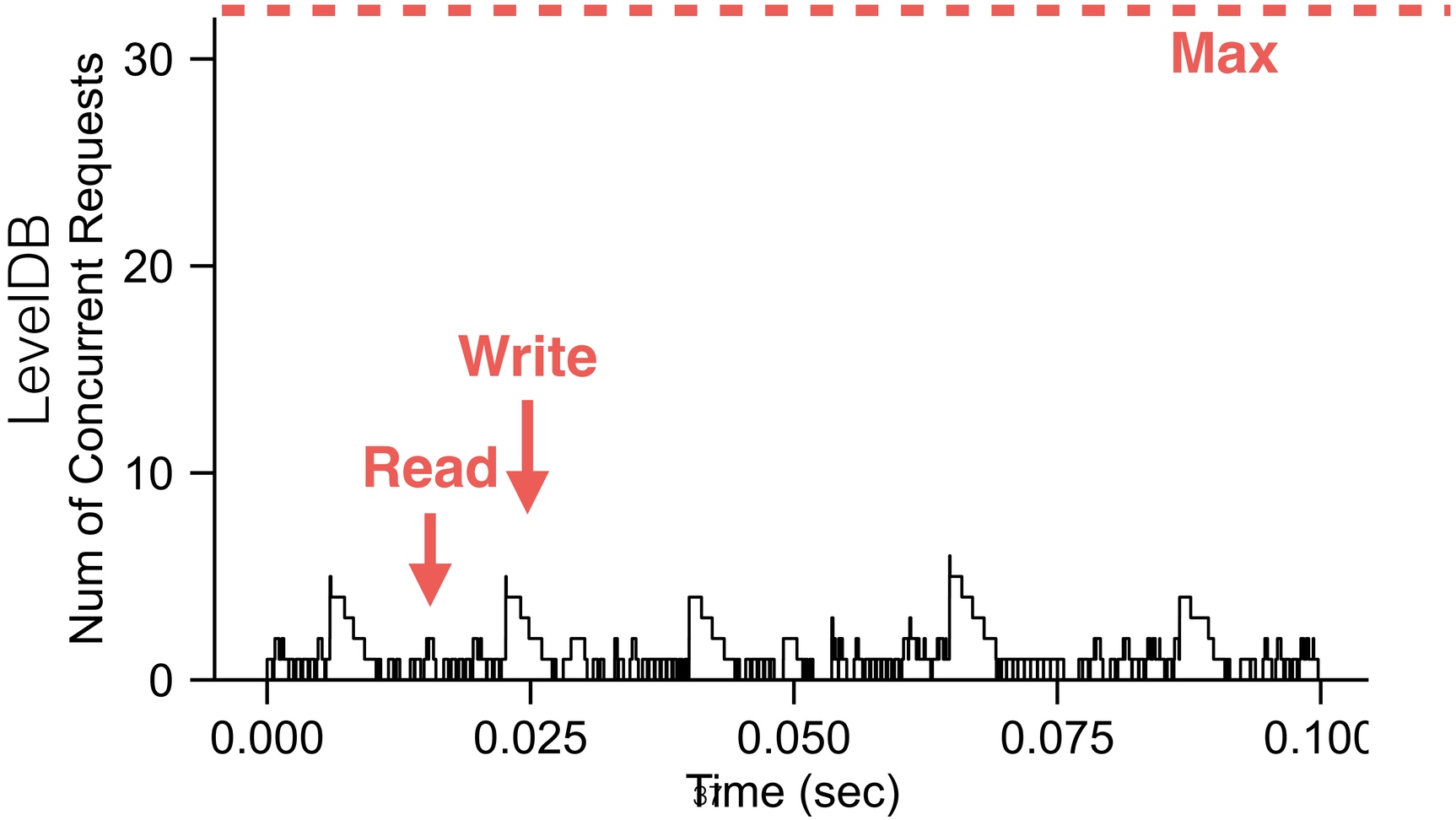
Num of concurrent requests is low during compaction



Num of concurrent requests is low during compaction



Num of concurrent requests is low during compaction



Page cache implementation splits and serializes user requests

App

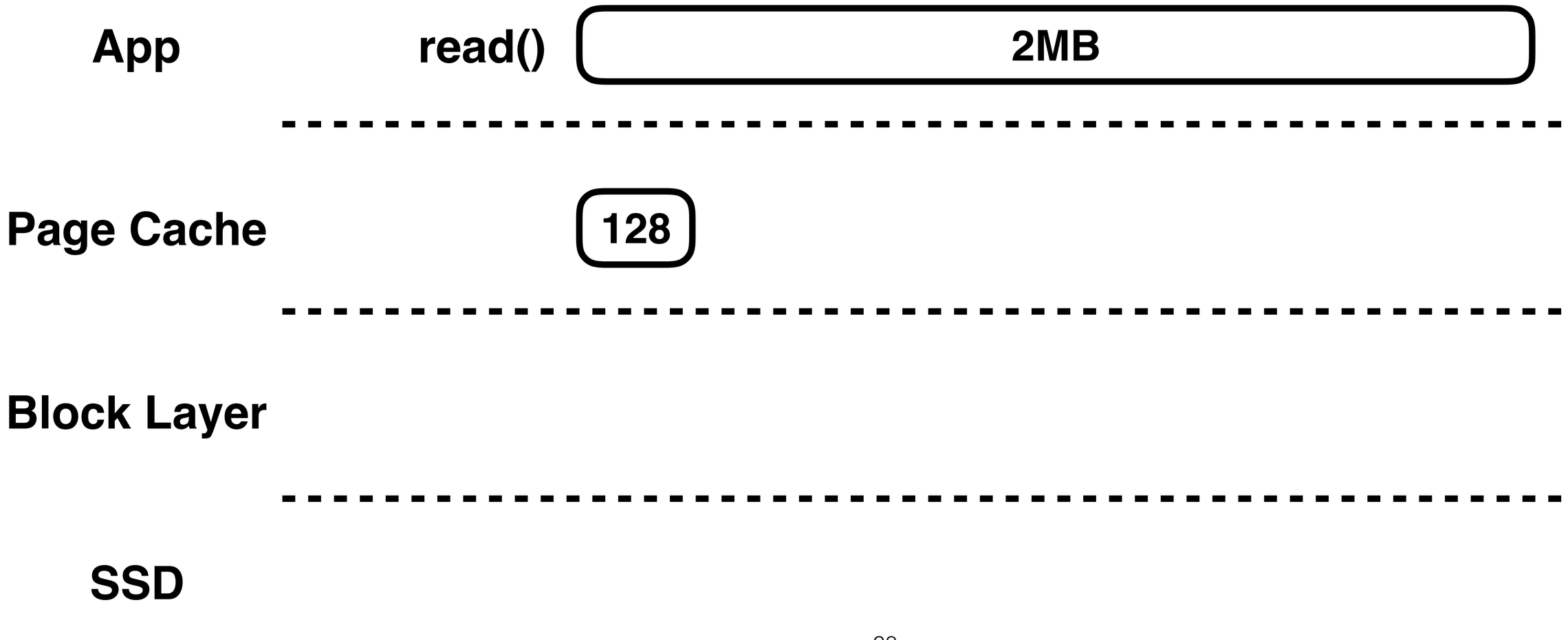
Page Cache

Block Layer

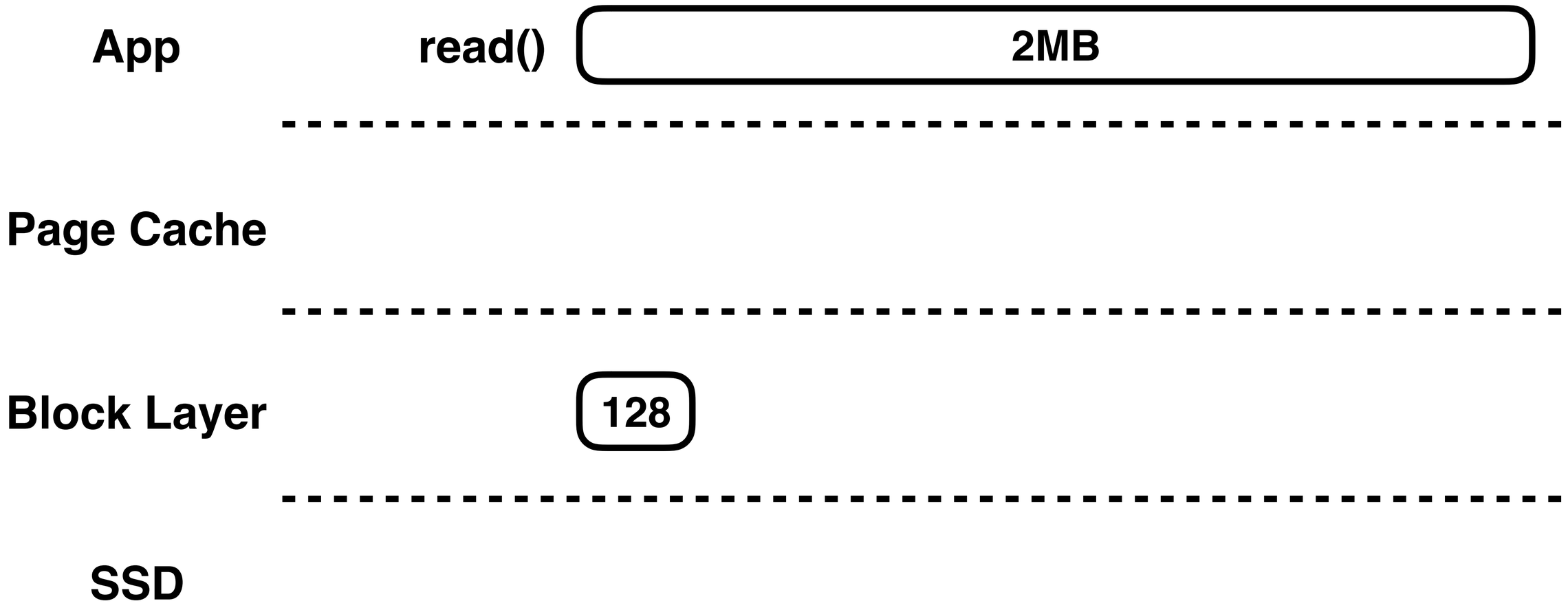
SSD

Page cache implementation splits and serializes user requests

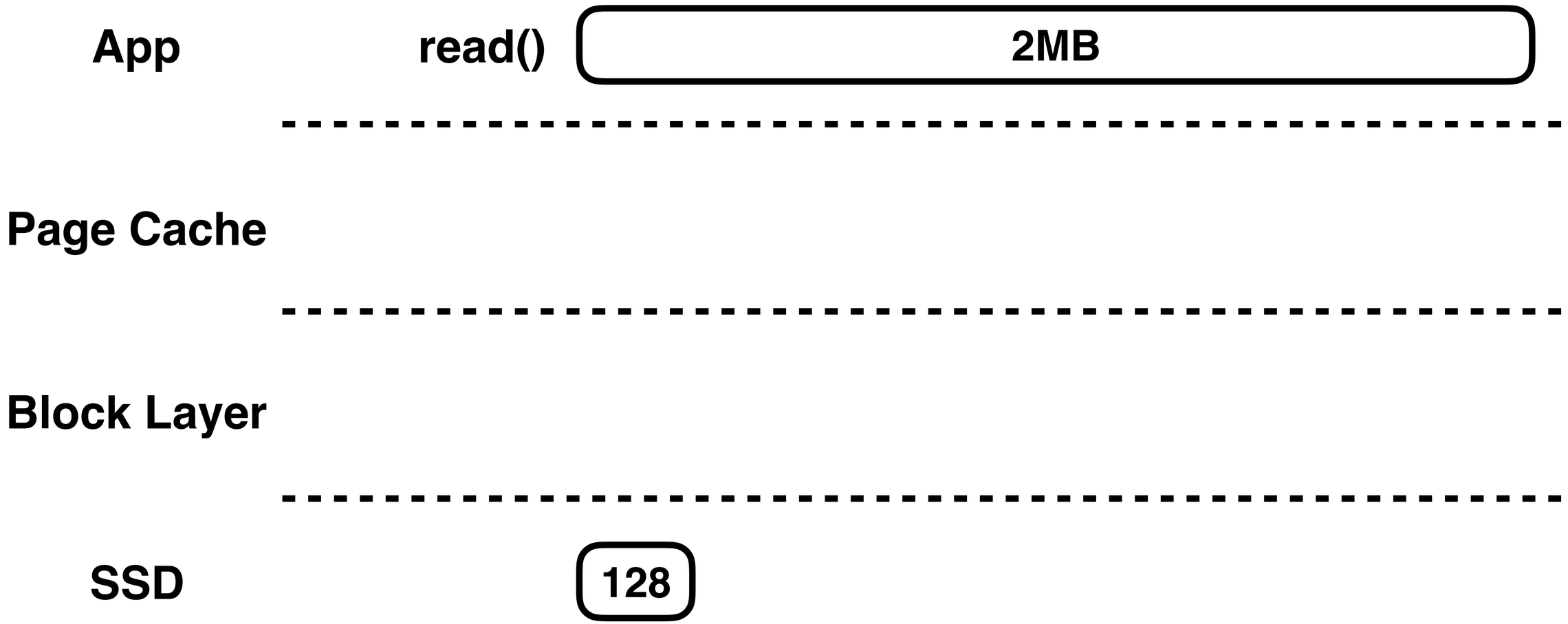
Page cache implementation splits and serializes user requests



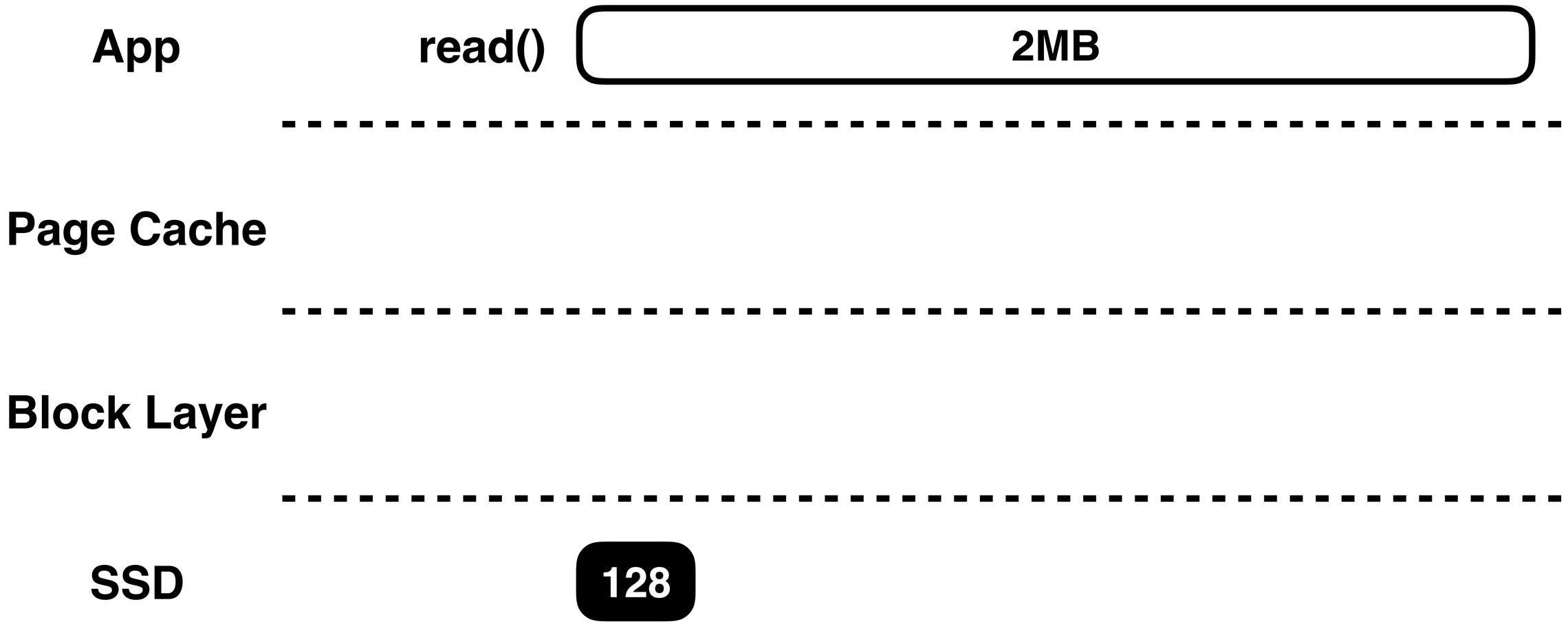
Page cache implementation splits and serializes user requests



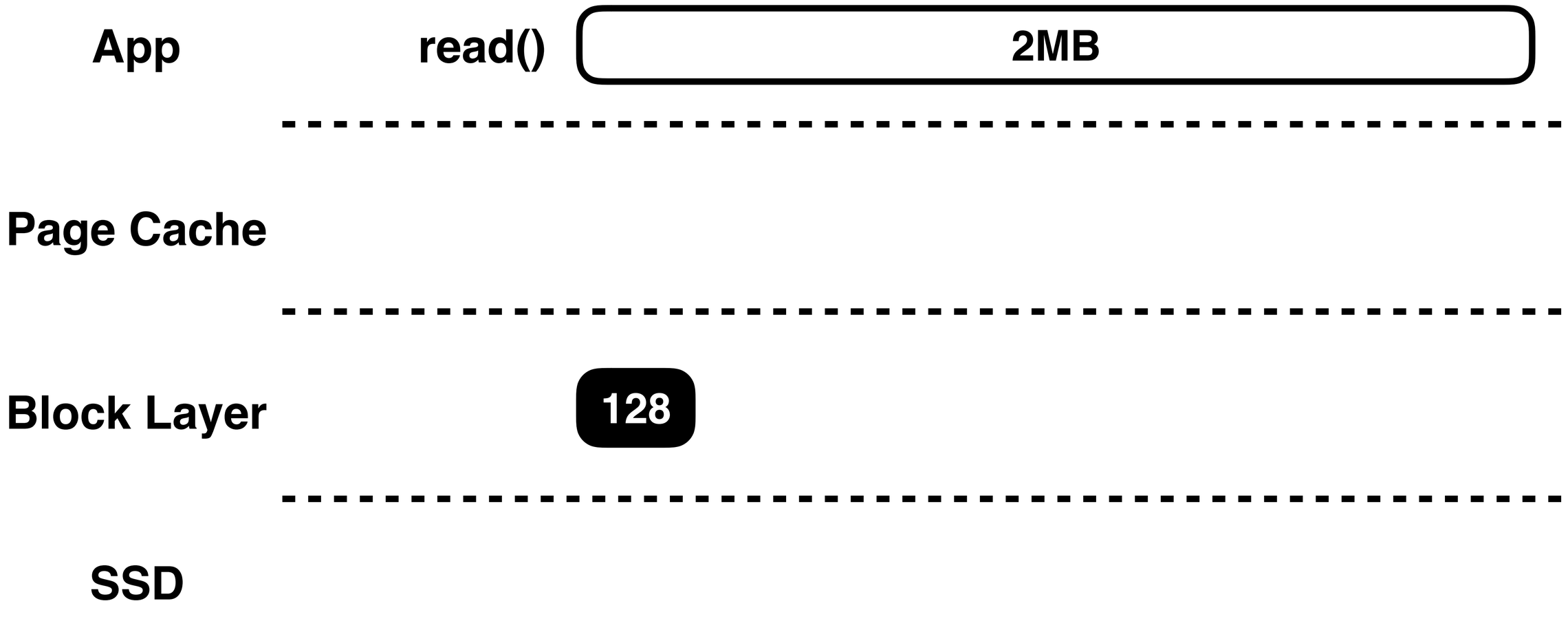
Page cache implementation splits and serializes user requests



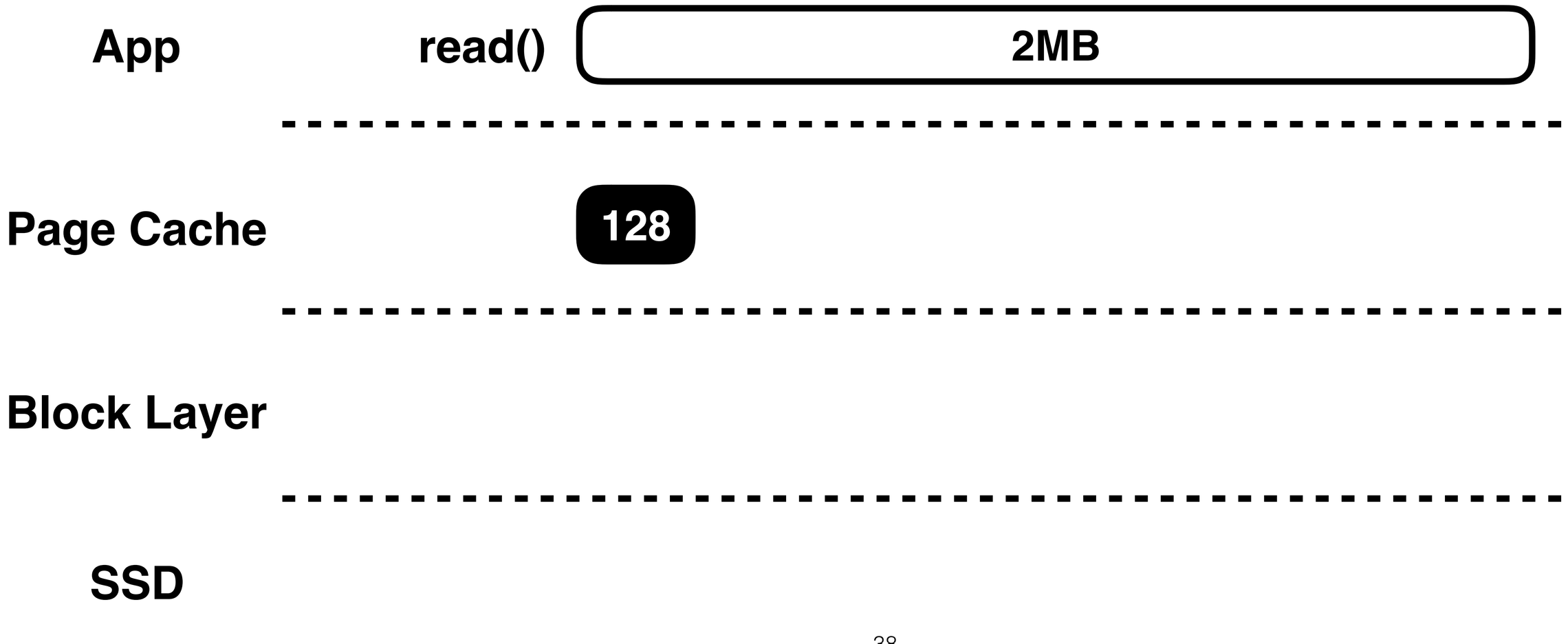
Page cache implementation splits and serializes user requests



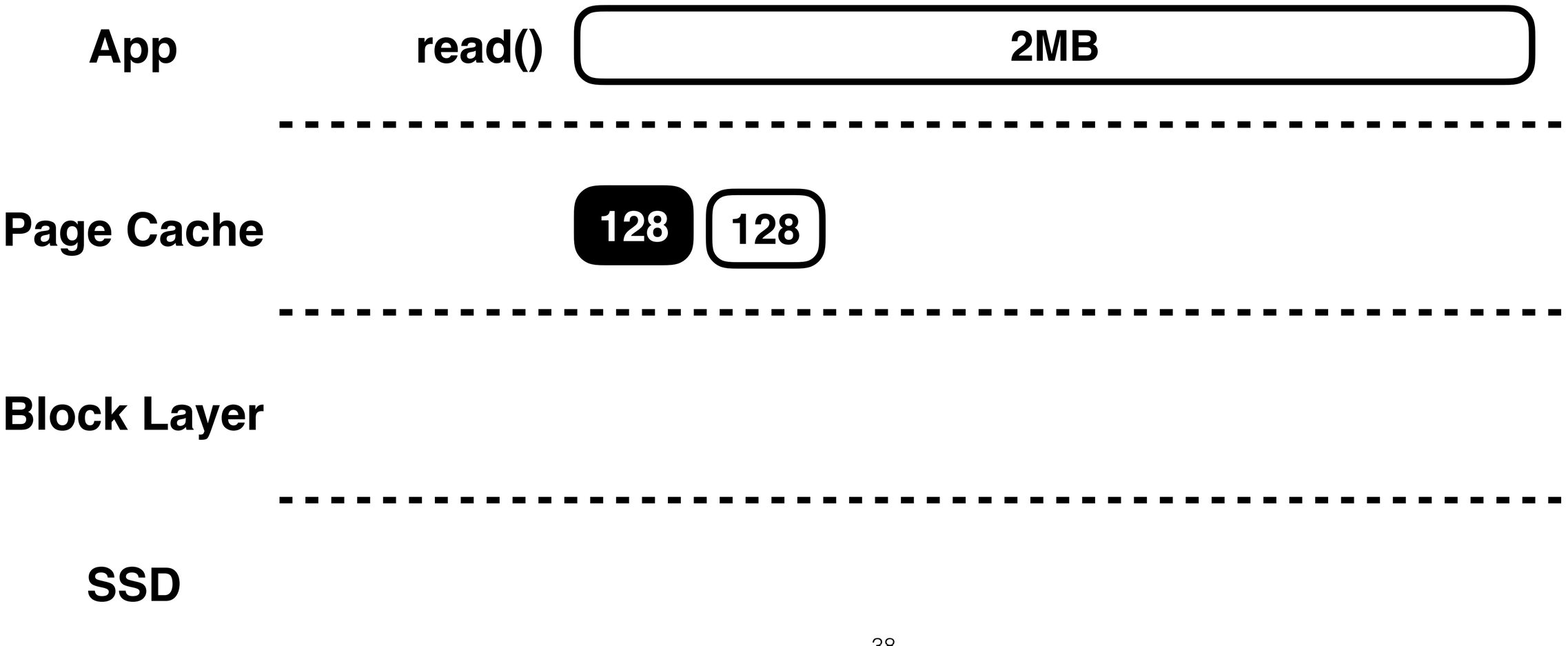
Page cache implementation splits and serializes user requests



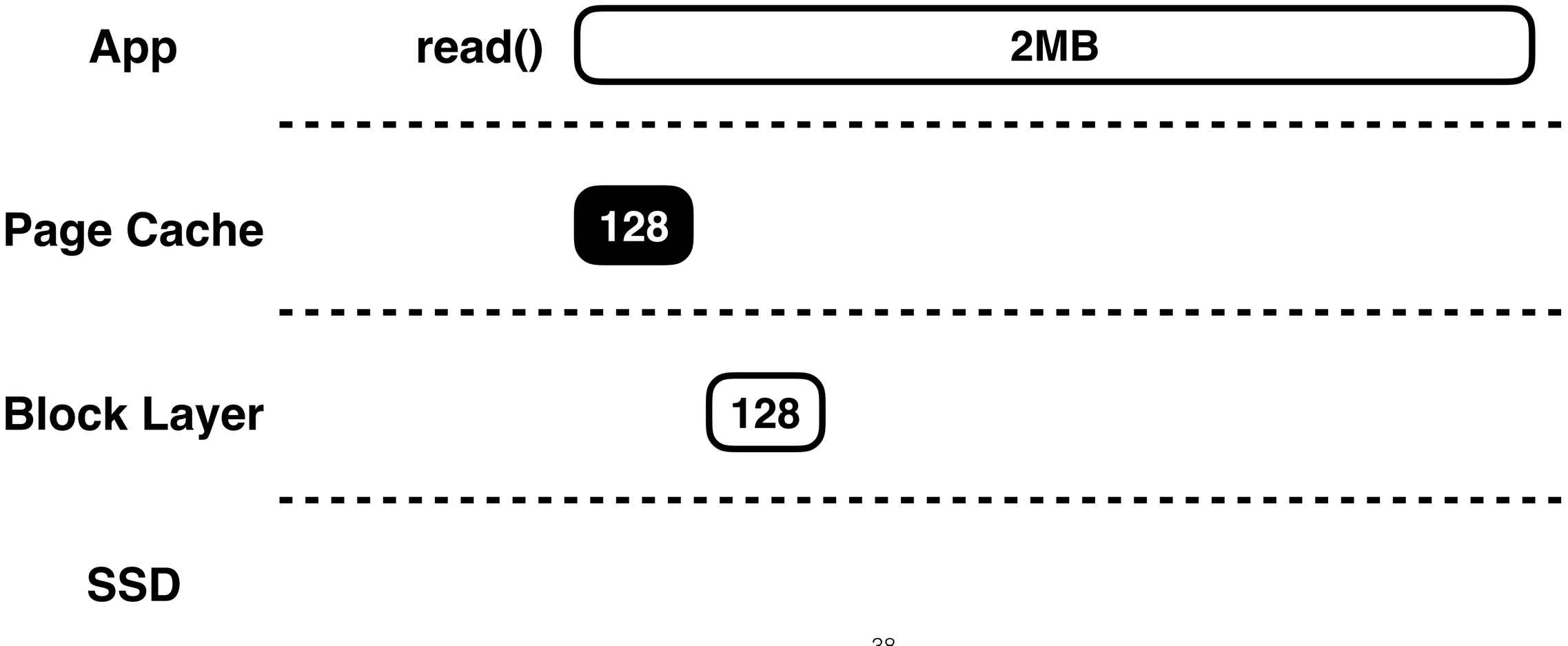
Page cache implementation splits and serializes user requests



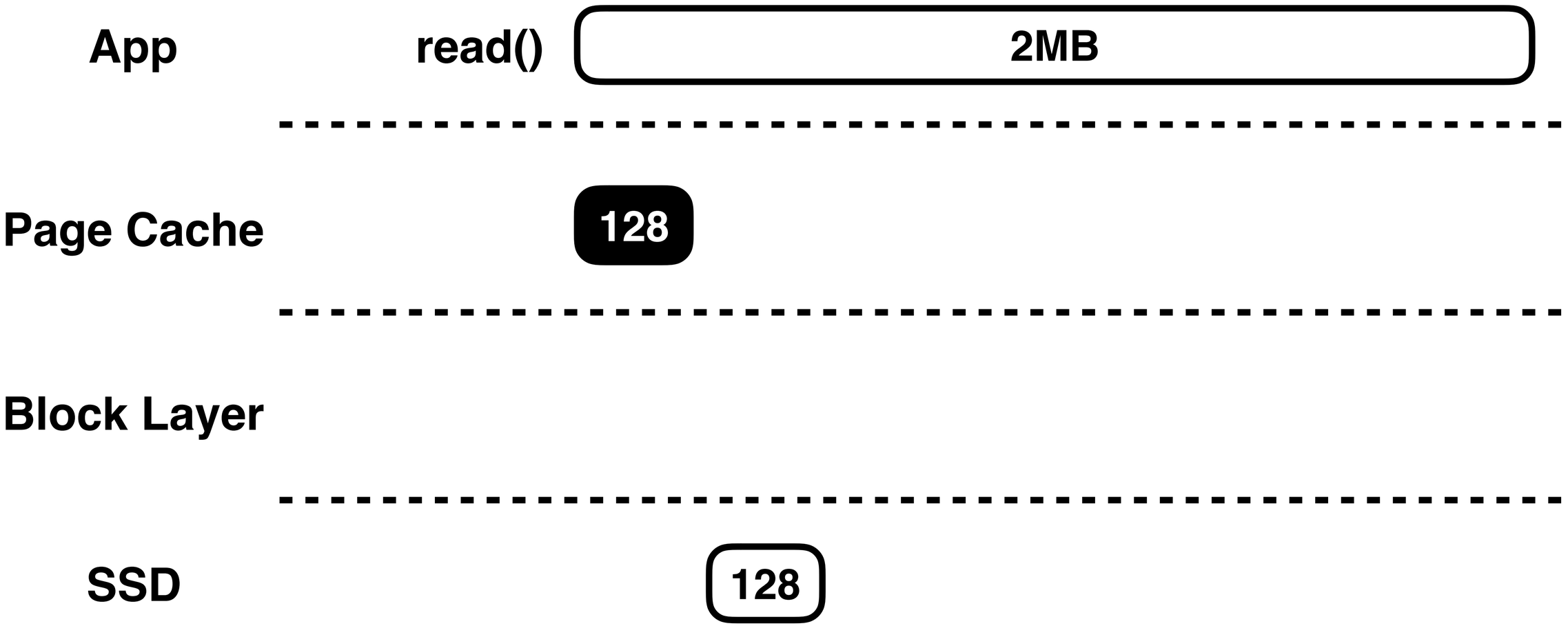
Page cache implementation splits and serializes user requests



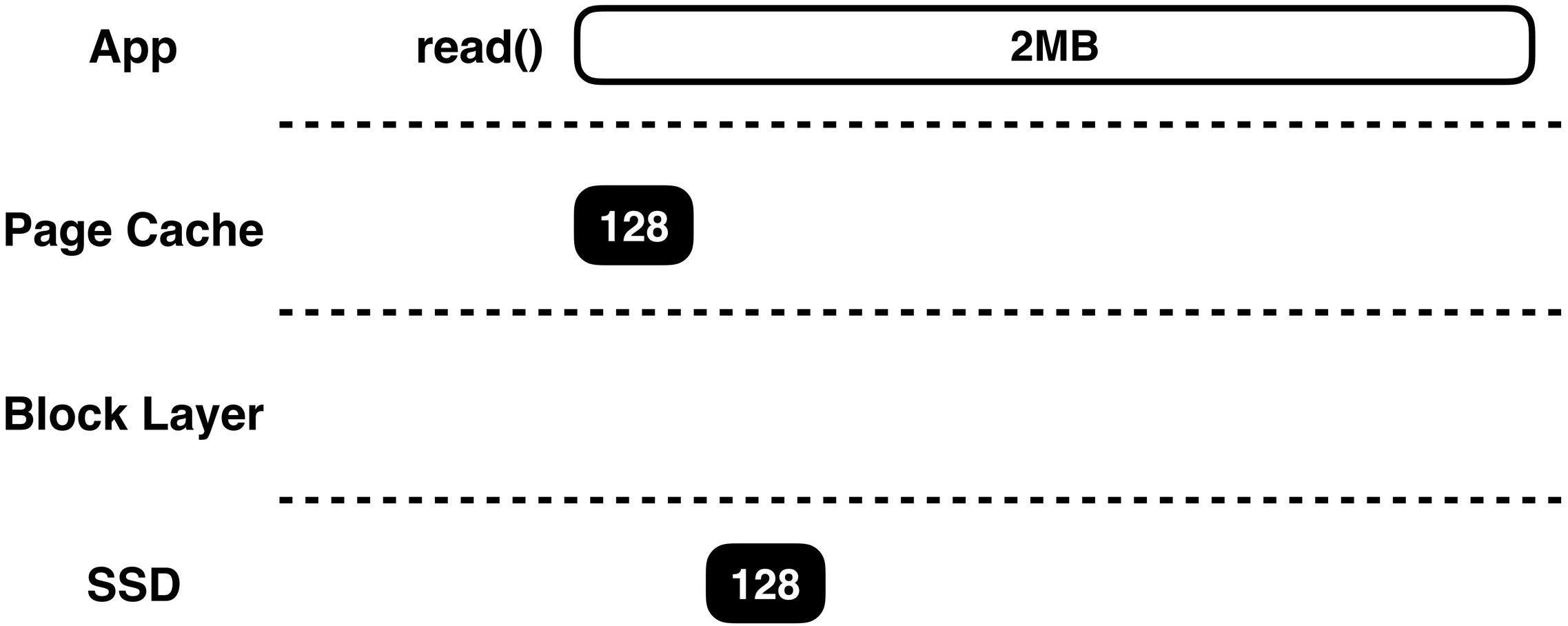
Page cache implementation splits and serializes user requests



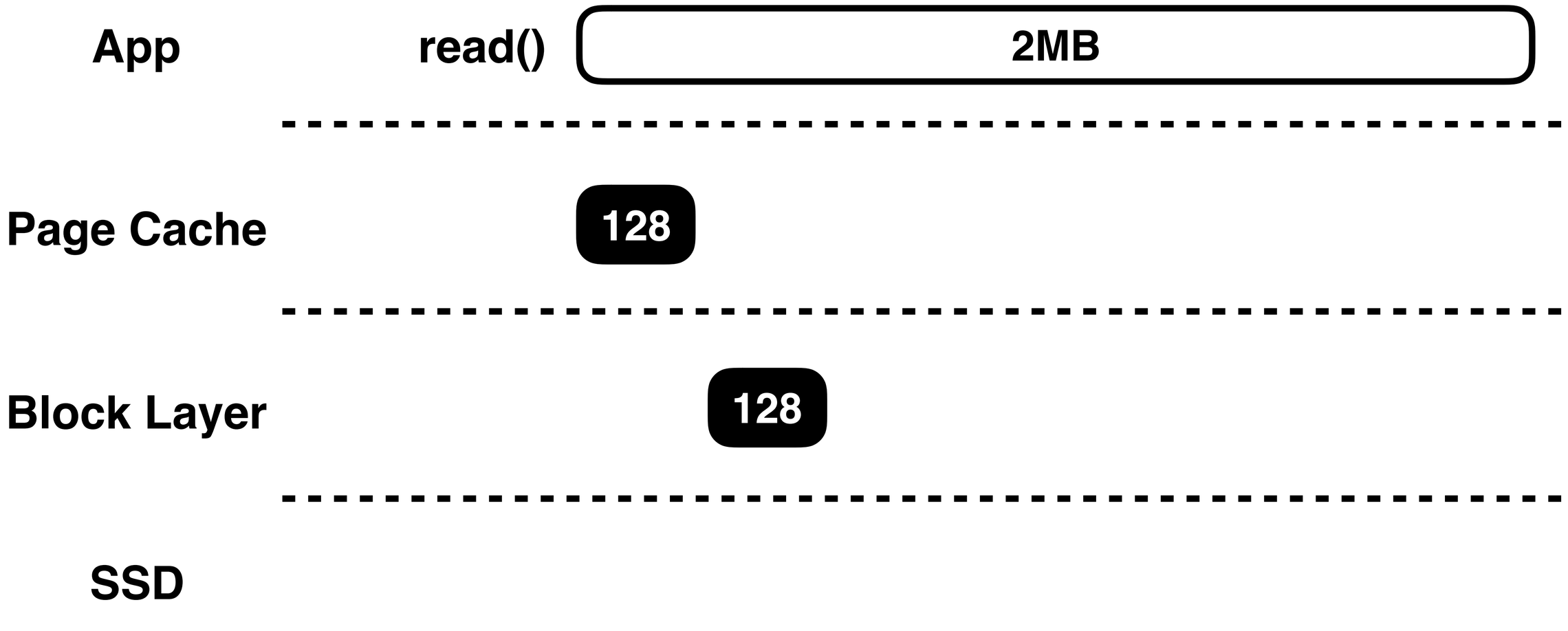
Page cache implementation splits and serializes user requests



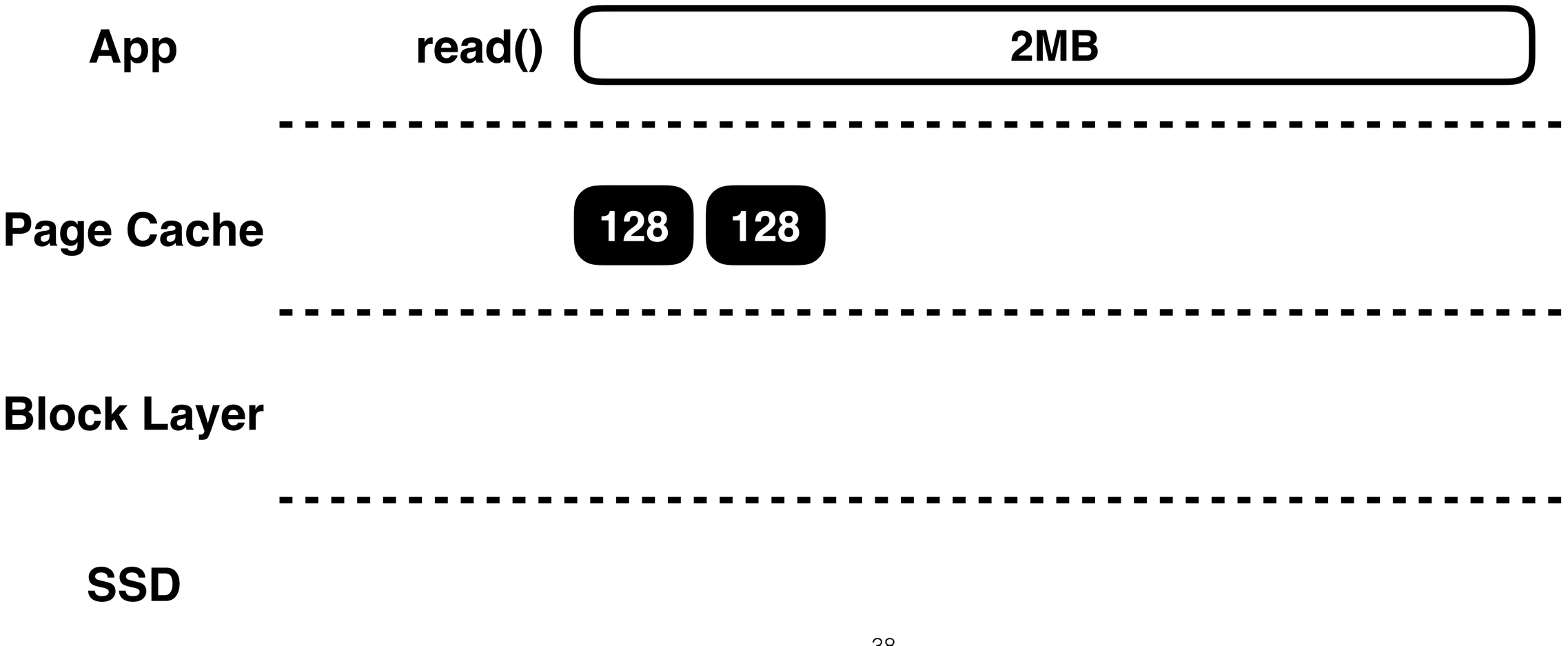
Page cache implementation splits and serializes user requests



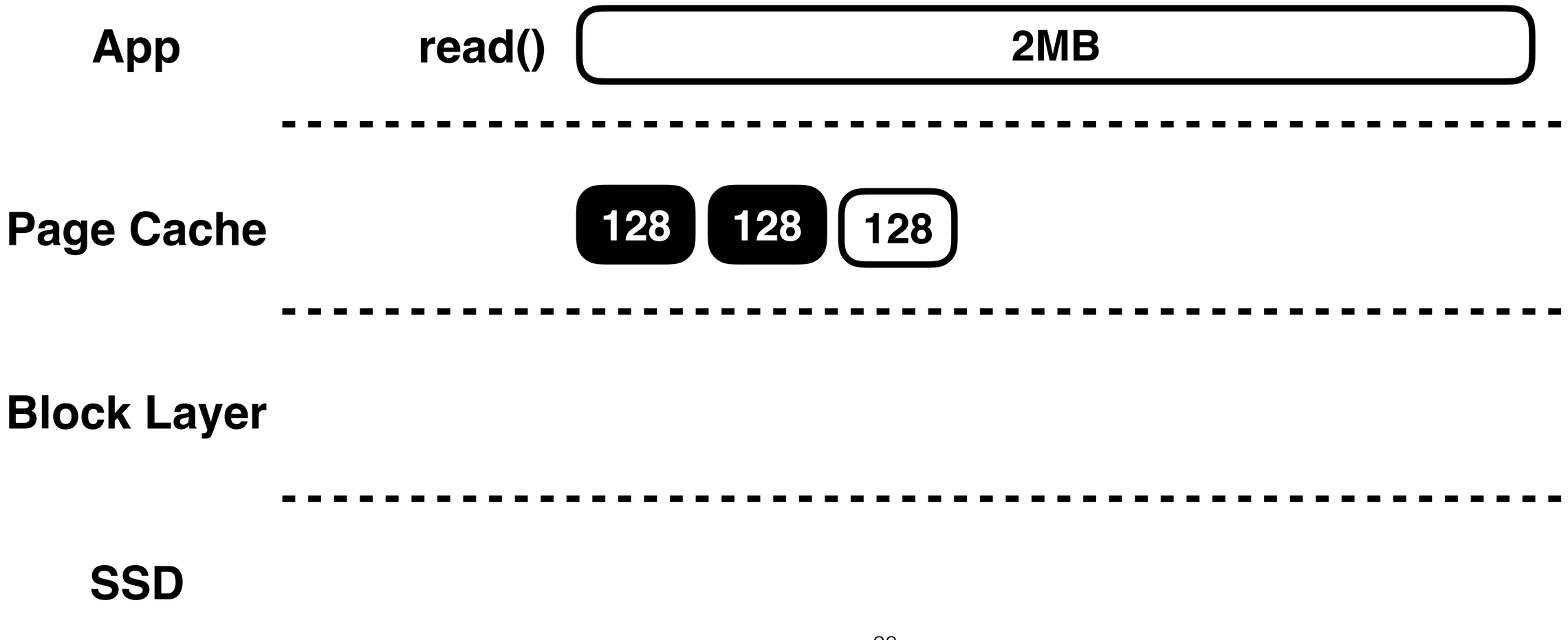
Page cache implementation splits and serializes user requests



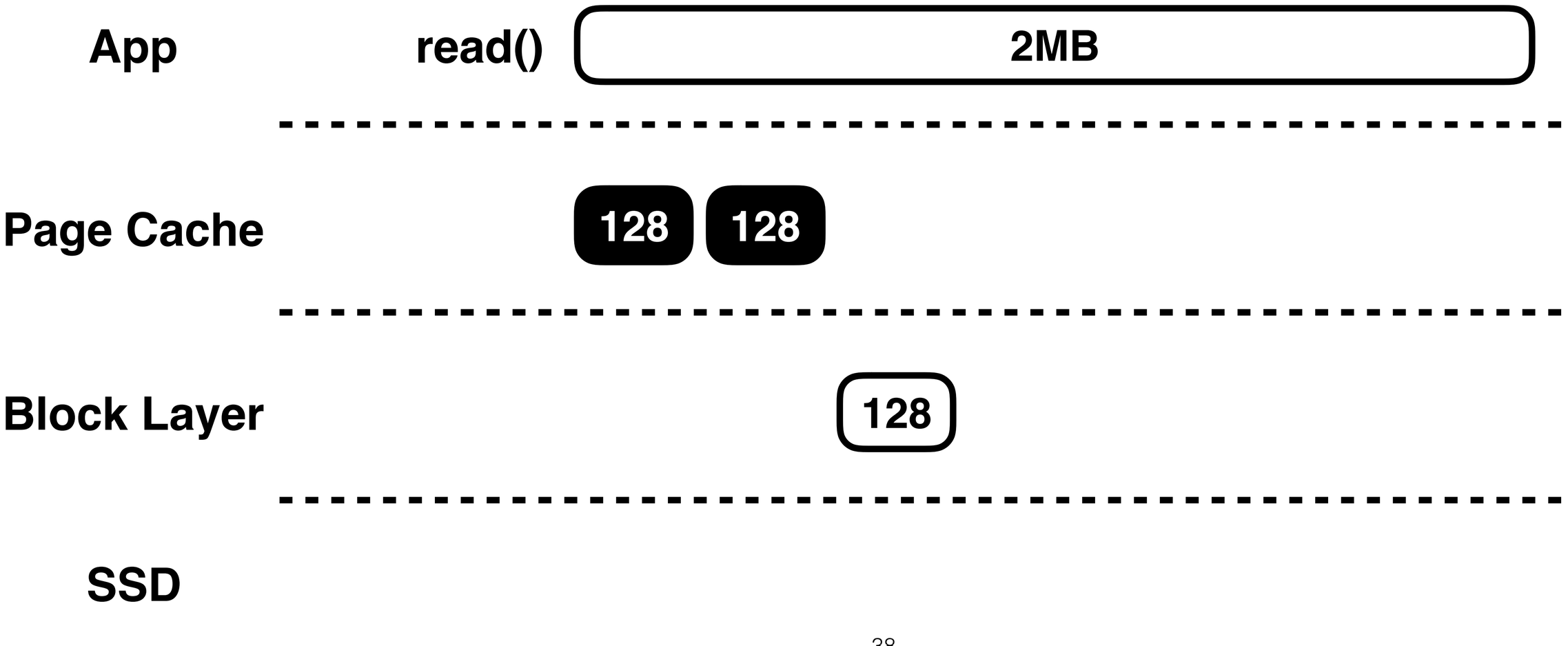
Page cache implementation splits and serializes user requests



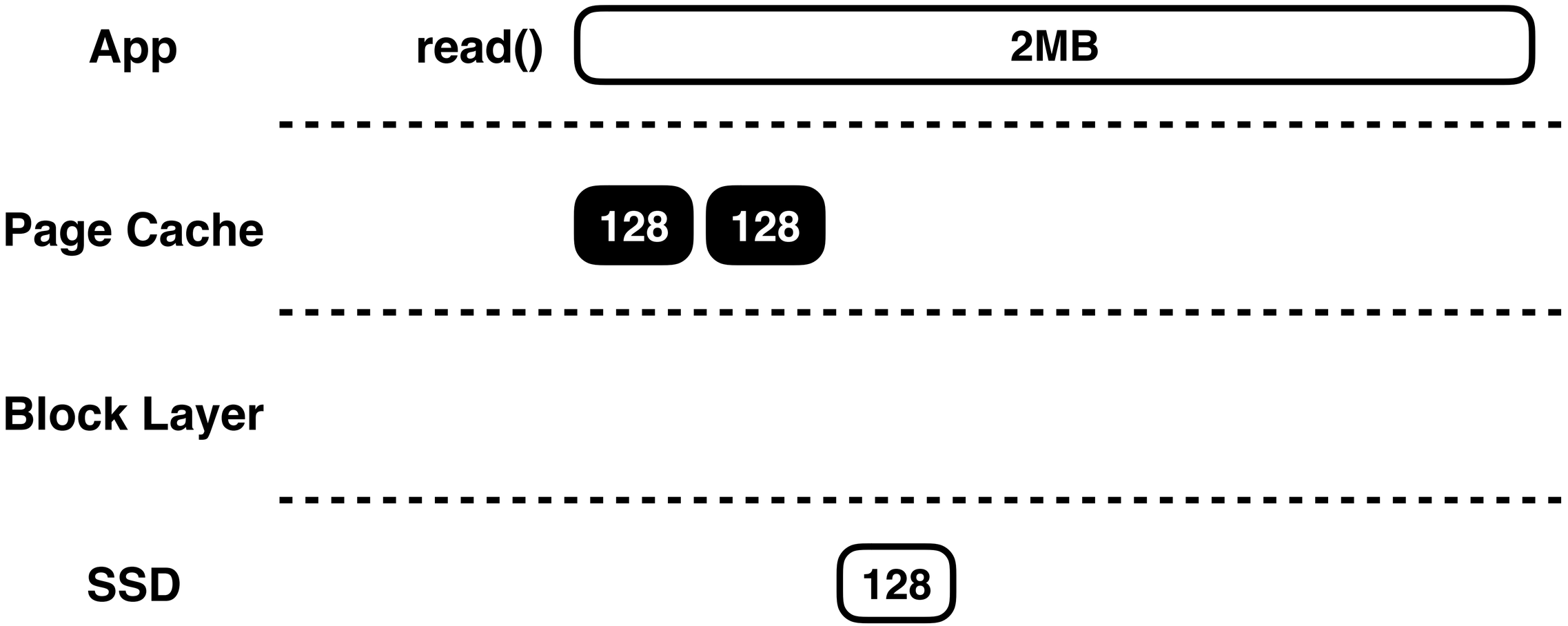
Page cache implementation splits and serializes user requests



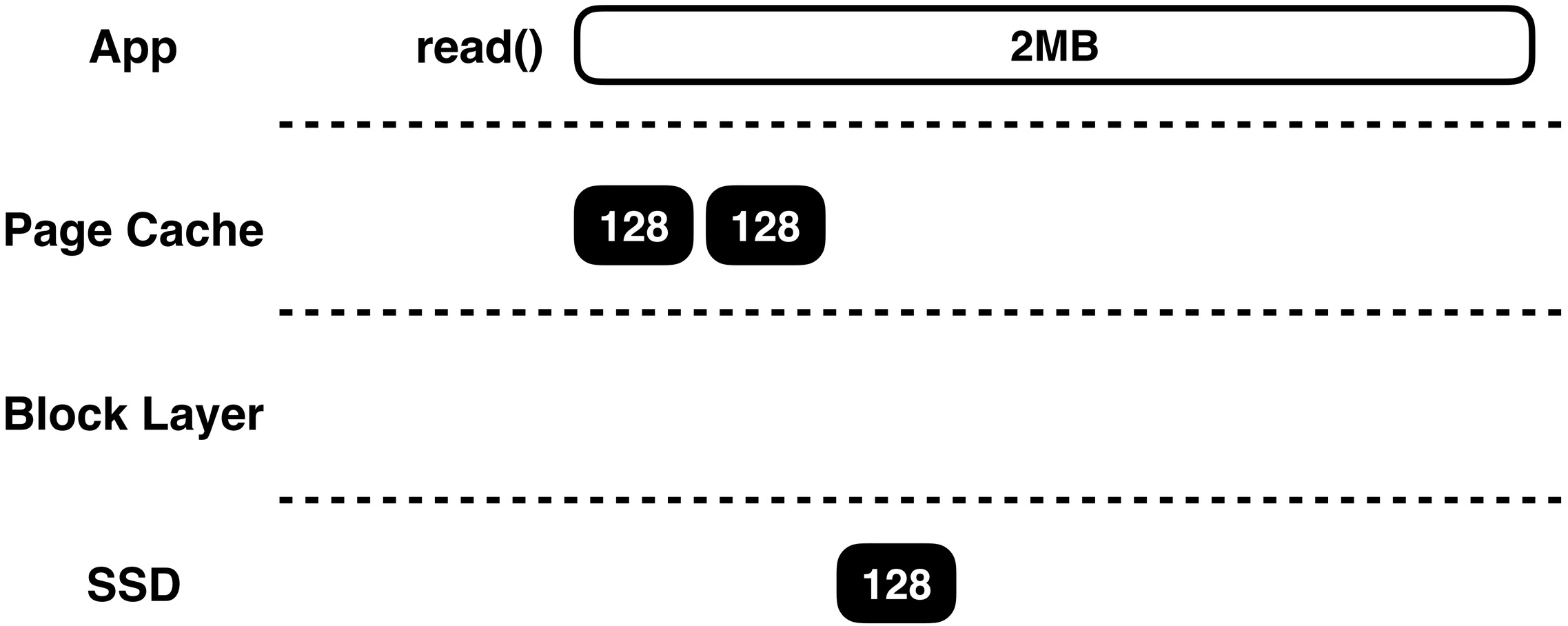
Page cache implementation splits and serializes user requests



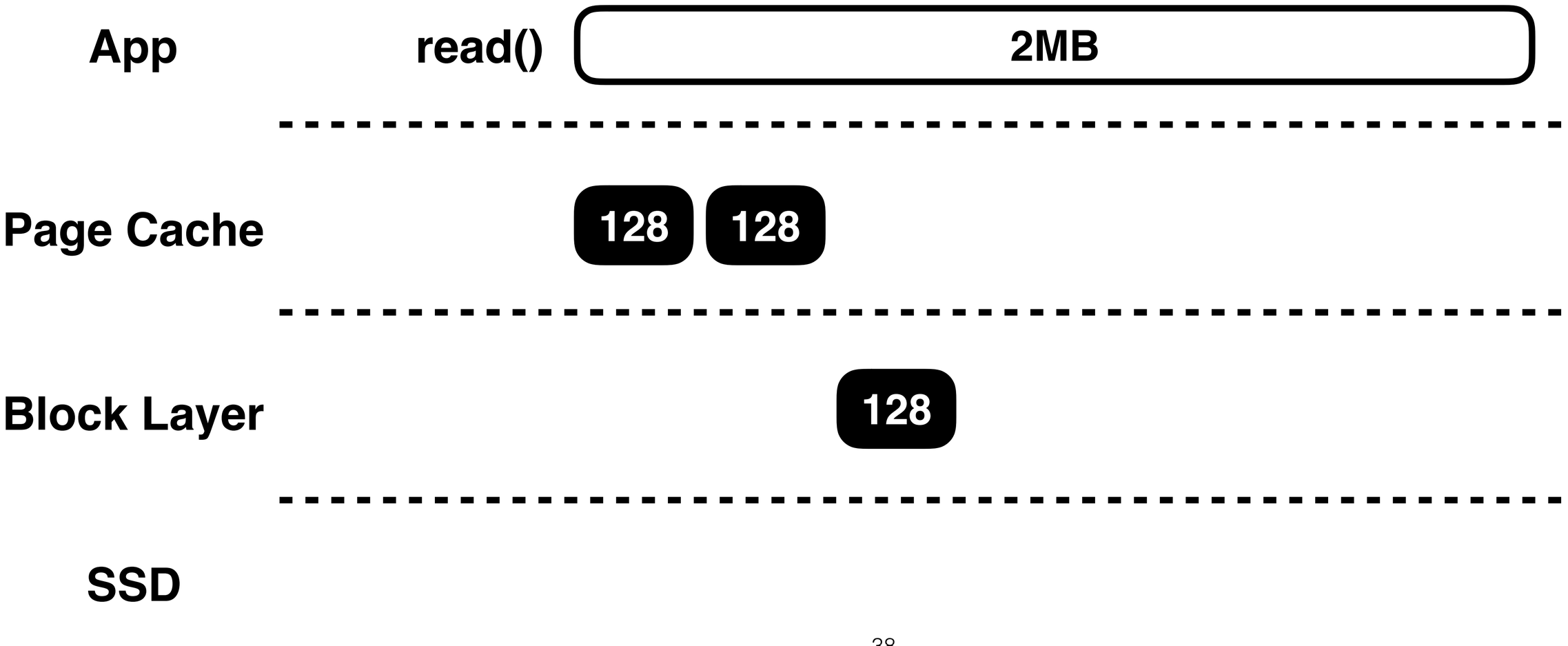
Page cache implementation splits and serializes user requests



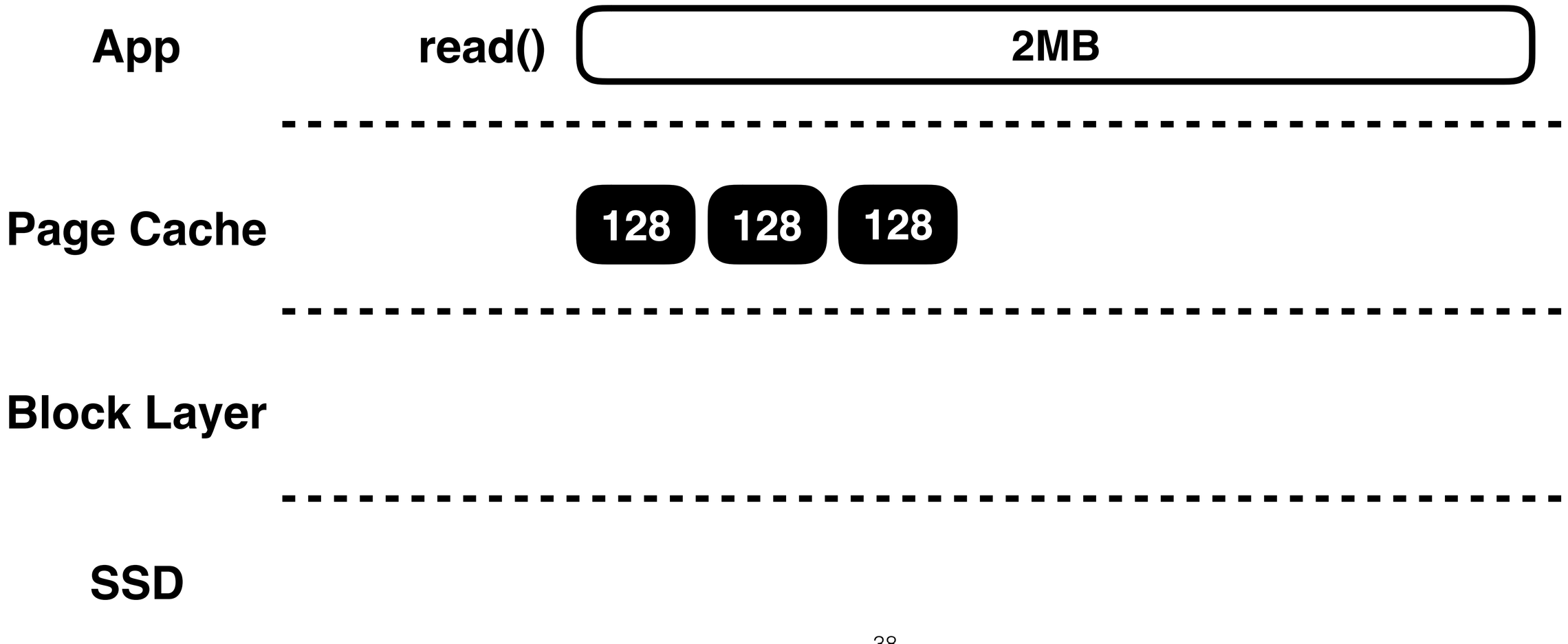
Page cache implementation splits and serializes user requests



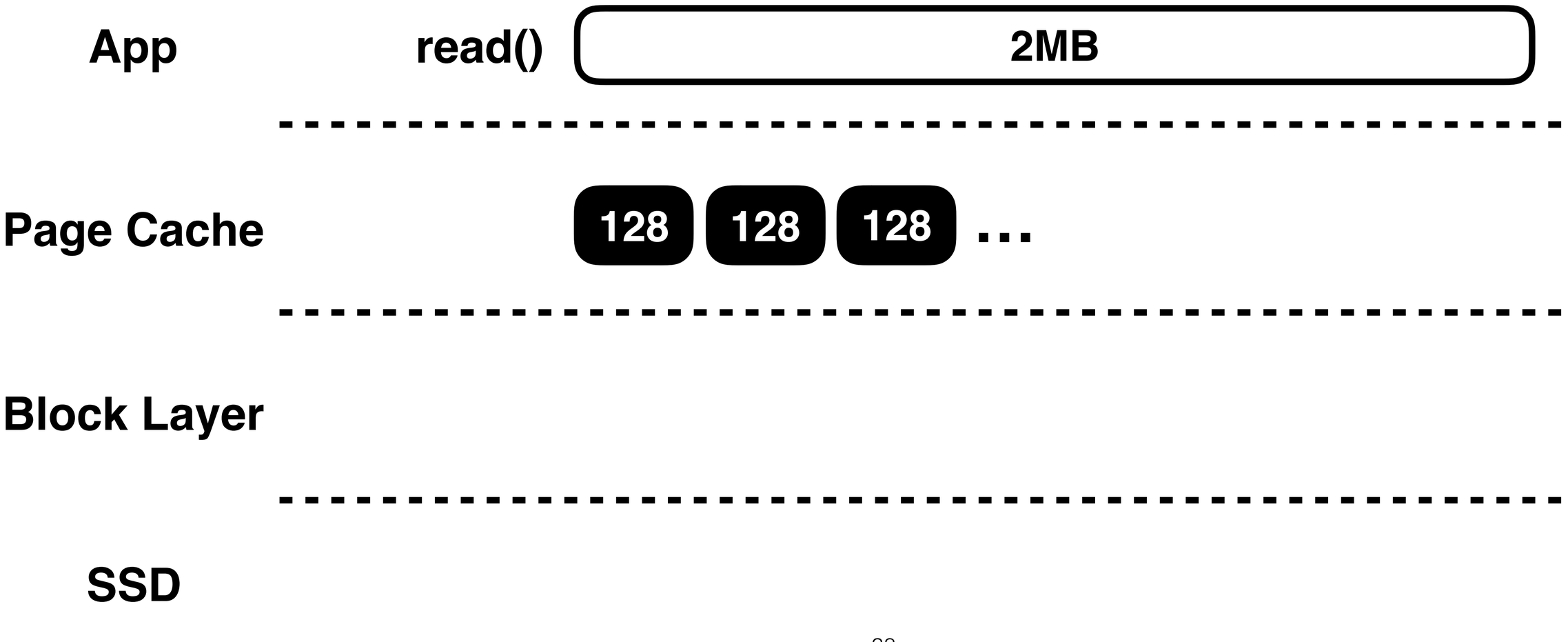
Page cache implementation splits and serializes user requests



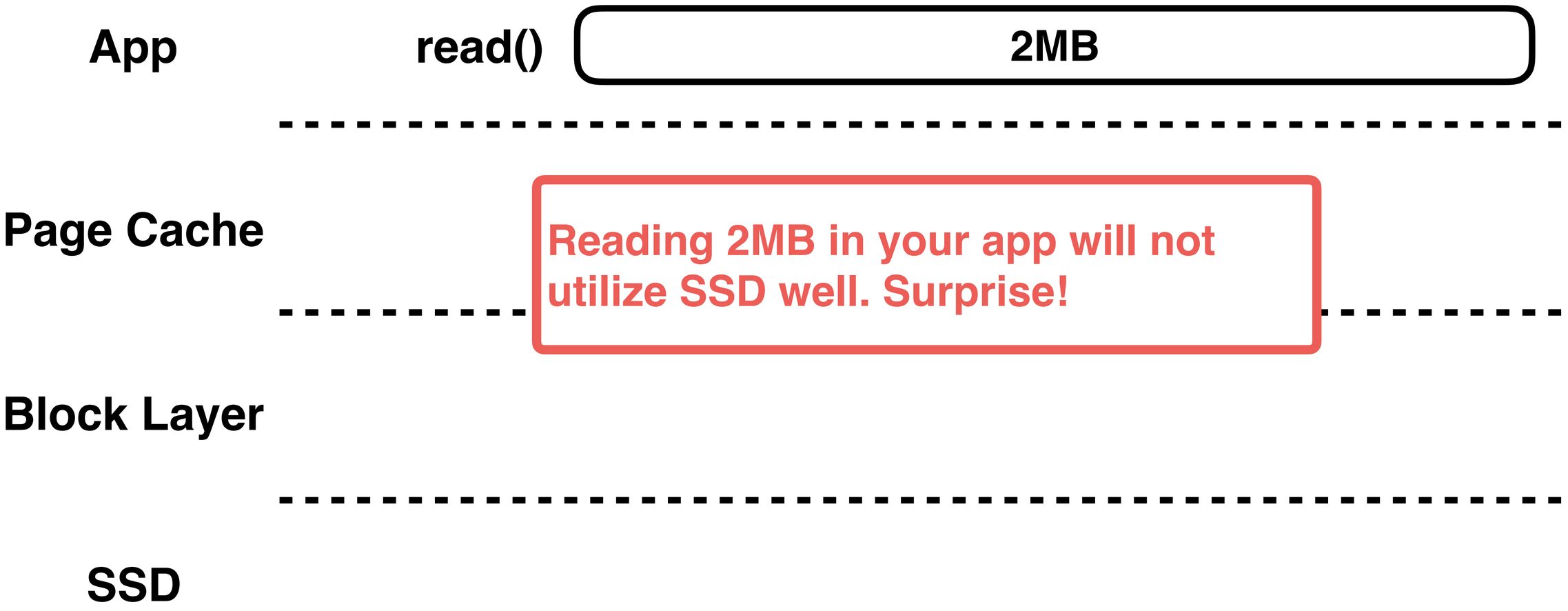
Page cache implementation splits and serializes user requests



Page cache implementation splits and serializes user requests



Page cache implementation splits and serializes user requests



Page cache implementation splits and serializes user requests

App

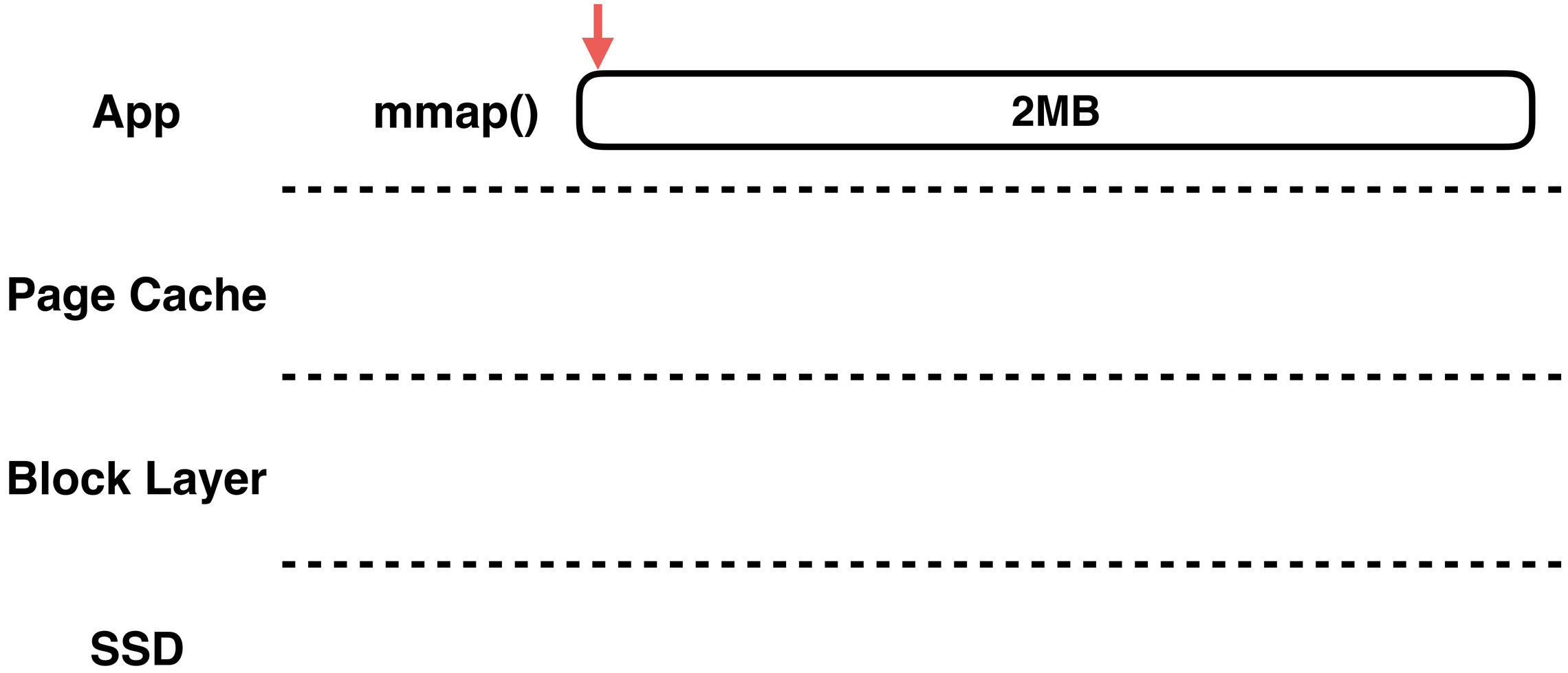
Page Cache

Block Layer

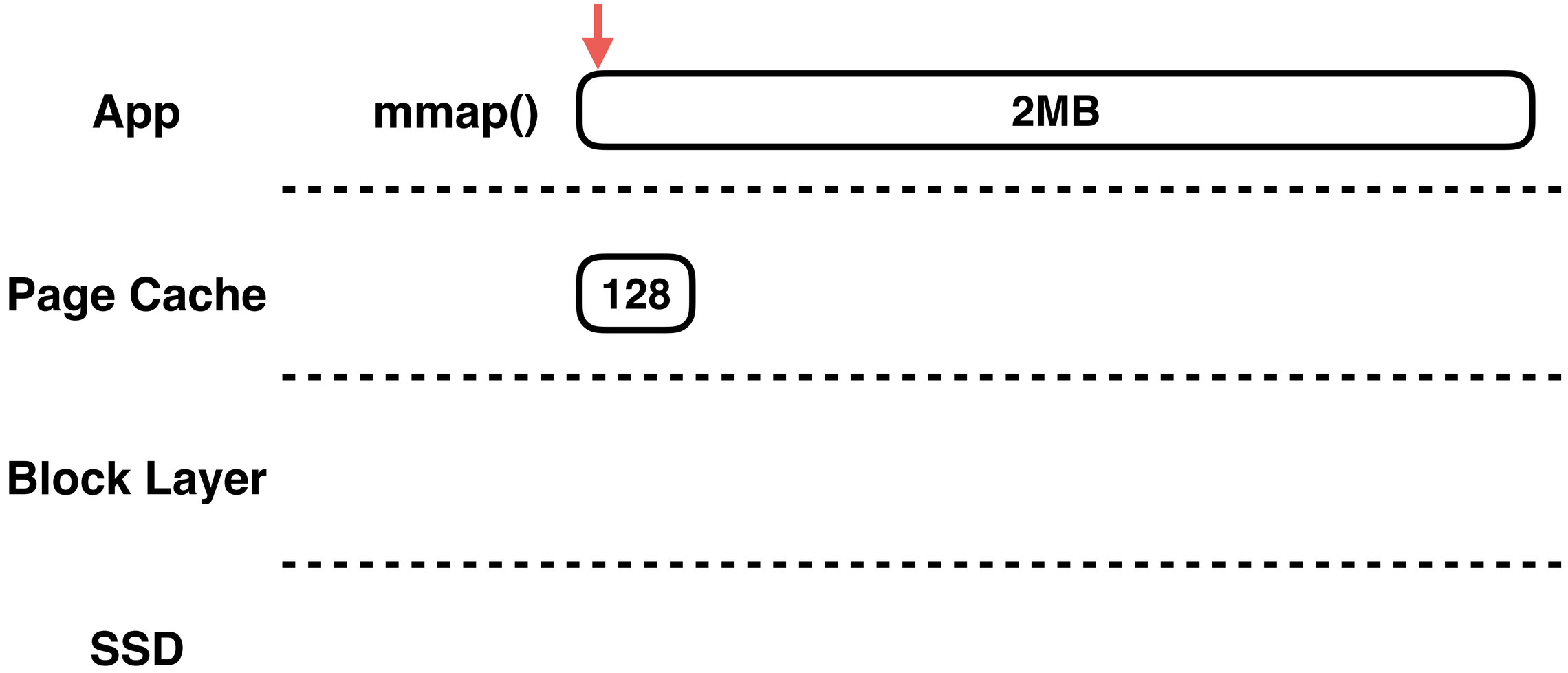
SSD

Page cache implementation splits and serializes user requests

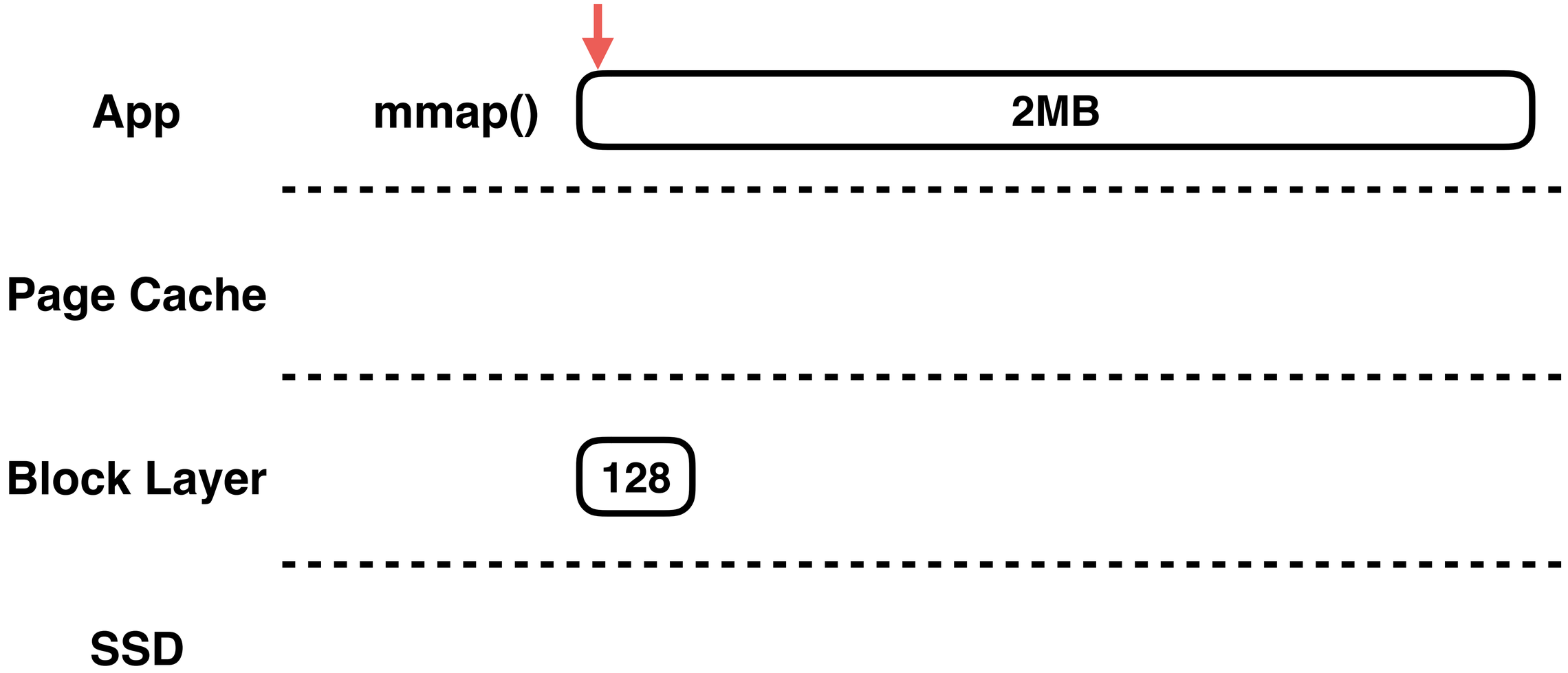
Page cache implementation splits and serializes user requests



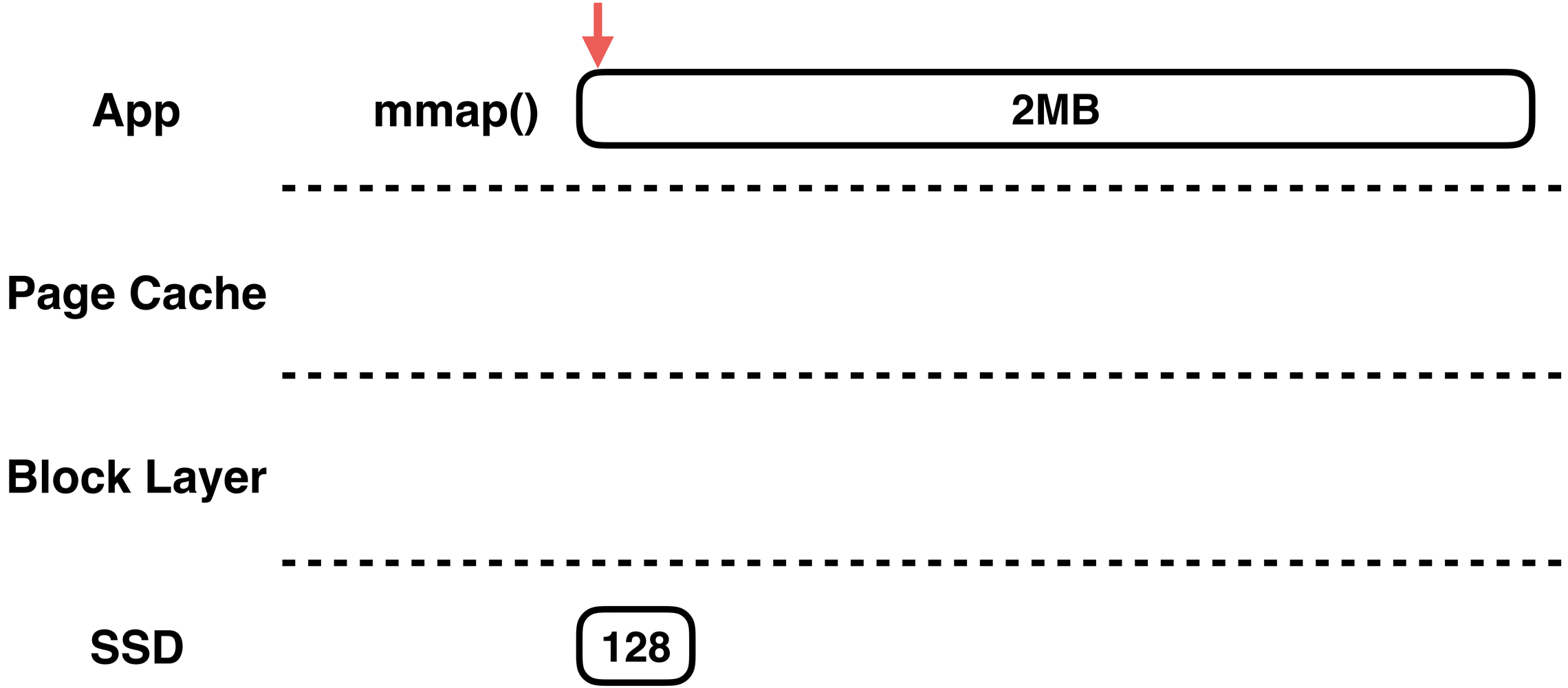
Page cache implementation splits and serializes user requests



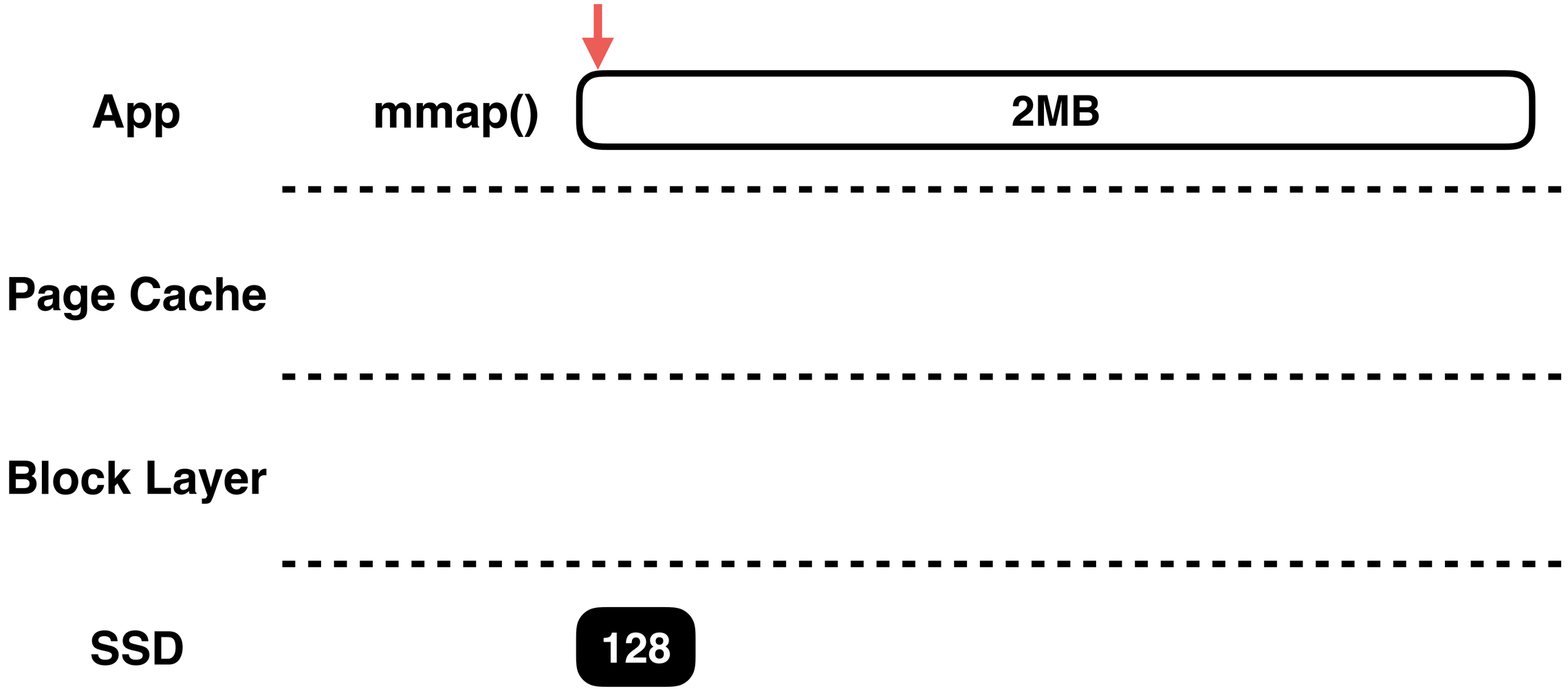
Page cache implementation splits and serializes user requests



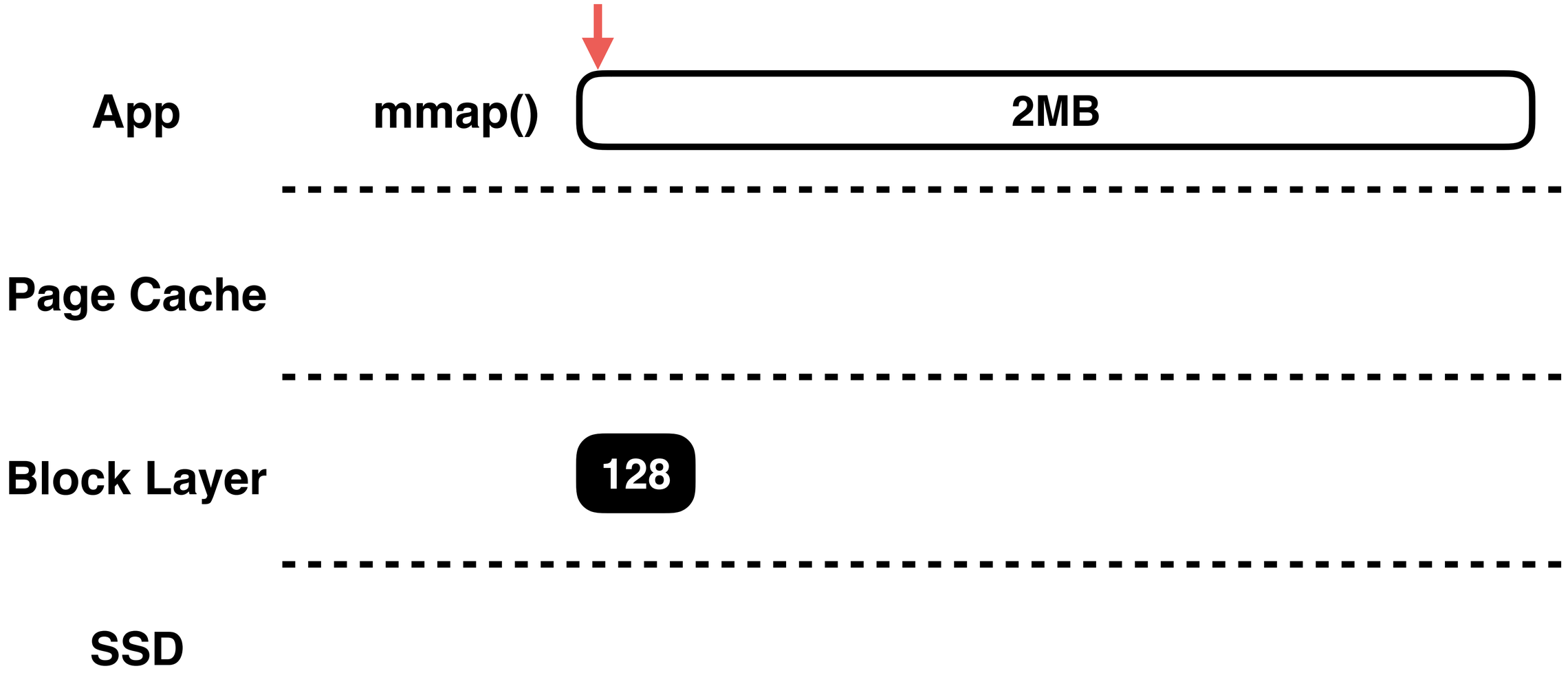
Page cache implementation splits and serializes user requests



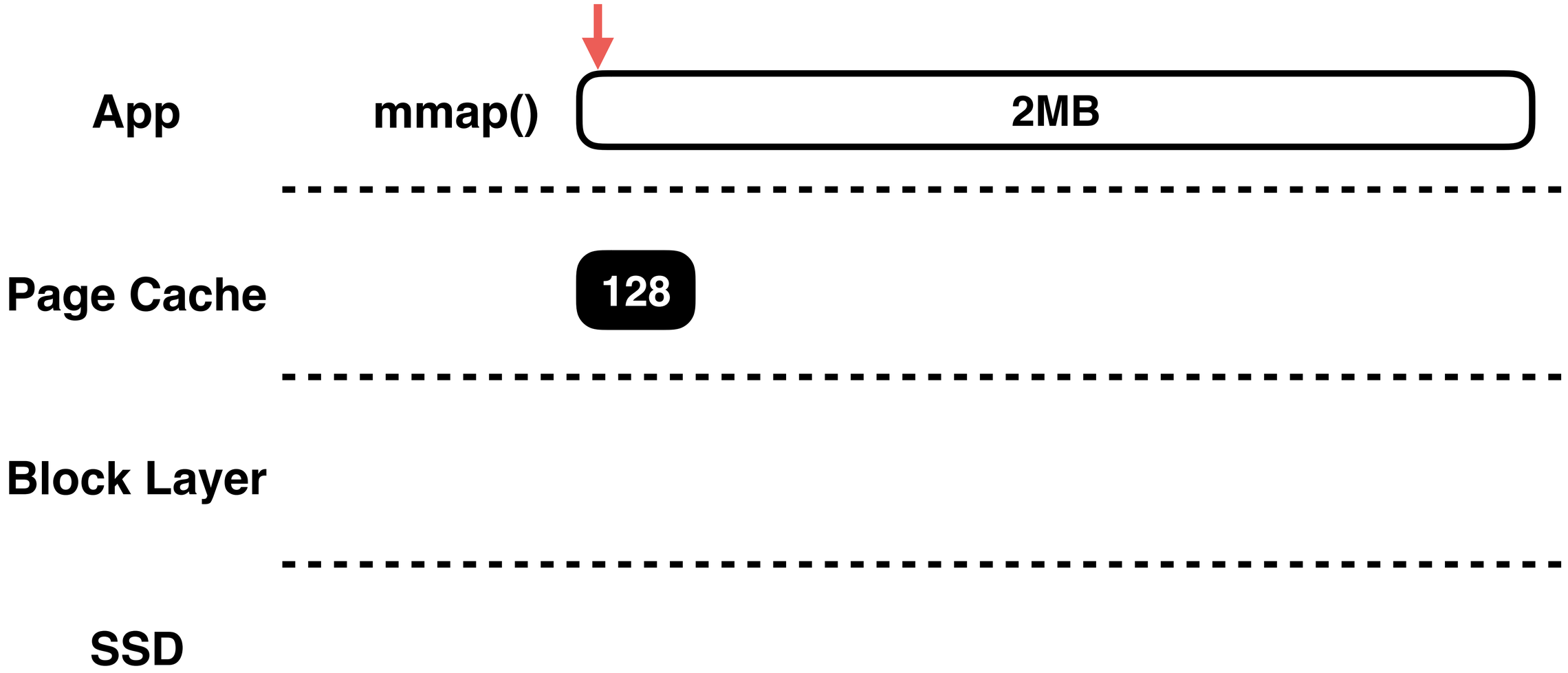
Page cache implementation splits and serializes user requests



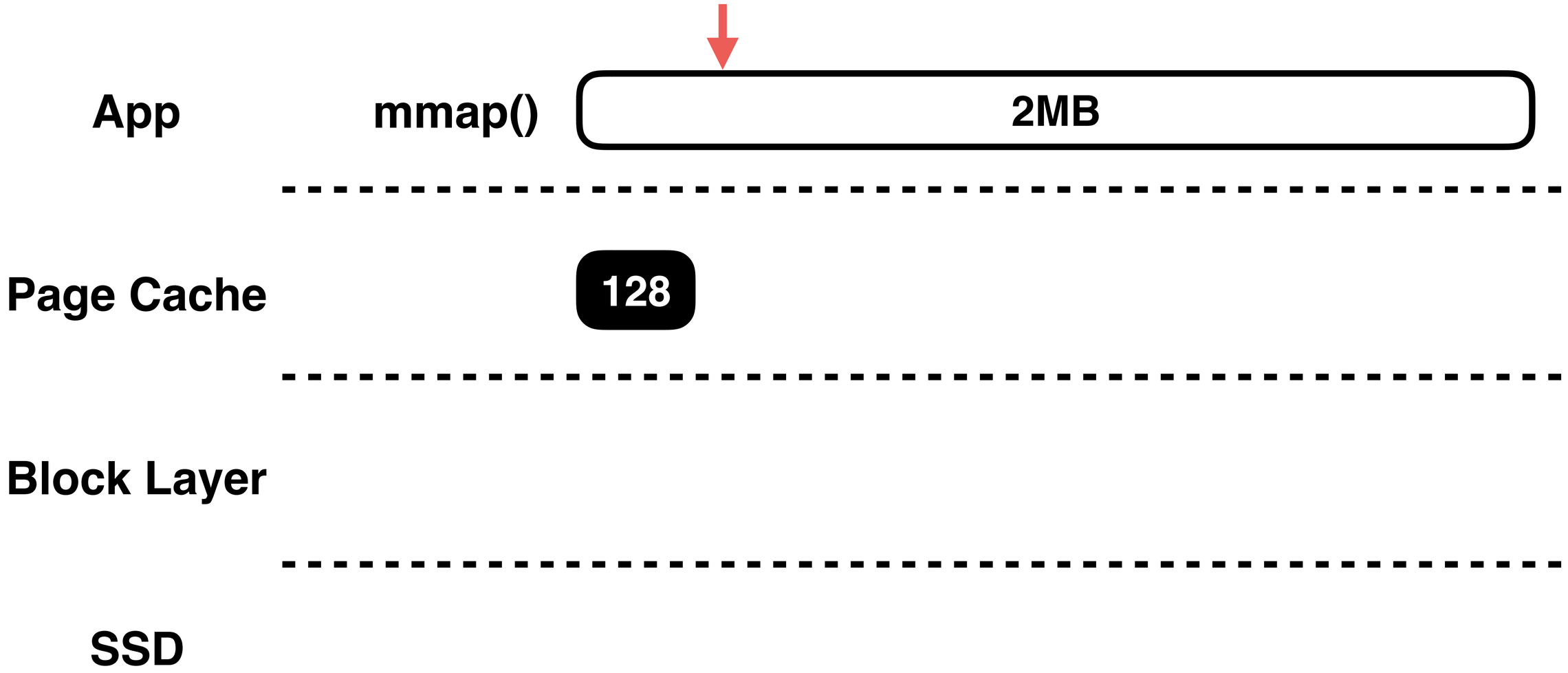
Page cache implementation splits and serializes user requests



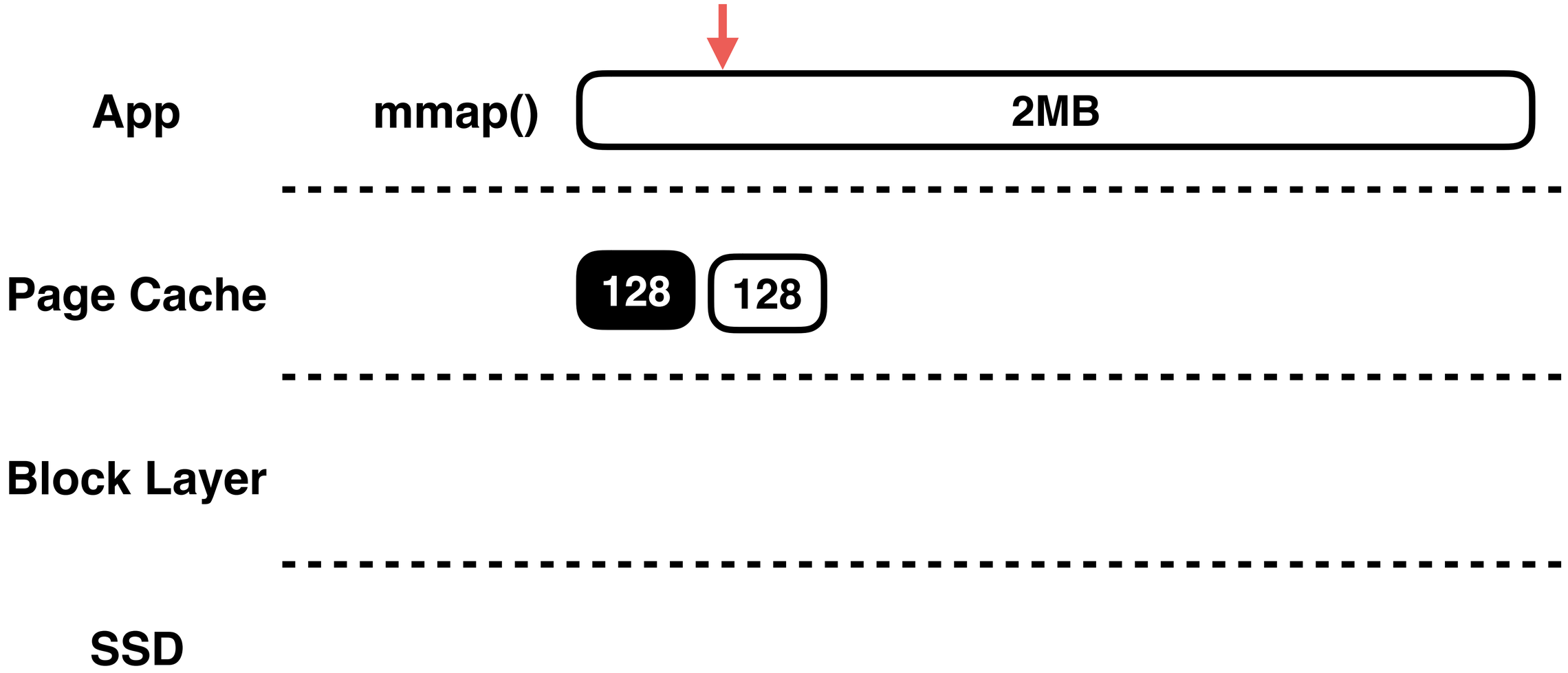
Page cache implementation splits and serializes user requests



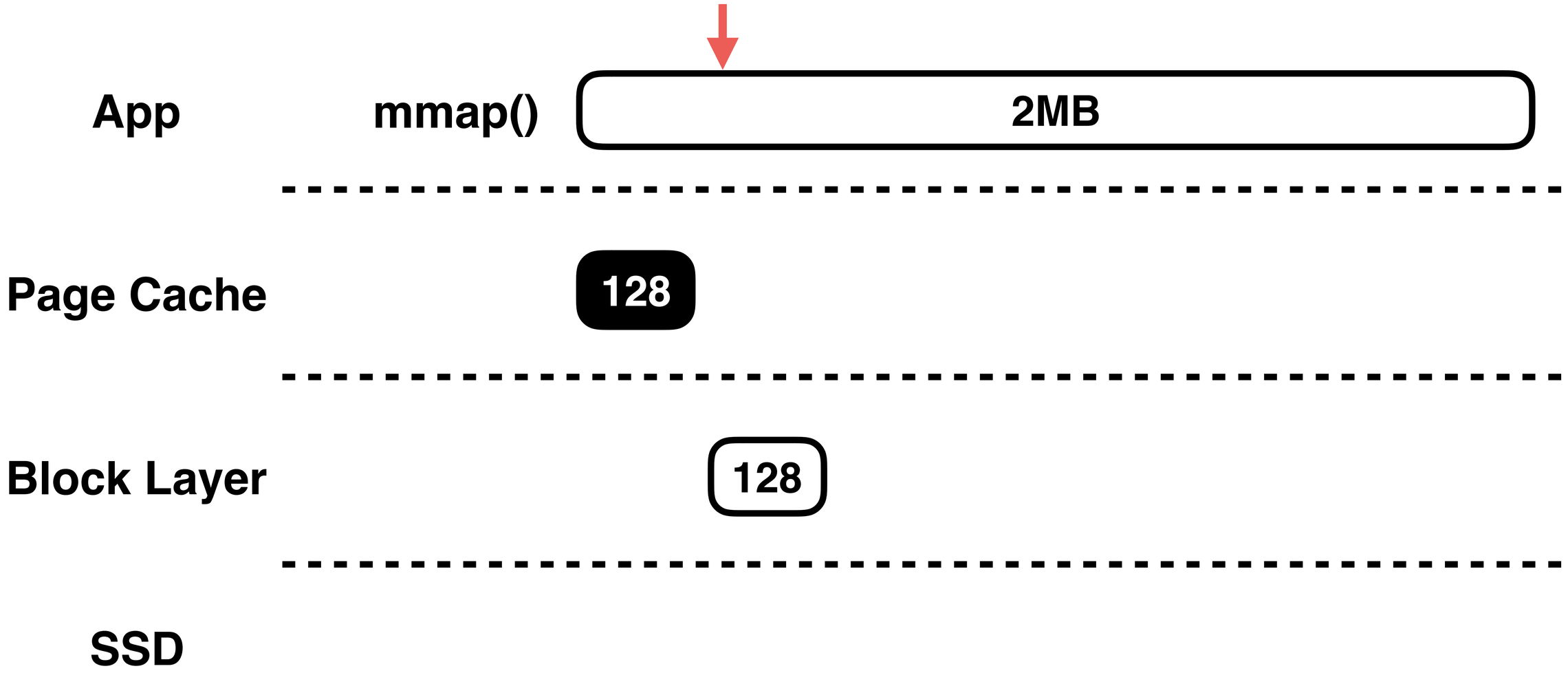
Page cache implementation splits and serializes user requests



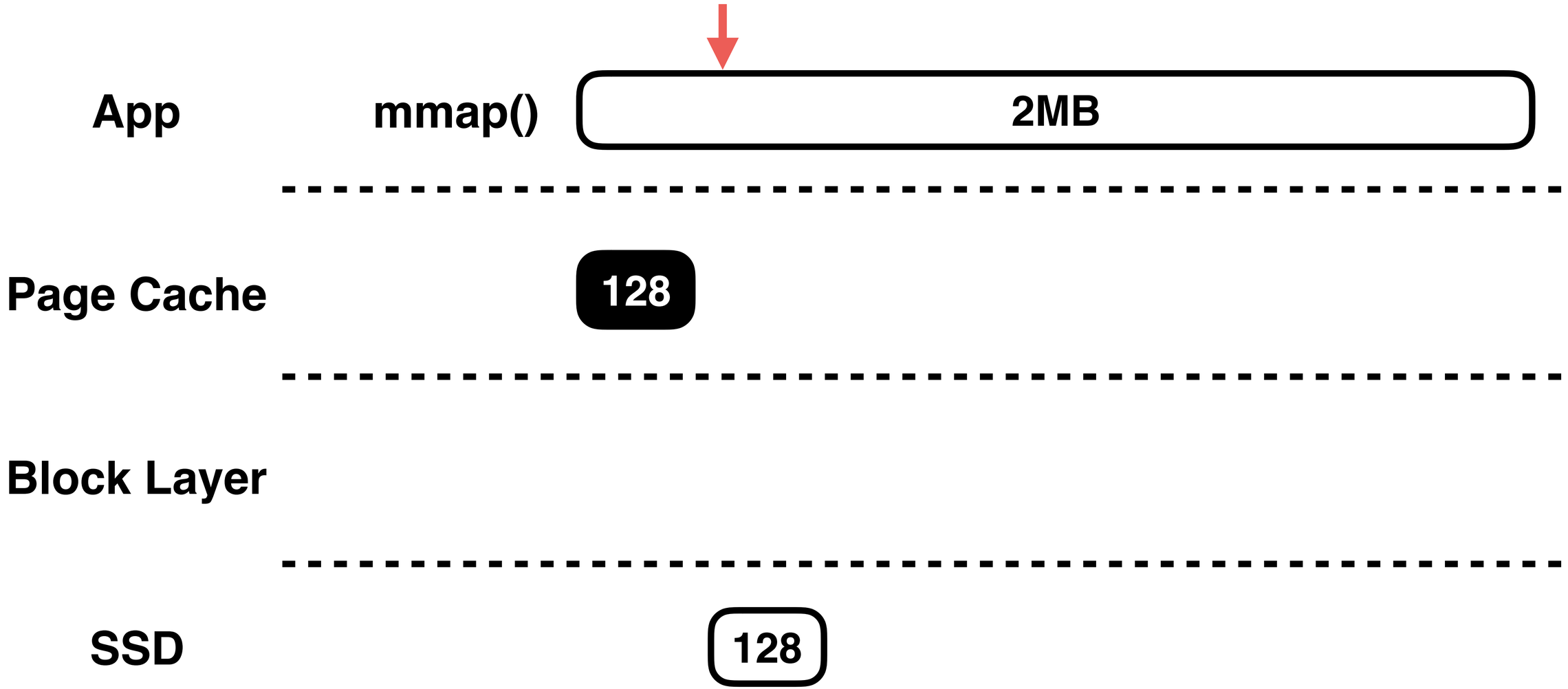
Page cache implementation splits and serializes user requests



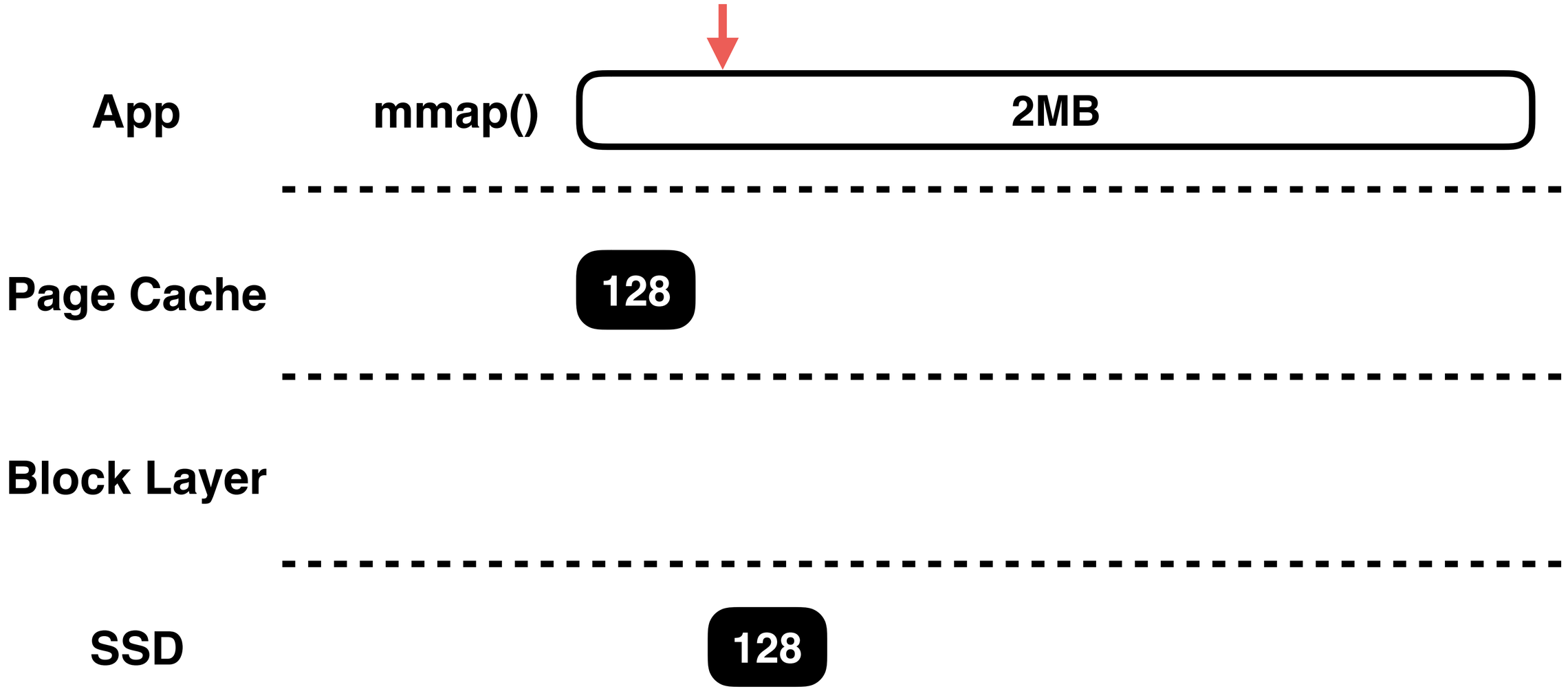
Page cache implementation splits and serializes user requests



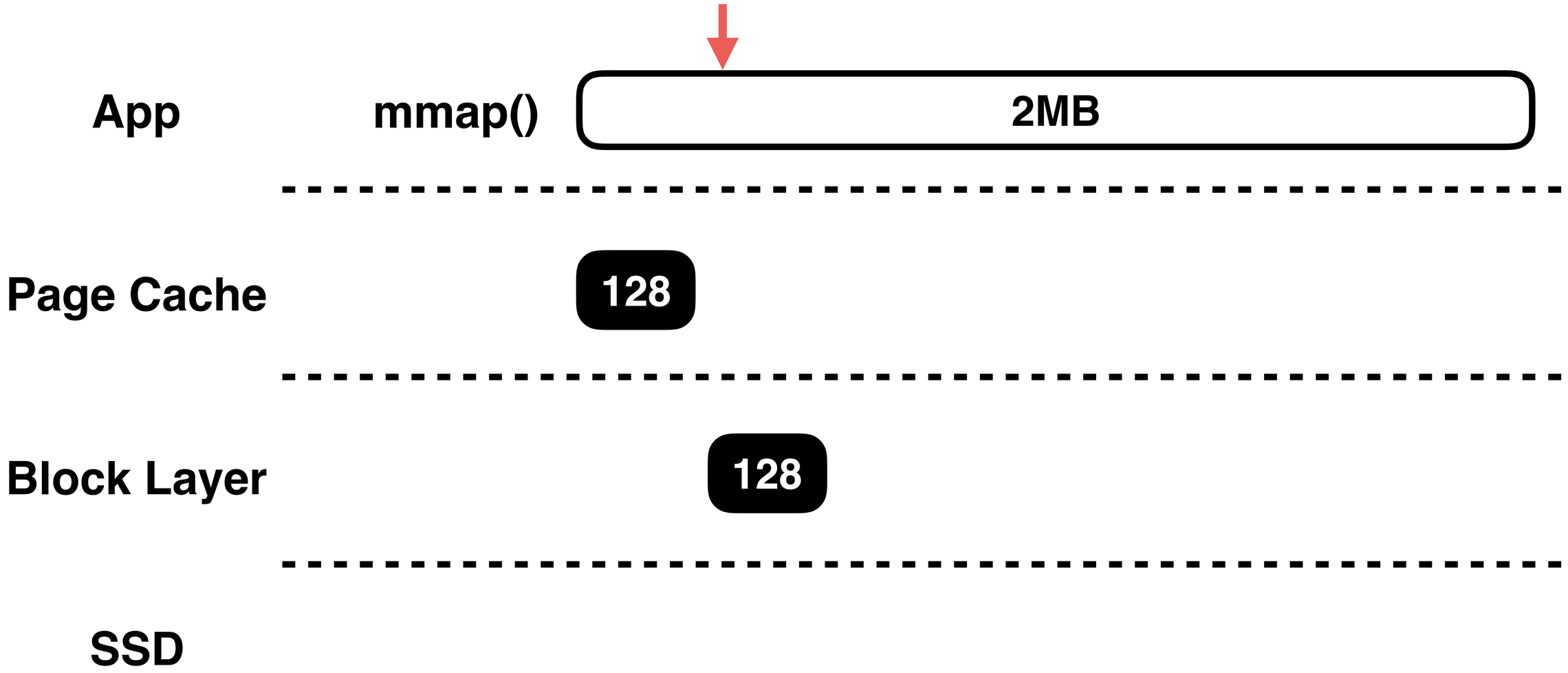
Page cache implementation splits and serializes user requests



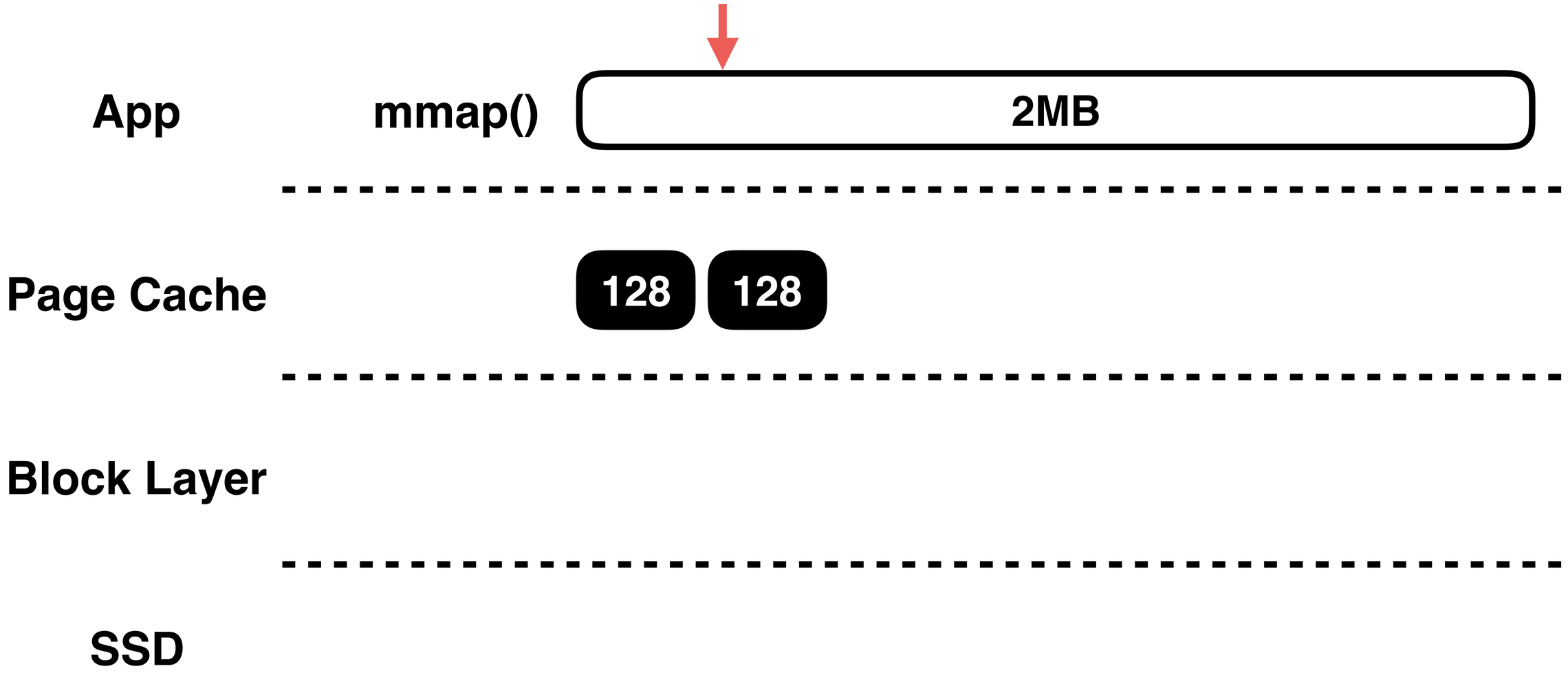
Page cache implementation splits and serializes user requests



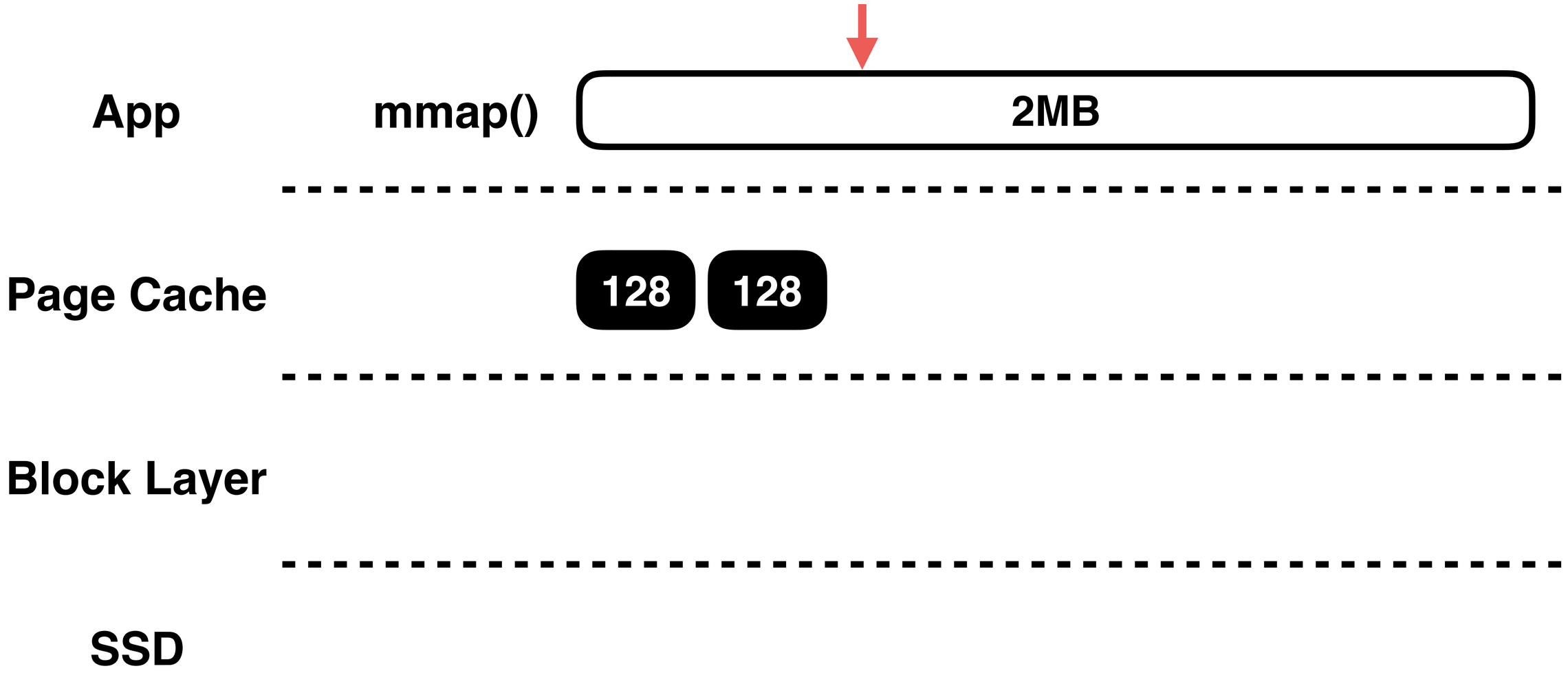
Page cache implementation splits and serializes user requests



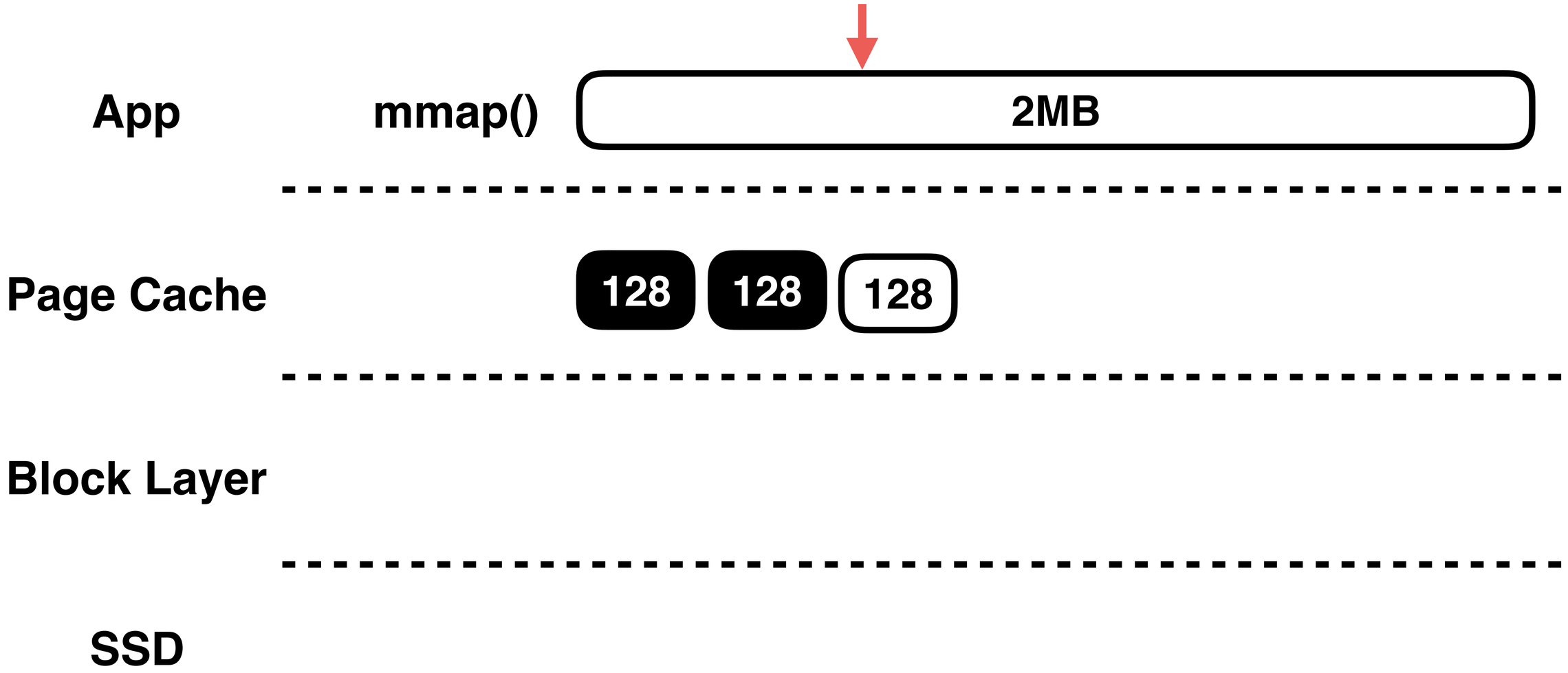
Page cache implementation splits and serializes user requests



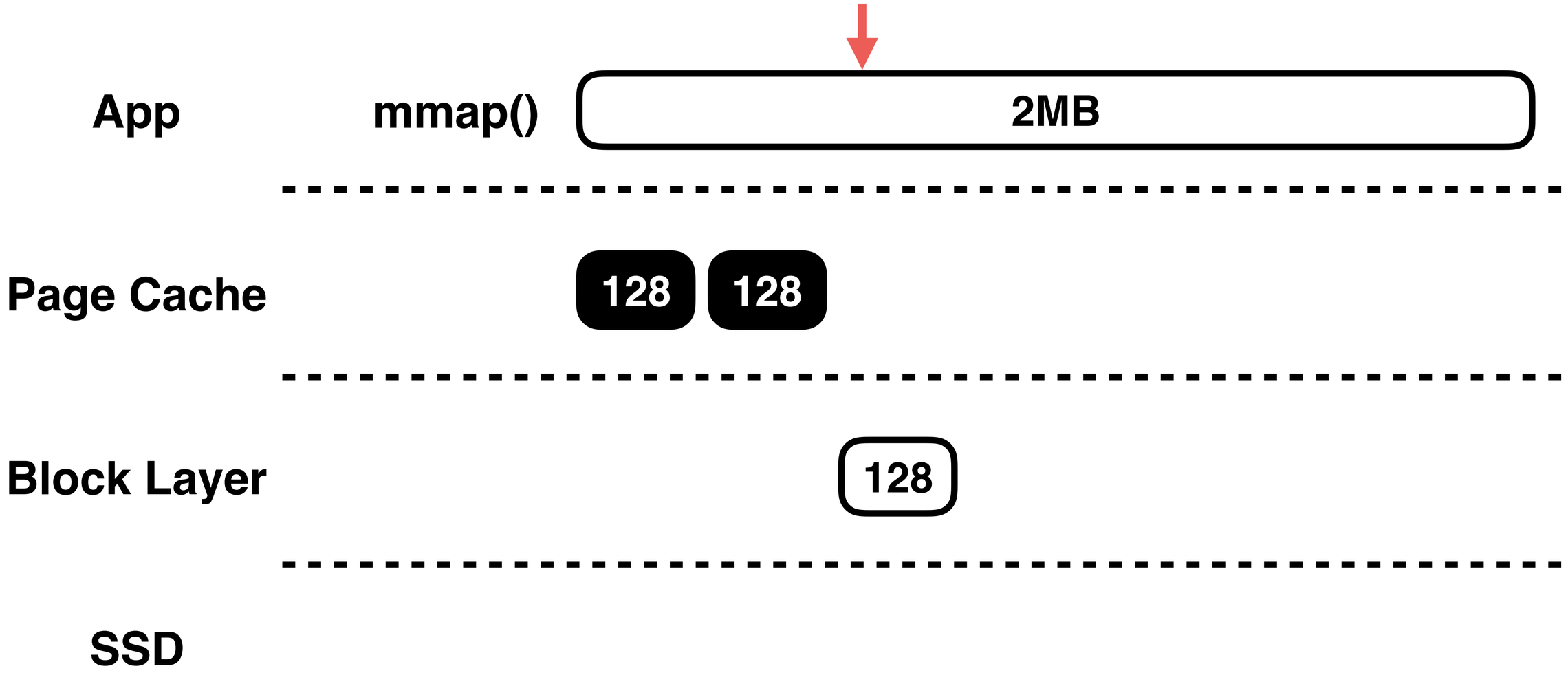
Page cache implementation splits and serializes user requests



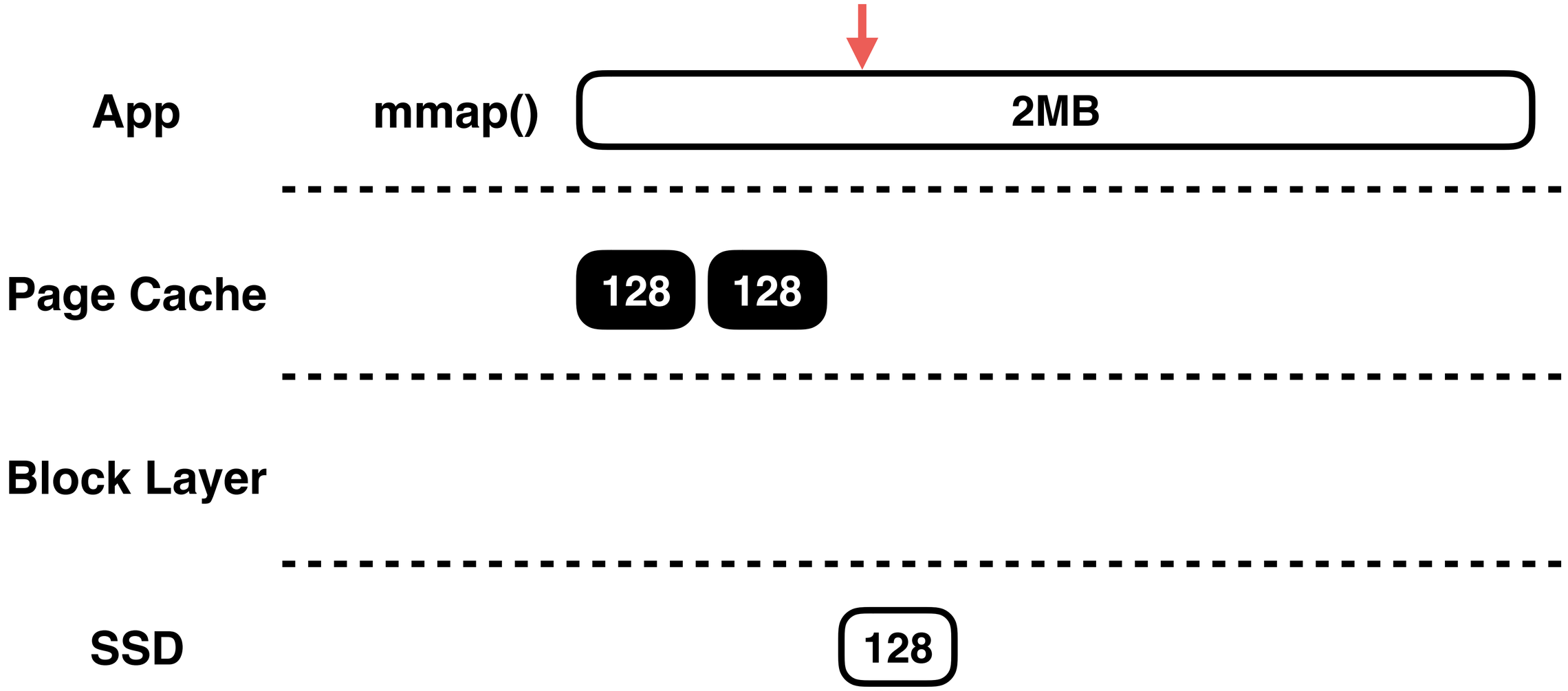
Page cache implementation splits and serializes user requests



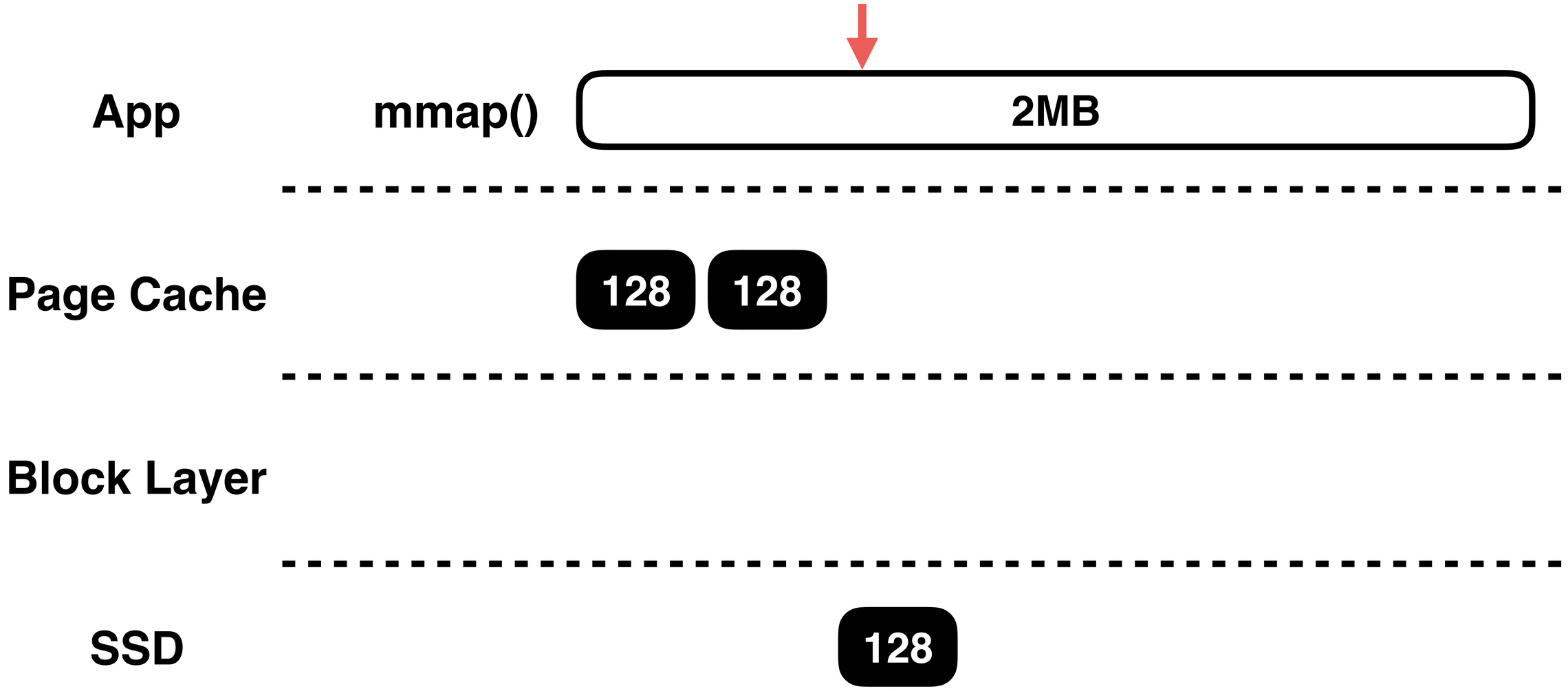
Page cache implementation splits and serializes user requests



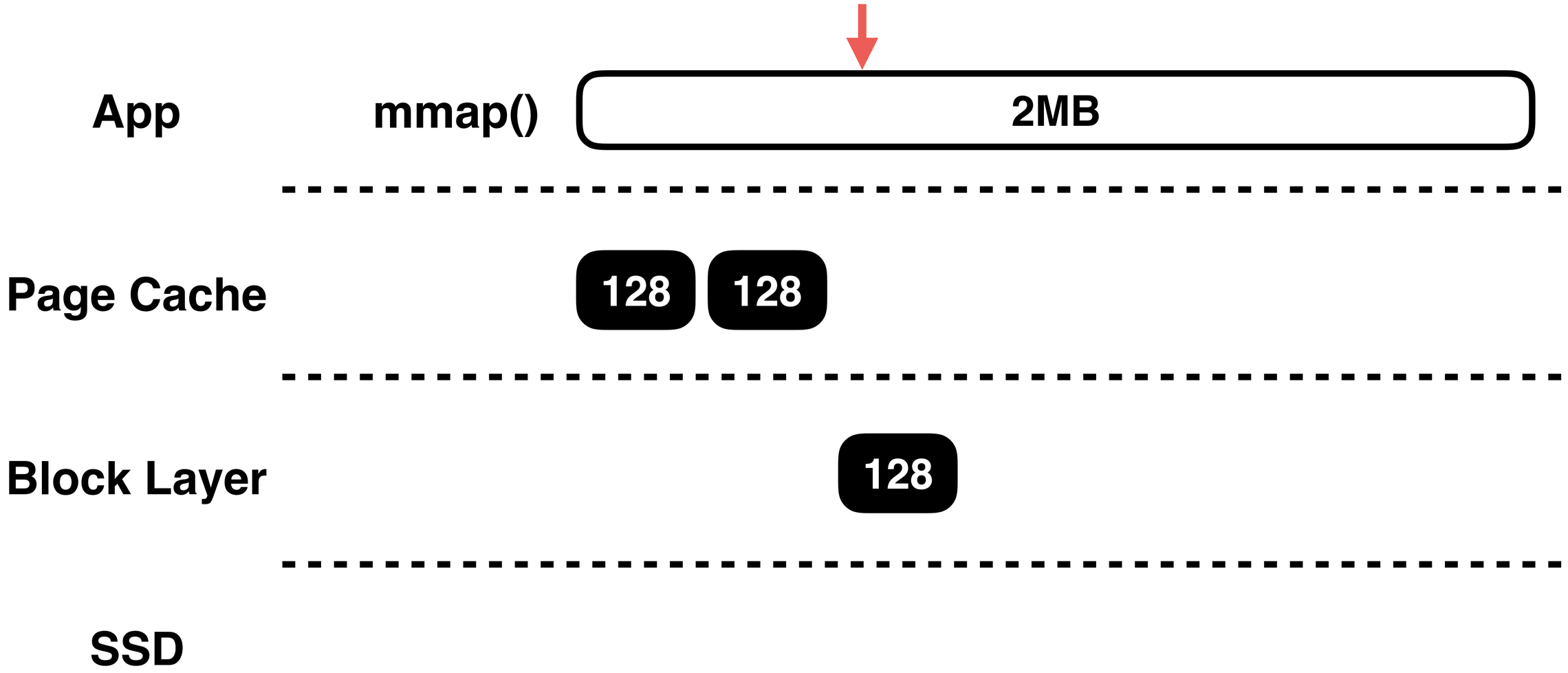
Page cache implementation splits and serializes user requests



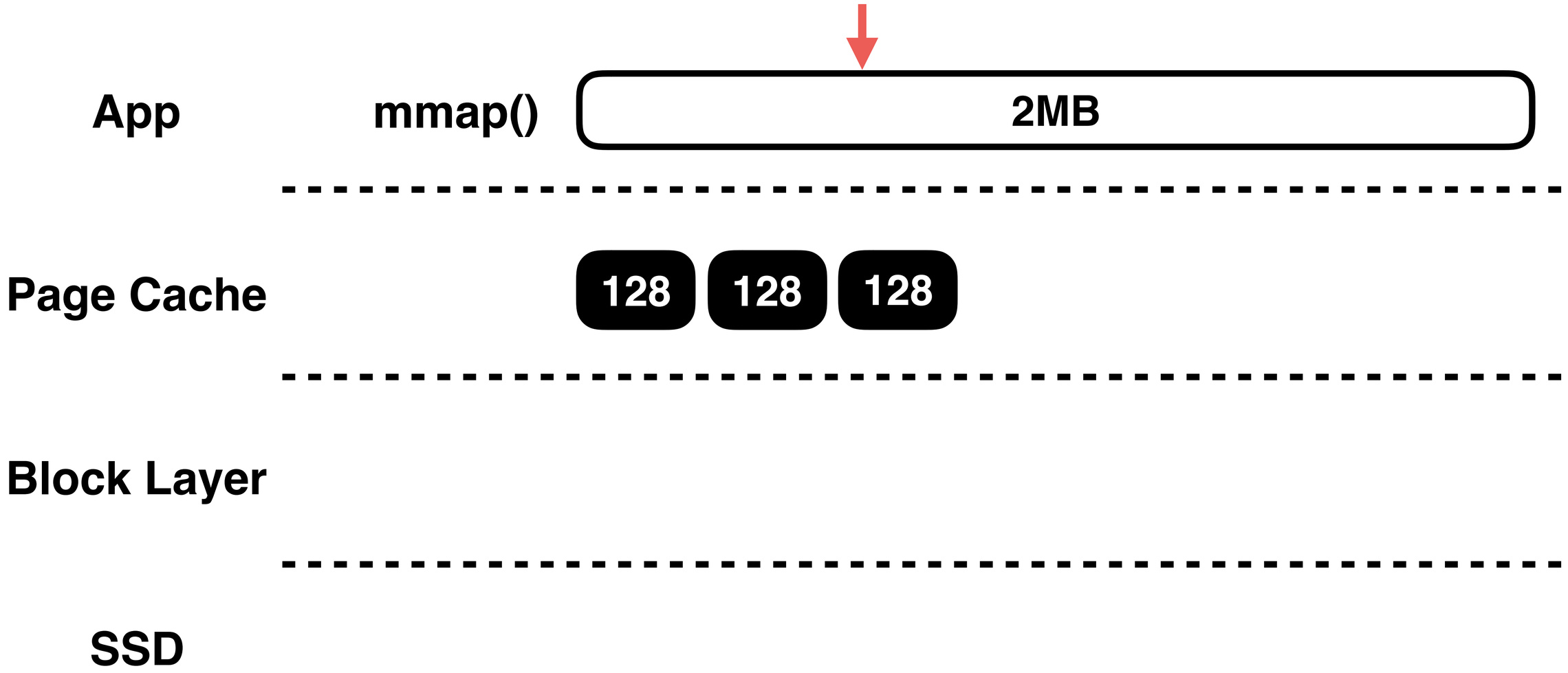
Page cache implementation splits and serializes user requests



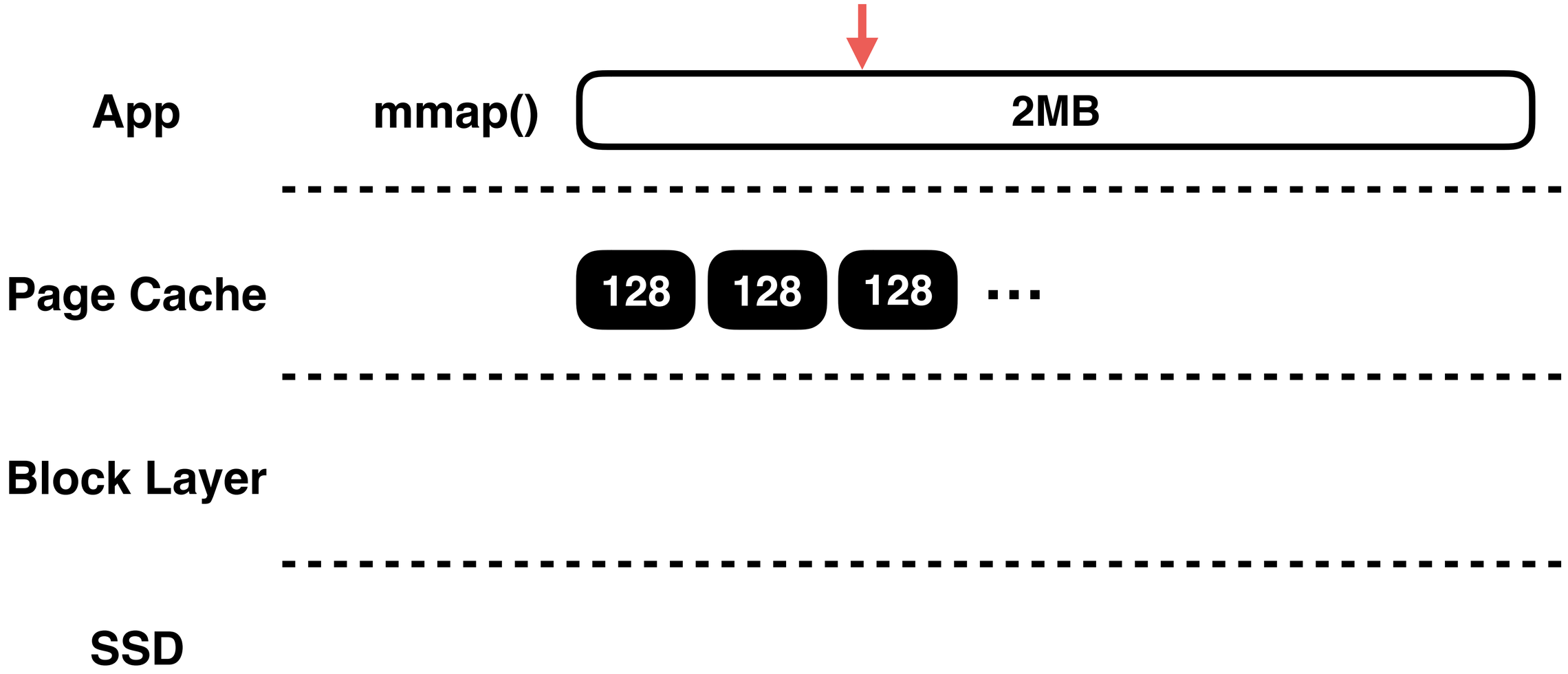
Page cache implementation splits and serializes user requests



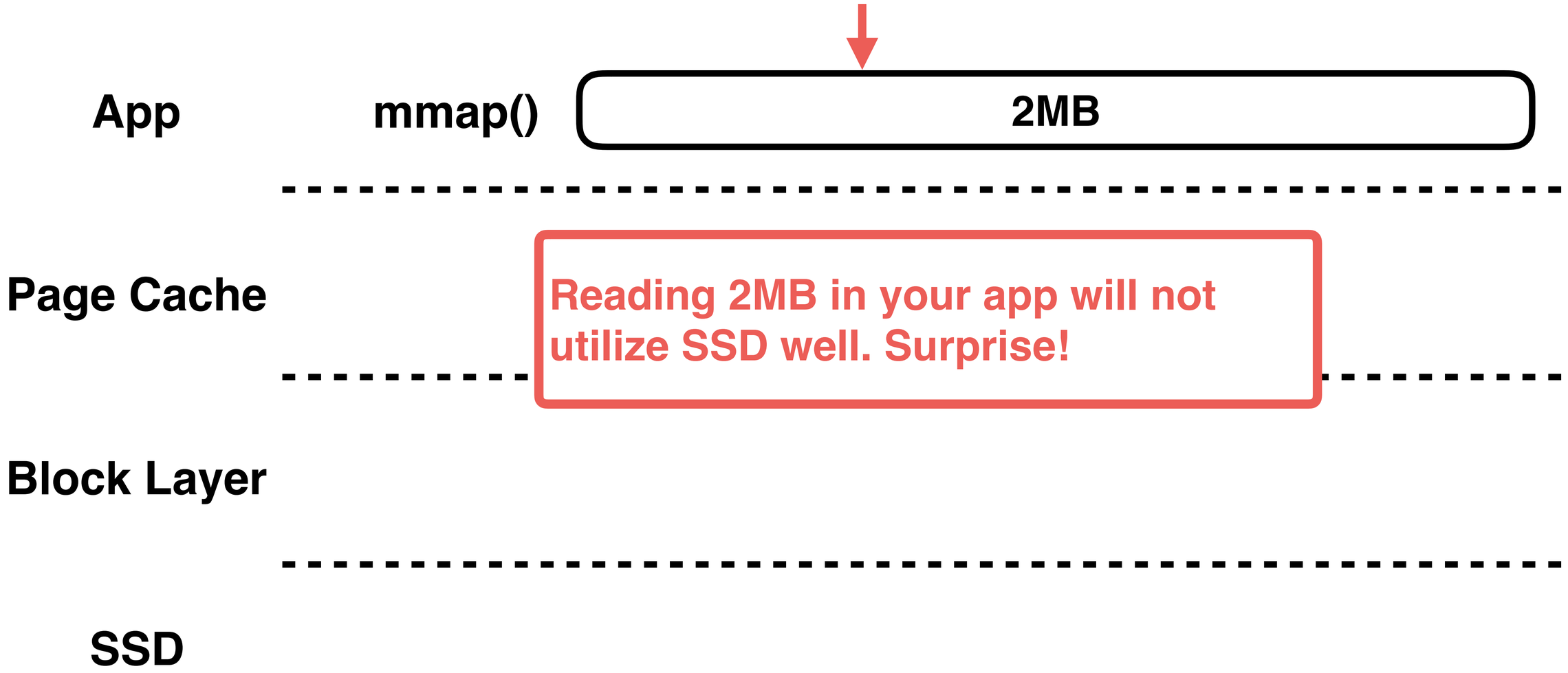
Page cache implementation splits and serializes user requests



Page cache implementation splits and serializes user requests

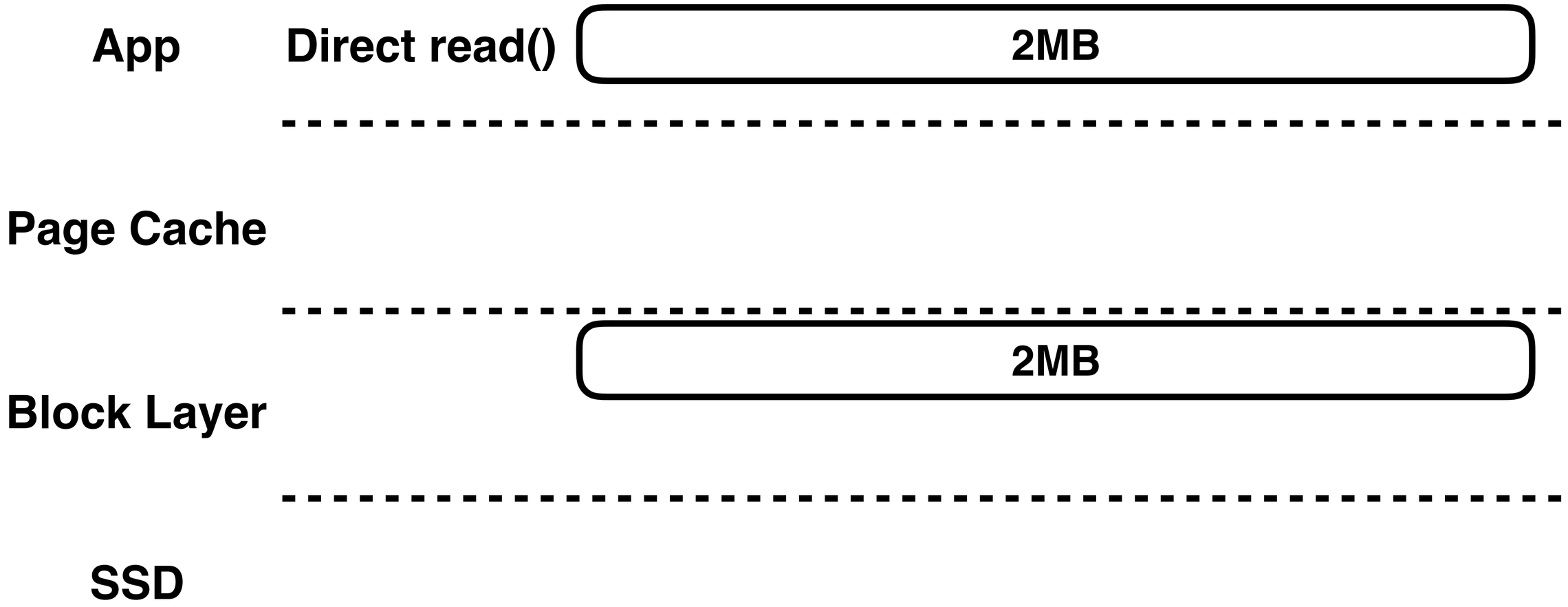


Page cache implementation splits and serializes user requests

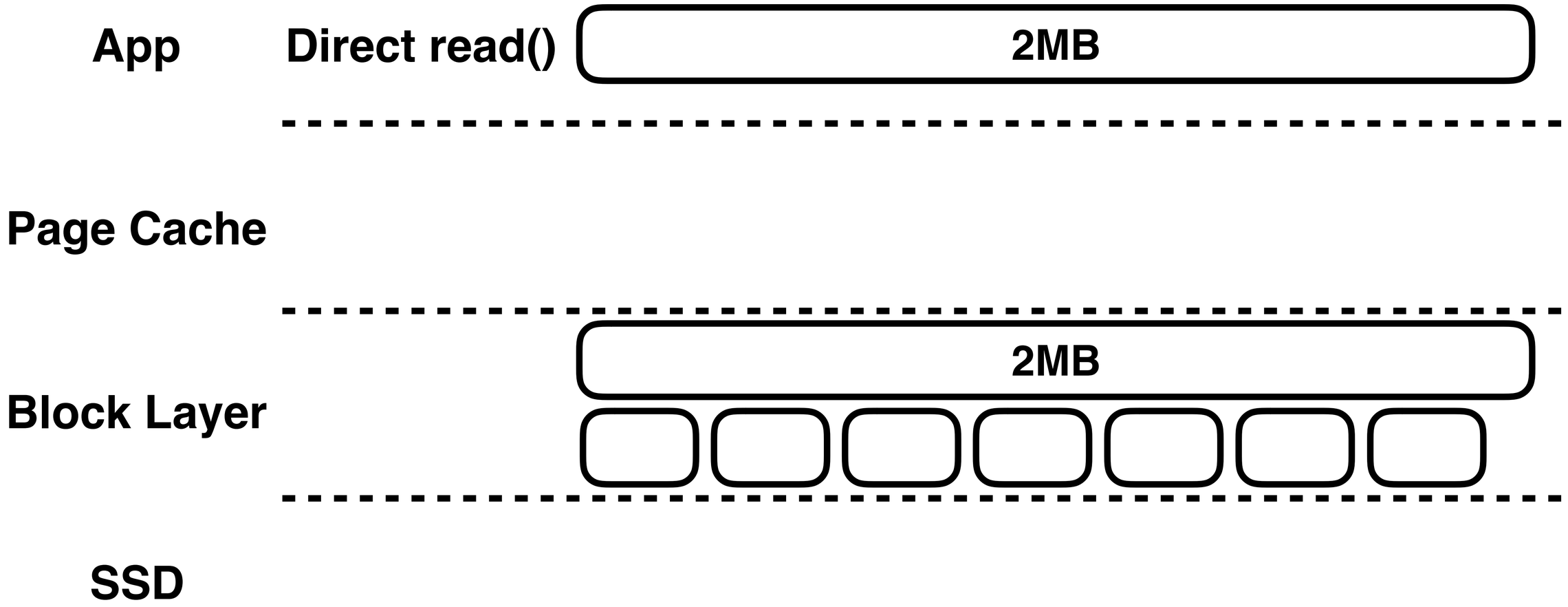


Direct IO sends requests in whole

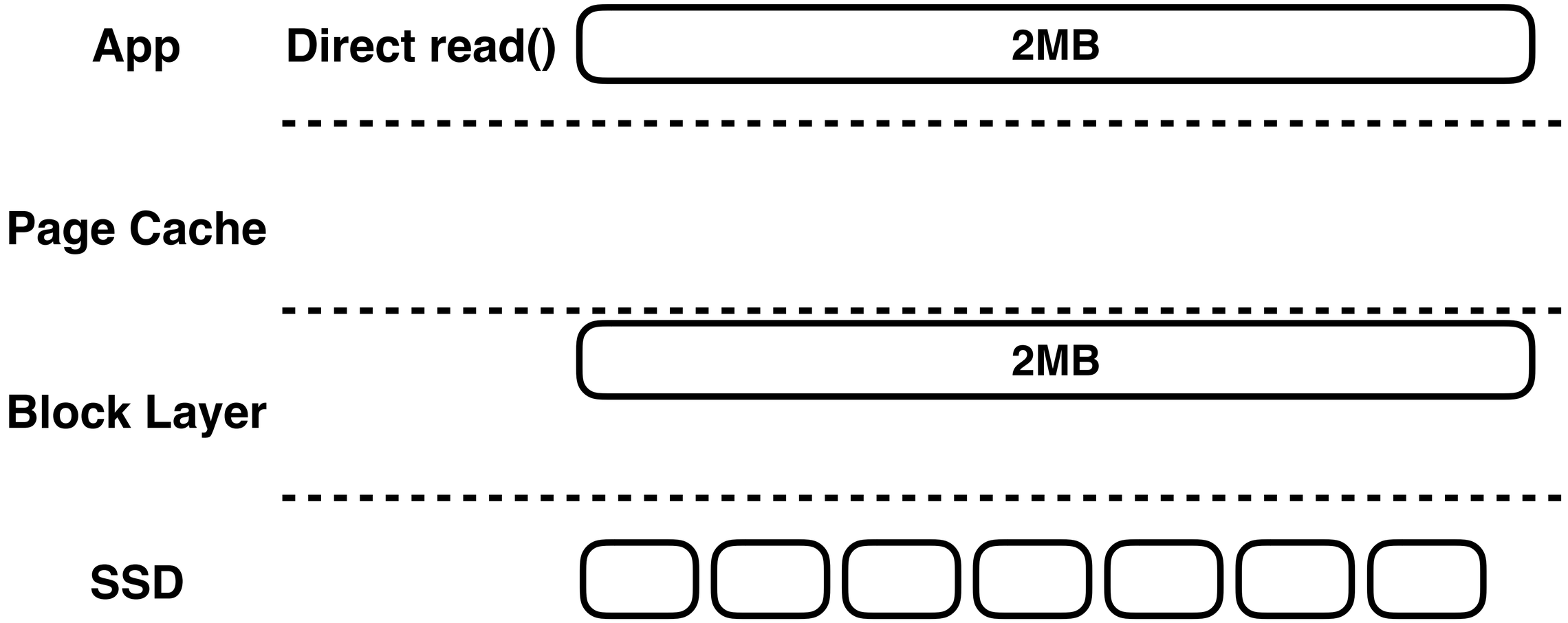
Direct IO sends requests in whole



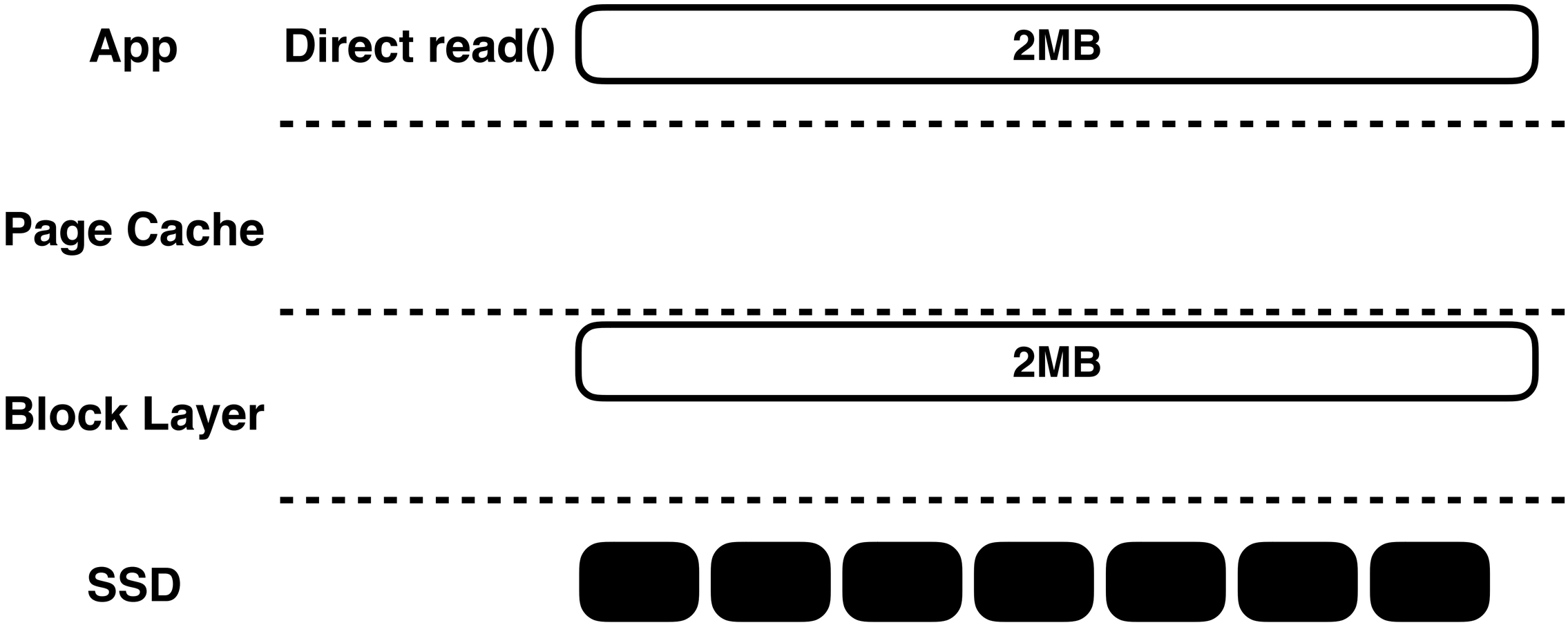
Direct IO sends requests in whole



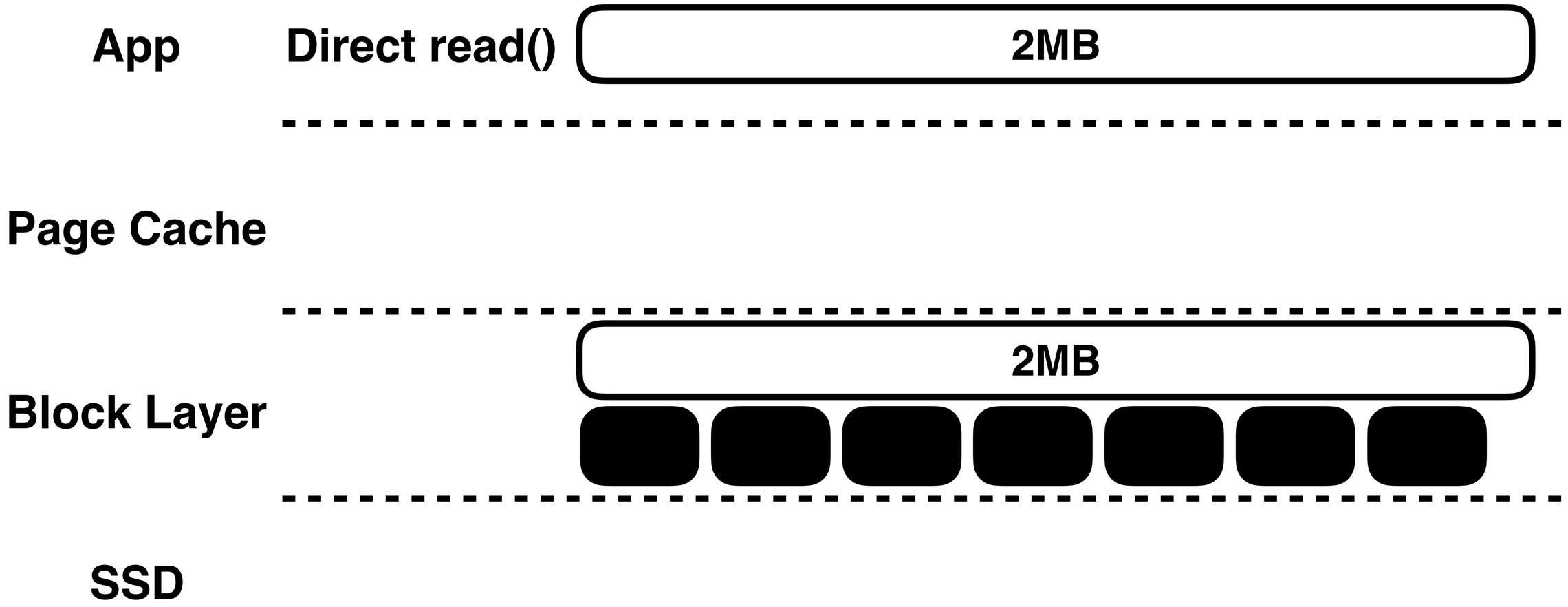
Direct IO sends requests in whole



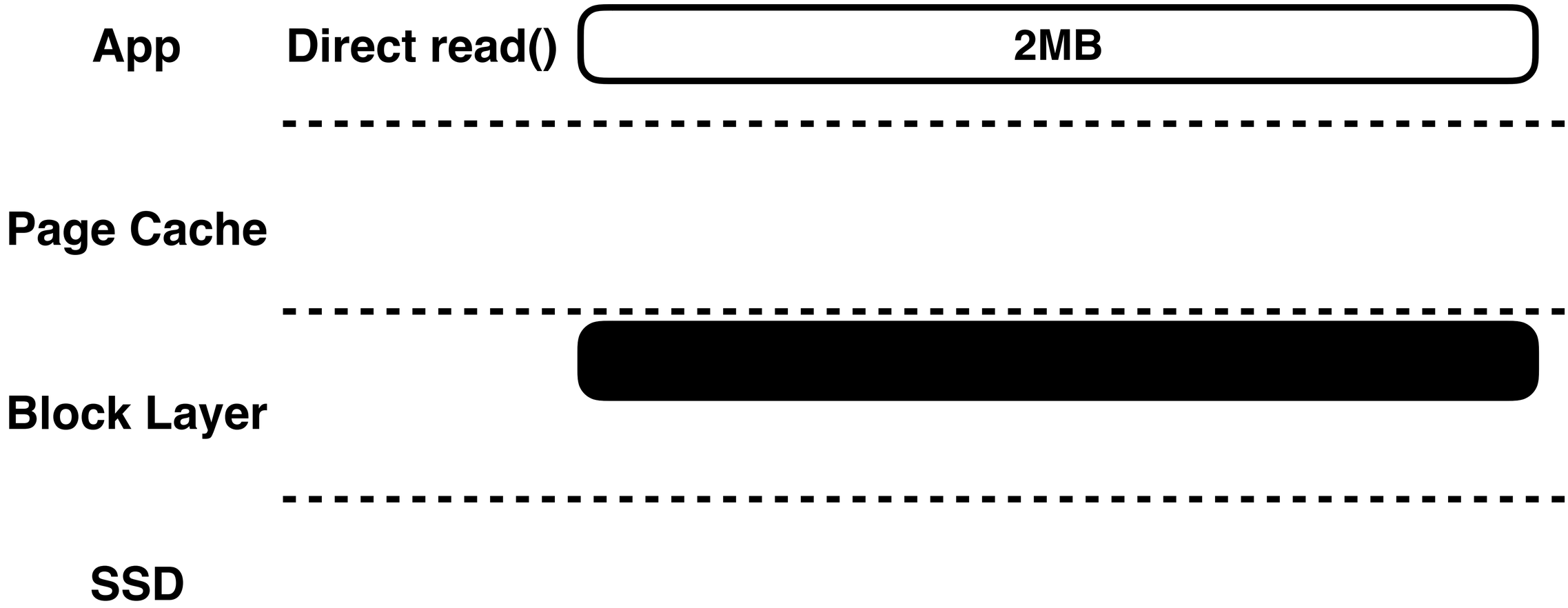
Direct IO sends requests in whole



Direct IO sends requests in whole

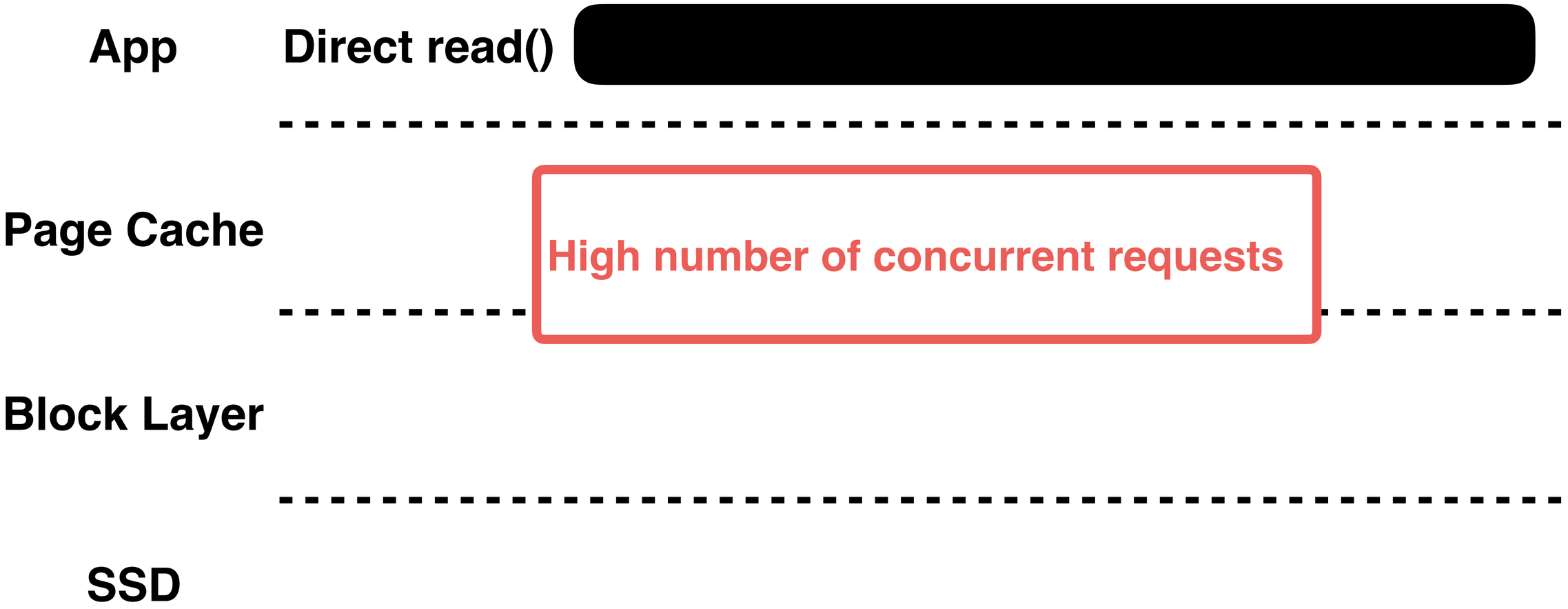


Direct IO sends requests in whole



Direct IO sends requests in whole

Direct IO sends requests in whole



Remedies?

- Increase *read_ahead_kb*
  - It forces other applications to read potentially useless data
  - It is limited by hard-coded bound “2MB”
- Use multiple threads OR async I/O
  - Complicates programming
- Use direct I/O
  - If you do not need page cache

Remedies?

- Increase *read_ahead_kb*
  - It forces other applications to read potentially useless data
  - It is
- Use n
  - Complicates programming
- Use direct I/O
  - If you do not need page cache

Problem to fix:
Large read is throttled by small prefetching(readahead)

A File System Summary for Request Scale

- ext4 & XFS have very similar request scale
- F2FS is very different to ext4 and XFS
  - F2FS could break data structures of applications
  - F2FS boosts request scale by delaying discards

Observations

- Rule 1: Request Scale
- Implementation of Linux limits performance
- Rule 3: Grouping by Death Time
  - F2FS incurs more GC overhead than ext4/XFS
  - Application log structuring does not reduce GC

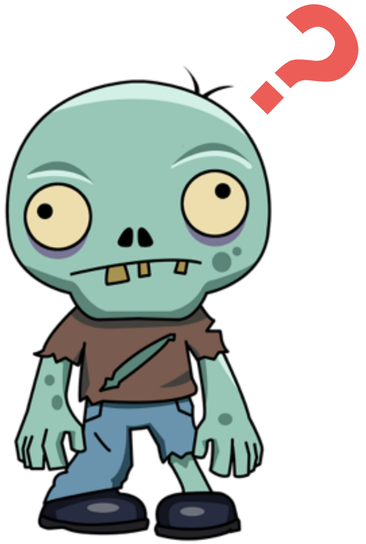
Observations

- Rule 1: Request Scale
  - Implementation of Linux limits performance
- Rule 3: Grouping by Death Time
  - F2FS incurs more GC overhead than ext4/XFS
  - Application log structuring does not reduce GC

We study Rule 3 (grouping by death time) by zombie curves

We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?



We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?

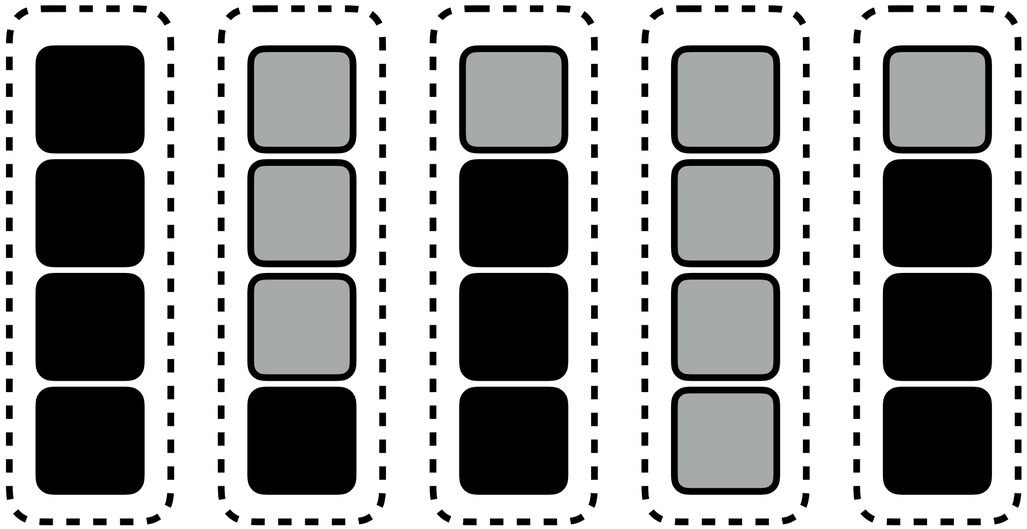
We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?

Run workloads with infinite space over-provisioning

We study Rule 3 (grouping by death time) by zombie curves

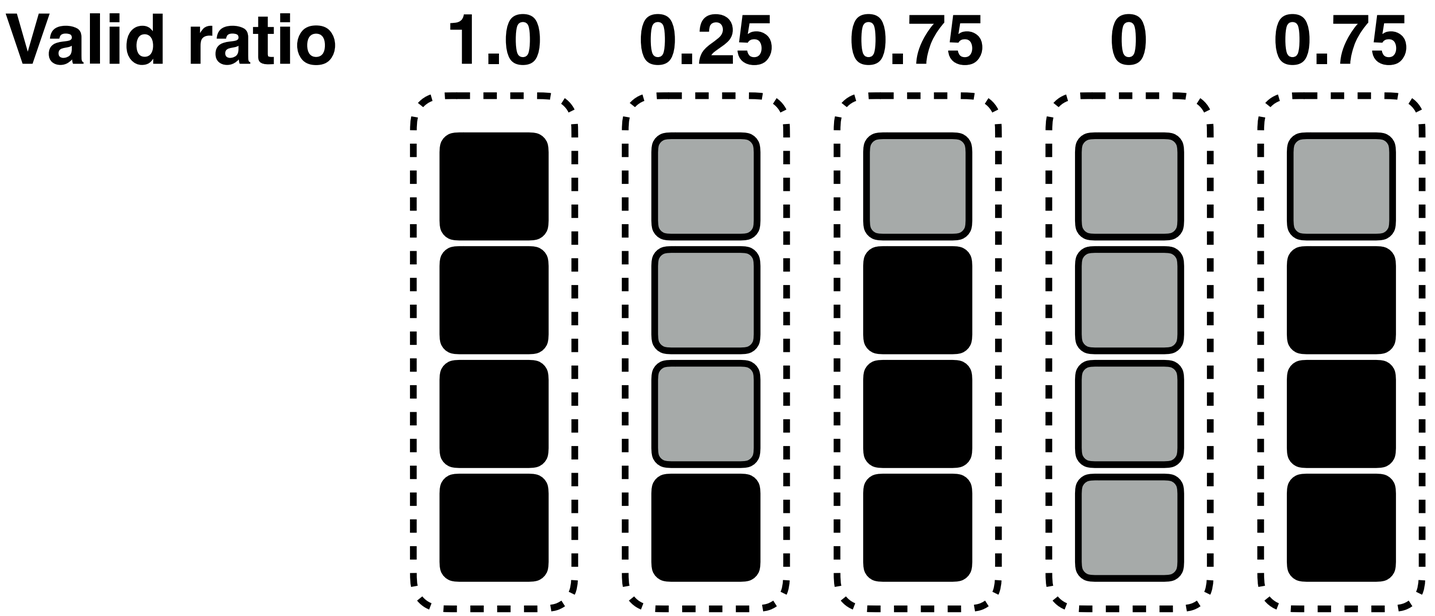
What's a zombie curve?



Run workloads with infinite space over-provisioning

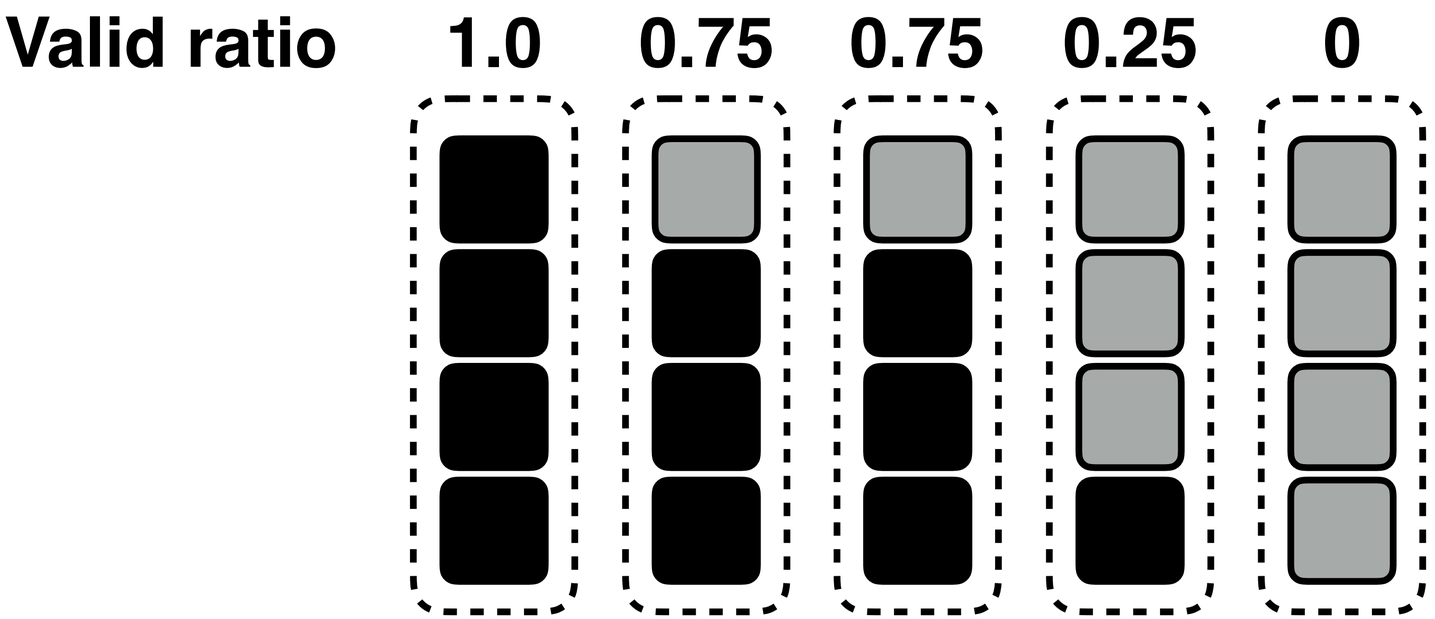
We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?



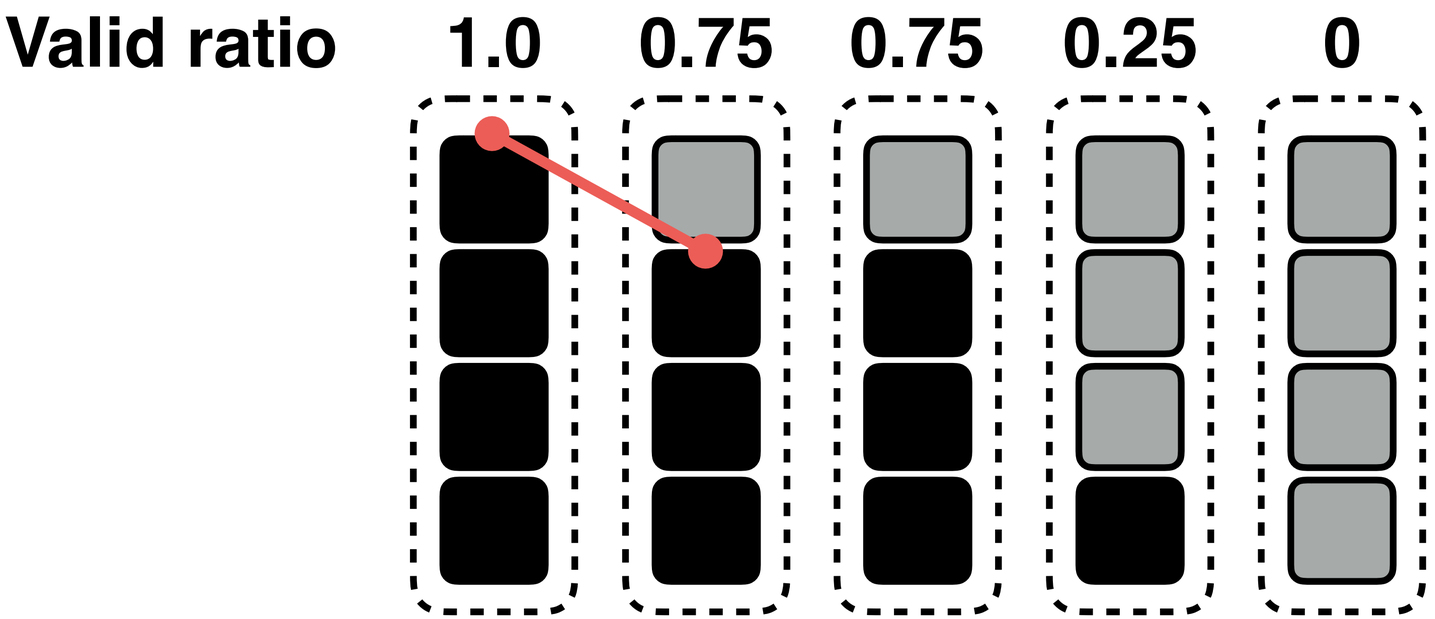
We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?



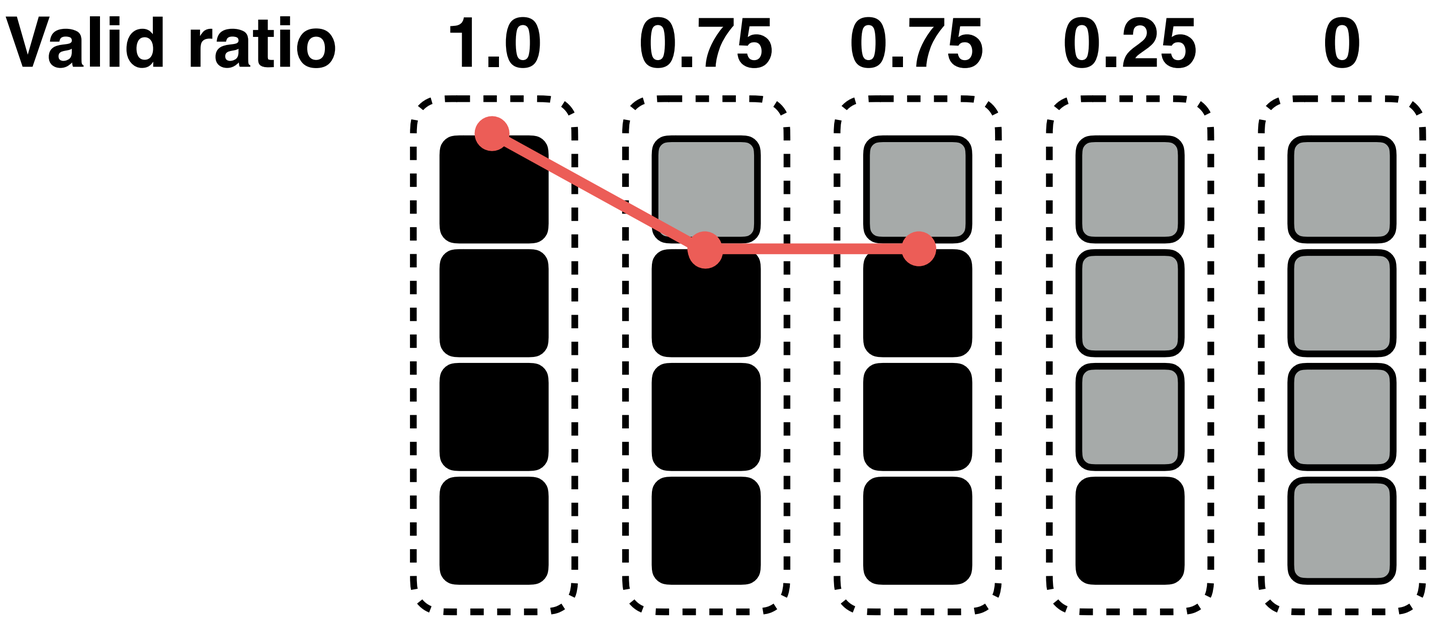
We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?



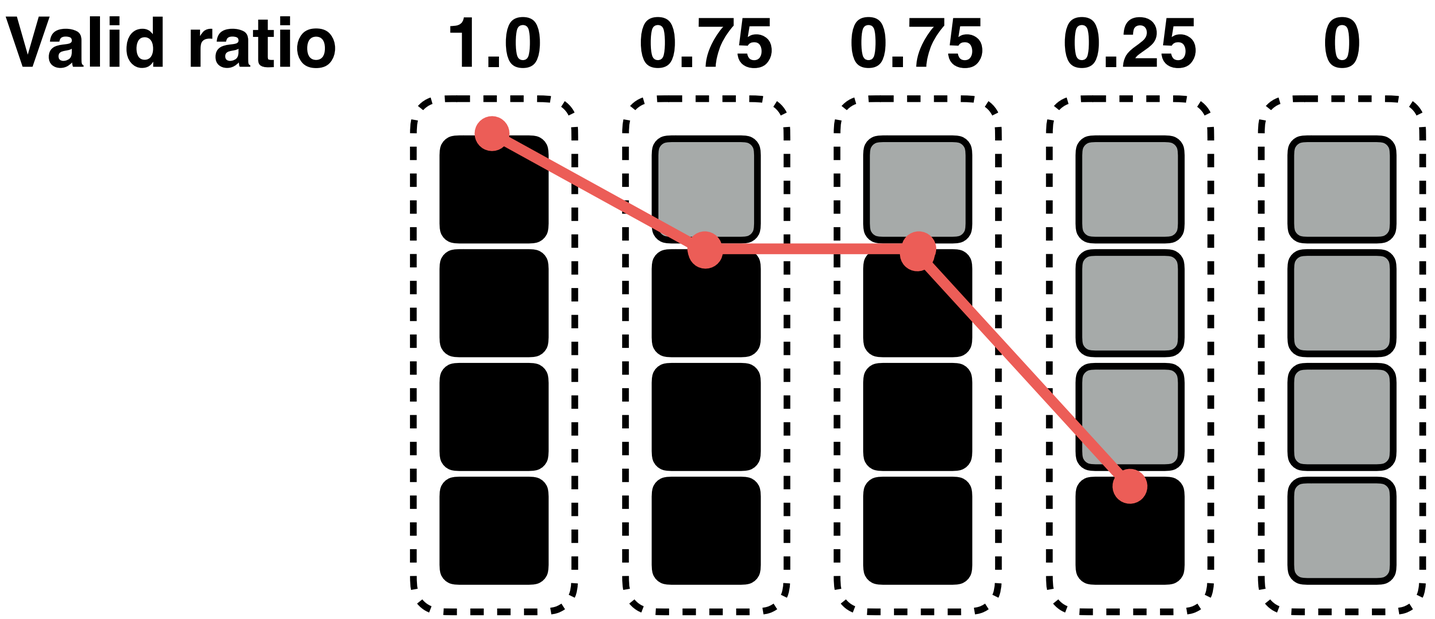
We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?



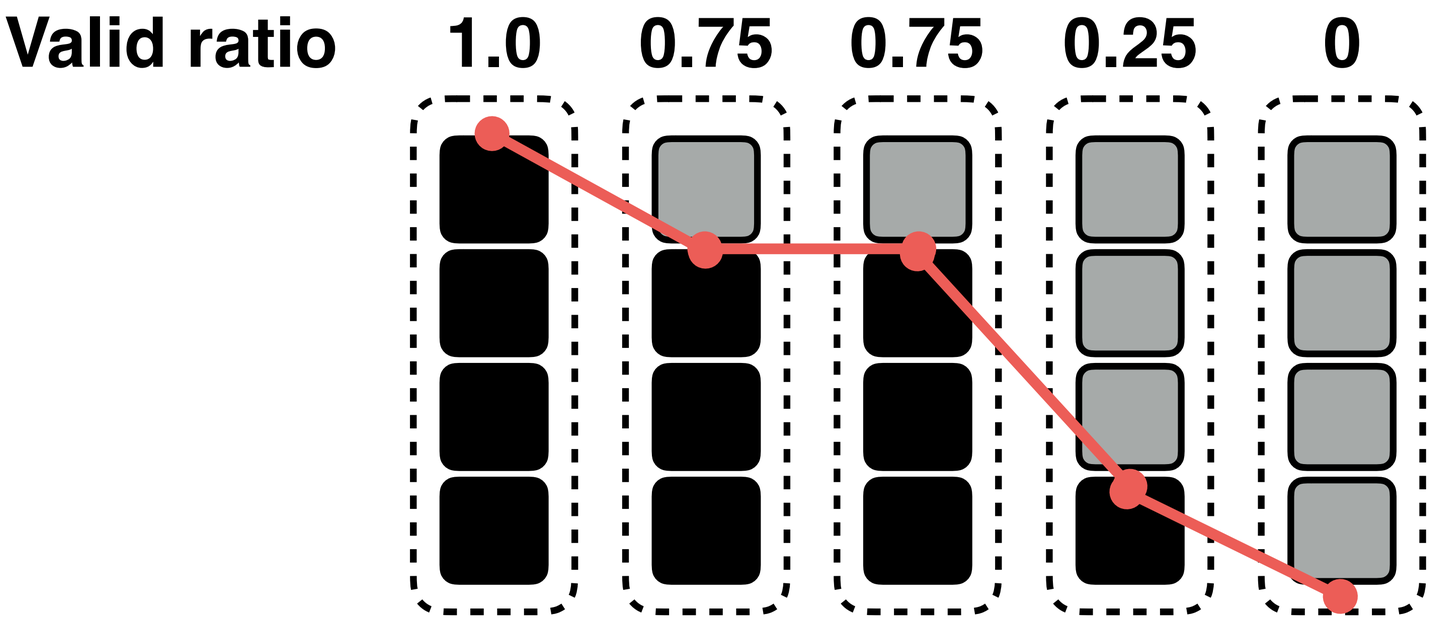
We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?



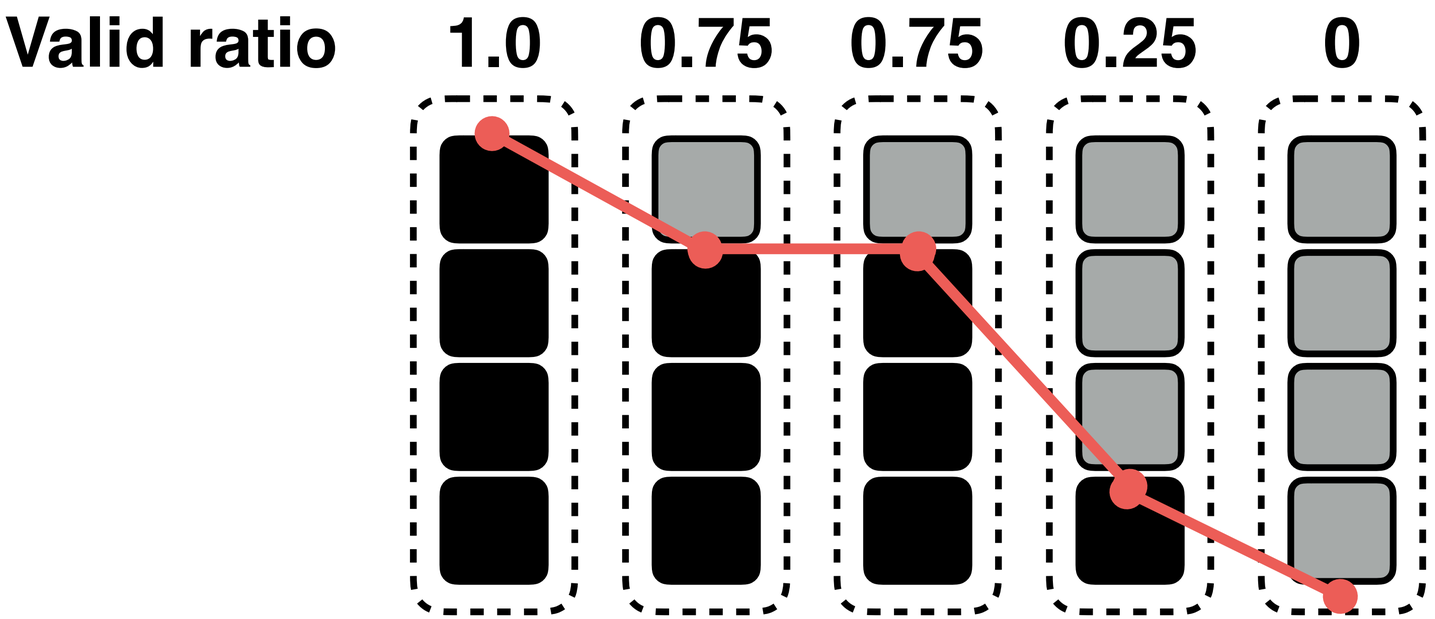
We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?



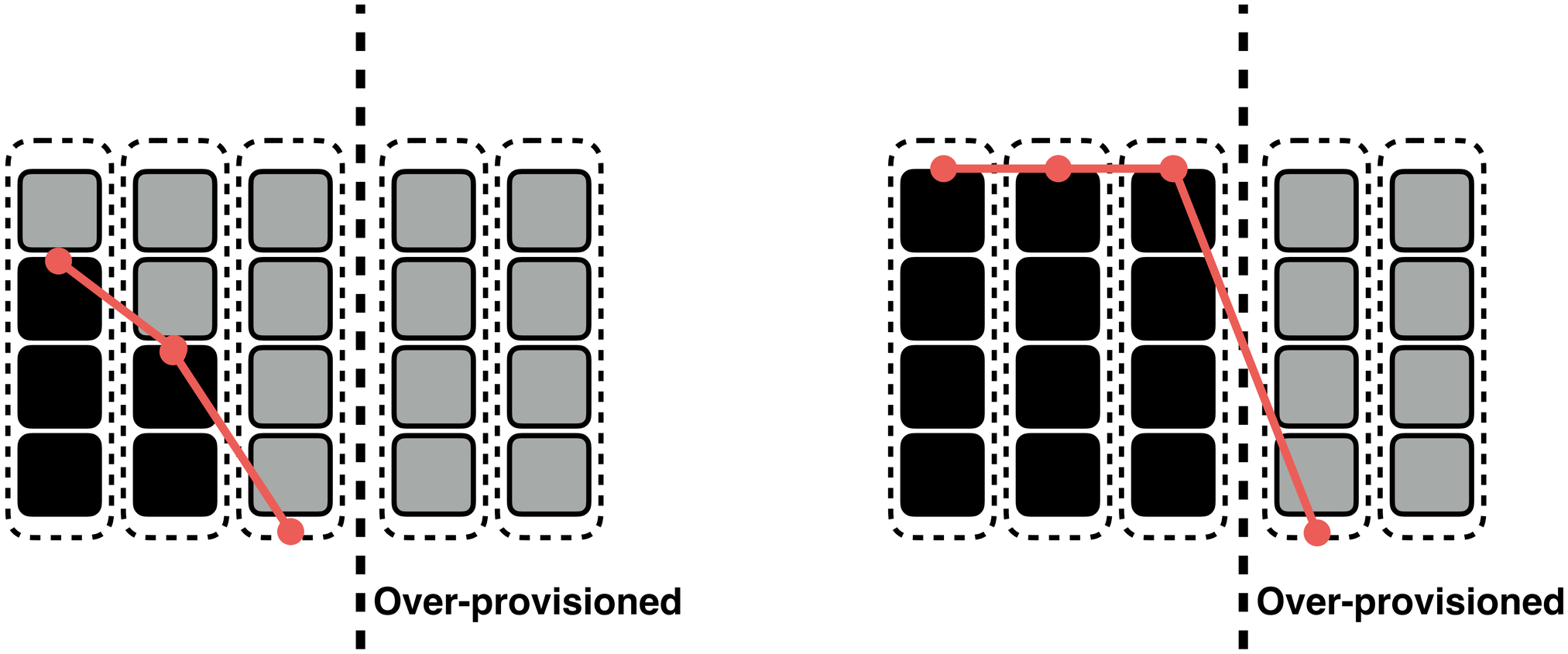
We study Rule 3 (grouping by death time) by zombie curves

What's a zombie curve?

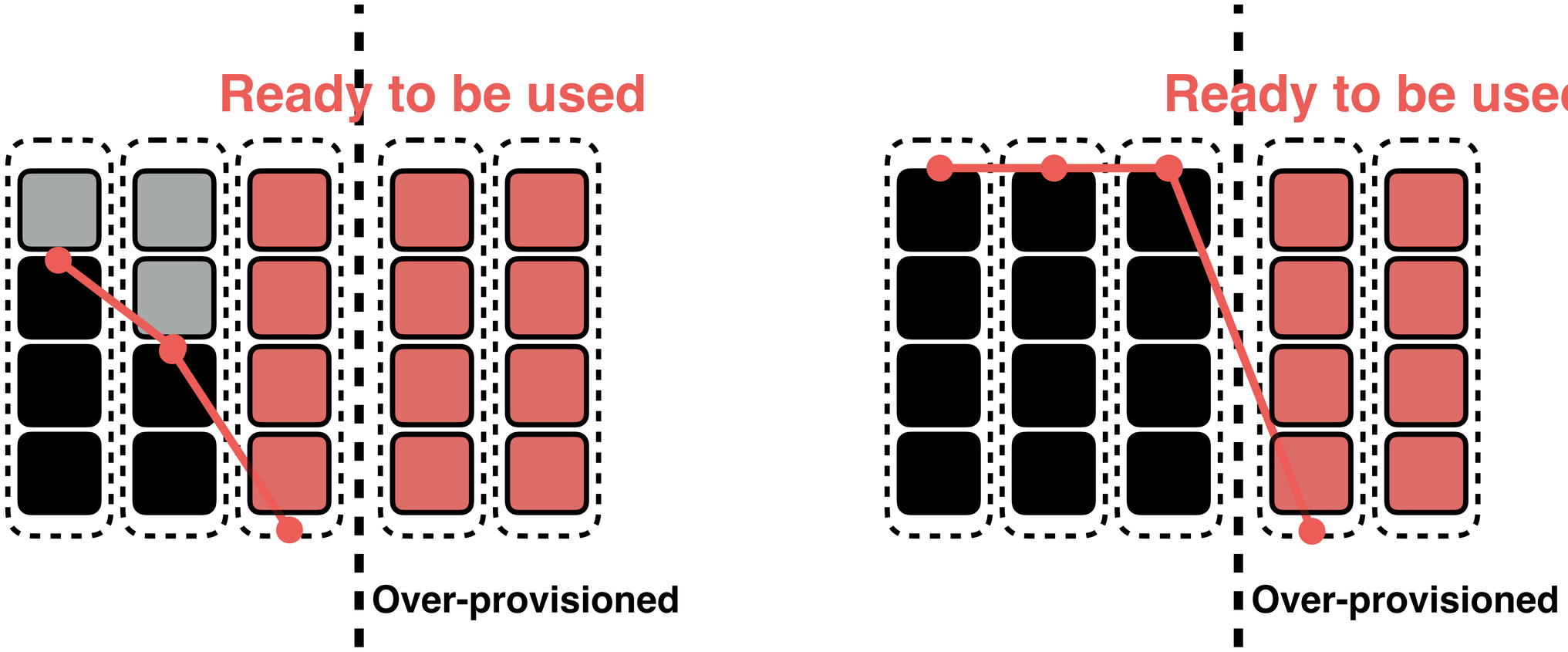


Zombie blocks are the blocks with partially valid pages.

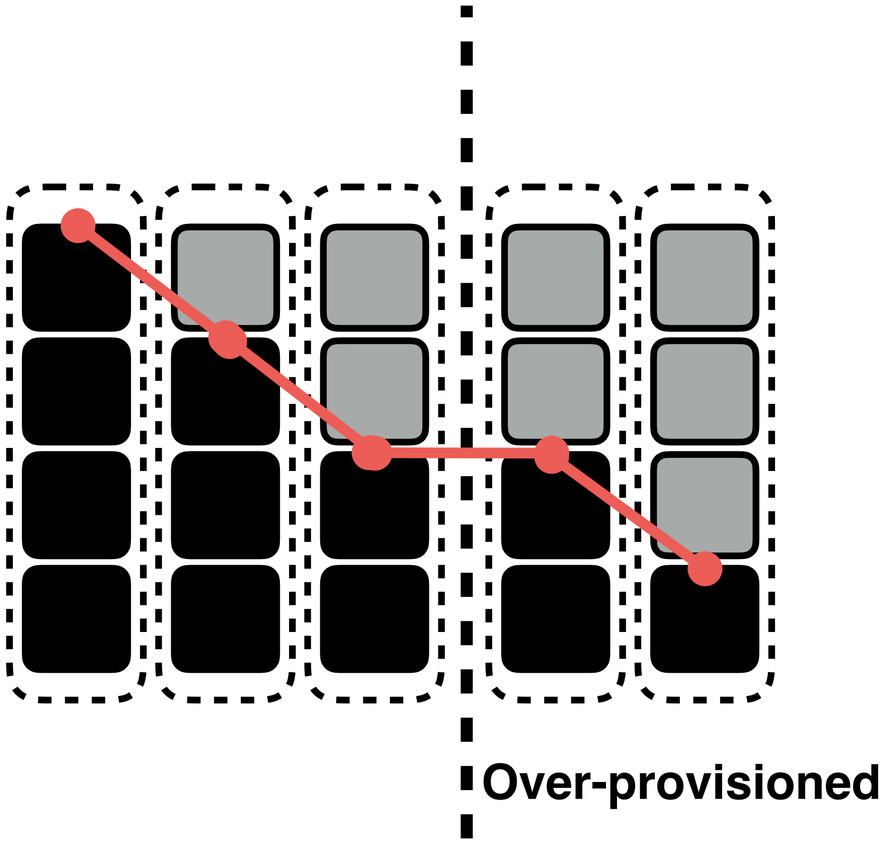
What’s a good zombie curve?



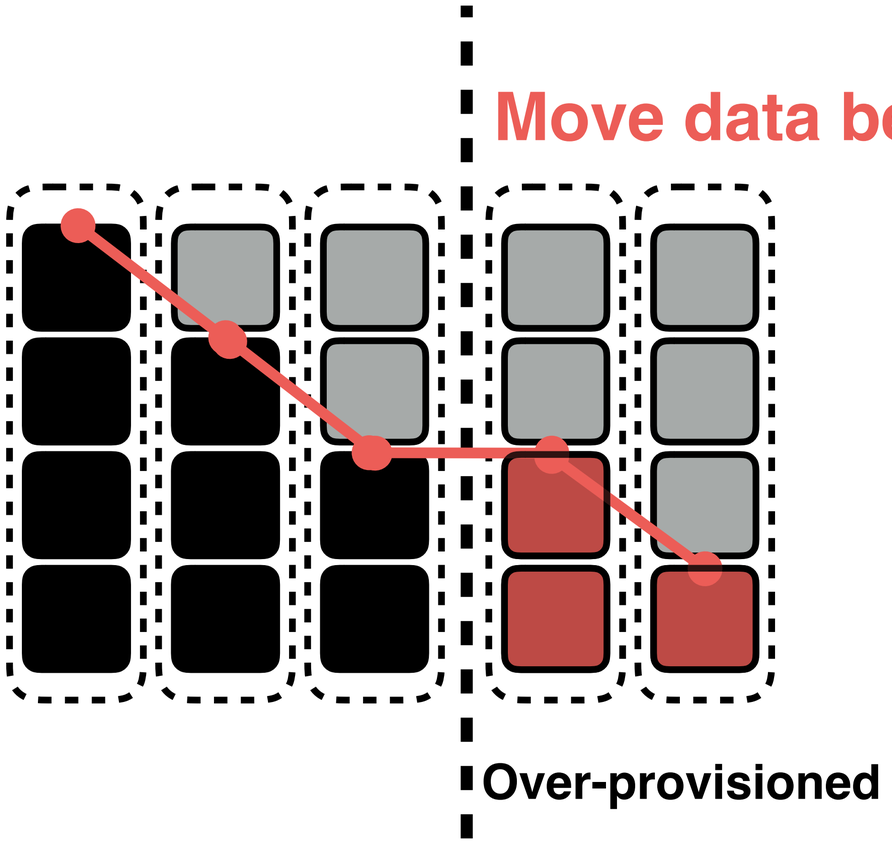
What’s a good zombie curve?



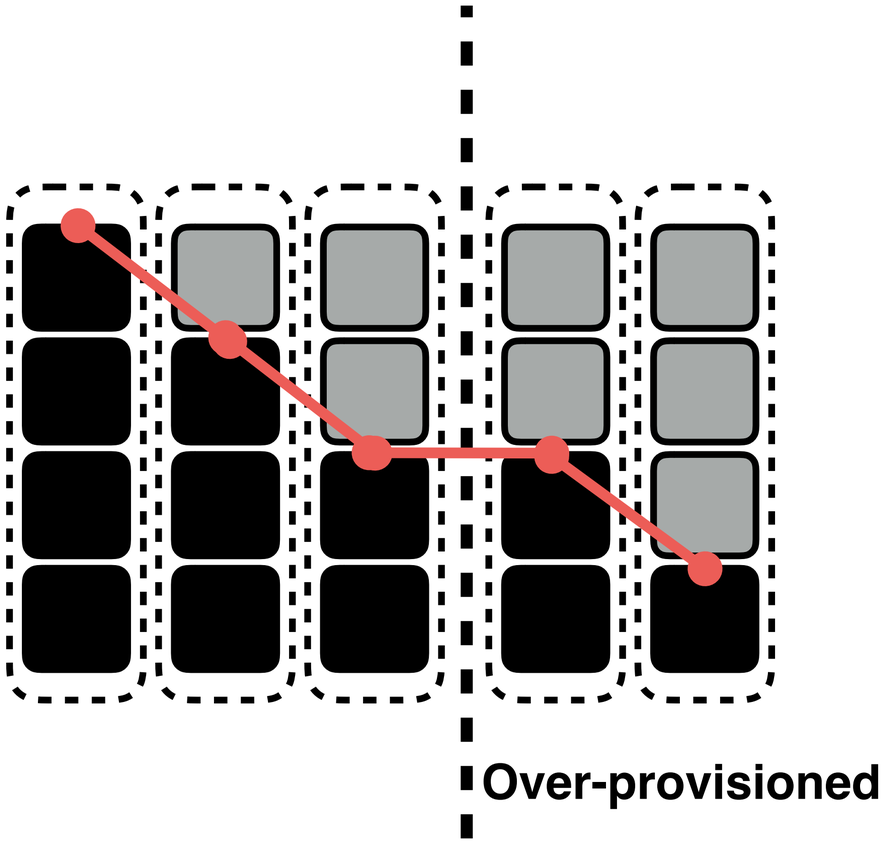
What’s a bad zombie curve?



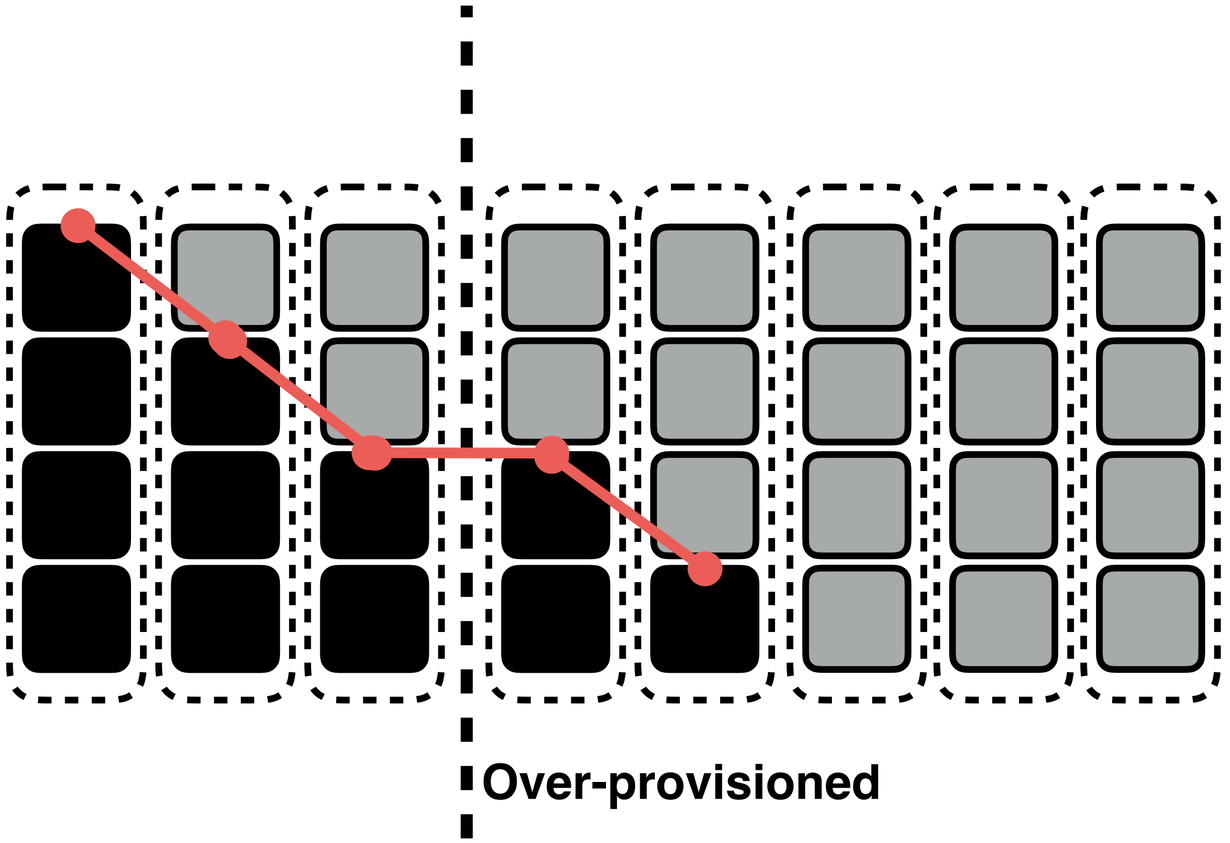
What’s a bad zombie curve?



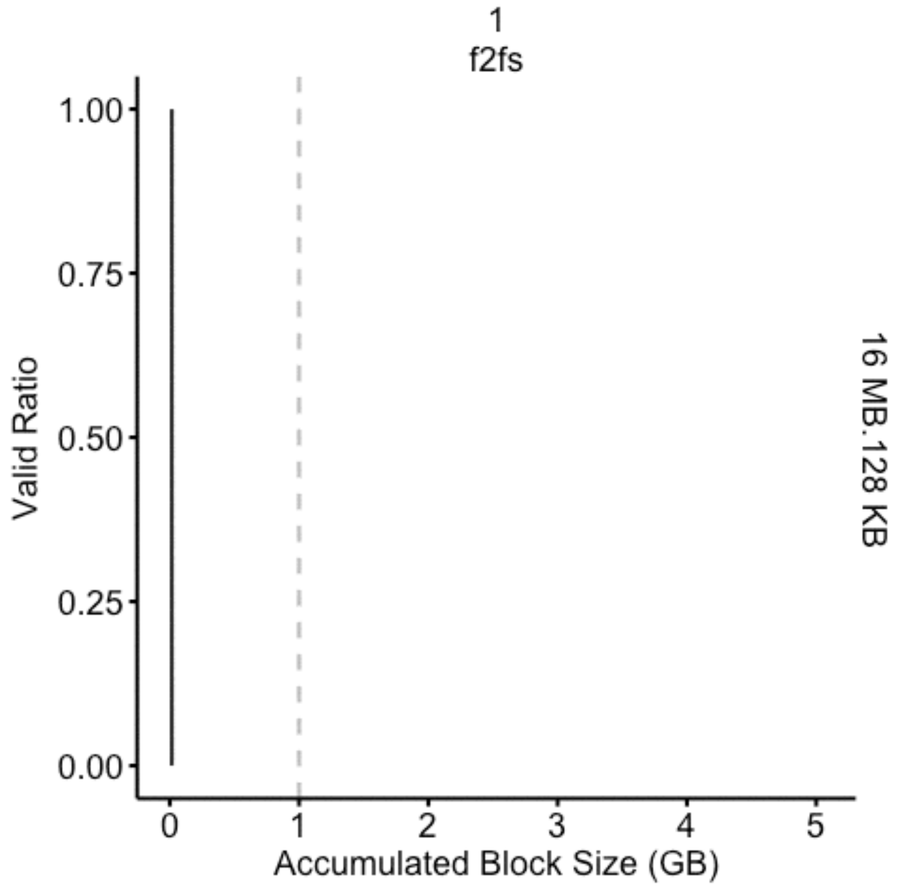
BTW, zombie curve helps you choose over-provisioning ratio



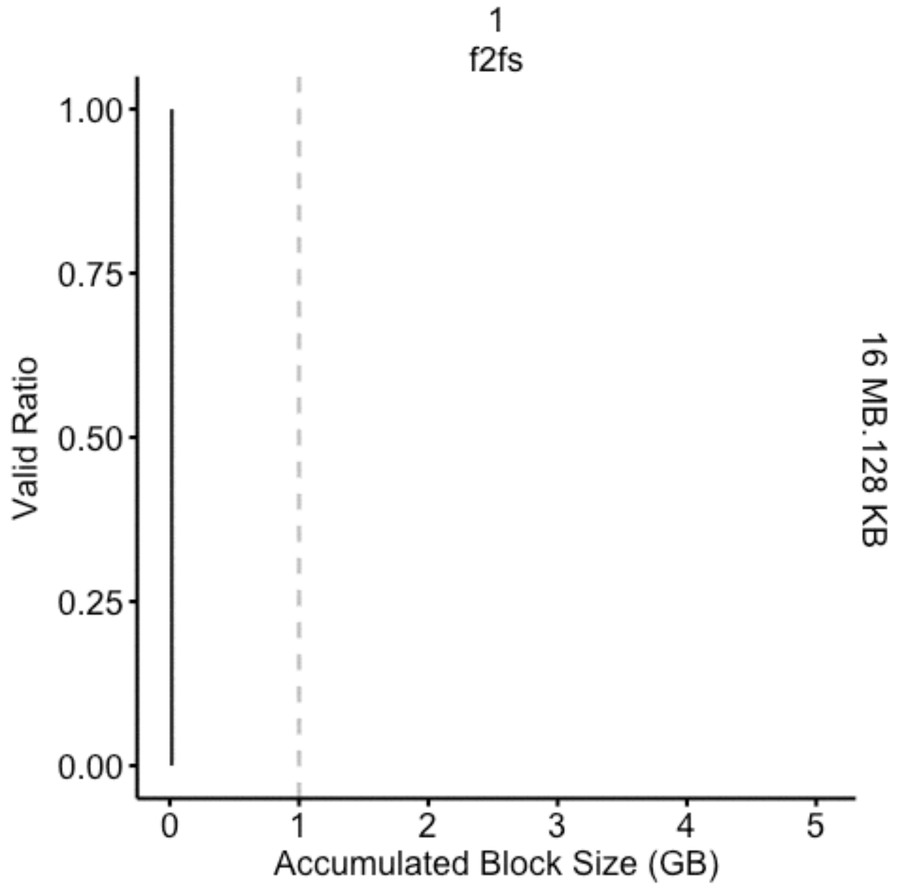
BTW, zombie curve helps you choose over-provisioning ratio

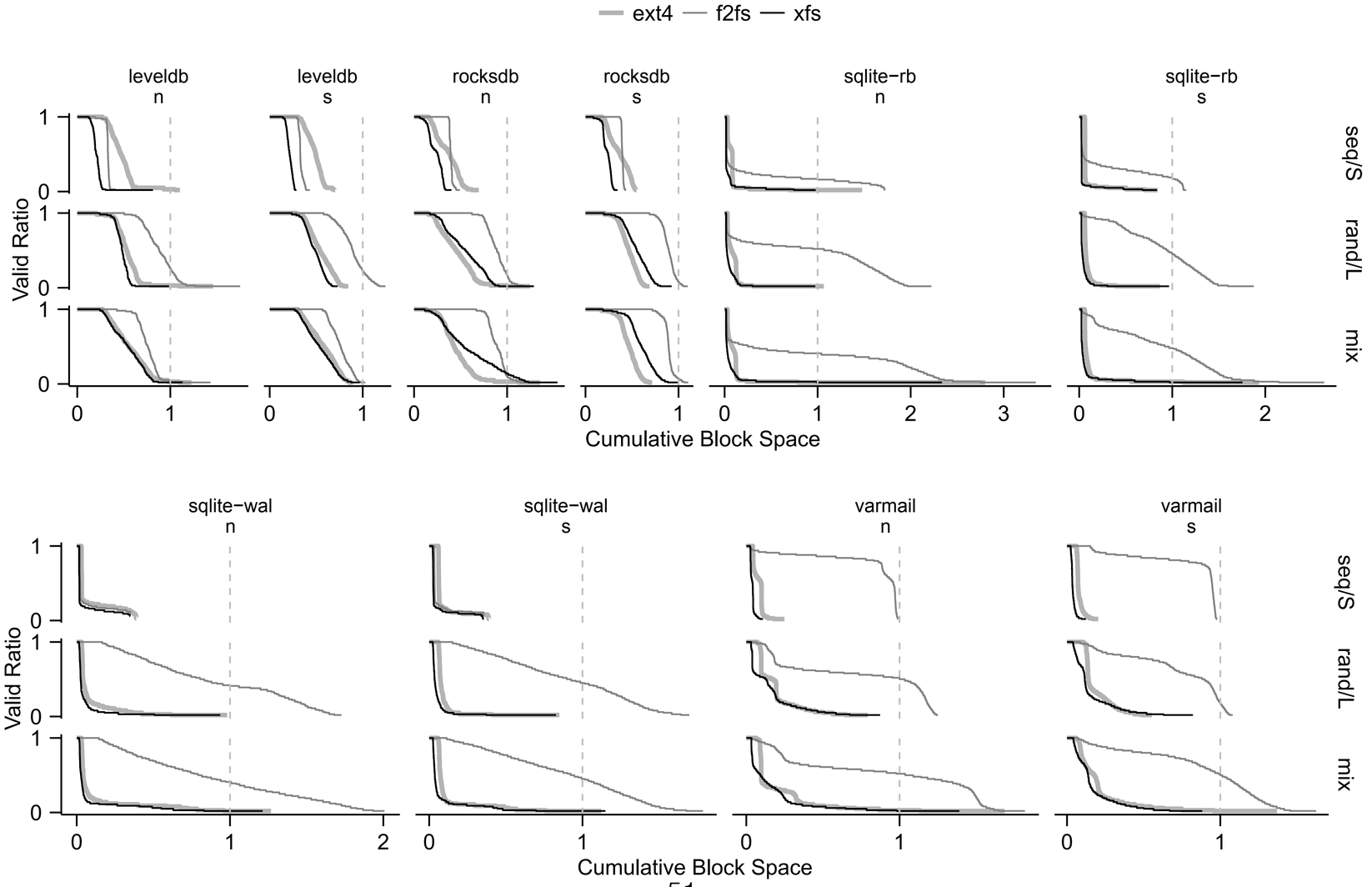


We use stable zombie curves to characterize workloads

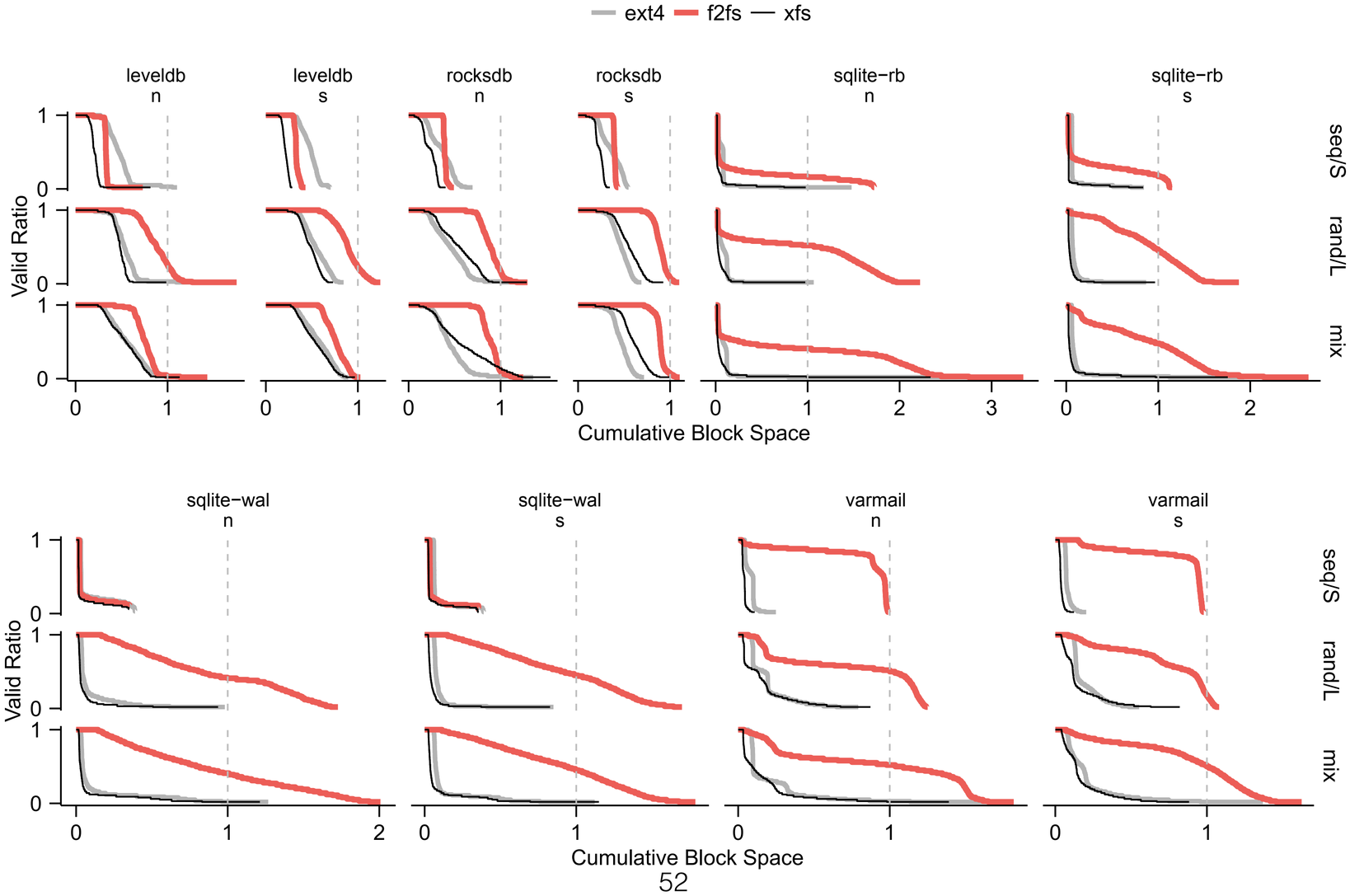


We use stable zombie curves to characterize workloads

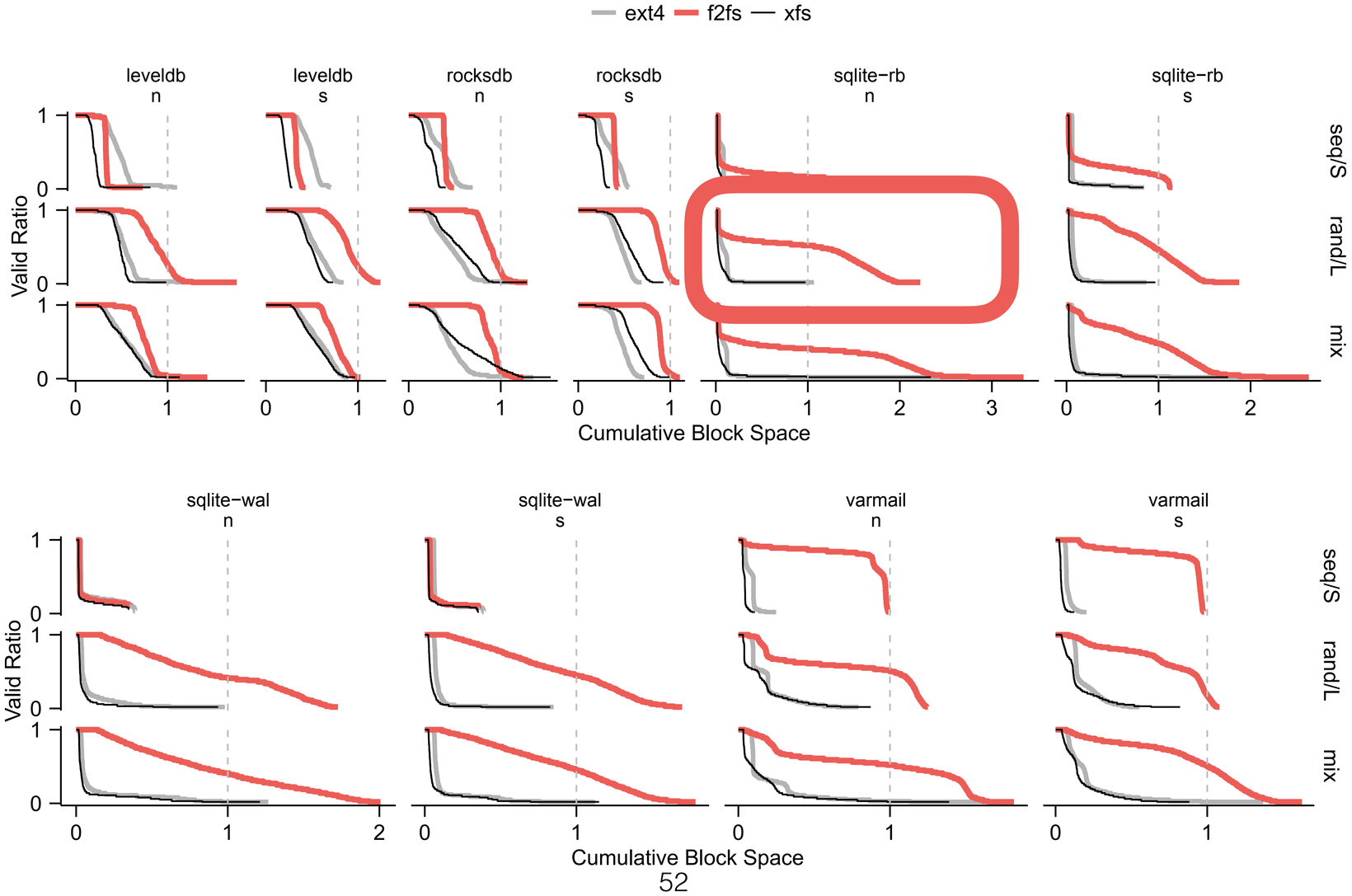



ext4
f2fs
xfs
leveldb n
leveldb s
rocksdb n
rocksdb s
sqlite−rb n
sqlite−rb s
sqlite−wal n
sqlite−wal s
varmail n
varmail s
seq/S
rand/L
mix
Valid Ratio
Cumulative Block Space

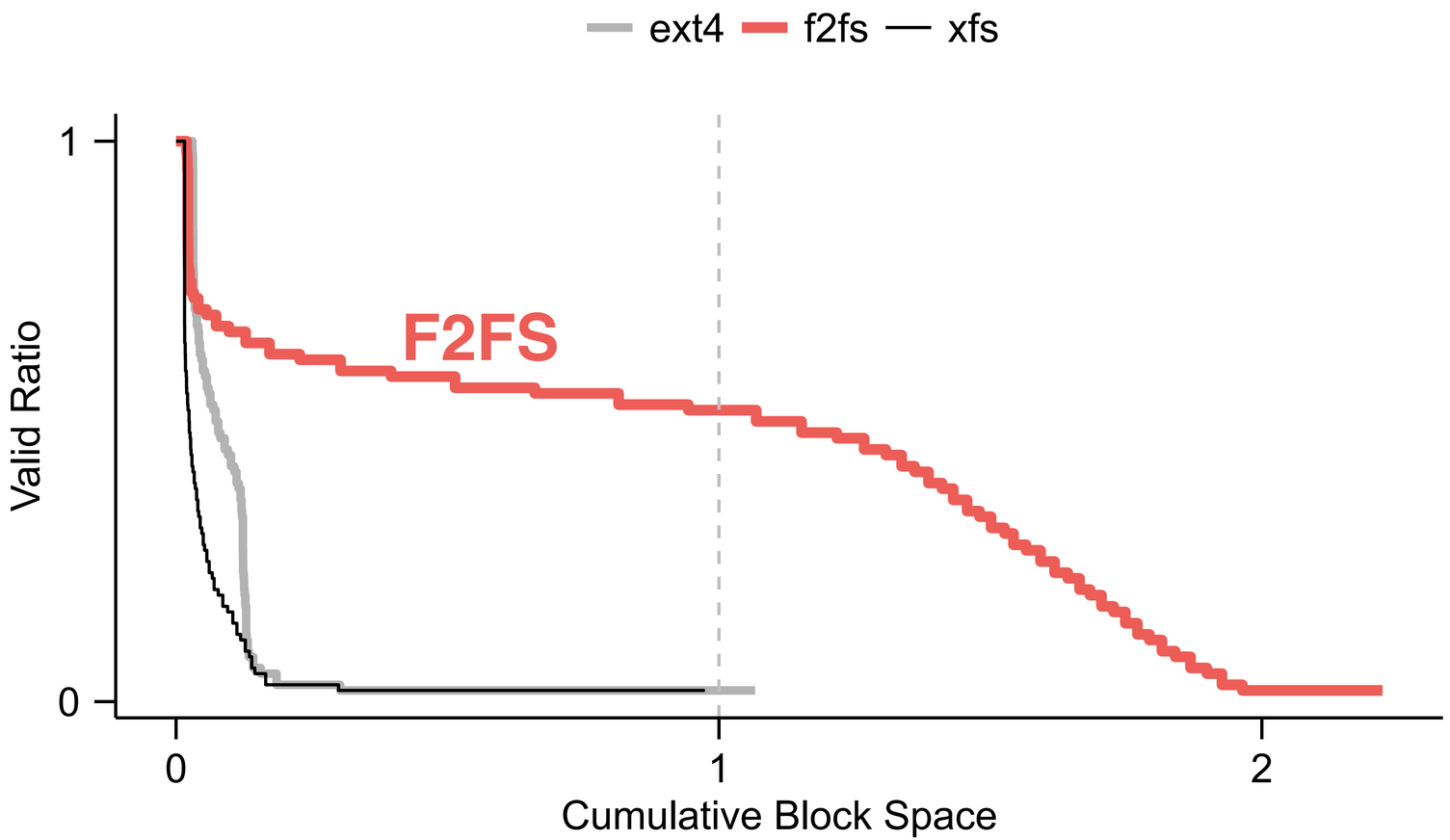
F2FS often has bad zombie curves



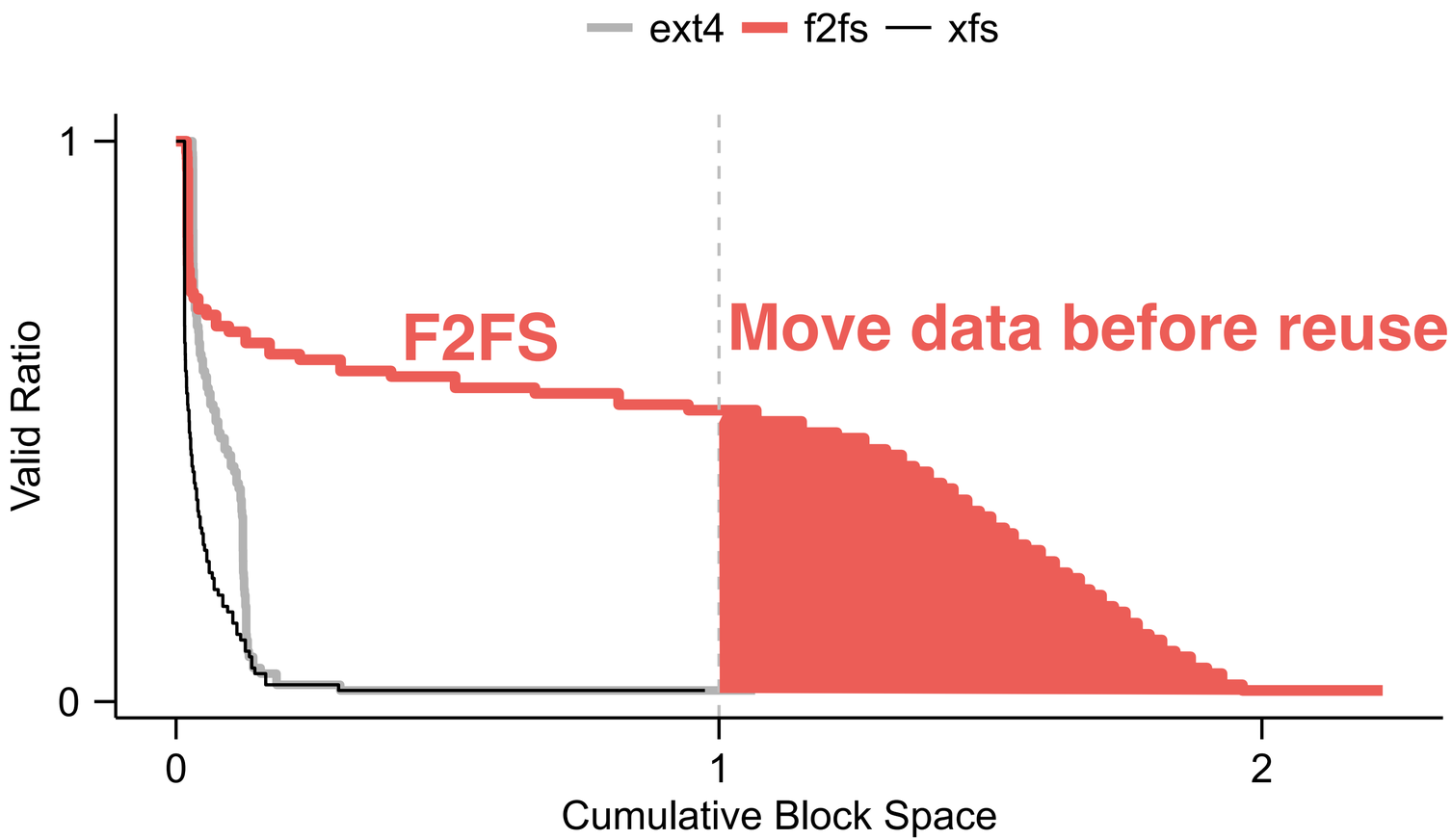
F2FS often has bad zombie curves



Random Insertions to SQLite-RollBack



Random Insertions to SQLite-RollBack



How SQLite-RollBack commit works?

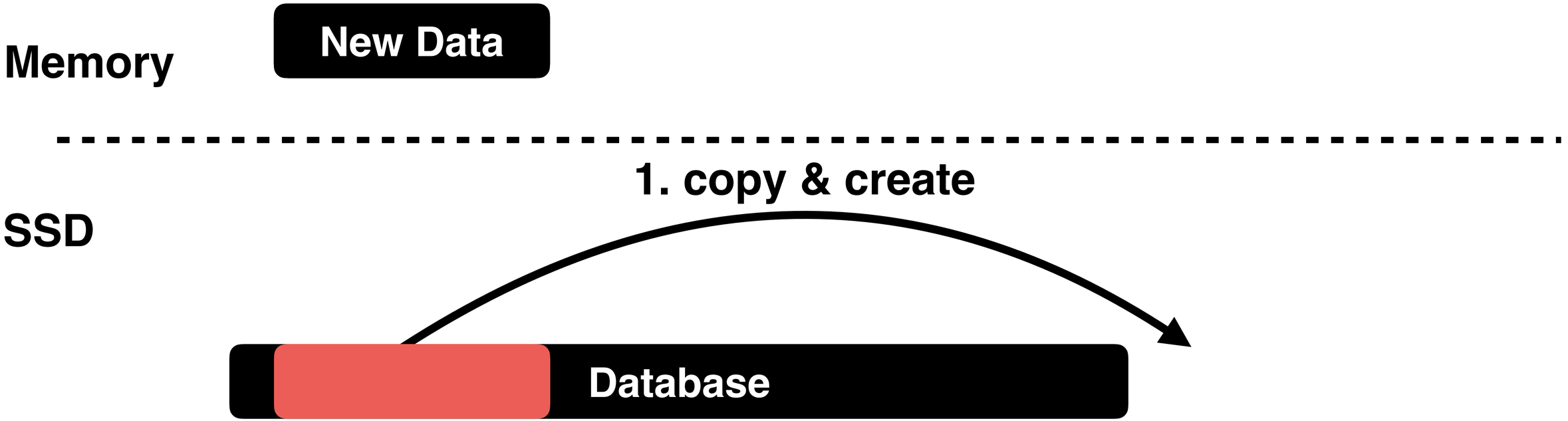
Memory

SSD

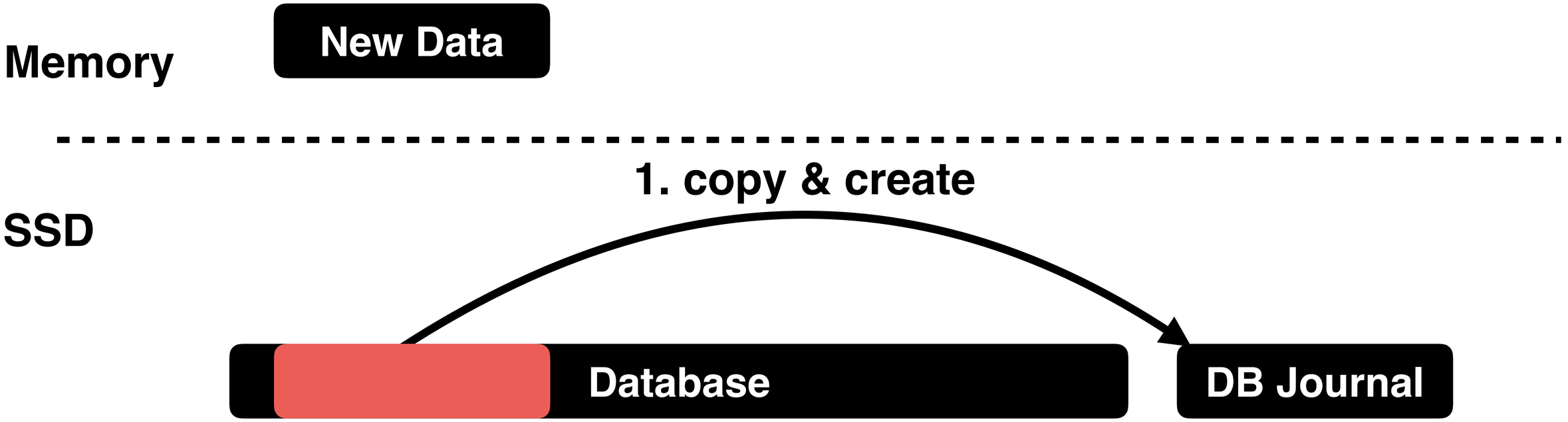
Database

How SQLite-RollBack commit works?

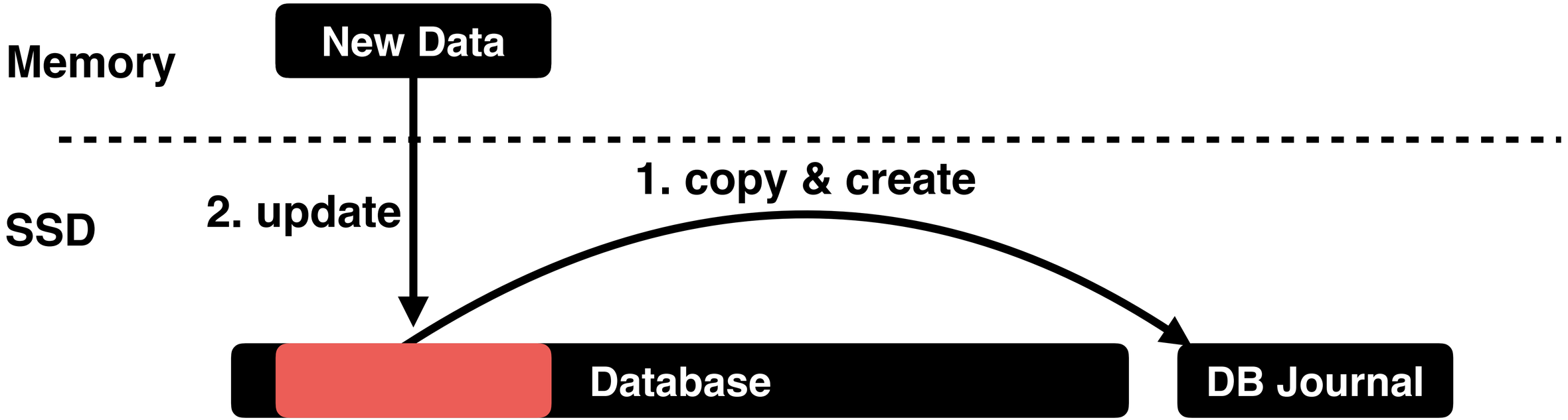
How SQLite-RollBack commit works?



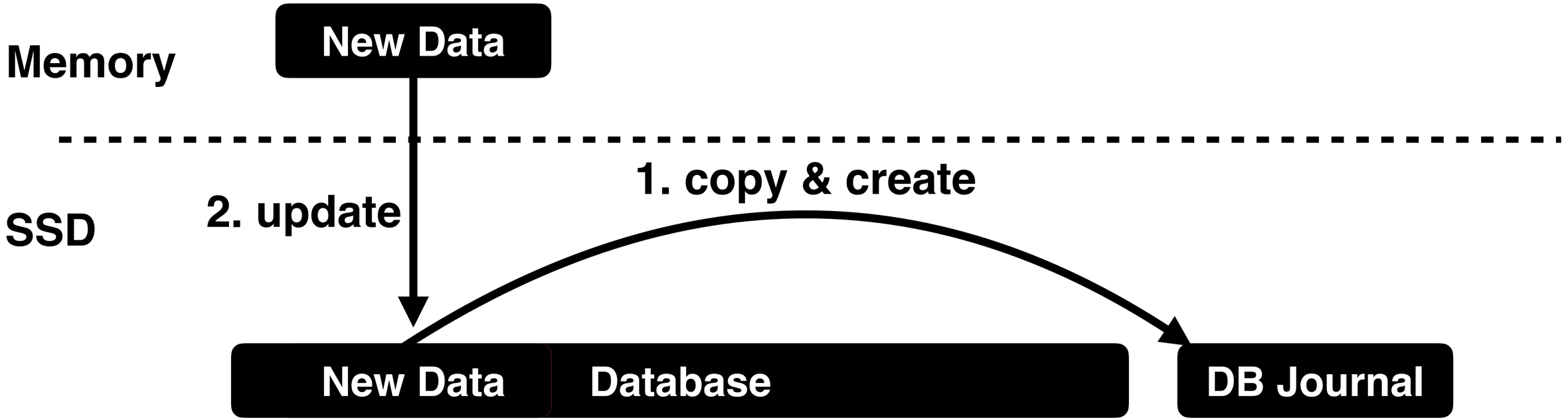
How SQLite-RollBack commit works?



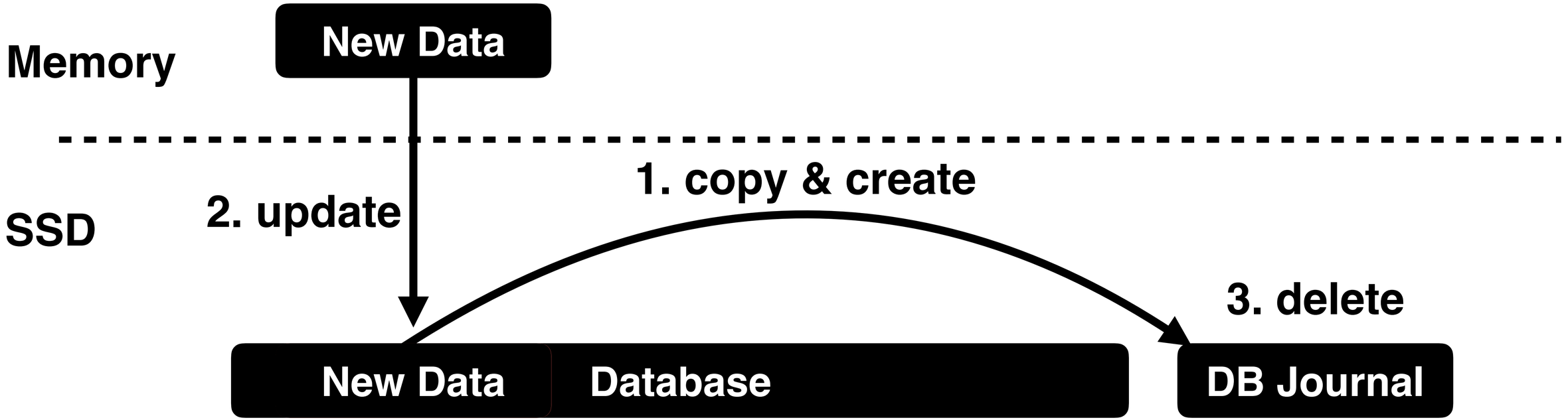
How SQLite-RollBack commit works?



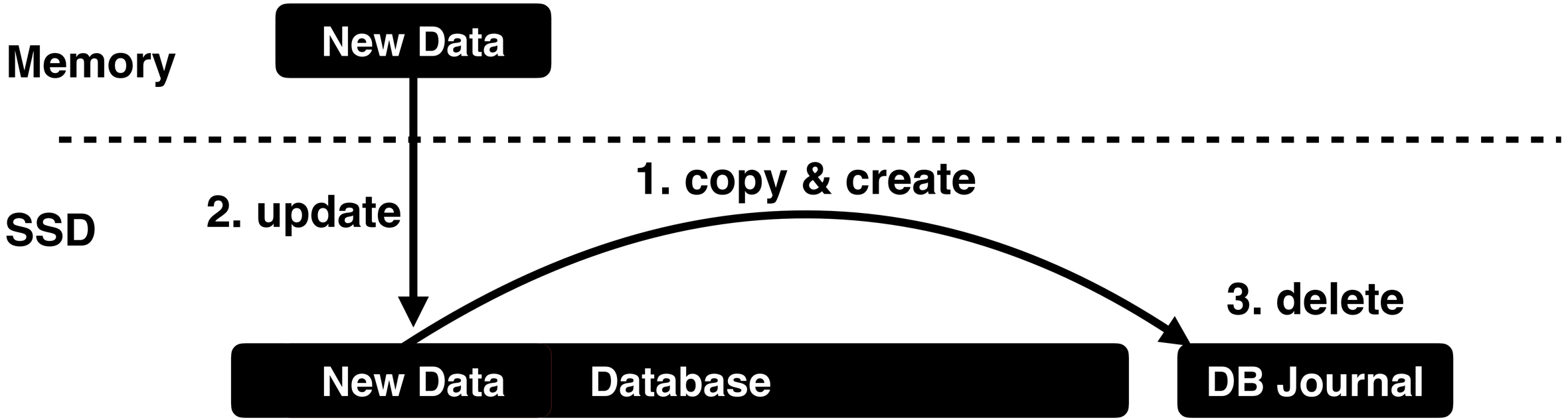
How SQLite-RollBack commit works?



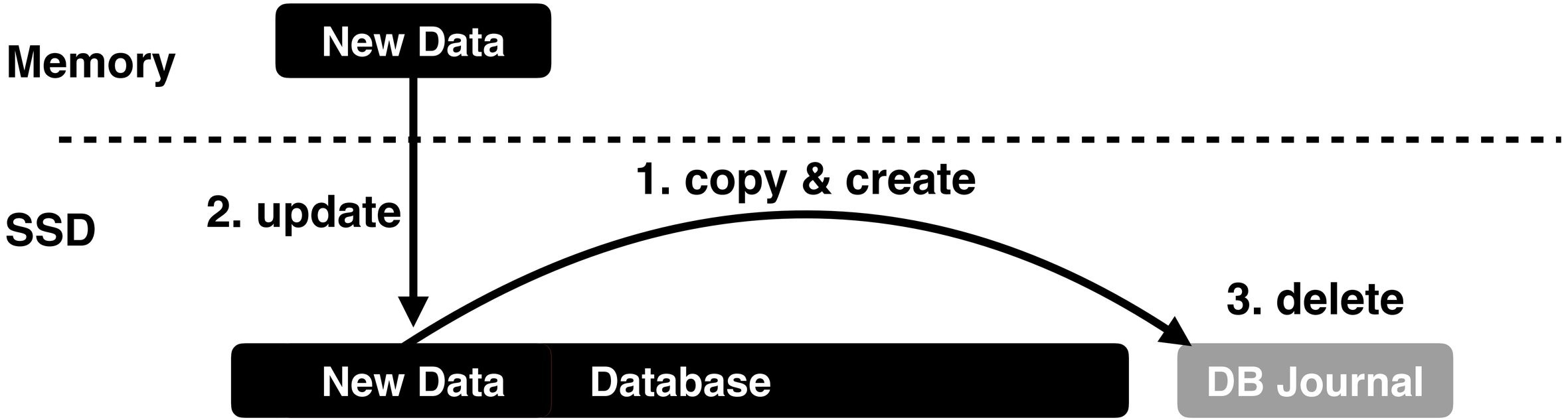
How SQLite-RollBack commit works?



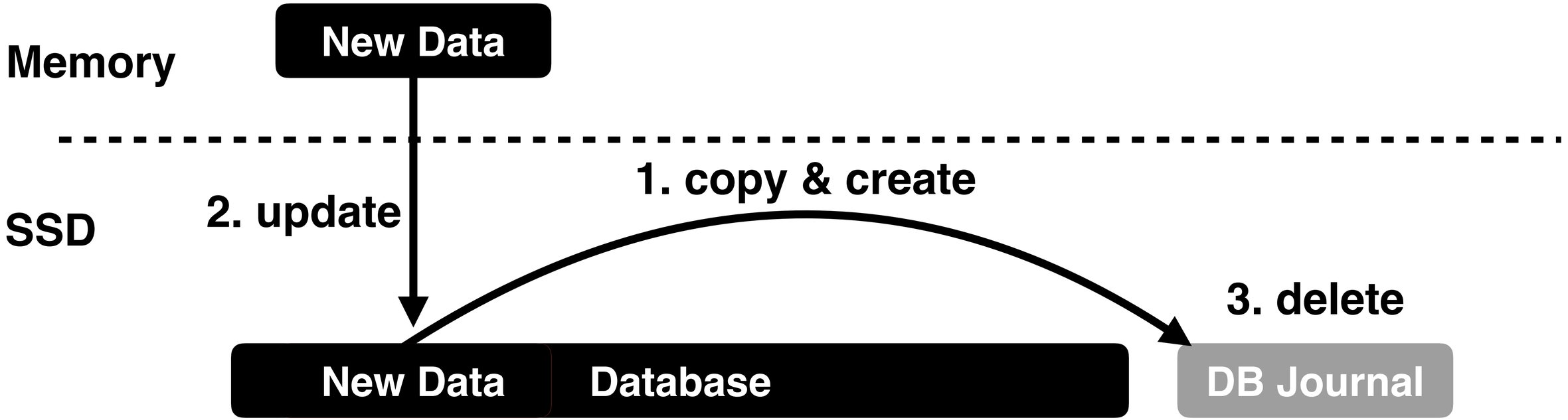
How SQLite-RollBack commit works?



How SQLite-RollBack commit works?

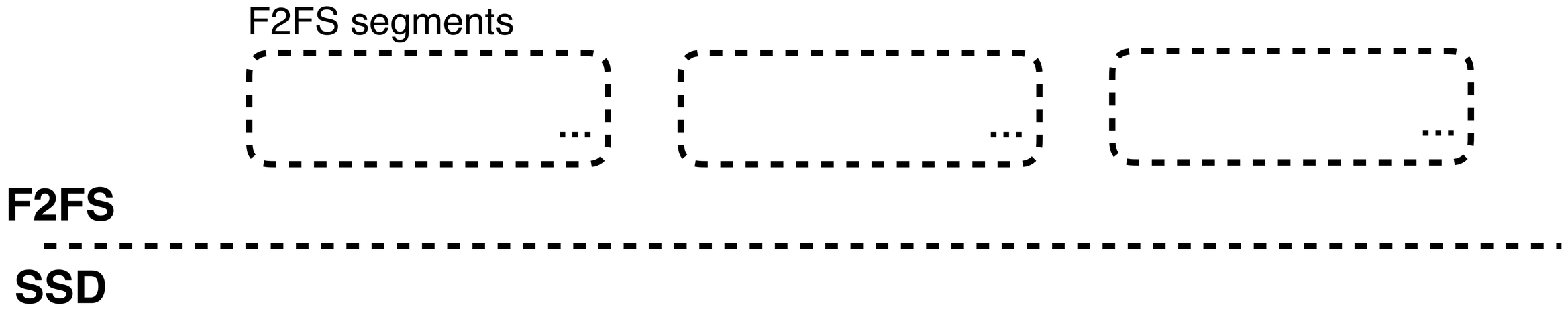


How SQLite-RollBack commit works?

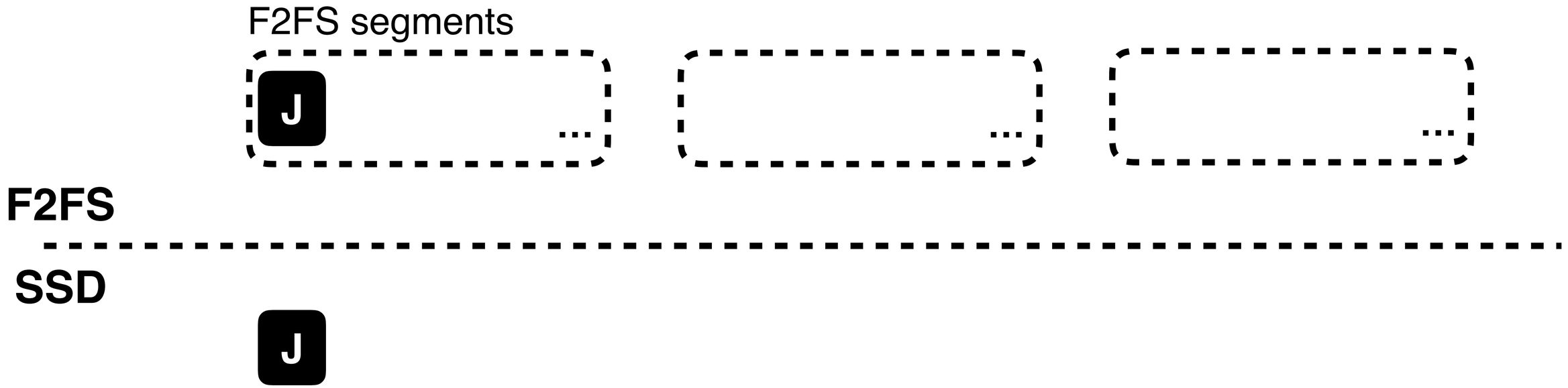


Small DB journal file is frequently created and deleted.

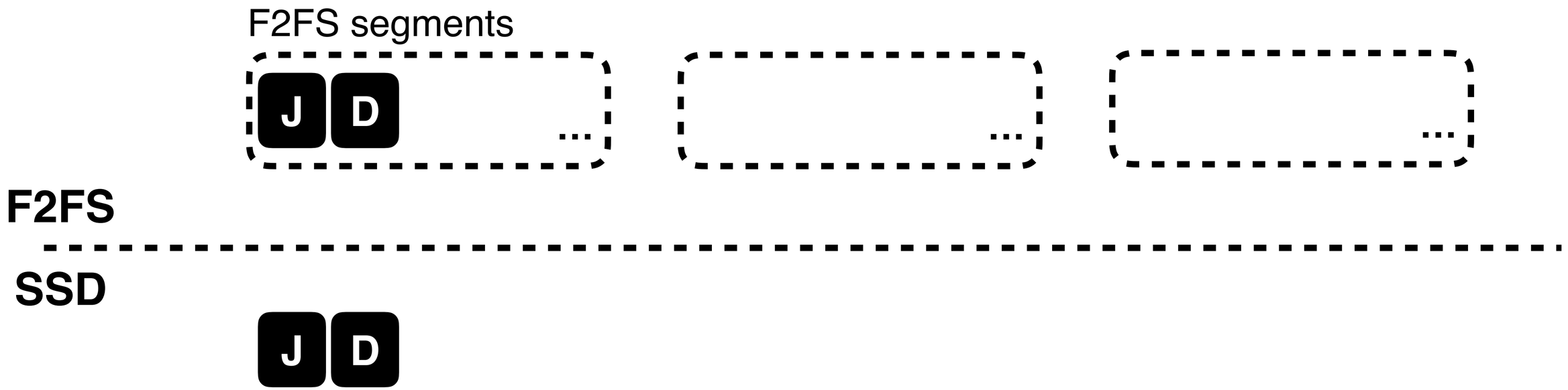
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



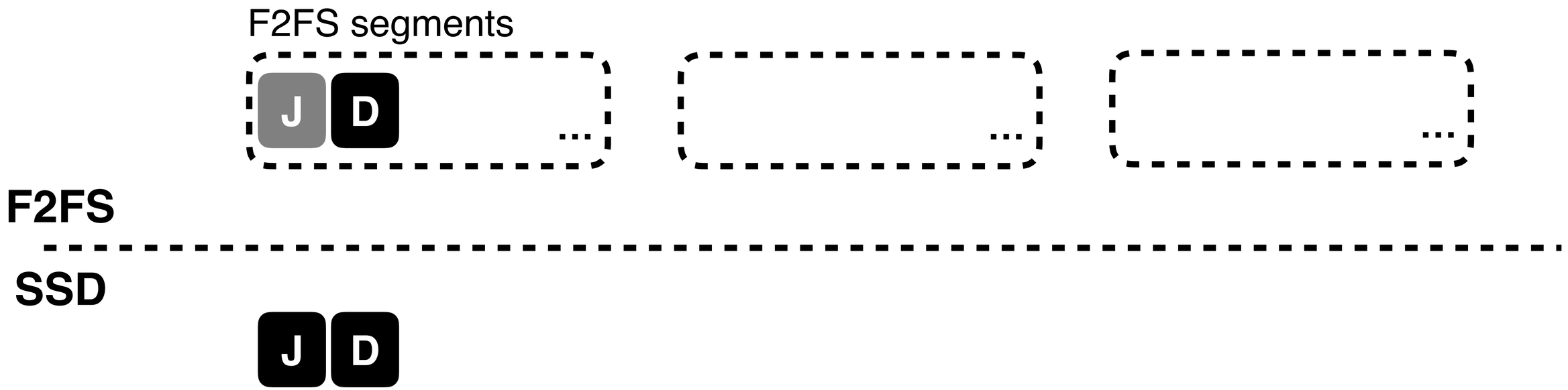
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



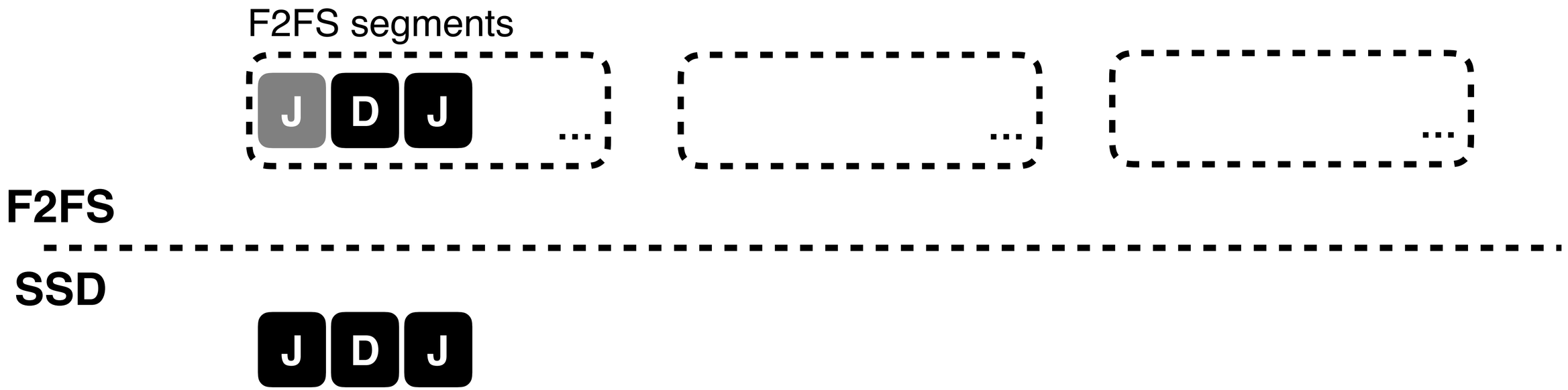
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



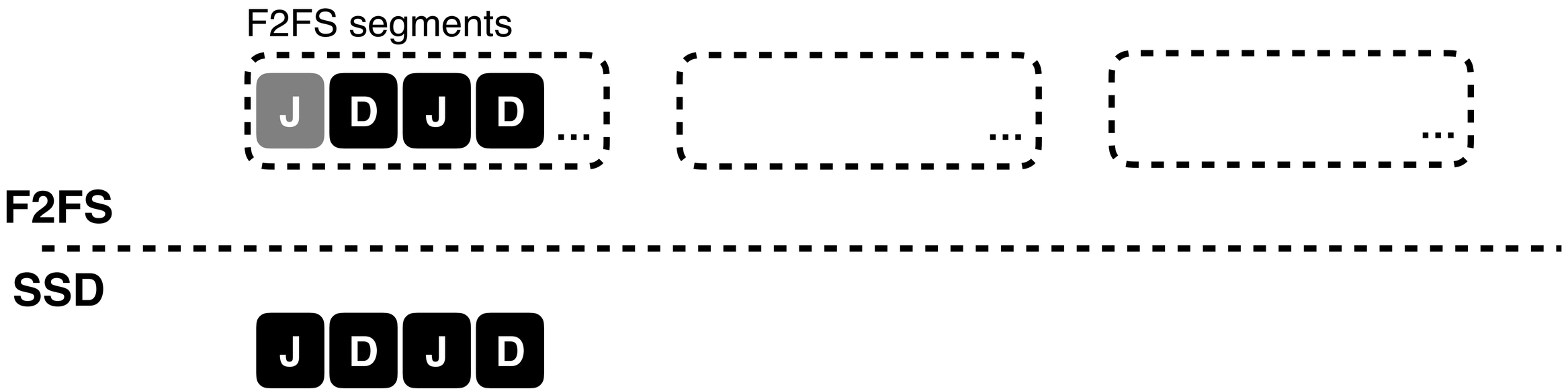
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



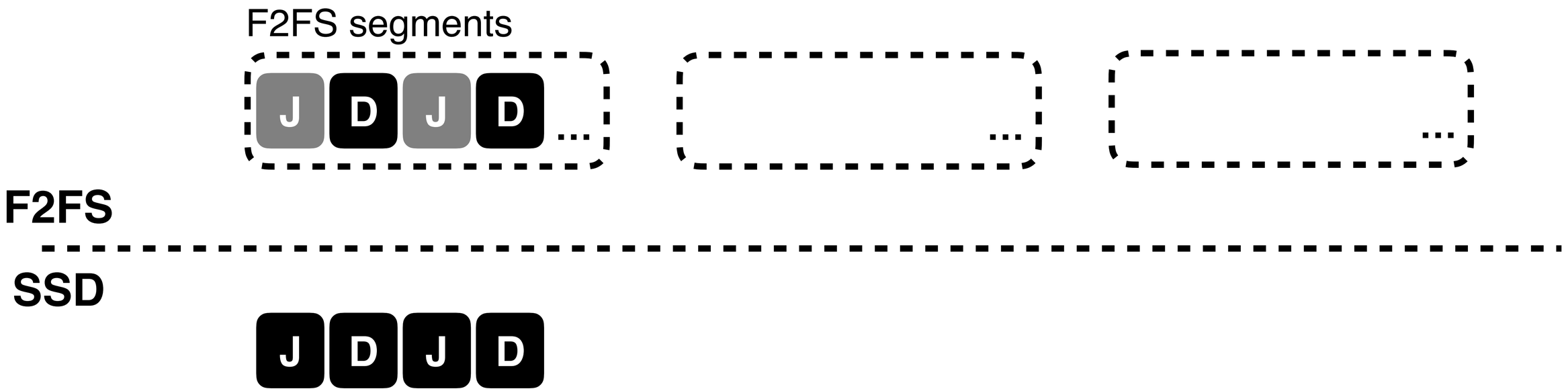
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



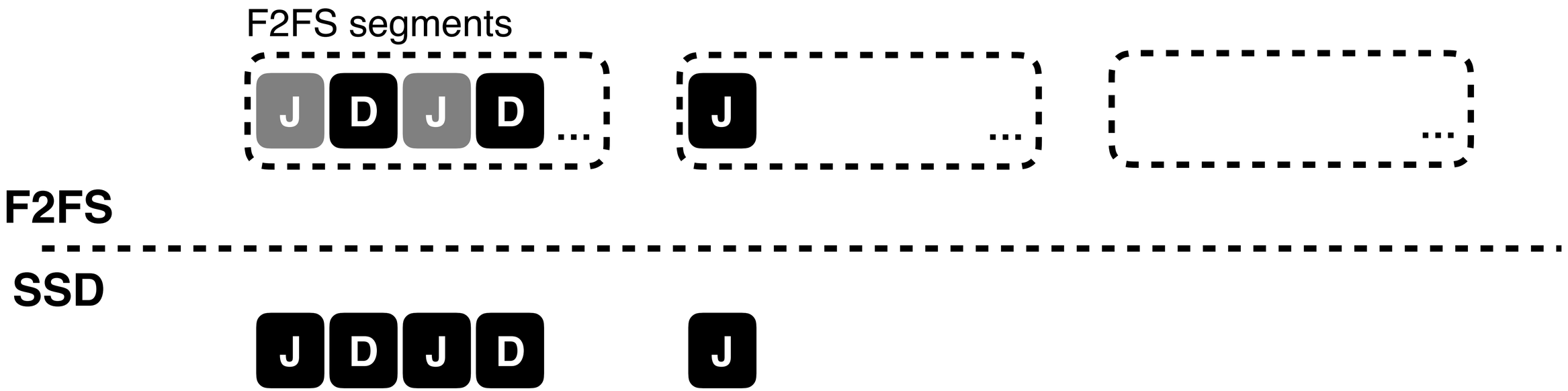
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



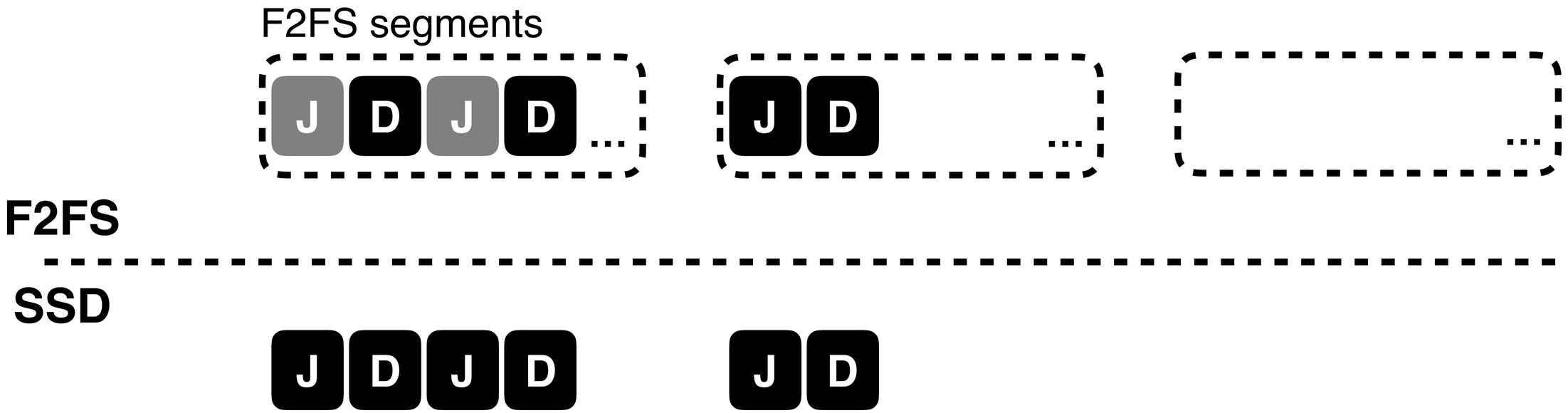
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



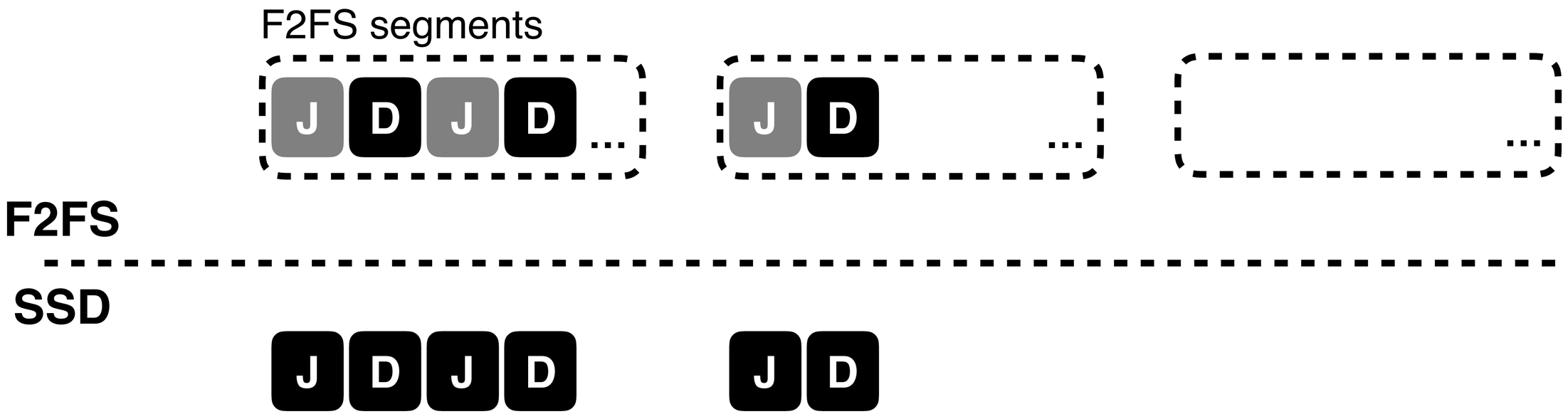
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



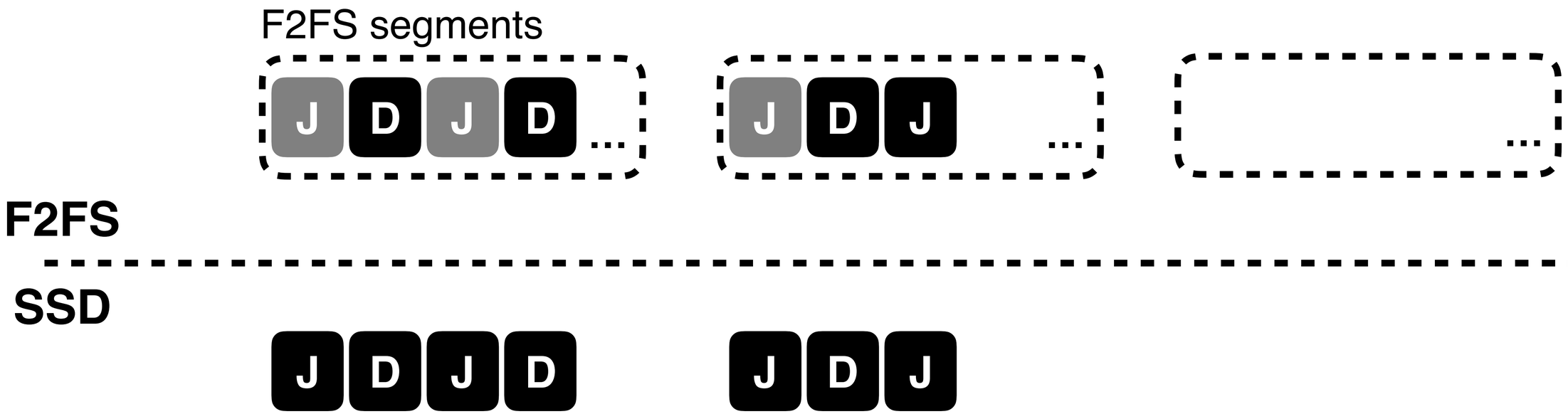
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



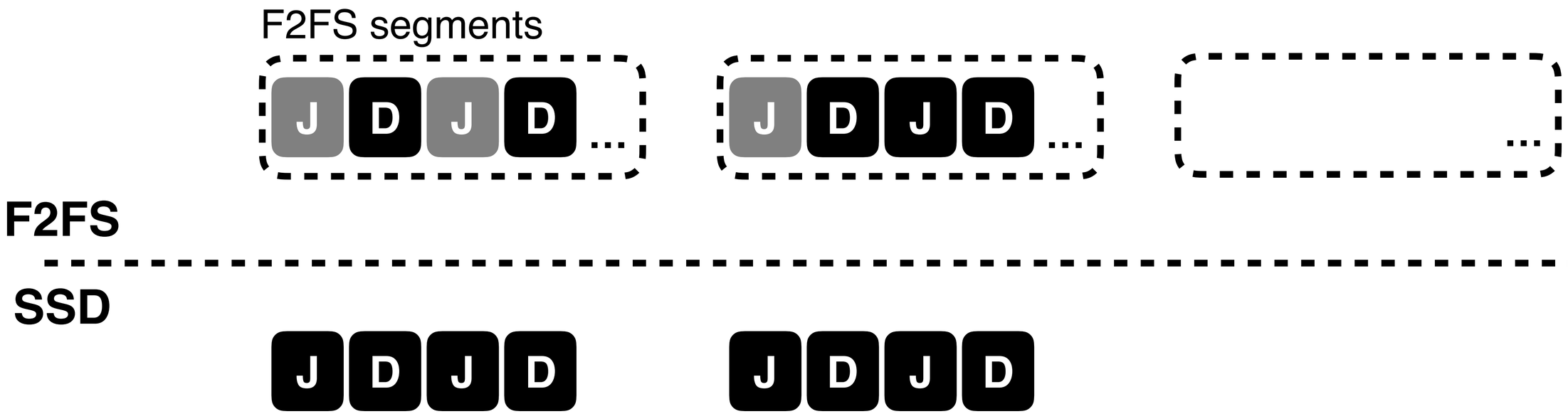
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



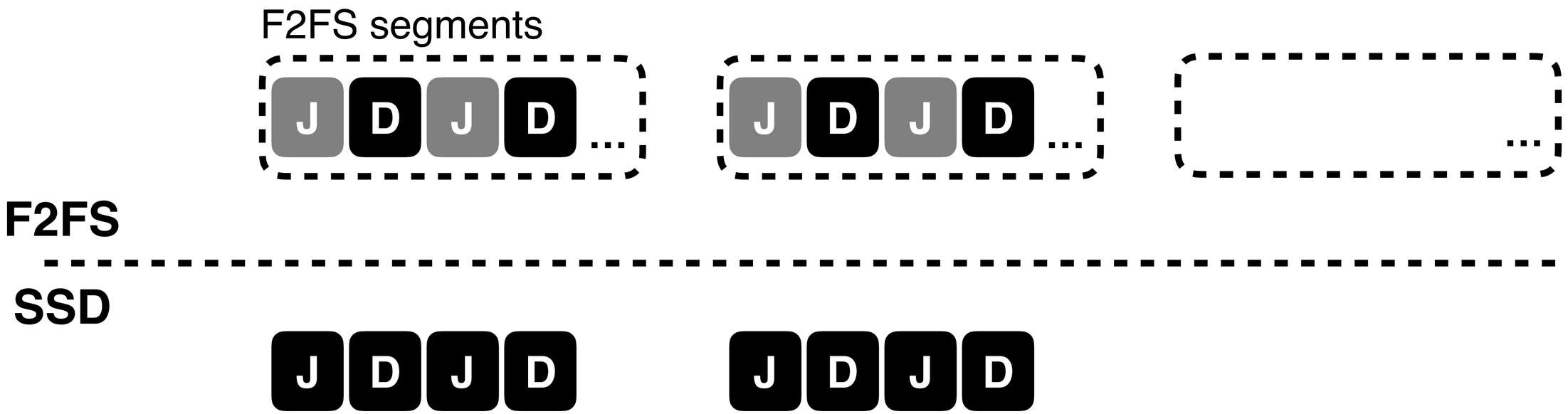
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



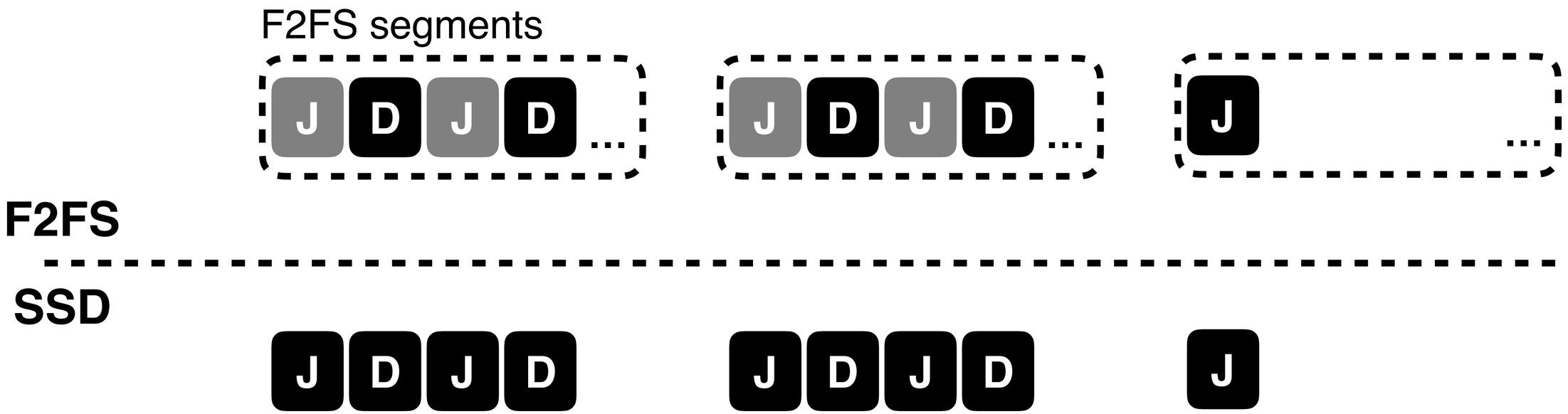
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



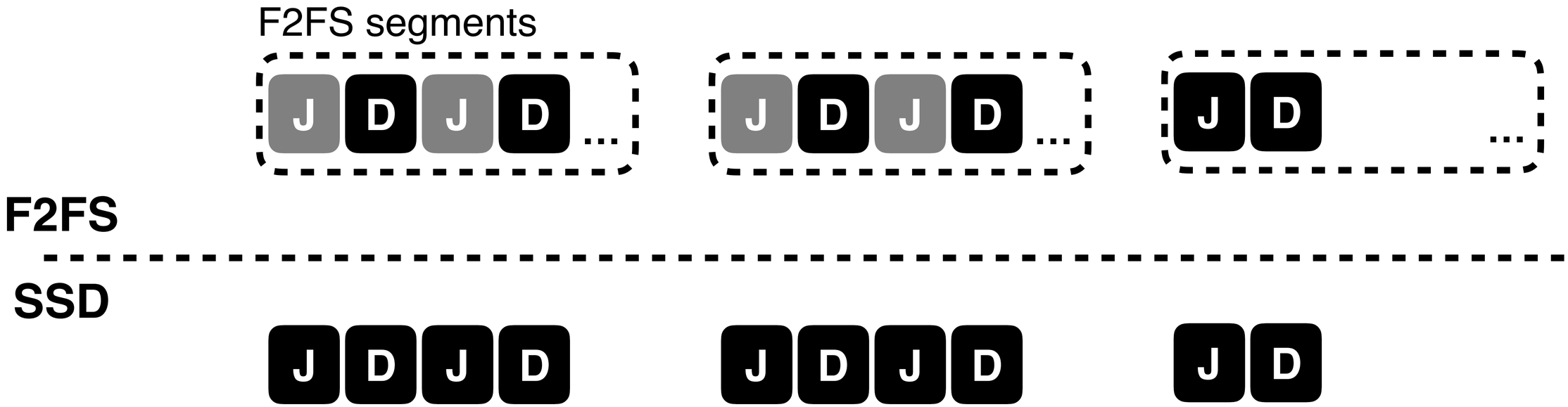
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



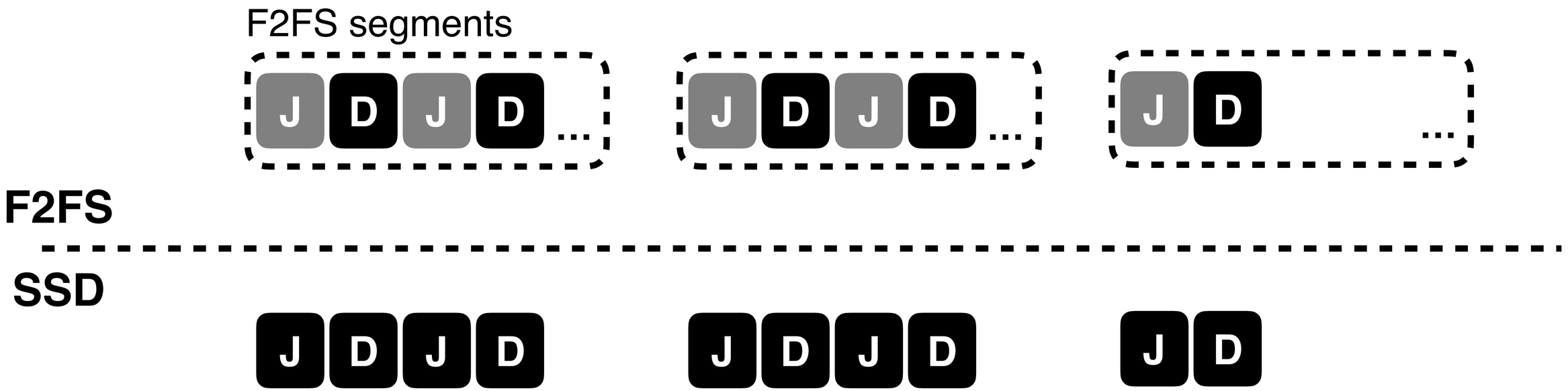
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



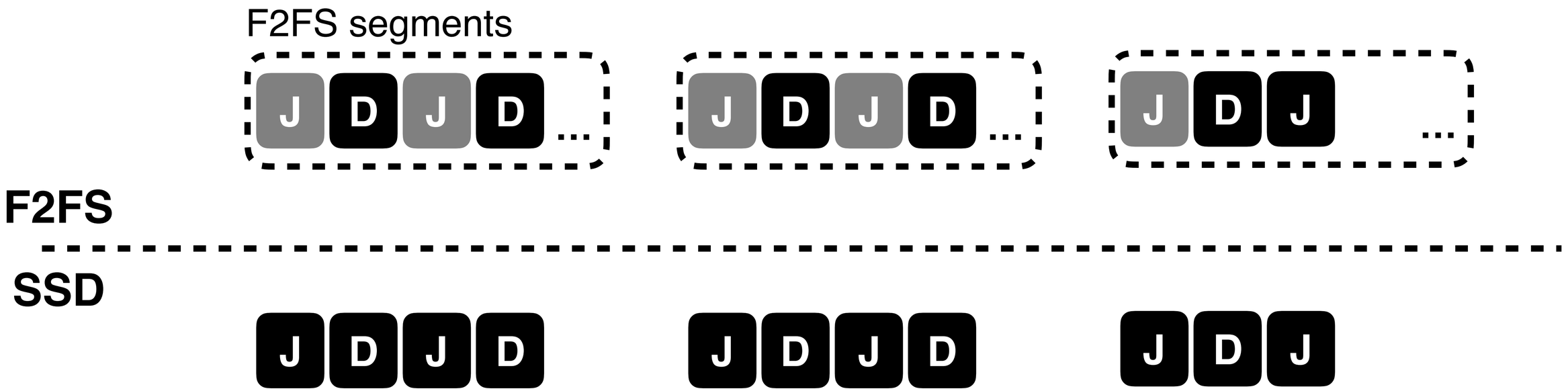
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



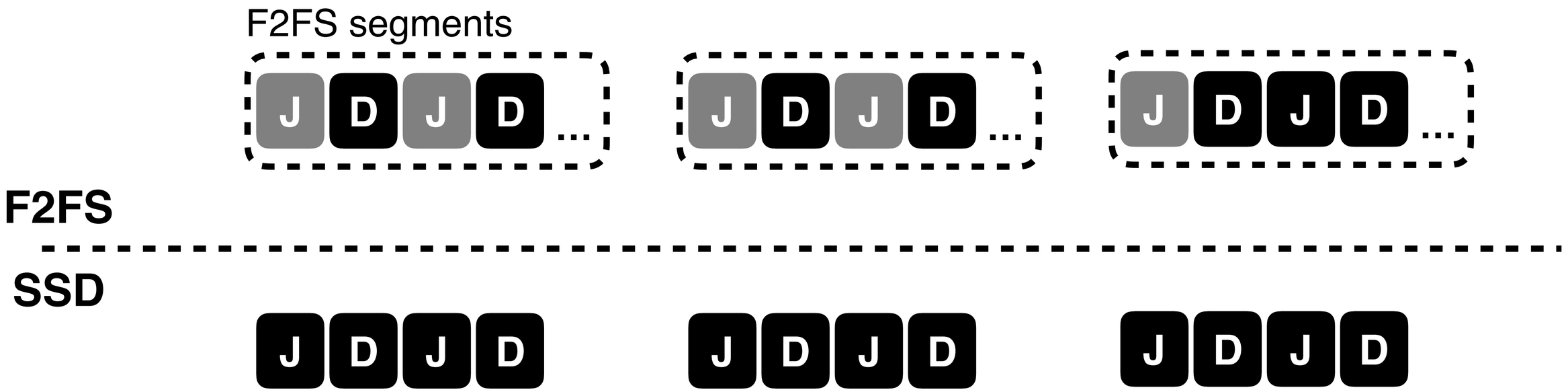
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



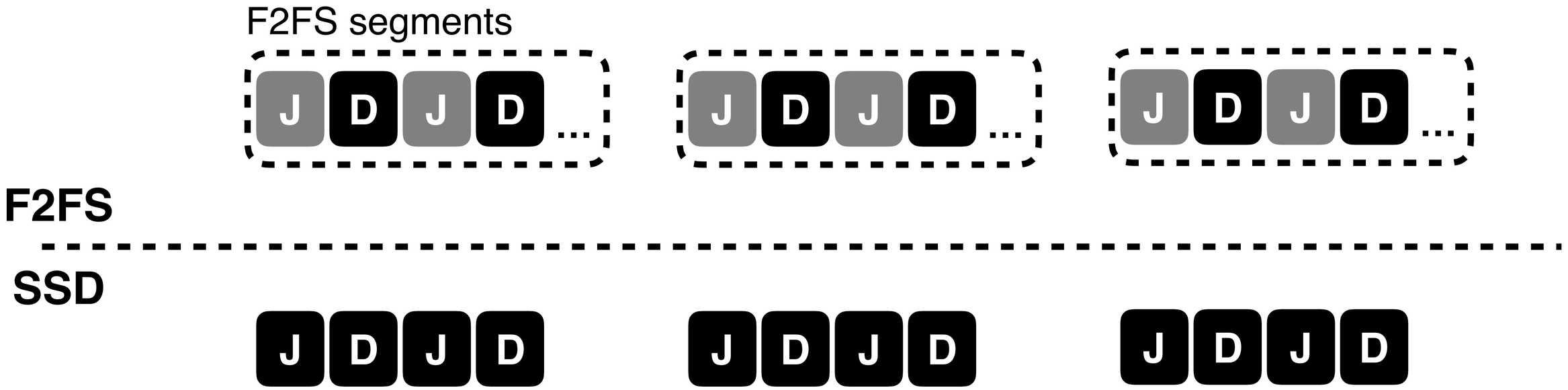
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



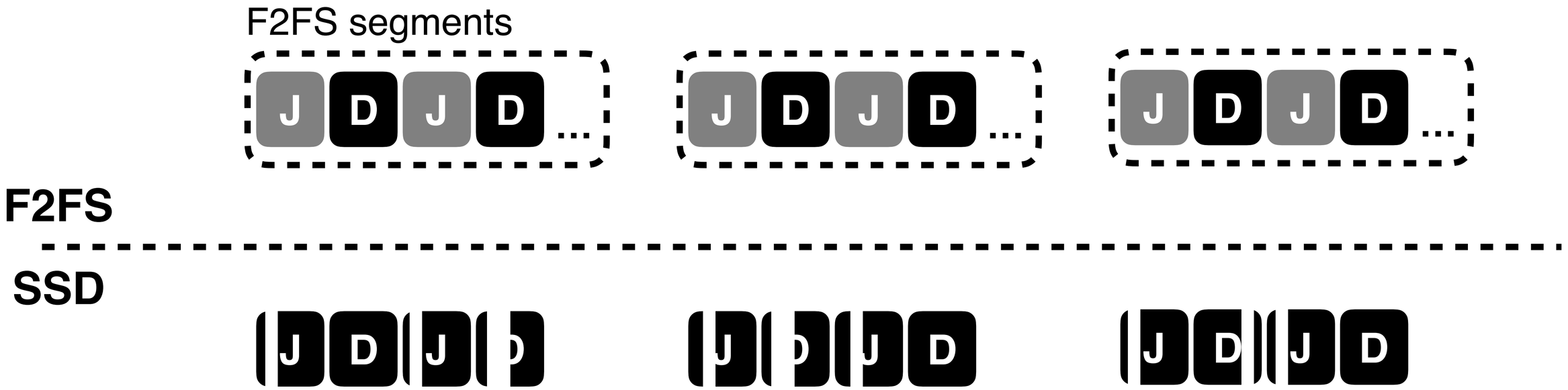
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



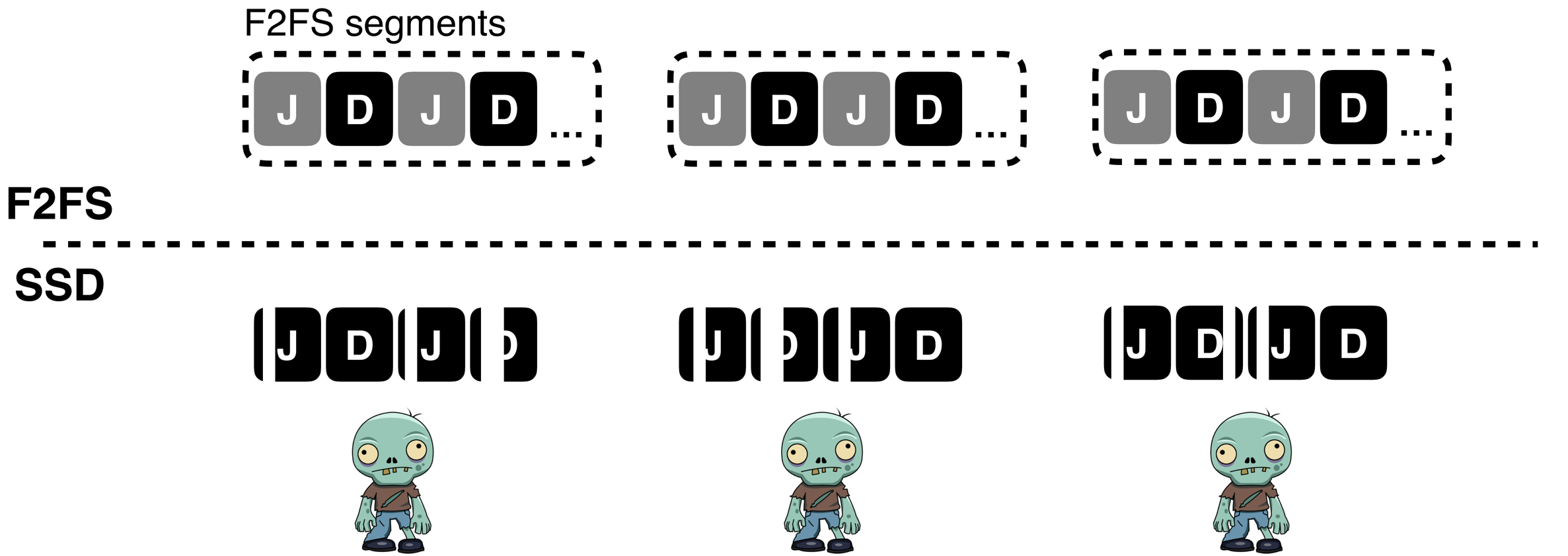
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



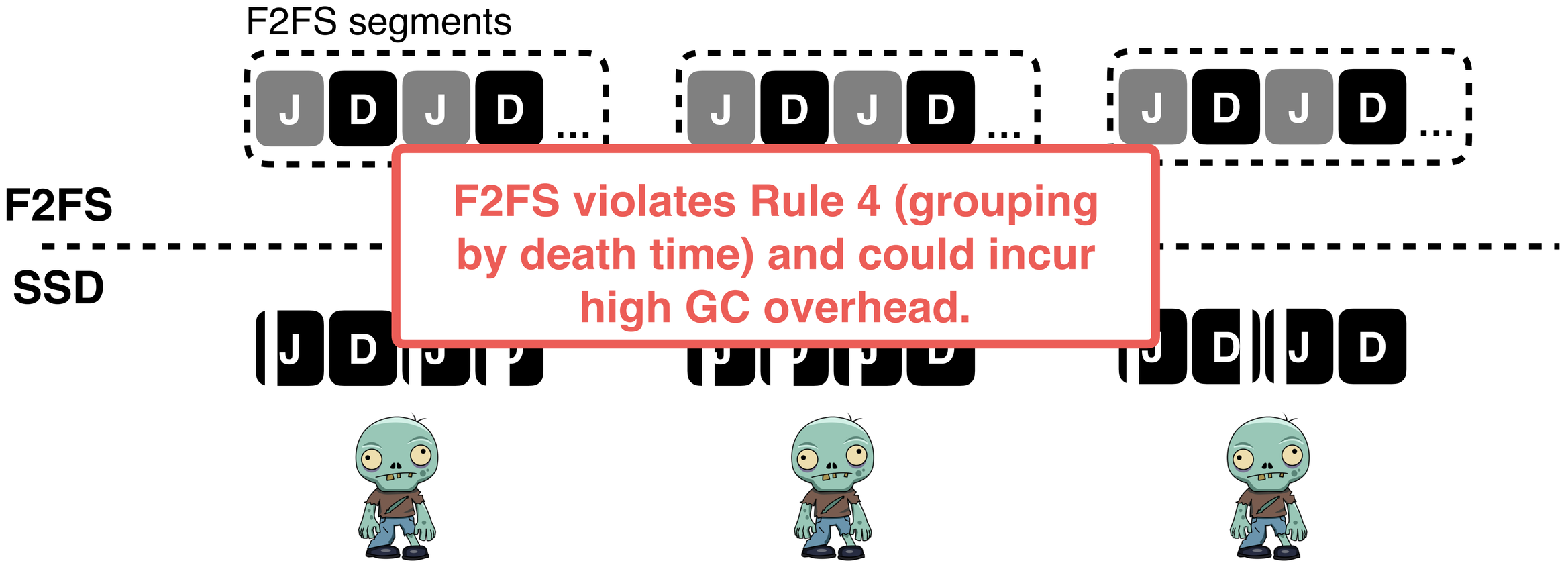
F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



F2FS delays discarding DB journal files; DB journal files are considered valid for a long time inside SSD



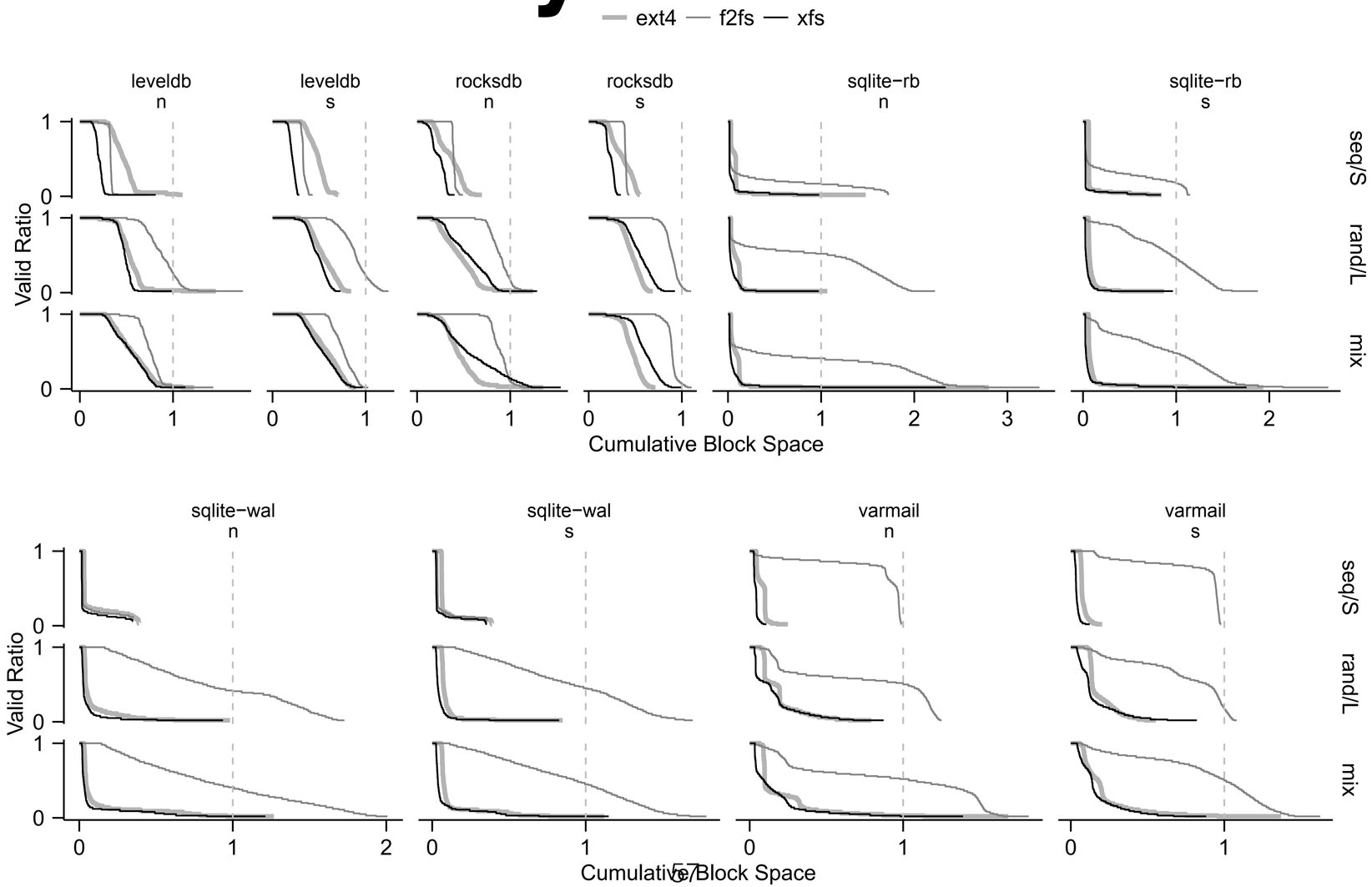
Observations

- Rule 1: Request Scale
  - Implementation of Linux limits performance
- Rule 3: Grouping by Death Time
  - F2FS incurs more GC overhead than ext4/XFS
  - Application log structuring does not reduce GC

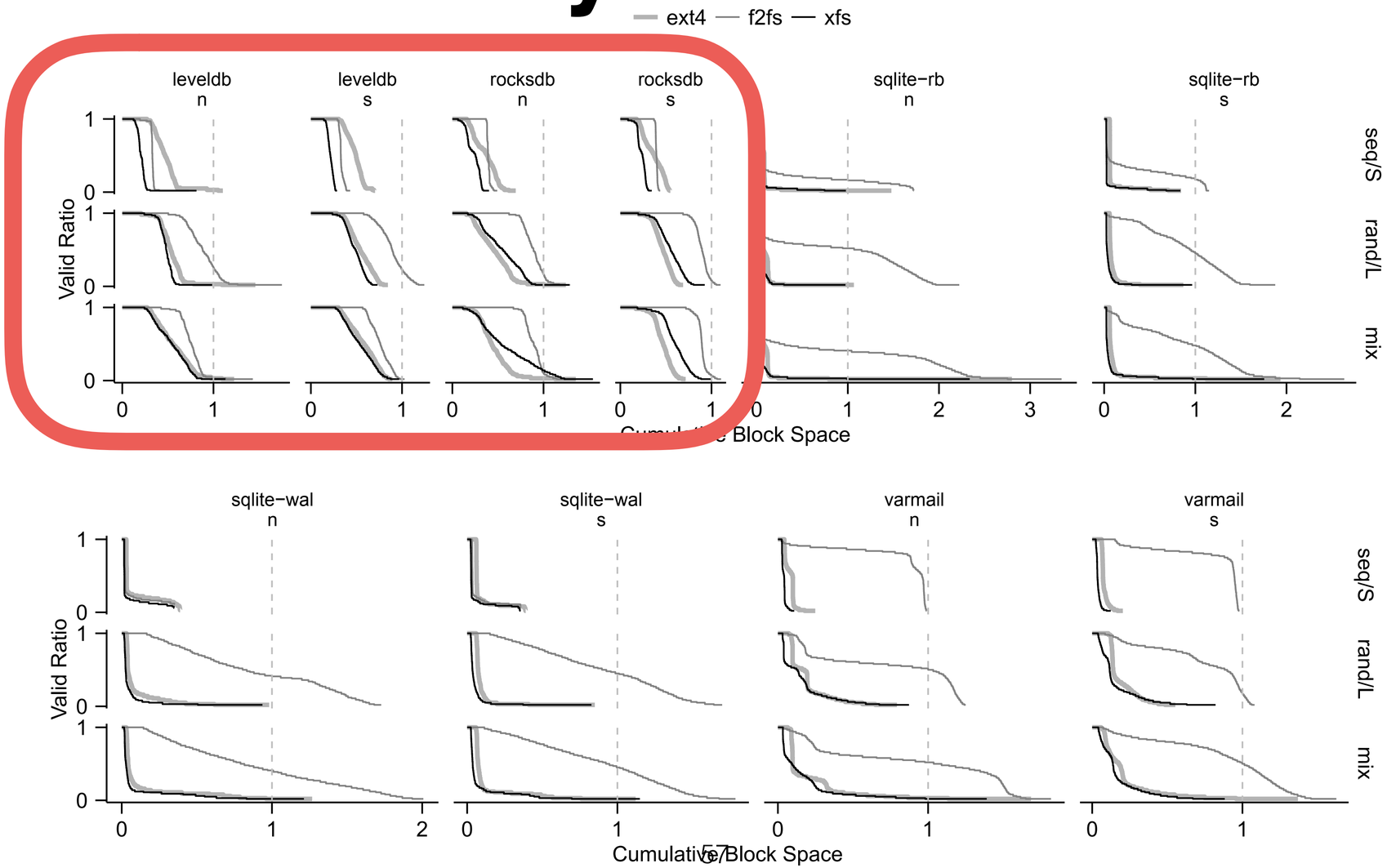
Observations

- Rule 1: Request Scale
  - Implementation of Linux limits performance
- Rule 3: Grouping by Death Time
  - F2FS incurs more GC overhead than ext4/XFS
  - Application log structuring does not reduce GC

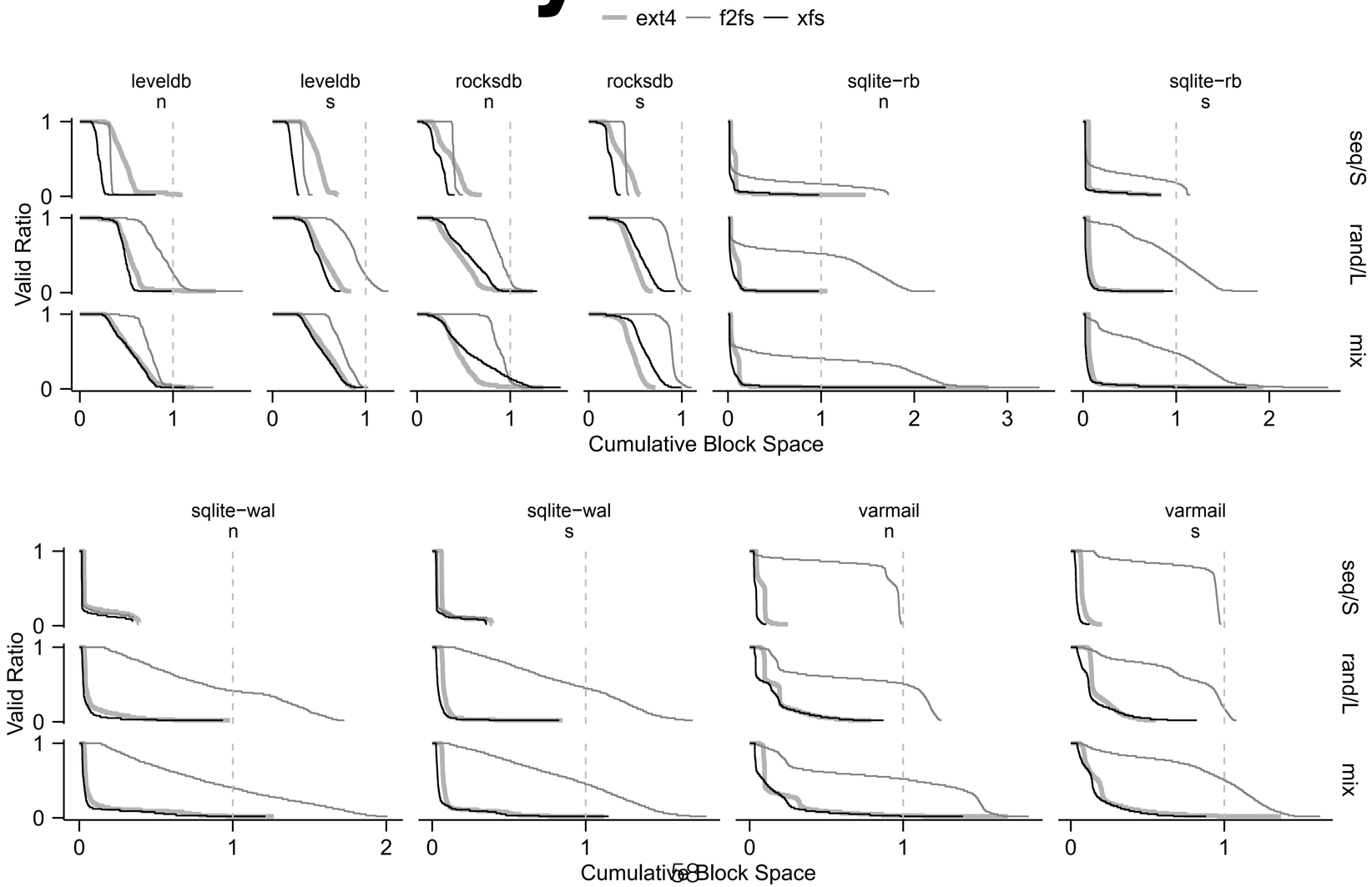
LevelDB & RocksDB often incur many zombie blocks



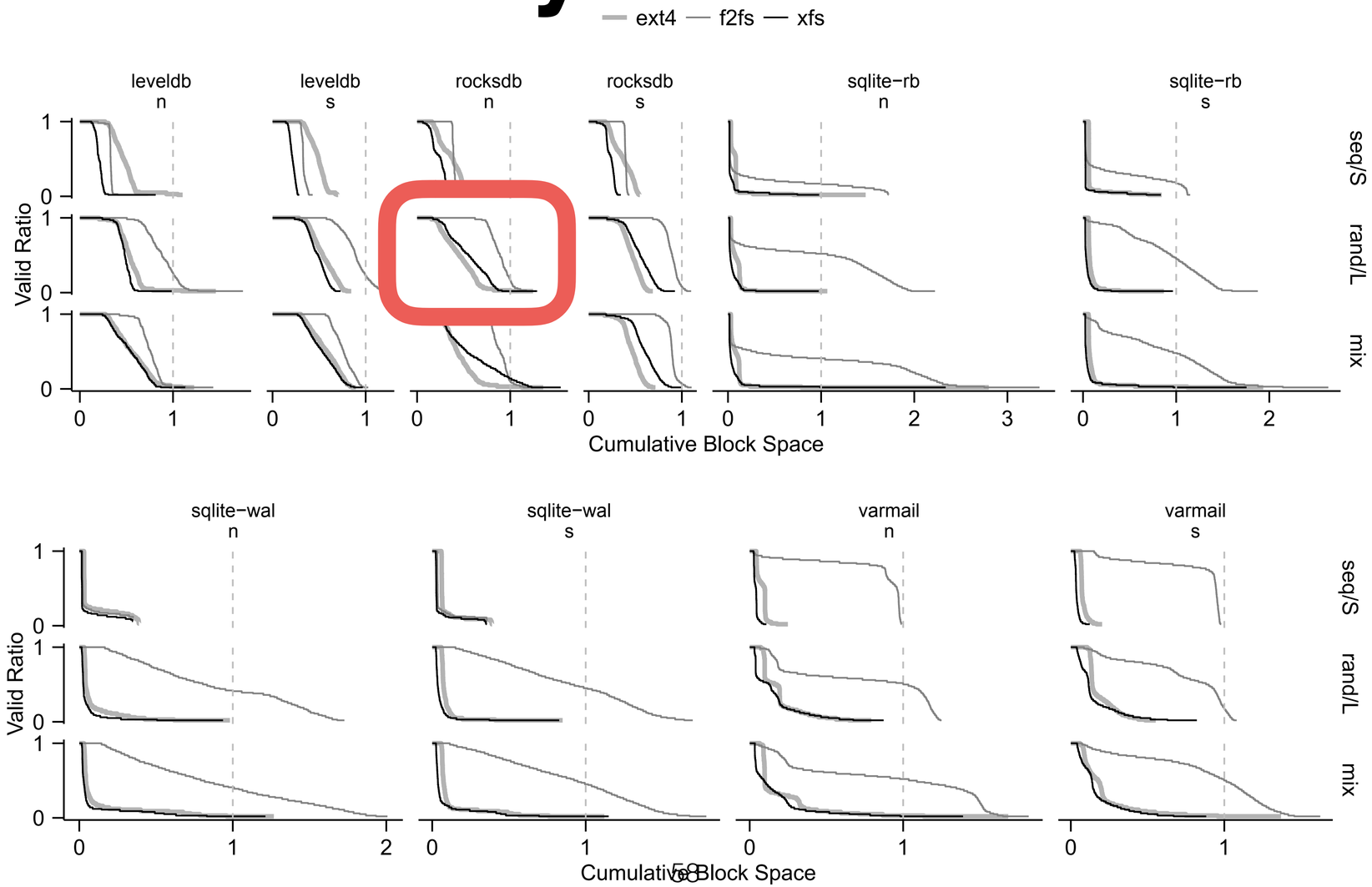
LevelDB & RocksDB often incur many zombie blocks



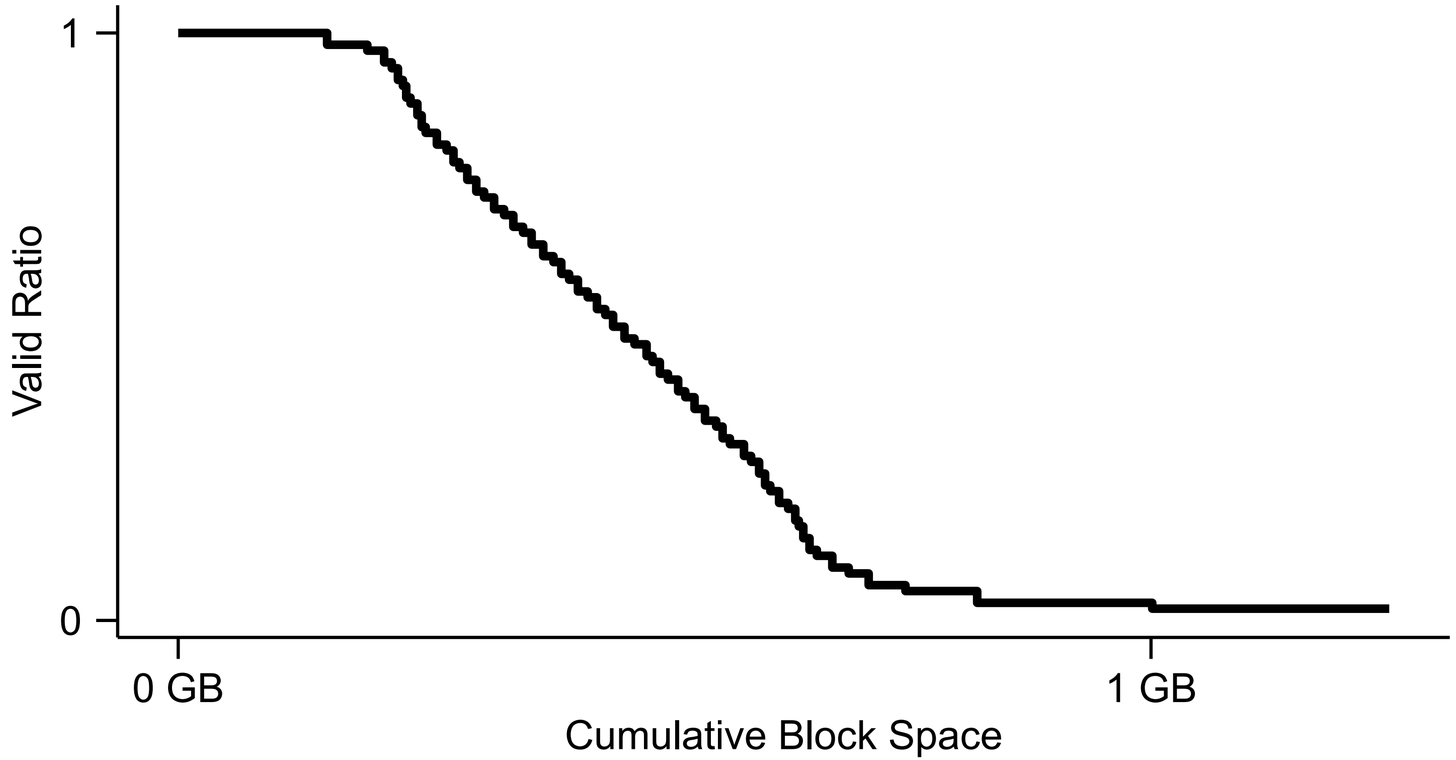
LevelDB & RocksDB often incur many zombie blocks



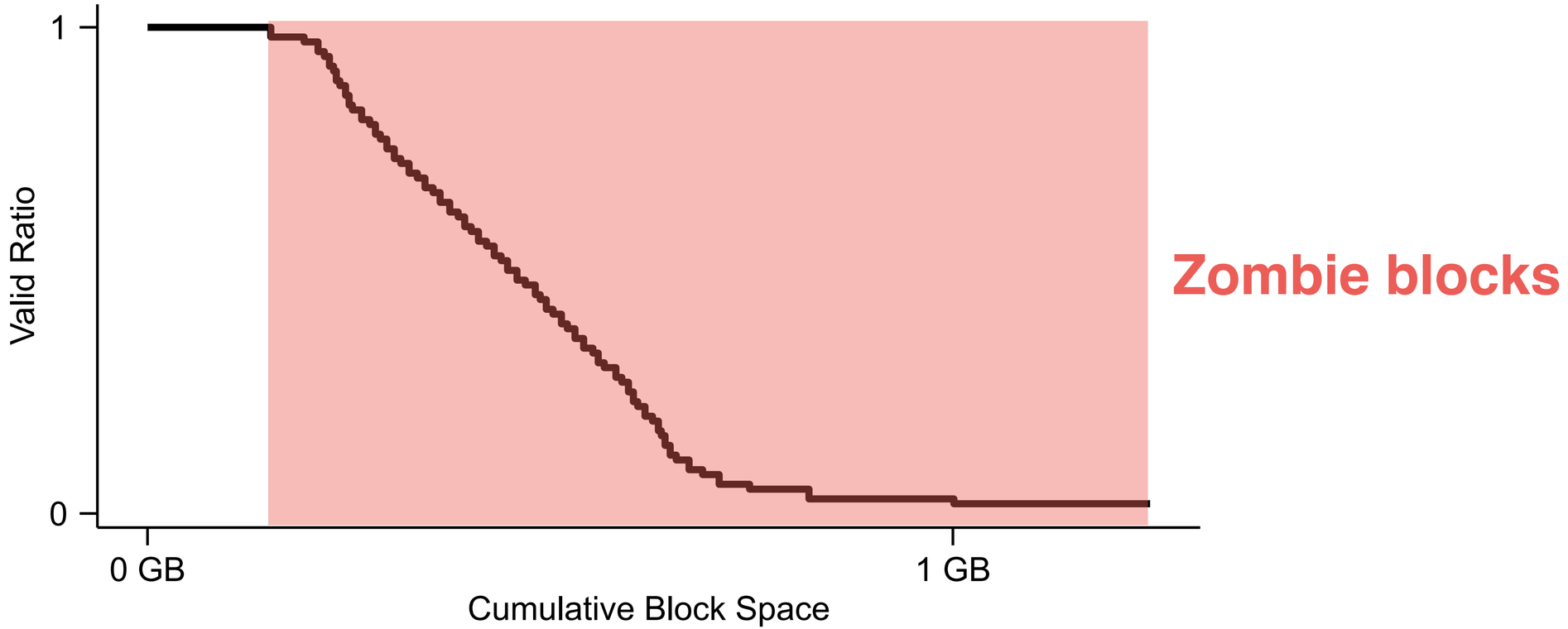
LevelDB & RocksDB often incur many zombie blocks



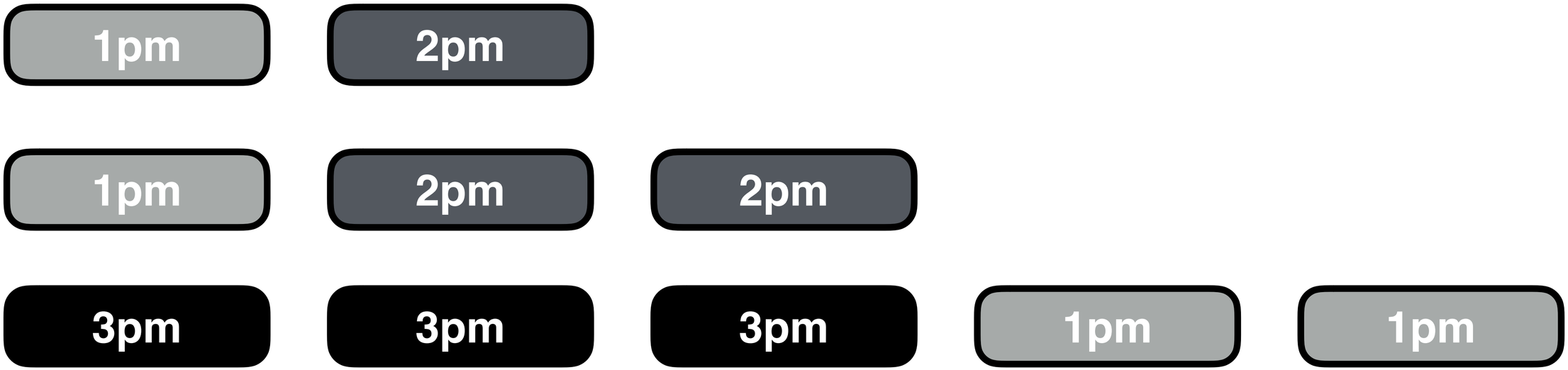
RocksDB Random Insertions on ext4



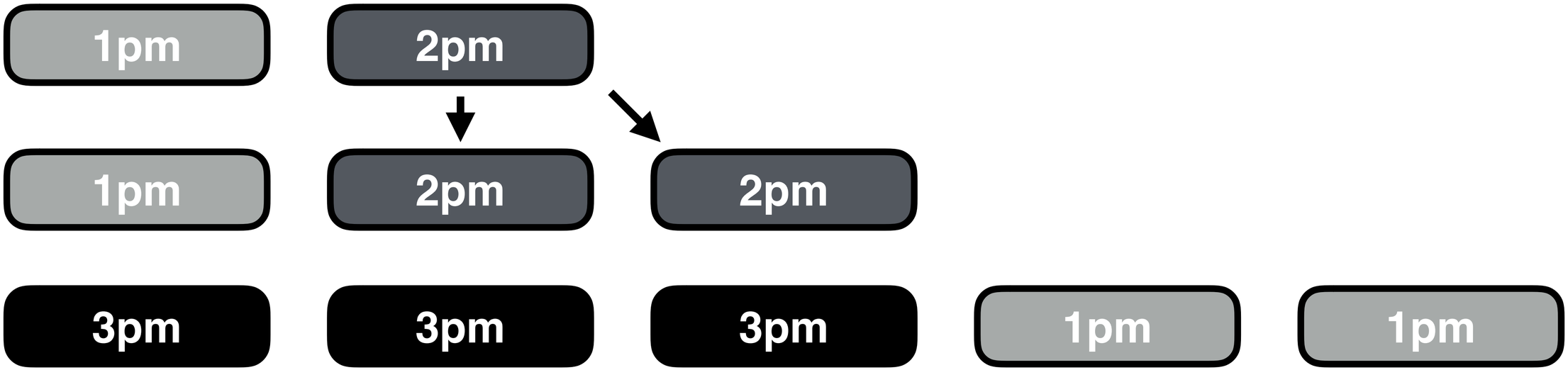
RocksDB Random Insertions on ext4



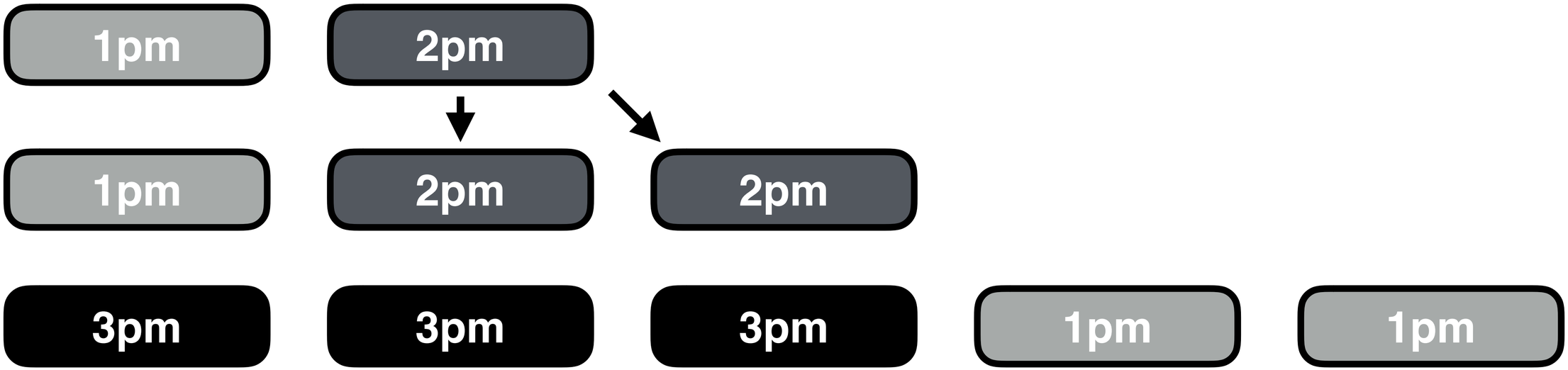
LevelDB & RocksDB files have different death times



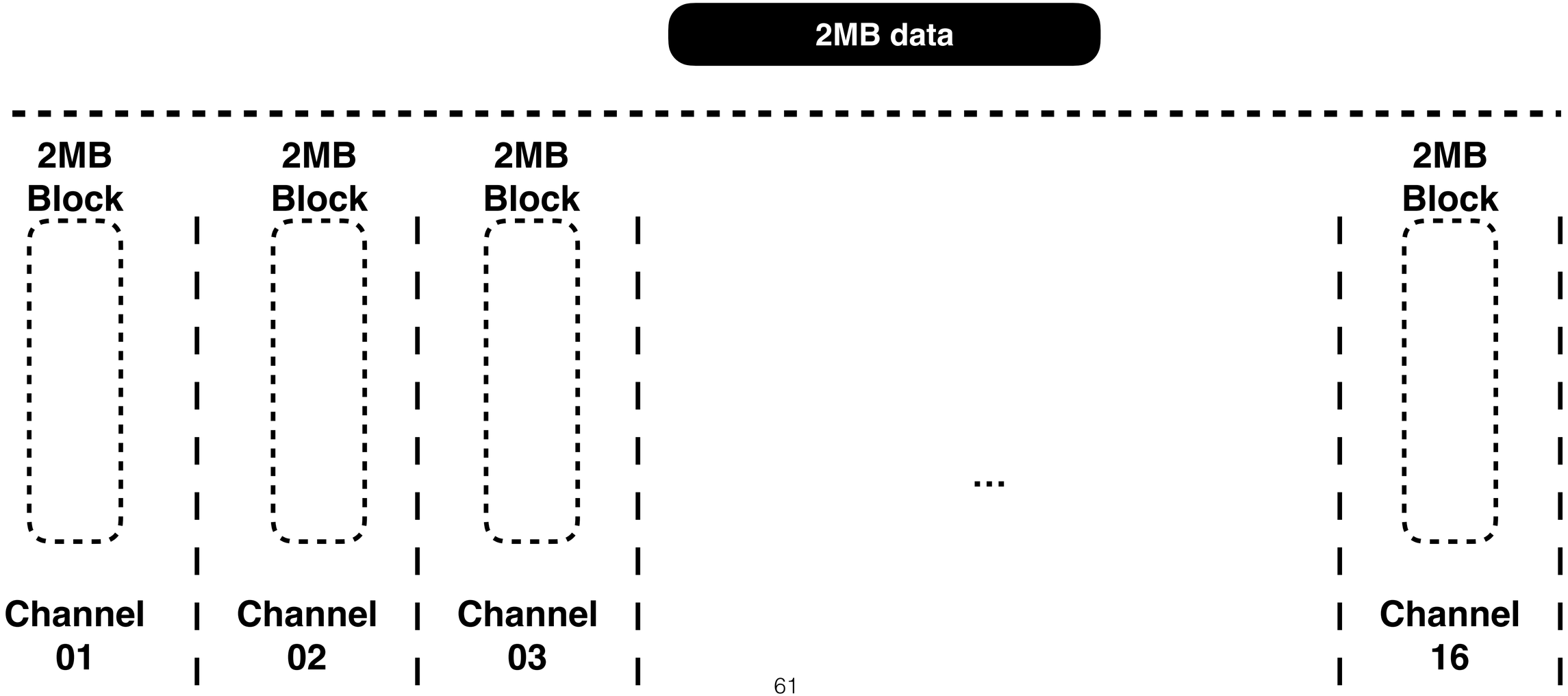
LevelDB & RocksDB files have different death times



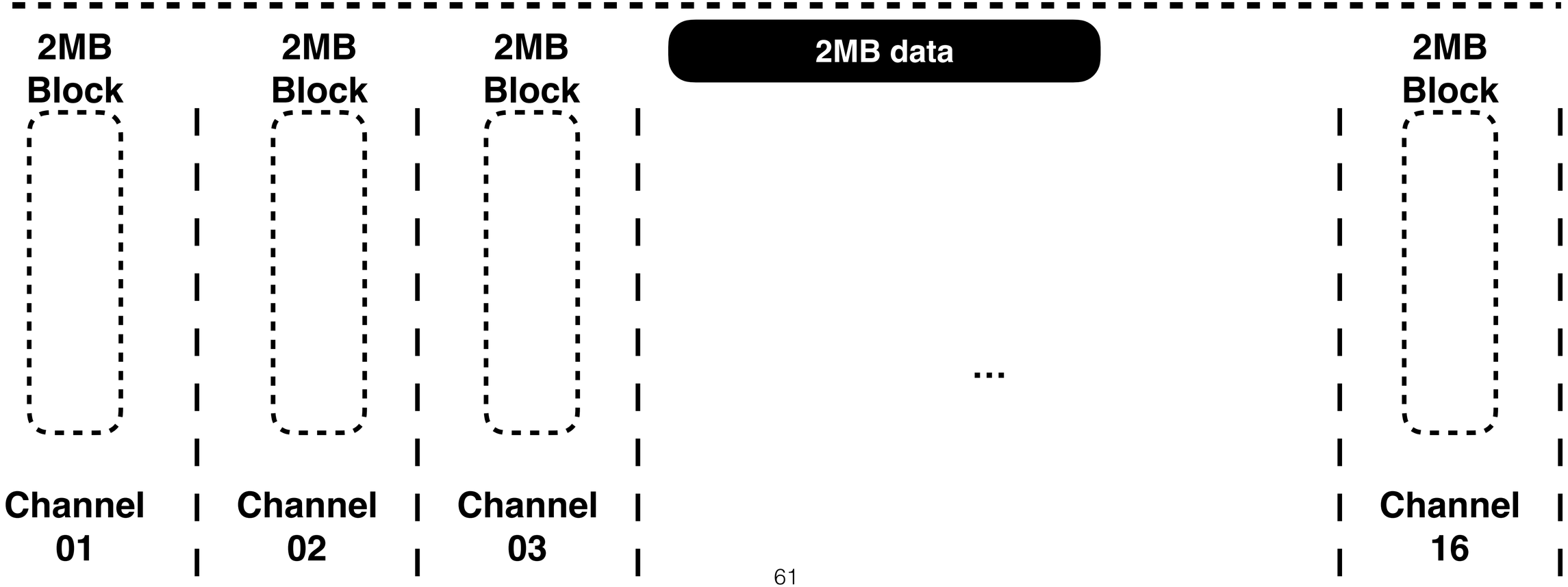
LevelDB & RocksDB files have different death times



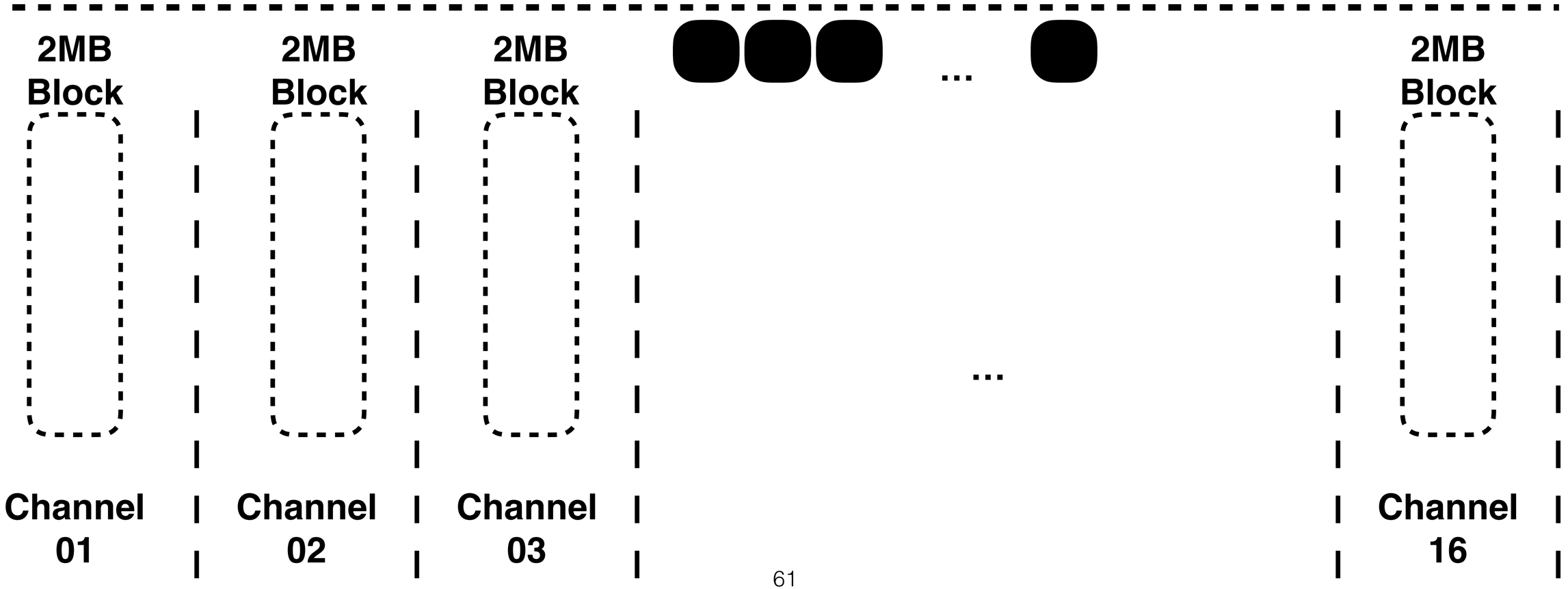
Data of different files is mixed to the same blocks



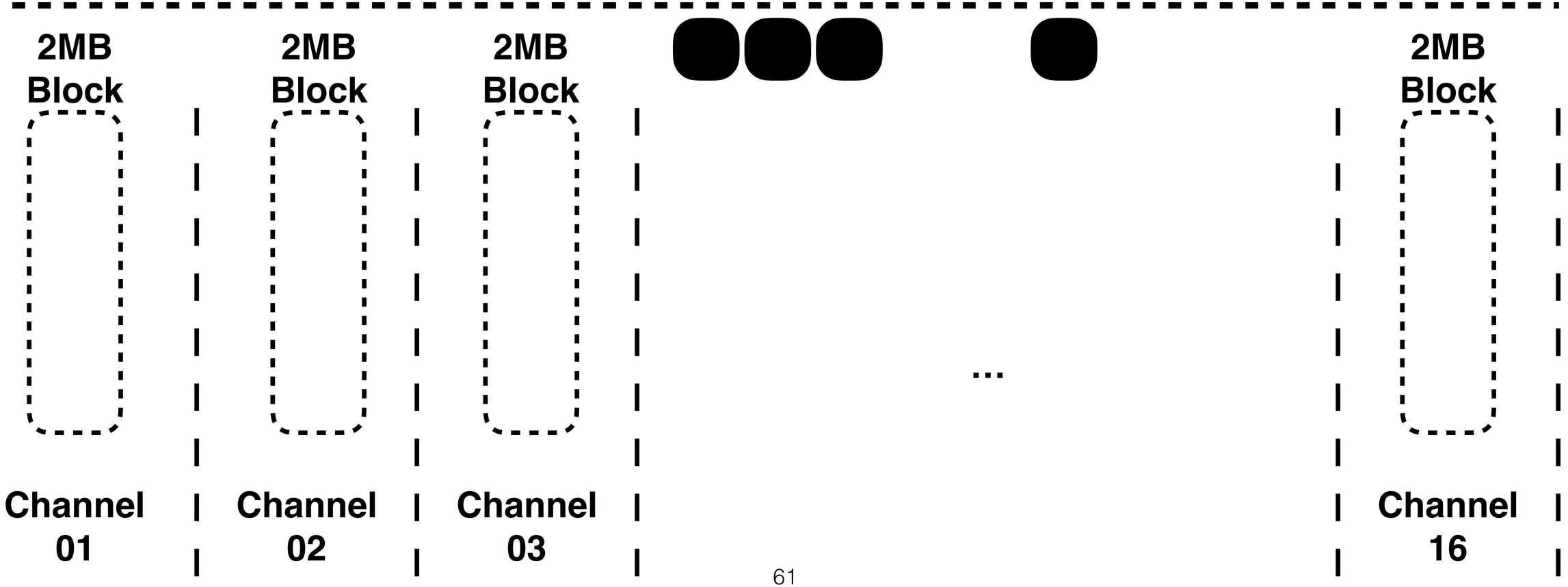
Data of different files is mixed to the same blocks



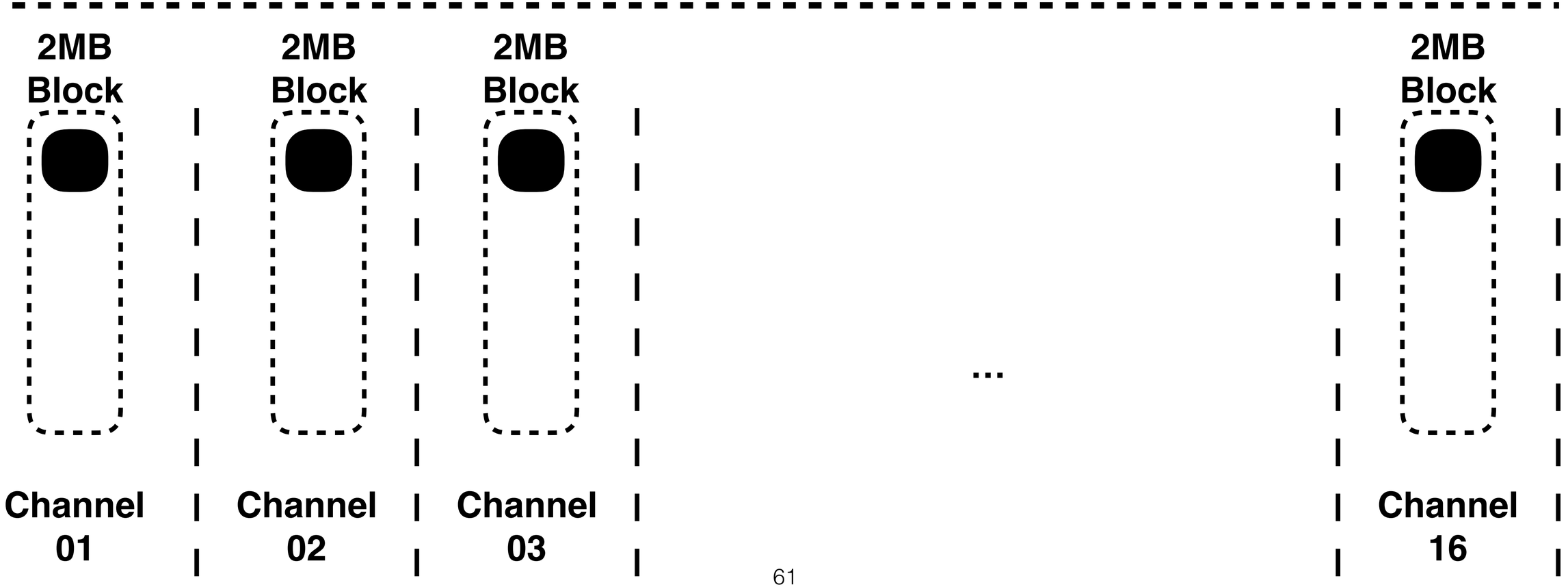
Data of different files is mixed to the same blocks



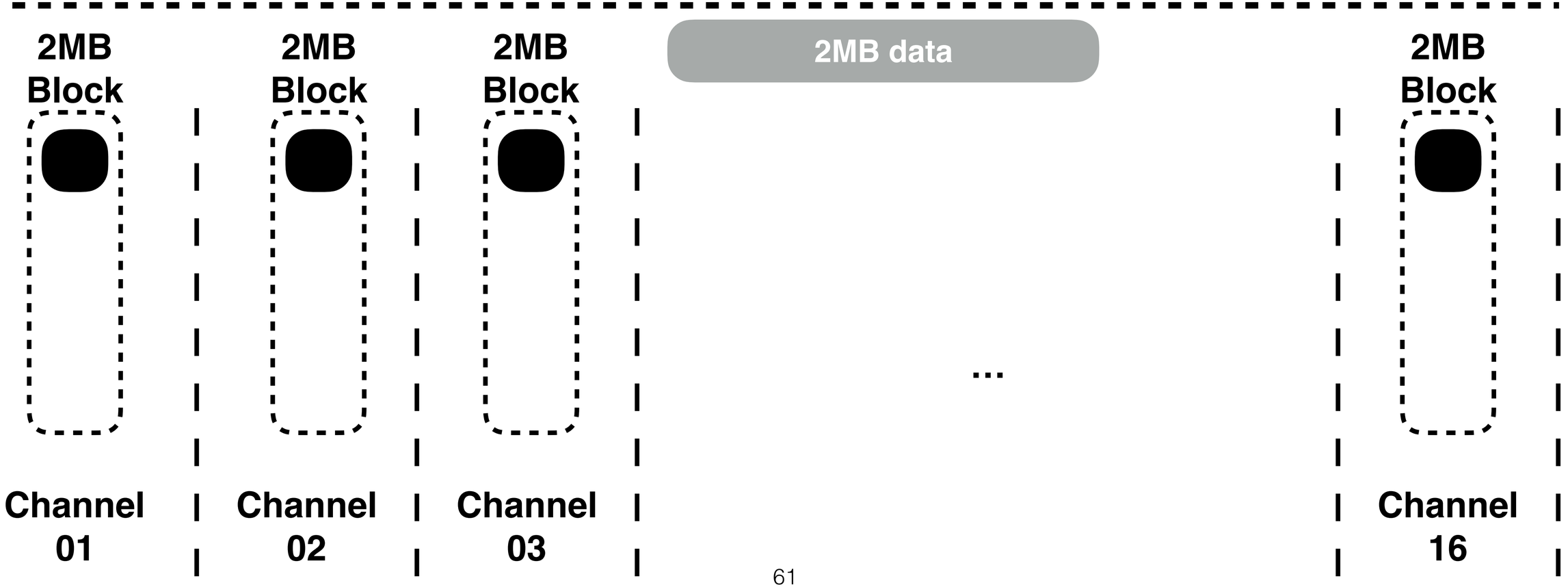
Data of different files is mixed to the same blocks



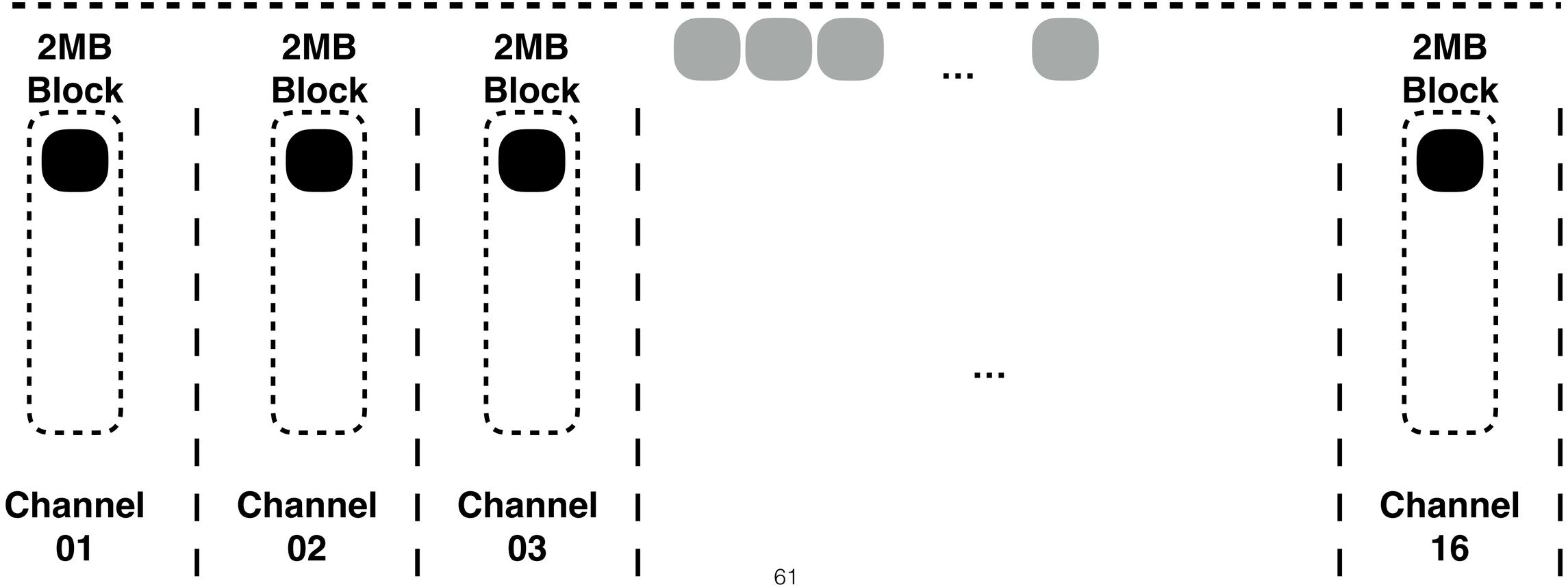
Data of different files is mixed to the same blocks



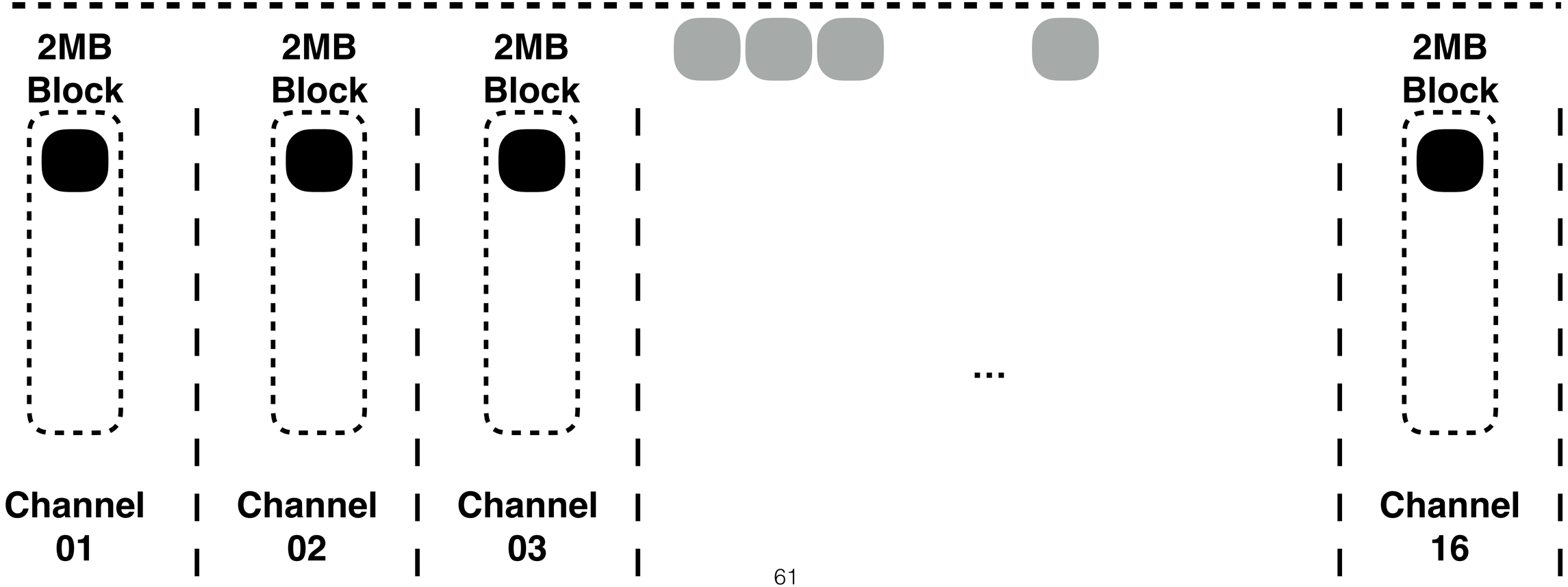
Data of different files is mixed to the same blocks



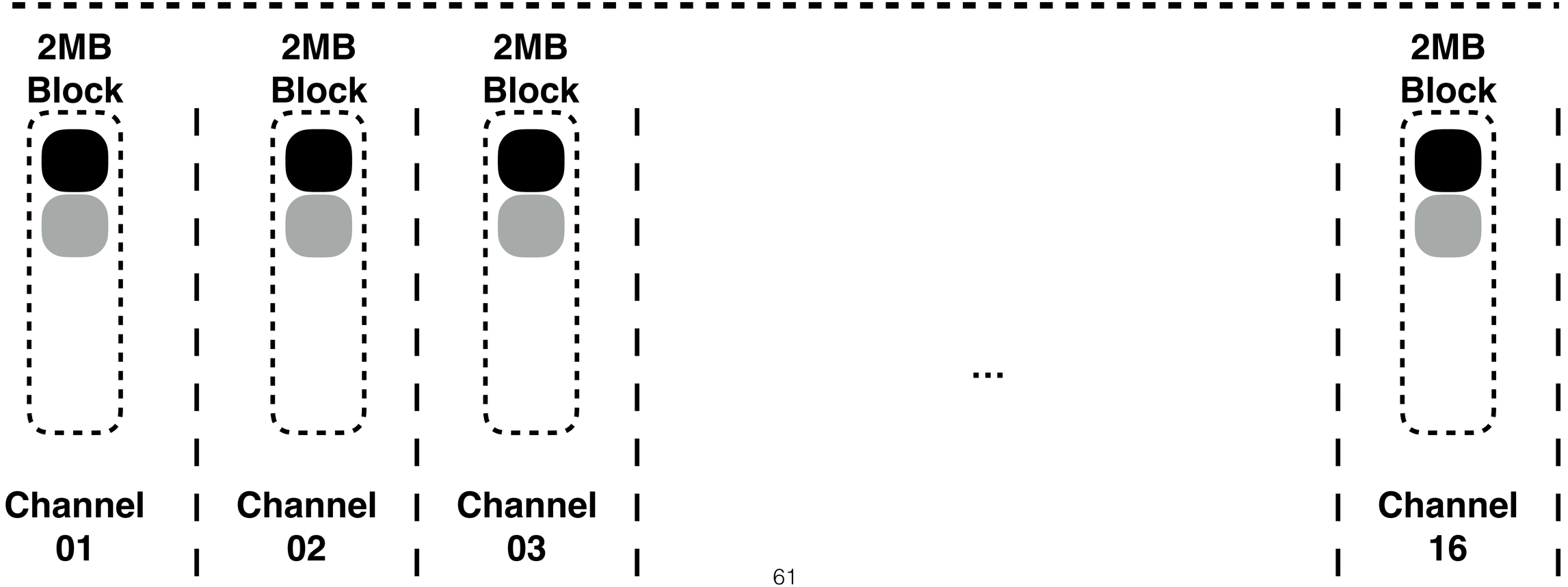
Data of different files is mixed to the same blocks



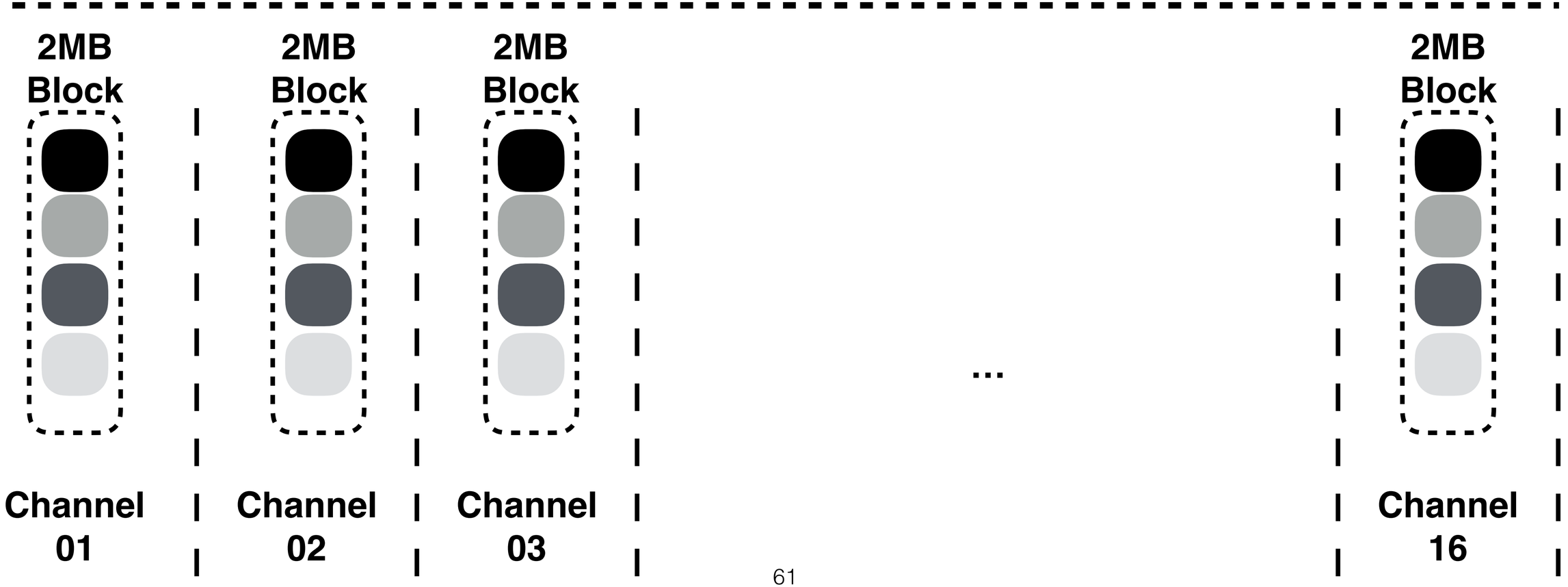
Data of different files is mixed to the same blocks



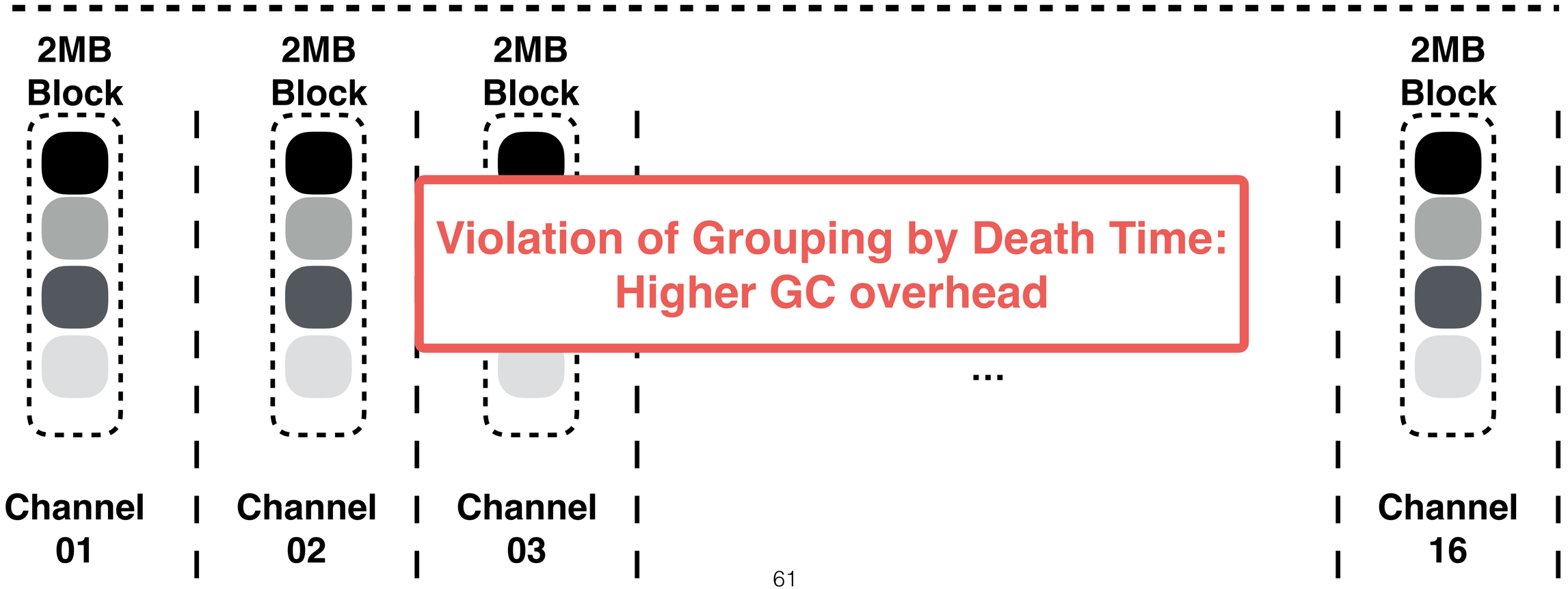
Data of different files is mixed to the same blocks



Data of different files is mixed to the same blocks



Data of different files is mixed to the same blocks



A File System Summary for Grouping by Death Time

- Immediate discarding reduces GC overhead
- All file systems violate the rule
  - XFS violates the least
  - ext4 violates due partially to HDD optimizations
  - F2FS violates the most (delayed discards + IPU)
- F2FS creates another layer of problems
  - Now you have to group data to different F2FS segments by death time, in order to reduce F2FS GC
  - F2FS does not group file data (except for mp3, avi etc.)
  - F2FS will have zombie segments
- Stacking logs may be problematic because of uncoordinated GCs across layers

Lessons Learned

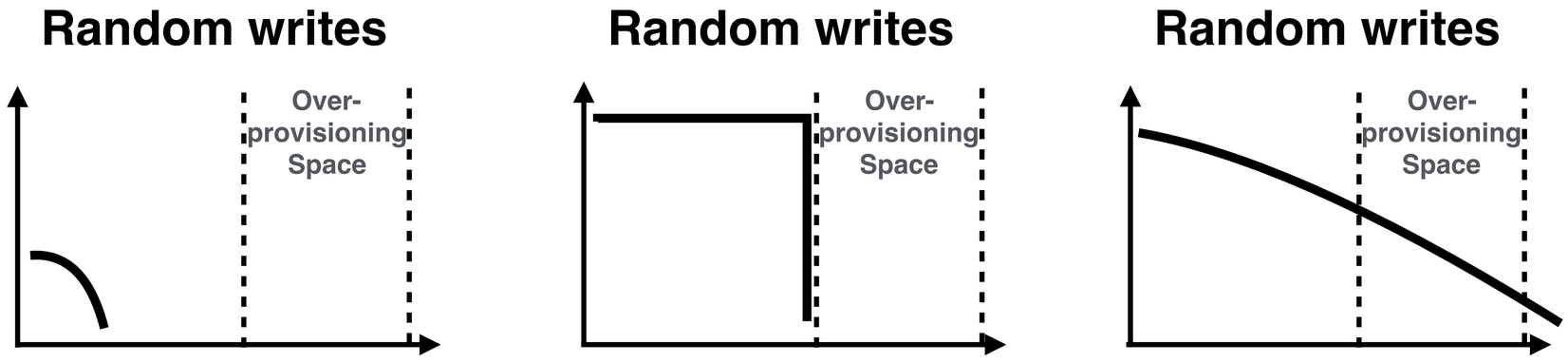
- **The SSD contract is multi-dimensional.**
  - We need more sophisticated tools to analyze workloads.
  - Optimizing for one dimension is not enough.
- **Although not perfect, ext4/XFS perform well upon SSDs.**
- **Myths spread if the unwritten contract is not clarified.**

Myth: random writes increase GC overhead

- Pessimistic views have spread
- Systems are designed based on such views
- However, random writes are irrelevant to GC

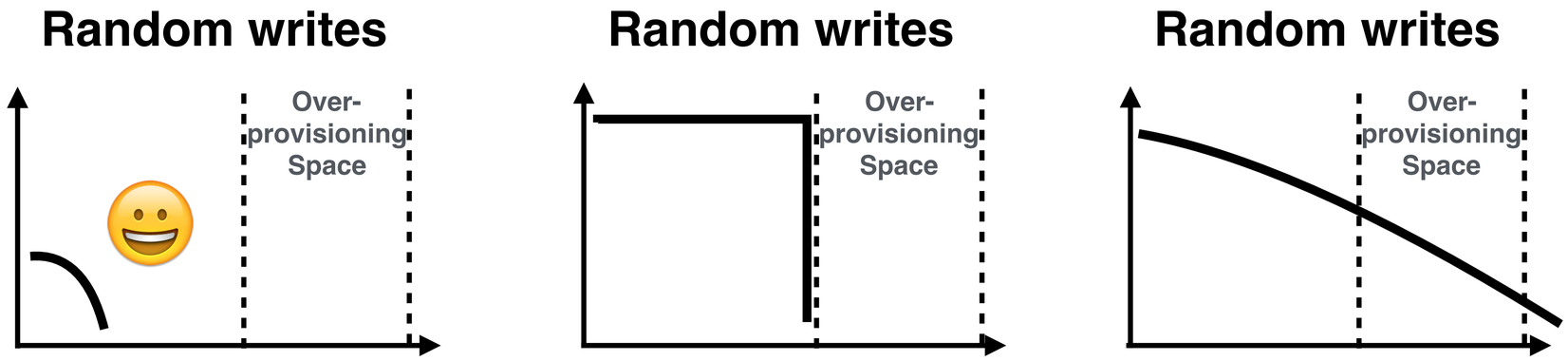
Myth: random writes increase GC overhead

- Pessimistic views have spread
- Systems are designed based on such views
- However, random writes are irrelevant to GC



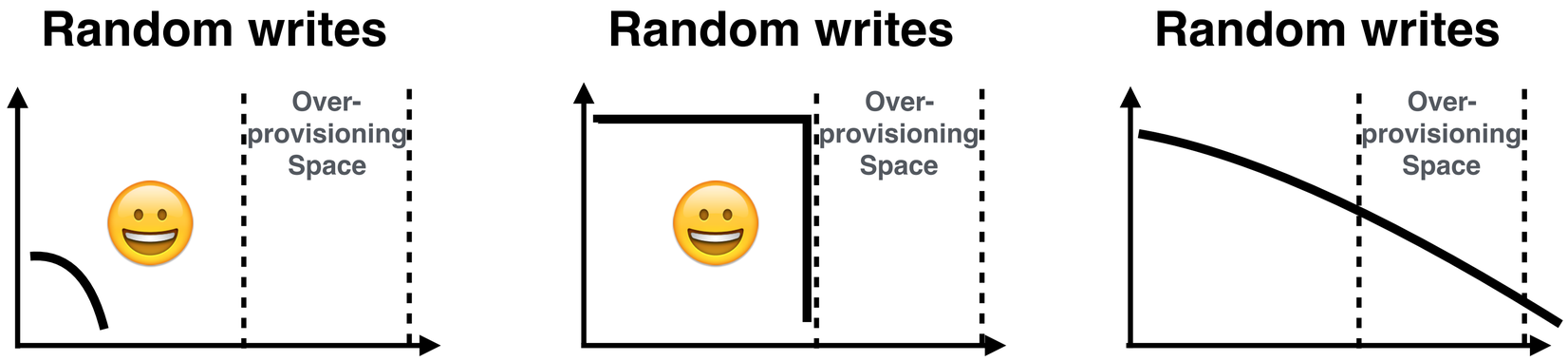
Myth: random writes increase GC overhead

- Pessimistic views have spread
- Systems are designed based on such views
- However, random writes are irrelevant to GC



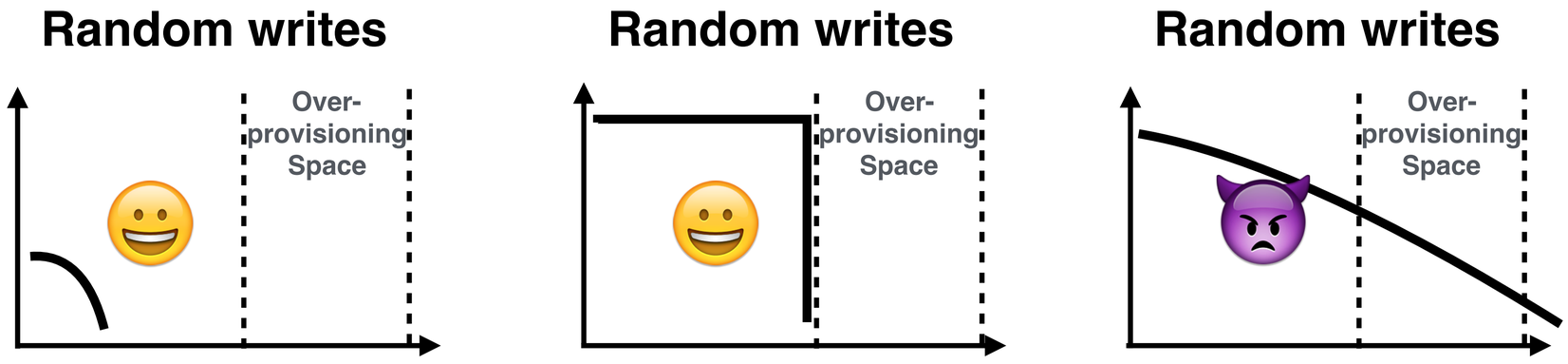
Myth: random writes increase GC overhead

- Pessimistic views have spread
- Systems are designed based on such views
- However, random writes are irrelevant to GC



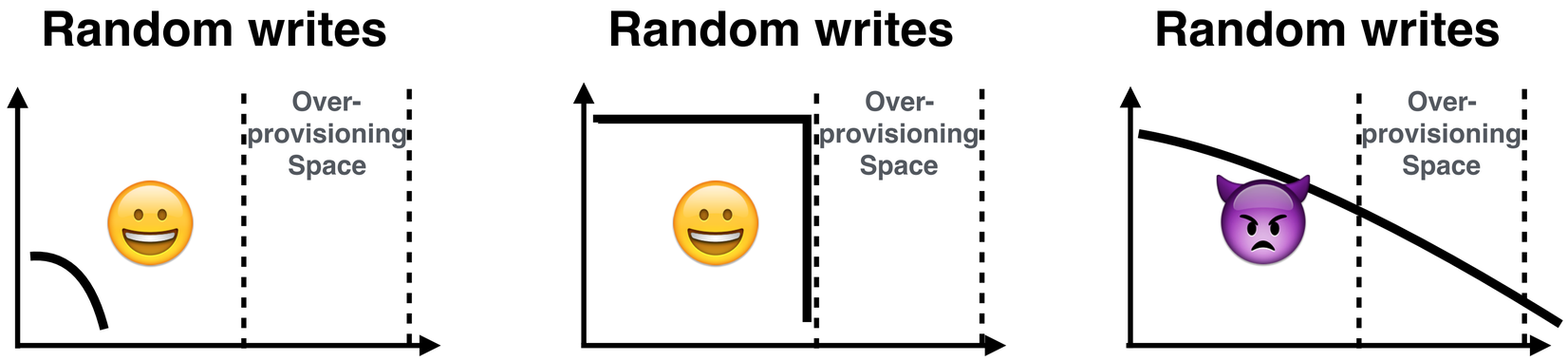
Myth: random writes increase GC overhead

- Pessimistic views have spread
- Systems are designed based on such views
- However, random writes are irrelevant to GC



Myth: random writes increase GC overhead

- Pessimistic views have spread
- Systems are designed based on such views
- However, random writes are irrelevant to GC



Stop using “random writes” to discuss GC

Conclusions

- We reveal the five rules of SSD unwritten contract
- We conduct vertical analysis of applications, file systems and FTLs
- We design tools for SSD workload analysis
- Our rules guide future workload analysis and system designs

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