Enabling Efficient Deletes in Log-Structured KV-Stores

BOSTON
UNIVERSITY

Subhadeep Sarkar

Enabling Efficient Deletes in Log-Structured KV-Stores

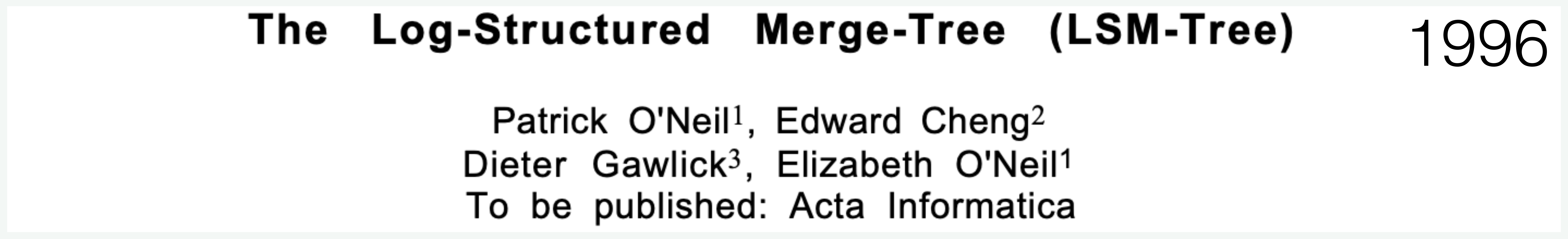
Lethe [\ˈlē-thē\ ]

n: the goddess of forgetfulness

Log-**S**tructured **M**erge-tree

LSM-tree

LSM-tree



The Log-Structured Merge-Tree (LSM-Tree)

Patrick O'Neil[1], Edward Cheng[2]
Dieter Gawlick[3], Elizabeth O'Neil[1]
To be published: Acta Informatica

1996

LSM-tree

O'Neil *et al.*

1996

LSM-tree

O’Neil *et al.*

1996

LSM-tree

O’Neil *et al.*

1996

Bigtable

2006

LSM-tree

O'Neil *et al.*

1996

Bigtable

2006

APACHE HBASE

2007

LSM-tree

O’Neil *et al.*

1996

2006

2007

2010

Bigtable
APACHE HBASE
cassandra

Bigtable
APACHE HBASE
cassandra
levelDB

LSM-tree

O’Neil *et al.*

1996

2006 2007 2010 2011

LSM-tree

O'Neil *et al.*

1996

Bigtable
APACHE HBASE
cassandra
levelDB
RocksDB

2006 2007 2010 2011 2013

LSM-tree
O'Neil *et al.*
1996

Bigtable
2006
APACHE HBASE
2007
cassandra
2010
levelDB
2011
RocksDB
2013
2021

LSM-tree
O'Neil *et al.*

Bigtable
APACHE HBASE
cassandra
levelDB
RocksDB

1996 2006 2007 2010 2011 2013 2021

LSM-tree

WT

levelDB

SCYLLA

cassandra

RocksDB

DynamoDB

accumulo™

influxdb

QuasarDB

riak

SQLite

Bigtable

tarantool

APACHE HBASE

2021

LSM-tree

APACHE HBASE

WT

levelDB

SCYLLA

cassandra

tarantool

RocksDB

Bigtable

DynamoDB

accumulo™

SQLite

influxdb

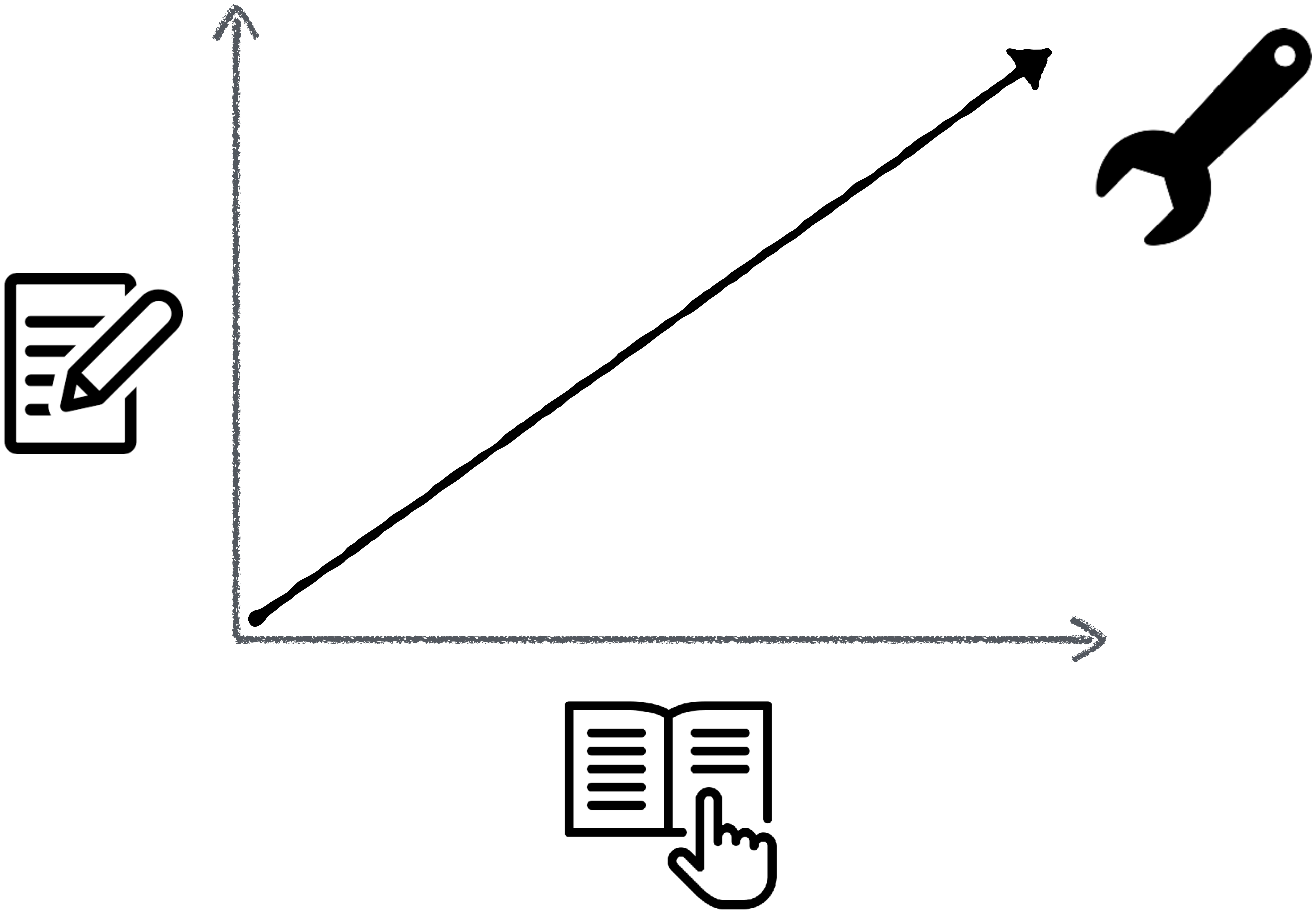
riak

QuasarDB

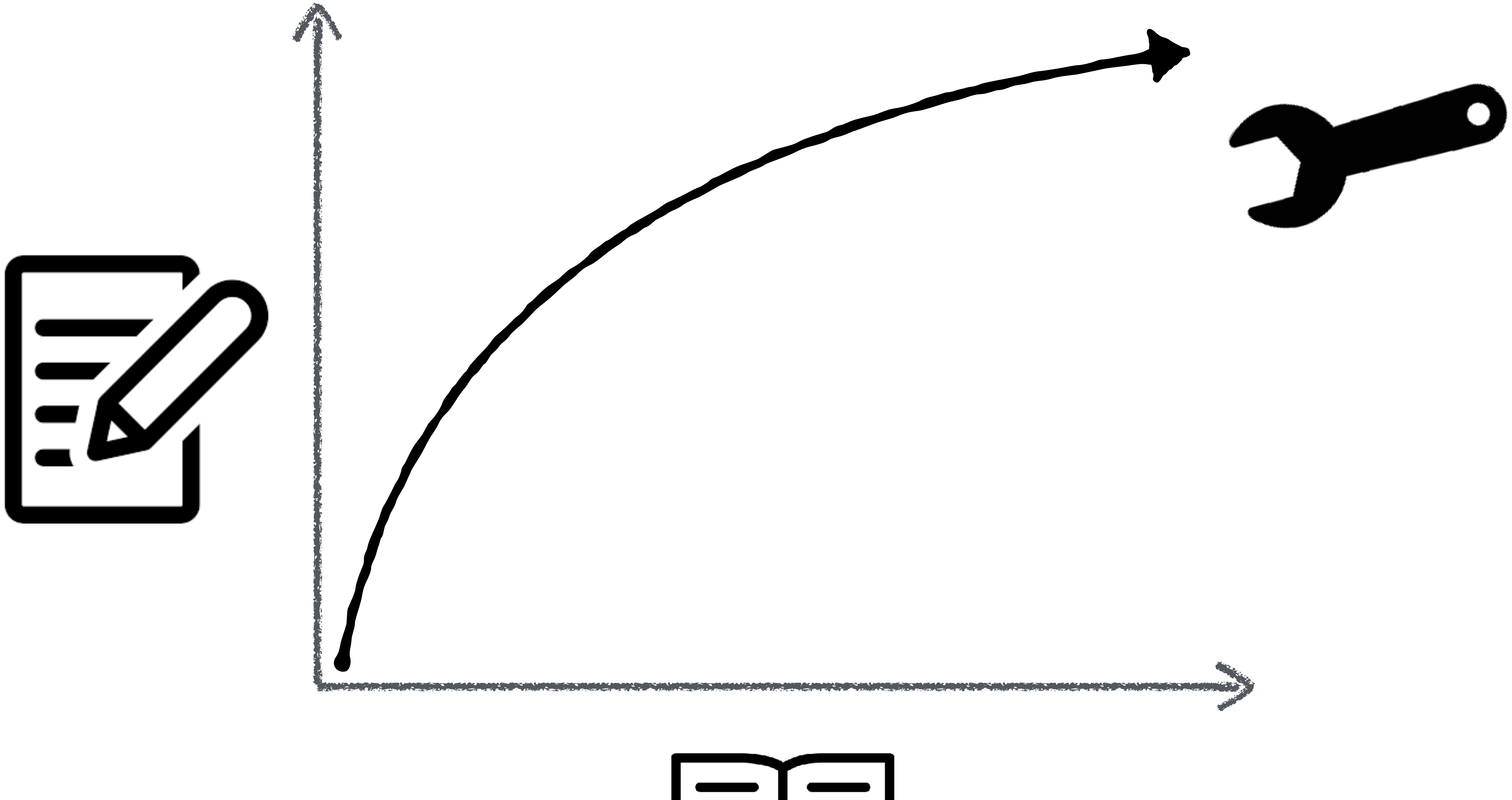
Why **LSM** ?

Why **LSM** ?

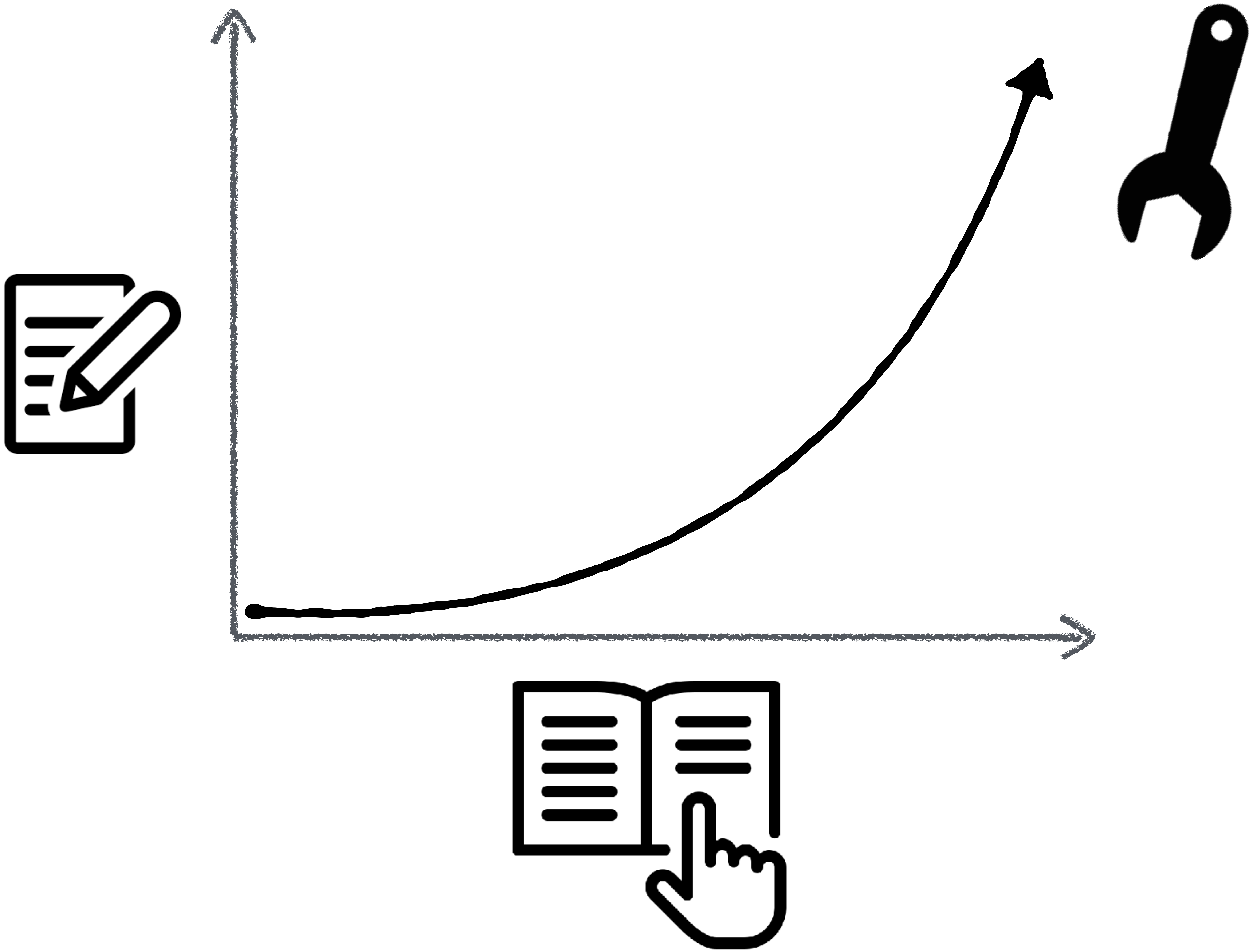
Why **LSM** ?



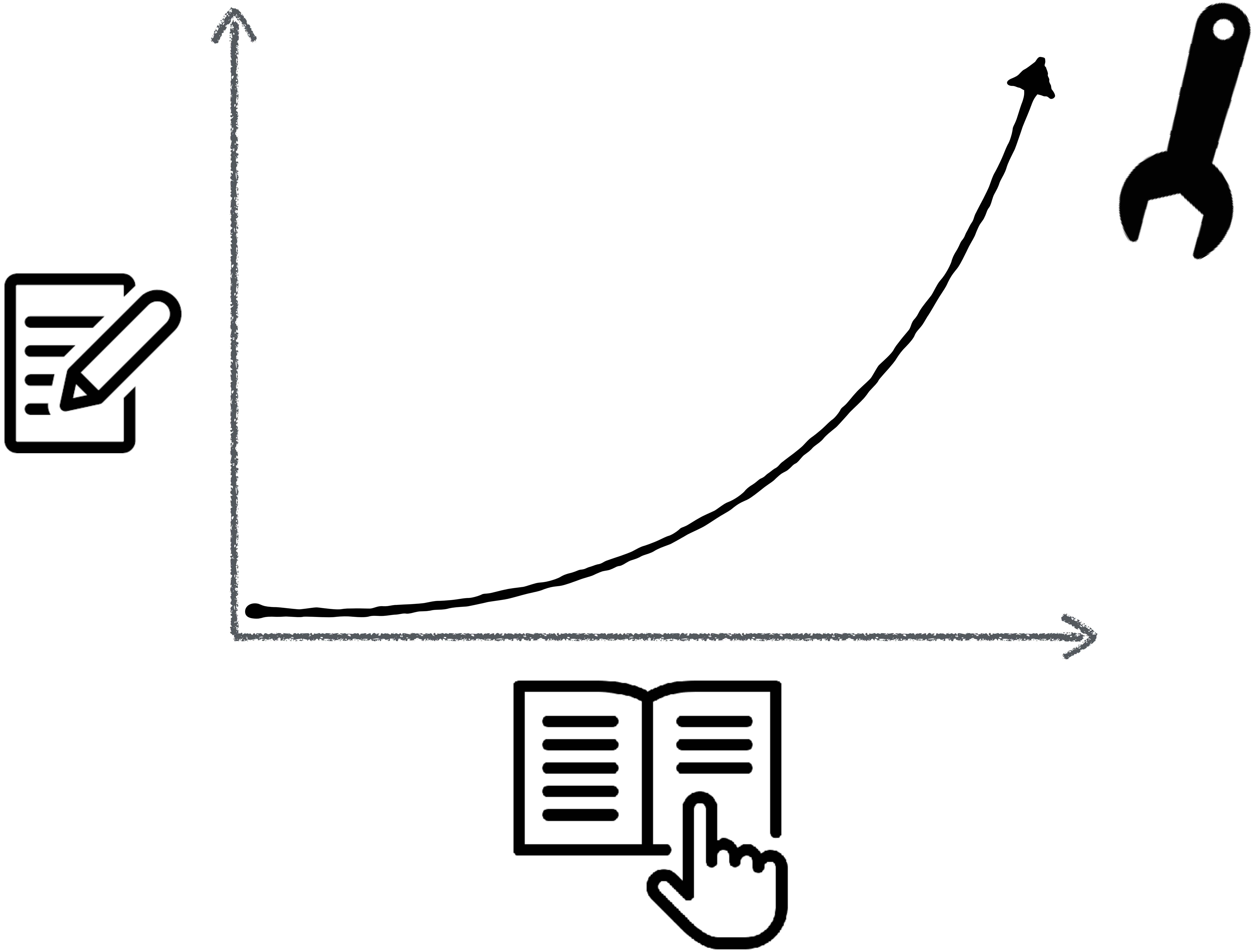
Why **LSM** ?



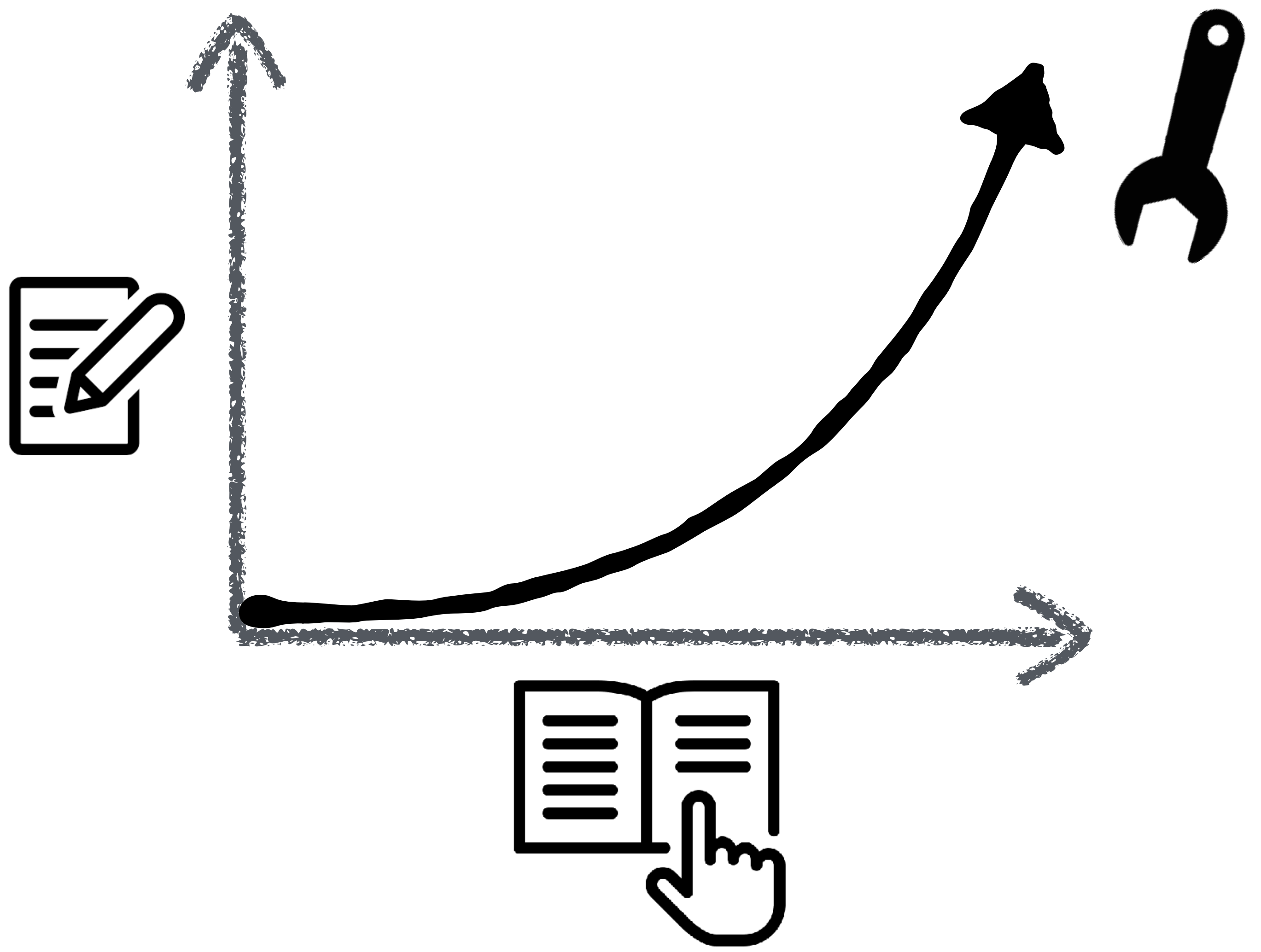
Why **LSM** ?



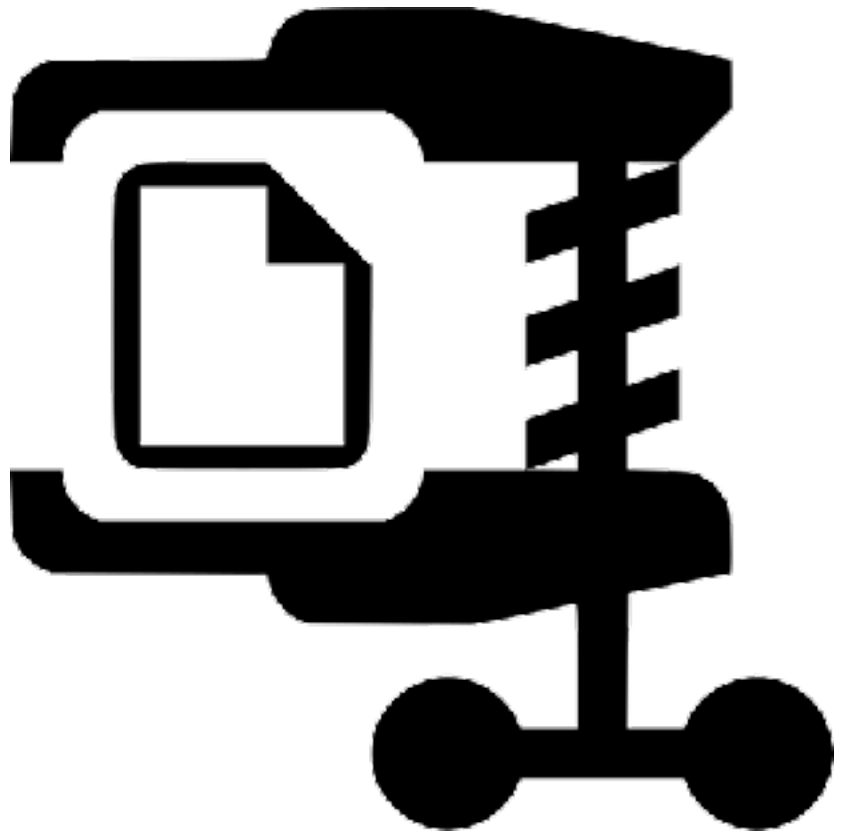
Why **LSM** ?



Why **LSM** ?



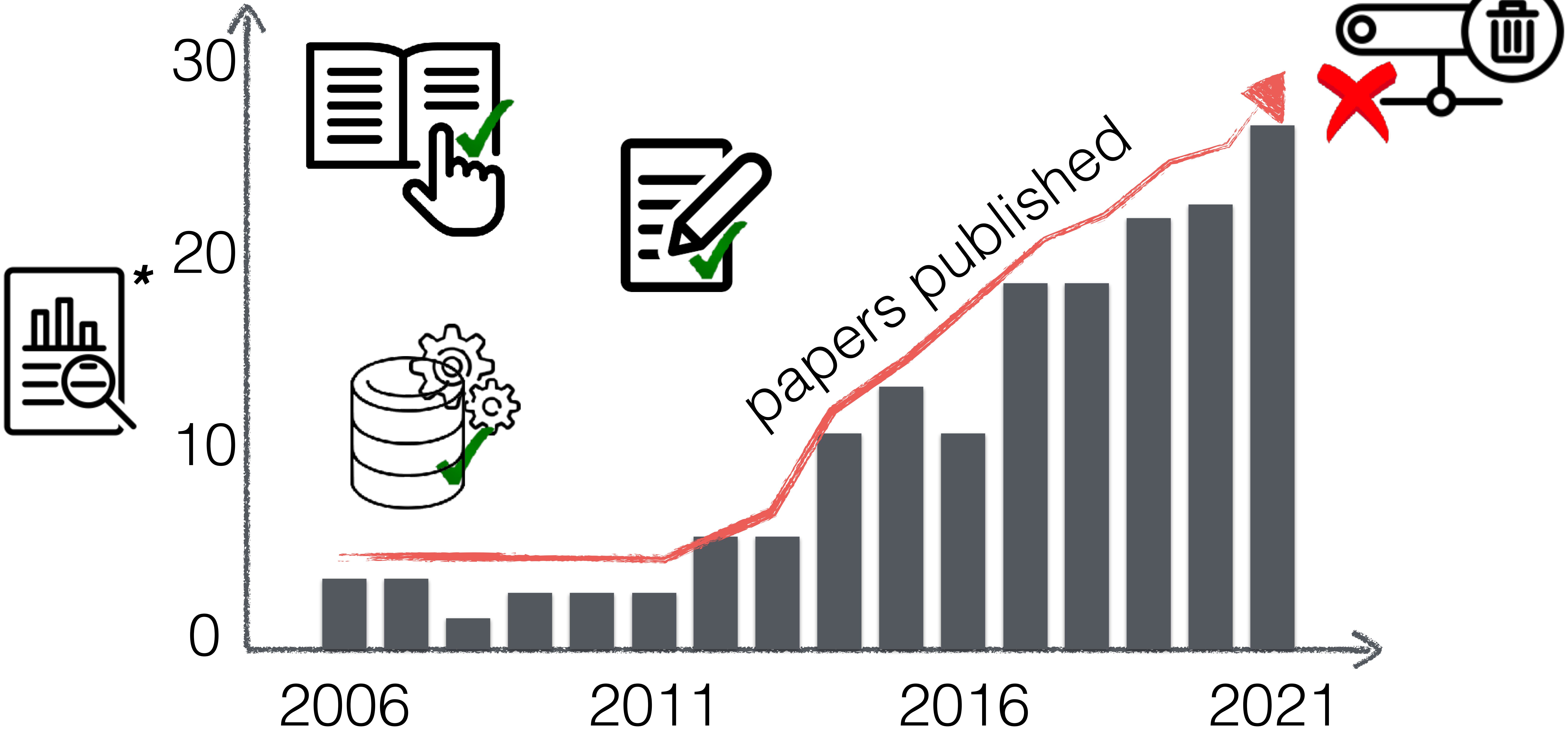
tunable read-write performance



good space utilization

scales well

Research Trend

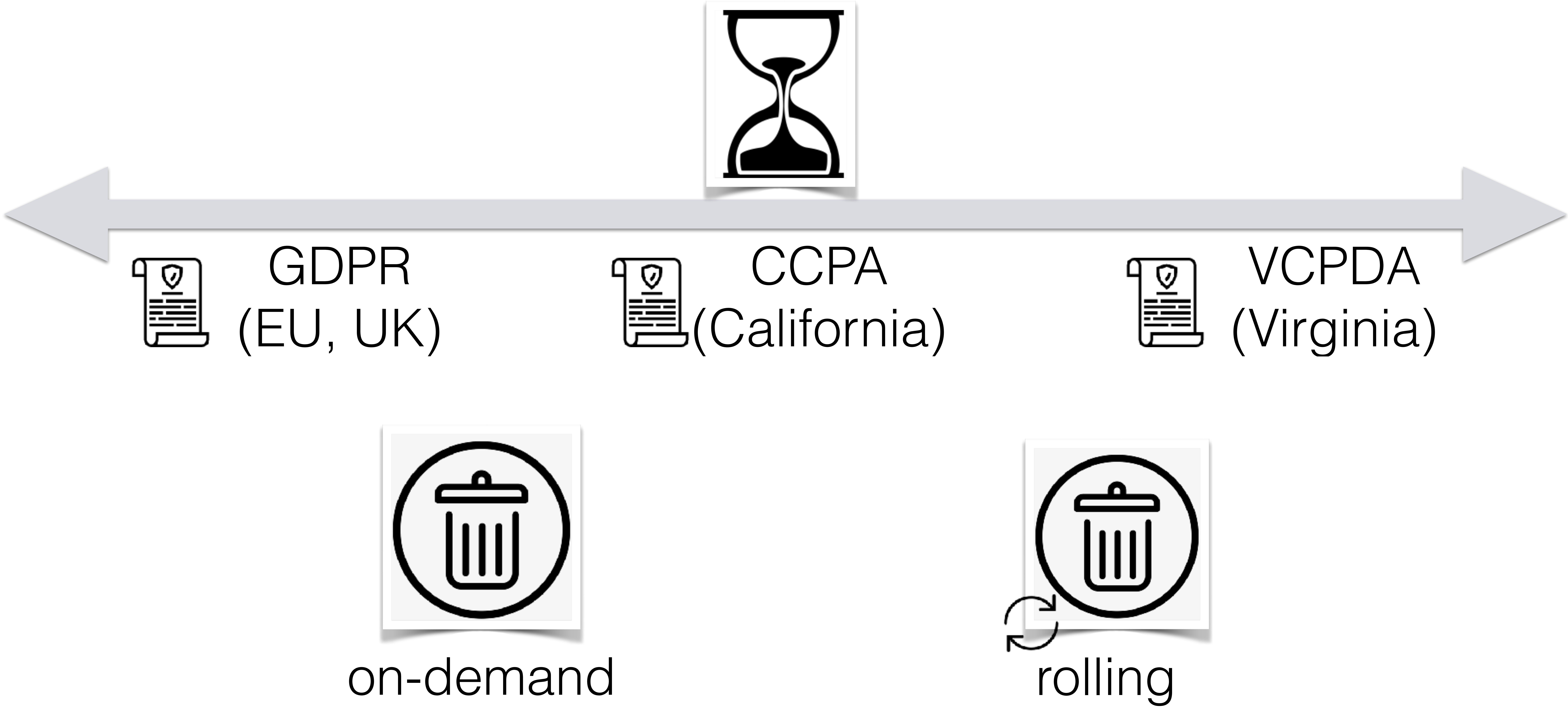


* *data from DBLP*

Large-scale production	Internal db ops	Privacy
ZippyDB **25.2M/day**	table drop	CCPA (California)
	data migration	GDPR (EU, UK)
UP2X **92.5M merge through deletes**	cleanup /gc	VCPDA (Virginia)

Large-scale production	Internal db ops	Privacy
ZippyDB **25.2M/day**	table drop	CCPA (California)
UP2X **92.5M merge through deletes**	data migration	GDPR (EU, UK)
	cleanup /gc	VCPDA (Virginia)

Large-scale production	Internal db ops	Privacy
ZippyDB **25.2M/day**	table drop	GDPR (EU, UK)
UP2X **92.5M merge through deletes**	data migration	CCPA (California)
	cleanup /gc	VCPDA (Virginia)



GDPR (EU, UK)

CCPA (California)

VCPDA (Virginia)

on-demand

rolling

delete all data for user X within D days

keep deleting all data older than D days

What do **LSM** systems lack today?

supporting diverse delete types

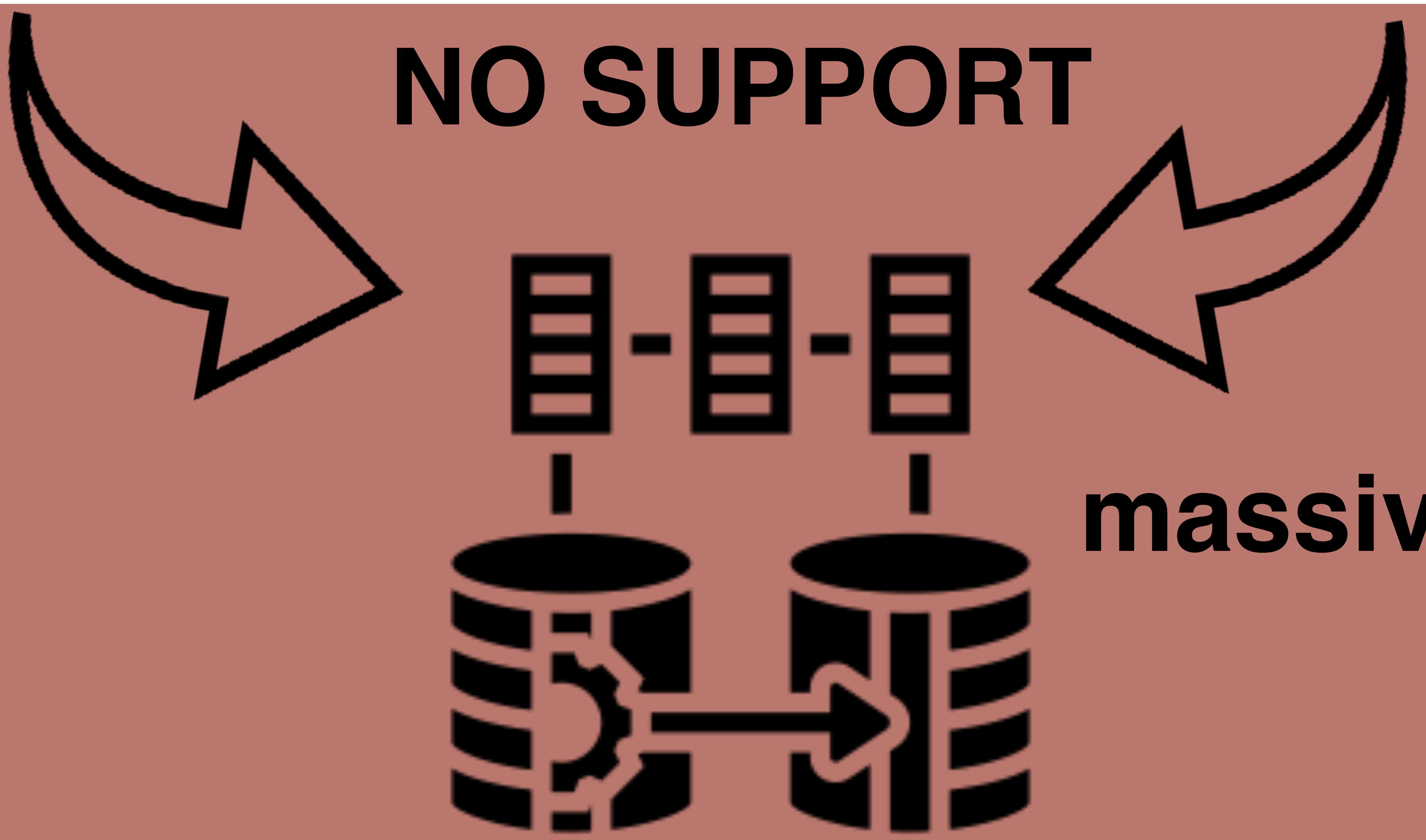
time-bound deletes

NO SUPPORT

What do **LSM** systems lack today?

supporting diverse delete types

time-bound deletes



NO SUPPORT

massive data movement

What do LSM systems **need** today?

time-bound deletes

overall performance

supporting diverse deletes

Outline

Part 1: **LSM Basics**

Part 2: **The Problem: Deletes in LSMs**

Part 3: **Lethe: Enabling Efficient Deletes**

Outline

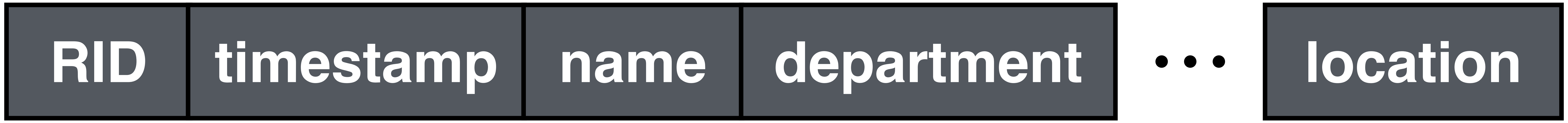
Part 1: **LSM Basics**

Part 2: **The Problem: Deletes in LSMs**

Part 3: **Lethe: Enabling Efficient Deletes**

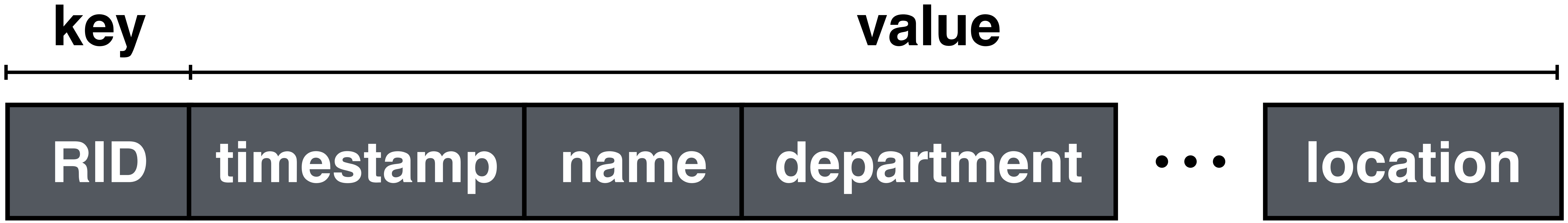
LSM Basics

key-value pairs



RID timestamp name department ••• location

LSM Basics

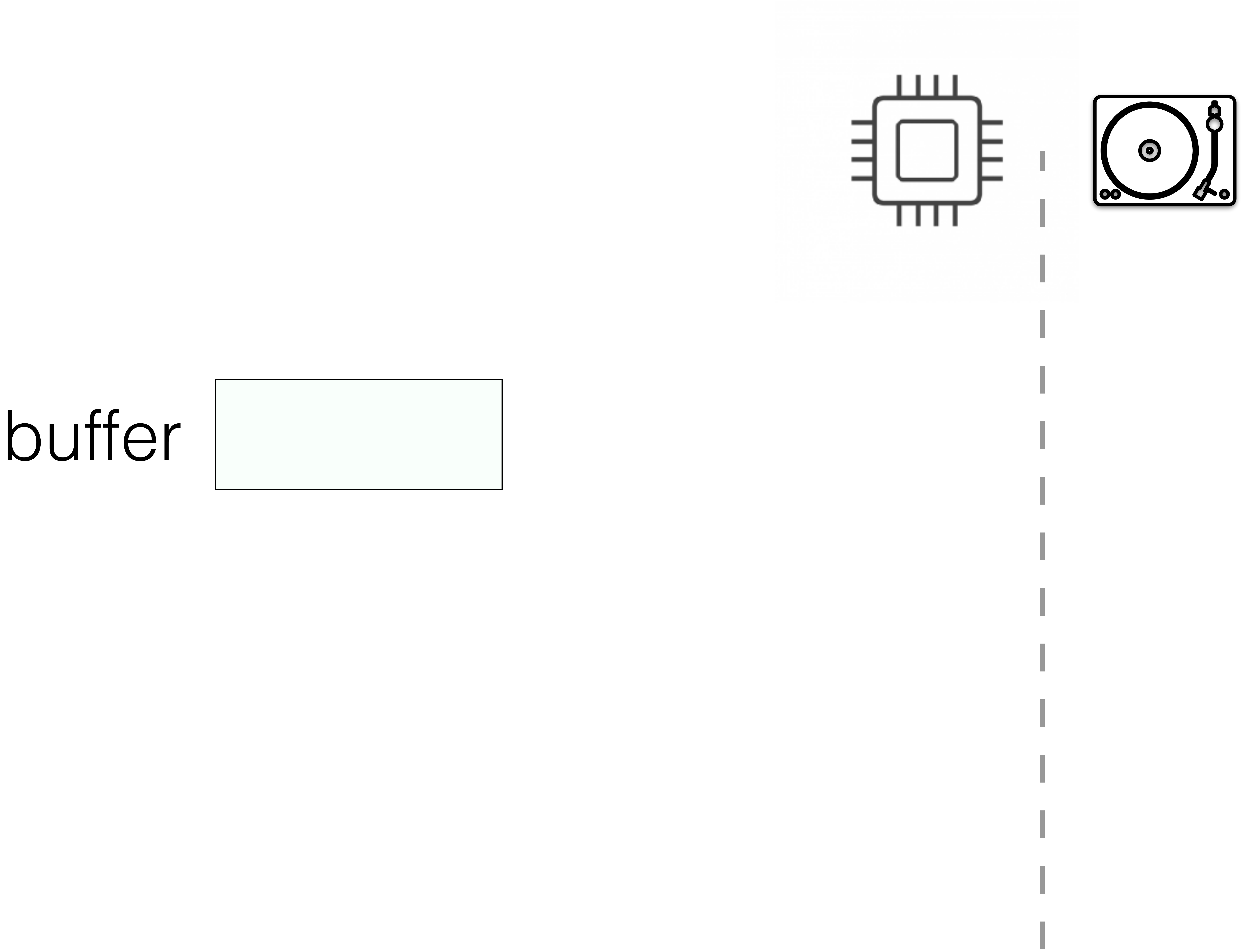


key

value

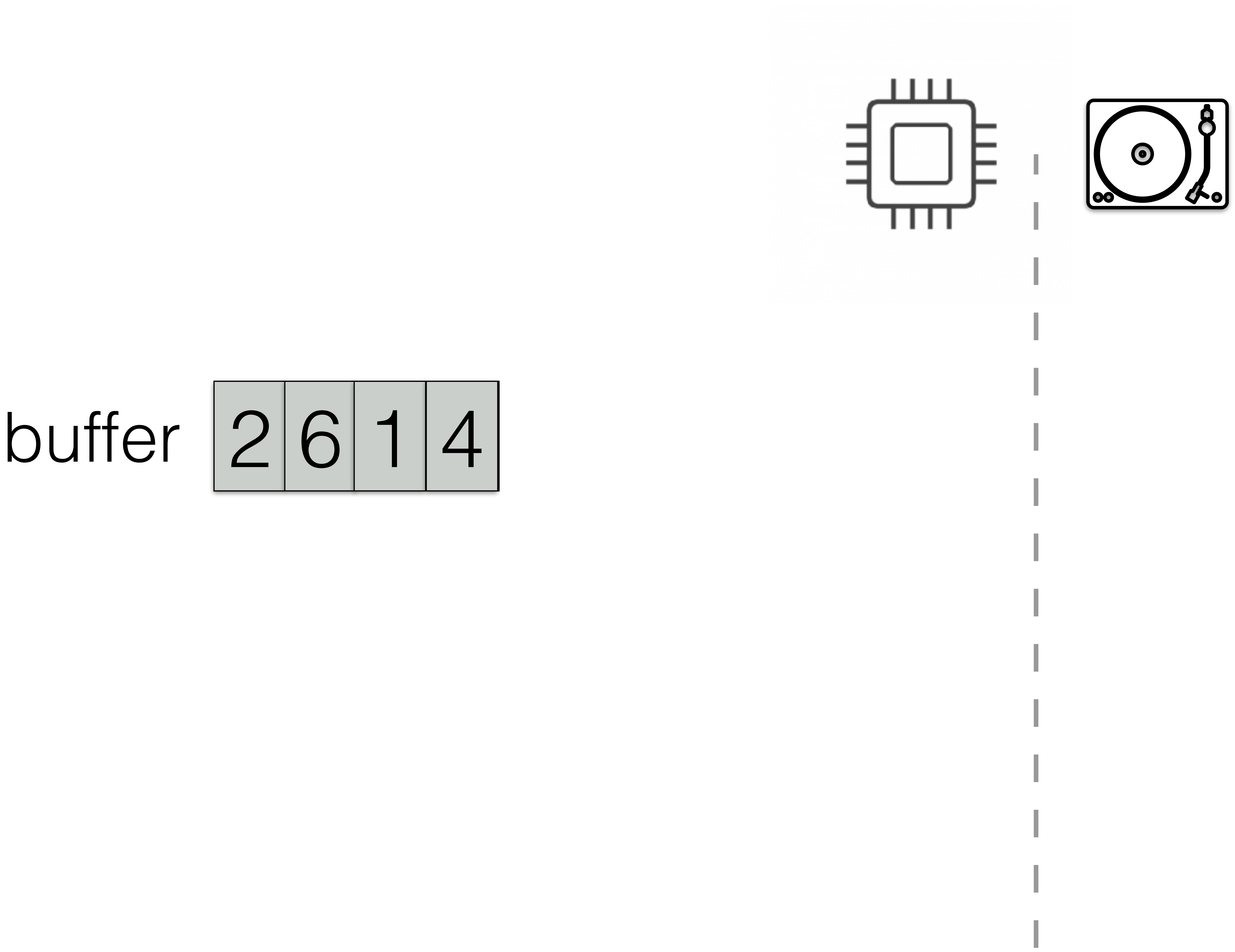
RID | timestamp | name | department ··· location

LSM Basics



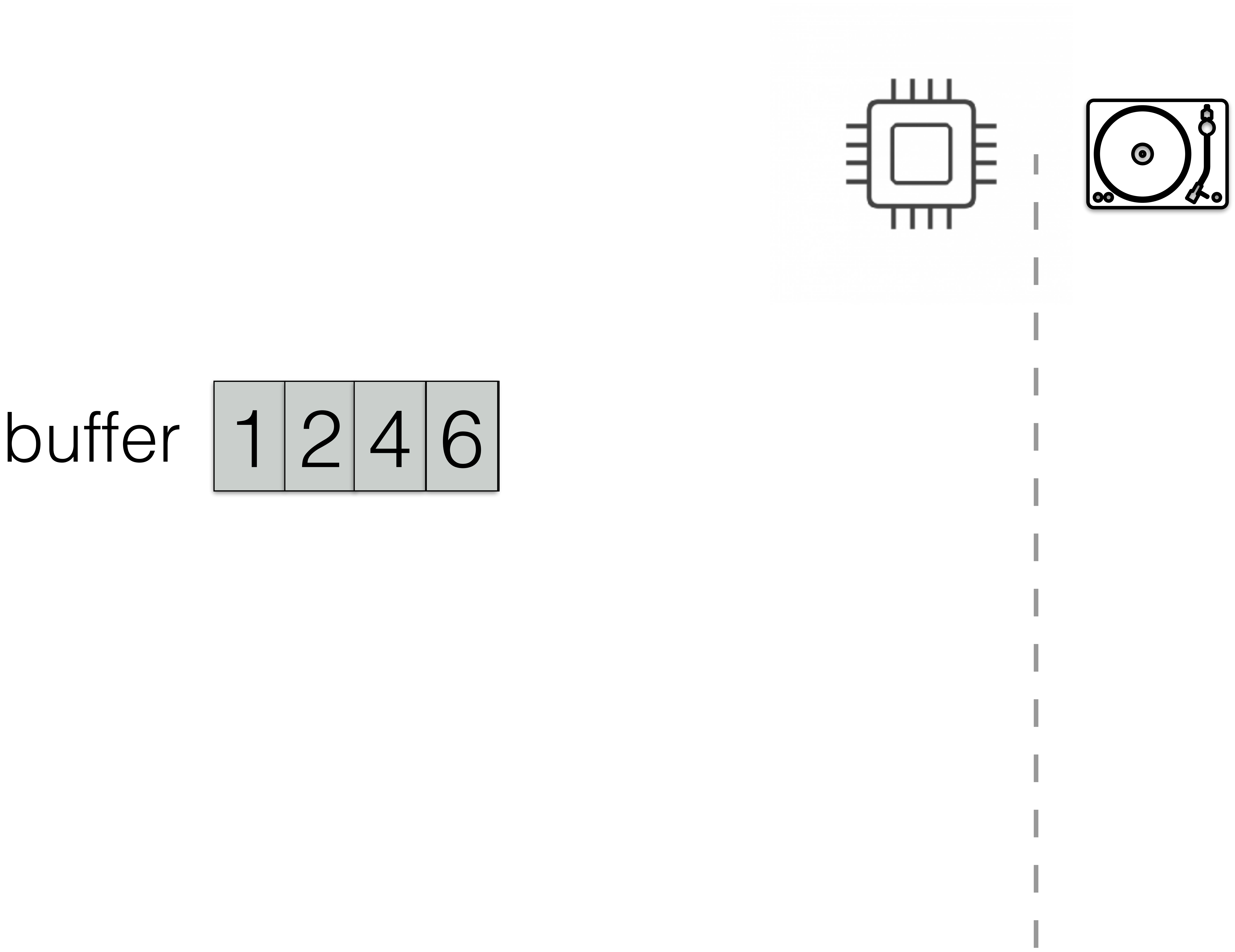
buffer

LSM Basics



buffer	2	6	1	4

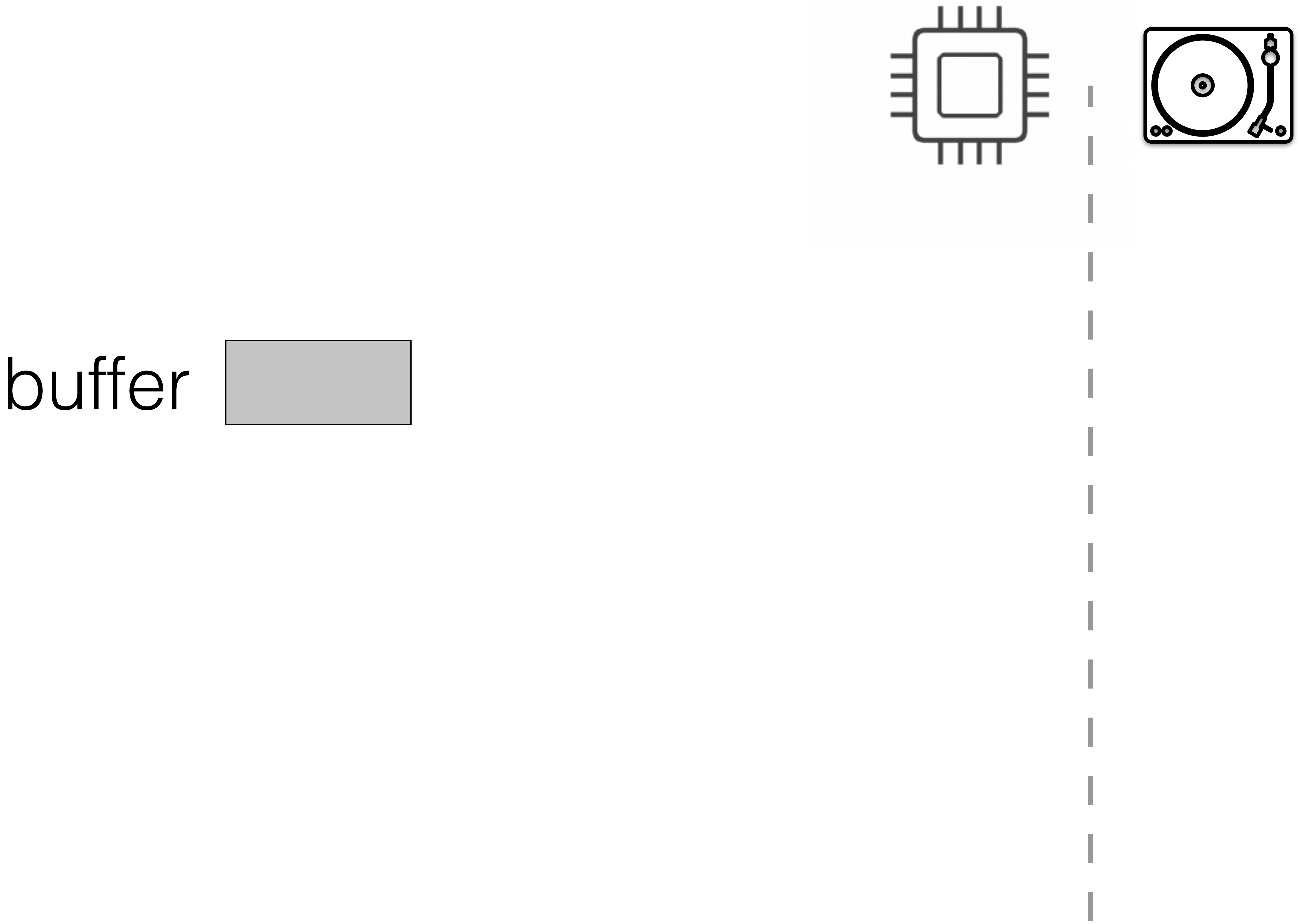
LSM Basics



buffer

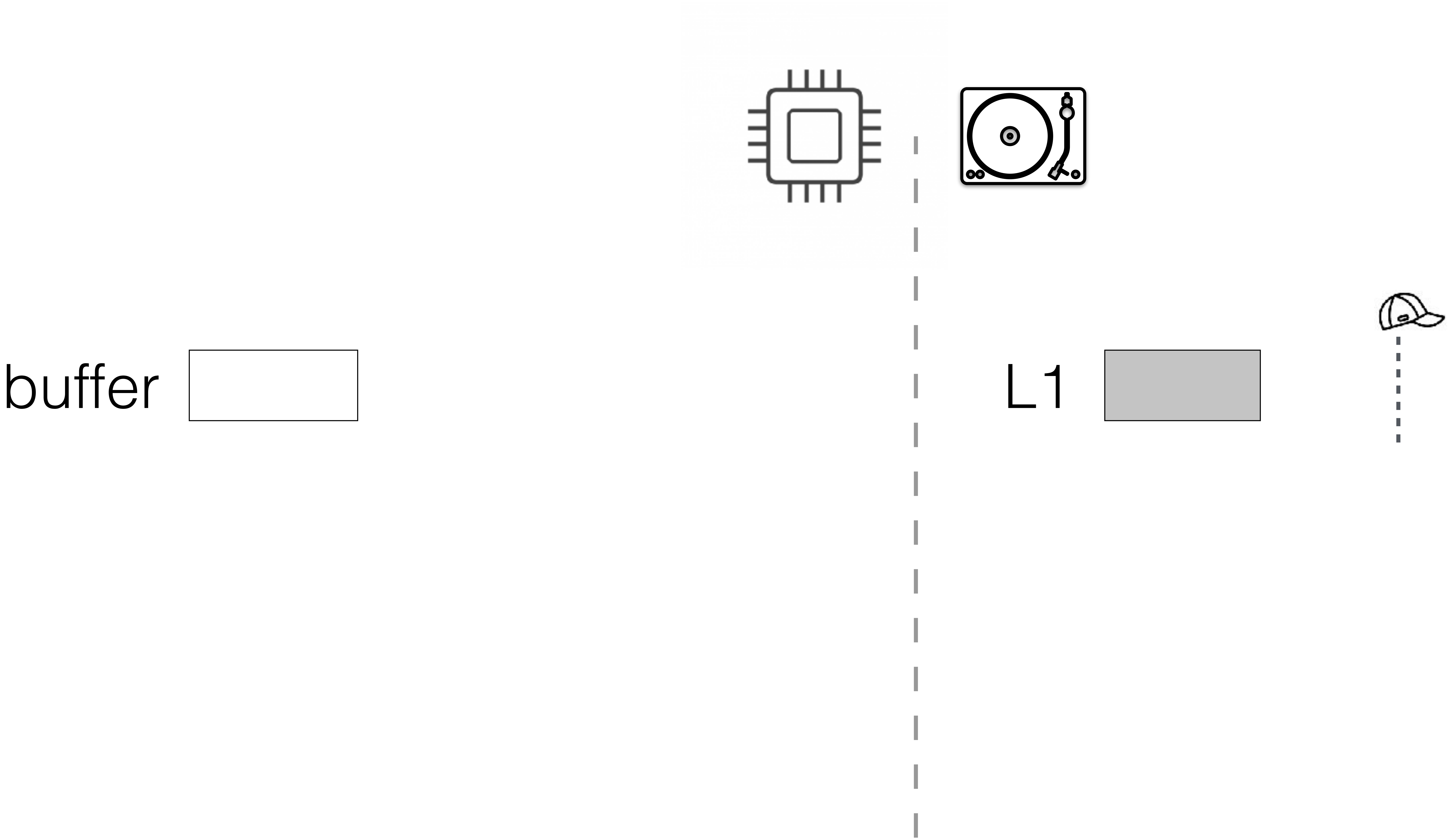
1	2	4	6

LSM Basics



buffer

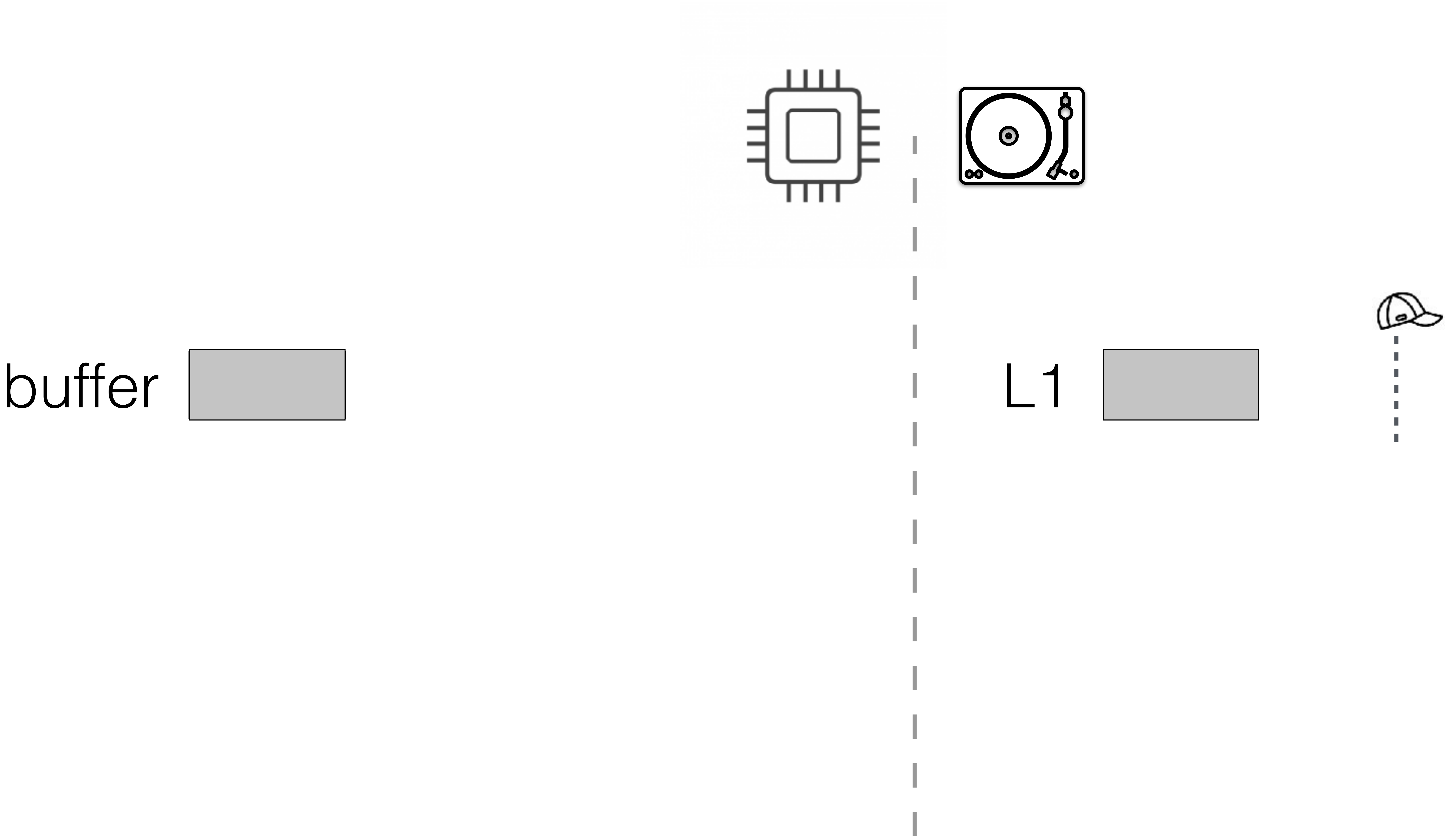
LSM Basics



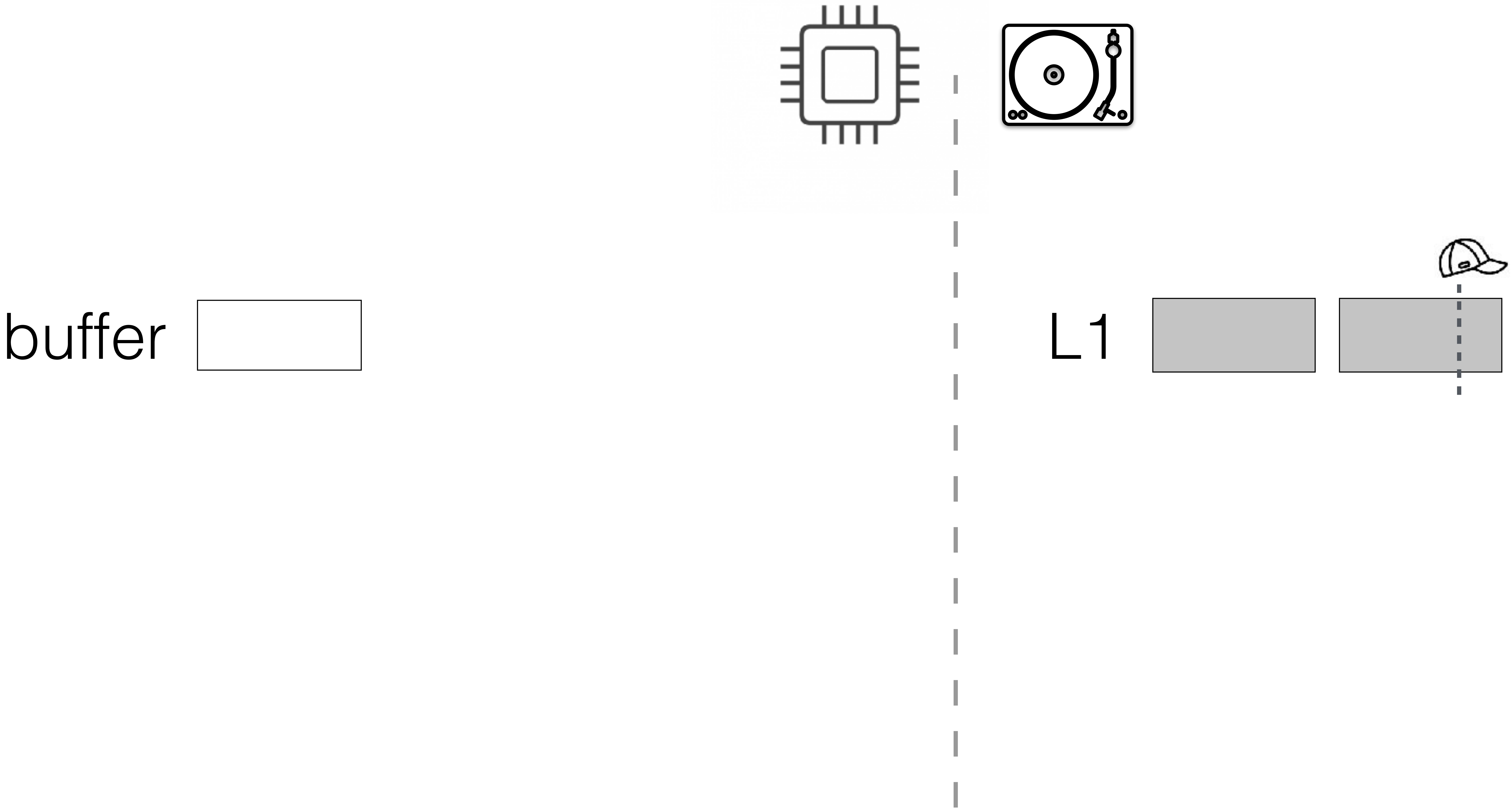
buffer

L1

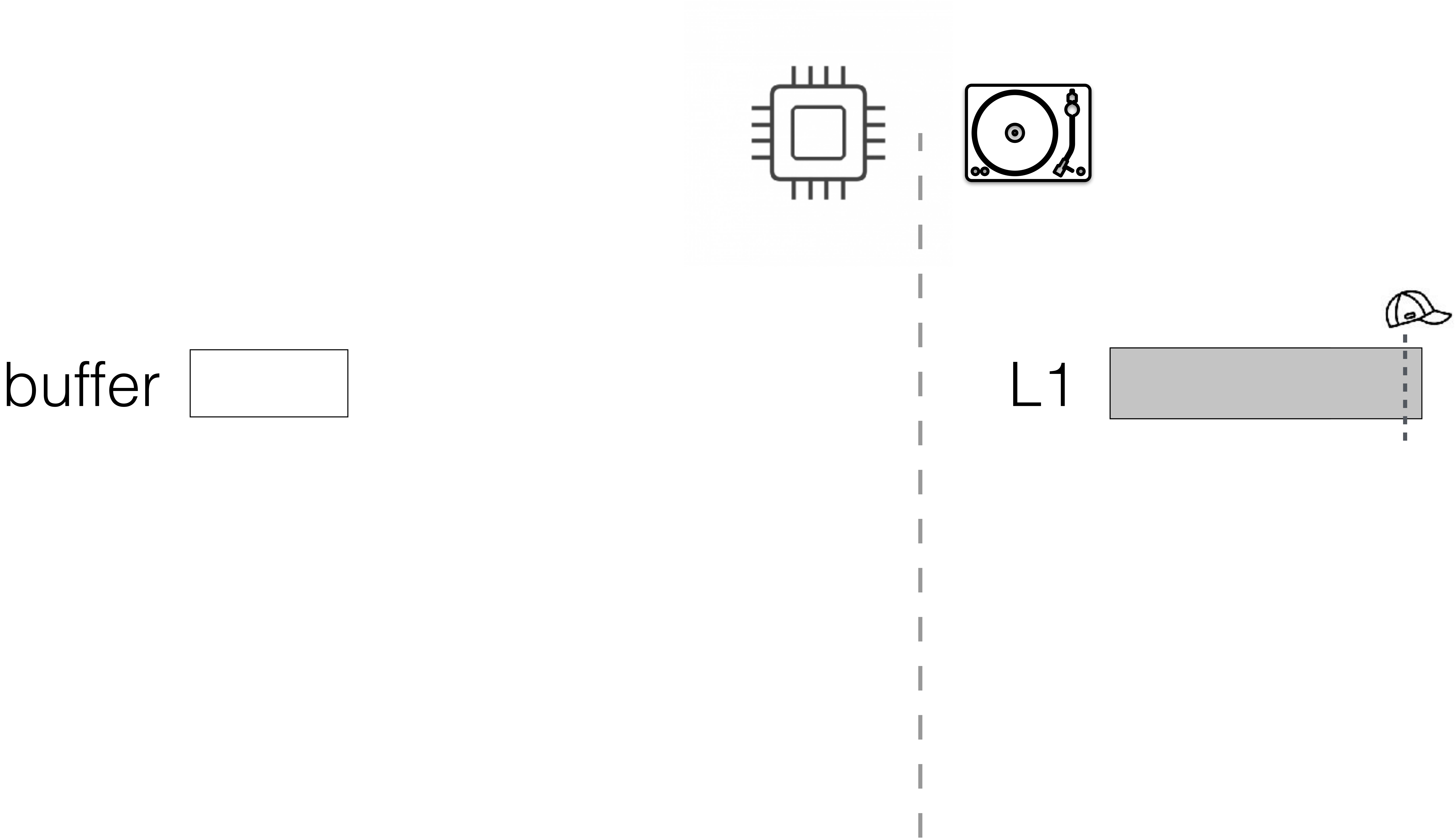
LSM Basics



LSM Basics



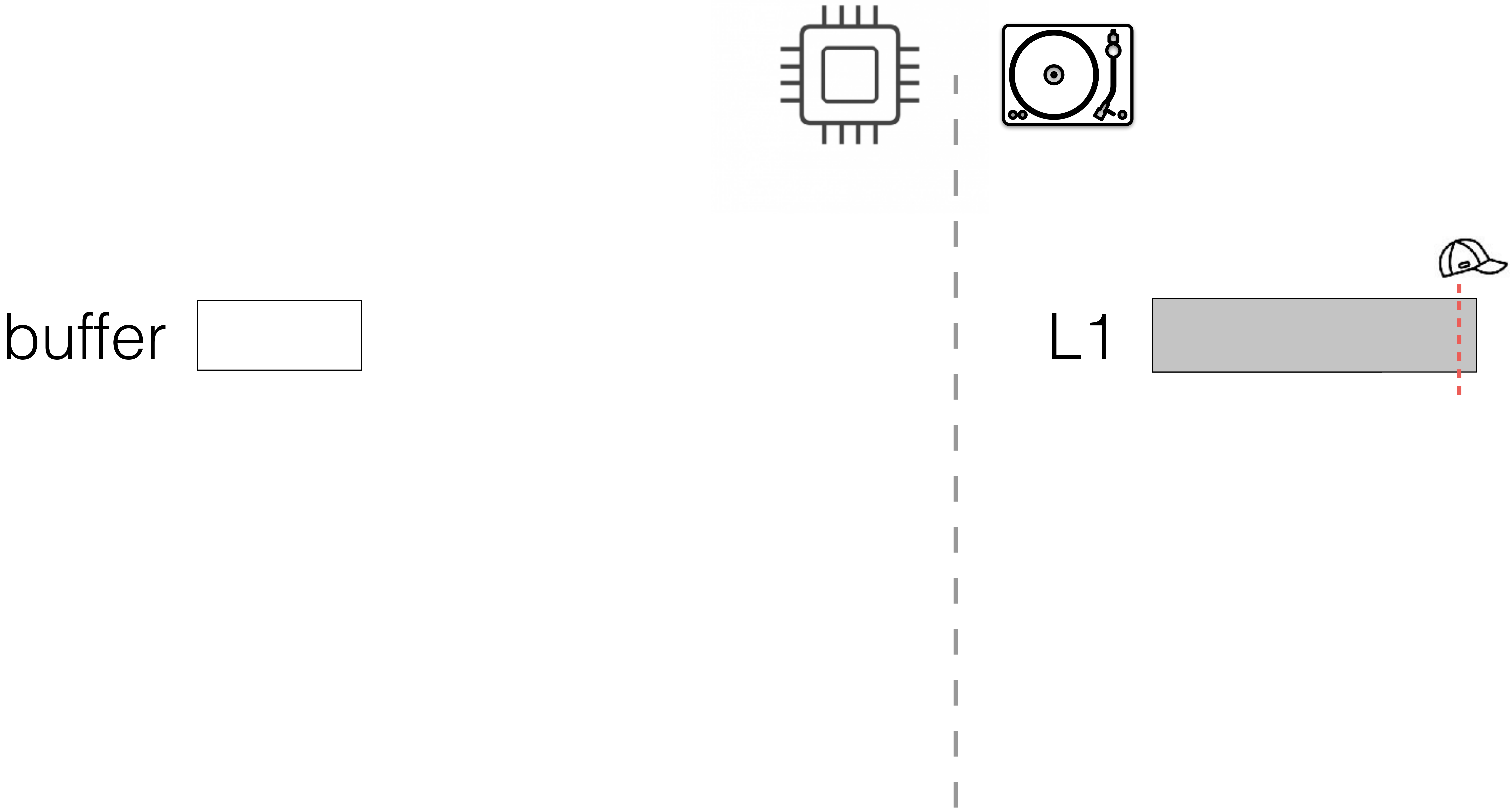
LSM Basics



buffer

L1

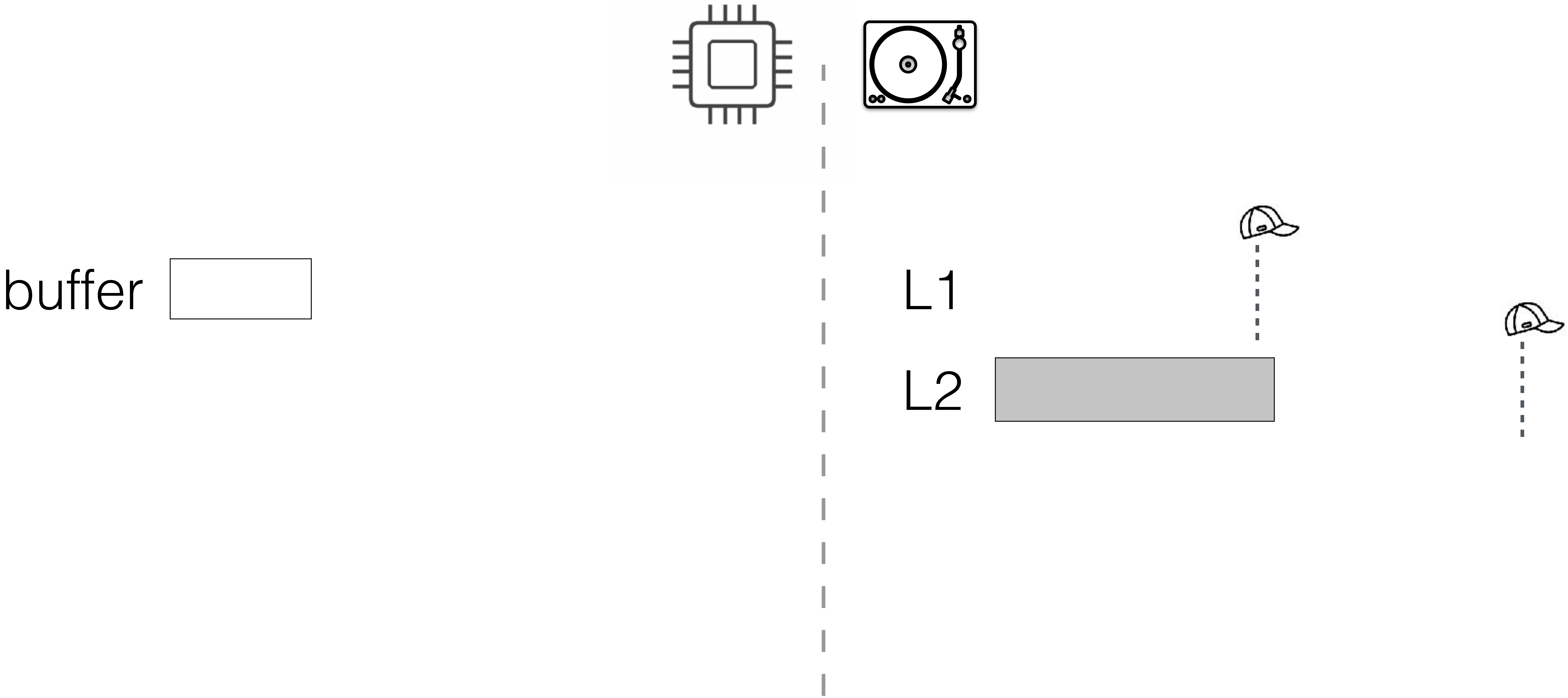
LSM Basics



buffer

L1

LSM Basics

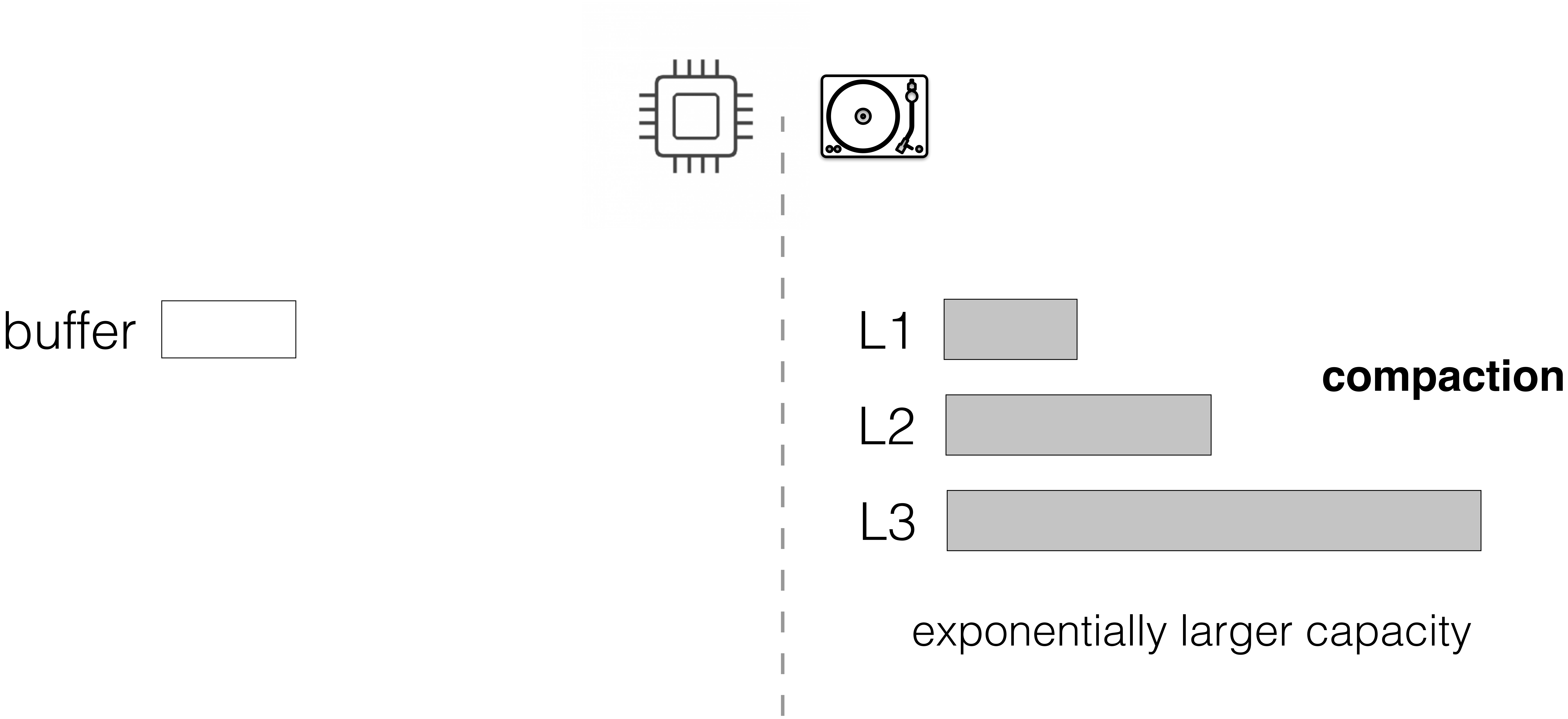


buffer

L1

L2

LSM Basics



buffer

L1

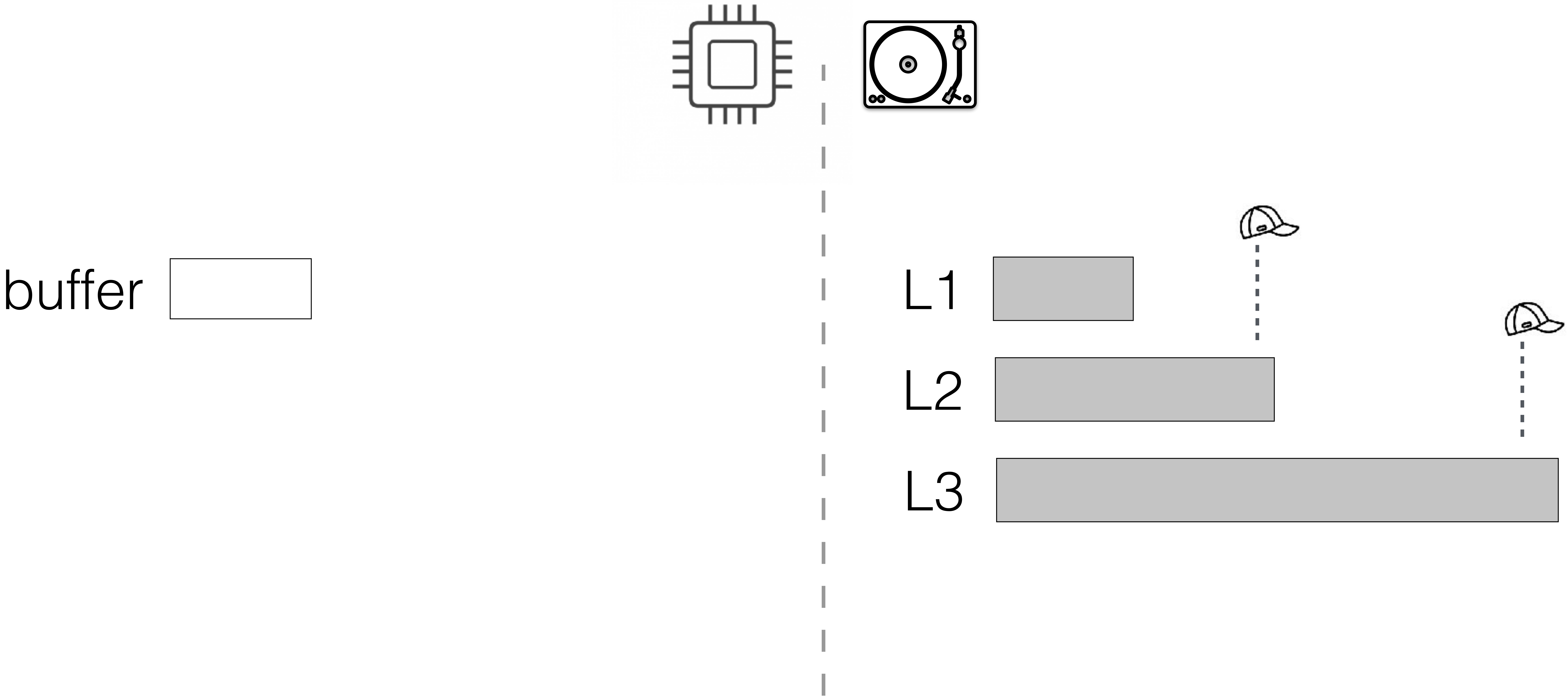
L2

L3

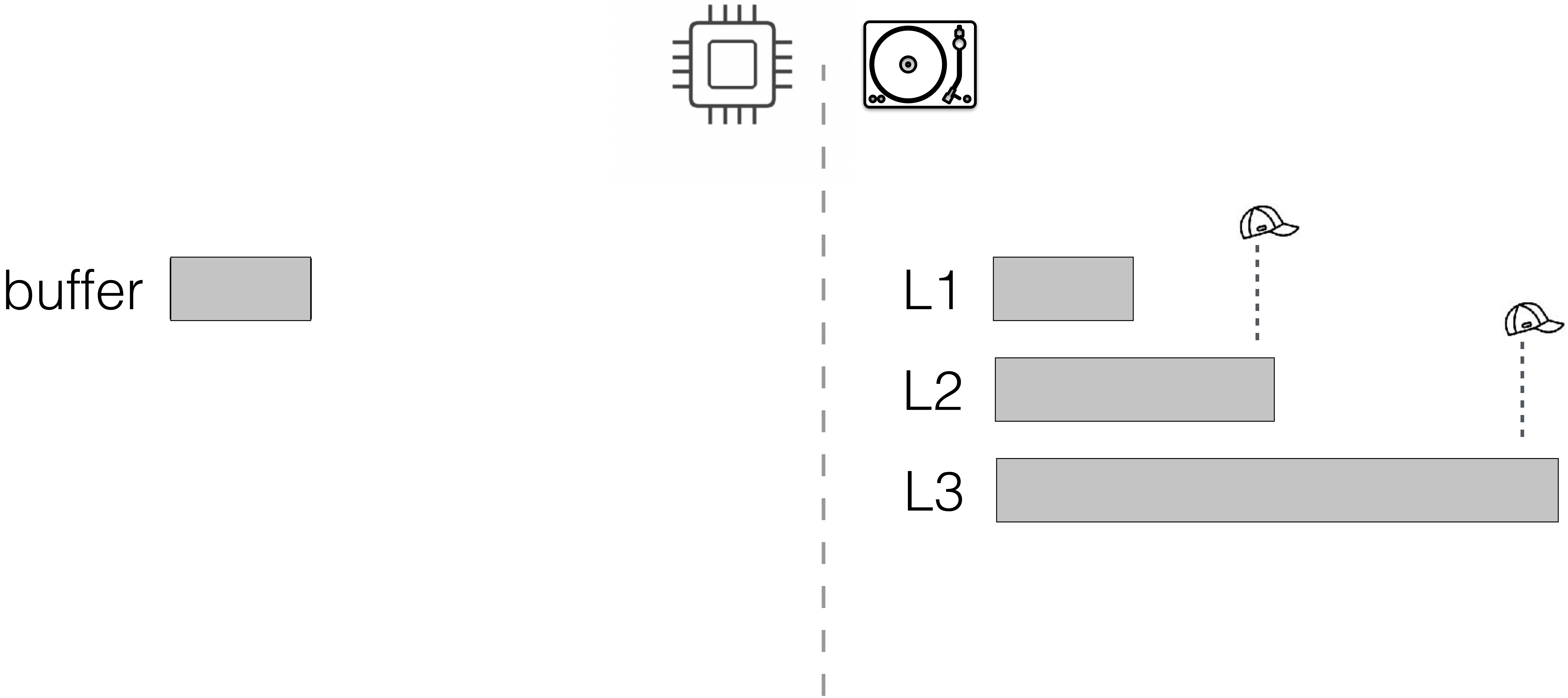
compaction

exponentially larger capacity

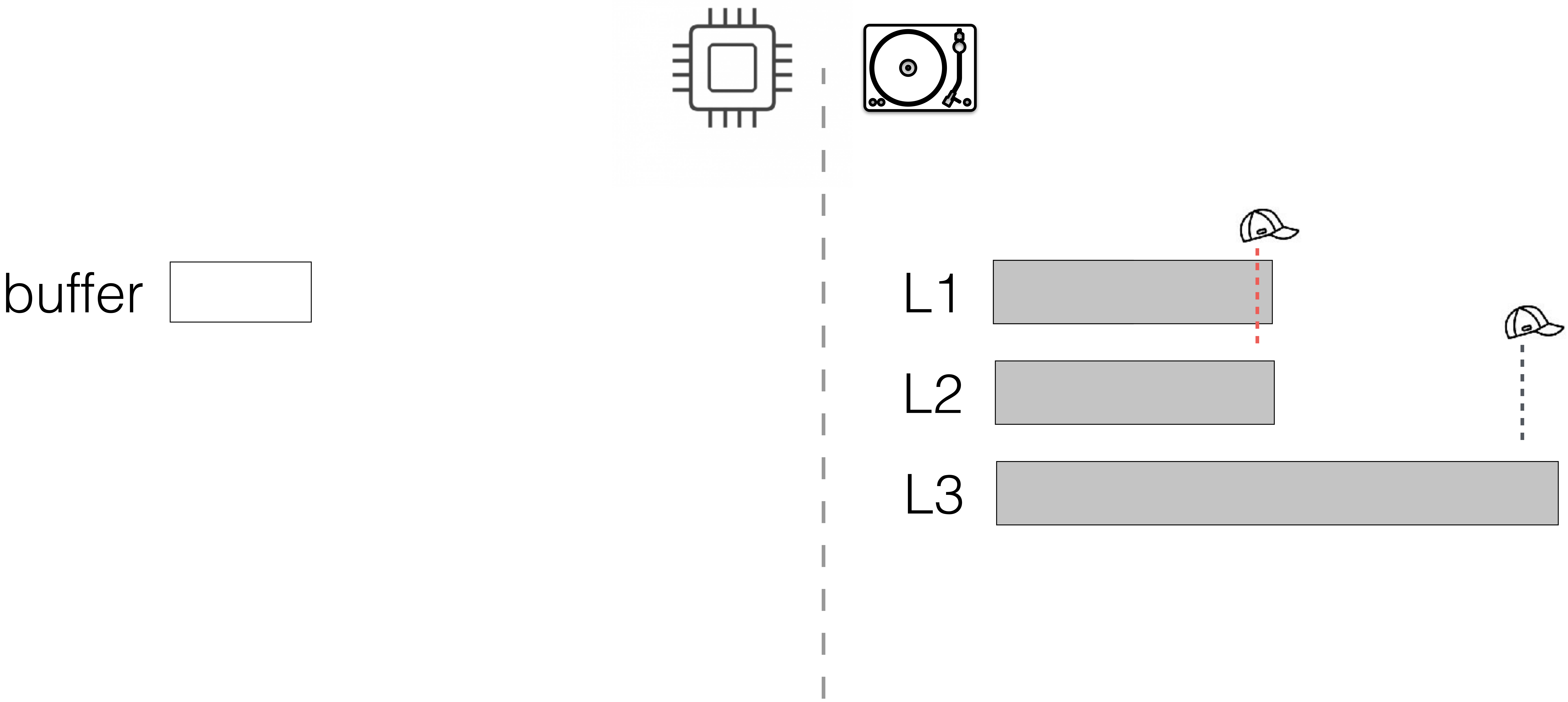
LSM Basics



LSM Basics



LSM Basics



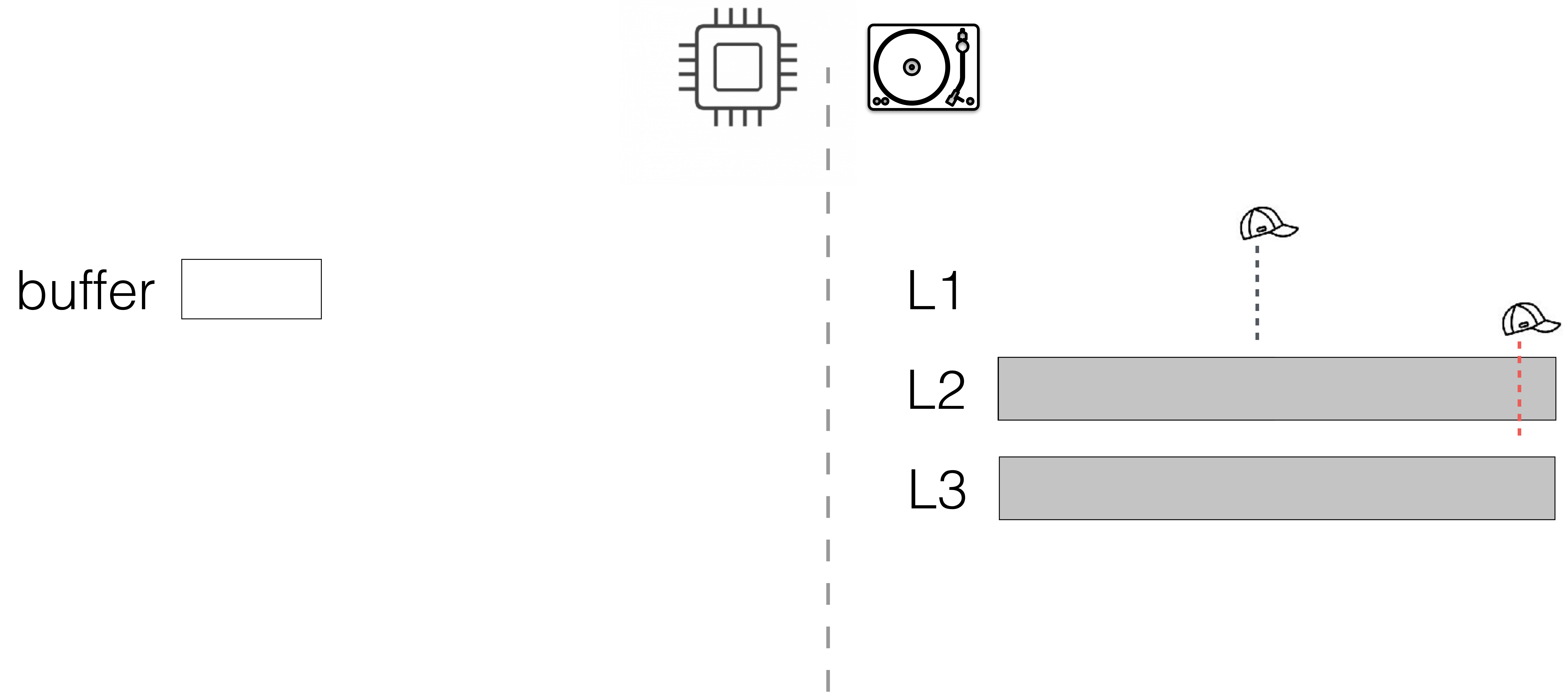
buffer

L1

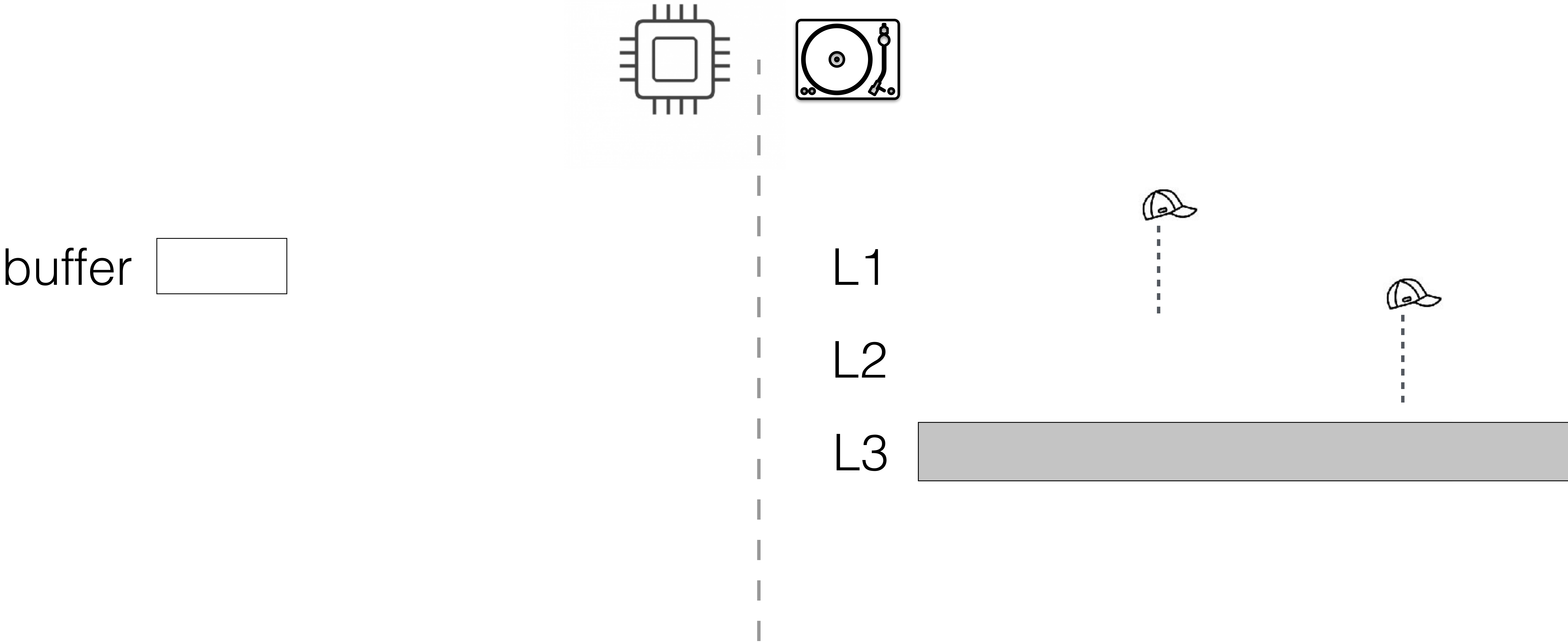
L2

L3

LSM Basics



LSM Basics



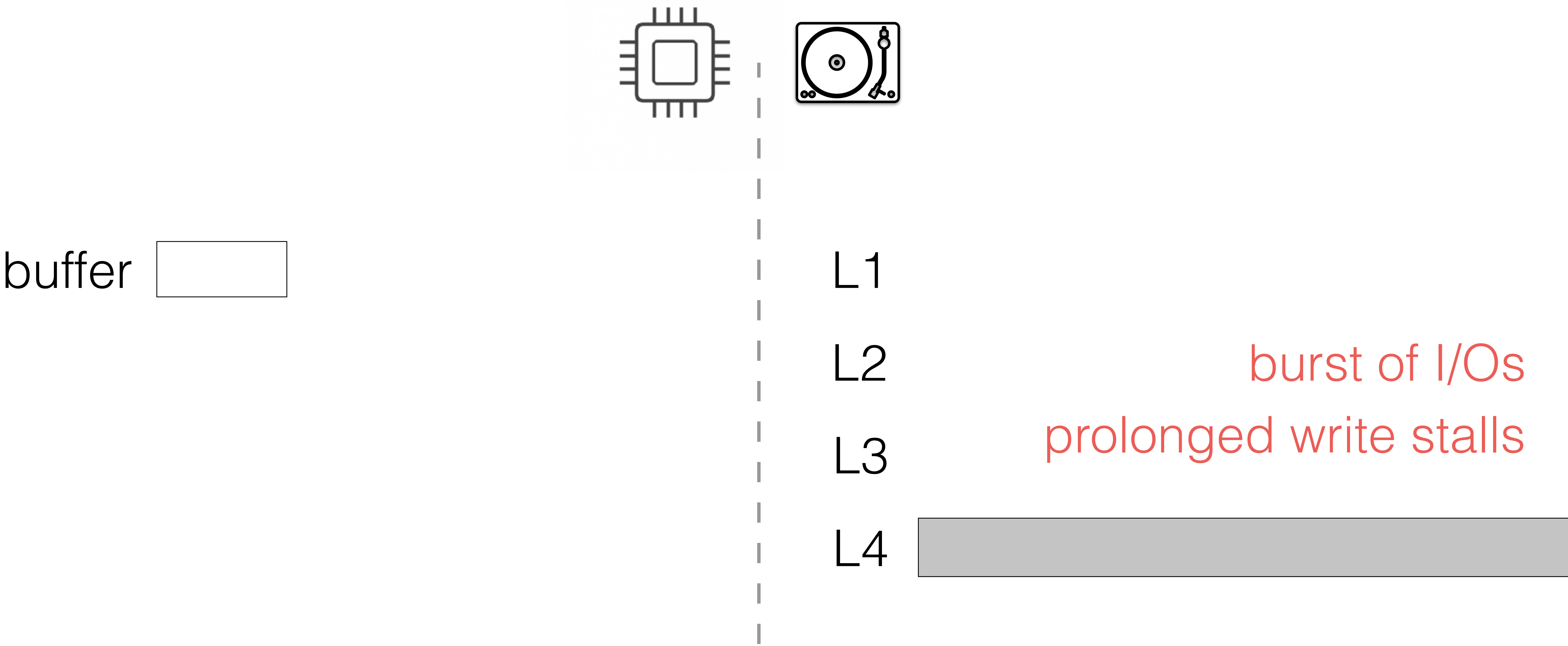
buffer

L1

L2

L3

LSM Basics



buffer

L1

L2

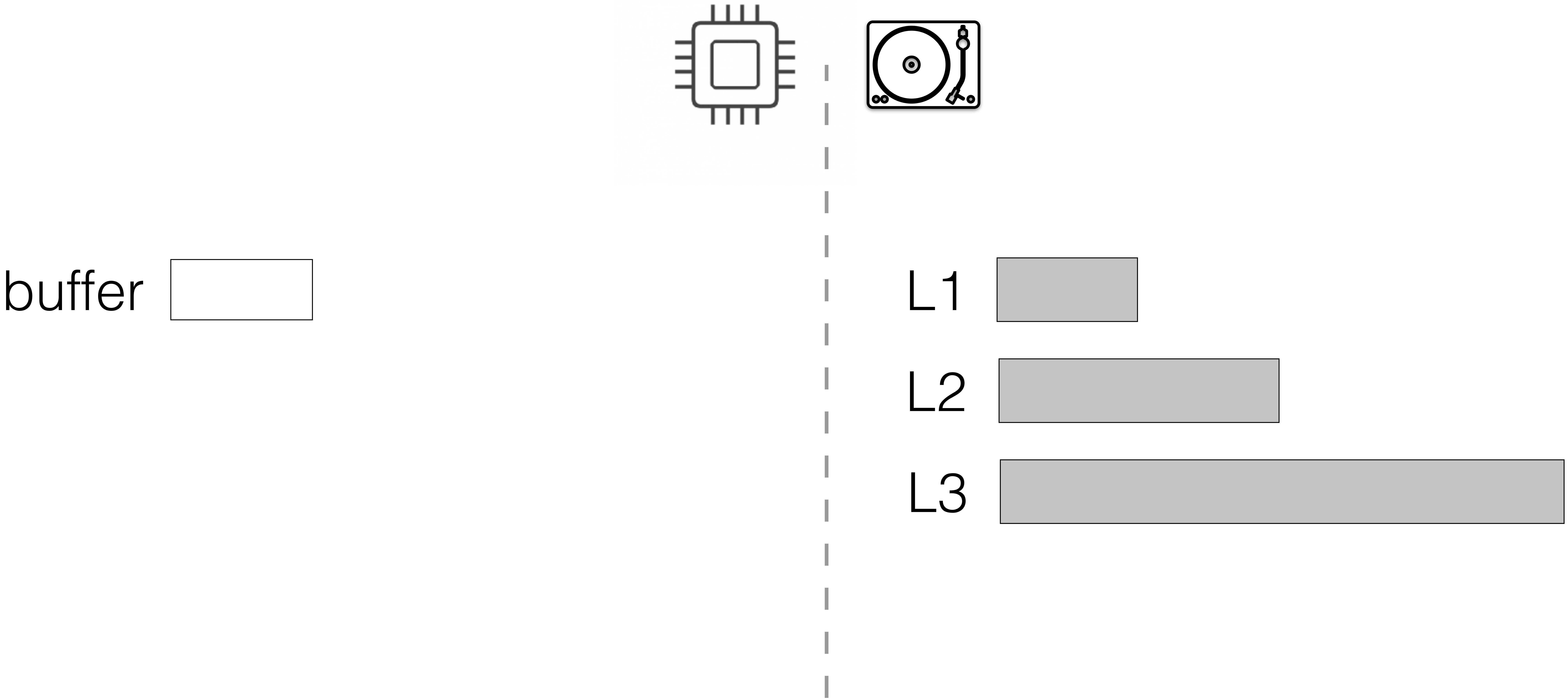
L3

L4

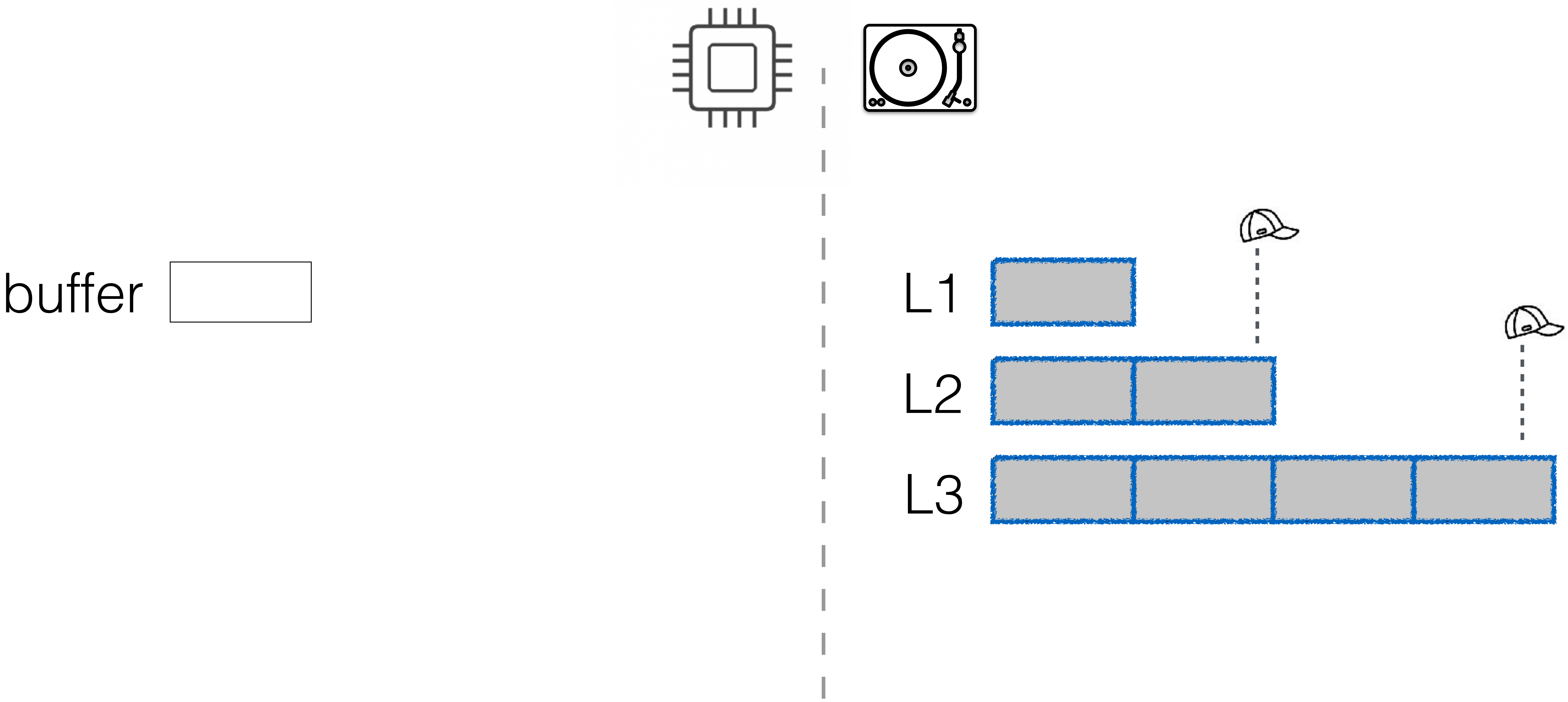
burst of I/Os

prolonged write stalls

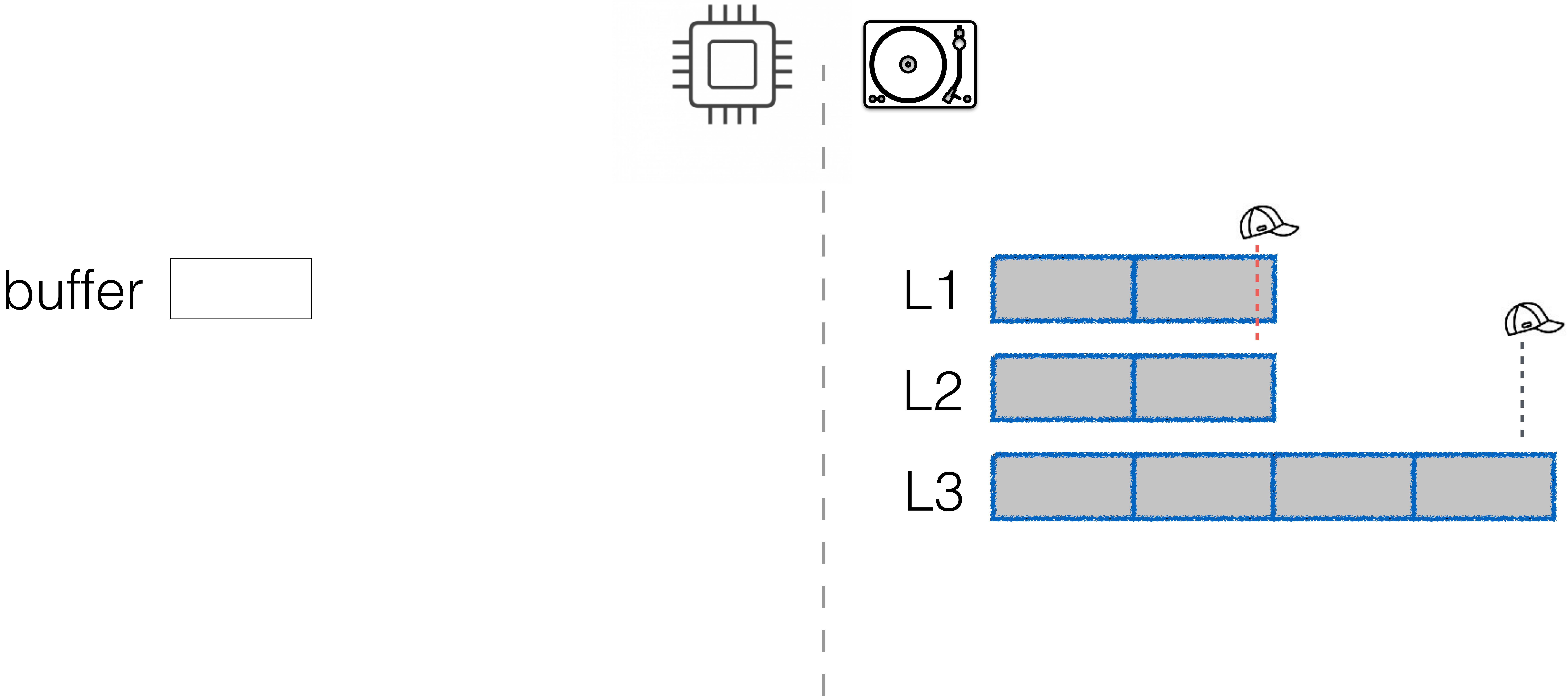
Partial Compaction



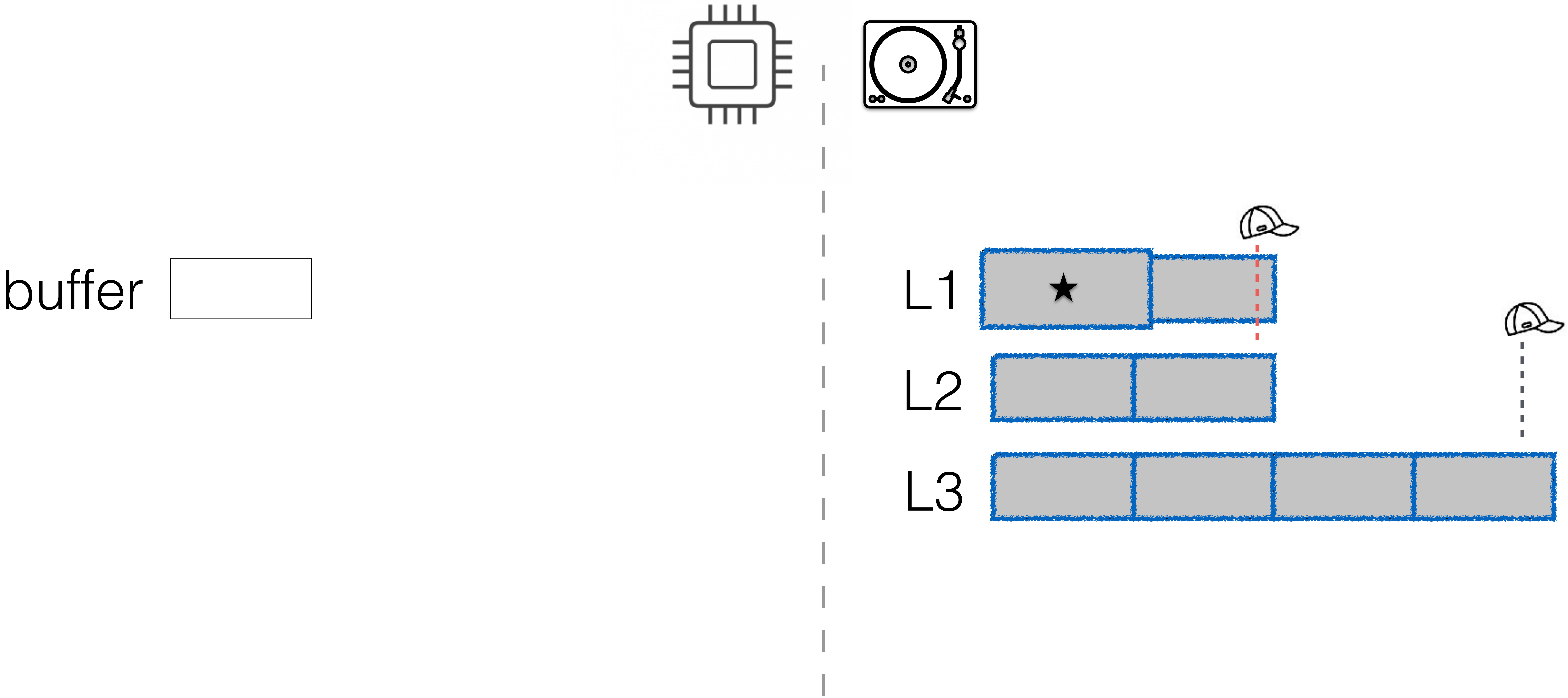
Partial Compaction



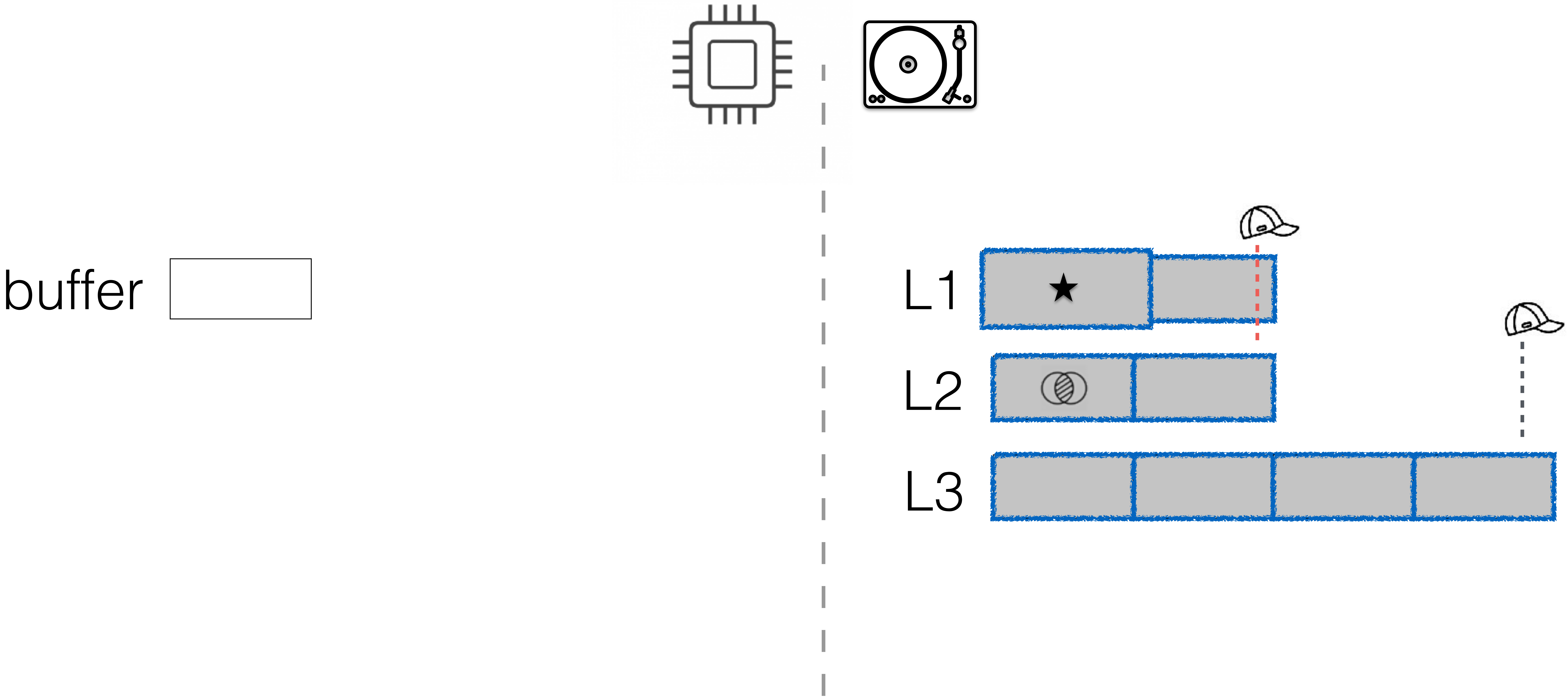
Partial Compaction



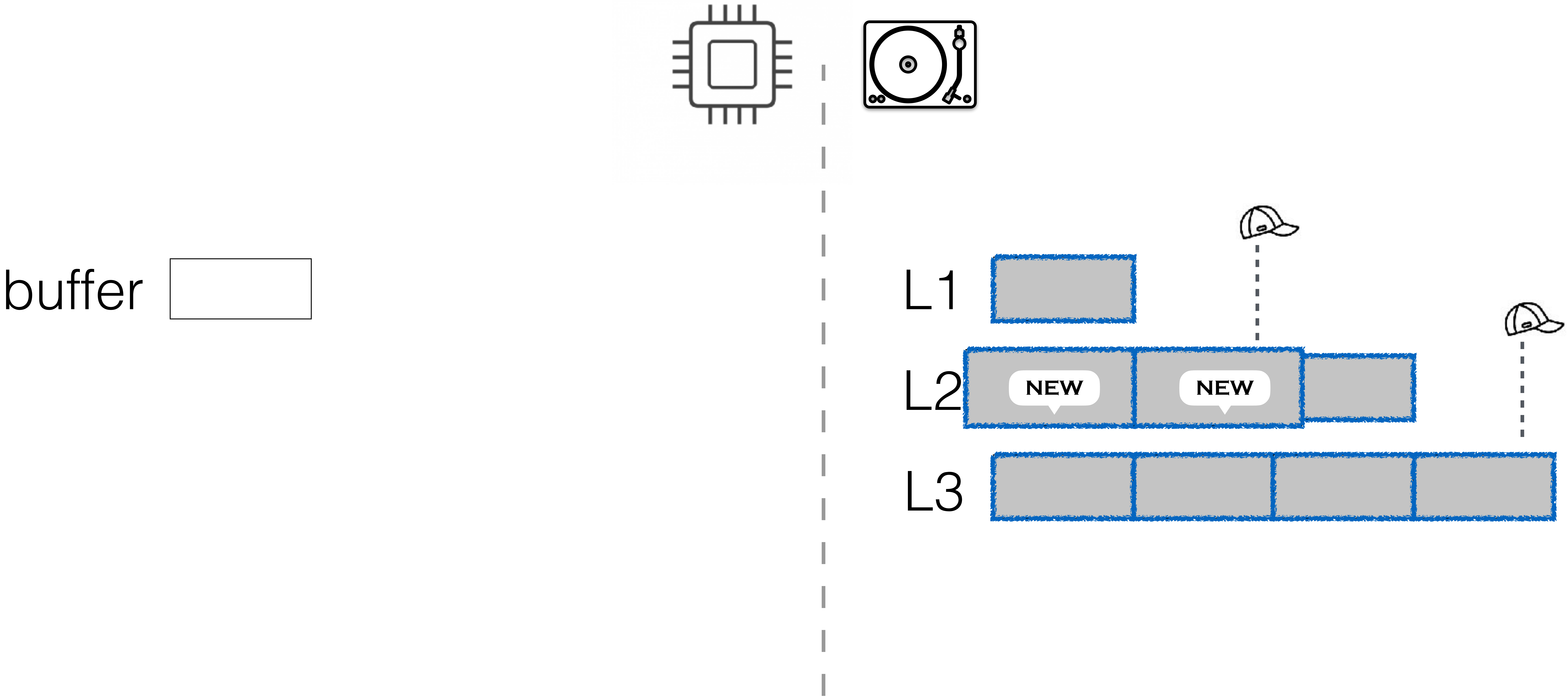
Partial Compaction



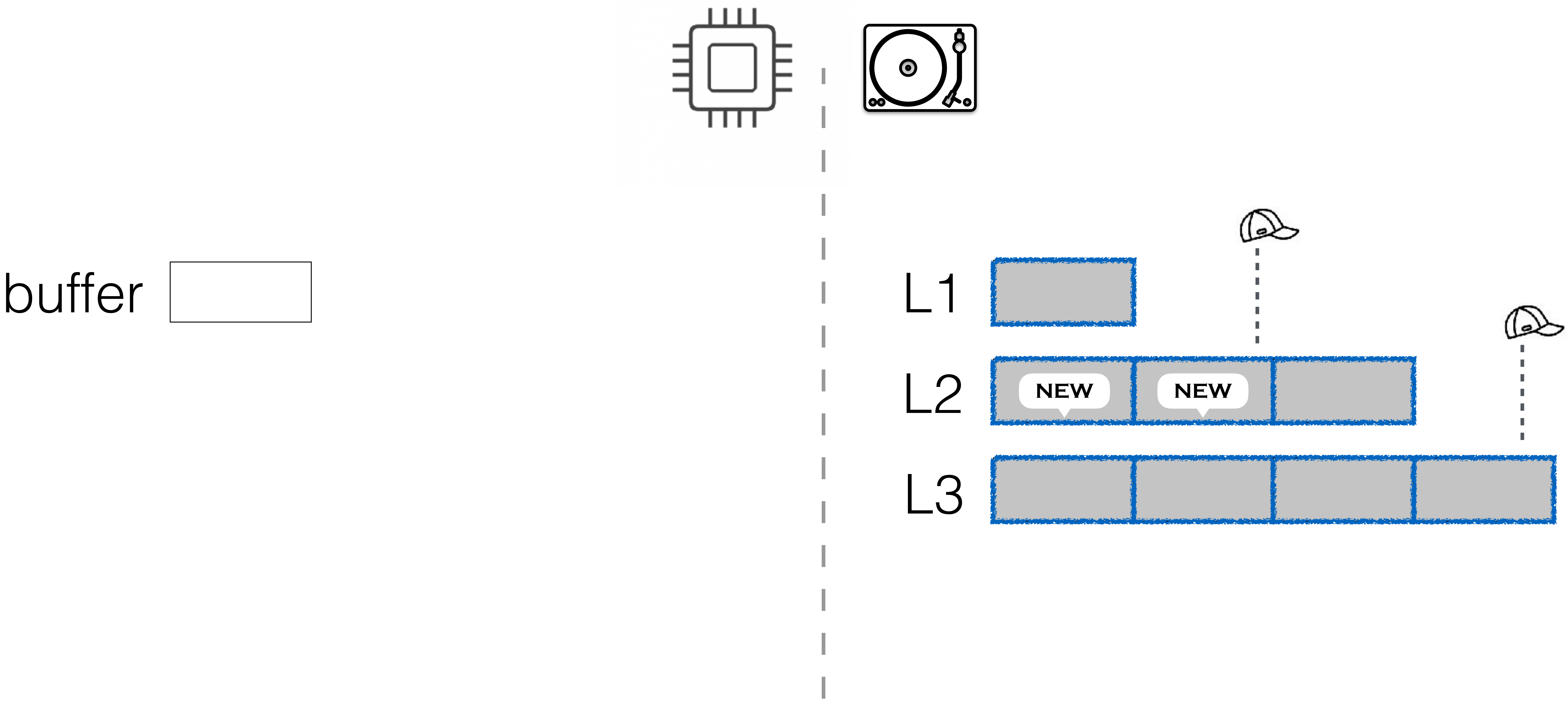
Partial Compaction



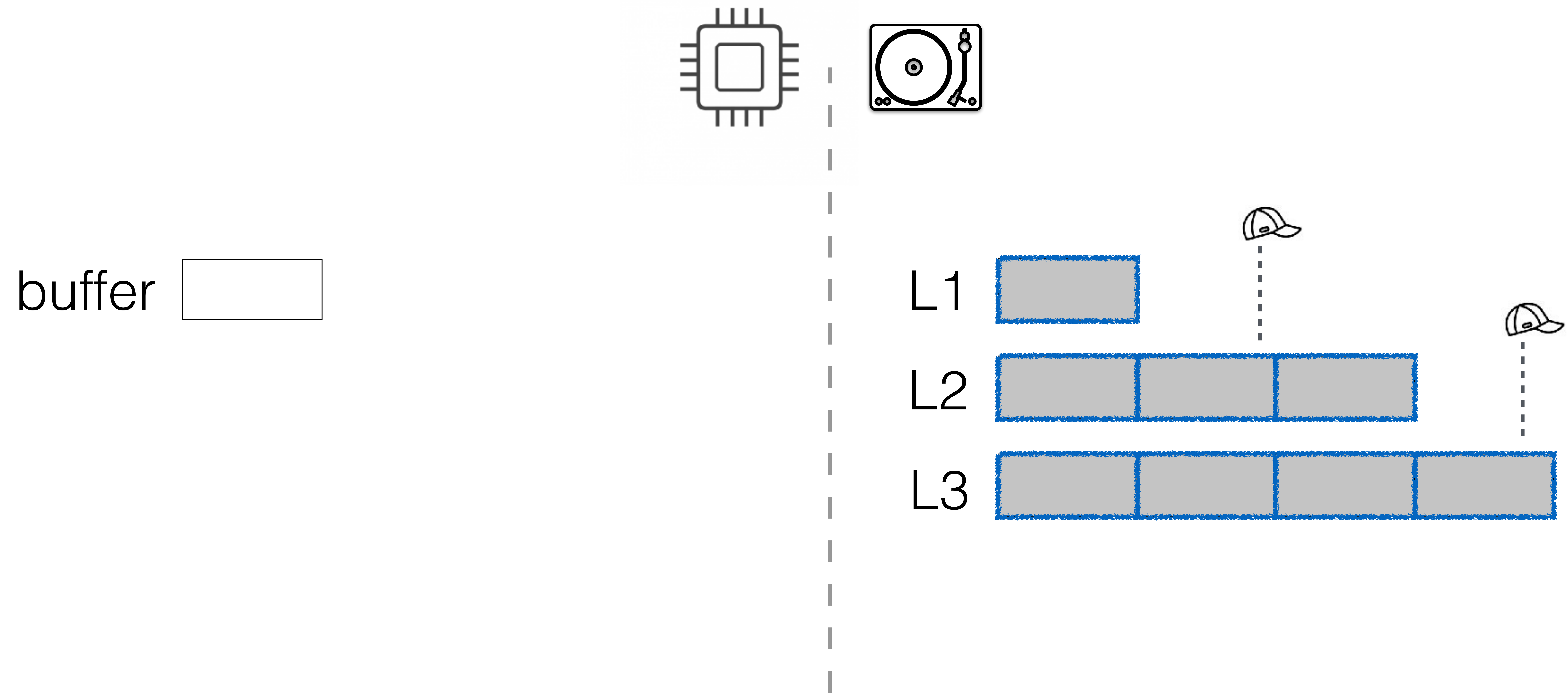
Partial Compaction



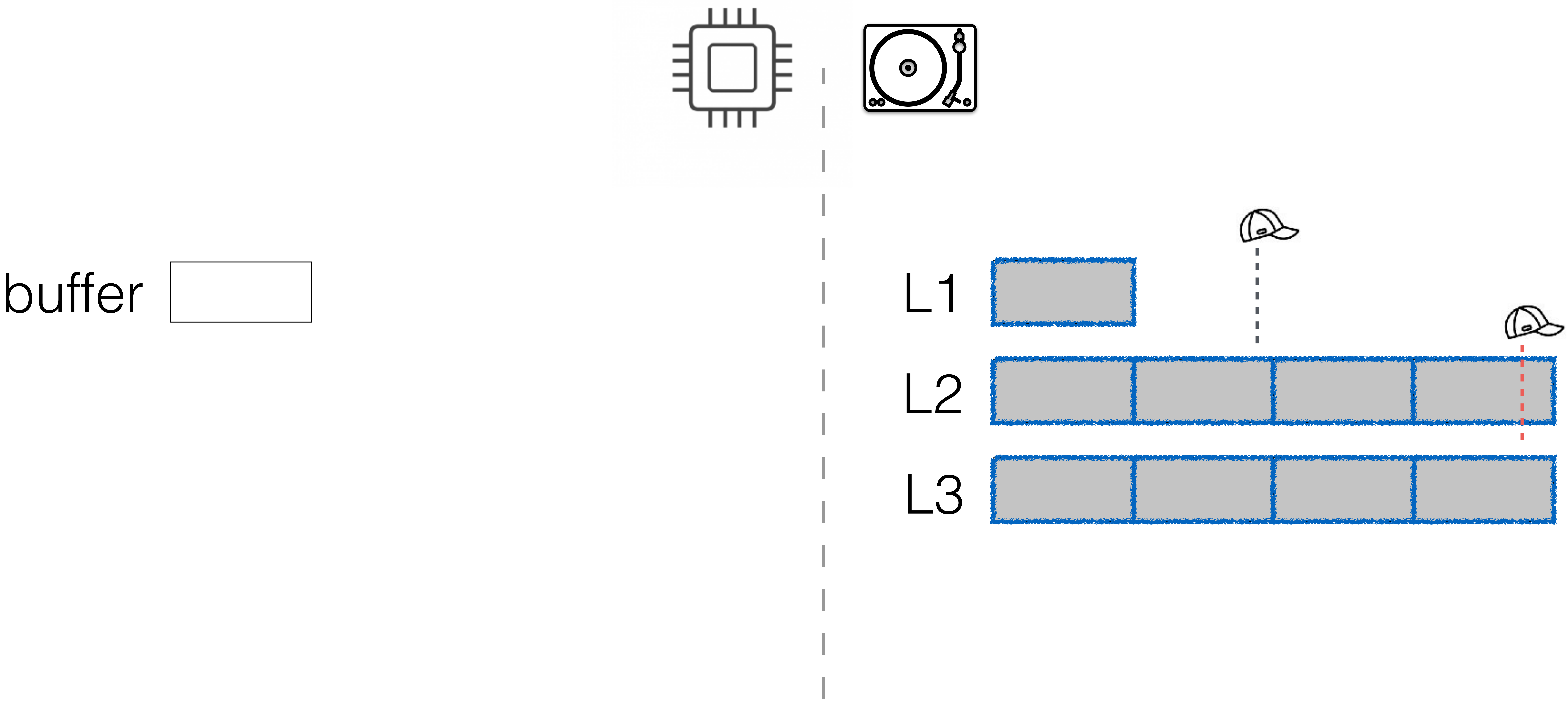
Partial Compaction



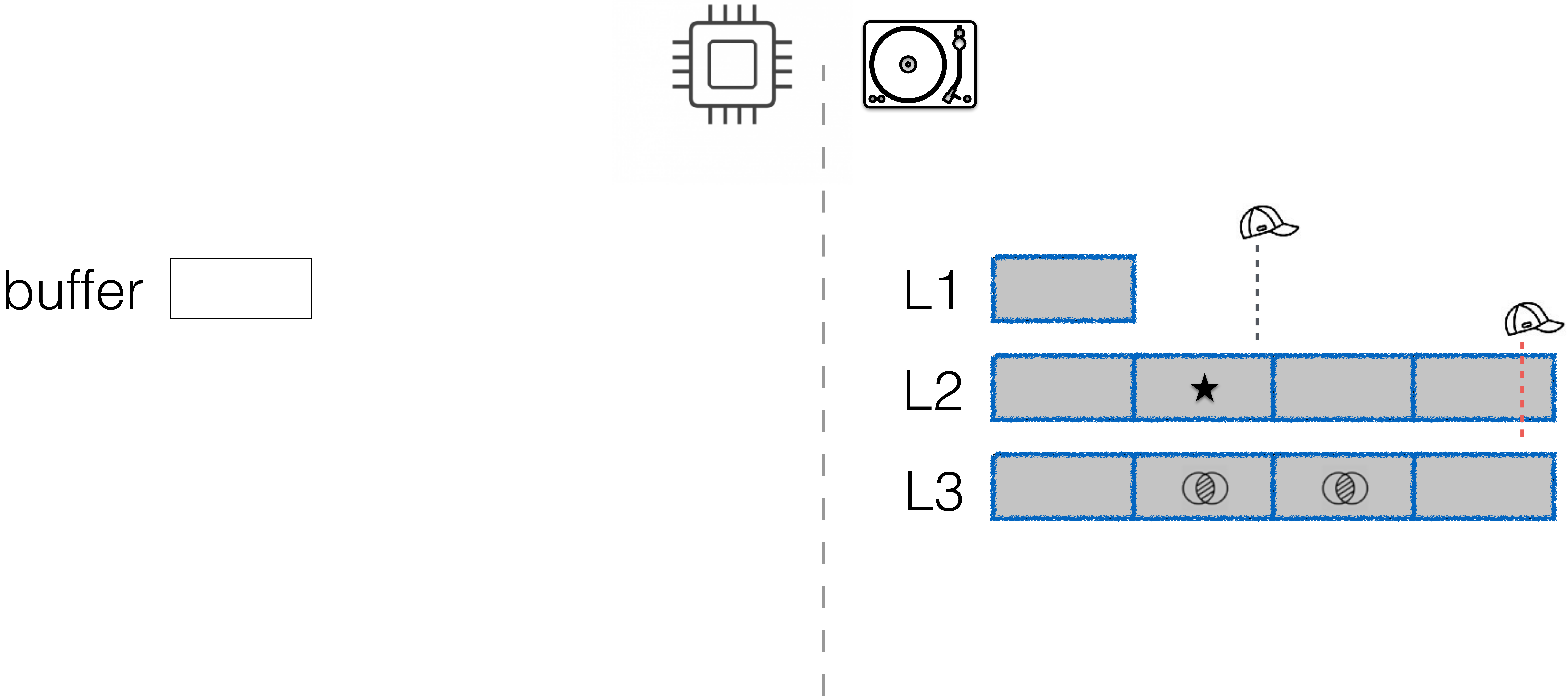
Partial Compaction



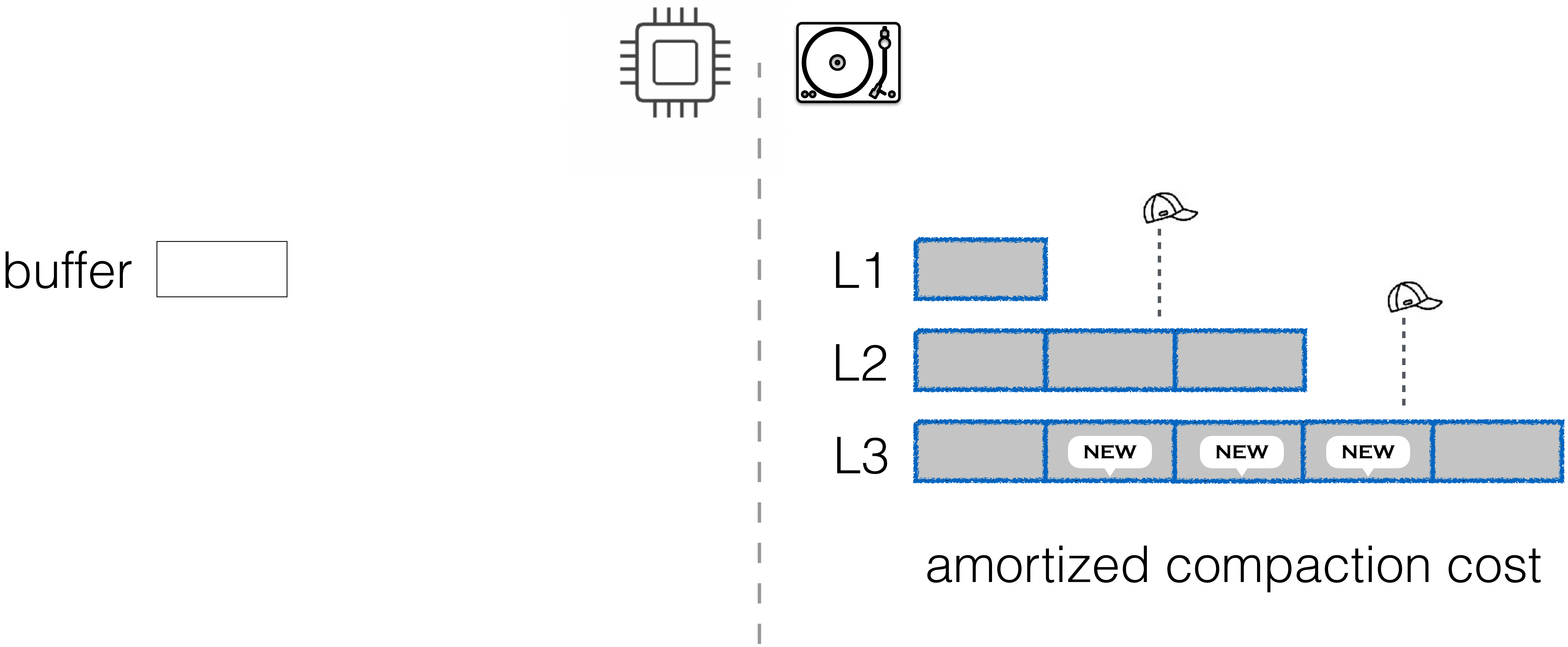
Partial Compaction



Partial Compaction

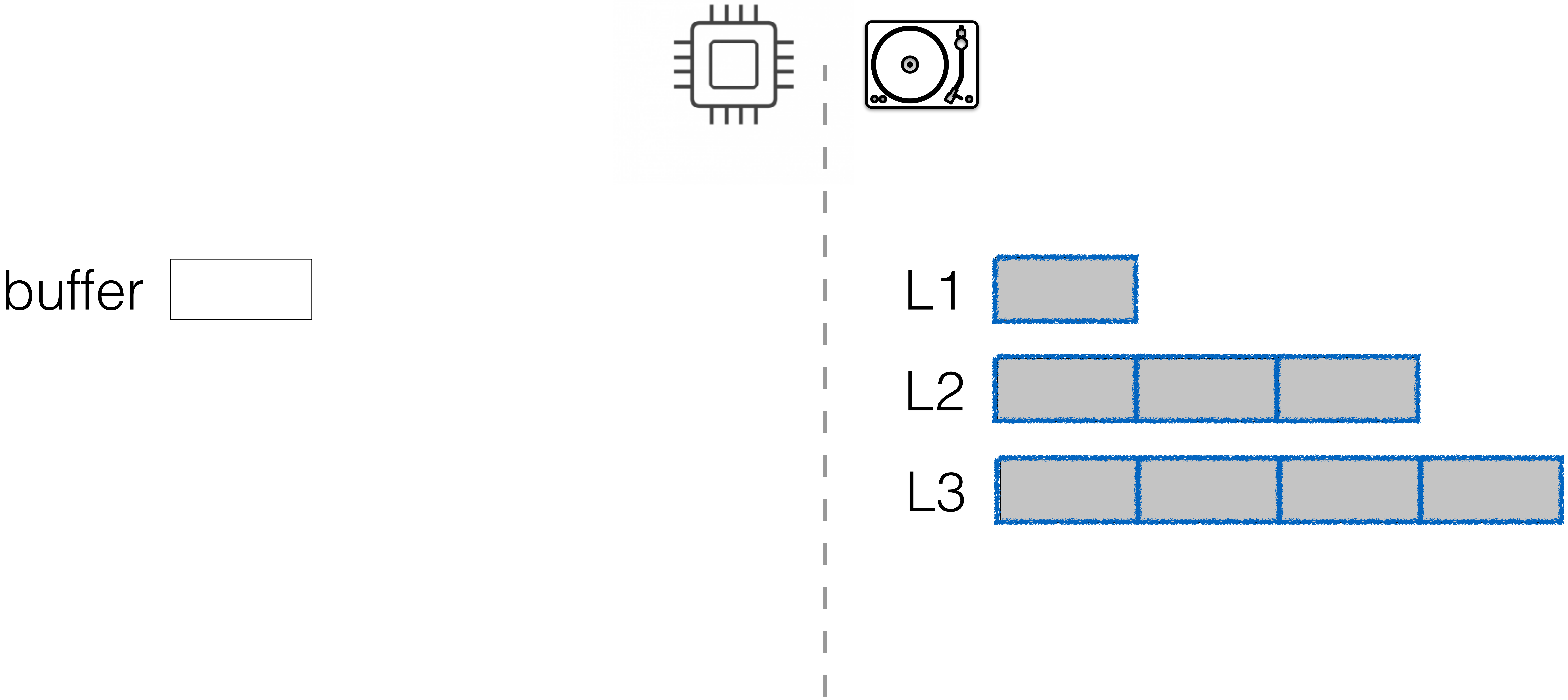


Partial Compaction

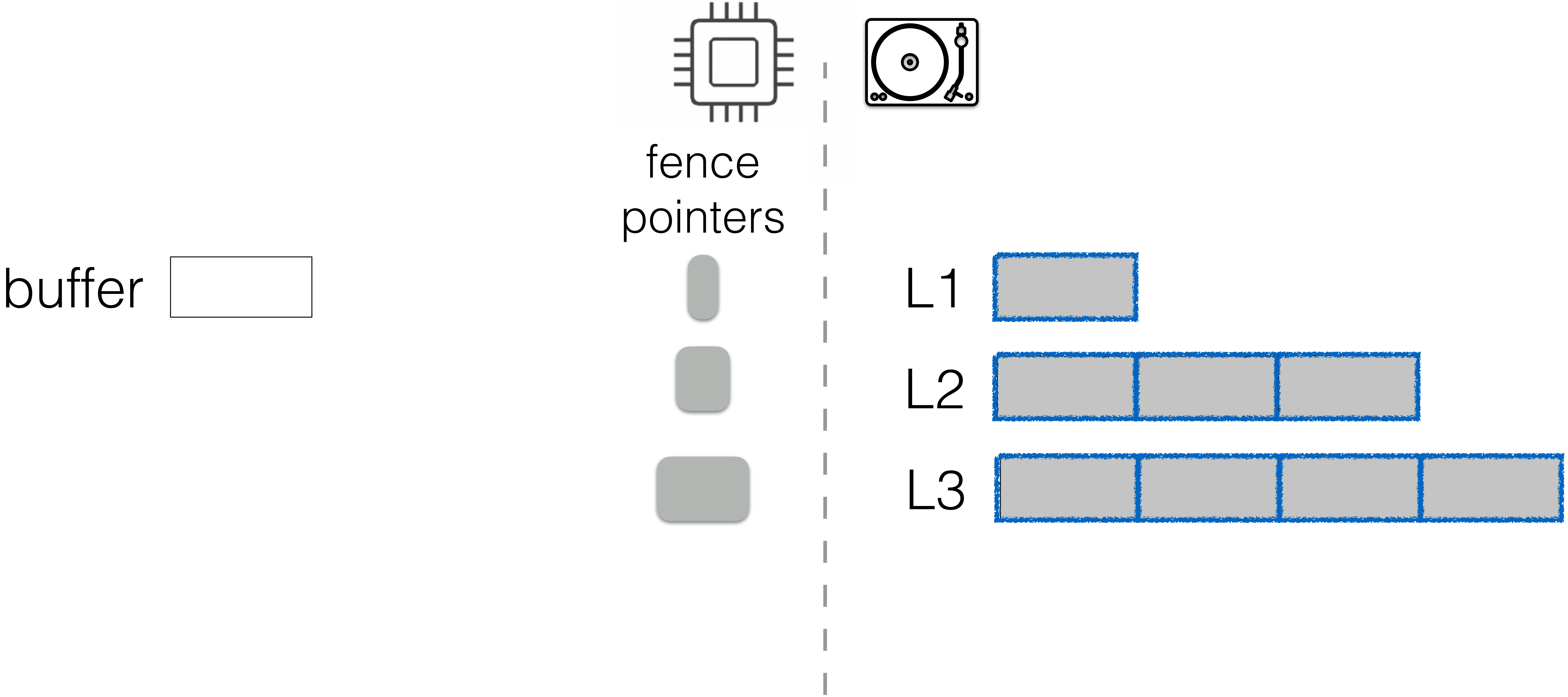


amortized compaction cost

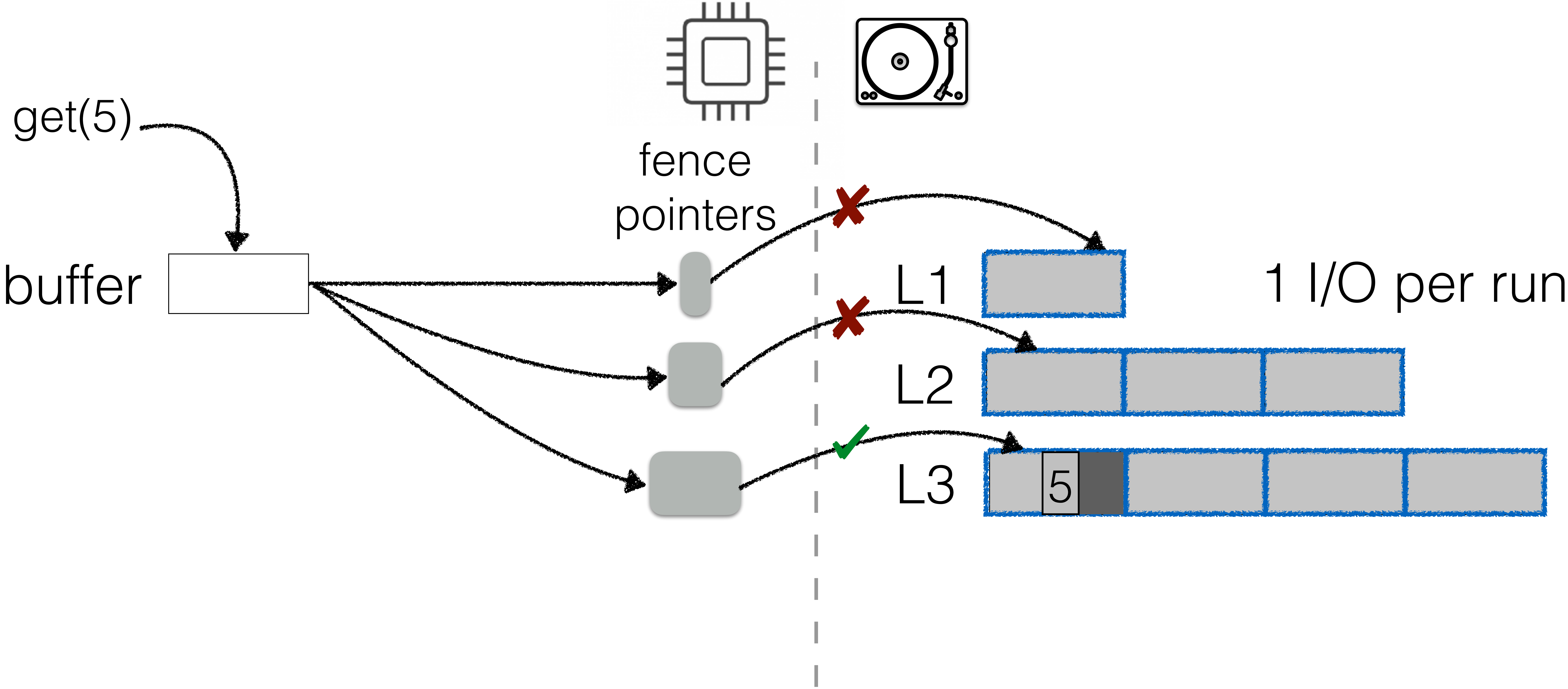
Lookups



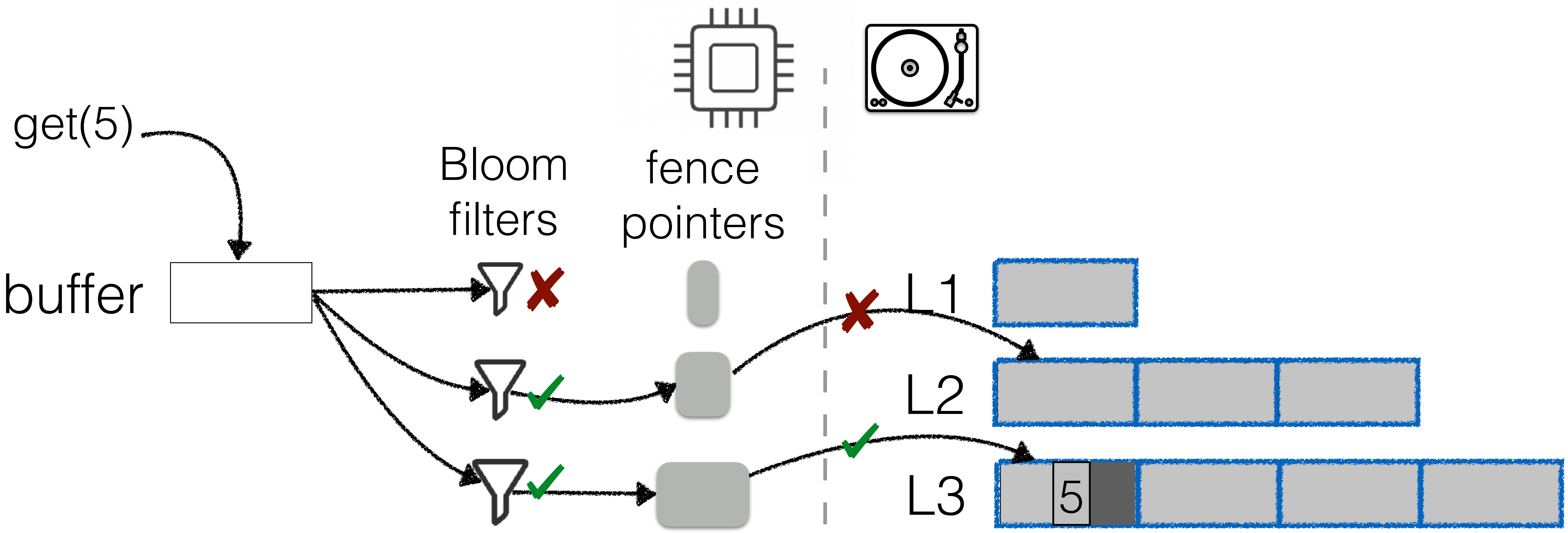
Lookups



Lookups



Lookups



fewer disk I/Os

Outline

Part 1: **LSM Basics**

Part 2: **The Problem: Deletes in LSMs**

Part 3: **Lethe: Enabling Efficient Deletes**

Outline

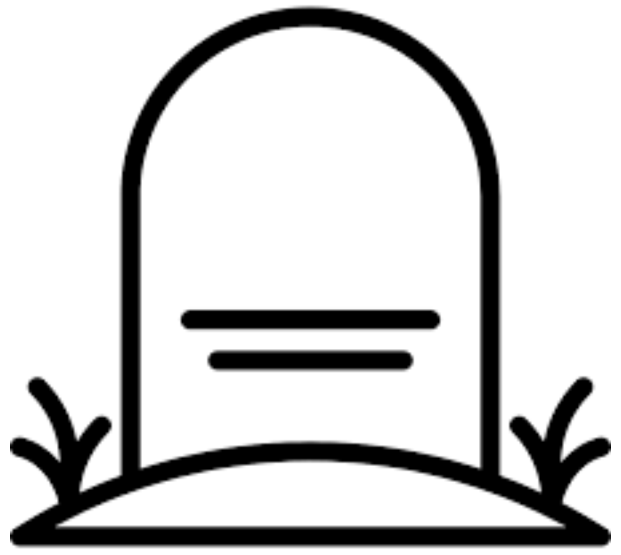
Part 1: **LSM Basics**

Part 2: **The Problem: Deletes in LSMs**

Part 3: **Lethe: Enabling Efficient Deletes**

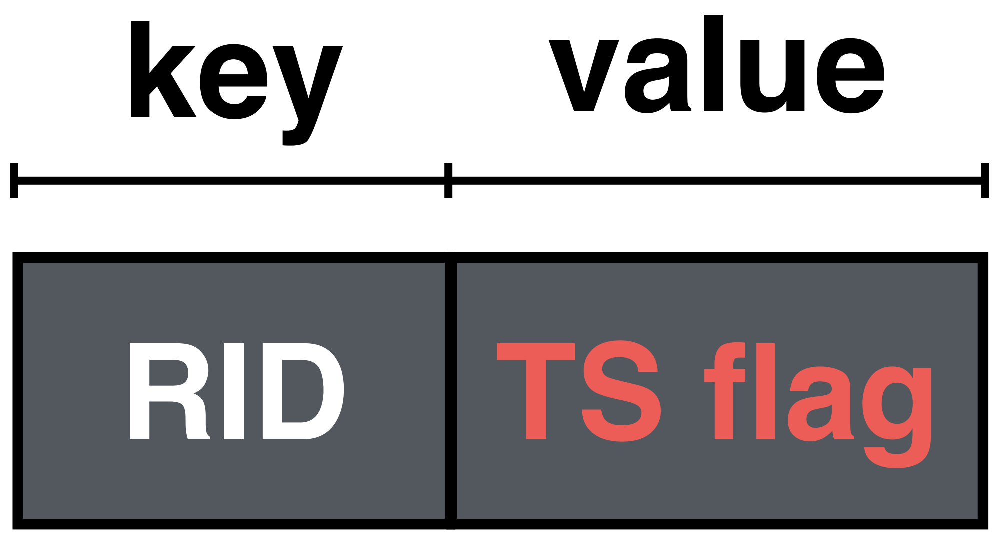
Deletes in LSMs

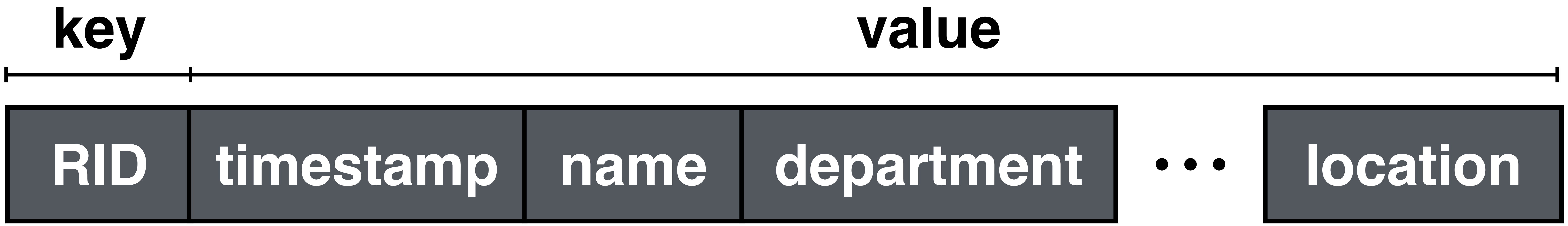
delete



Deletes in LSMs

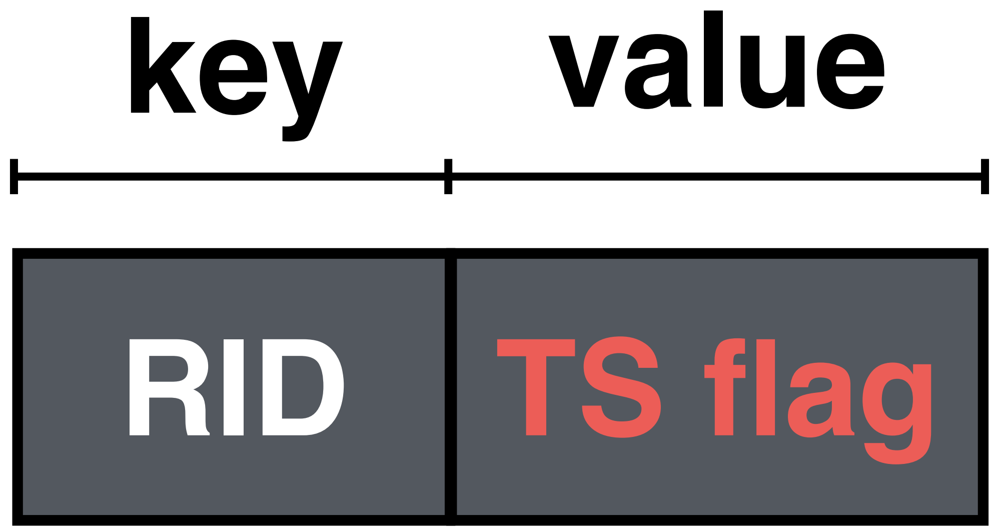
delete := insert tombstone





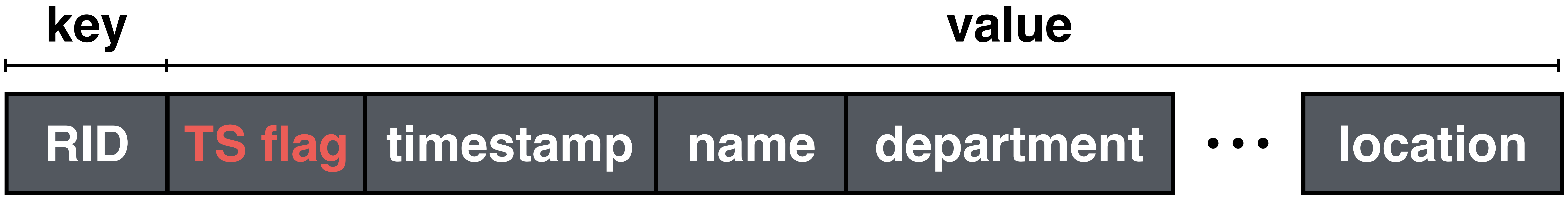
Deletes in LSMs

delete := insert tombstone



key value

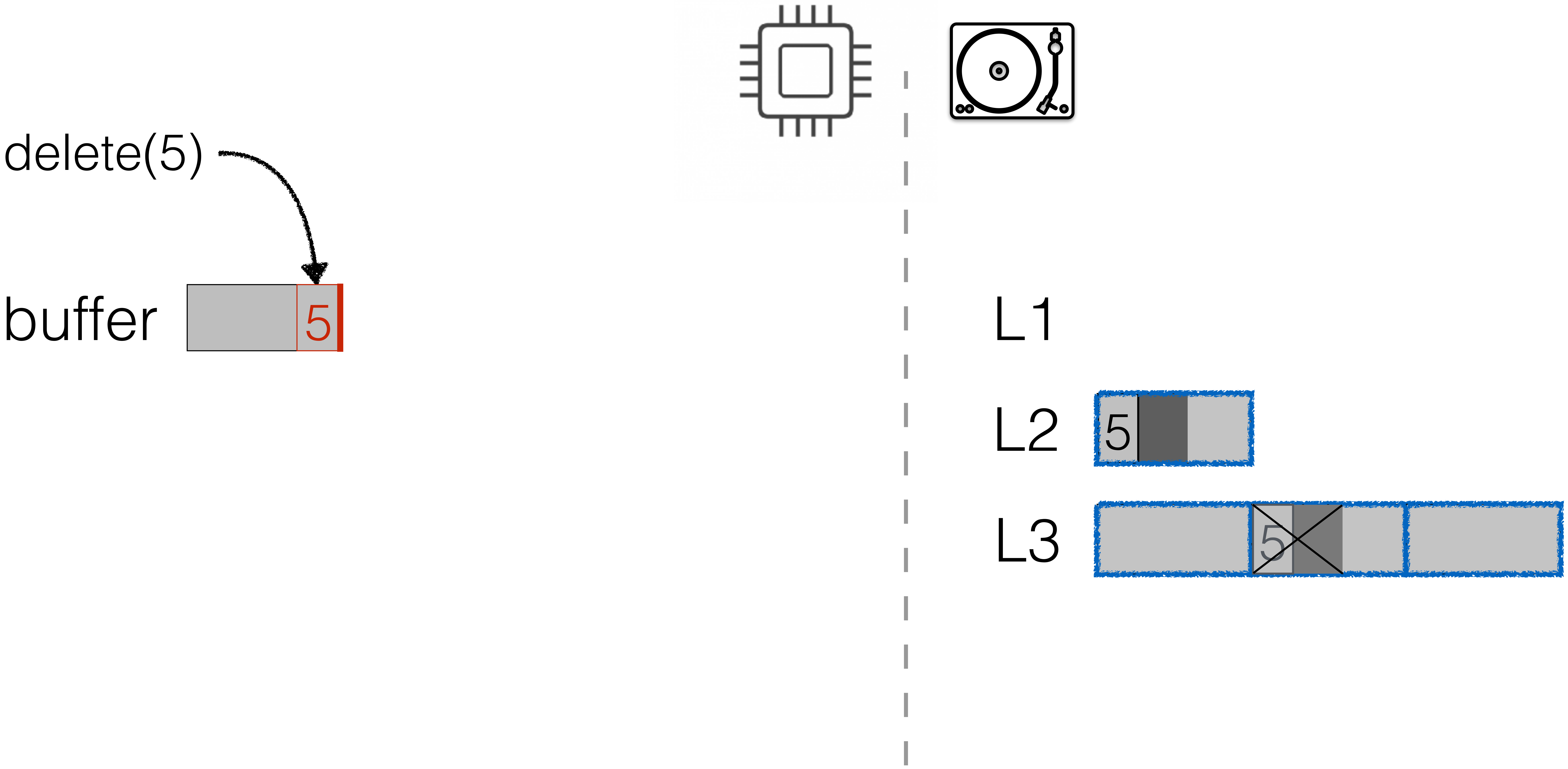
RID | TS flag



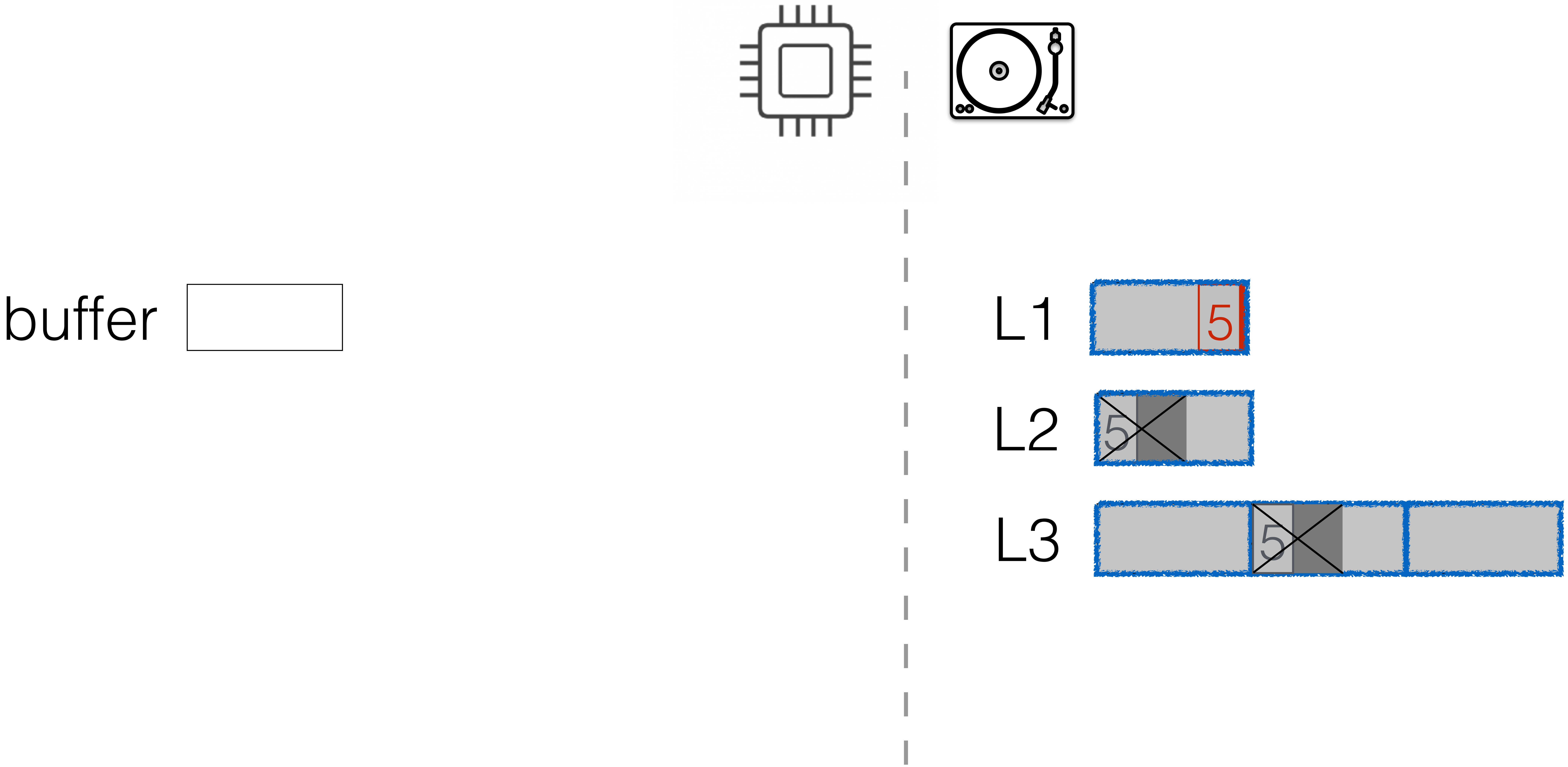
key value

RID | TS flag | timestamp | name | department | • • • | location

Deletes in LSMs



Deletes in LSMs



buffer

L1

5

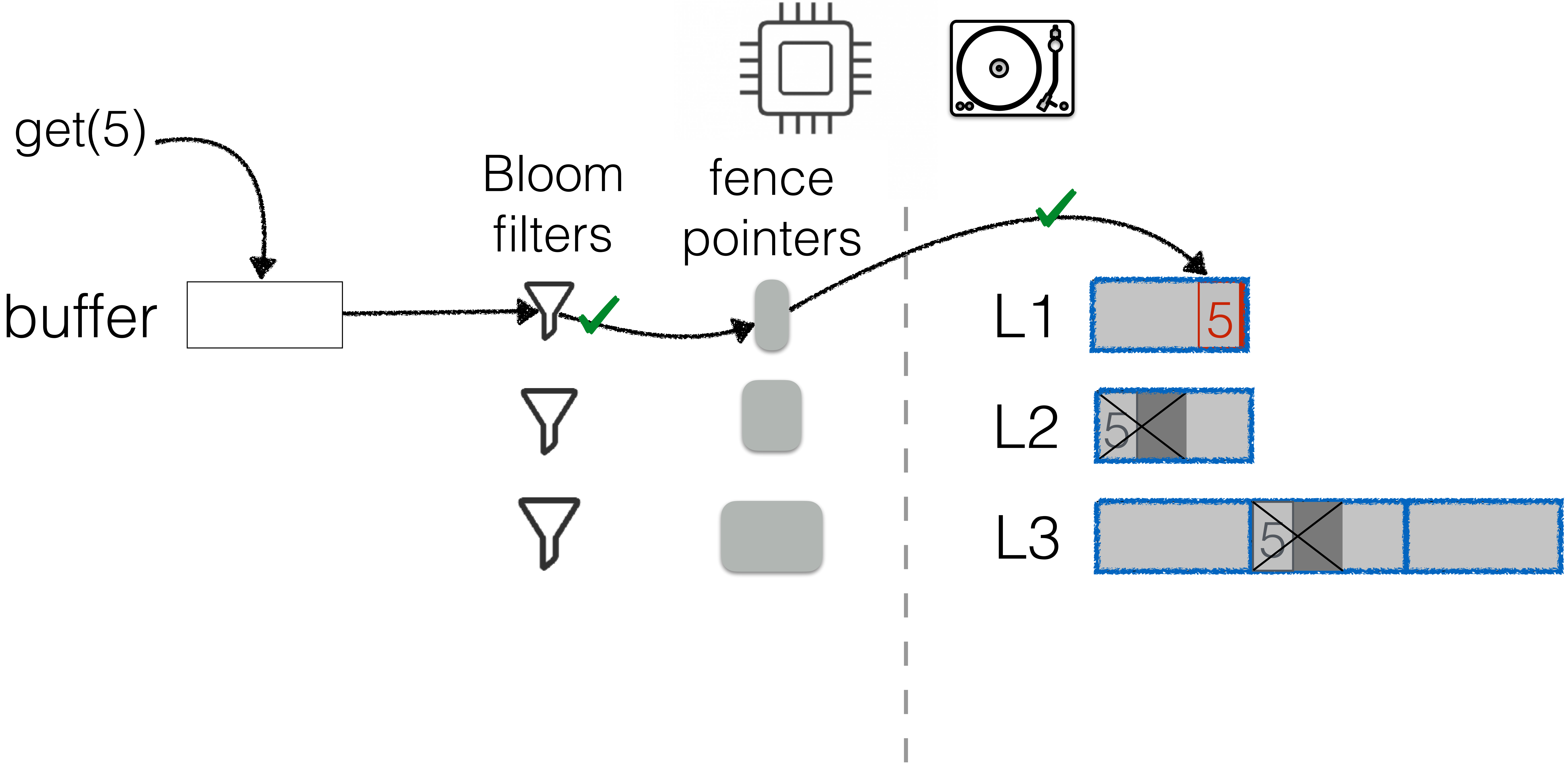
L2

5

L3

5

Deletes in LSMs

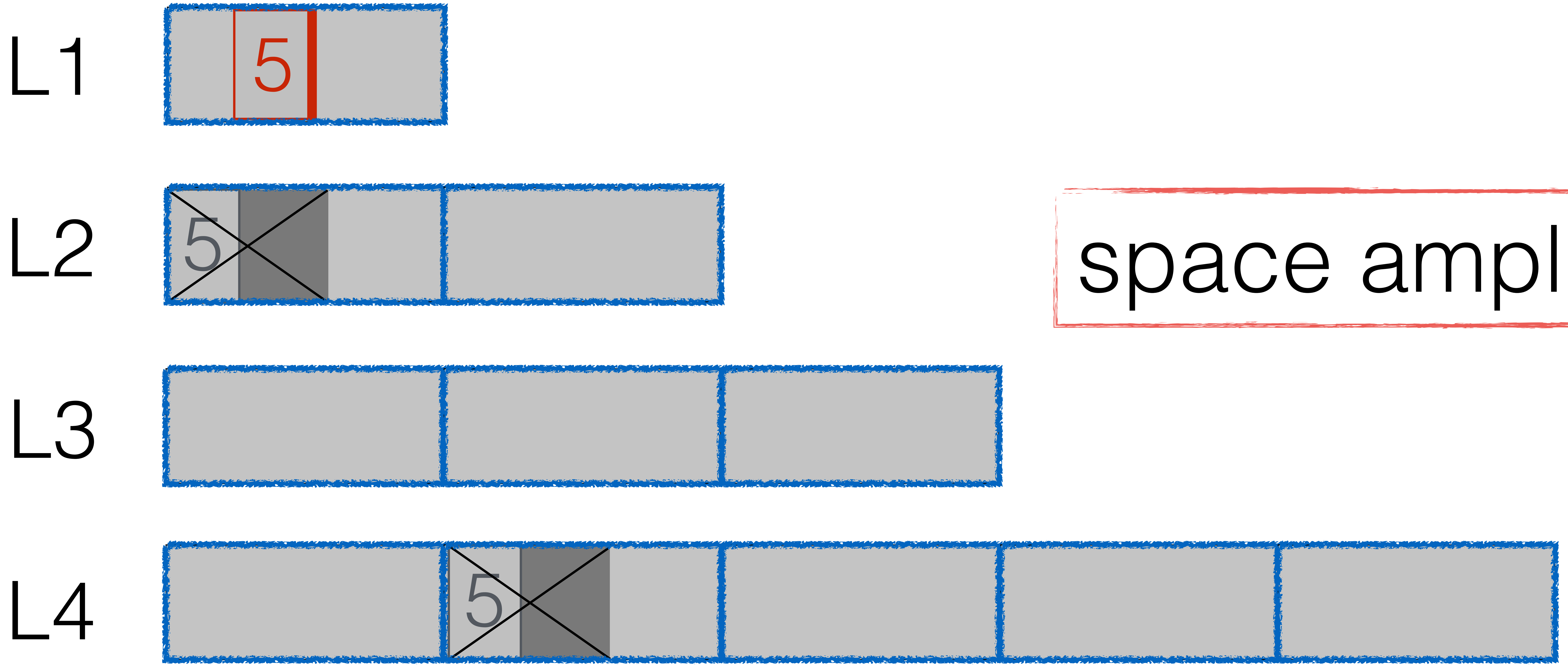


The **Problem**

The **Problem**

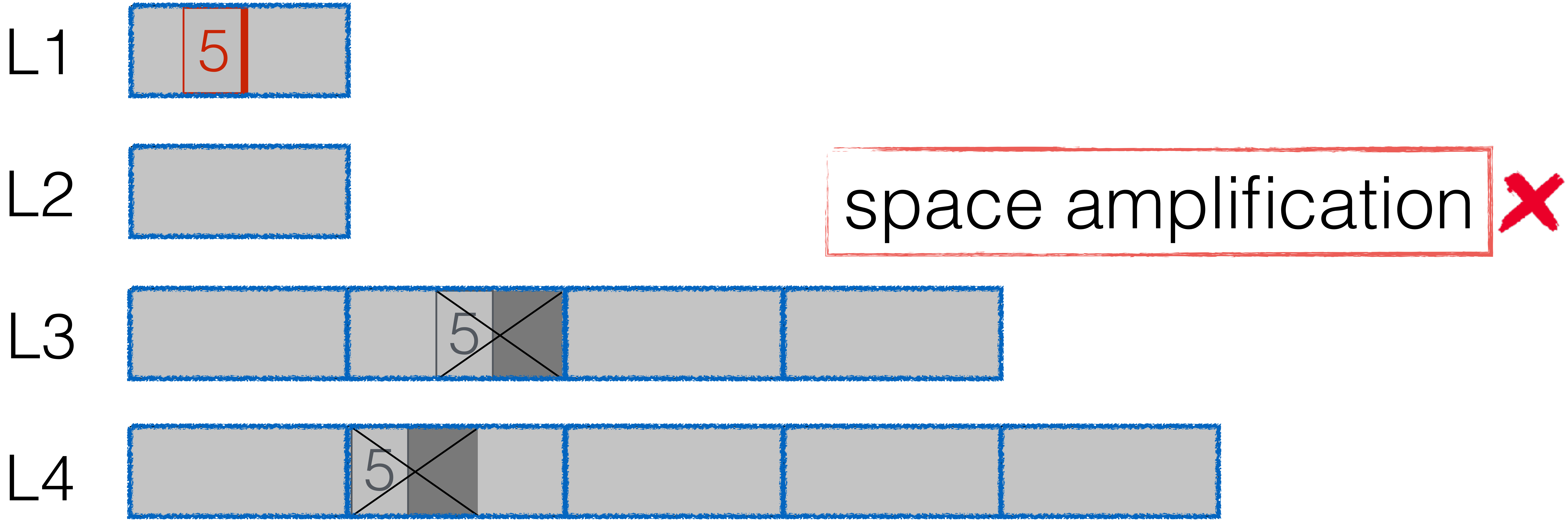
Performance Roadblock

Performance Roadblock

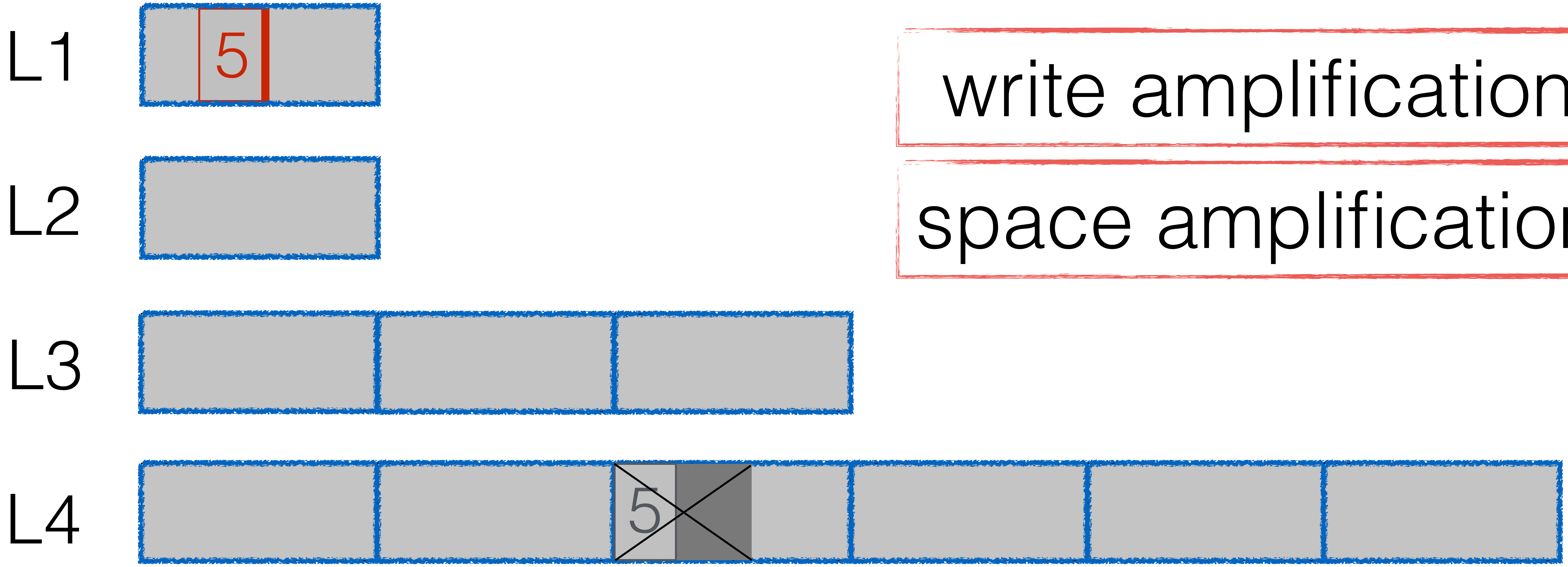


space amplification ✗

Performance Roadblock



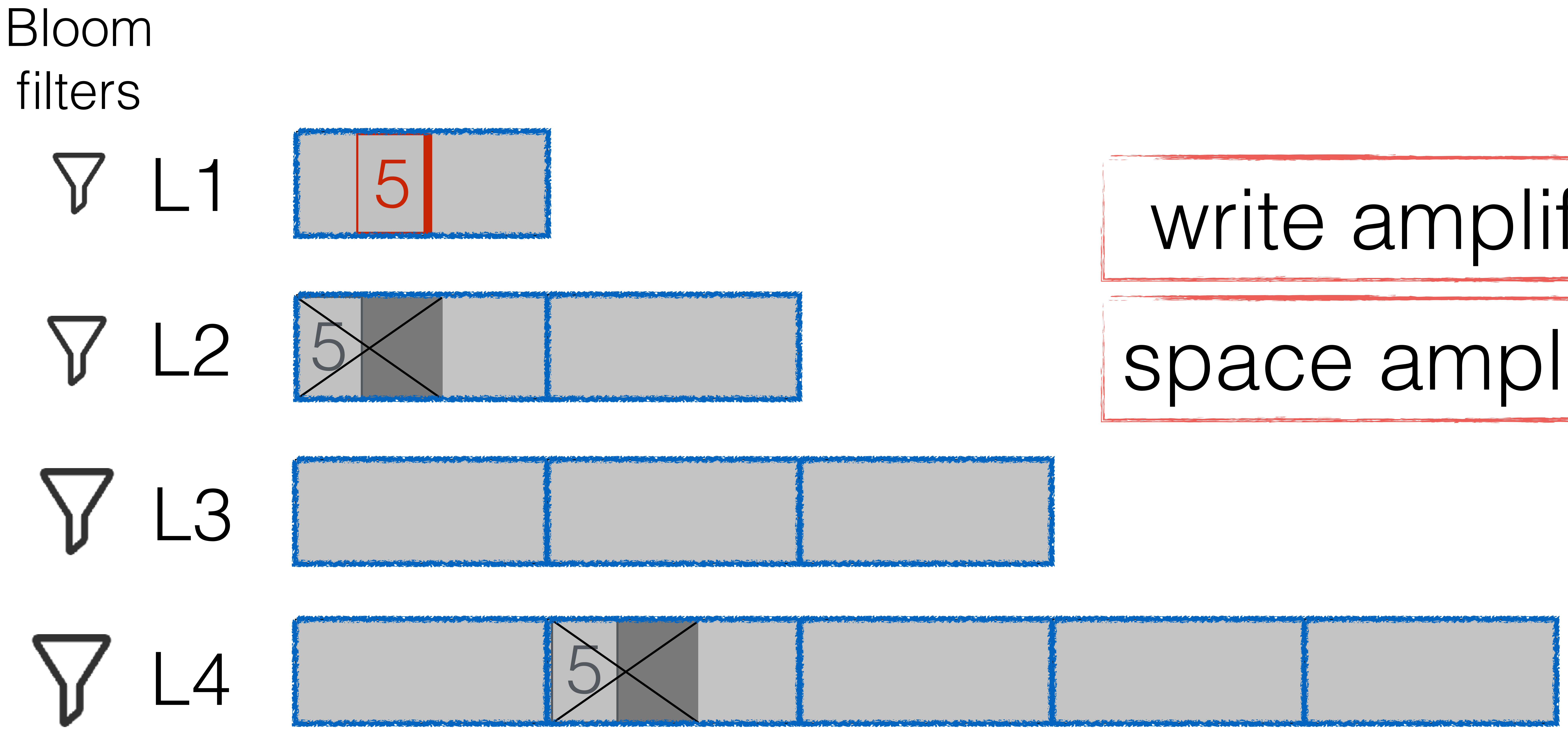
Performance Roadblock



write amplification ✗

space amplification ✗

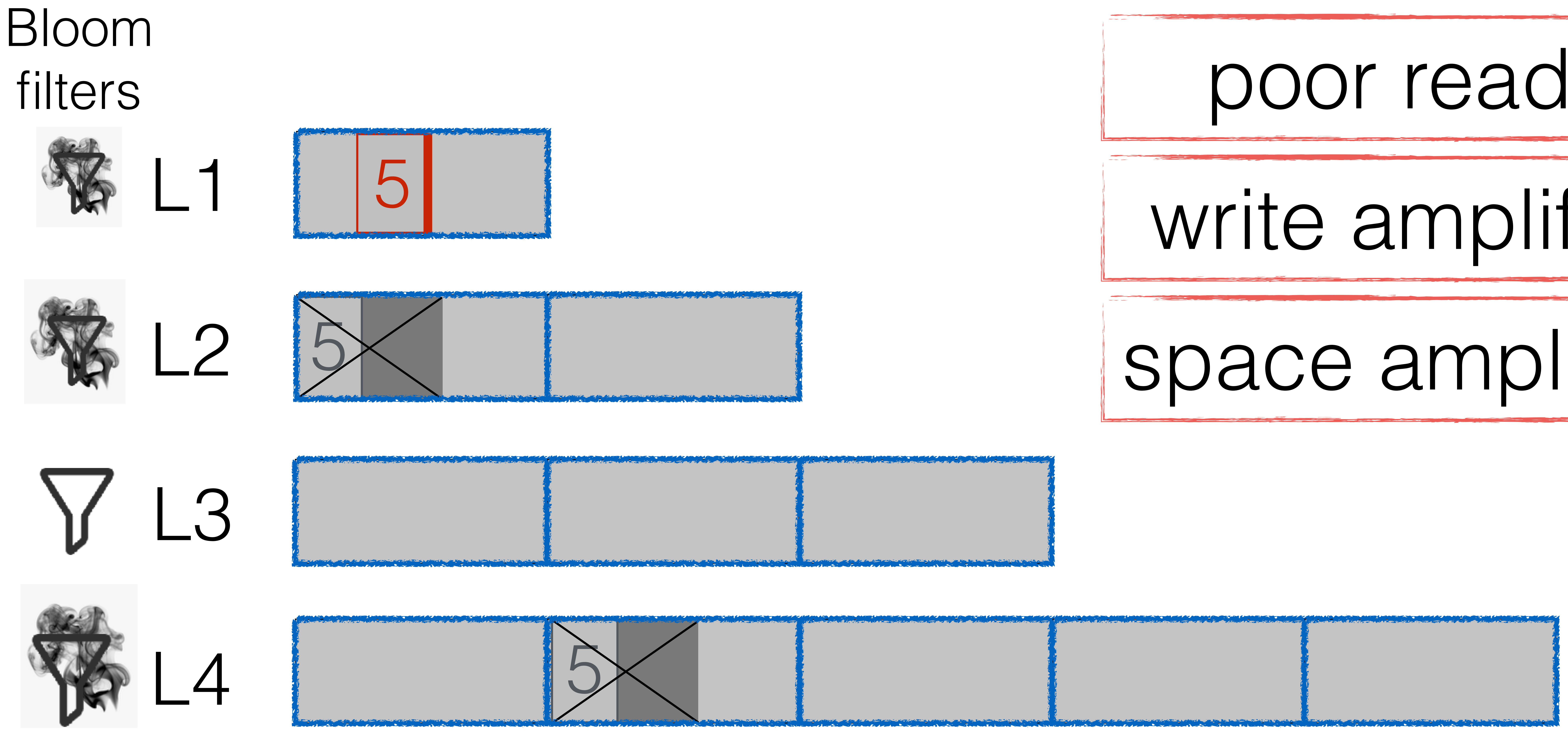
Performance Roadblock



write amplification ✗

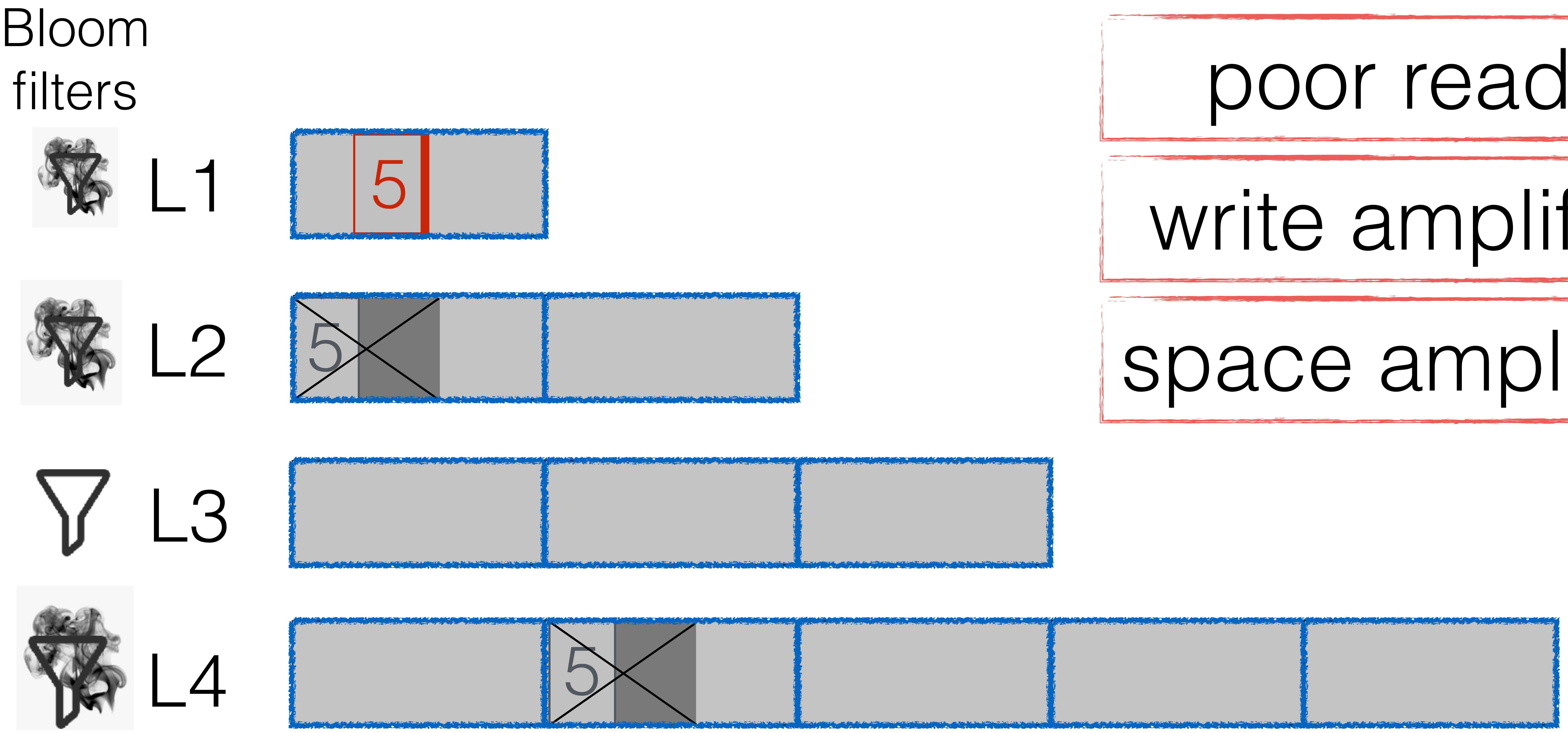
space amplification ✗

Performance Roadblock



- poor read perf. ✗
- write amplification ✗
- space amplification ✗

Performance Roadblock



- poor read perf. ✗
- write amplification ✗
- space amplification ✗

The **Problem**

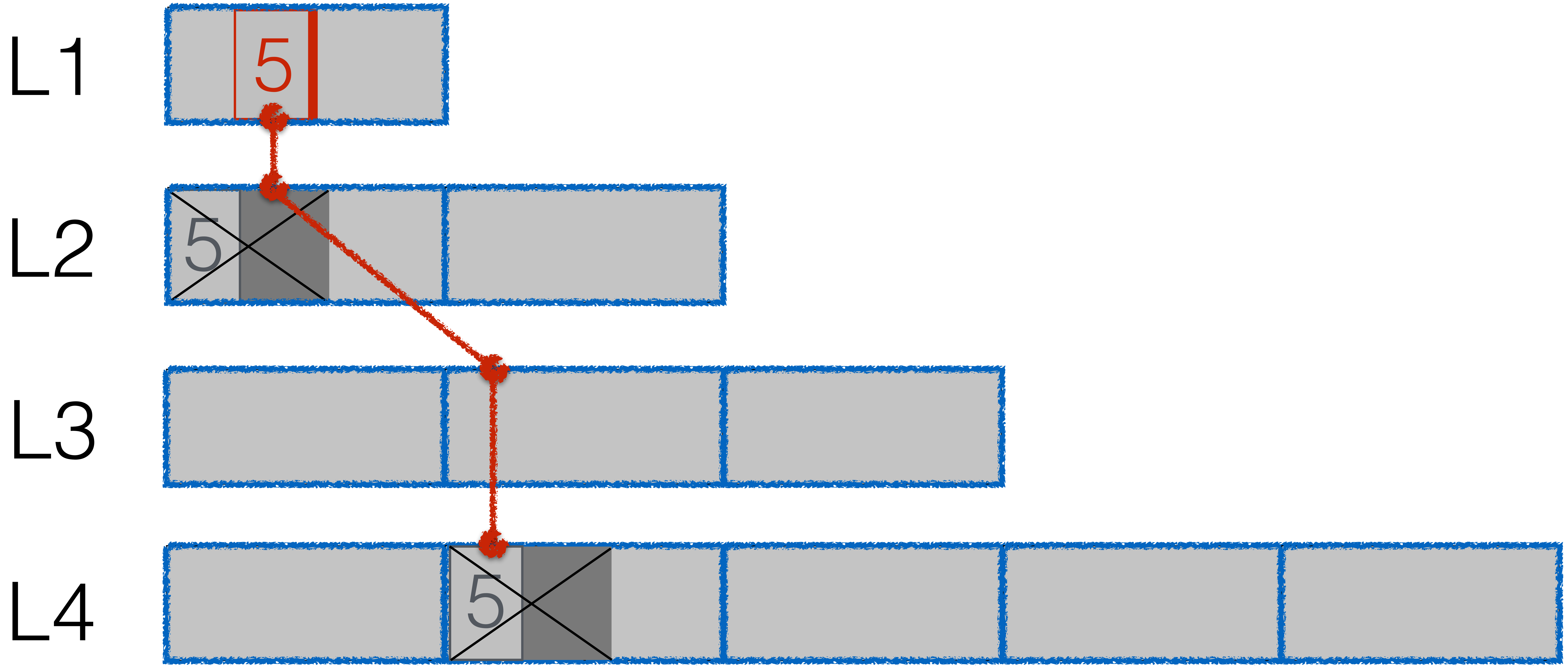
poor read perf.

write amplification

space amplification

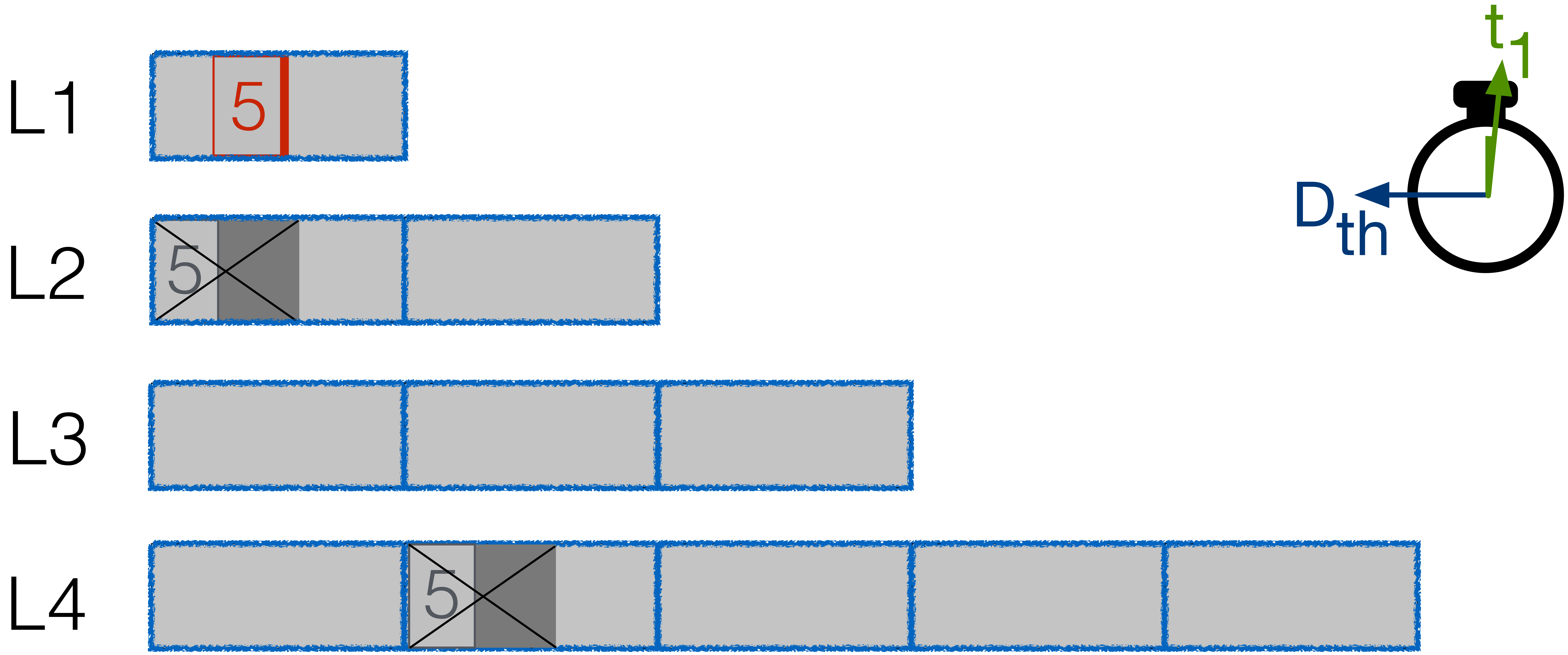
delete persistence latency

delete persistence latency



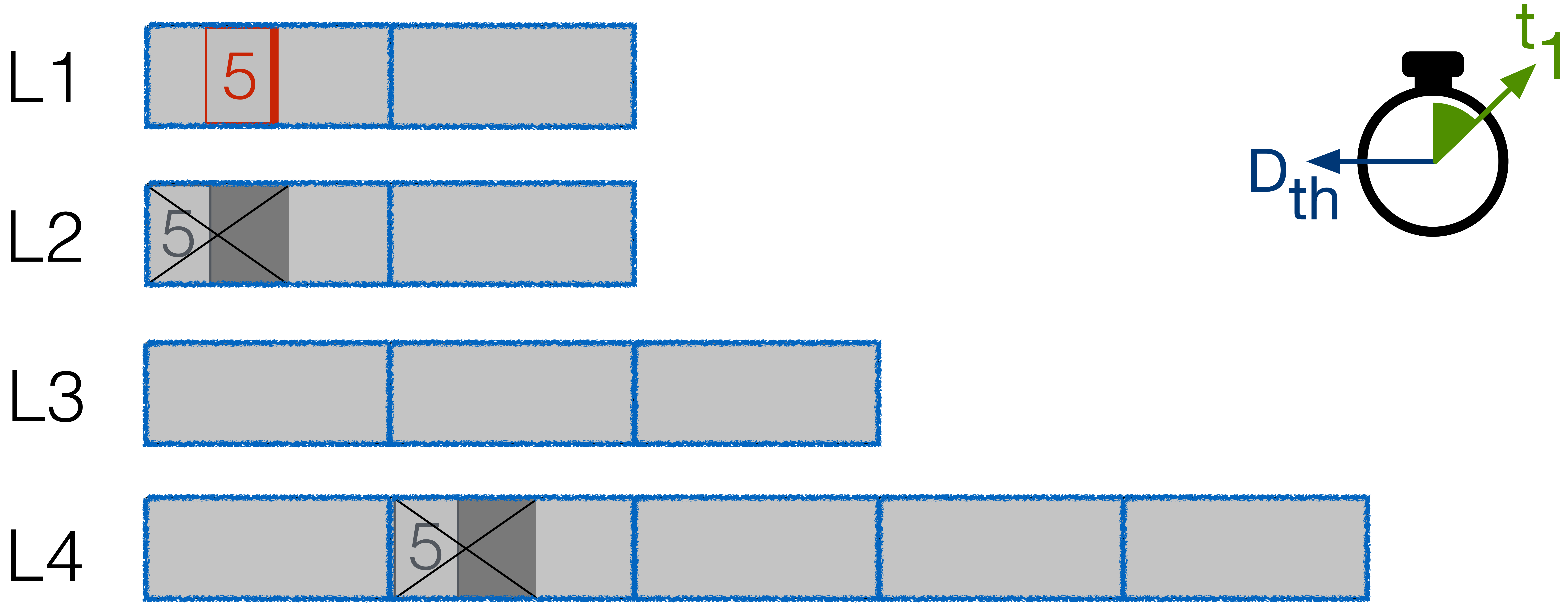
delete persistence latency

delete(5) within a threshold time: **D_{th}**



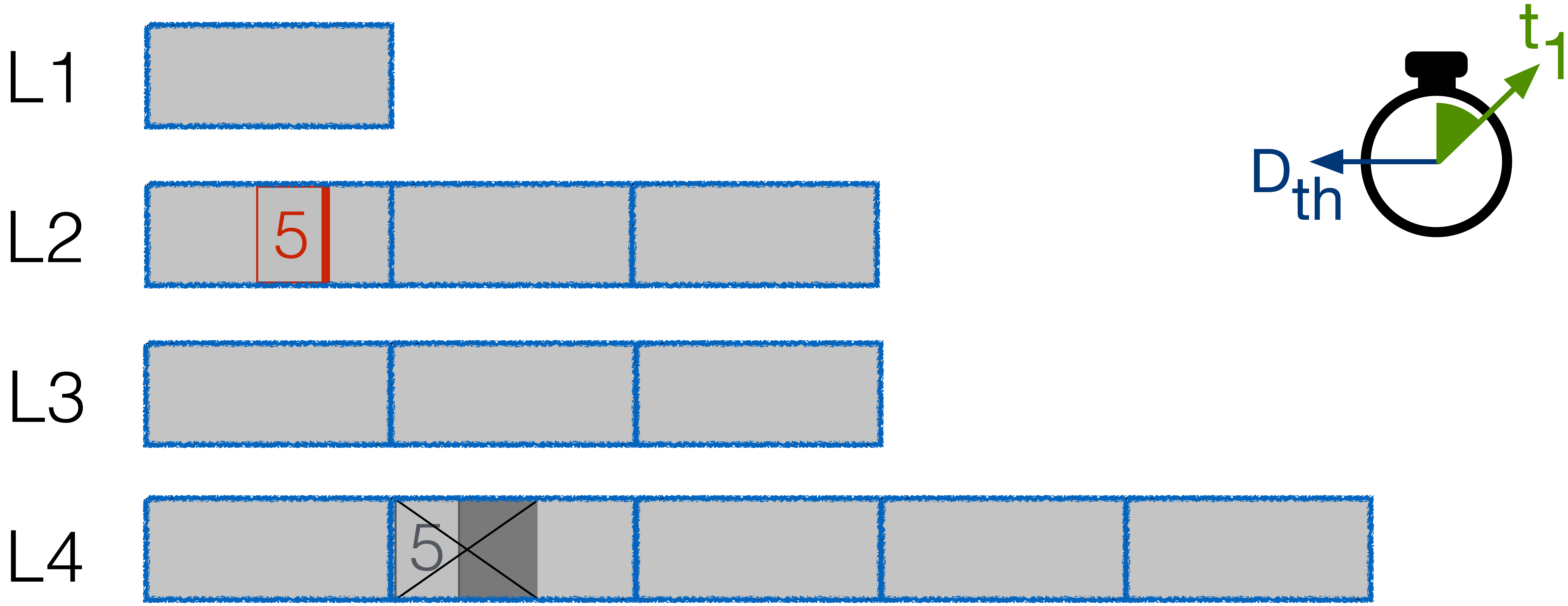
delete persistence latency

delete(5) within a threshold time: $\mathbf{D_{th}}$



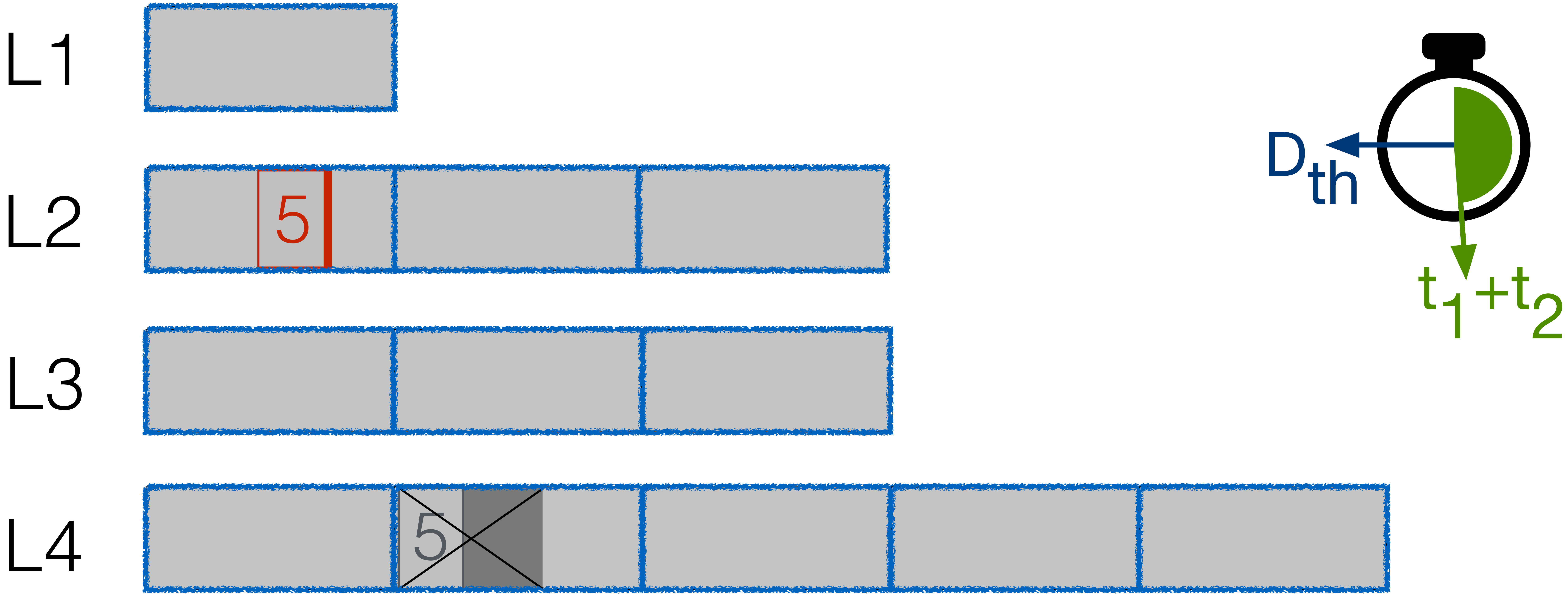
delete persistence latency

delete(5) within a threshold time: **D_{th}**



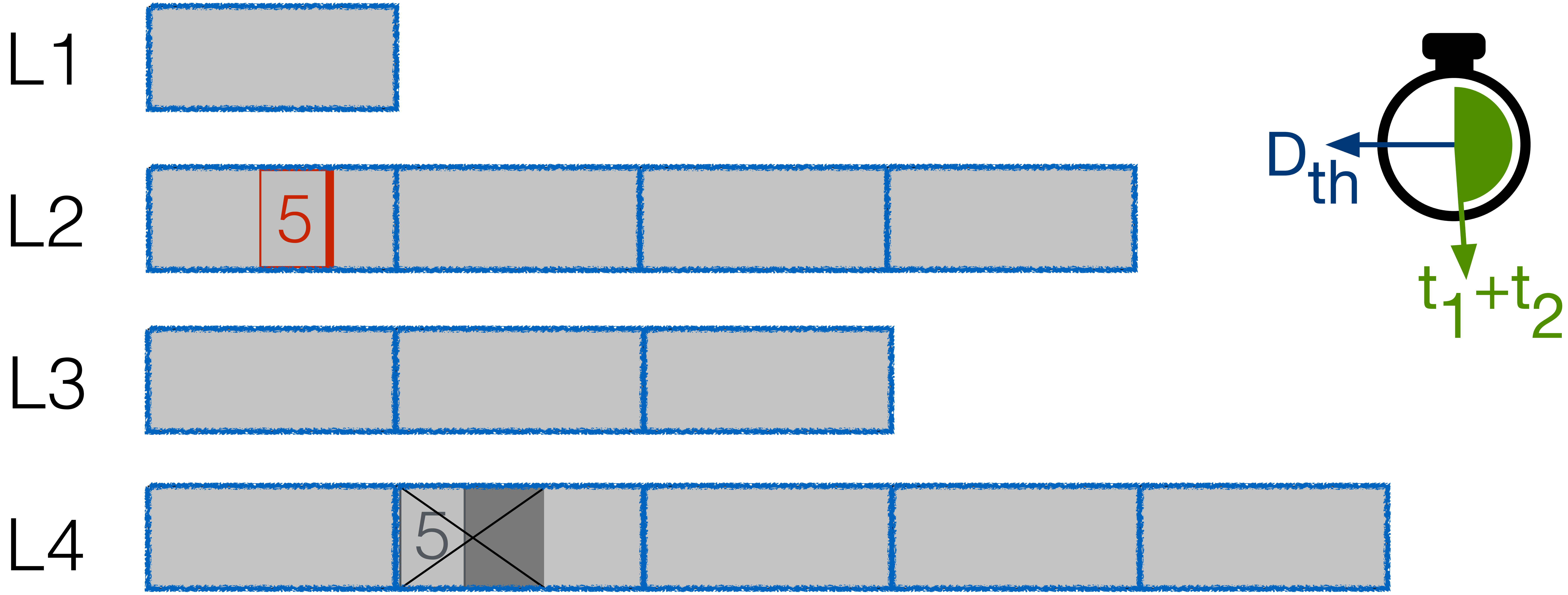
delete persistence latency

delete(5) within a threshold time: **D_{th}**



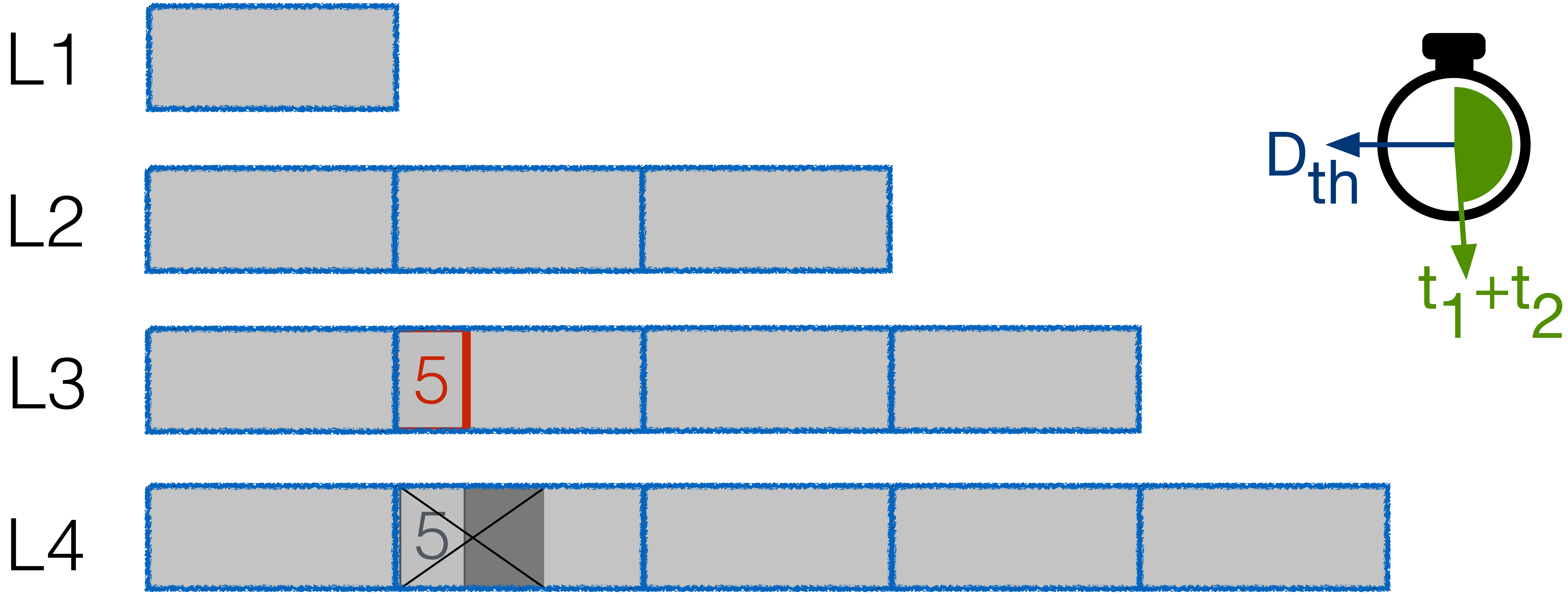
delete persistence latency

delete(5) within a threshold time: **D_{th}**



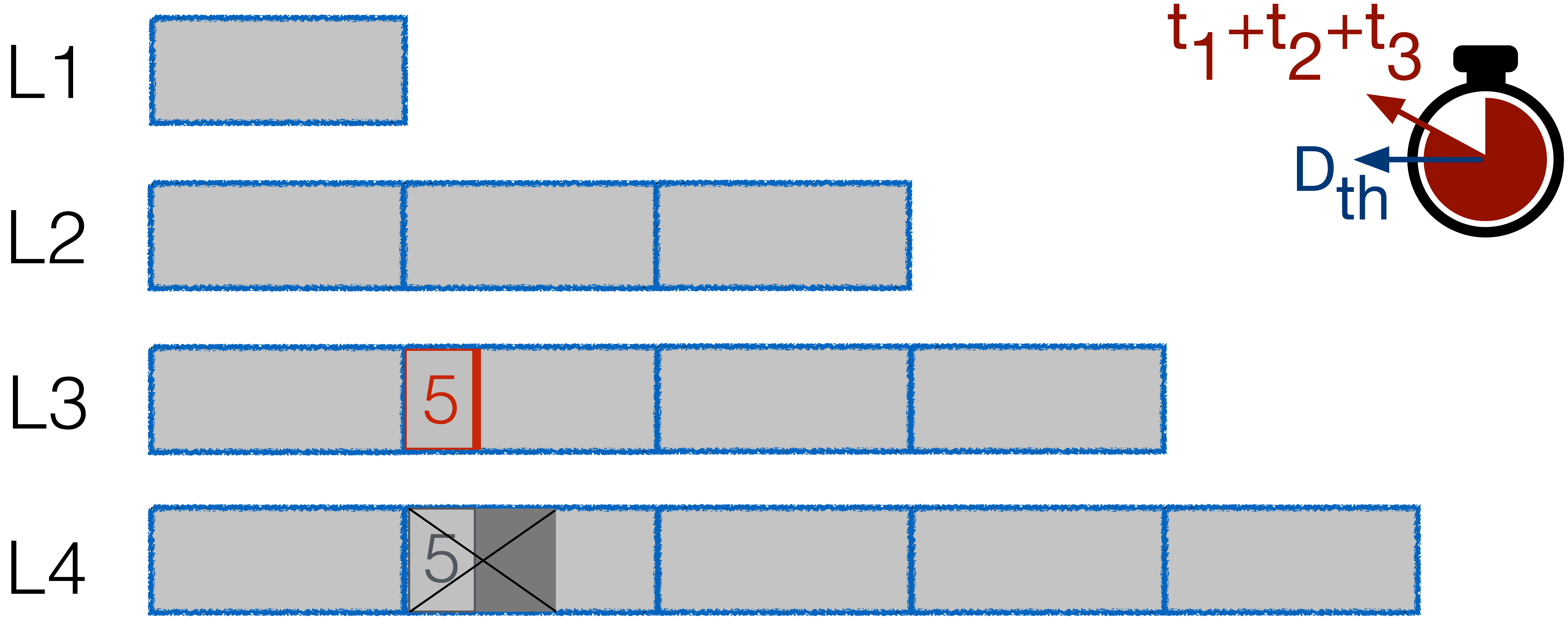
delete persistence latency

delete(5) within a threshold time: $\mathbf{D_{th}}$



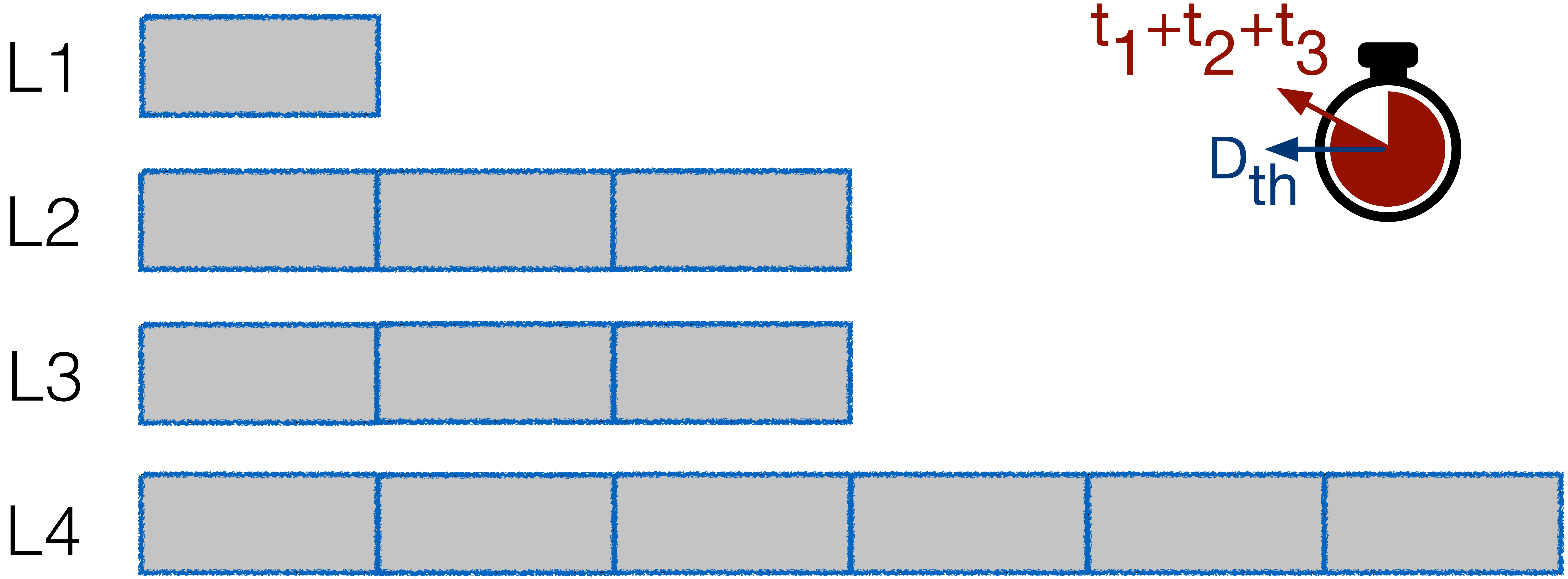
delete persistence latency

delete(5) within a threshold time: **D_{th}**



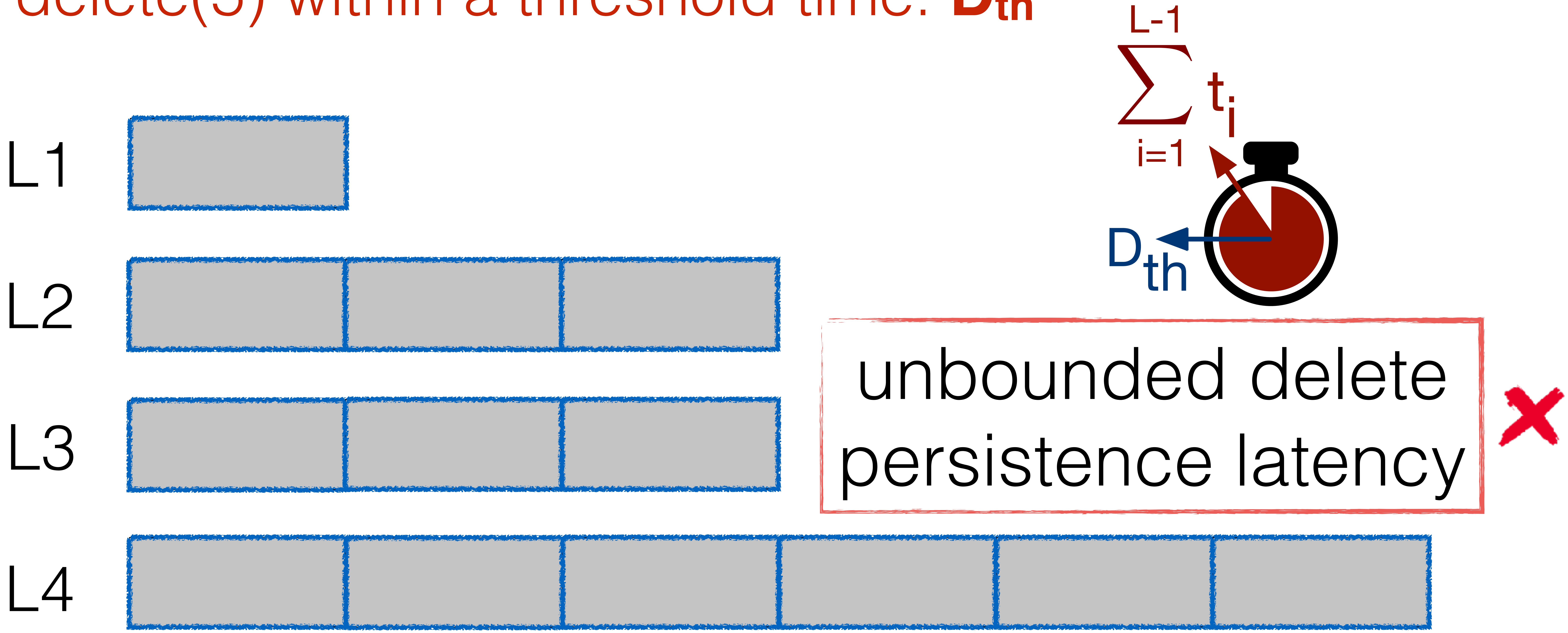
delete persistence latency

delete(5) within a threshold time: **D_{th}**



delete persistence latency

delete(5) within a threshold time: $\mathbf{D_{th}}$



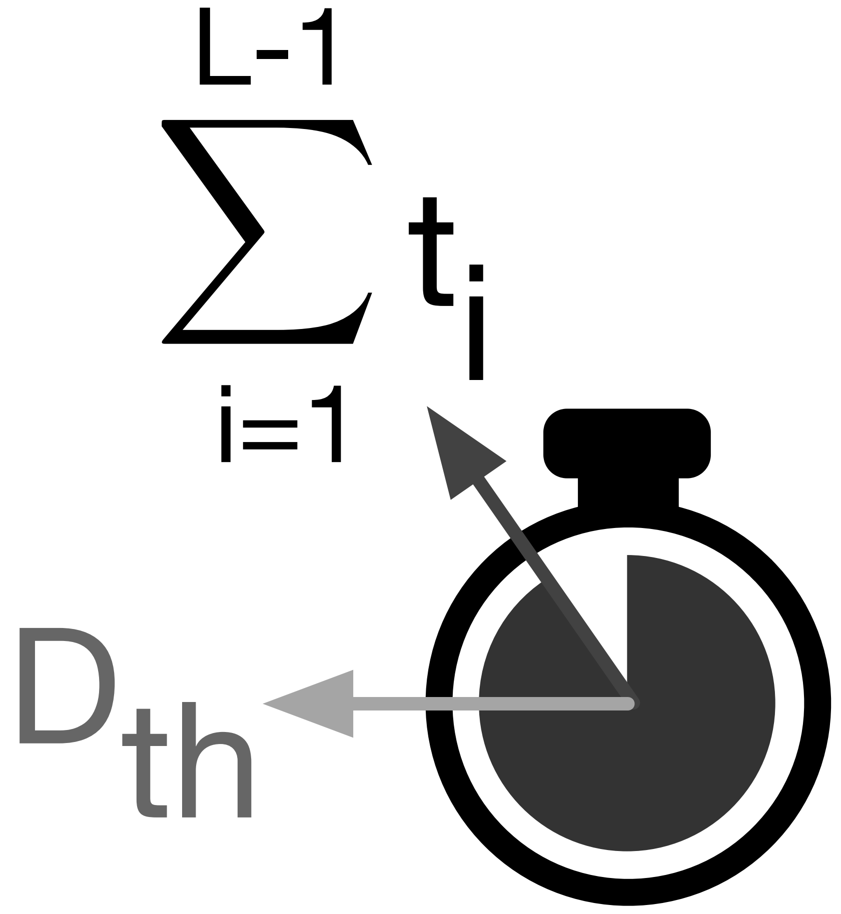
unbounded delete persistence latency ✗

The **Problem**

poor read perf.

write amplification

space amplification

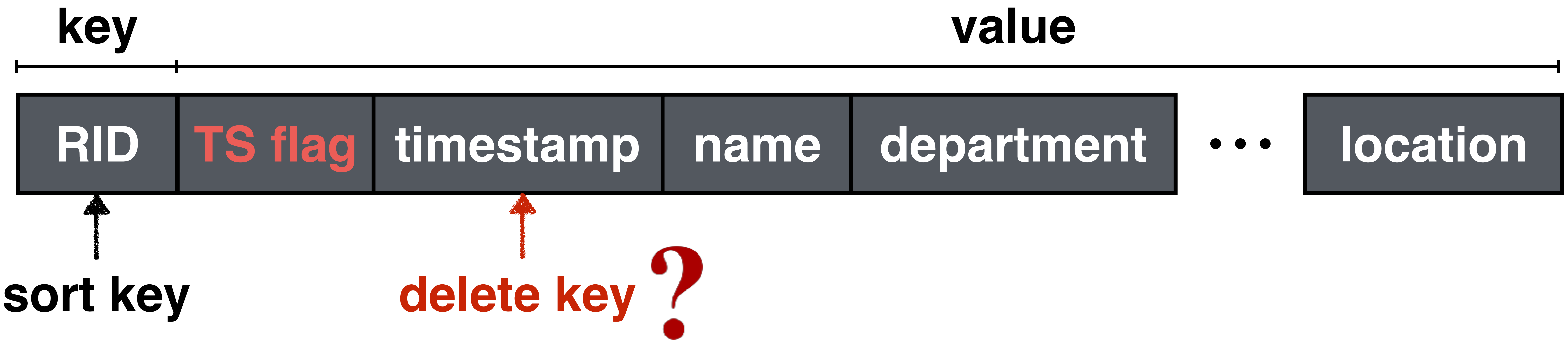


unbounded delete persistence latency

deletes on a secondary attribute

deletes on a secondary attribute

delete all entries older than: **D days**



key

value

RID

TS flag

timestamp

name

department

• • •

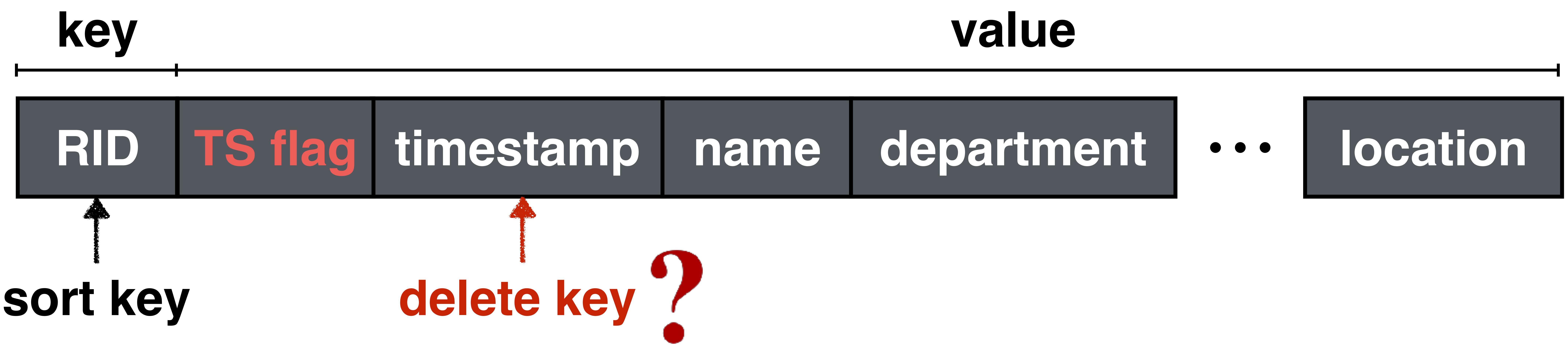
location

sort key

delete key ?

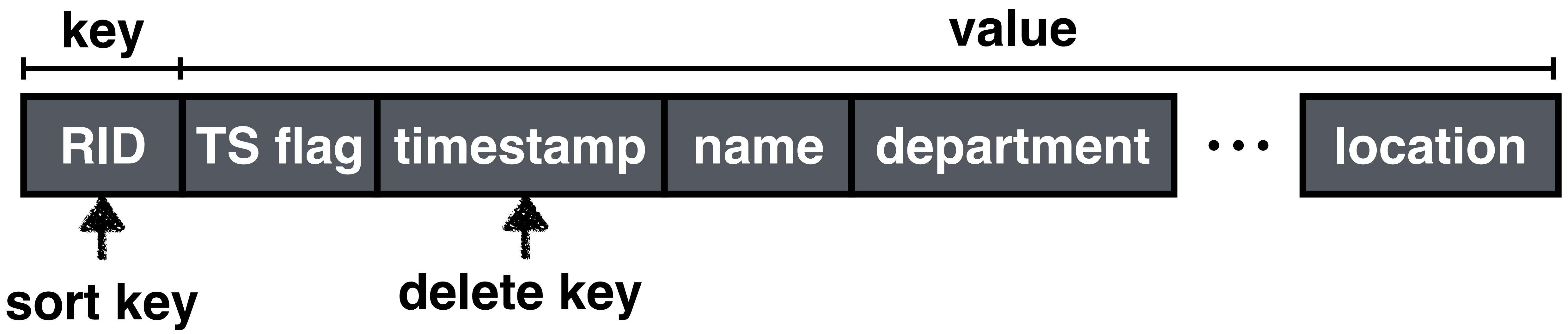
deletes on a secondary attribute

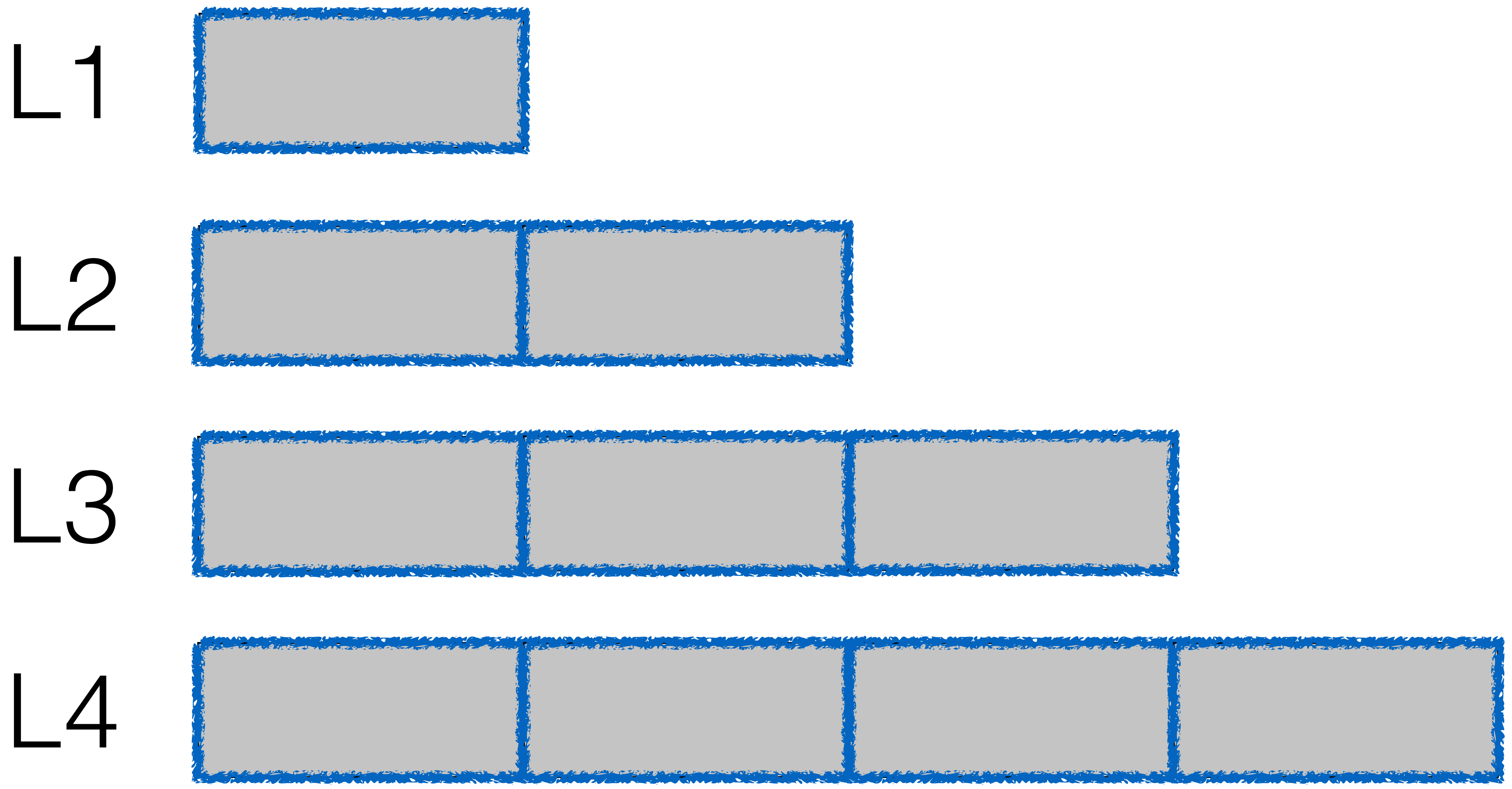
delete all entries older than: **D days**



deletes on a secondary attribute

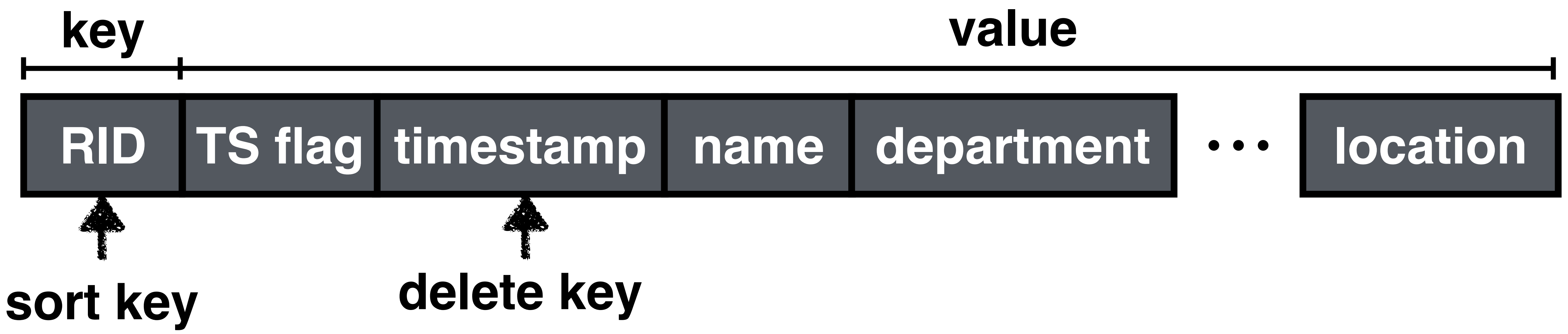
delete all entries older than: **D days**

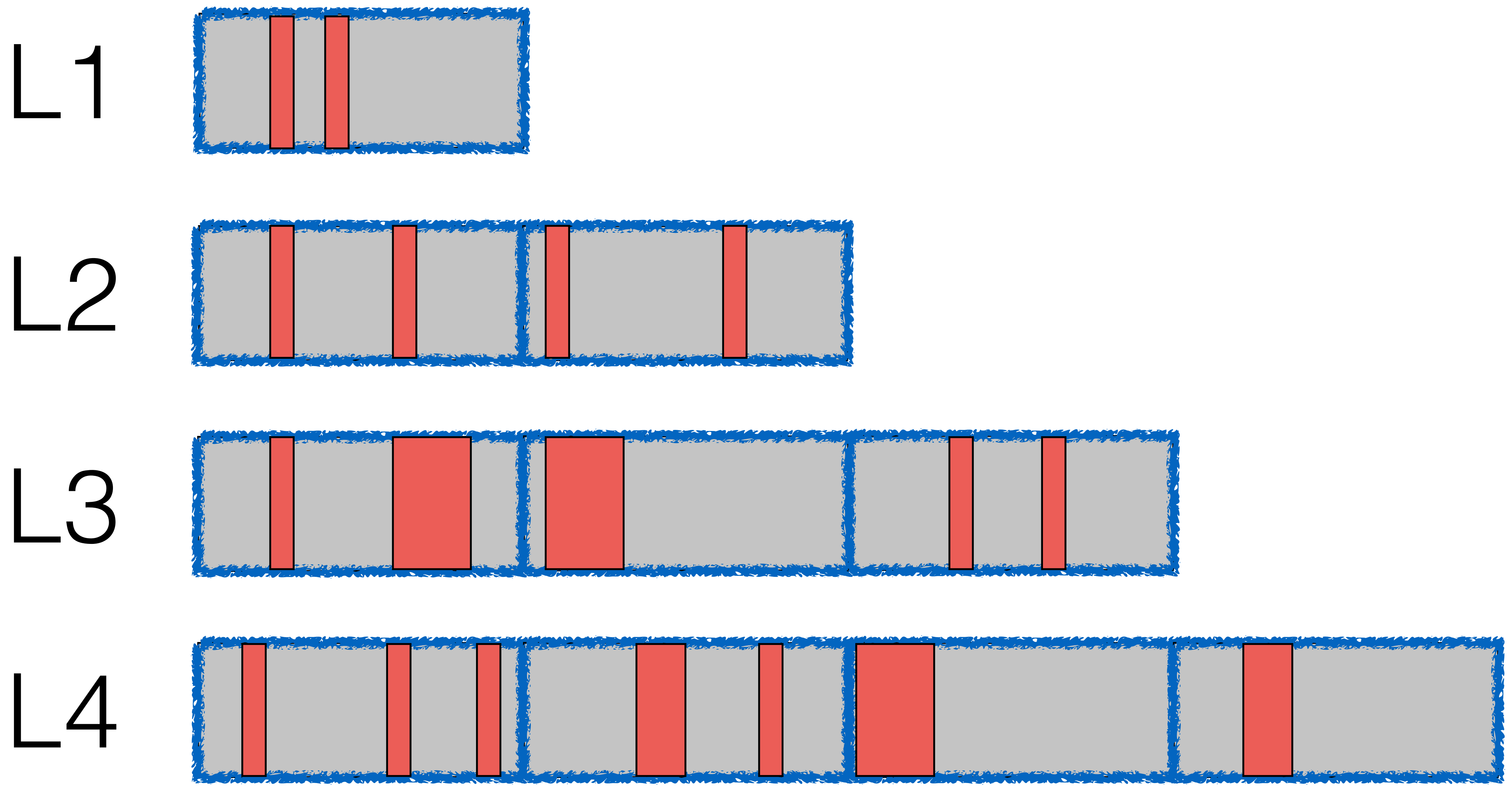




deletes on a secondary attribute

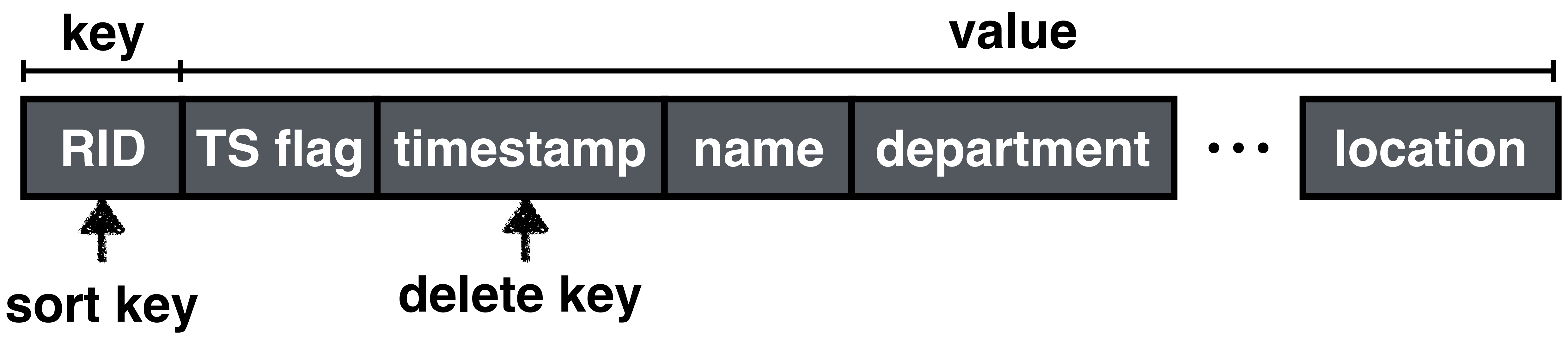
delete all entries older than: **D days**





deletes on a secondary attribute

delete all entries older than: **D days**



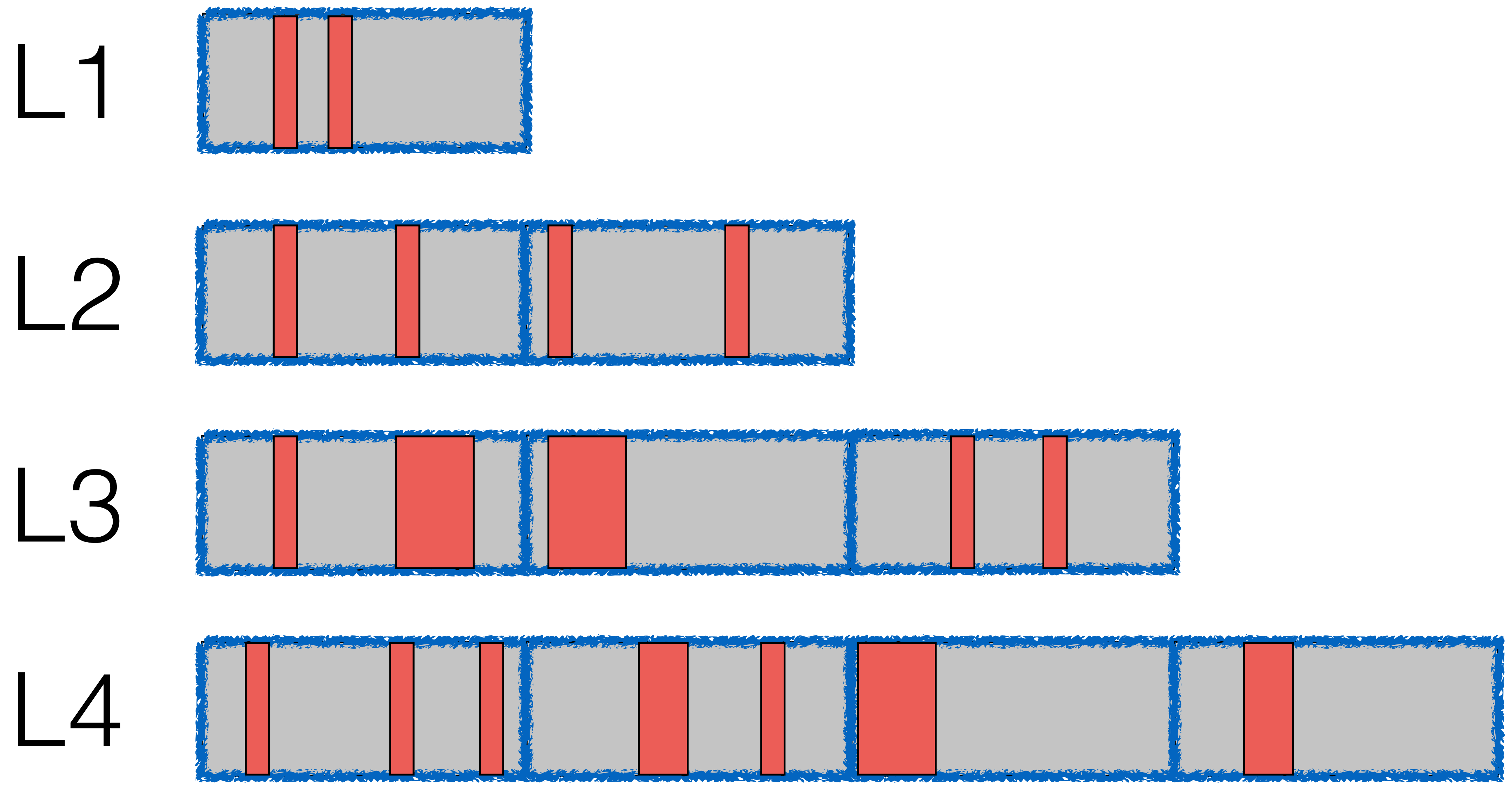
key

value

RID | TS flag | timestamp | name | department | ··· | location

sort key

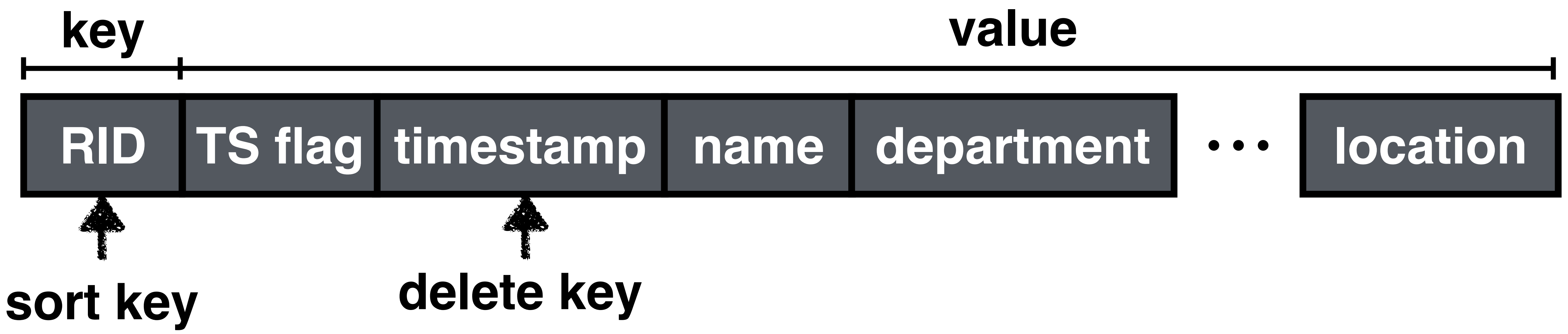
delete key



scattered occurrences

deletes on a secondary attribute

delete all entries older than: **D days**



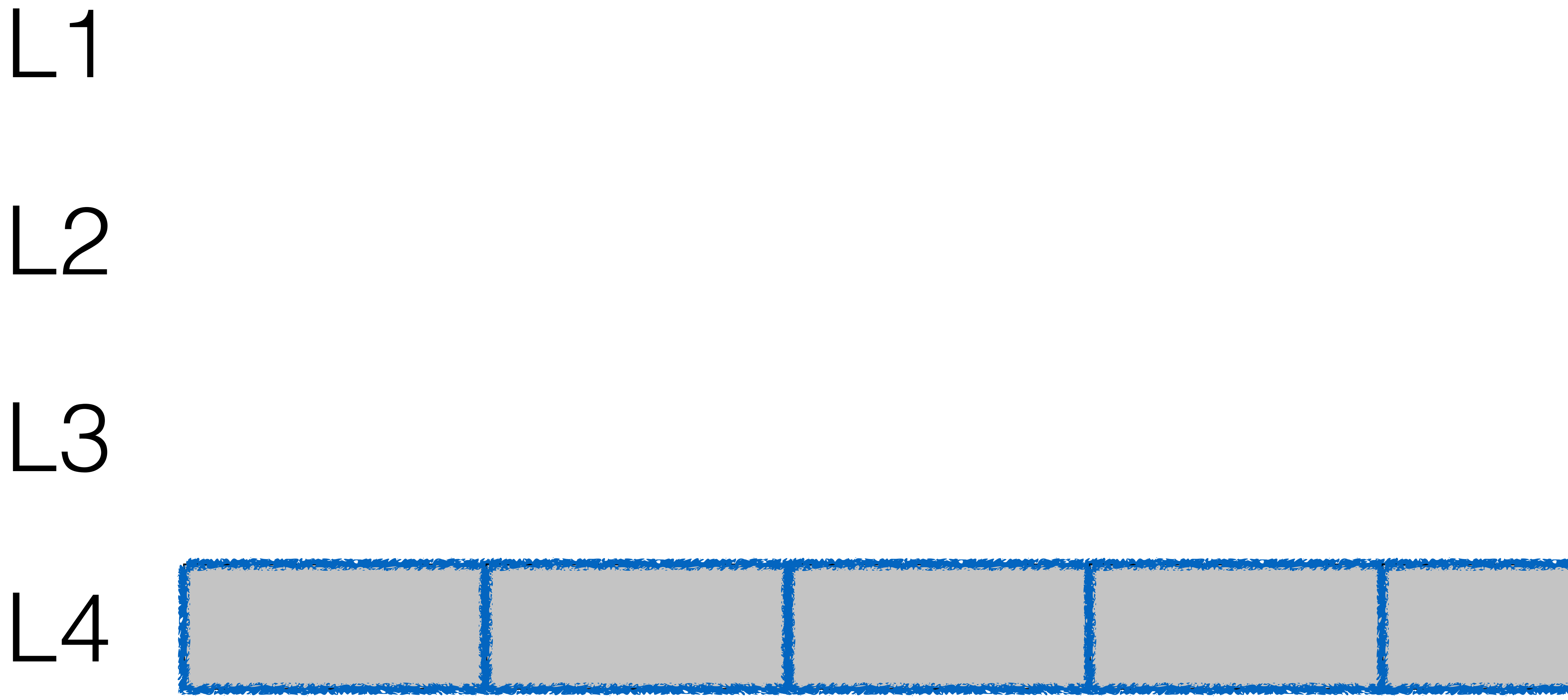
key

value

RID | TS flag | timestamp | name | department | ··· | location

sort key

delete key



L1

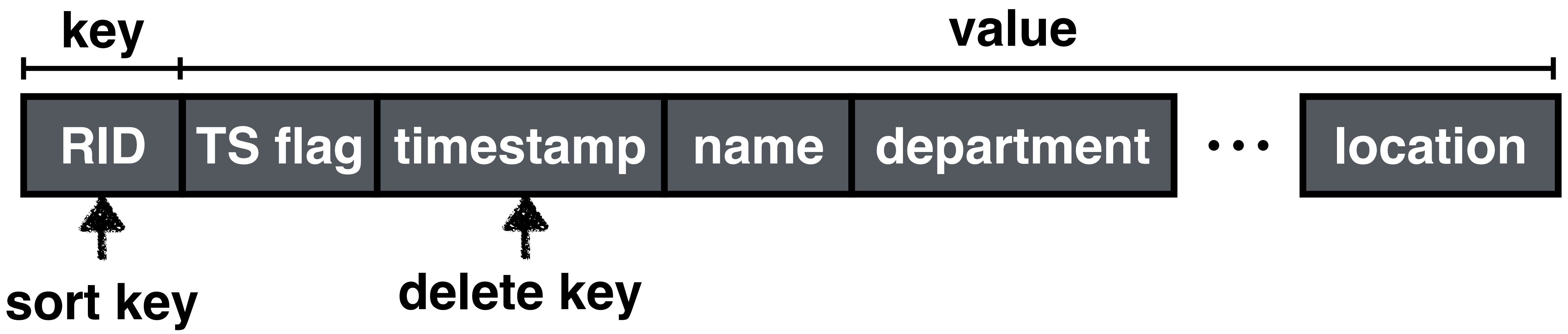
L2

L3

L4

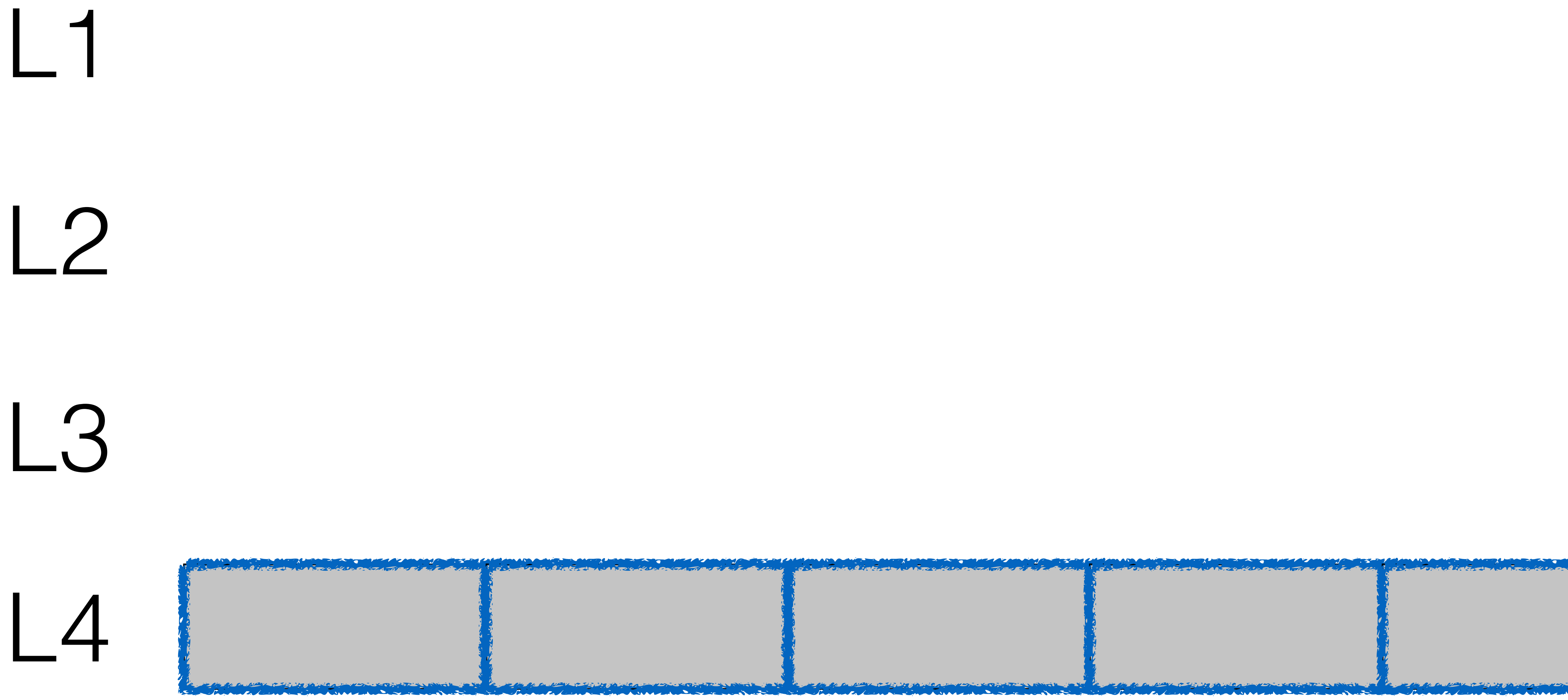
deletes on a secondary attribute

delete all entries older than: **D days**



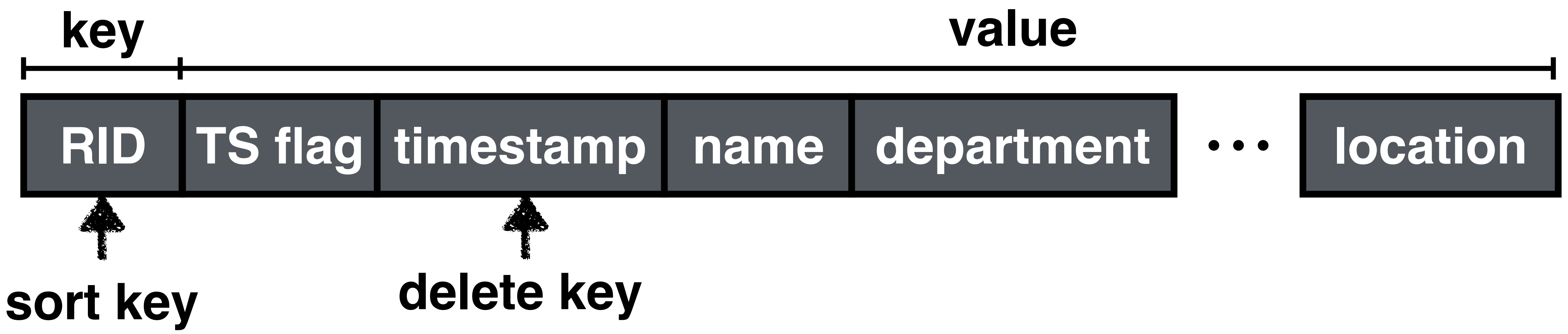
latency spikes ✗

superfluous I/Os ✗



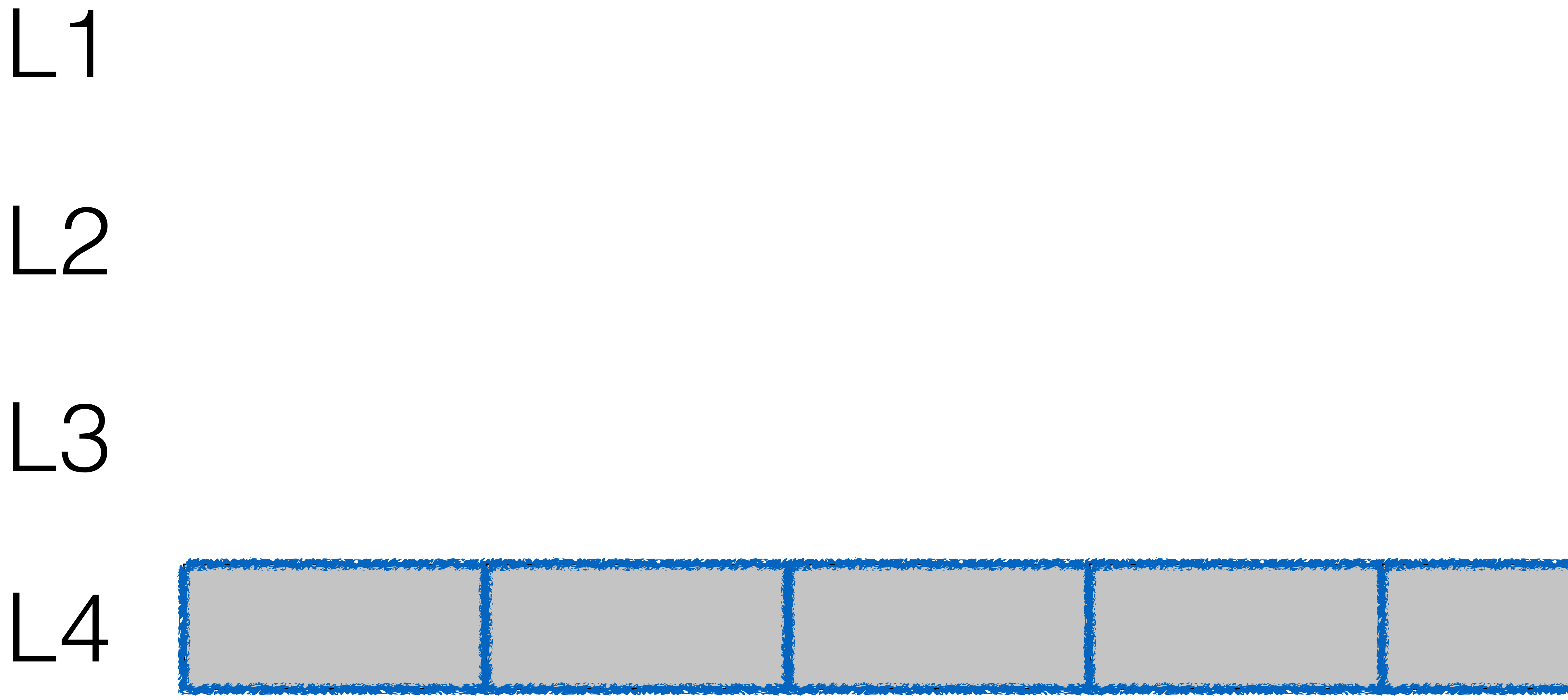
deletes on a secondary attribute

delete all entries older than: **D days**



latency spikes ✗

superfluous I/Os ✗

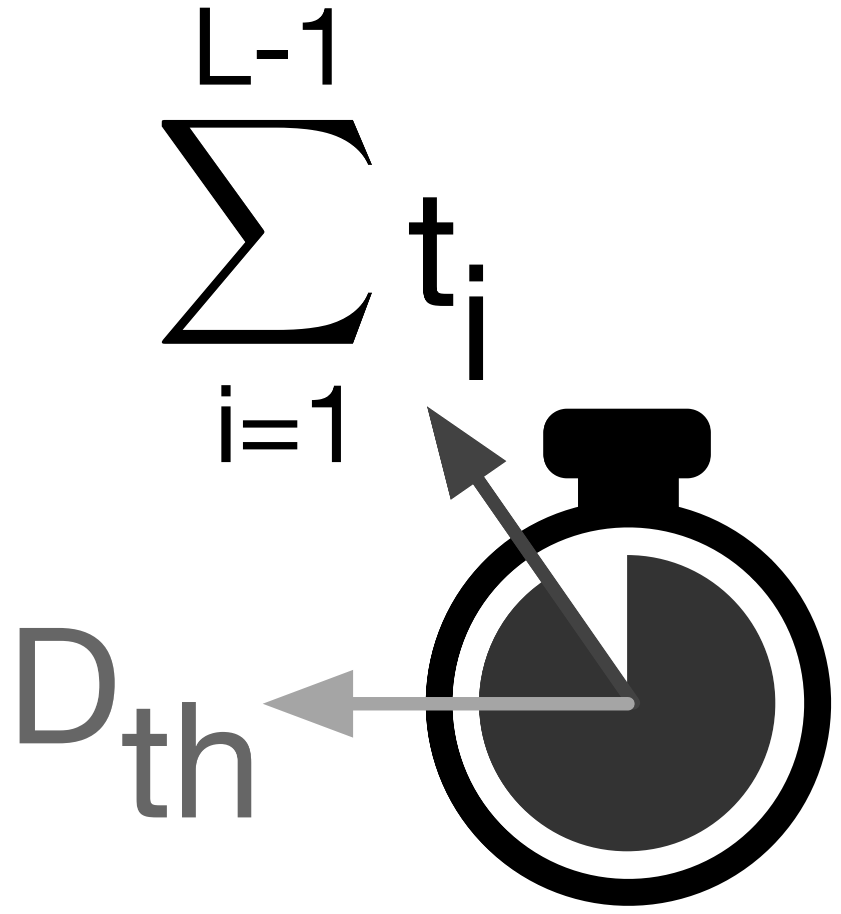


The **Problem**

poor read perf.

write amplification

space amplification



unbounded delete persistence latency

latency spikes

superfluous I/Os

Outline

Part 1: **LSM Basics**

Part 2: **The Problem: Deletes in LSMs**

Part 3: **Lethe: Enabling Efficient Deletes**

Outline

Part 1: **LSM Basics**

Part 2: **The Problem: Deletes in LSMs**

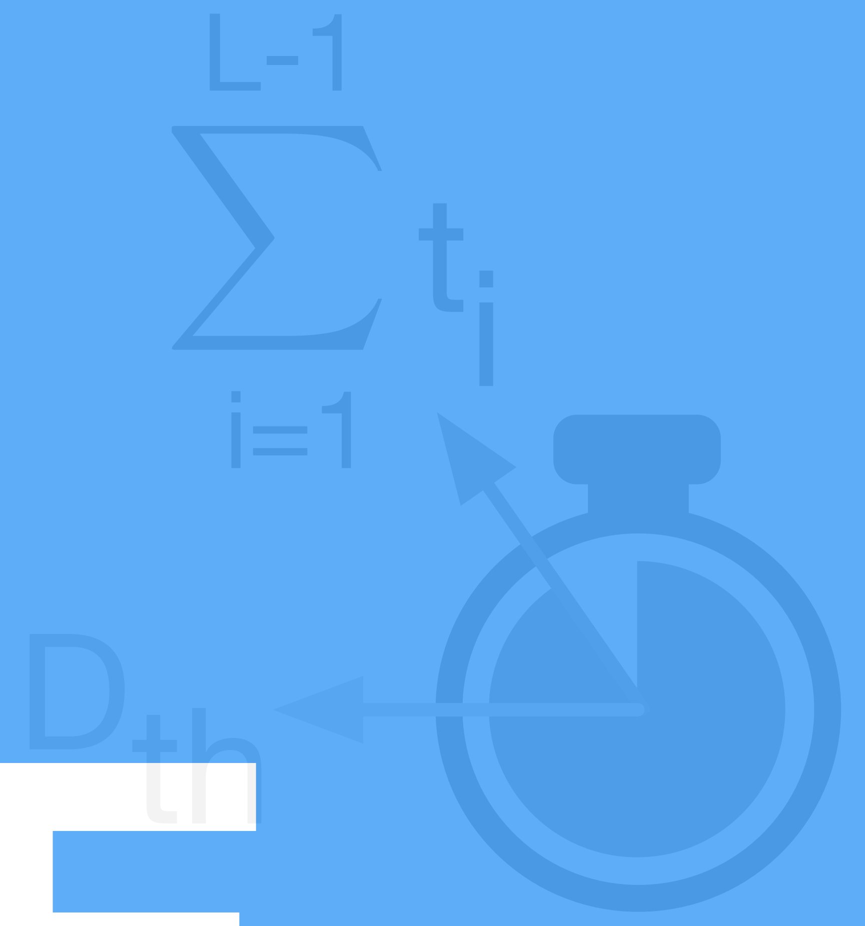
Part 3: **Lethe: Enabling Efficient Deletes**

FADE

poor read perf.

write amplification

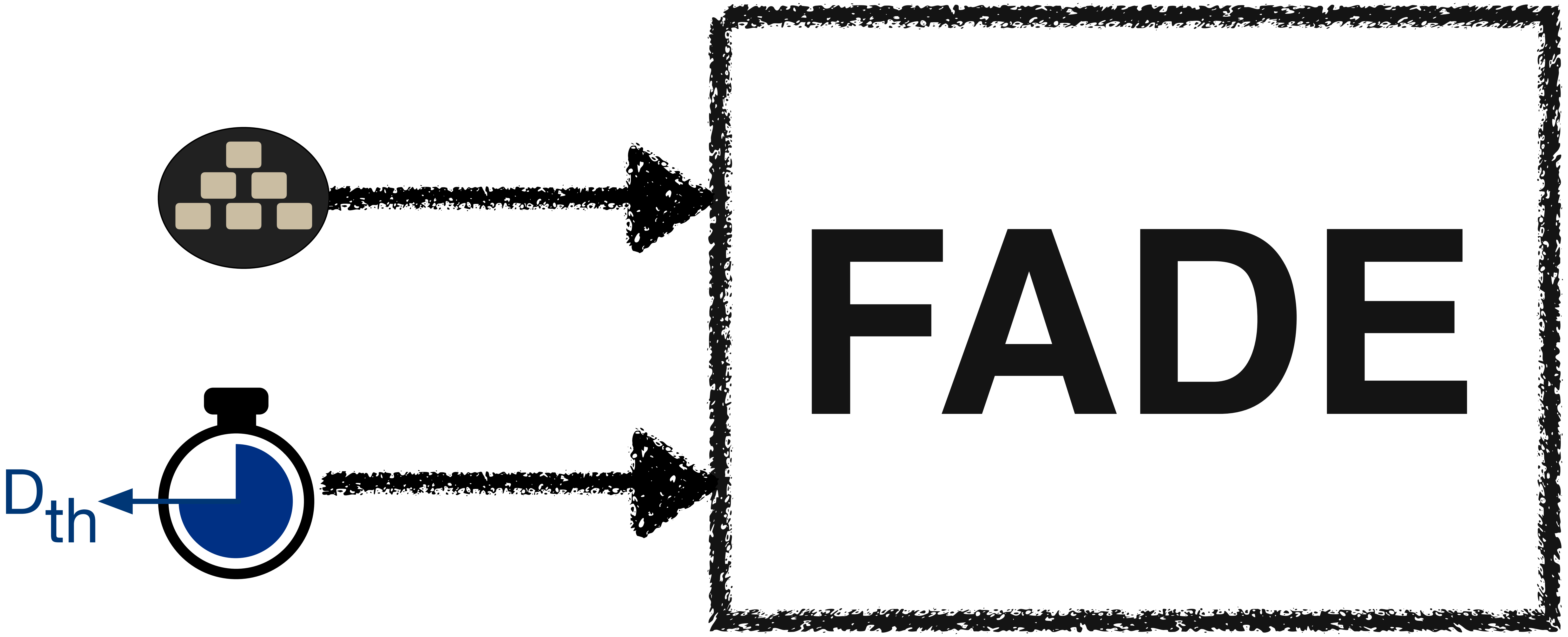
space amplification



unbounded delete persistence latency

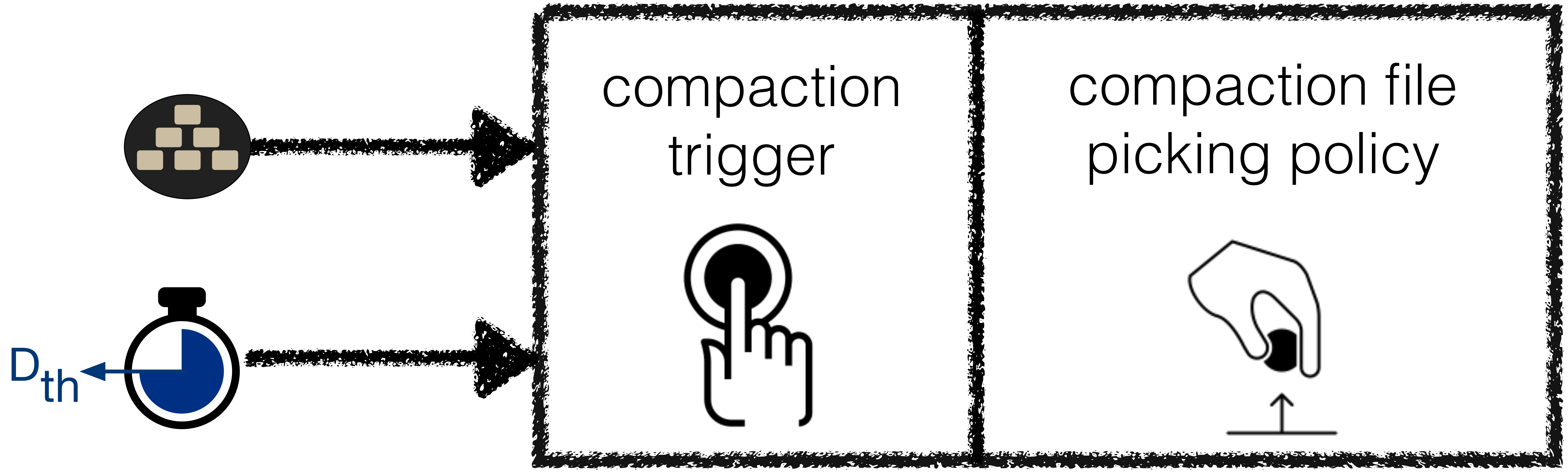
latency spikes

superfluous I/Os



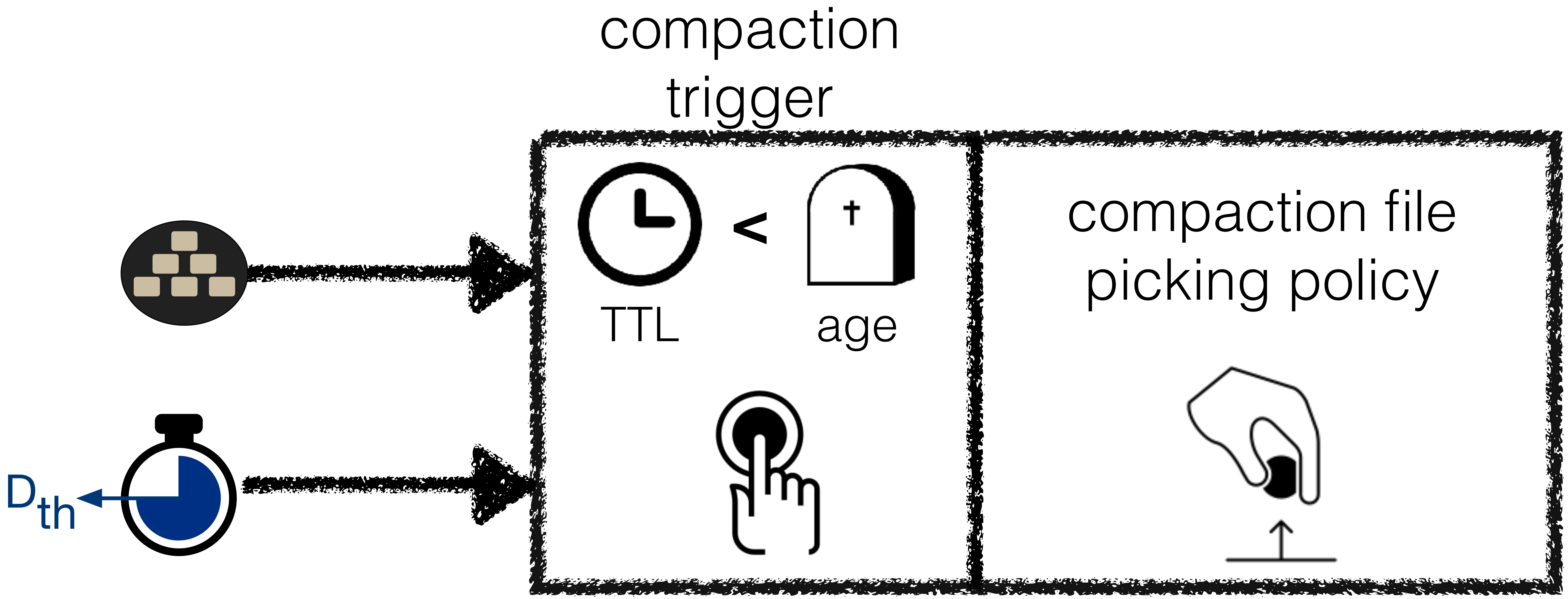
D_{th}

FADE



compaction trigger

compaction file picking policy

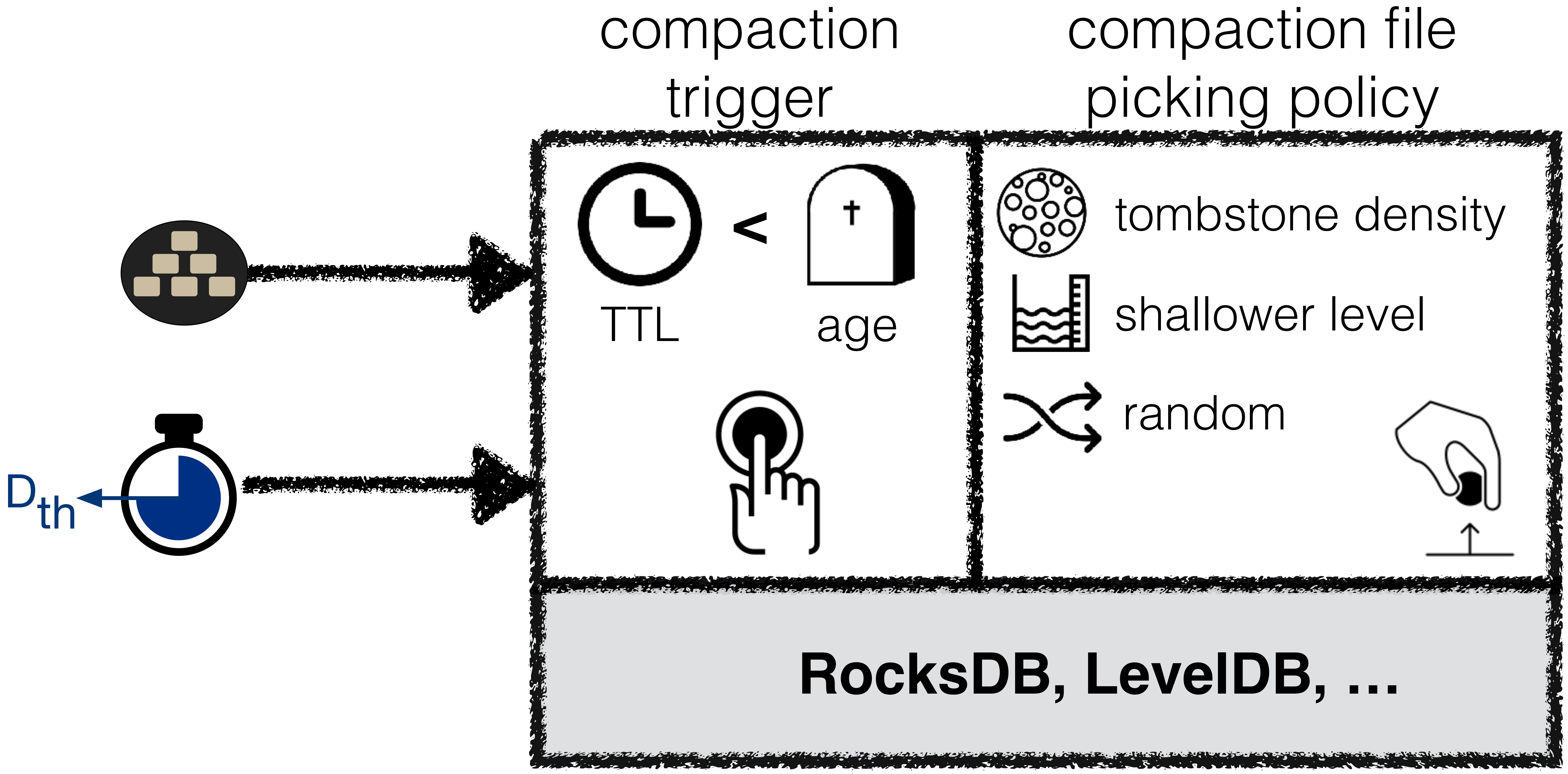


compaction trigger

TTL < age

compaction file picking policy

D_{th}



compaction trigger

compaction file picking policy

D_{th}

TTL < age

tombstone density

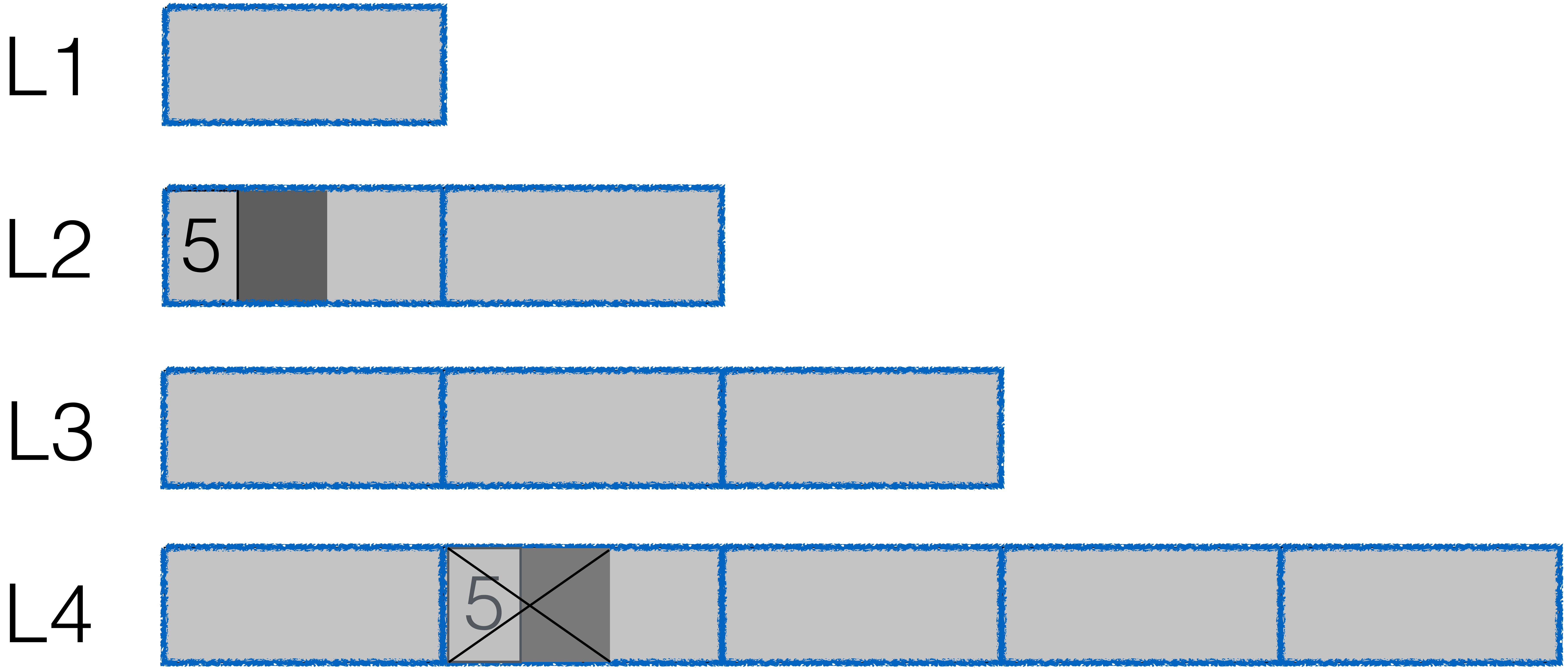
shallower level

random

RocksDB, LevelDB, …

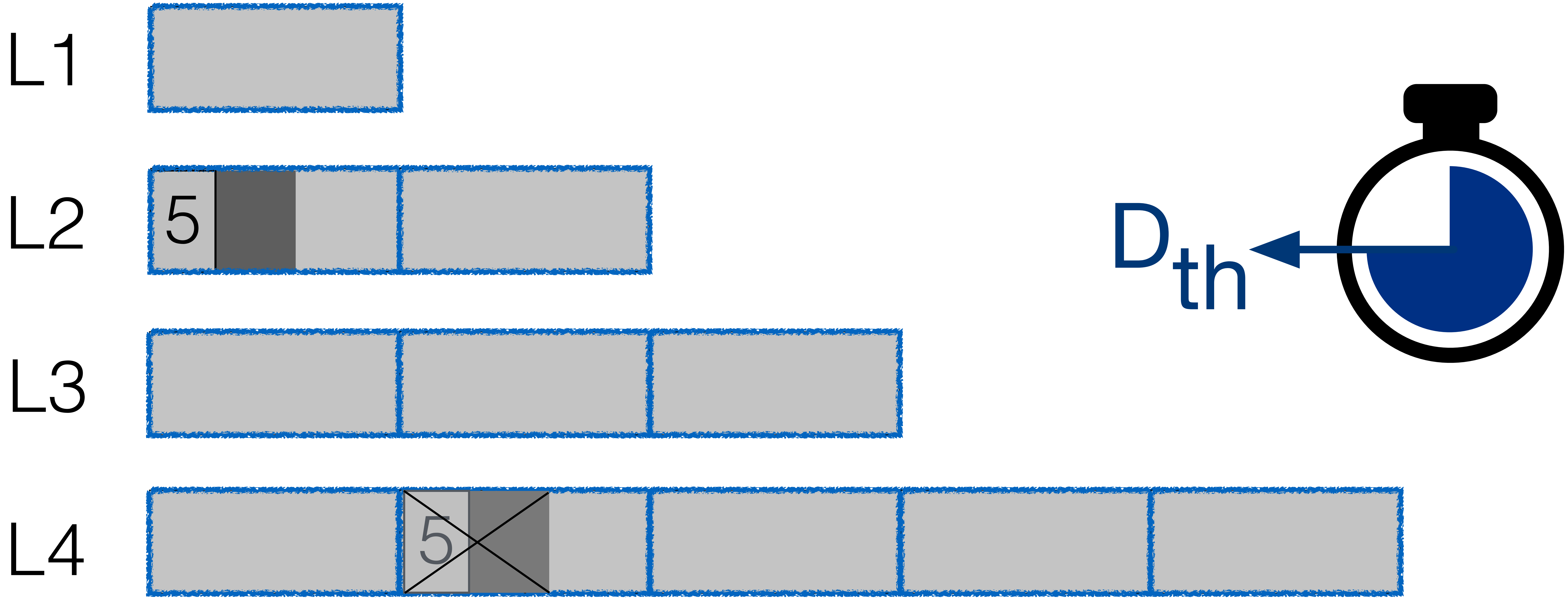
FAst DElete

delete(5) within a threshold time: $\mathbf{D_{th}}$



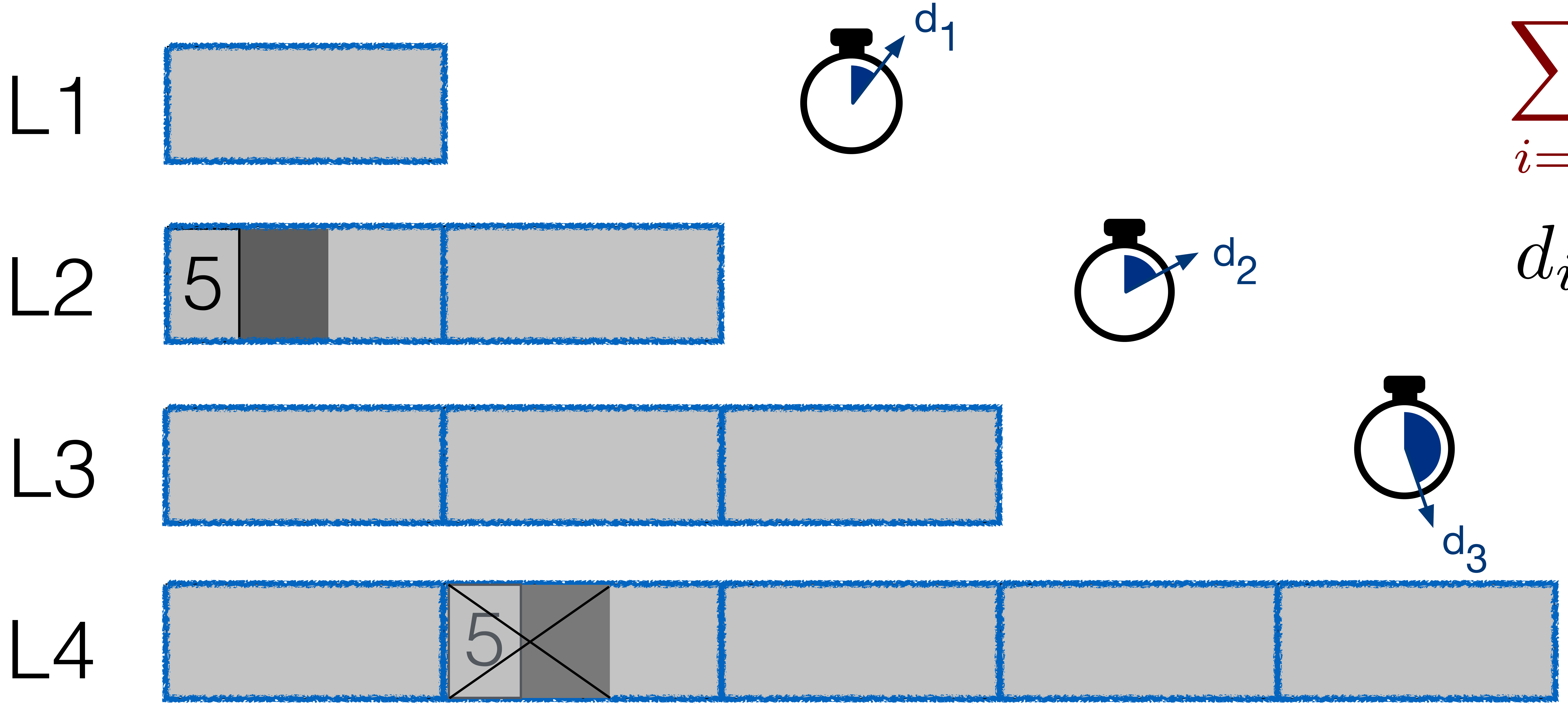
FAst DElete

delete(5) within a threshold time: **D_{th}**



FAst DElete

delete(5) within a threshold time: **D_{th}**

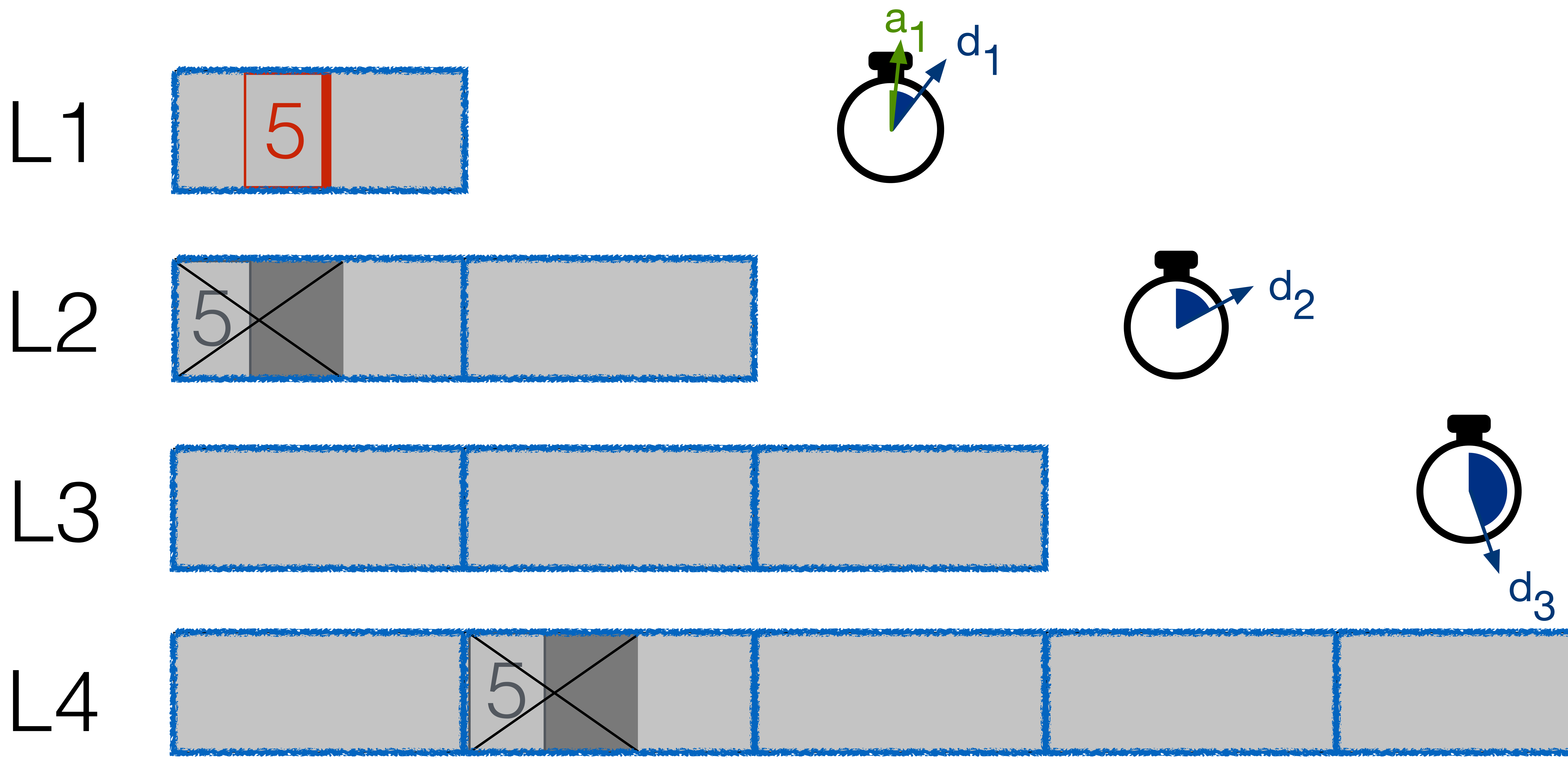


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

$$d_i = T \cdot d_{i-1}$$

FAst DElete

delete(5) within a threshold time: **D_{th}**

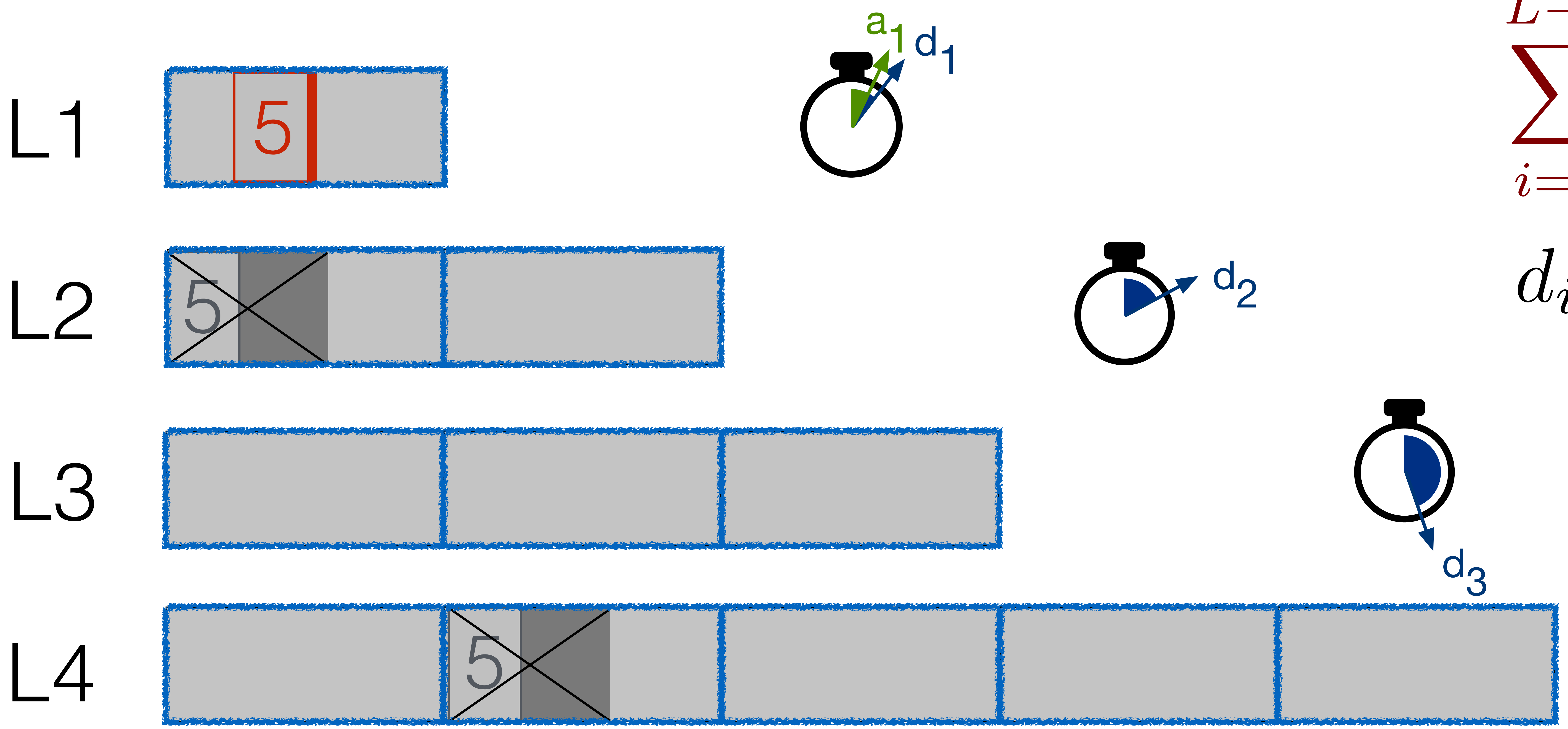


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

$$d_i = T \cdot d_{i-1}$$

FAst DElete

delete(5) within a threshold time: **D_{th}**

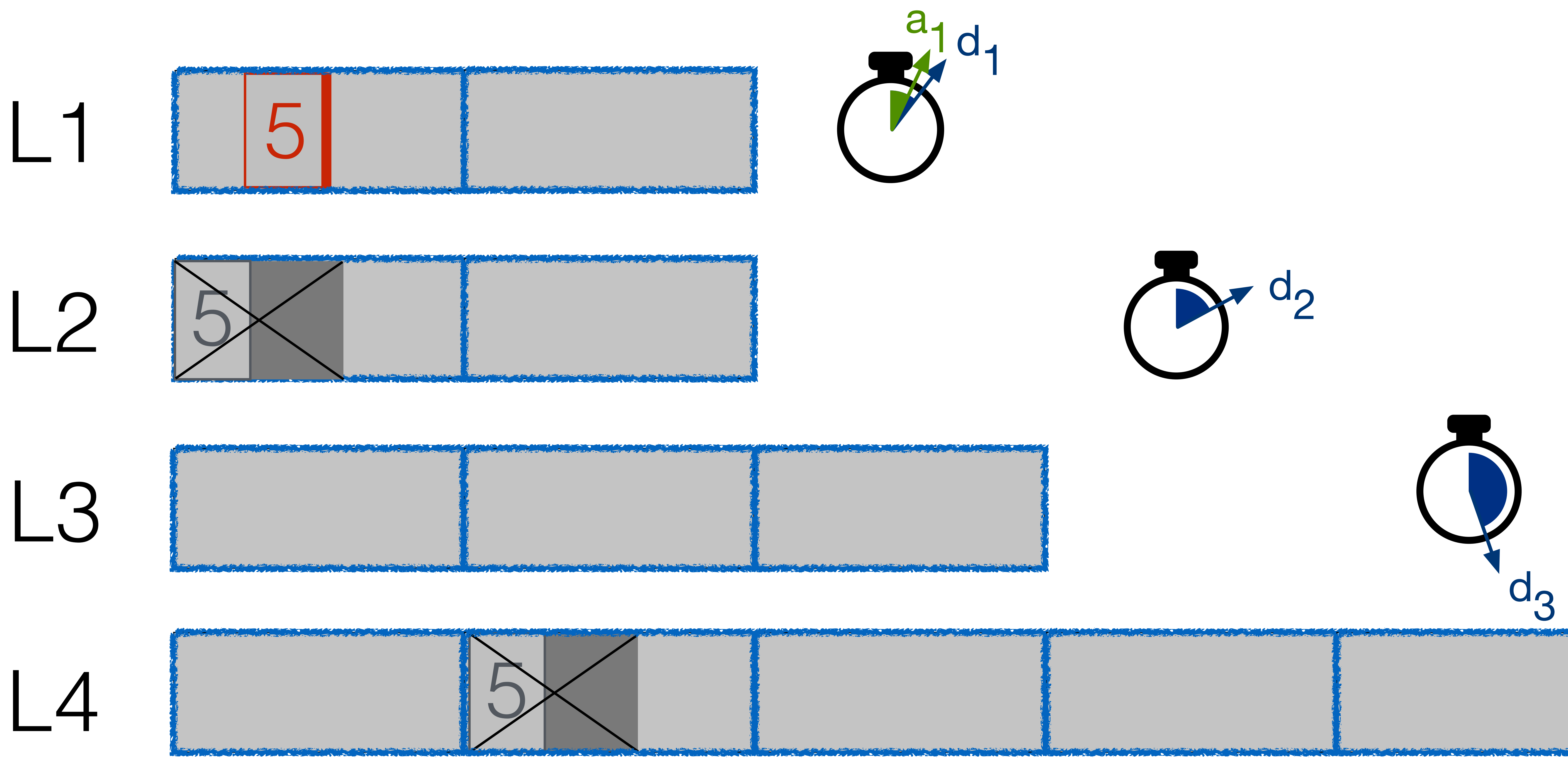


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

$$d_i = T \cdot d_{i-1}$$

FAst DElete

delete(5) within a threshold time: **D_{th}**

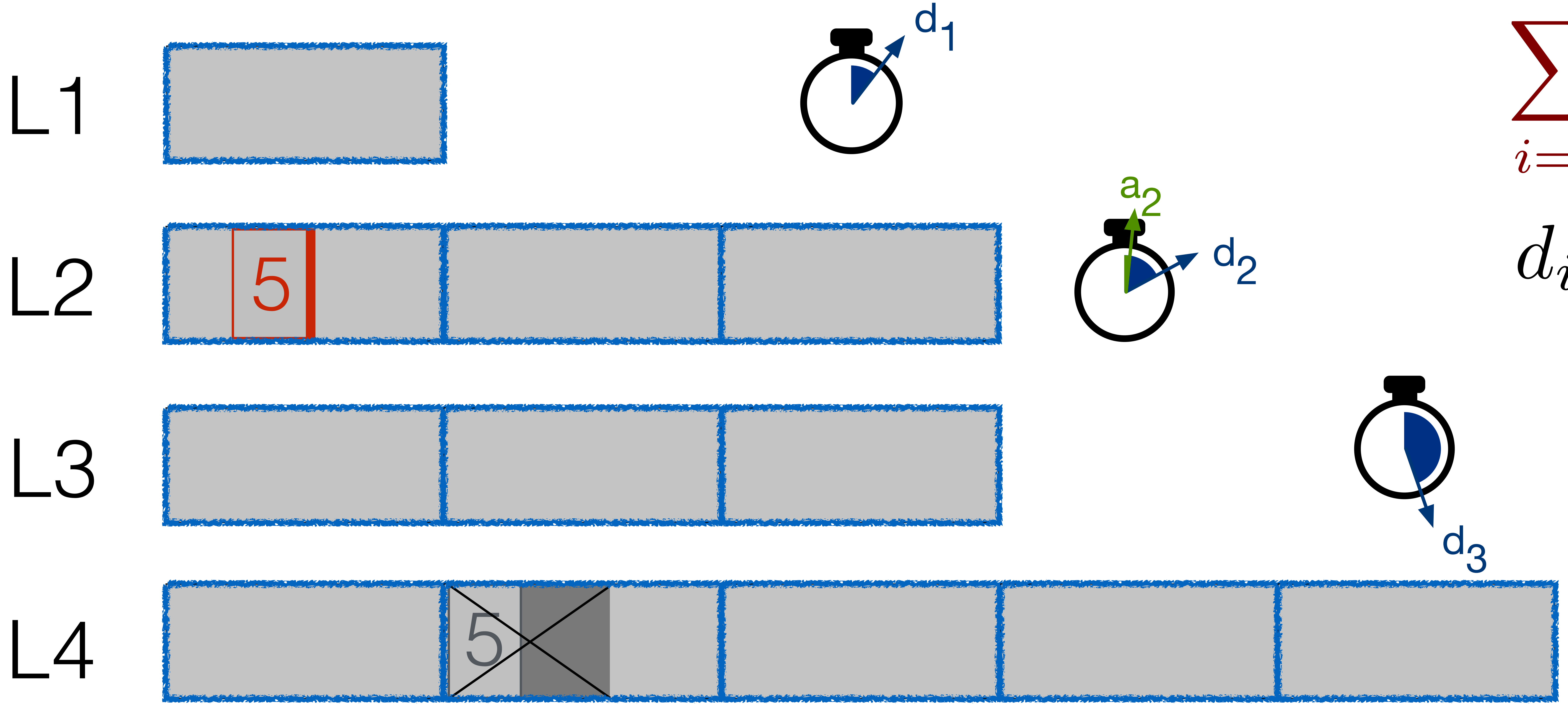


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

$$d_i = T \cdot d_{i-1}$$

FAst DElete

delete(5) within a threshold time: **D_{th}**

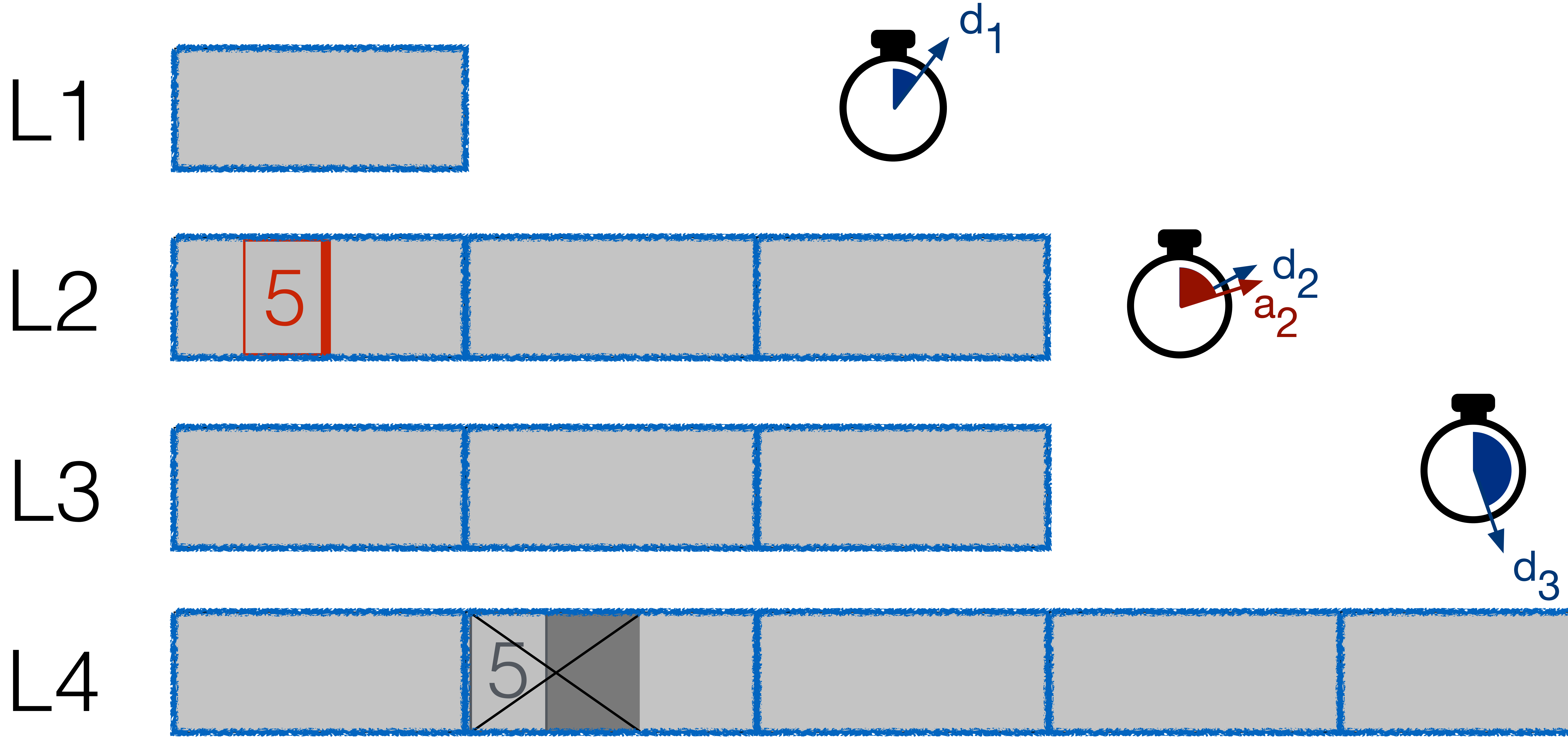


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

$$d_i = T \cdot d_{i-1}$$

FAst DElete

delete(5) within a threshold time: **D_th**

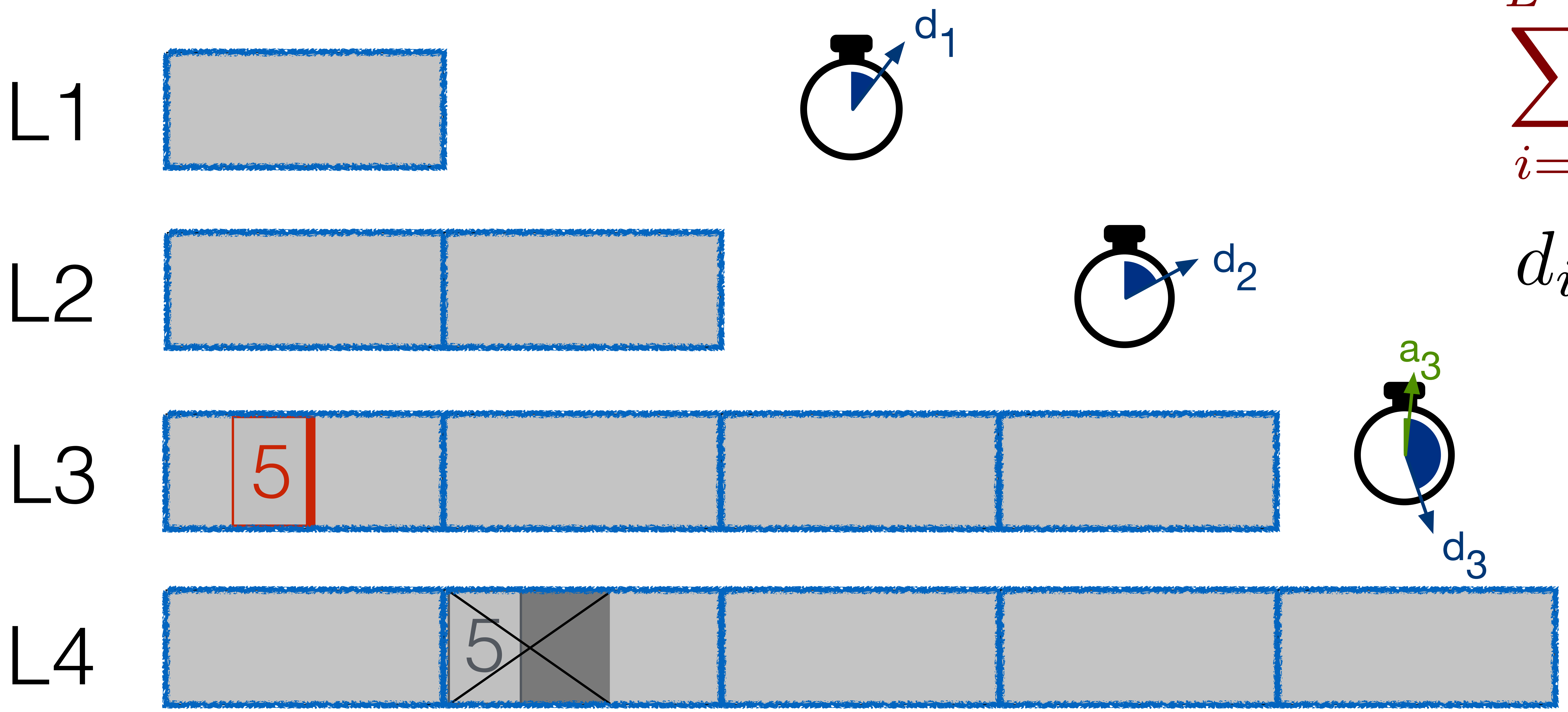


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

$$d_i = T \cdot d_{i-1}$$

FAst DElete

delete(5) within a threshold time: **D_{th}**

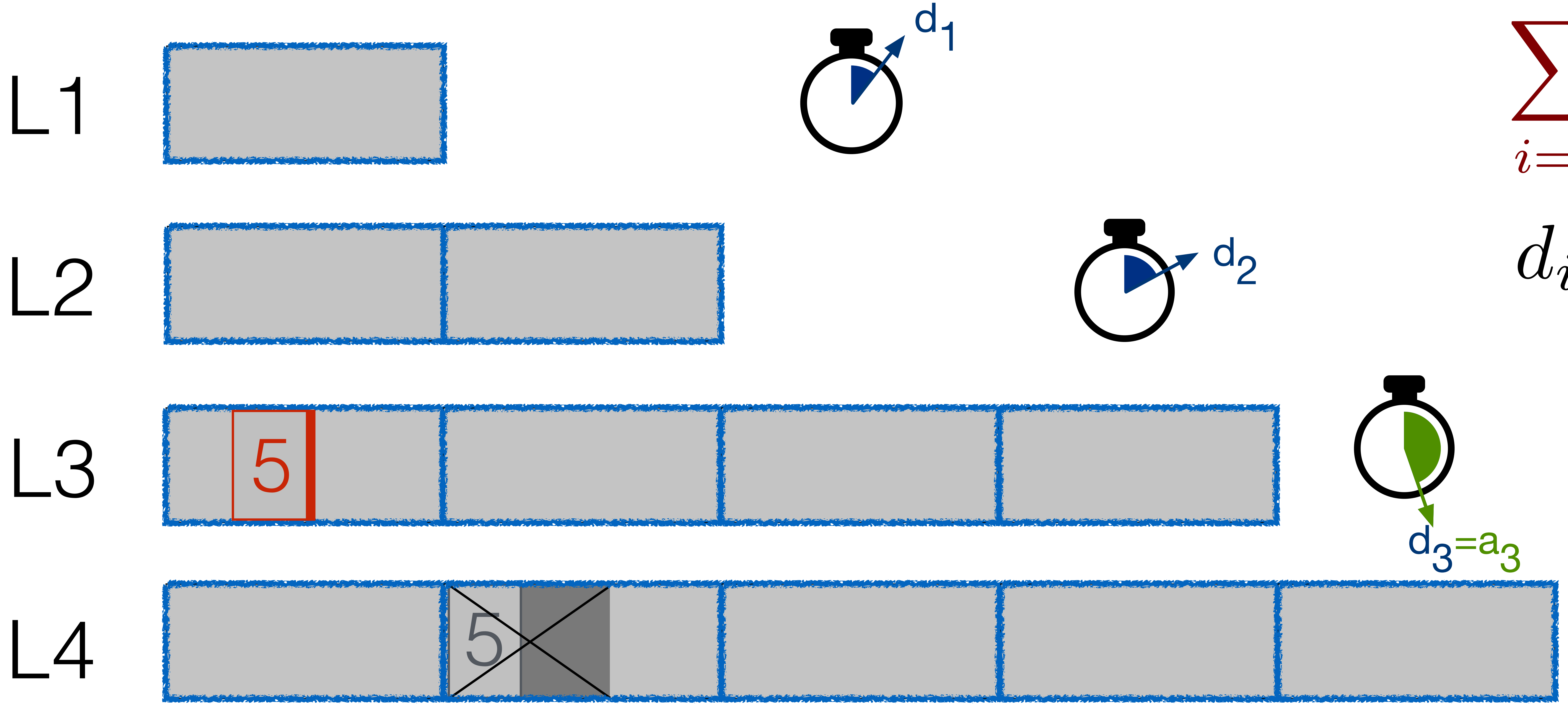


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

$$d_i = T \cdot d_{i-1}$$

FAst DElete

delete(5) within a threshold time: **D_th**

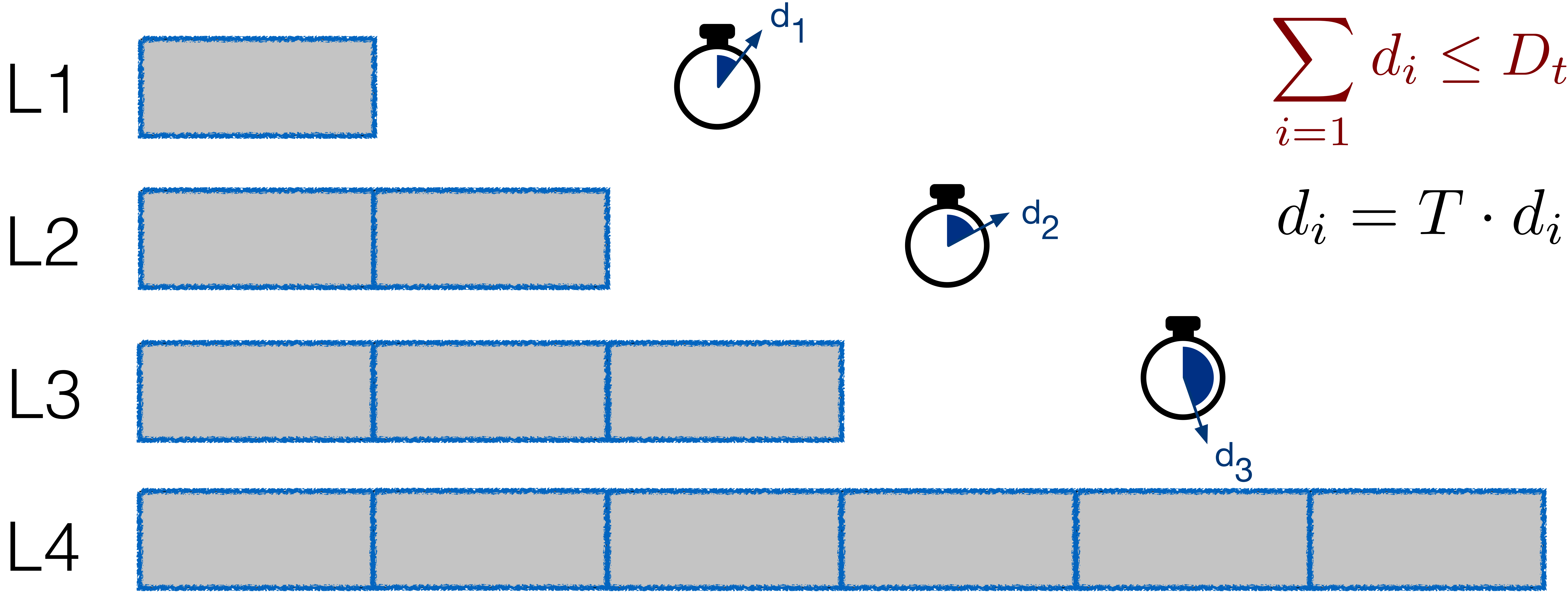


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

$$d_i = T \cdot d_{i-1}$$

FAst DElete

delete(5) within a threshold time: **D_{th}**

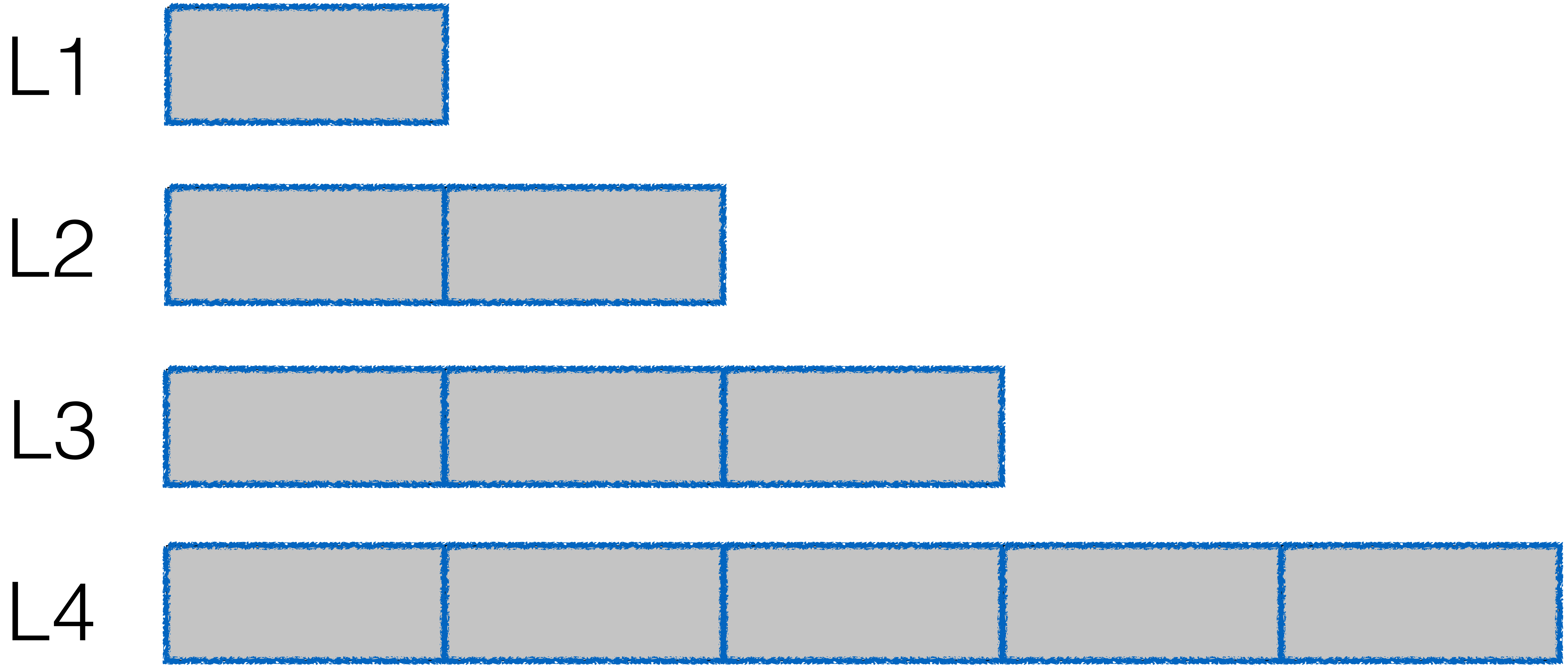


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

$$d_i = T \cdot d_{i-1}$$

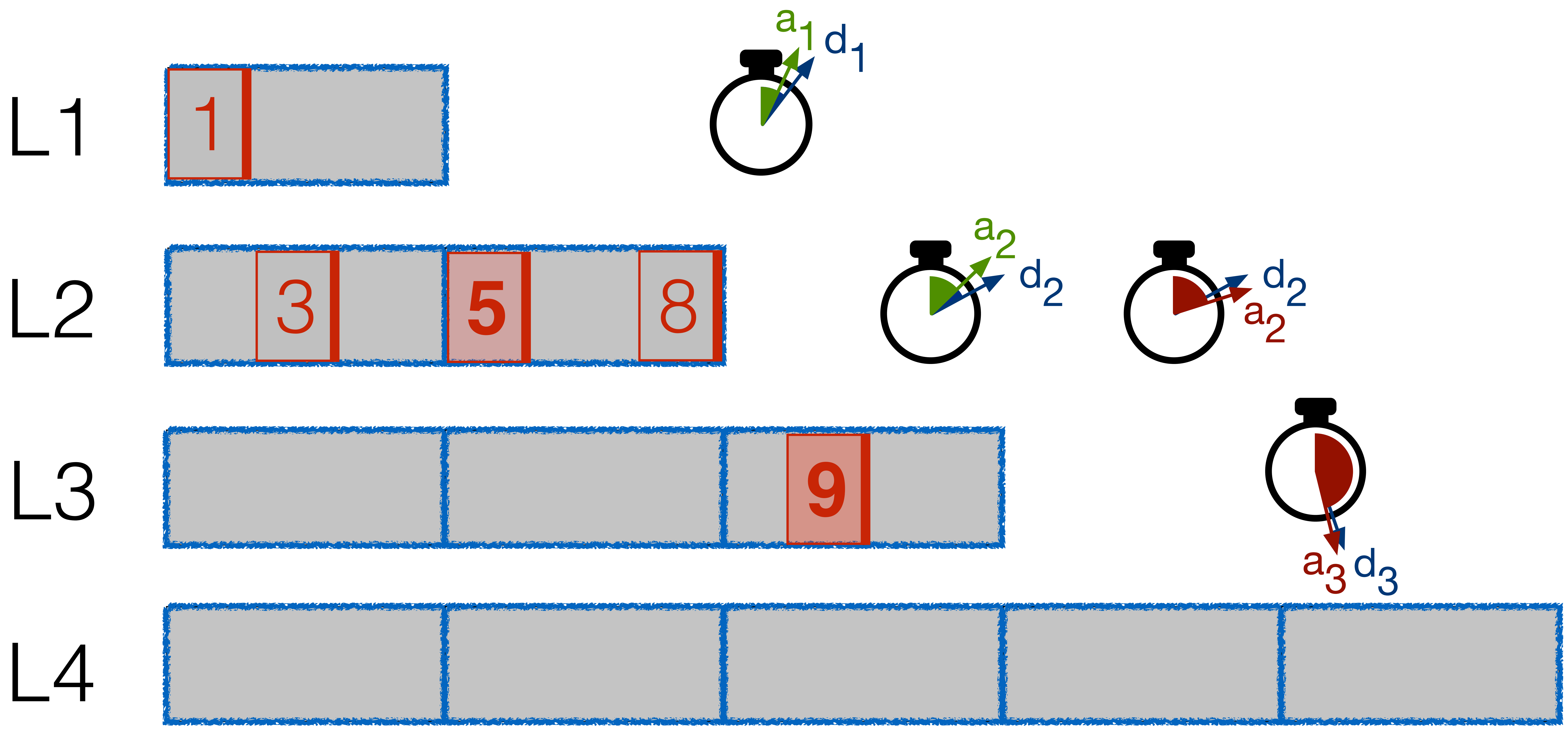
FAst DElete

breaking ties in practical workloads



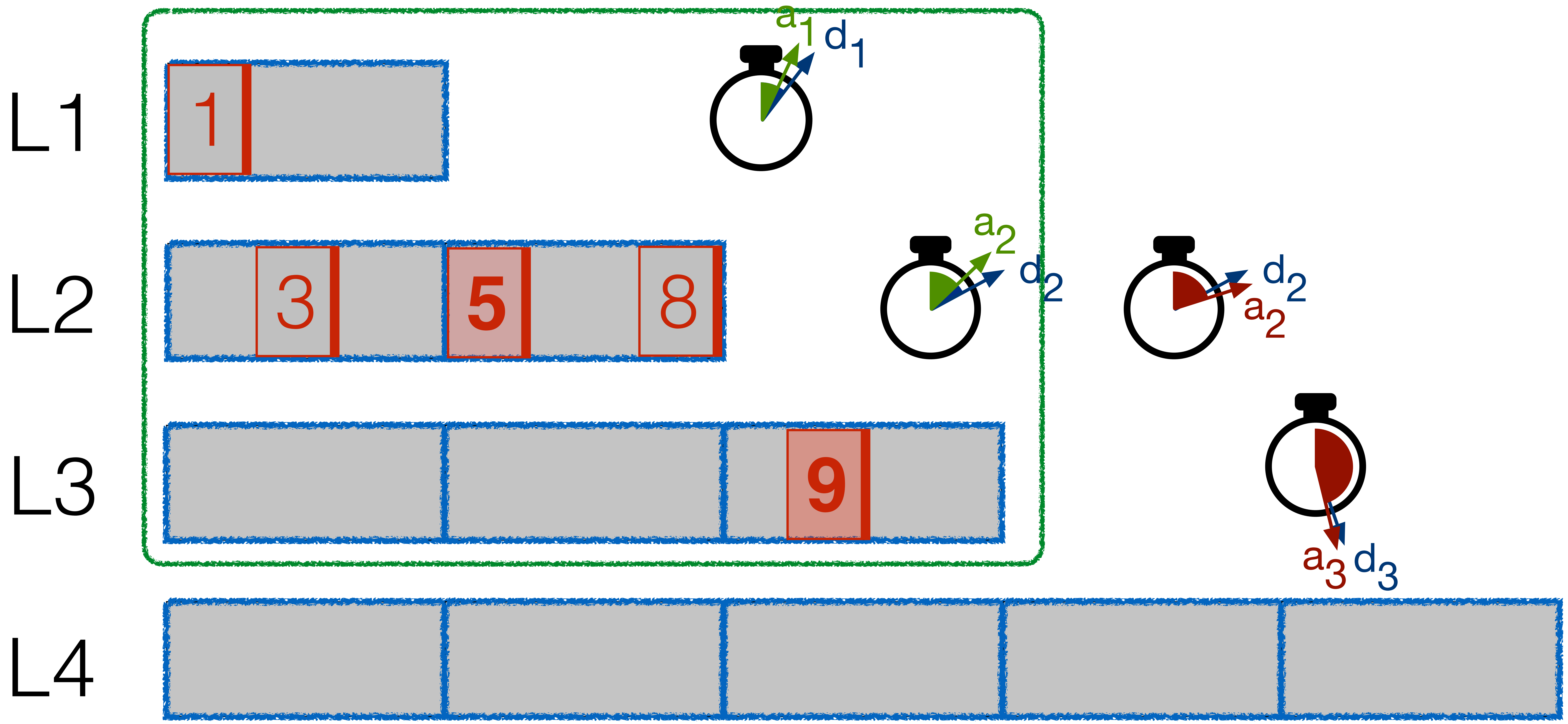
FAst DElete

breaking ties in practical workloads



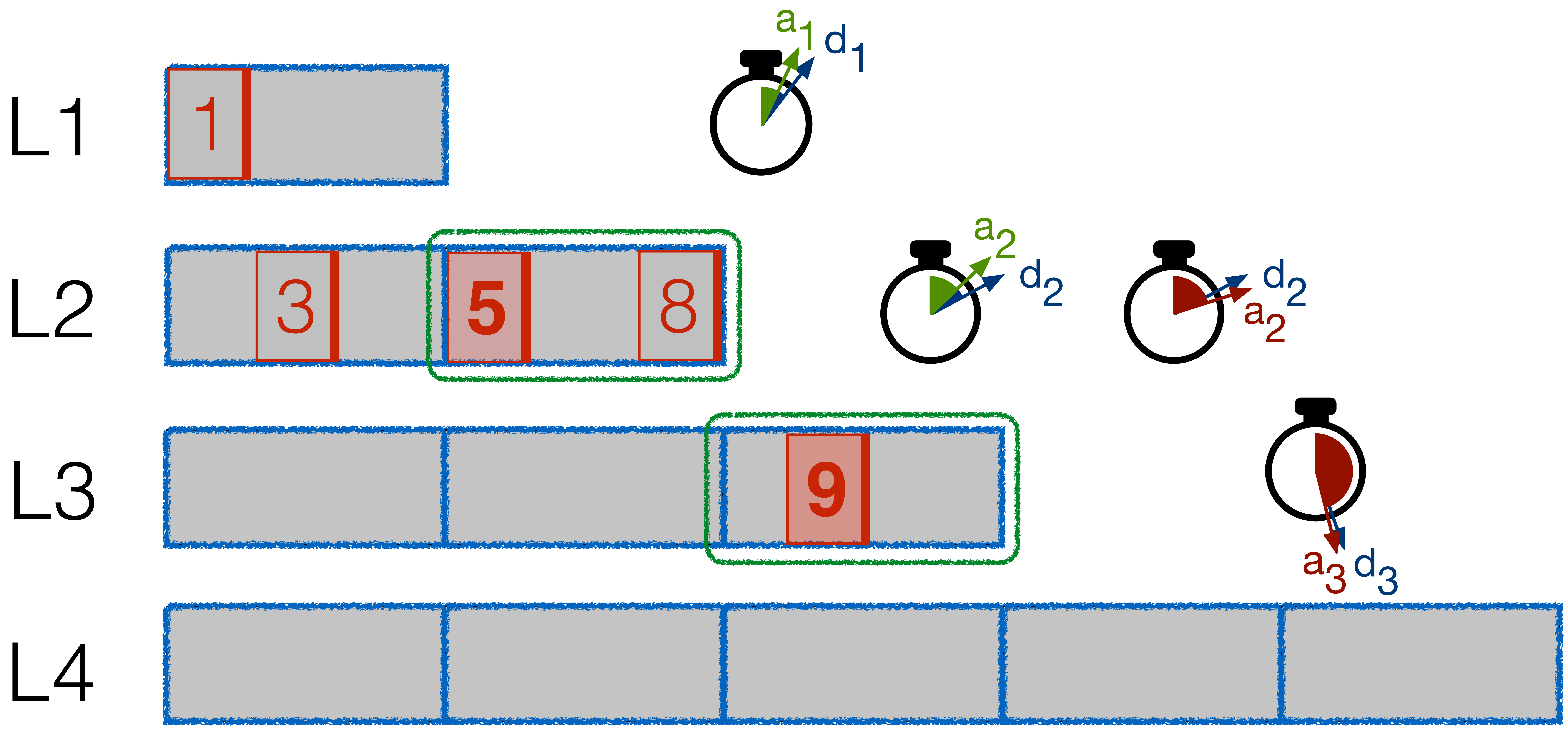
FAst DElete

breaking ties in practical workloads



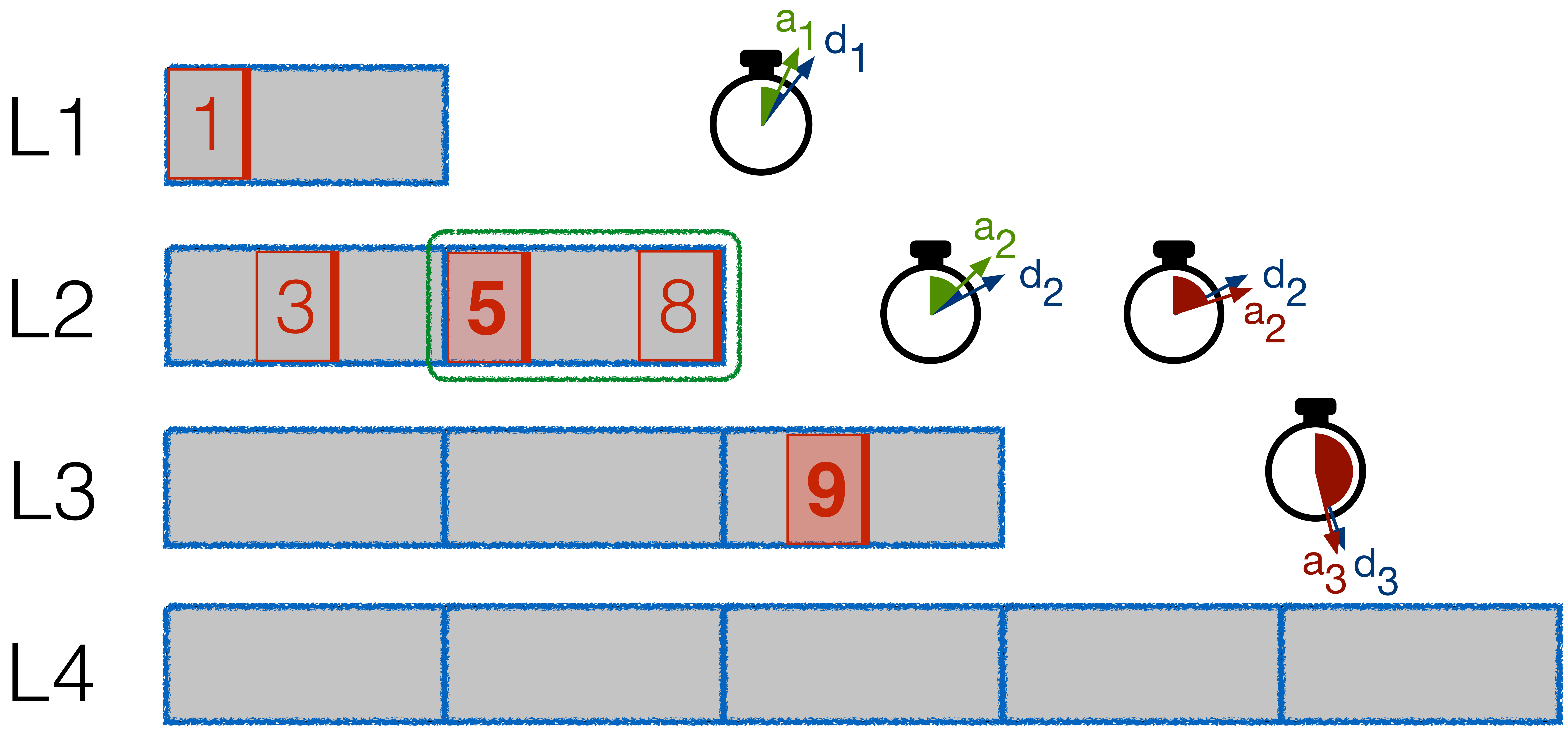
FAst DElete

breaking ties in practical workloads



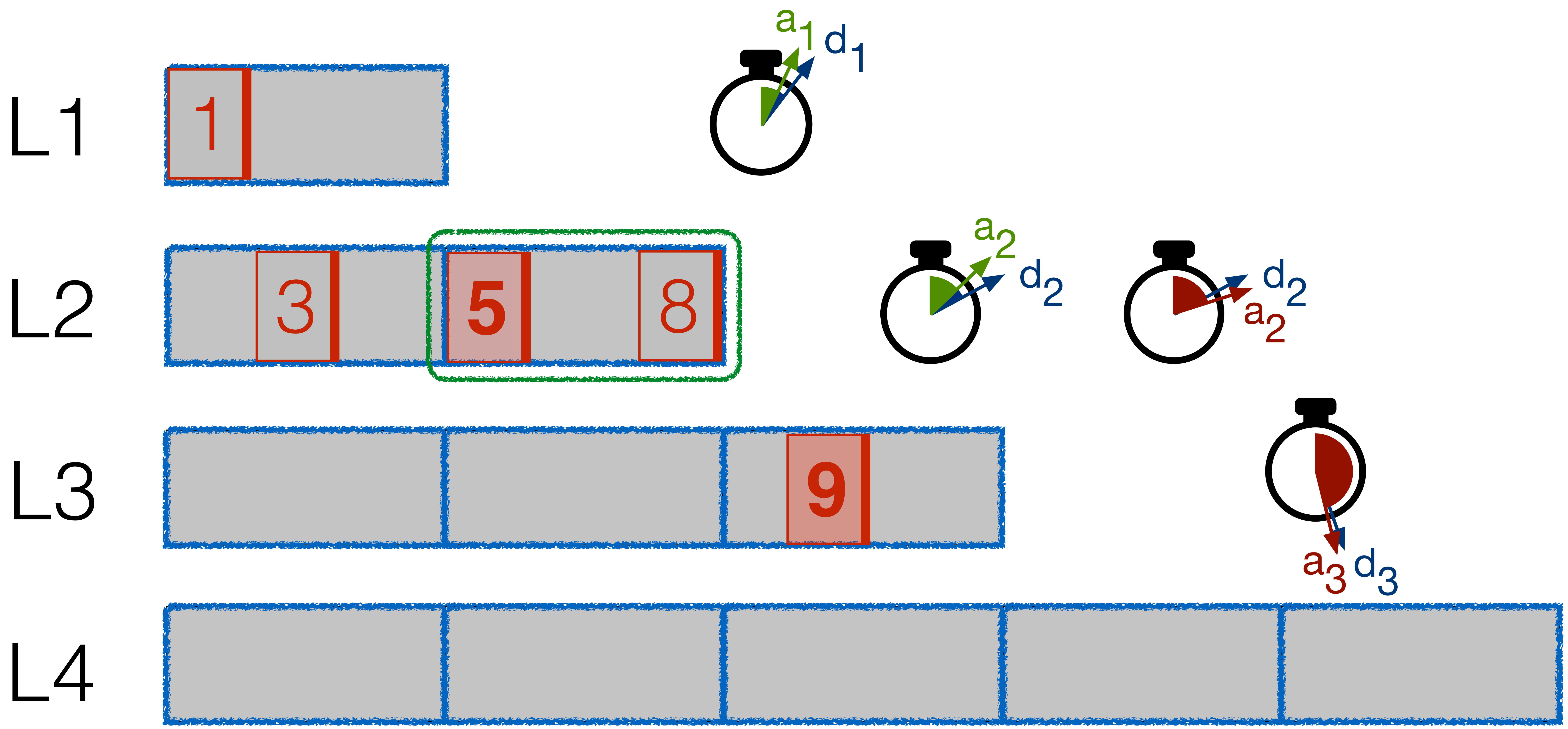
FAst DElete

breaking ties in practical workloads



FAst DElete

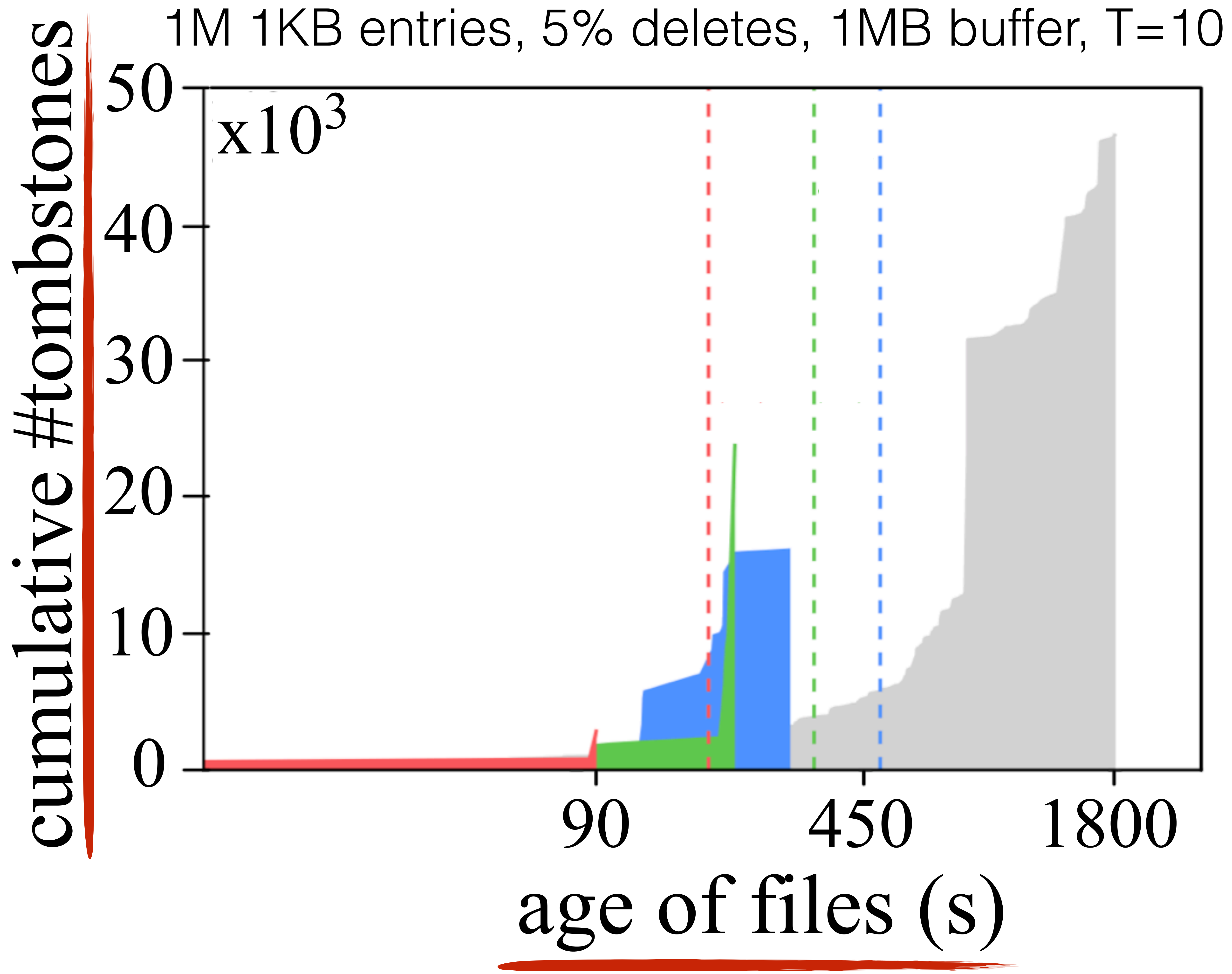
breaking ties in practical workloads



FAst DElete

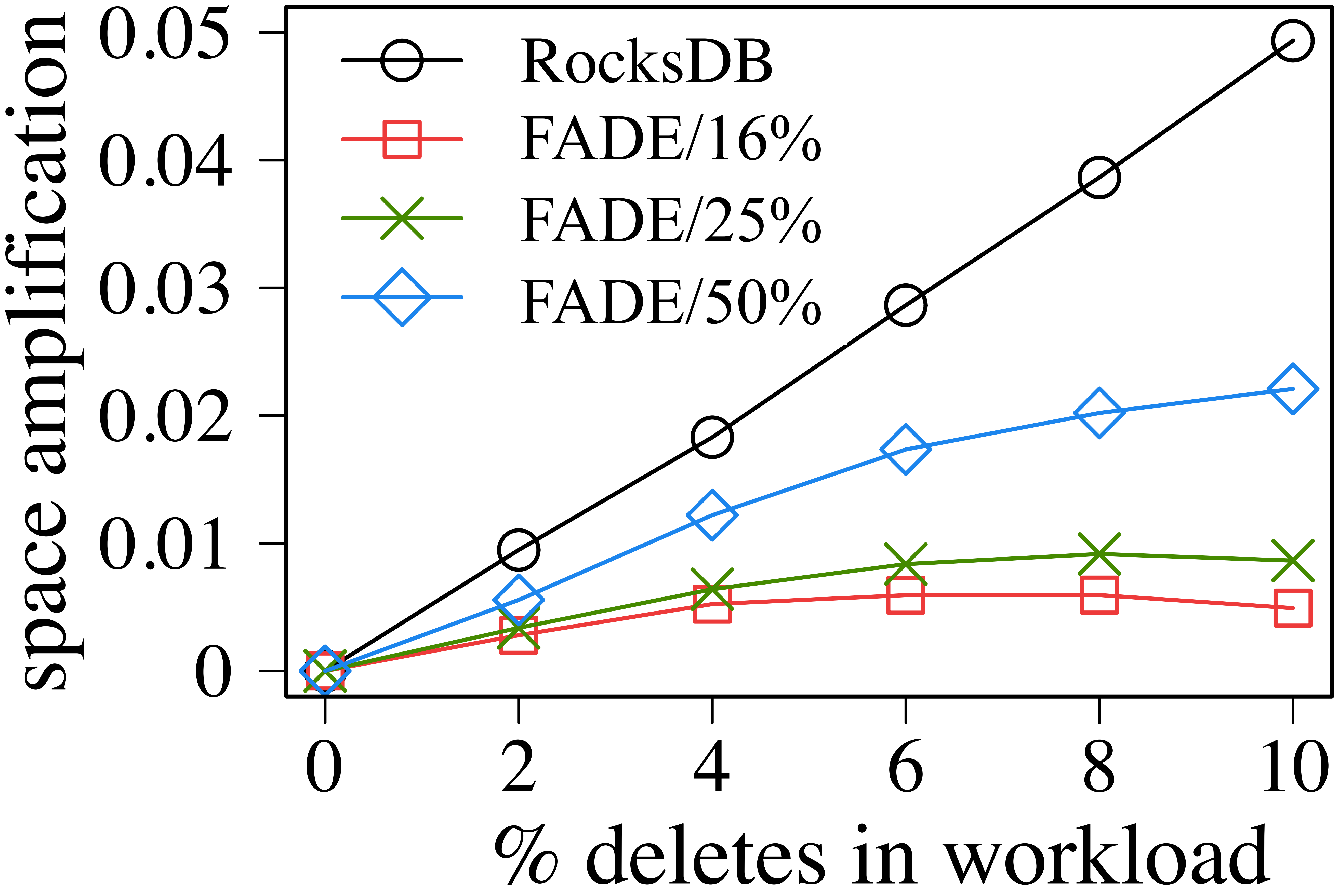
timely delete persistence ✓

within D_{th}



FAst DElete

1M 1KB entries, 1MB buffer, T=10



reduced space amplification ✓

2.1 - 9.8 x

timely delete persistence ✓

within D_{th}

FAst DElete

improved read performance ✓

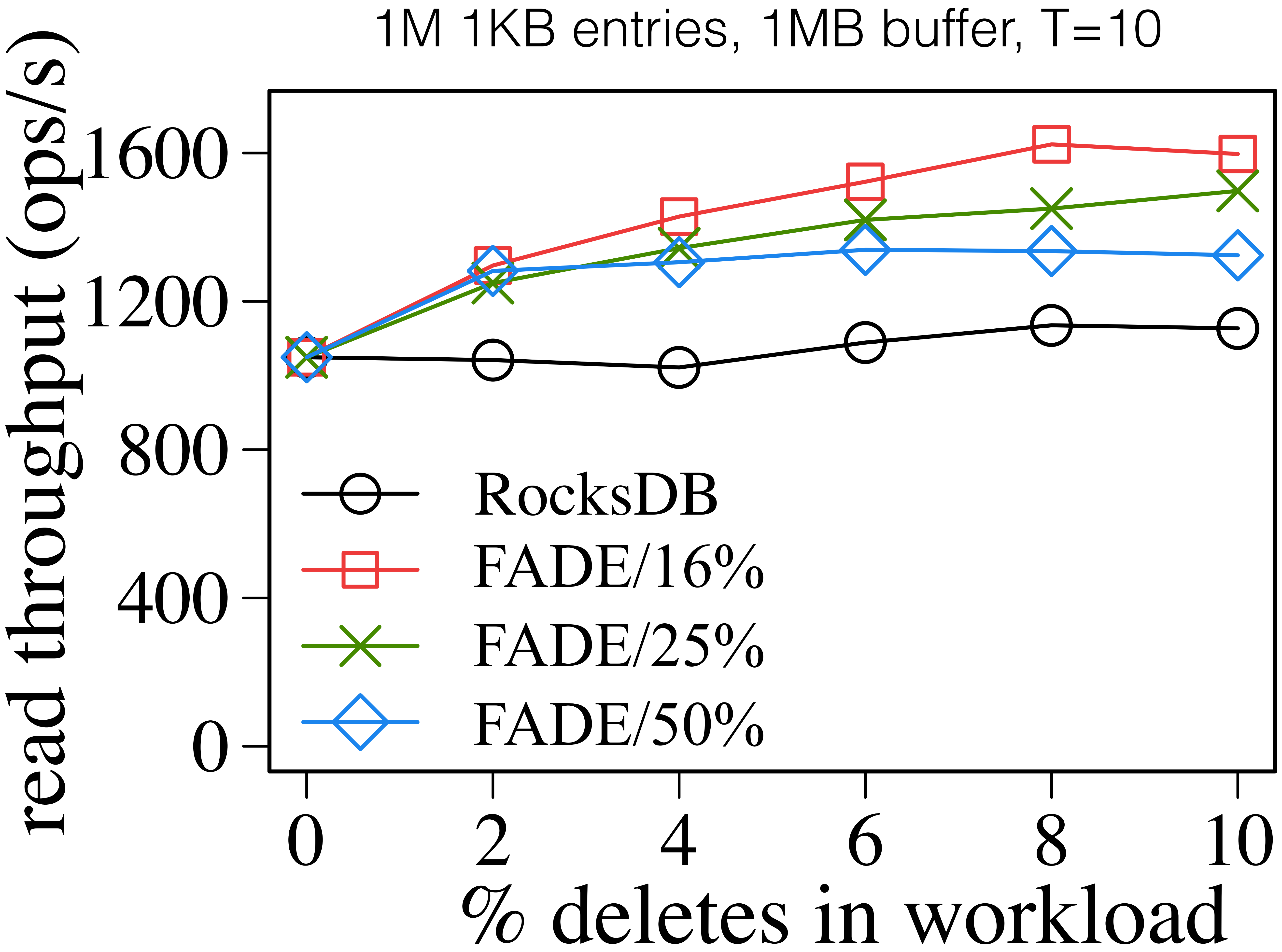
1.2 - 1.4 x

reduced space amplification ✓

2.1 - 9.8 x

timely delete persistence ✓

within D_{th}



FAst DElete

higher write amplification ↑

4 - 25 %

improved read performance ✓

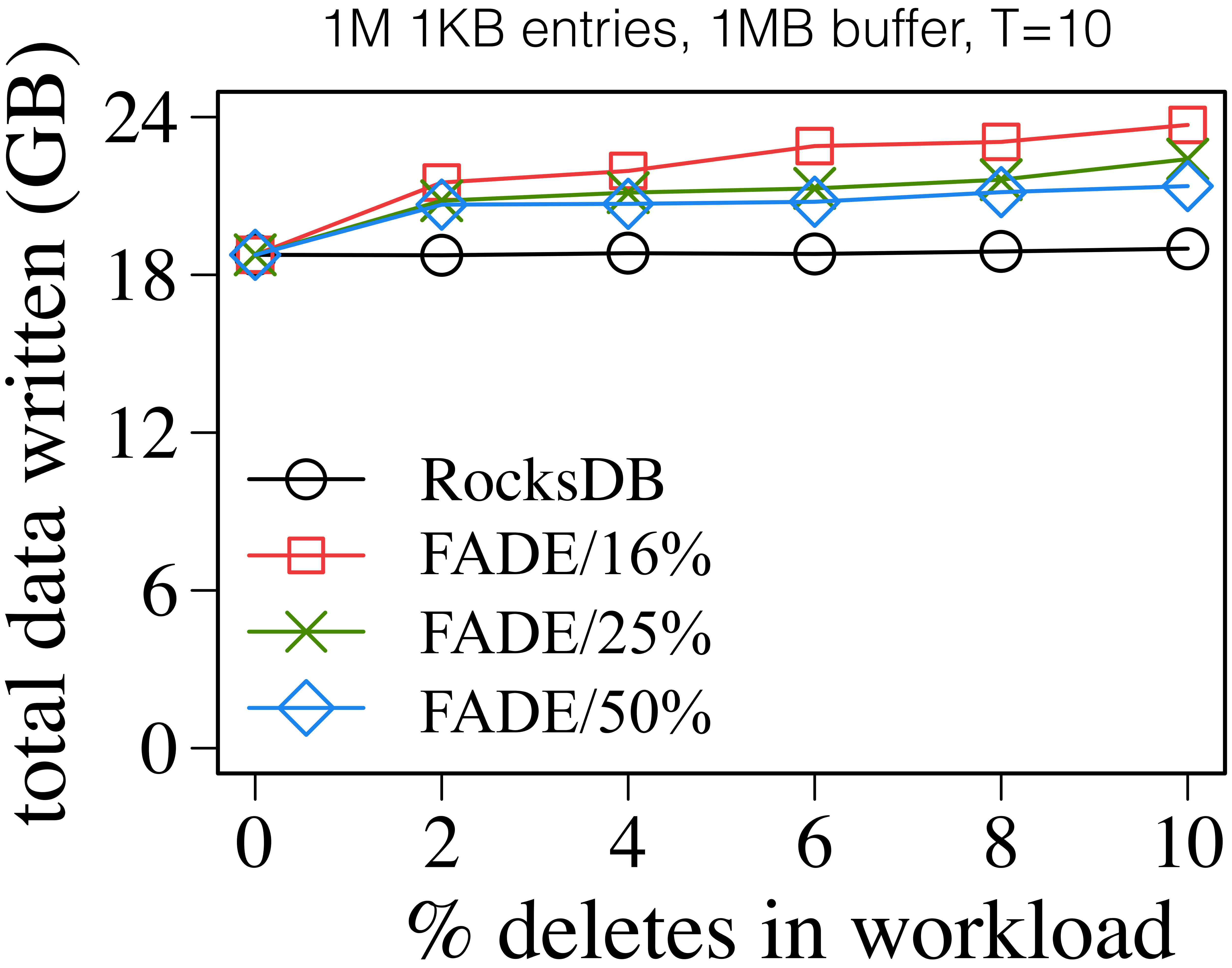
1.2 - 1.4 x

reduced space amplification ✓

2.1 - 9.8 x

timely delete persistence ✓

within D_{th}



FAst DElete

higher write amplification ↑

4 - 25 %

improved read performance ✓

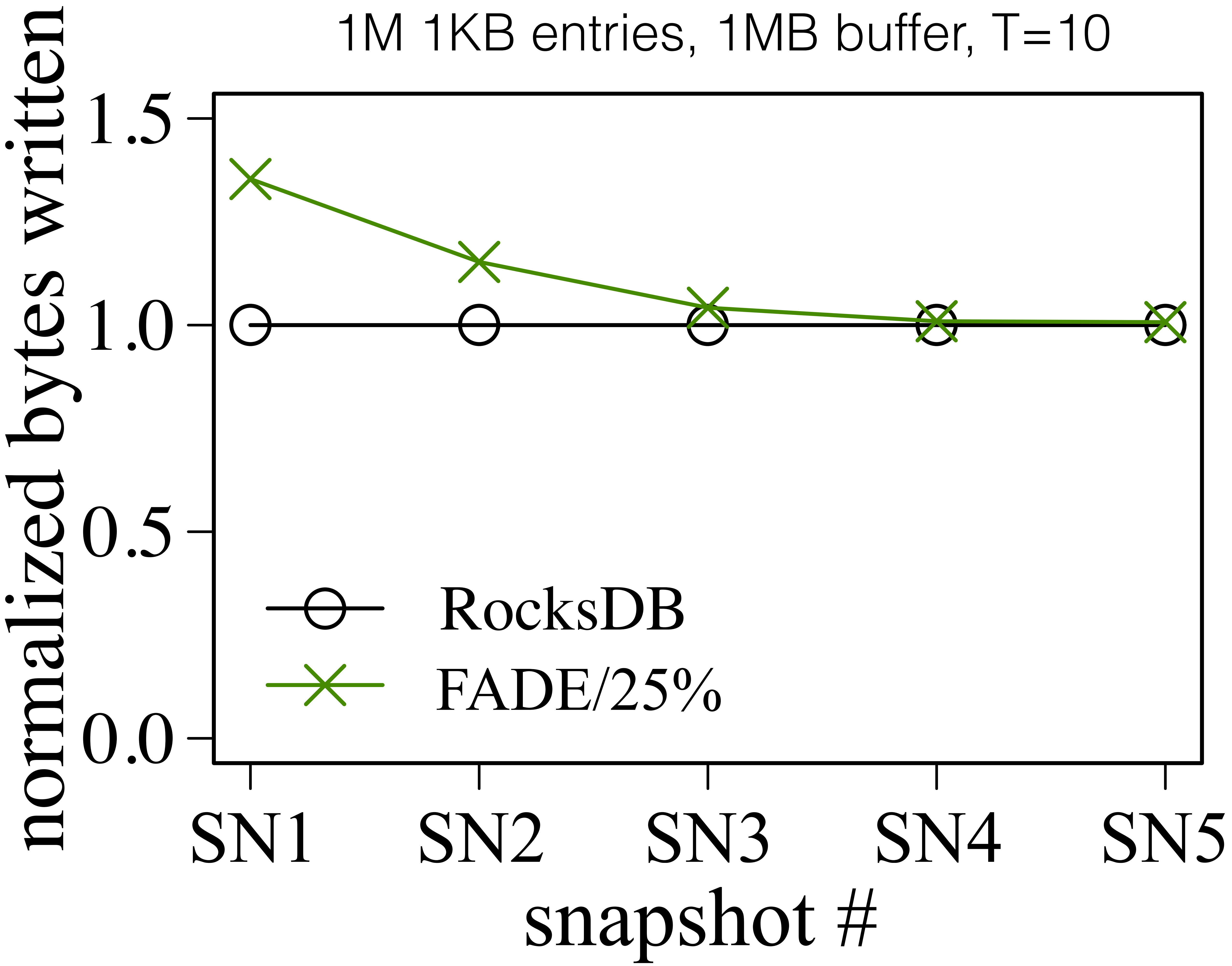
1.2 - 1.4 x

reduced space amplification ✓

2.1 - 9.8 x

timely delete persistence ✓

within D_{th}



FAst DElete

higher write amplification ●

0.7 %

improved read performance ✓

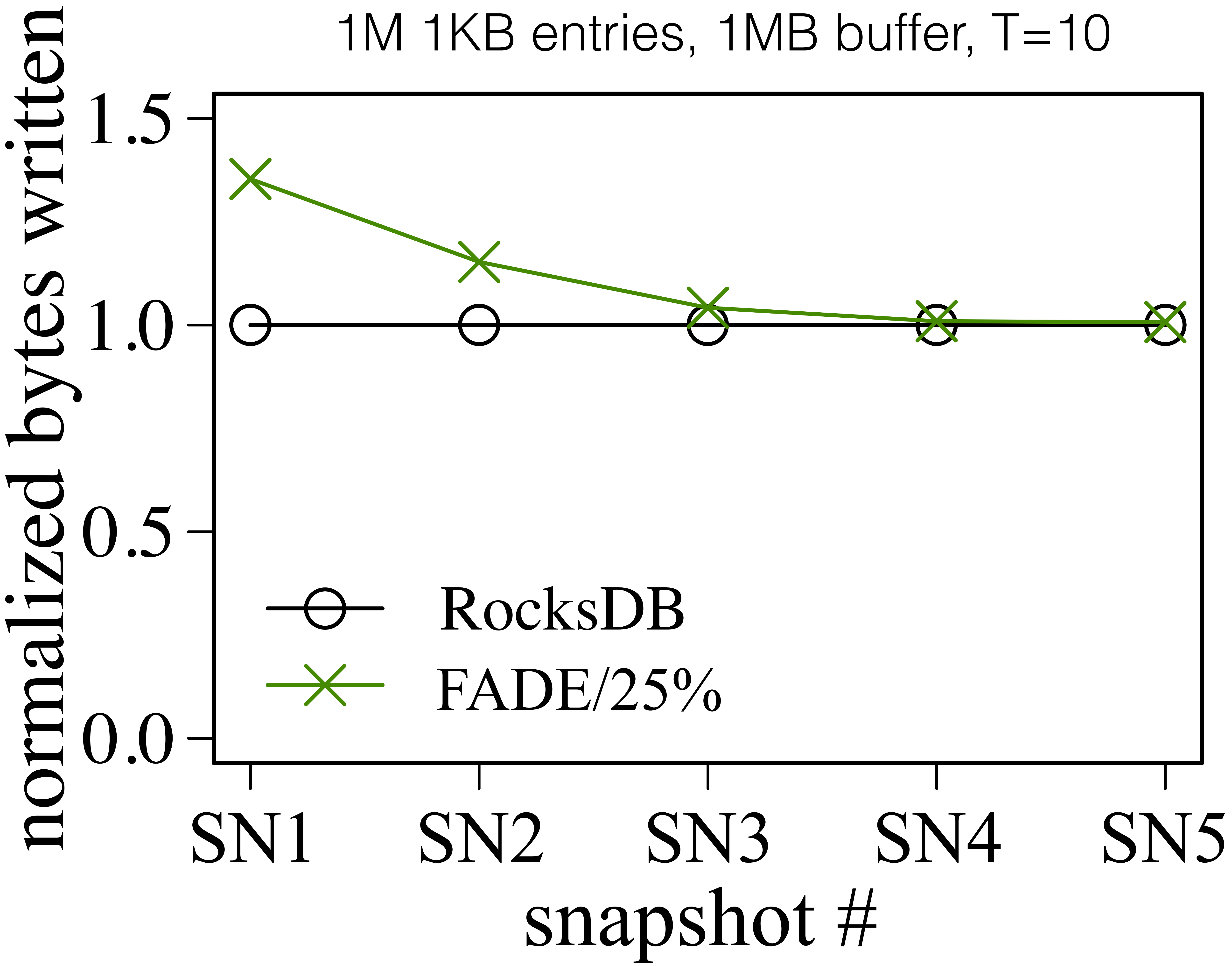
1.2 - 1.4 x

reduced space amplification ✓

2.1 - 9.8 x

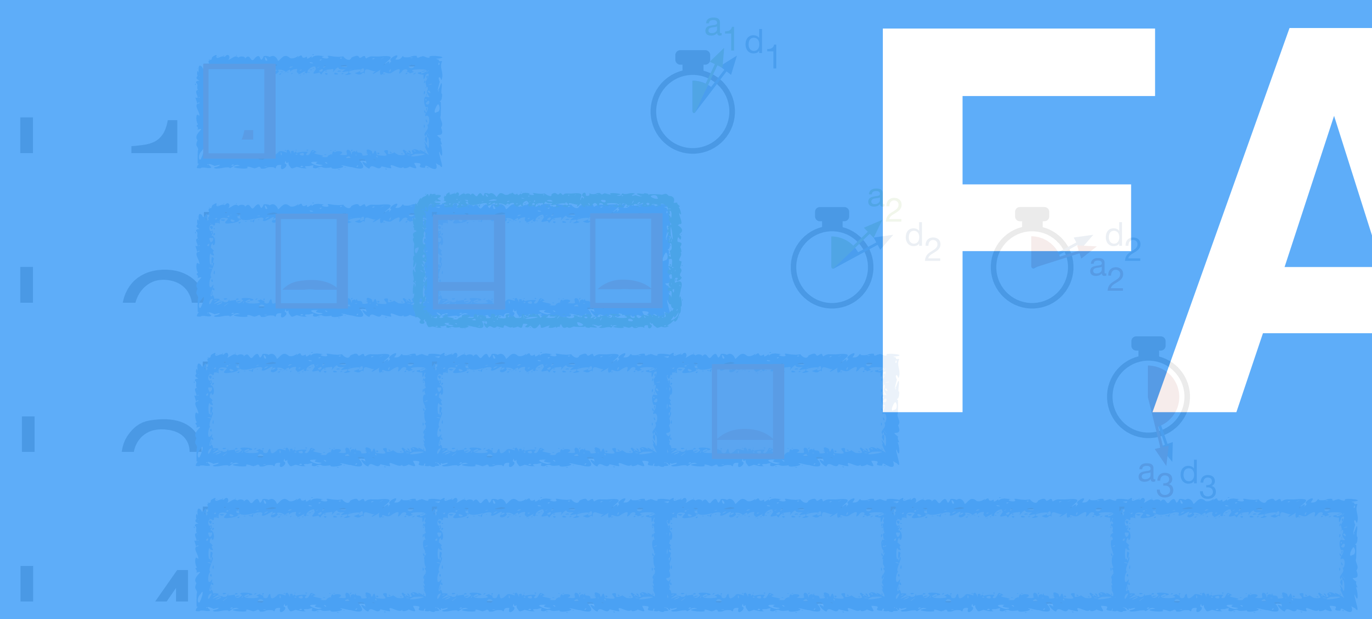
timely delete persistence ✓

within D_{th}



FADE

FAst DElete



higher write amplification ●

improved read performance ✓

reduced space amplification ✓

timely delete persistence ✓

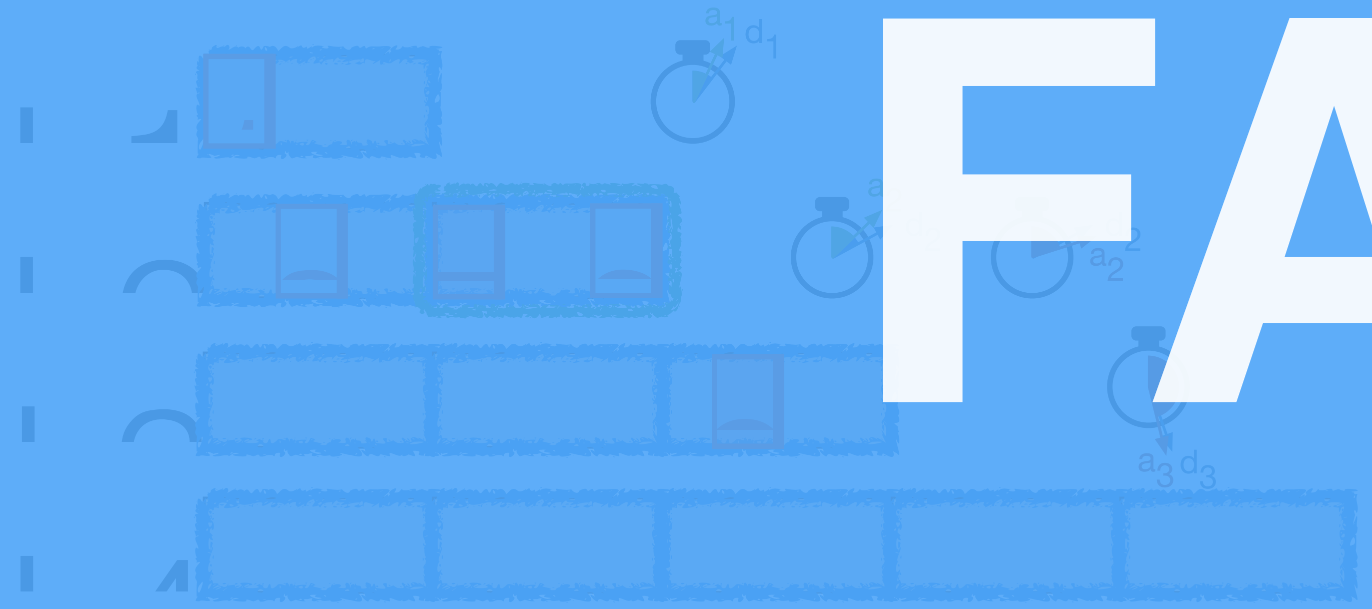
KiWi

latency spikes

superfluous I/Os

FADE

FAst DElete



higher write amplification ●

improved read performance ✓

reduced space amplification ✓

timely delete persistence ✓

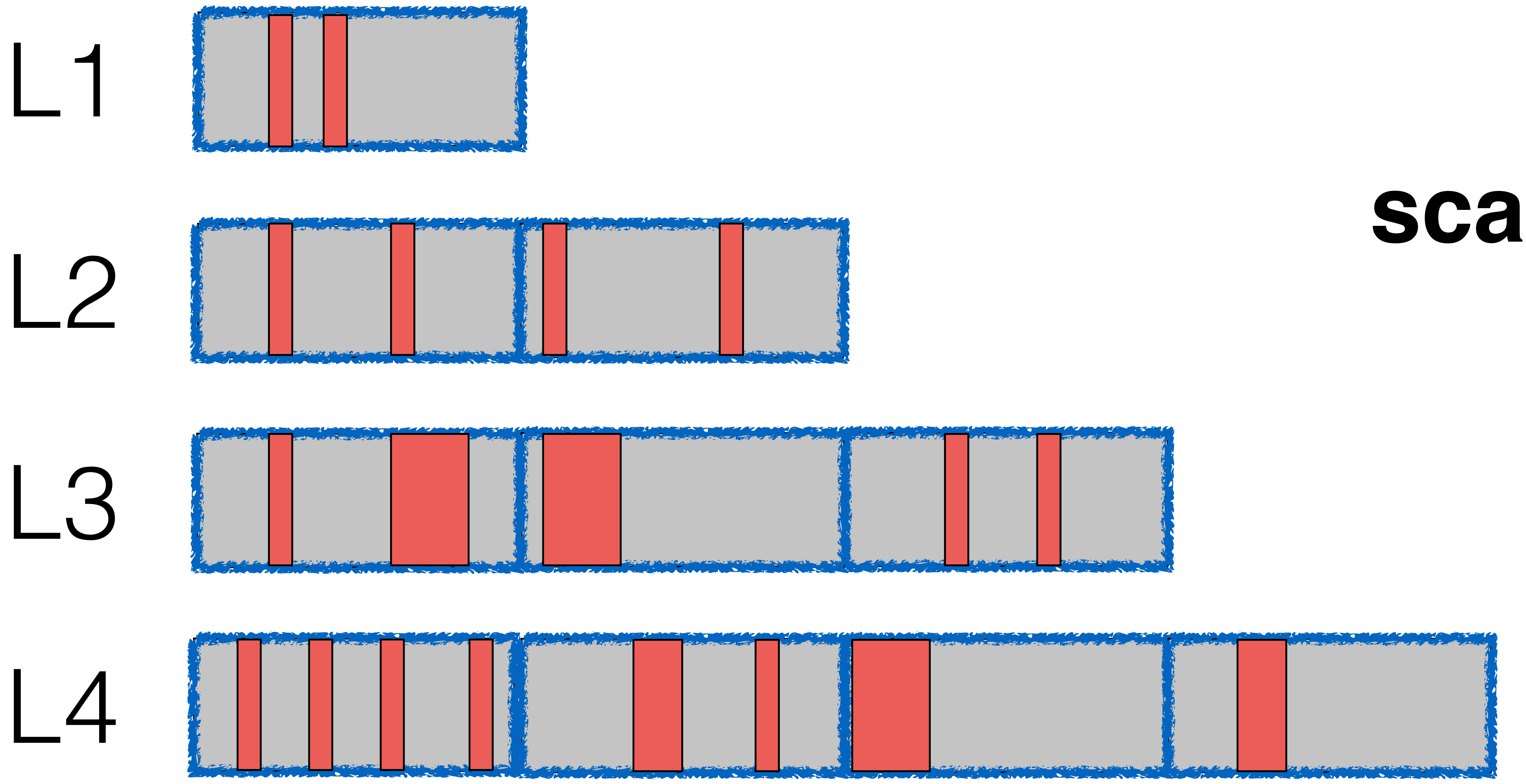
KiWi

latency spikes

superfluous I/Os

Key Weaving storage layout

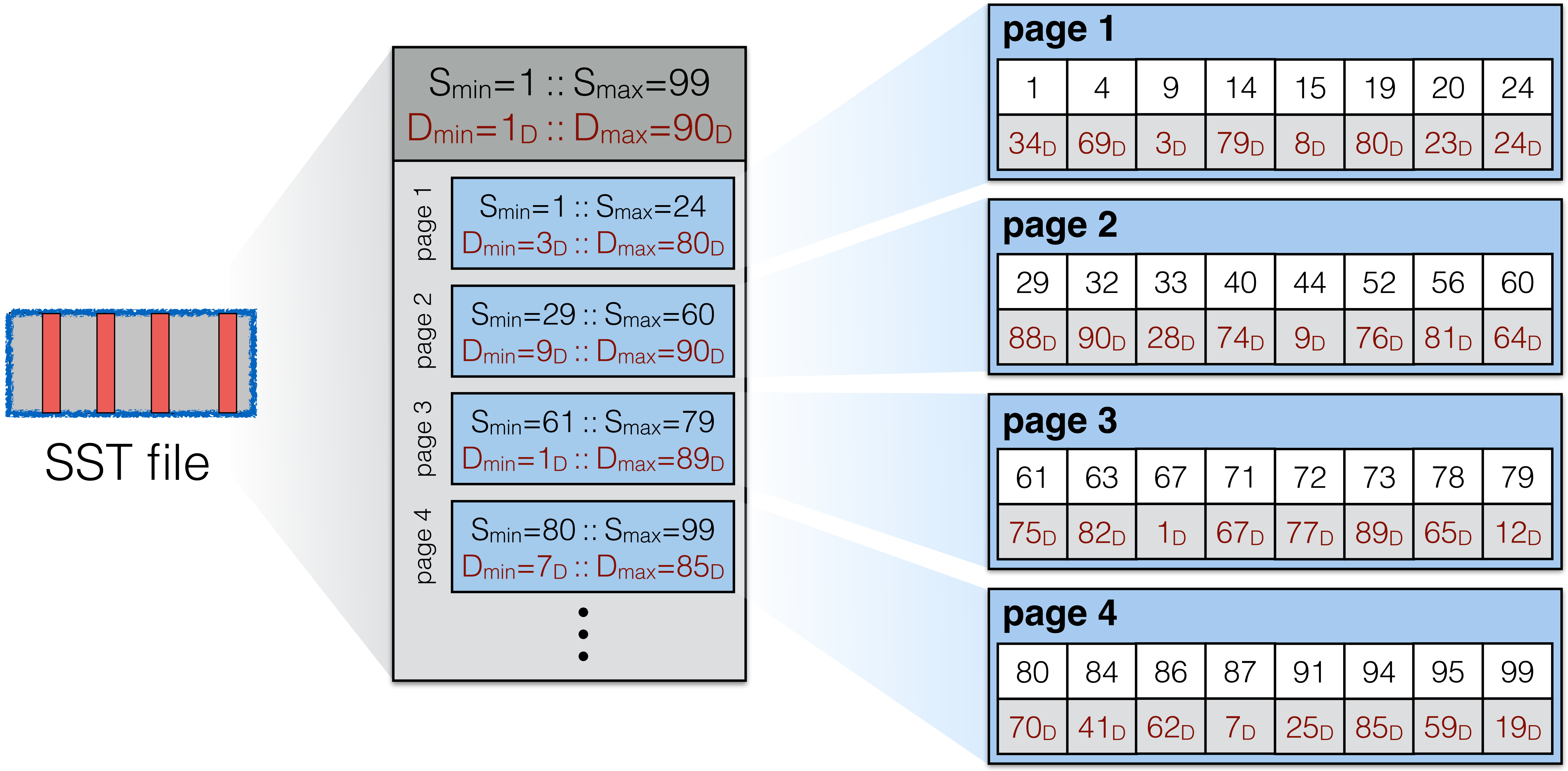
delete all entries older than: **D days**



scattered occurrences

Key Weaving storage layout

delete all entries with timestamp **<= 65_D**



$S_{min}=1$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=90_D$

page 1: $S_{min}=1$ :: $S_{max}=24$ / $D_{min}=3_D$ :: $D_{max}=80_D$

page 2: $S_{min}=29$ :: $S_{max}=60$ / $D_{min}=9_D$ :: $D_{max}=90_D$

page 3: $S_{min}=61$ :: $S_{max}=79$ / $D_{min}=1_D$ :: $D_{max}=89_D$

page 4: $S_{min}=80$ :: $S_{max}=99$ / $D_{min}=7_D$ :: $D_{max}=85_D$

⋮

page 1

1	4	9	14	15	19	20	24
34_D	69_D	3_D	79_D	8_D	80_D	23_D	24_D

page 2

29	32	33	40	44	52	56	60
88_D	90_D	28_D	74_D	9_D	76_D	81_D	64_D

page 3

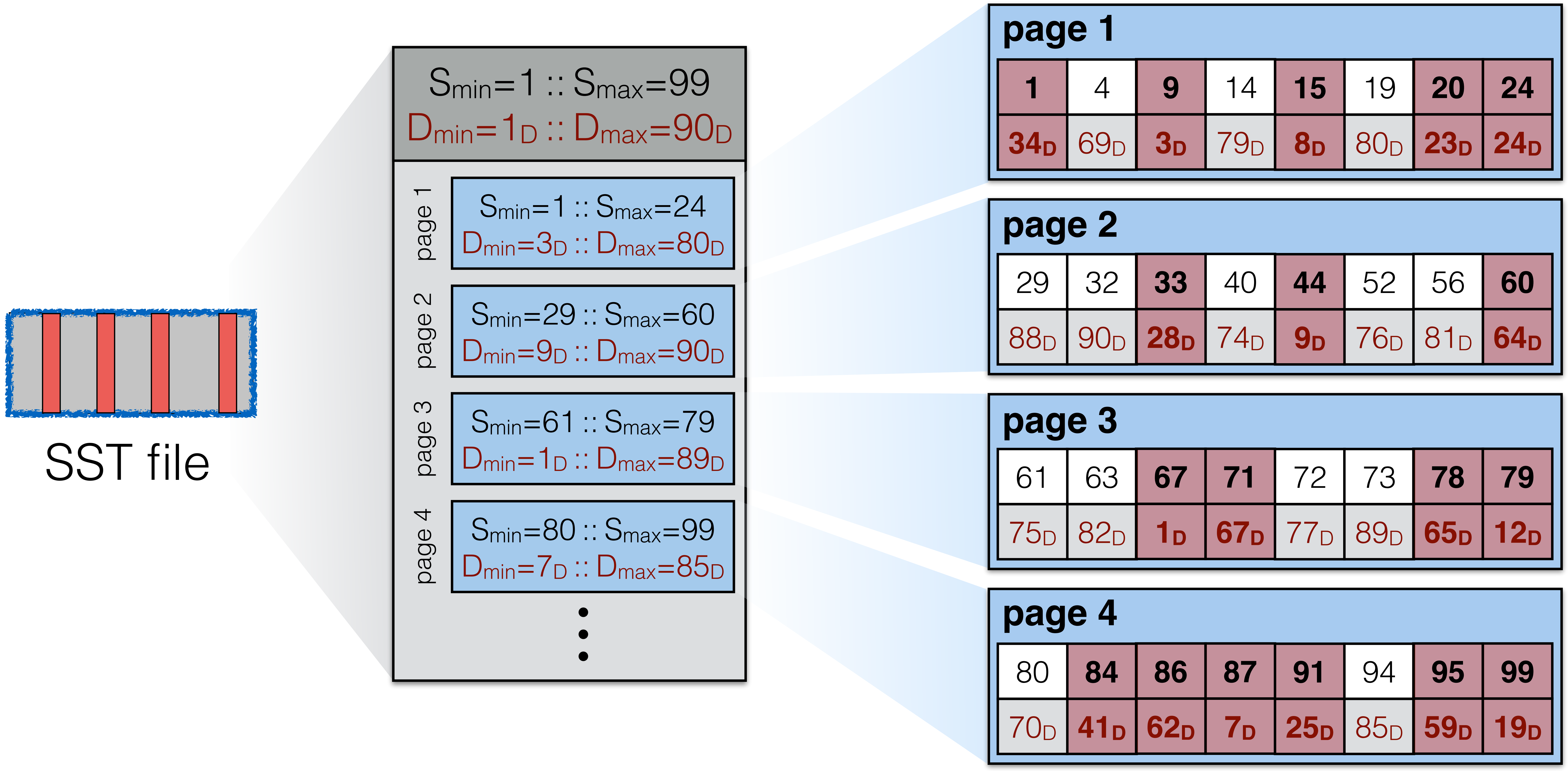
61	63	67	71	72	73	78	79
75_D	82_D	1_D	67_D	77_D	89_D	65_D	12_D

page 4

80	84	86	87	91	94	95	99
70_D	41_D	62_D	7_D	25_D	85_D	59_D	19_D

Key Weaving storage layout

delete all entries with timestamp **<= 65_D**



SST file
$S_{min}=1$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=90_D$
page 1
$S_{min}=1$ :: $S_{max}=24$
$D_{min}=3_D$ :: $D_{max}=80_D$
page 2
$S_{min}=29$ :: $S_{max}=60$
$D_{min}=9_D$ :: $D_{max}=90_D$
page 3
$S_{min}=61$ :: $S_{max}=79$
$D_{min}=1_D$ :: $D_{max}=89_D$
page 4
$S_{min}=80$ :: $S_{max}=99$
$D_{min}=7_D$ :: $D_{max}=85_D$
⋮

page 1

1	4	**9**	14	**15**	19	**20**	**24**
34_D	69_D	**3_D**	79_D	**8_D**	80_D	**23_D**	**24_D**

1 I/O

page 2

29	32	**33**	40	**44**	52	56	**60**
88_D	90_D	**28_D**	74_D	**9_D**	76_D	81_D	**64_D**

1 I/O

page 3

61	63	**67**	**71**	72	73	**78**	**79**
75_D	82_D	**1_D**	**67_D**	77_D	89_D	**65_D**	**12_D**

1 I/O

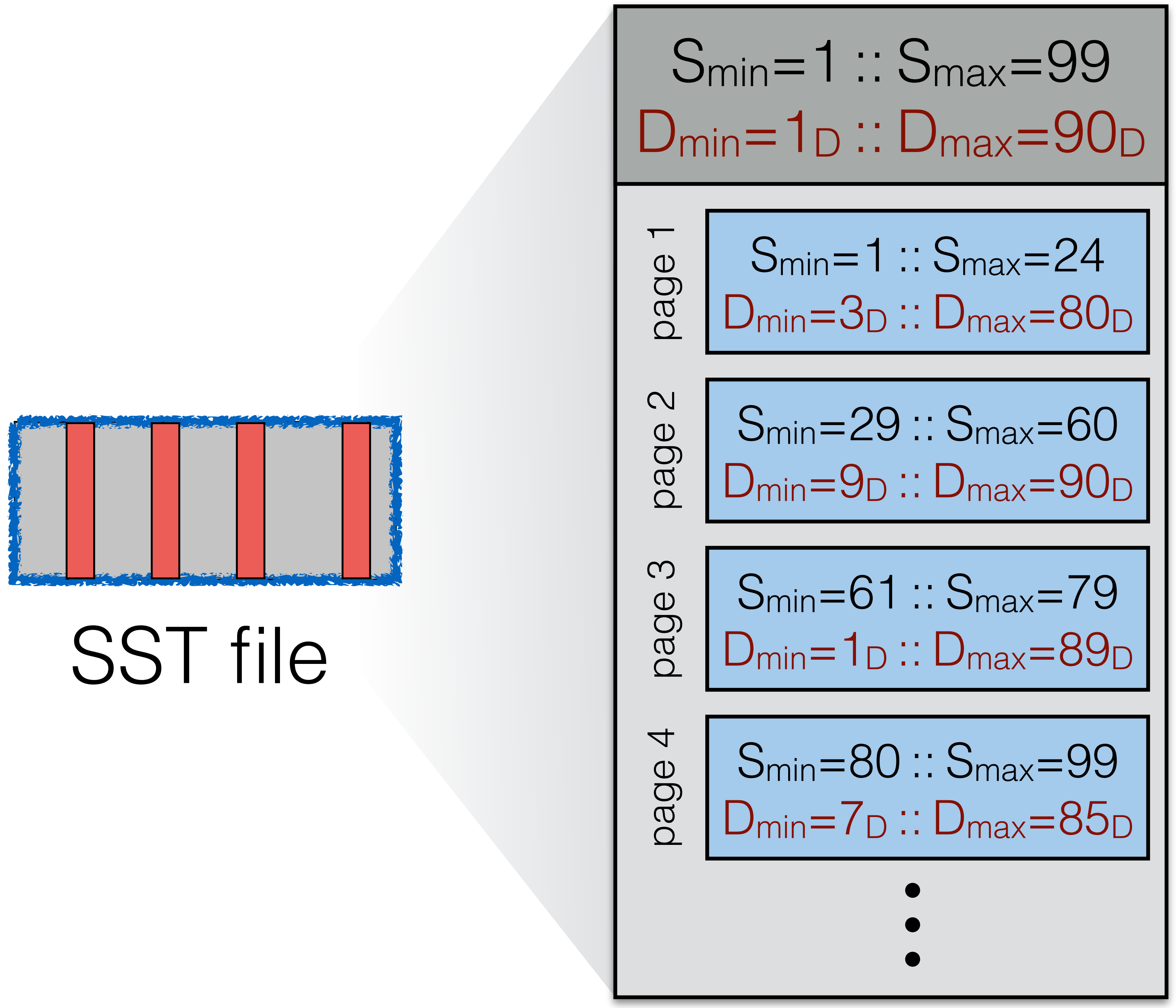
page 4

80	**84**	**86**	**87**	**91**	94	**95**	**99**
70_D	**41_D**	**62_D**	**7_D**	**25_D**	85_D	**59_D**	**19_D**

1 I/O

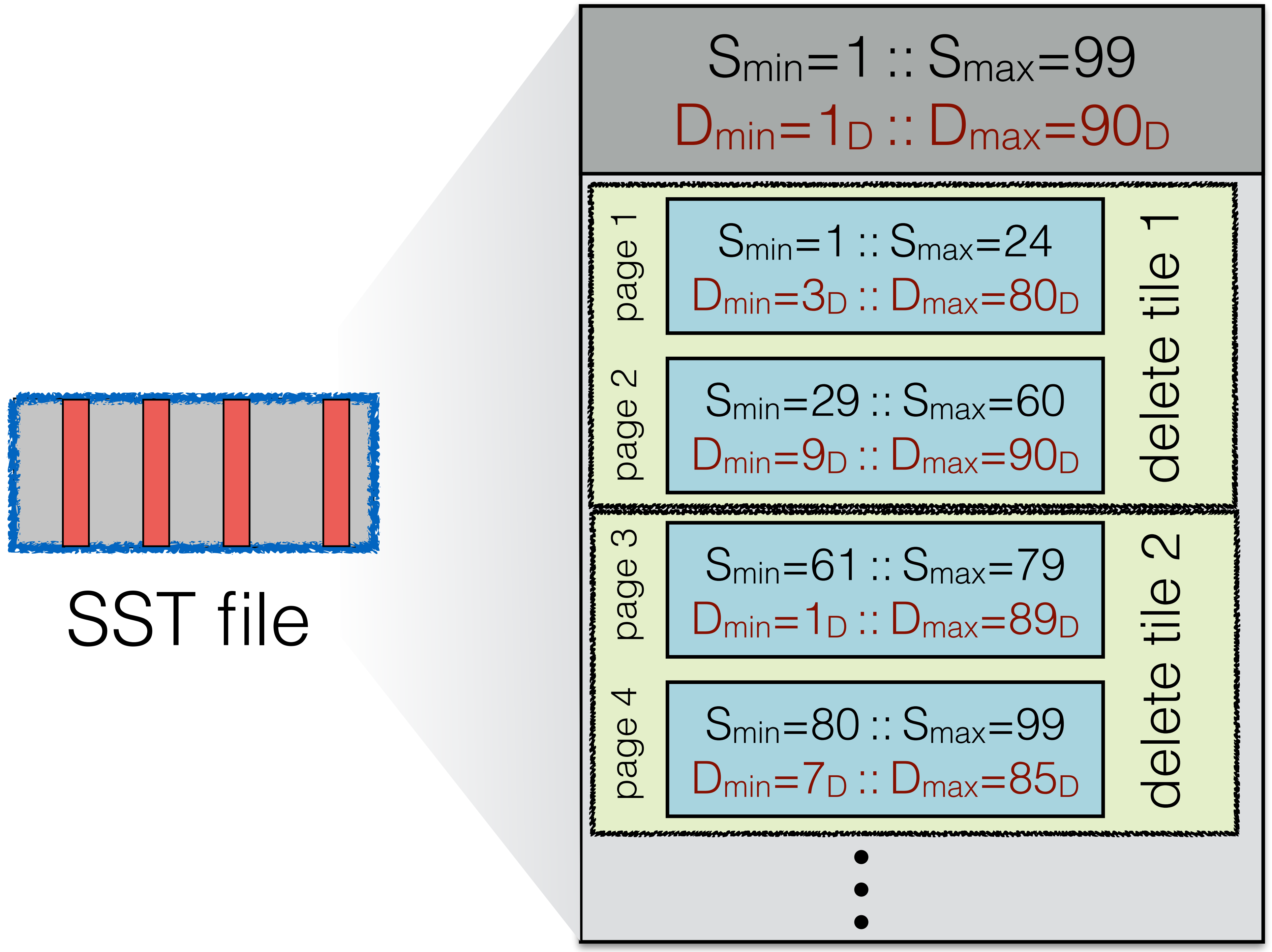
Key Weaving storage layout

delete all entries with timestamp **<= 65$_D$**



Key Weaving storage layout

delete all entries with timestamp **<= 65$_D$**

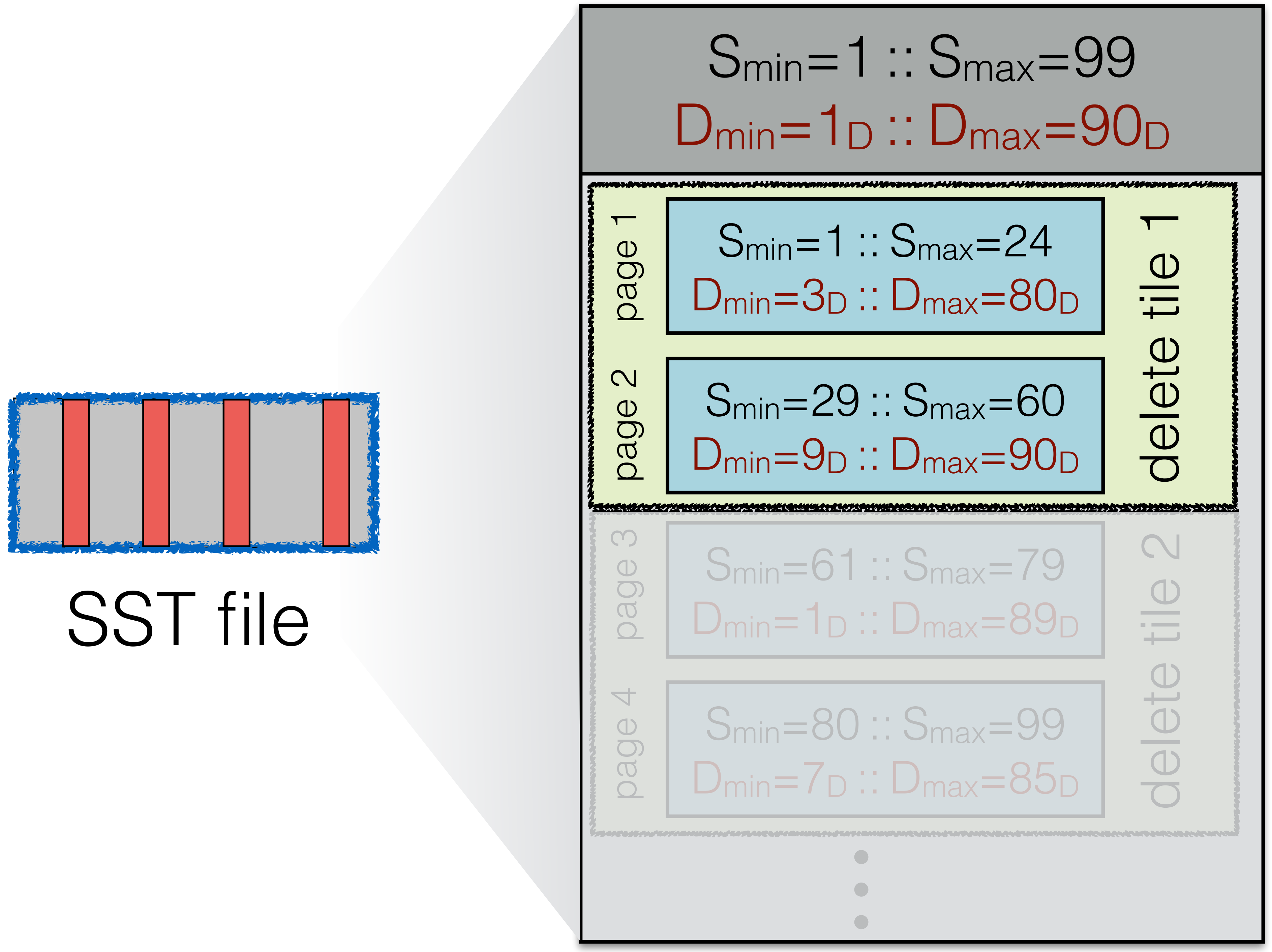


SST file
S_{min}=1 :: S_{max}=99
D_{min}=1$_D$ :: D_{max}=90$_D$
page 1
S_{min}=1 :: S_{max}=24
D_{min}=3$_D$ :: D_{max}=80$_D$
page 2
S_{min}=29 :: S_{max}=60
D_{min}=9$_D$ :: D_{max}=90$_D$
delete tile 1
page 3
S_{min}=61 :: S_{max}=79
D_{min}=1$_D$ :: D_{max}=89$_D$
page 4
S_{min}=80 :: S_{max}=99
D_{min}=7$_D$ :: D_{max}=85$_D$
delete tile 2

partitioned on S

Key Weaving storage layout

delete all entries with timestamp **<= 65_D**

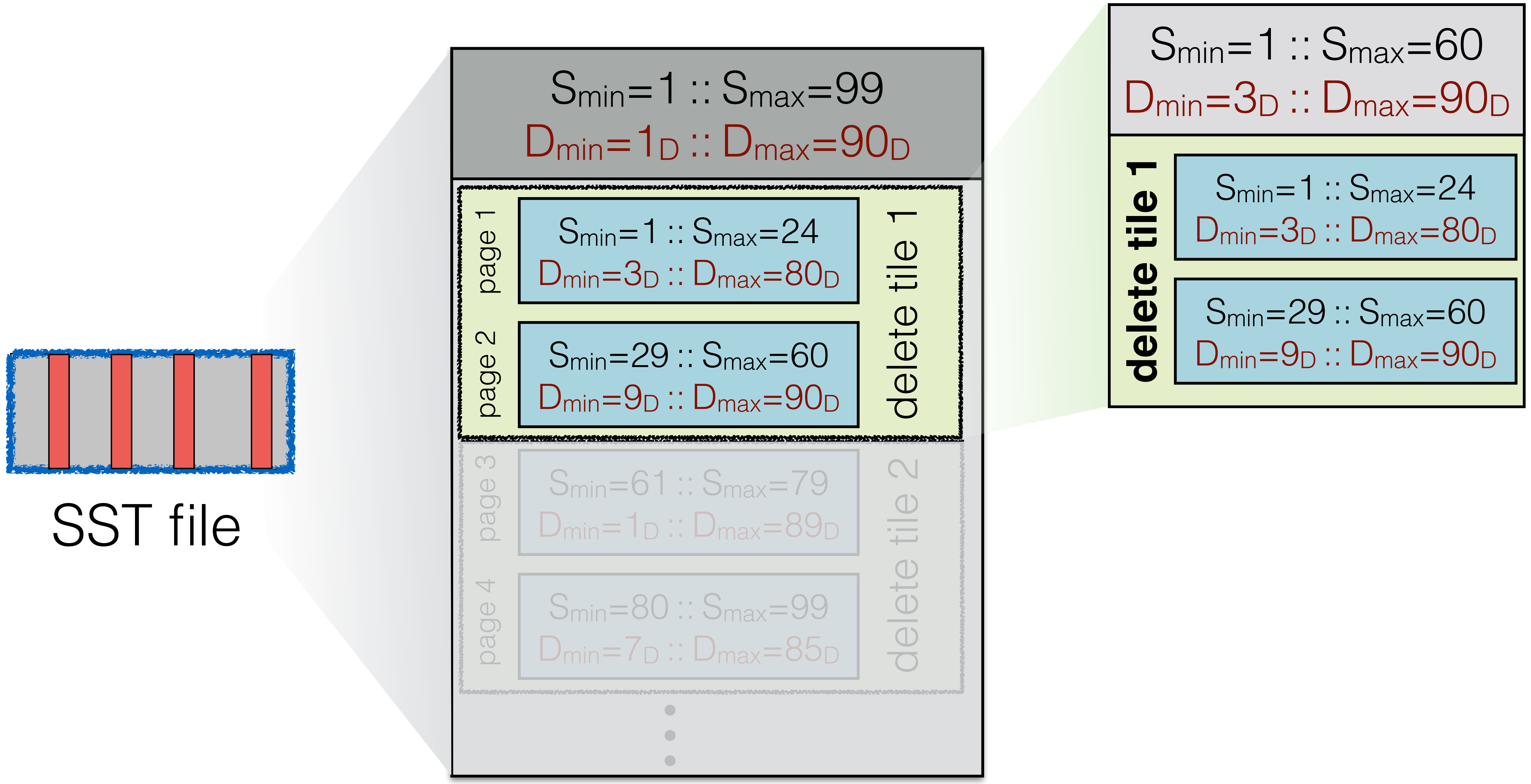


$S_{min}=1$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=90_D$
page 1
$S_{min}=1$ :: $S_{max}=24$
$D_{min}=3_D$ :: $D_{max}=80_D$
page 2
$S_{min}=29$ :: $S_{max}=60$
$D_{min}=9_D$ :: $D_{max}=90_D$
delete tile 1
page 3
$S_{min}=61$ :: $S_{max}=79$
$D_{min}=1_D$ :: $D_{max}=89_D$
page 4
$S_{min}=80$ :: $S_{max}=99$
$D_{min}=7_D$ :: $D_{max}=85_D$
delete tile 2
SST file

partitioned on S

Key Weaving storage layout

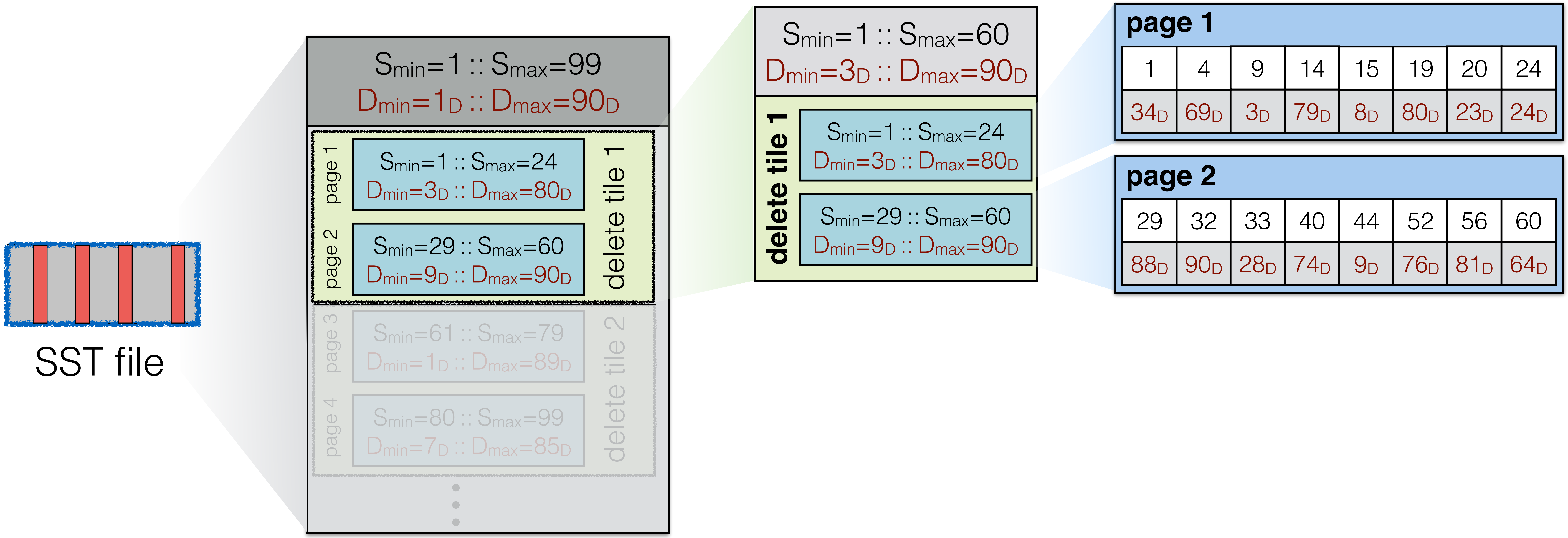
delete all entries with timestamp **<= 65$_D$**



partitioned on S

Key Weaving storage layout

delete all entries with timestamp **<= 65$_D$**


SST file
$S_{min}=1 :: S_{max}=99$
$D_{min}=1_D :: D_{max}=90_D$
page 1
$S_{min}=1 :: S_{max}=24$
$D_{min}=3_D :: D_{max}=80_D$
page 2
$S_{min}=29 :: S_{max}=60$
$D_{min}=9_D :: D_{max}=90_D$
delete tile 1
page 3
$S_{min}=61 :: S_{max}=79$
$D_{min}=1_D :: D_{max}=89_D$
page 4
$S_{min}=80 :: S_{max}=99$
$D_{min}=7_D :: D_{max}=85_D$
delete tile 2
$S_{min}=1 :: S_{max}=60$
$D_{min}=3_D :: D_{max}=90_D$
delete tile 1
$S_{min}=1 :: S_{max}=24$
$D_{min}=3_D :: D_{max}=80_D$
$S_{min}=29 :: S_{max}=60$
$D_{min}=9_D :: D_{max}=90_D$

page 1

1	4	9	14	15	19	20	24
34$_D$	69$_D$	3$_D$	79$_D$	8$_D$	80$_D$	23$_D$	24$_D$

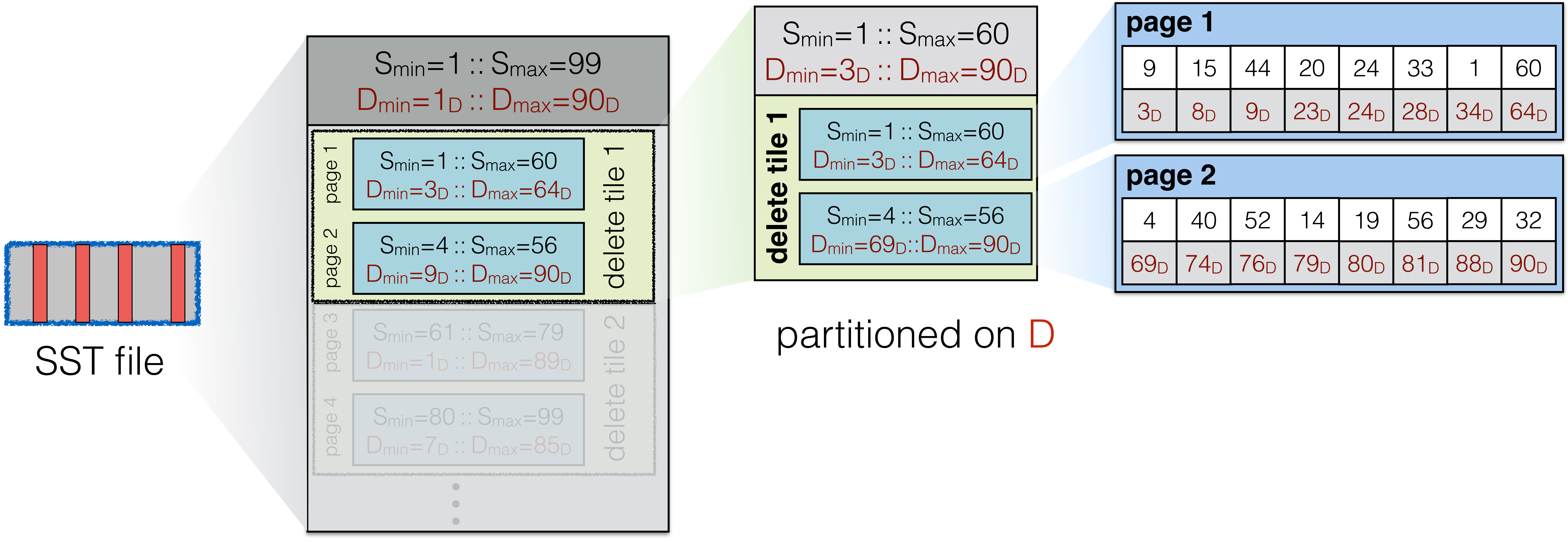
page 2

29	32	33	40	44	52	56	60
88$_D$	90$_D$	28$_D$	74$_D$	9$_D$	76$_D$	81$_D$	64$_D$

partitioned on S

Key Weaving storage layout

delete all entries with timestamp **<= 65_D**



SST file

$S_{min}=1 :: S_{max}=99$
$D_{min}=1_D :: D_{max}=90_D$

delete tile 1

page 1: $S_{min}=1 :: S_{max}=60$, $D_{min}=3_D :: D_{max}=64_D$

page 2: $S_{min}=4 :: S_{max}=56$, $D_{min}=9_D :: D_{max}=90_D$

delete tile 2

page 3: $S_{min}=61 :: S_{max}=79$, $D_{min}=1_D :: D_{max}=89_D$

page 4: $S_{min}=80 :: S_{max}=99$, $D_{min}=7_D :: D_{max}=85_D$

partitioned on S

$S_{min}=1 :: S_{max}=60$
$D_{min}=3_D :: D_{max}=90_D$

delete tile 1

$S_{min}=1 :: S_{max}=60$
$D_{min}=3_D :: D_{max}=64_D$

$S_{min}=4 :: S_{max}=56$
$D_{min}=69_D::D_{max}=90_D$

partitioned on D

page 1

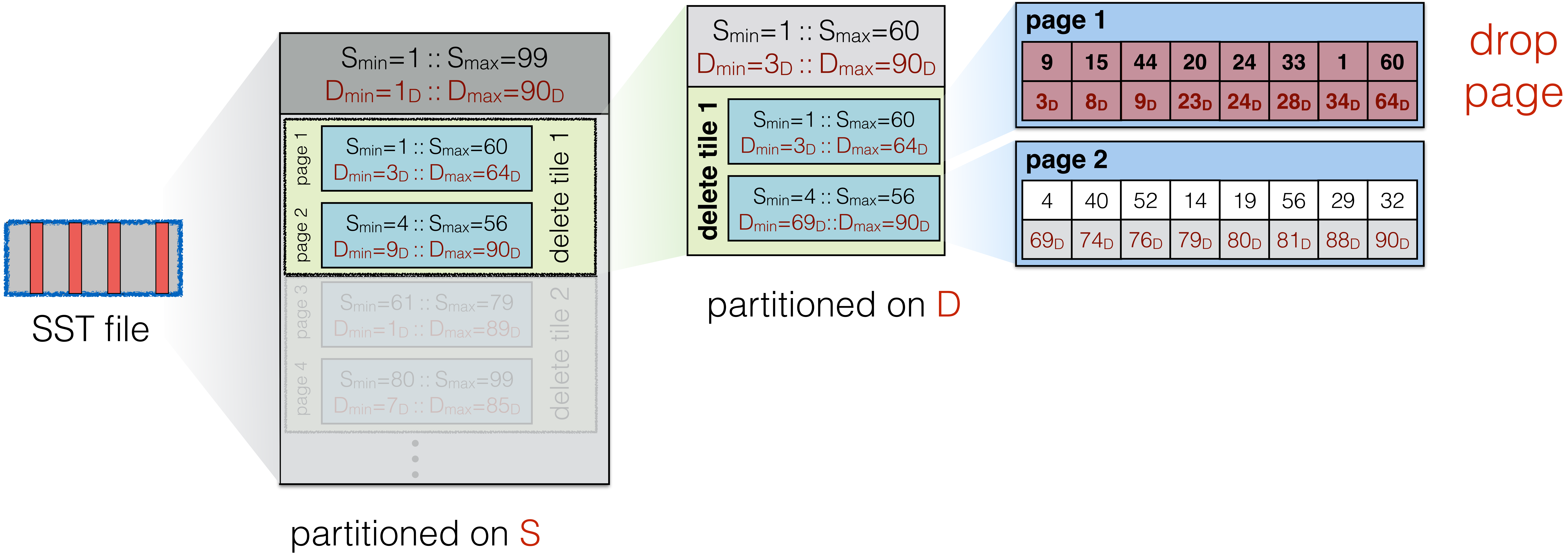
9	15	44	20	24	33	1	60
3_D	8_D	9_D	23_D	24_D	28_D	34_D	64_D

page 2

4	40	52	14	19	56	29	32
69_D	74_D	76_D	79_D	80_D	81_D	88_D	90_D

Key Weaving storage layout

delete all entries with timestamp **<= 65_D**



SST file

$S_{min}=1$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=90_D$

delete tile 1

page 1: $S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=64_D$

page 2: $S_{min}=4$ :: $S_{max}=56$
$D_{min}=9_D$ :: $D_{max}=90_D$

delete tile 2

page 3: $S_{min}=61$ :: $S_{max}=79$
$D_{min}=1_D$ :: $D_{max}=89_D$

page 4: $S_{min}=80$ :: $S_{max}=99$
$D_{min}=7_D$ :: $D_{max}=85_D$

partitioned on S

$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=90_D$

delete tile 1

$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=64_D$

$S_{min}=4$ :: $S_{max}=56$
$D_{min}=69_D$::$D_{max}=90_D$

partitioned on D

page 1

9	**15**	**44**	**20**	**24**	**33**	**1**	**60**
3_D	**8_D**	**9_D**	**23_D**	**24_D**	**28_D**	**34_D**	**64_D**

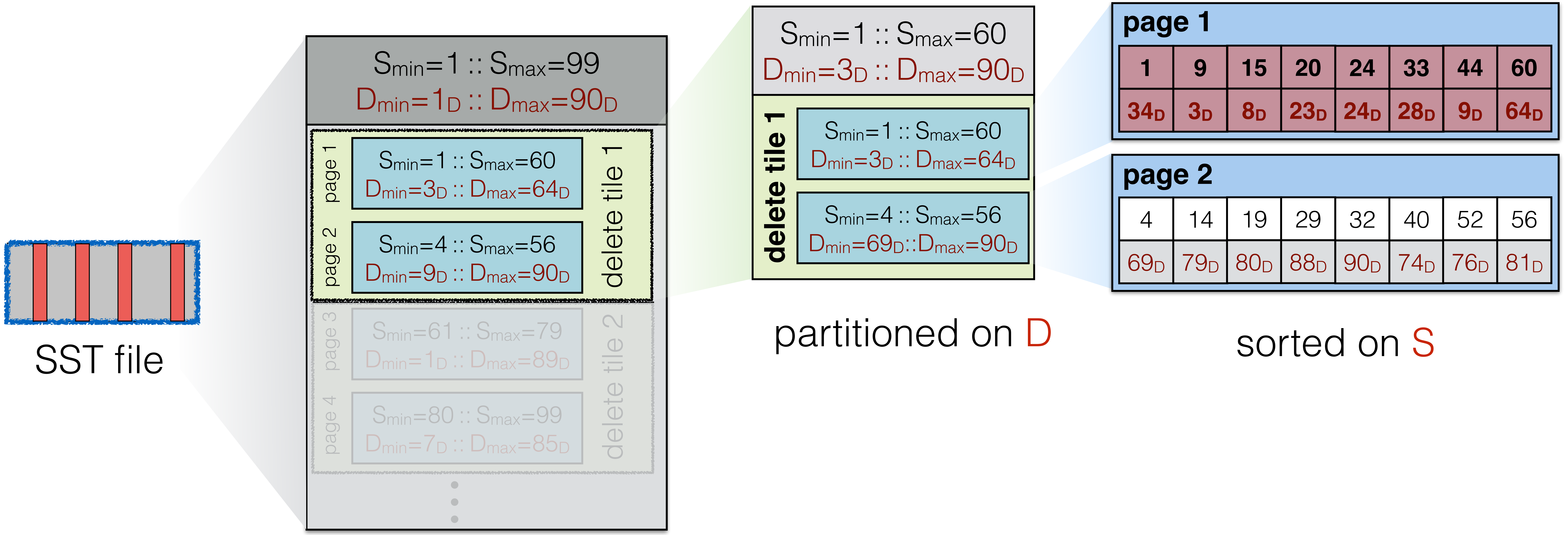
page 2

4	40	52	14	19	56	29	32
69_D	74_D	76_D	79_D	80_D	81_D	88_D	90_D

drop page

Key Weaving storage layout

delete all entries with timestamp **<= 65$_D$**


SST file
$S_{min}=1$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=90_D$
page 1
$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=64_D$
page 2
$S_{min}=4$ :: $S_{max}=56$
$D_{min}=9_D$ :: $D_{max}=90_D$
delete tile 1
page 3
$S_{min}=61$ :: $S_{max}=79$
$D_{min}=1_D$ :: $D_{max}=89_D$
page 4
$S_{min}=80$ :: $S_{max}=99$
$D_{min}=7_D$ :: $D_{max}=85_D$
delete tile 2
$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=90_D$
delete tile 1
$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=64_D$
$S_{min}=4$ :: $S_{max}=56$
$D_{min}=69_D$::$D_{max}=90_D$
page 1
page 2

1	**9**	**15**	**20**	**24**	**33**	**44**	**60**
34$_D$	**3$_D$**	**8$_D$**	**23$_D$**	**24$_D$**	**28$_D$**	**9$_D$**	**64$_D$**

4	14	19	29	32	40	52	56
69$_D$	79$_D$	80$_D$	88$_D$	90$_D$	74$_D$	76$_D$	81$_D$

drop page

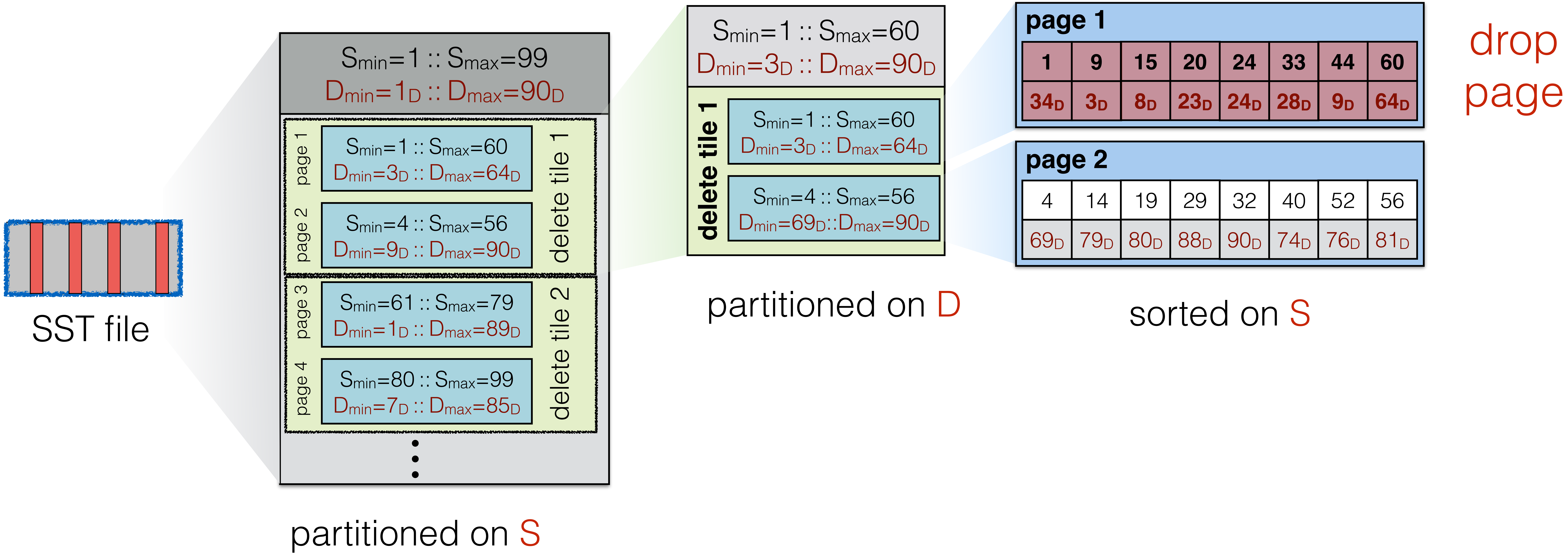
partitioned on D

sorted on S

partitioned on S

Key Weaving storage layout

delete all entries with timestamp **<= 65_D**



SST file

$S_{min}=1$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=90_D$

page 1
$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=64_D$
page 2
$S_{min}=4$ :: $S_{max}=56$
$D_{min}=9_D$ :: $D_{max}=90_D$
delete tile 1
page 3
$S_{min}=61$ :: $S_{max}=79$
$D_{min}=1_D$ :: $D_{max}=89_D$
page 4
$S_{min}=80$ :: $S_{max}=99$
$D_{min}=7_D$ :: $D_{max}=85_D$
delete tile 2

partitioned on S

$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=90_D$

delete tile 1
$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=64_D$
$S_{min}=4$ :: $S_{max}=56$
$D_{min}=69_D$::$D_{max}=90_D$

partitioned on D

page 1

1	**9**	**15**	**20**	**24**	**33**	**44**	**60**
34_D	**3_D**	**8_D**	**23_D**	**24_D**	**28_D**	**9_D**	**64_D**

page 2

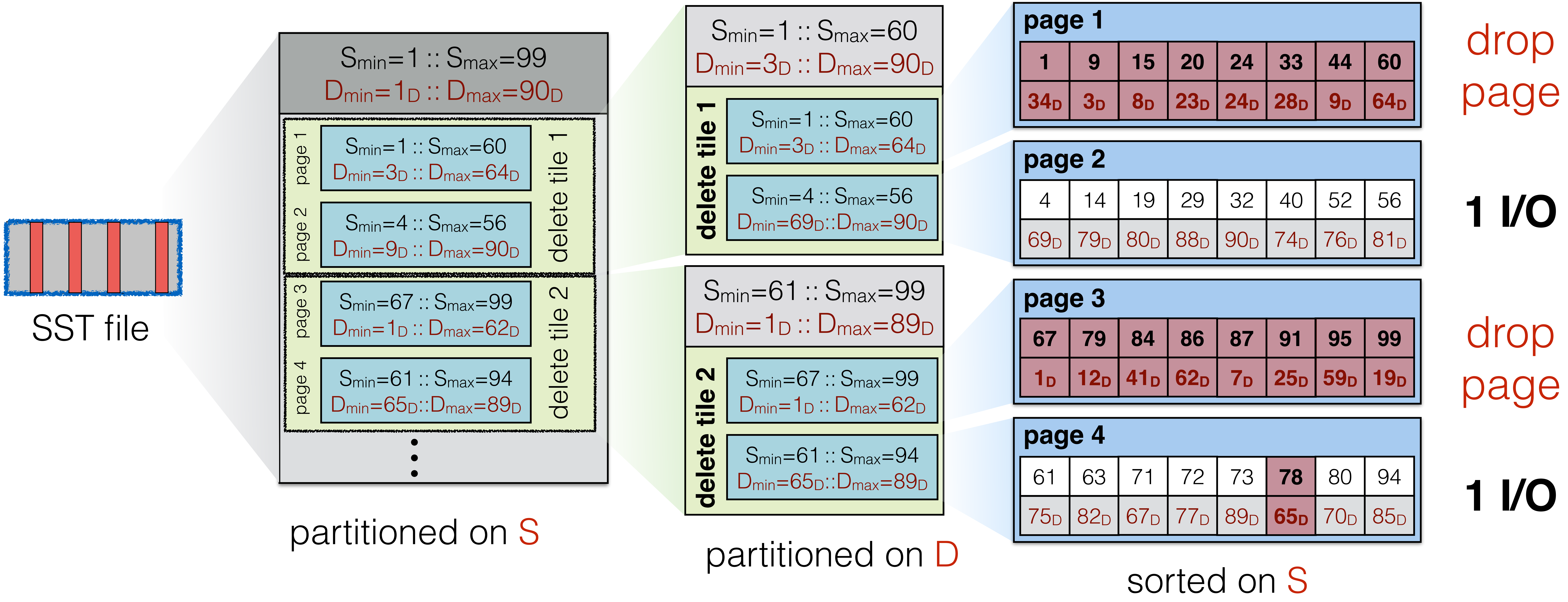
4	14	19	29	32	40	52	56
69_D	79_D	80_D	88_D	90_D	74_D	76_D	81_D

drop page

sorted on S

Key Weaving storage layout

delete all entries with timestamp **<= 65_D**



SST file
$S_{min}=1$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=90_D$
page 1
$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=64_D$
page 2
$S_{min}=4$ :: $S_{max}=56$
$D_{min}=9_D$ :: $D_{max}=90_D$
delete tile 1
page 3
$S_{min}=67$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=62_D$
page 4
$S_{min}=61$ :: $S_{max}=94$
$D_{min}=65_D$::$D_{max}=89_D$
delete tile 2
⋮
partitioned on S
$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=90_D$
delete tile 1
$S_{min}=1$ :: $S_{max}=60$
$D_{min}=3_D$ :: $D_{max}=64_D$
$S_{min}=4$ :: $S_{max}=56$
$D_{min}=69_D$::$D_{max}=90_D$
$S_{min}=61$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=89_D$
delete tile 2
$S_{min}=67$ :: $S_{max}=99$
$D_{min}=1_D$ :: $D_{max}=62_D$
$S_{min}=61$ :: $S_{max}=94$
$D_{min}=65_D$::$D_{max}=89_D$
partitioned on D
page 1
page 2
page 3
page 4
sorted on S
drop page
1 I/O
drop page
1 I/O

page 1

1	**9**	**15**	**20**	**24**	**33**	**44**	**60**
34_D	**3_D**	**8_D**	**23_D**	**24_D**	**28_D**	**9_D**	**64_D**

page 2

4	14	19	29	32	40	52	56
69_D	79_D	80_D	88_D	90_D	74_D	76_D	81_D

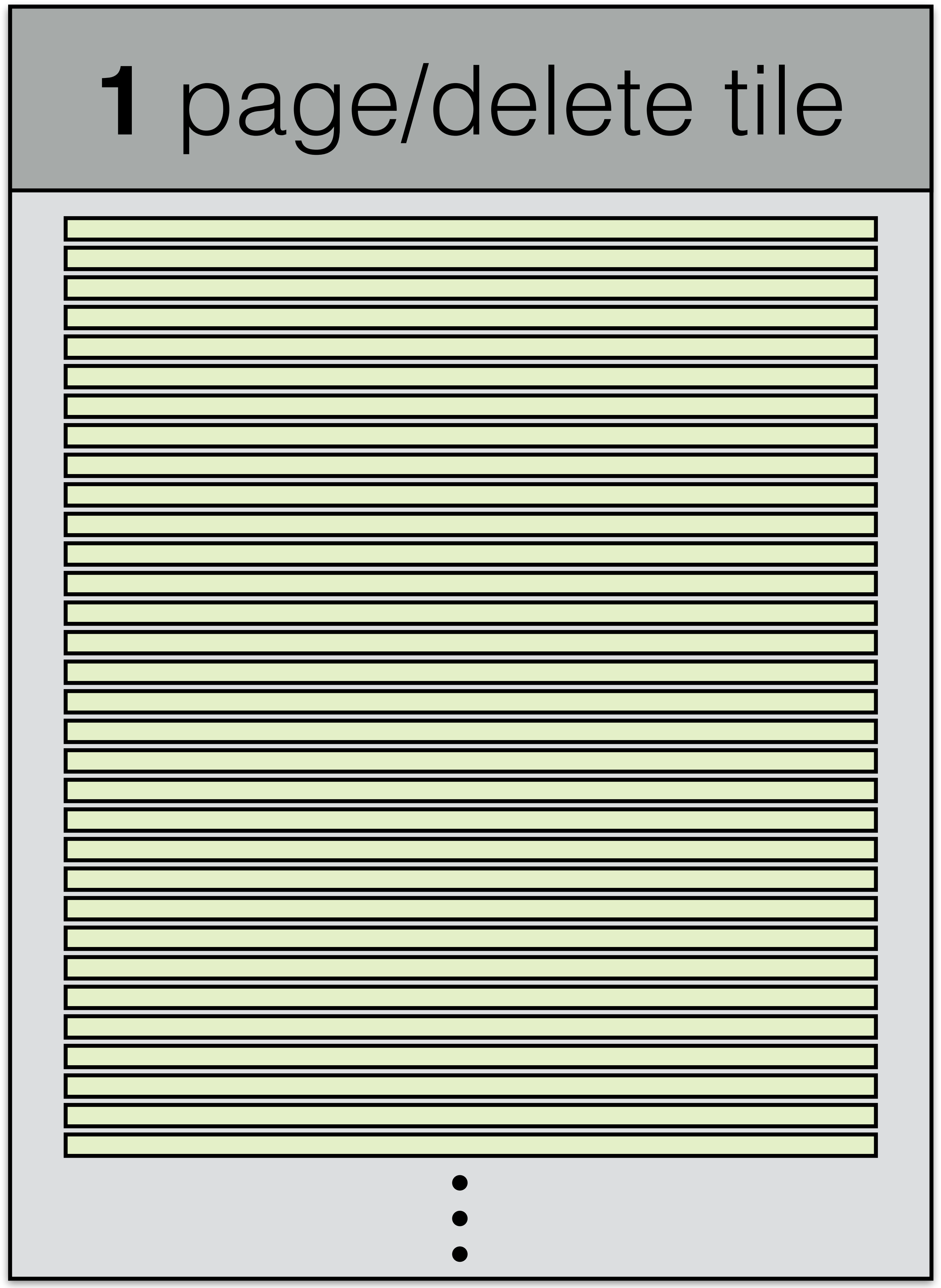
page 3

67	**79**	**84**	**86**	**87**	**91**	**95**	**99**
1_D	**12_D**	**41_D**	**62_D**	**7_D**	**25_D**	**59_D**	**19_D**

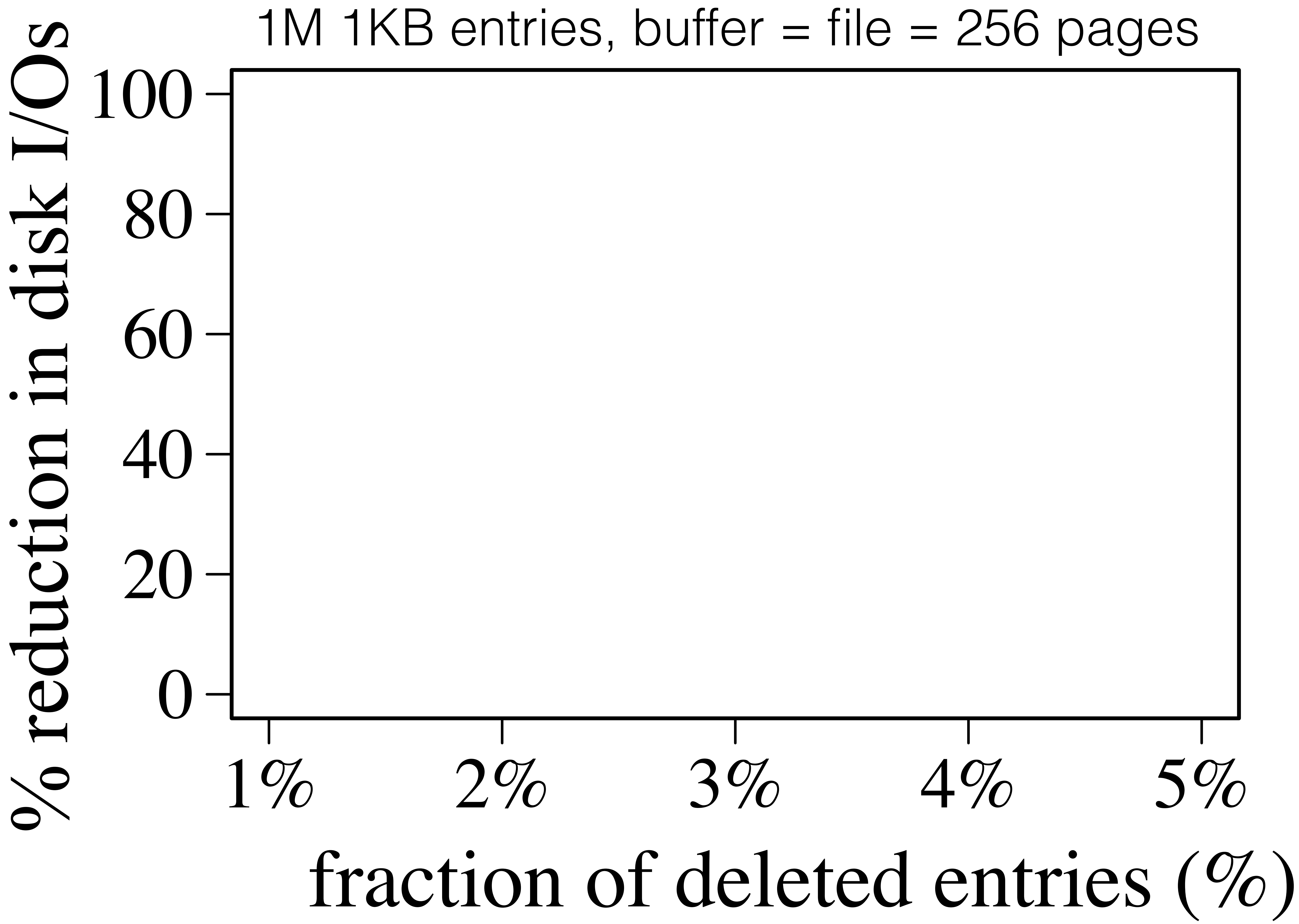
page 4

61	63	71	72	73	**78**	80	94
75_D	82_D	67_D	77_D	89_D	**65_D**	70_D	85_D

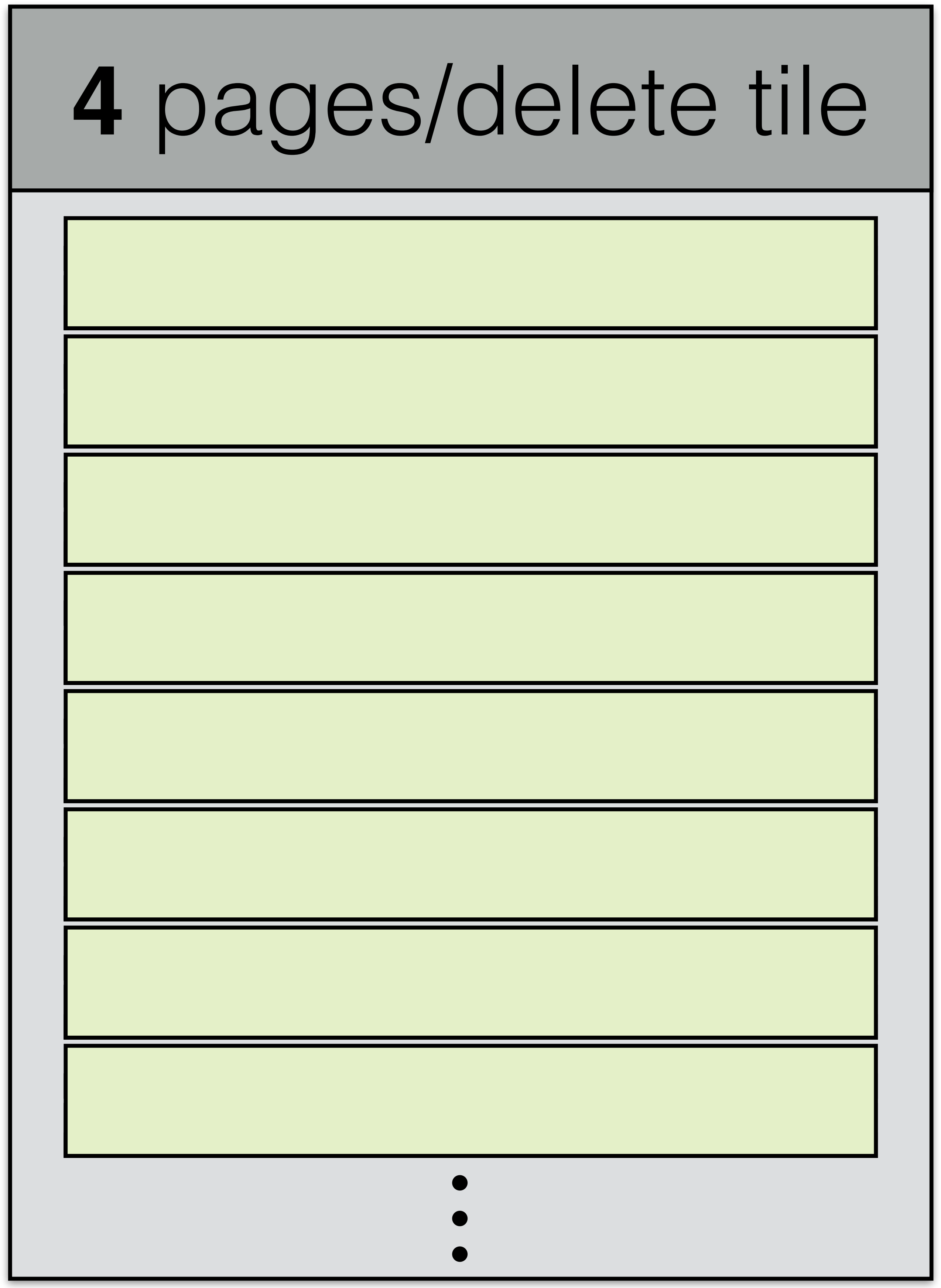
Key Weaving storage layout



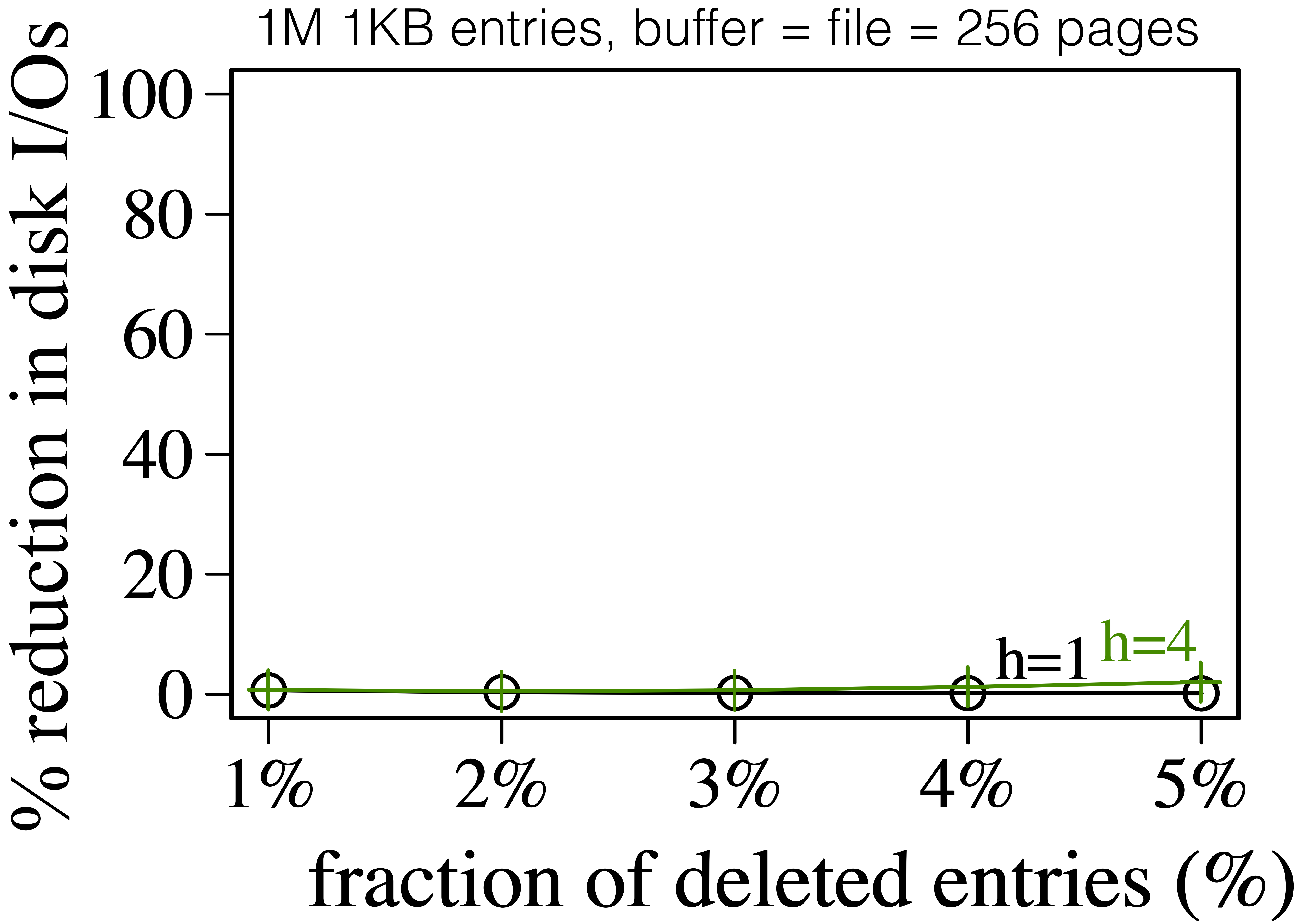
Internals of a file in KiWi



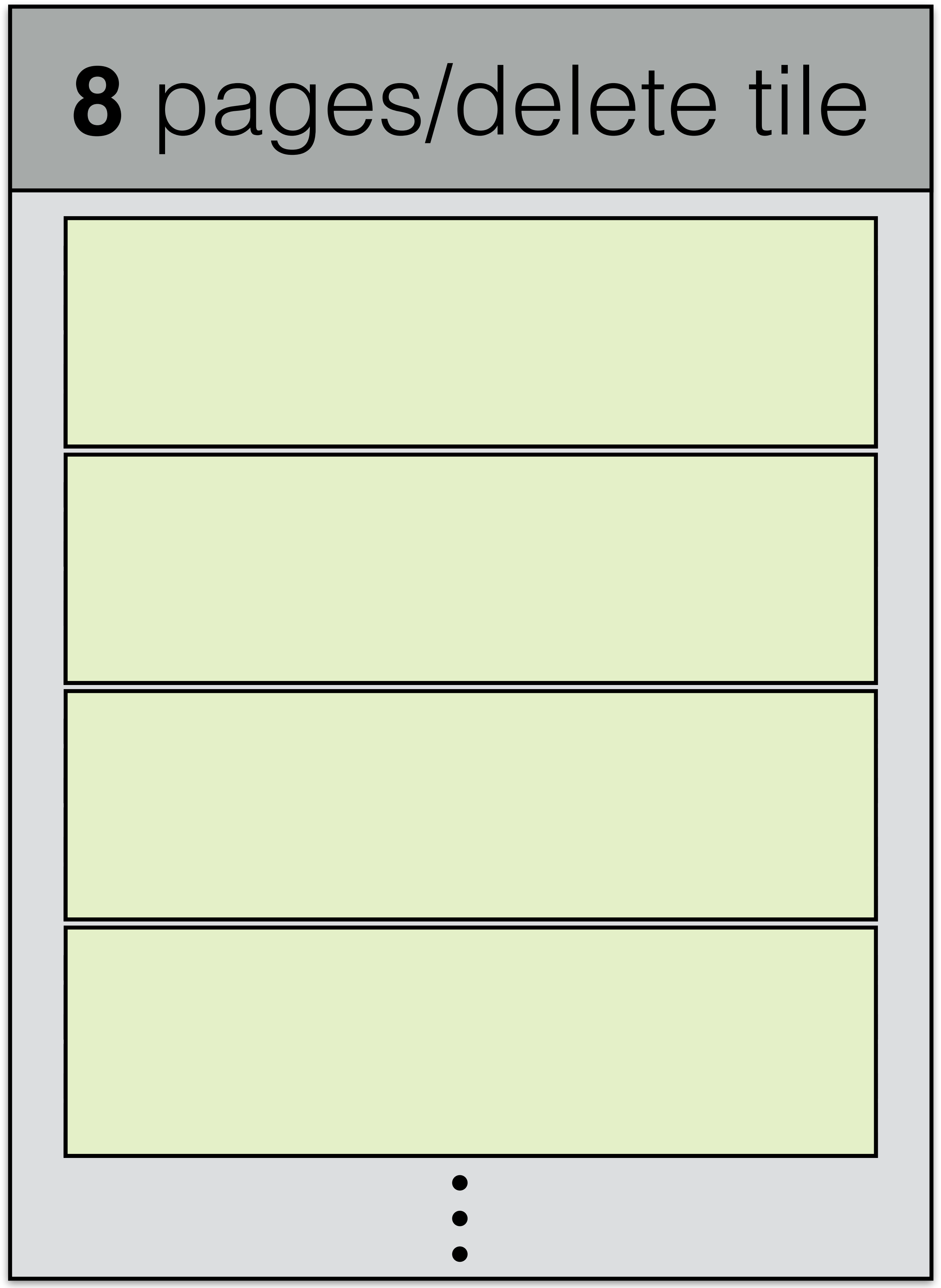
Key Weaving storage layout



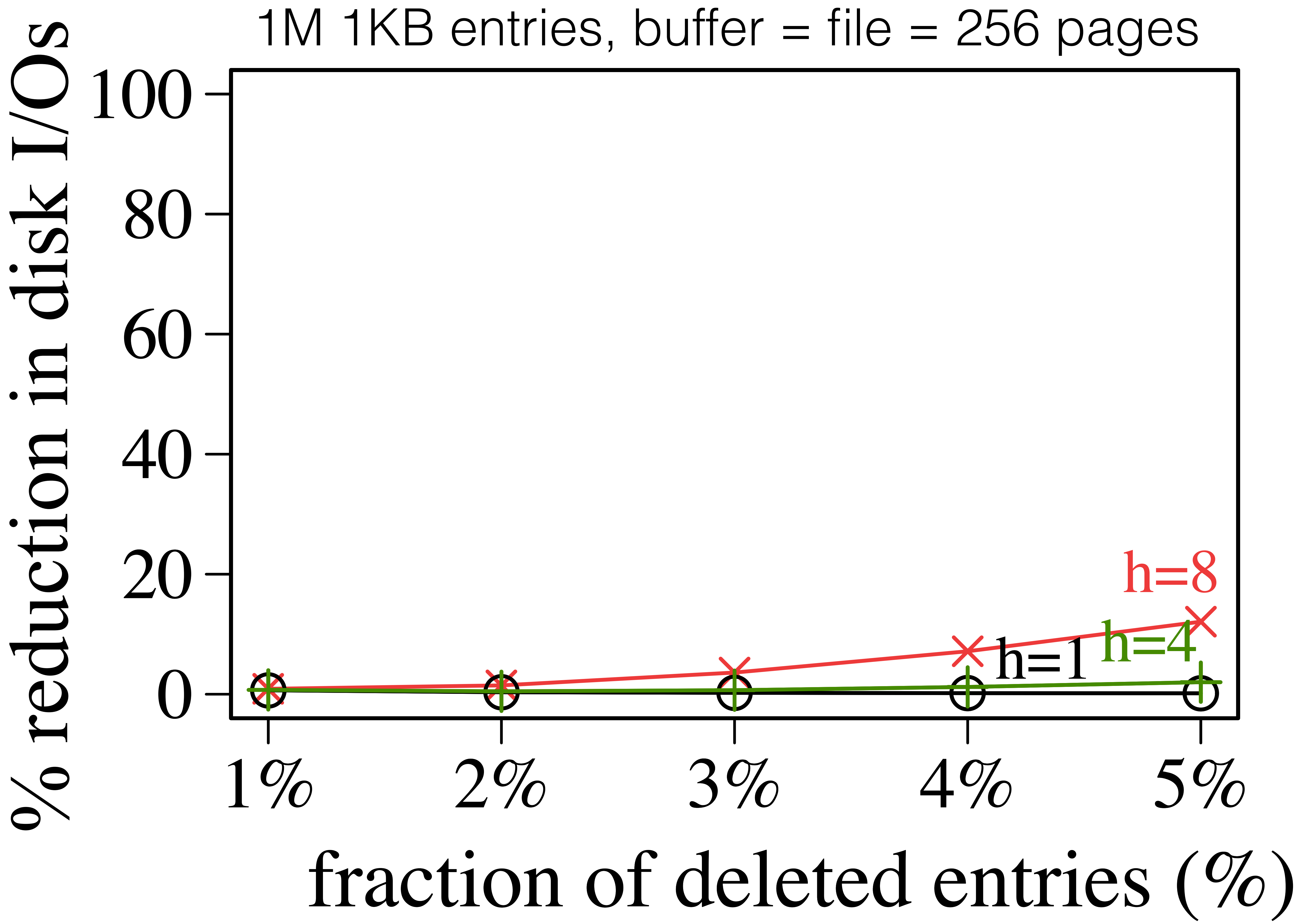
Internals of a file in KiWi



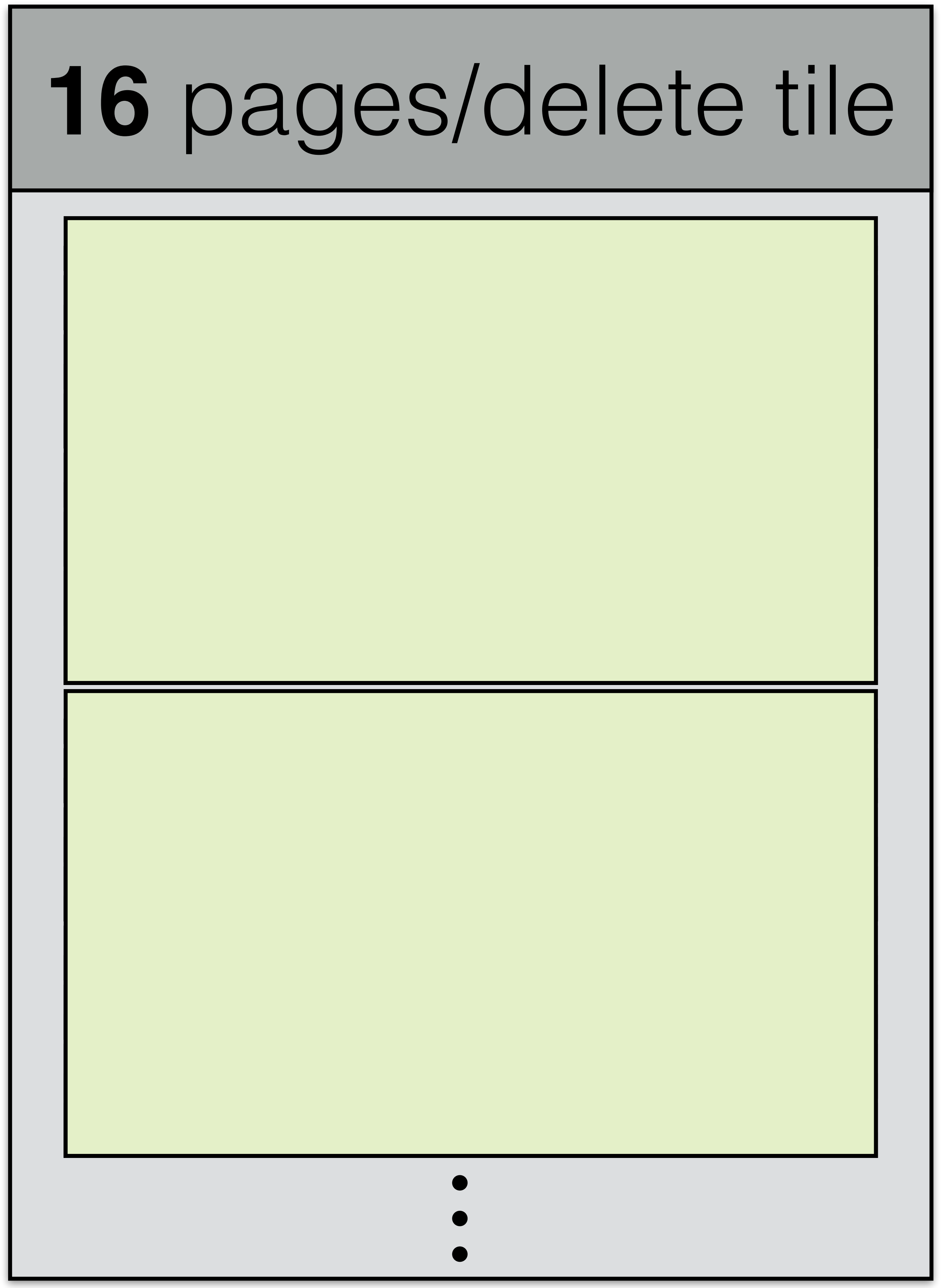
Key Weaving storage layout



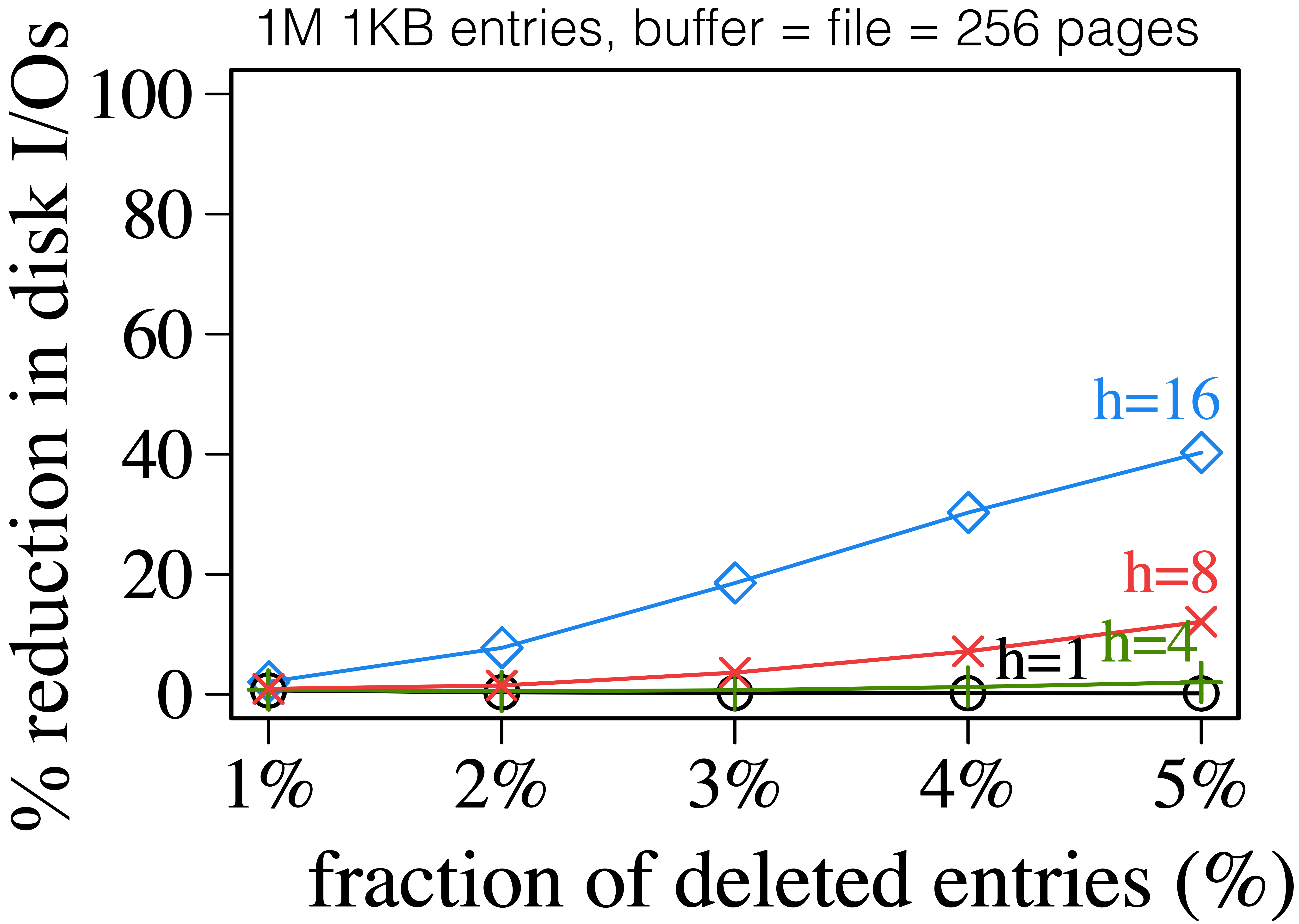
Internals of a file in KiWi



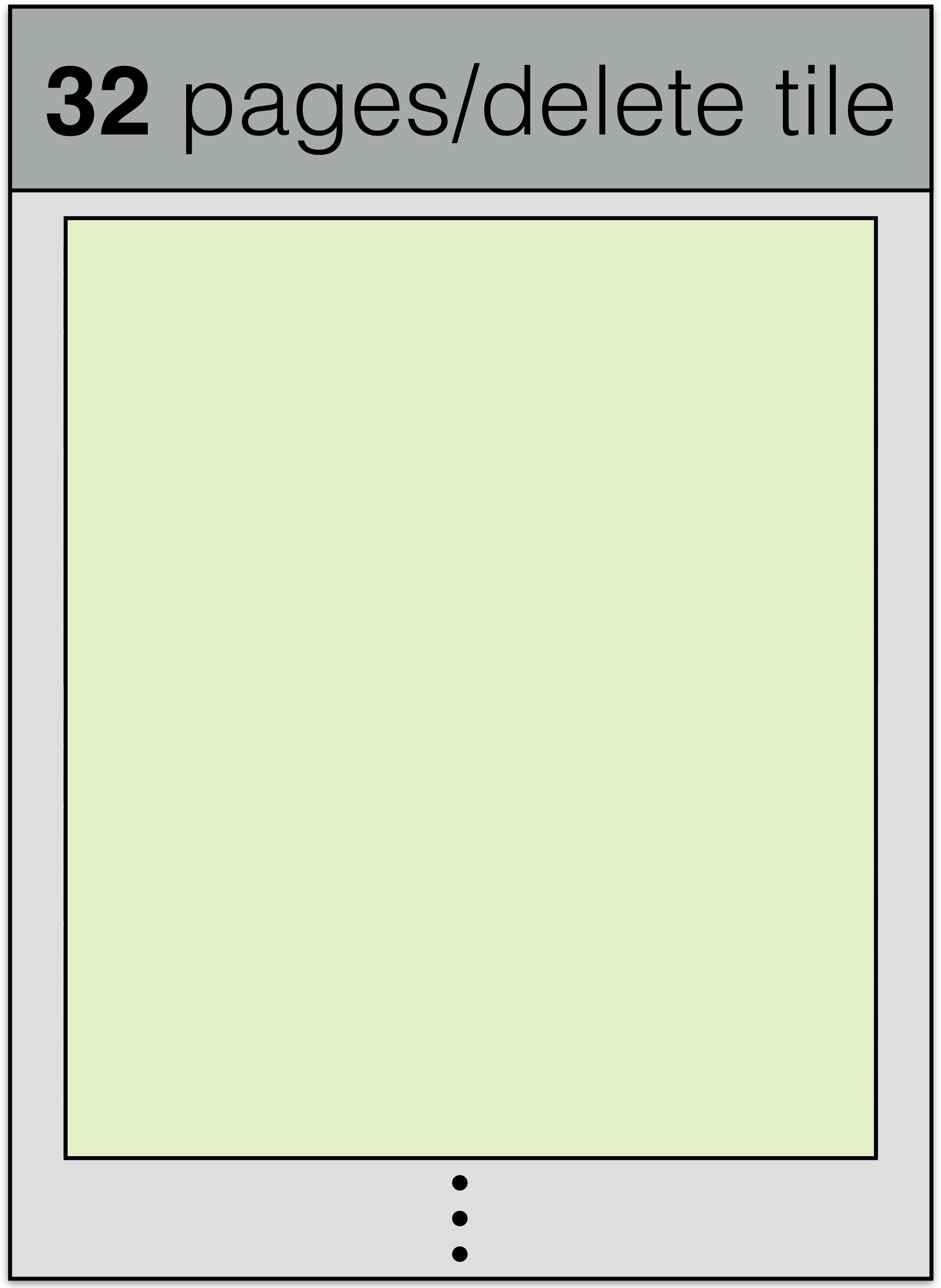
Key Weaving storage layout



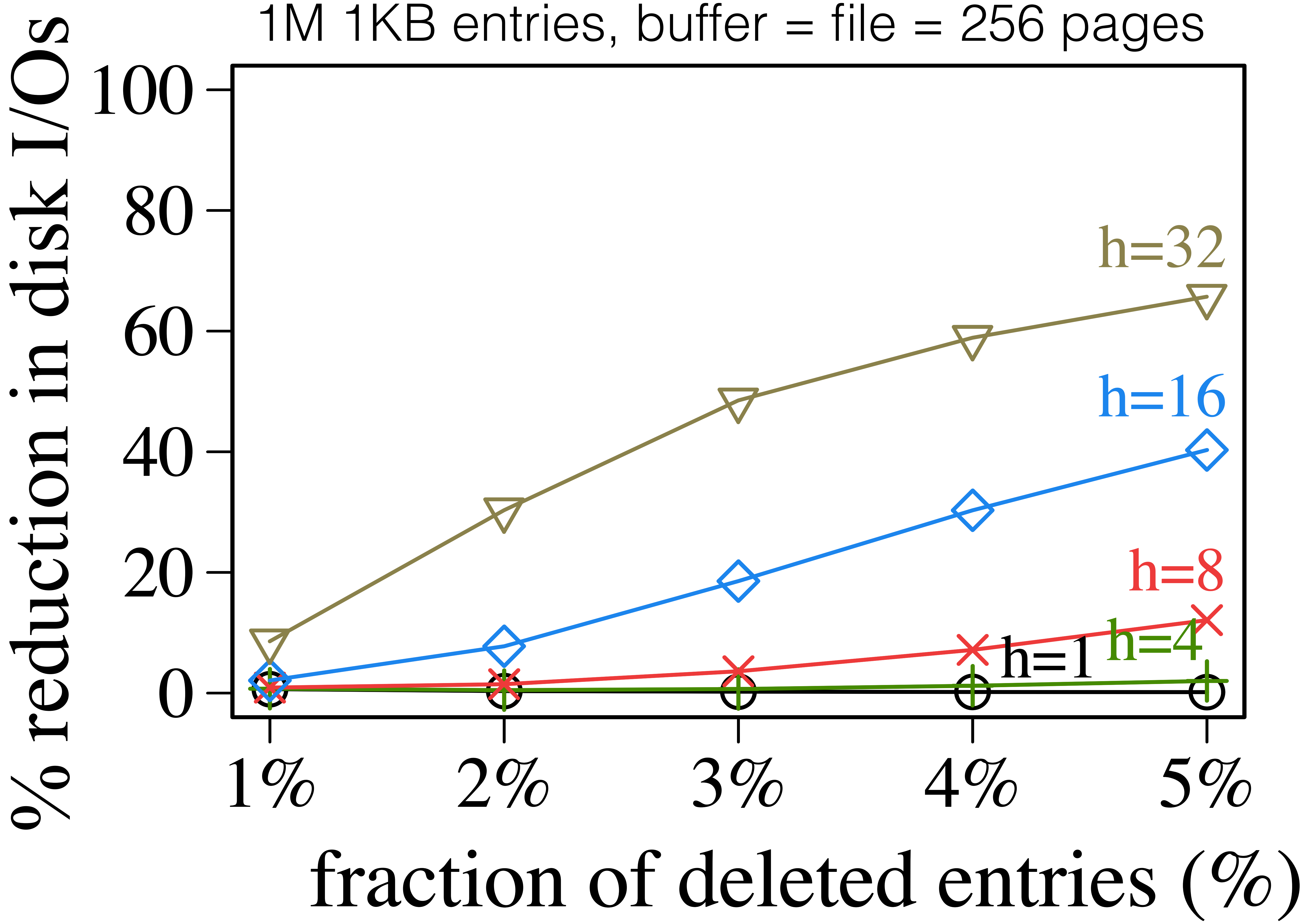
Internals of a file in KiWi



Key Weaving storage layout



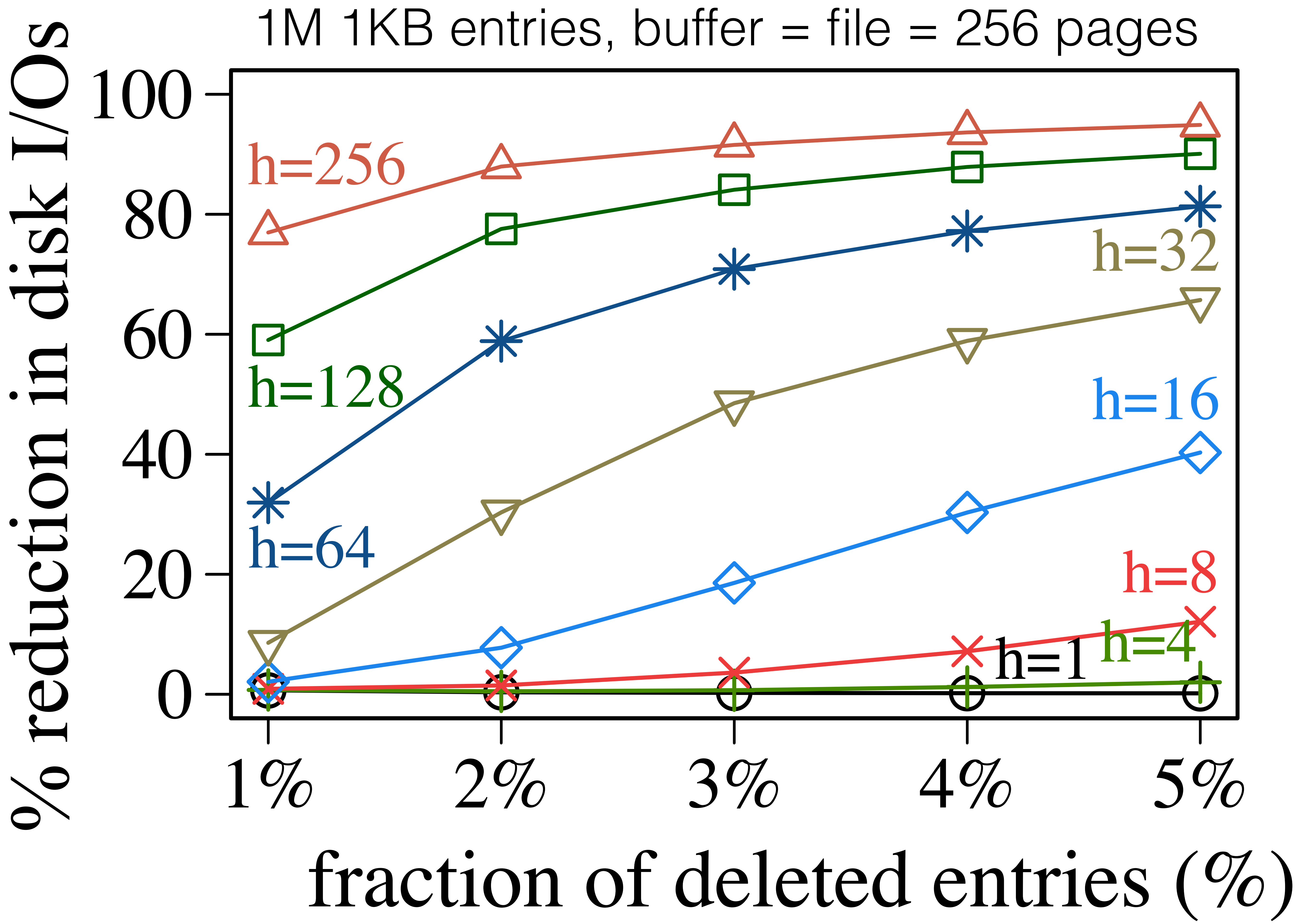
Internals of a file in KiWi



Key Weaving storage layout

reduced latency spikes ✓

full page drops reduces superfluous I/Os ✓



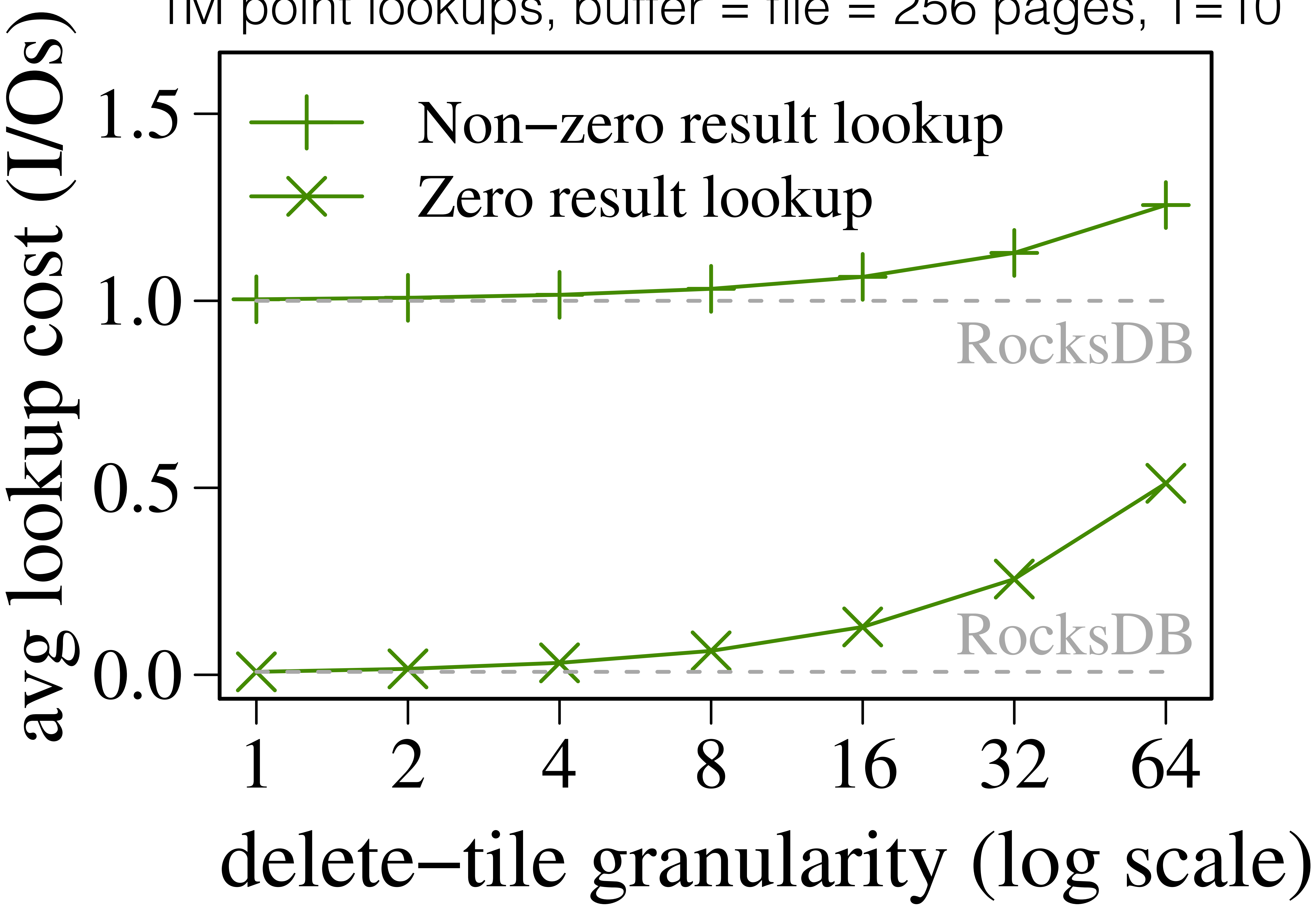
Key Weaving storage layout

higher lookup cost ↑

reduced latency spikes ✓

full page drops reduces superfluous I/Os ✓

1M point lookups, buffer = file = 256 pages, T=10



FADE

FAst DElete

amortized write ●

reduced space ✓

improved read ✓

timely delete persistence ✓

KiWi

higher lookup cost ↑

reduced latency spikes ✓

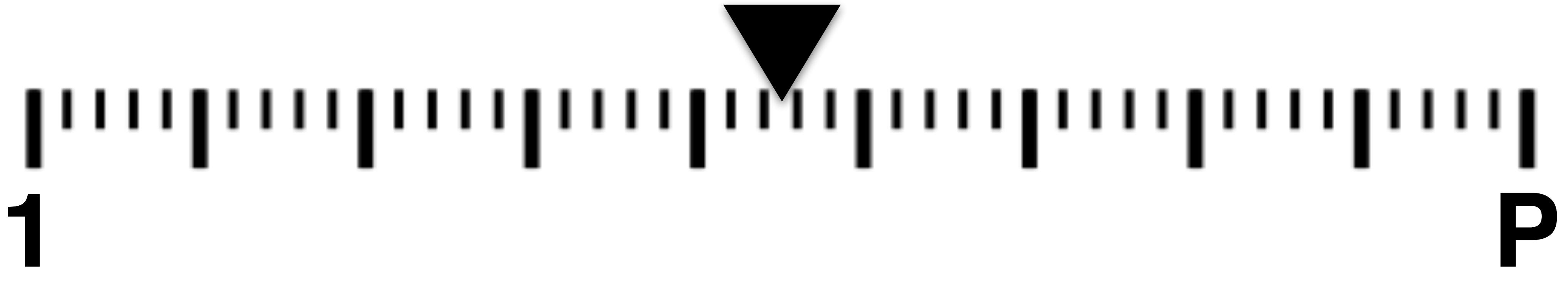
full page drops reduces superfluous I/Os ✓

FADE

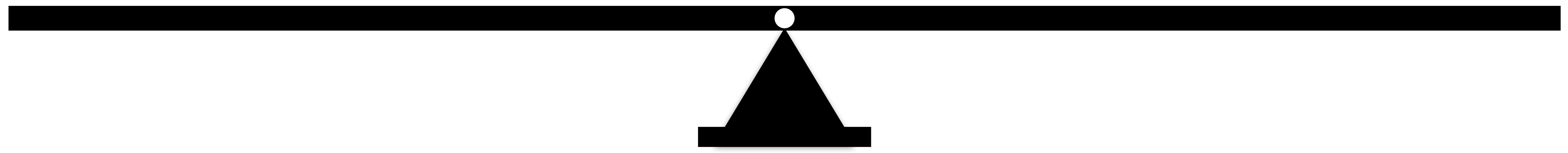
KiWi

Lethe

delete tile size

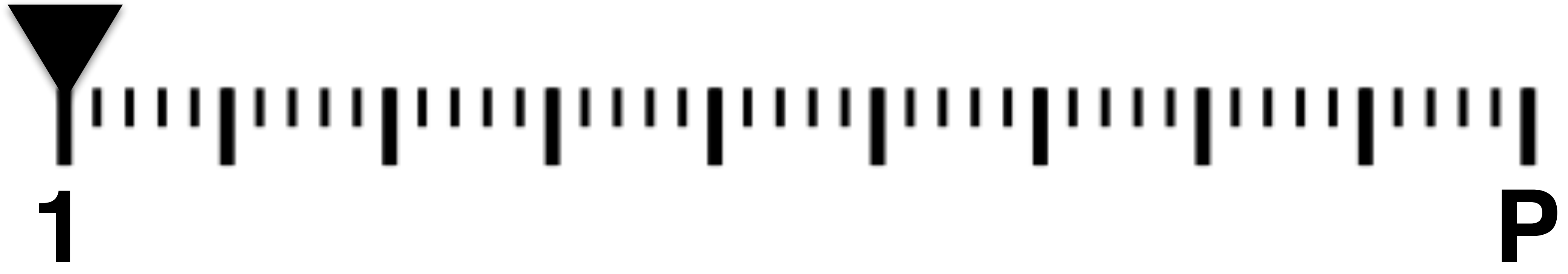

1

P


lookup
cost

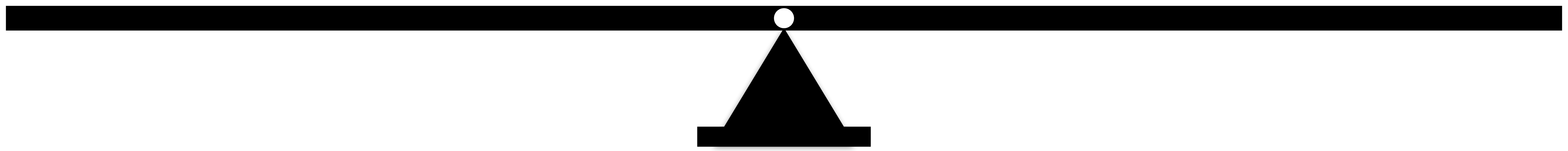
secondary range
delete cost

delete tile size



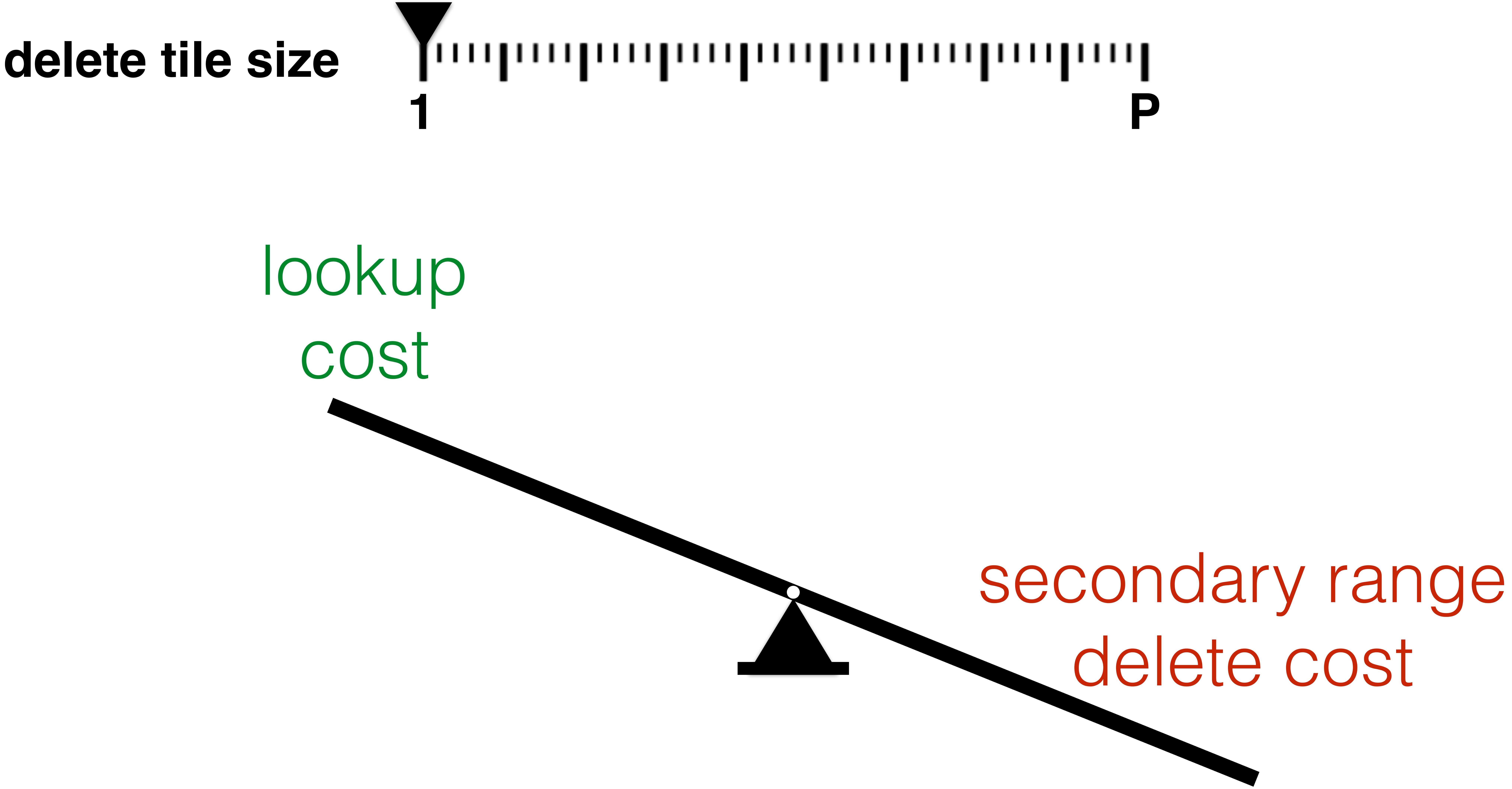
1

P



lookup cost

secondary range delete cost



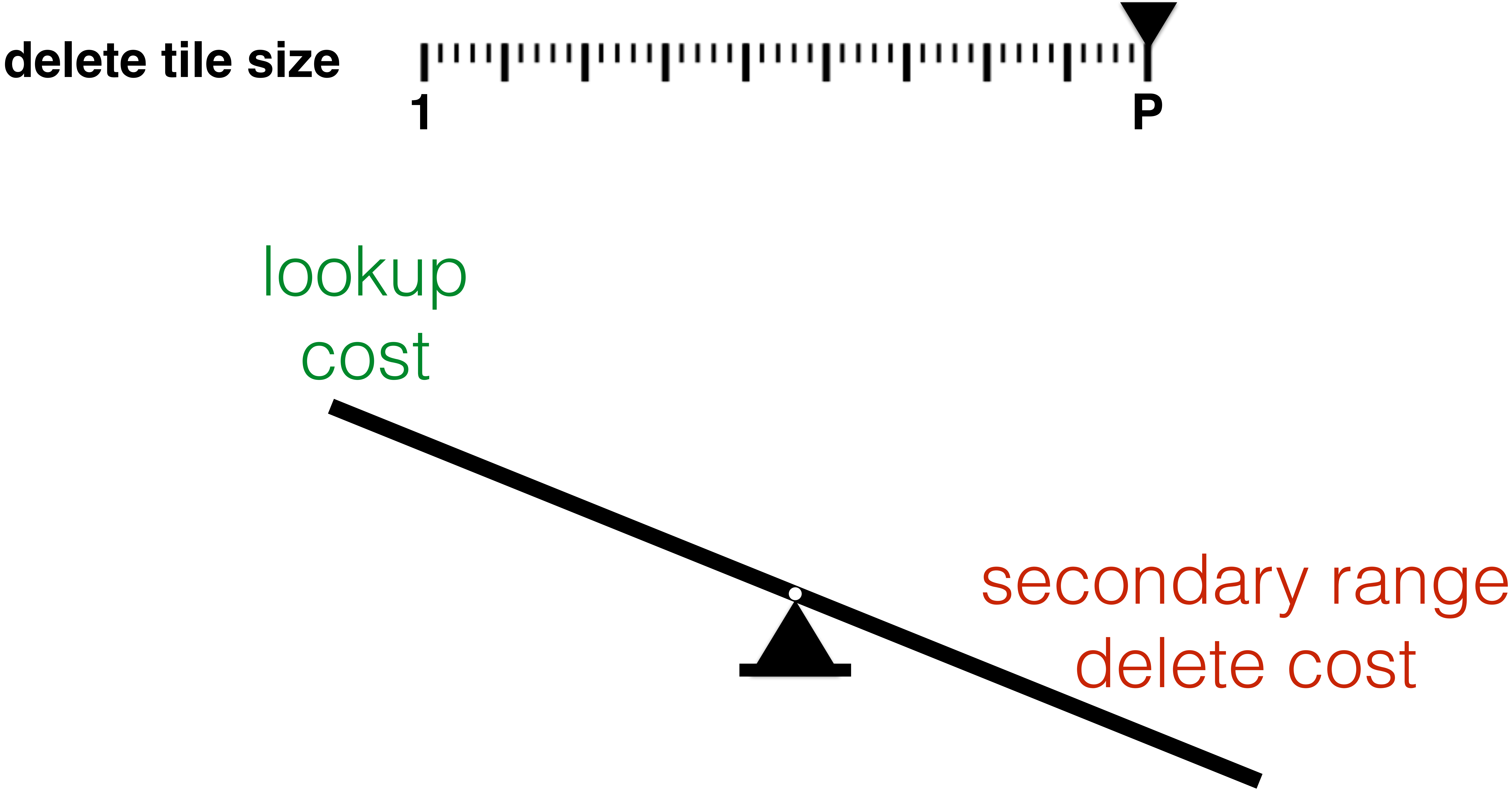
delete tile size

1

P

lookup cost

secondary range delete cost



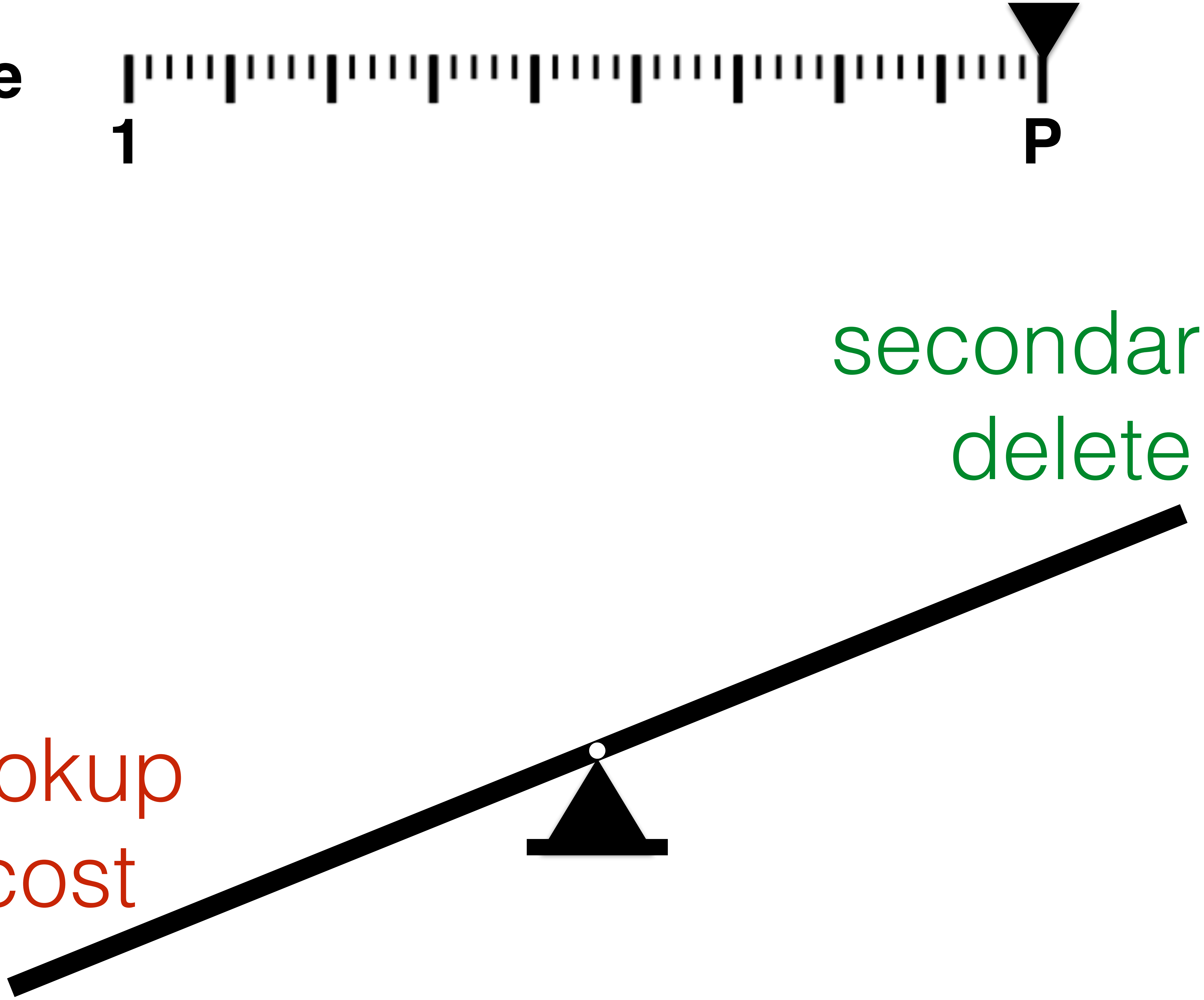
delete tile size

1

P

lookup cost

secondary range delete cost



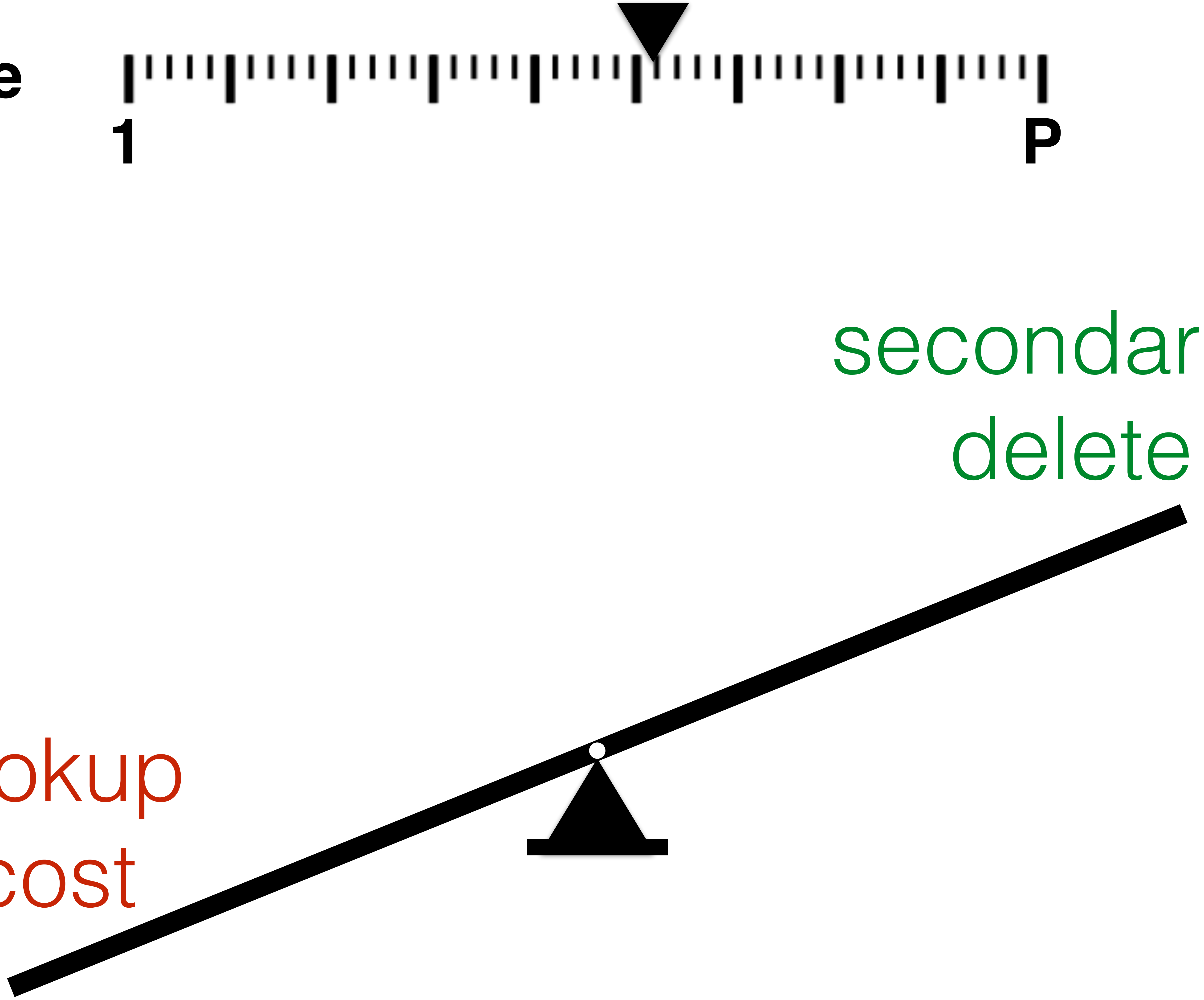
delete tile size

1

P

secondary range delete cost

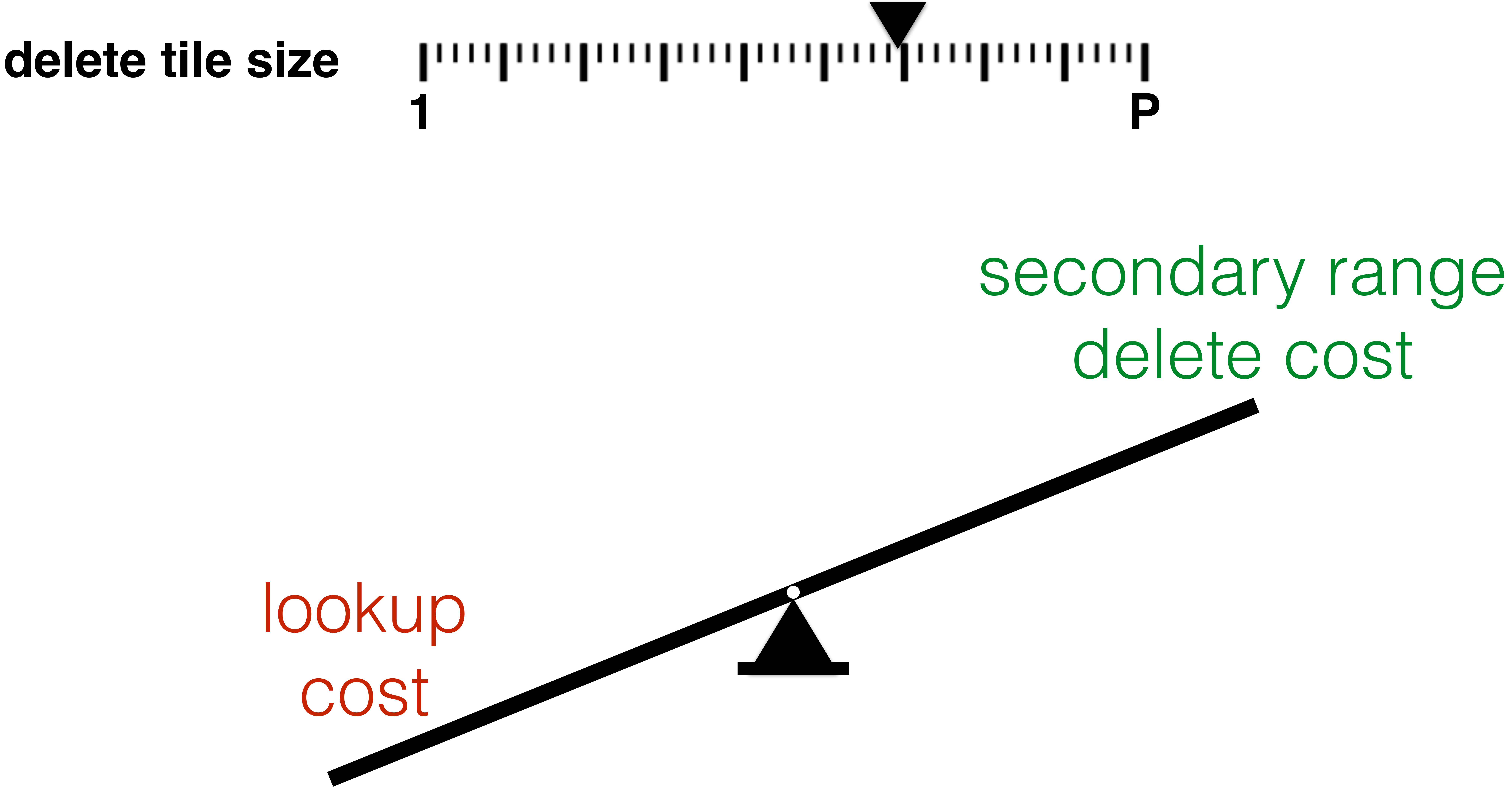
lookup cost



delete tile size

secondary range
delete cost

lookup
cost



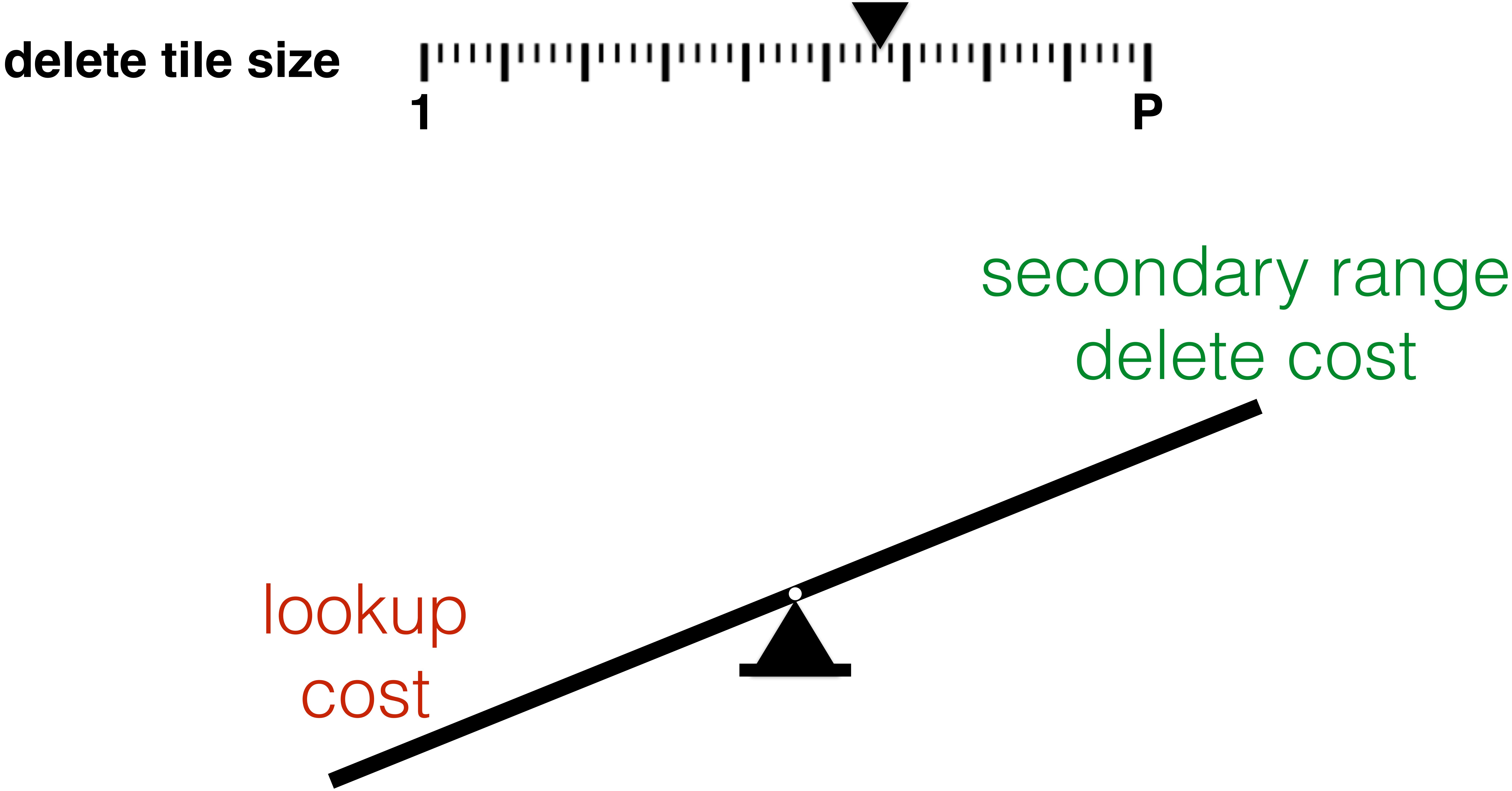
delete tile size

1

P

secondary range delete cost

lookup cost



delete tile size

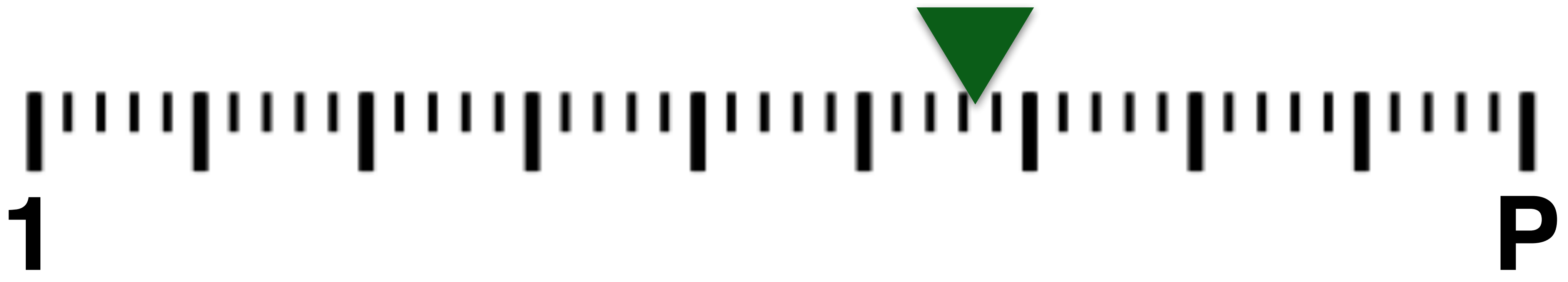
1

P

lookup cost

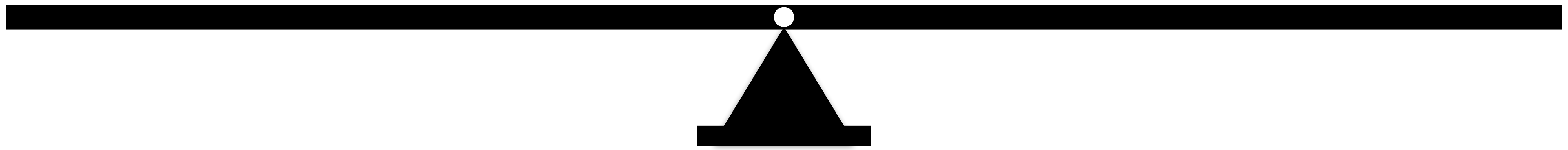
secondary range delete cost

delete tile size

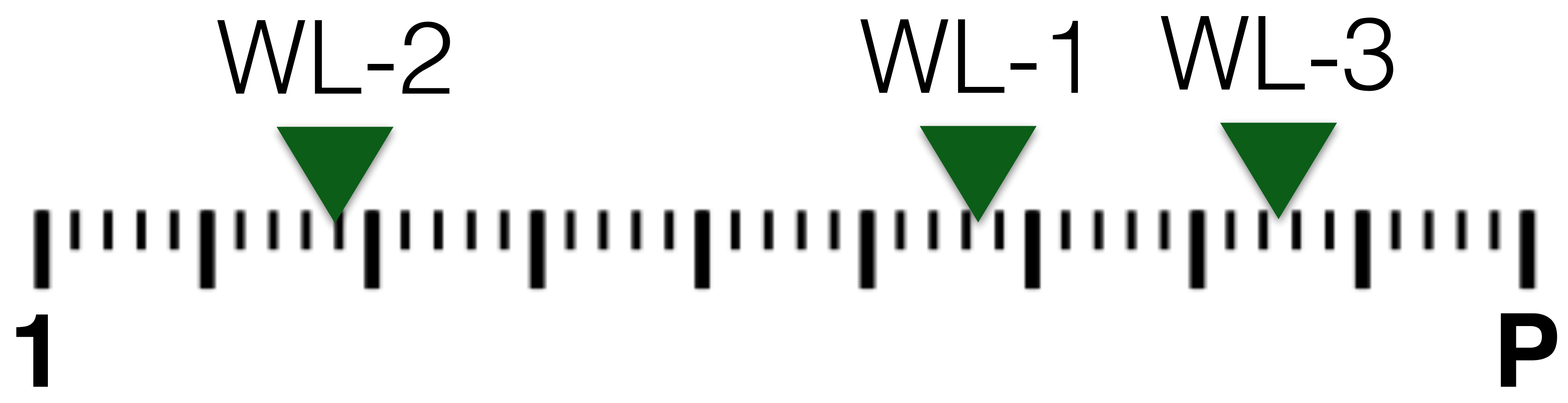


lookup cost

secondary range delete cost



delete tile size



WL-2

WL-1

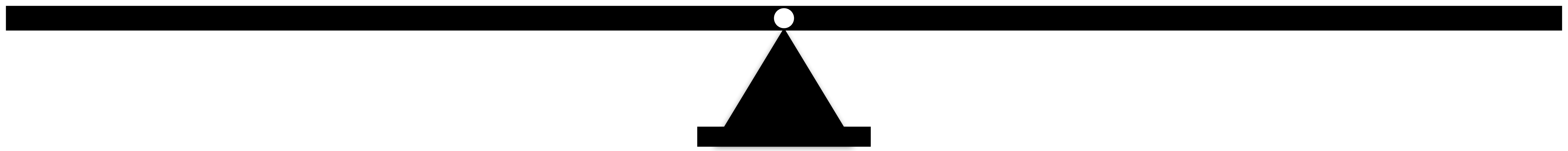
WL-3

1

P

lookup cost

secondary range delete cost



suboptimal state-of-the-art design

for workloads with deletes

FADE persists deletes timely

using latency-driven compactions

KiWi supports efficient
secondary range deletes

using key-interweaved data storage

suboptimal state of the art design

for workloads with deletes

FADE persists deletes timely

using latency-driven compactions

KiWi supports efficient secondary range deletes

by key-interweaved data layout

Lethe strikes balance between cost, performance, and latency

Thank You!

disc-projects.bu.edu/lethe/