

Enabling Efficient Deletes in Log-Structured KV-Stores



Subhadeep Sarkar

Enabling Efficient Deletes in Log-Structured KV-Stores

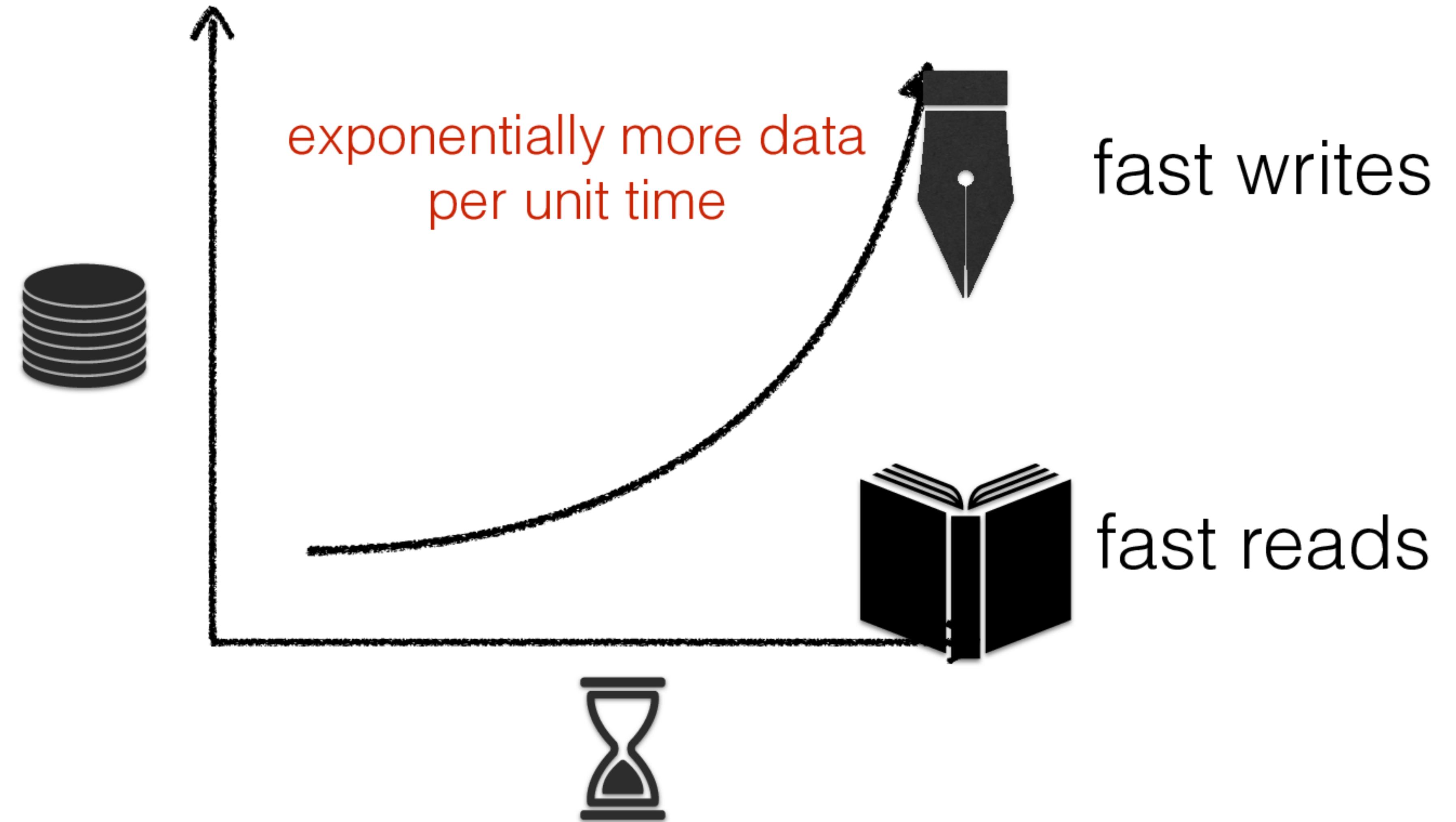


Subhadeep Sarkar

Enabling Efficient Deletes in Log-Structured KV-Stores



Lethe [\lē-thē\]
n: the goddess of
forgetfulness



key-value pairs

RID	timestamp	name	department	...	location
------------	------------------	-------------	-------------------	-----	-----------------

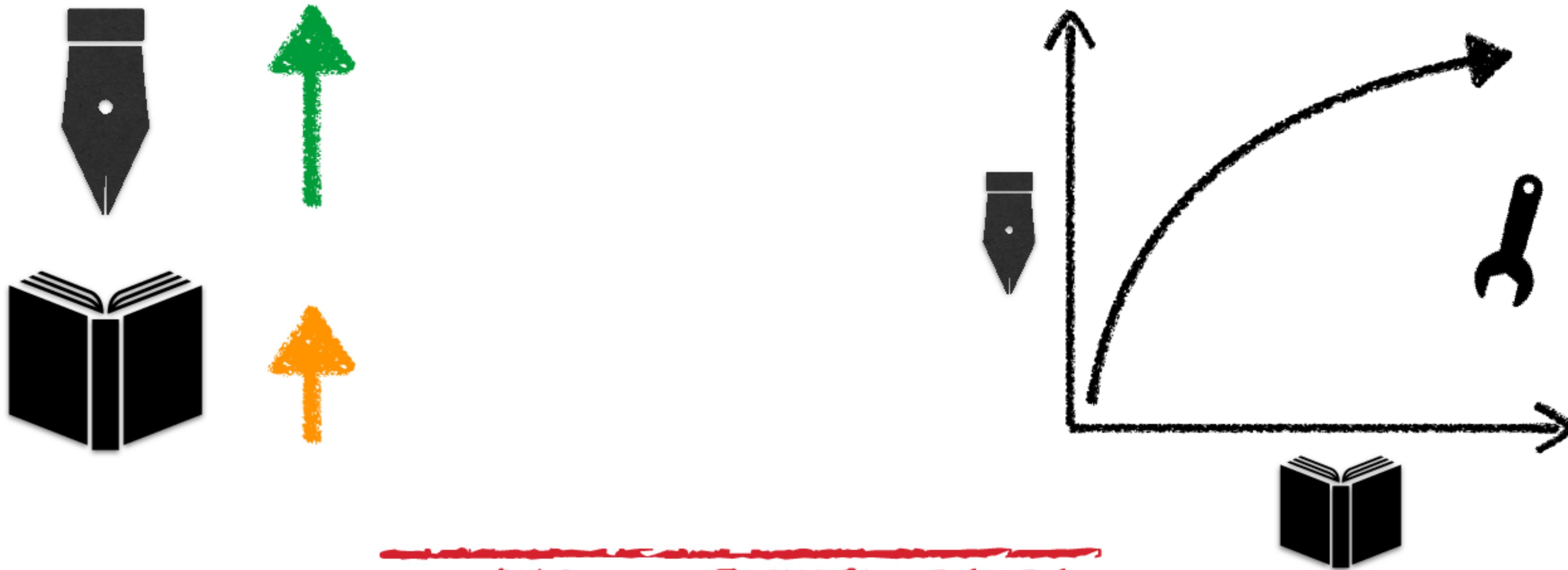




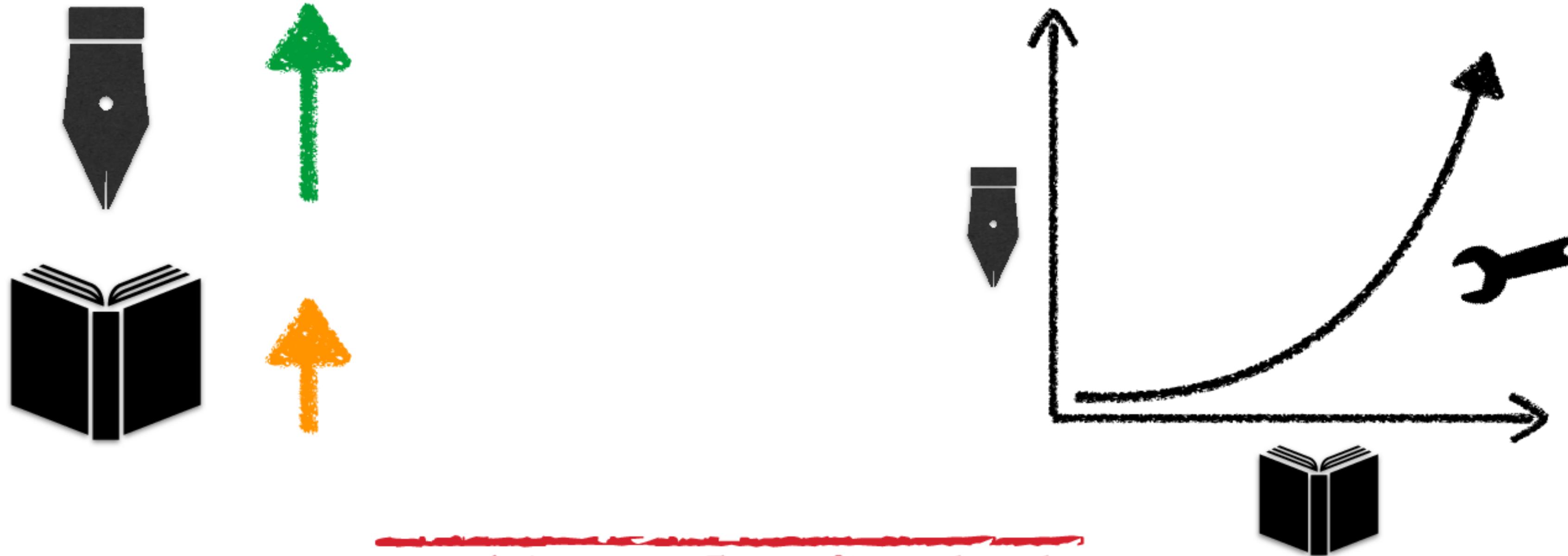
LSM-TREE



LSM-TREE



LSMI-TREE



LSM-TREE



cassandra



levelDB



RocksDB



SQLite



DynamoDB



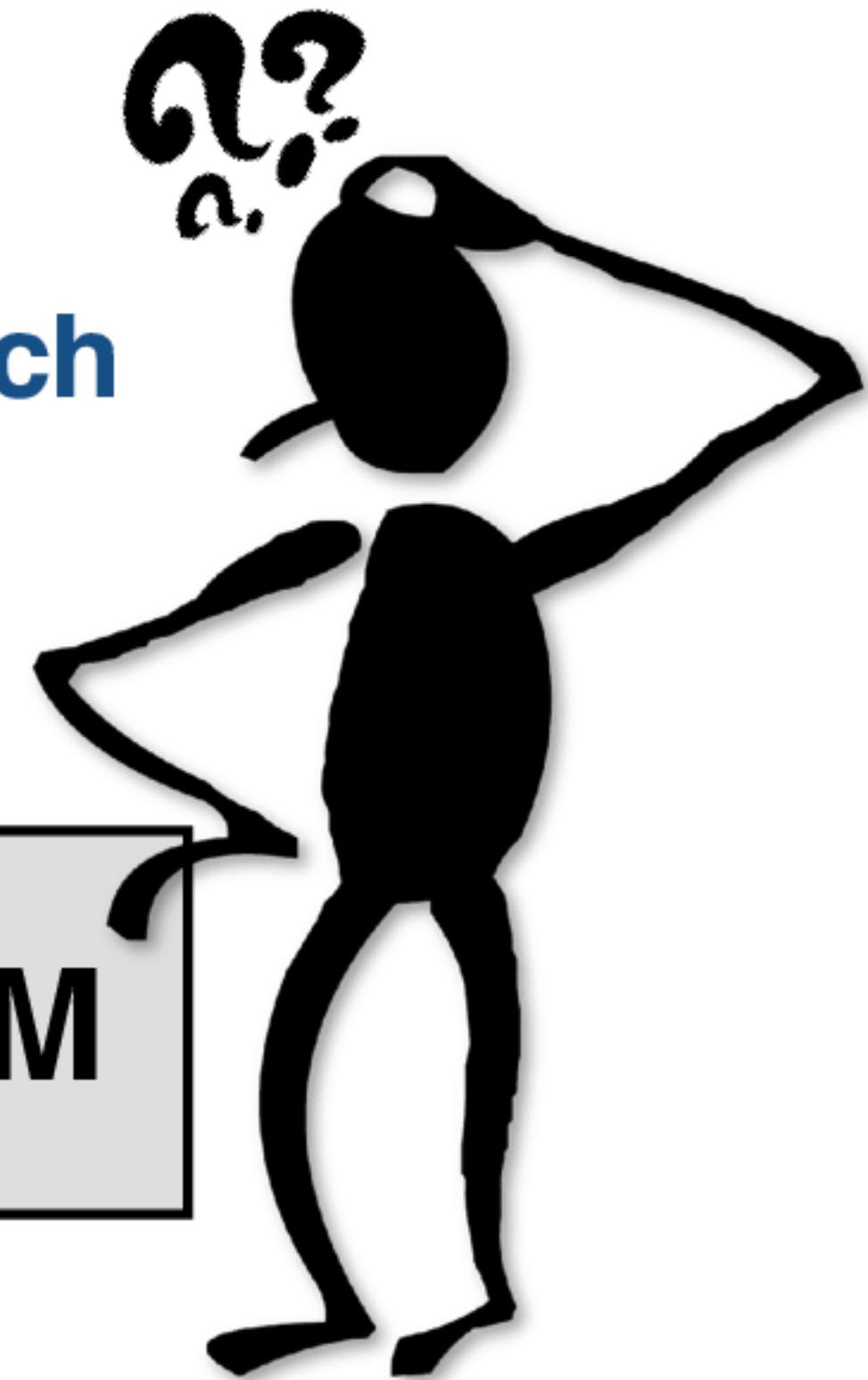
ACCUMULOTM

Even years later, Twitter doesn't delete your direct messages

TechCrunch
Feb '19

Small Datum
Jan '20

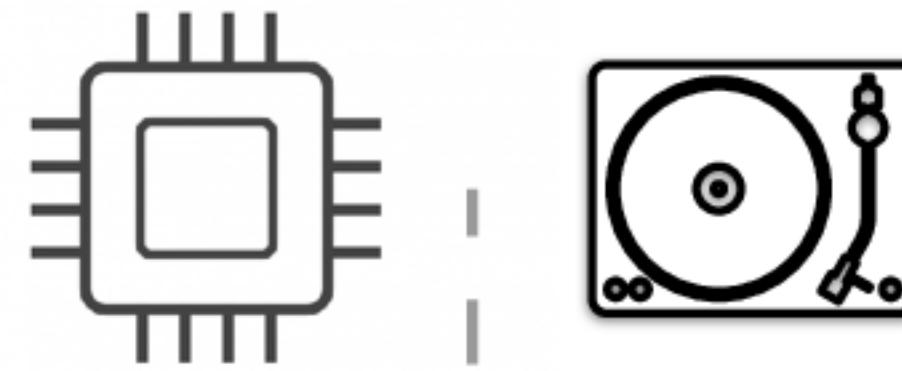
Deletes are fast and slow in an LSM



“LSM-based data stores perform suboptimally for workloads with deletes.”

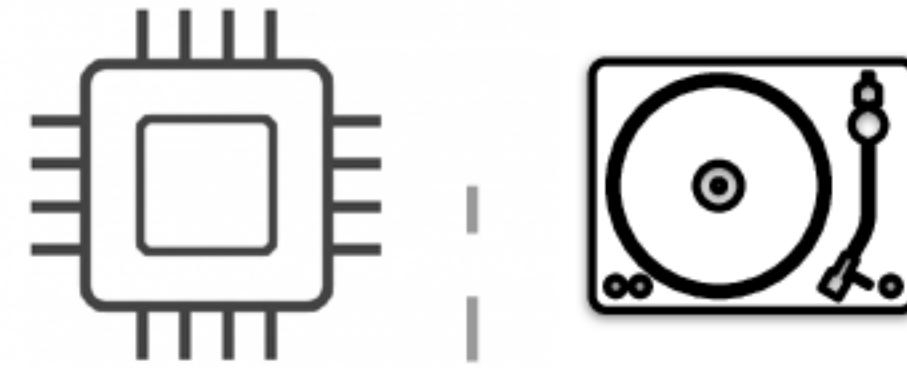
log-structured merge-tree

buffer



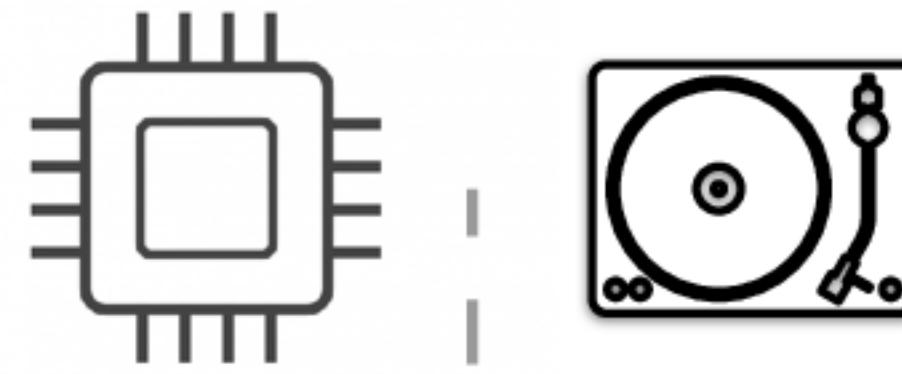
log-structured merge-tree

buffer 



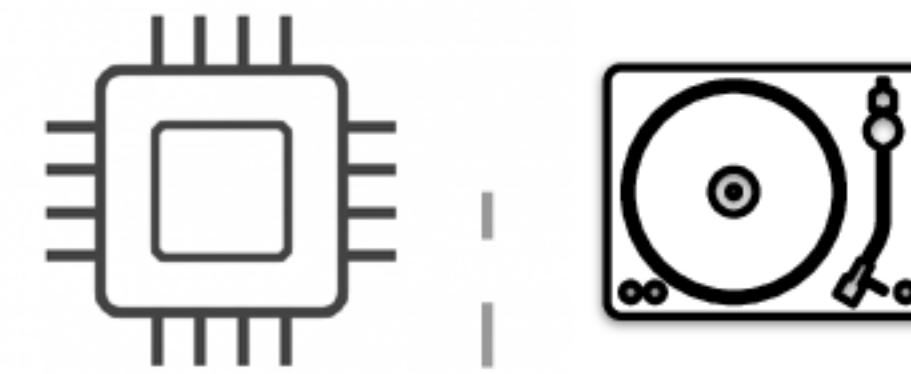
log-structured merge-tree

buffer 



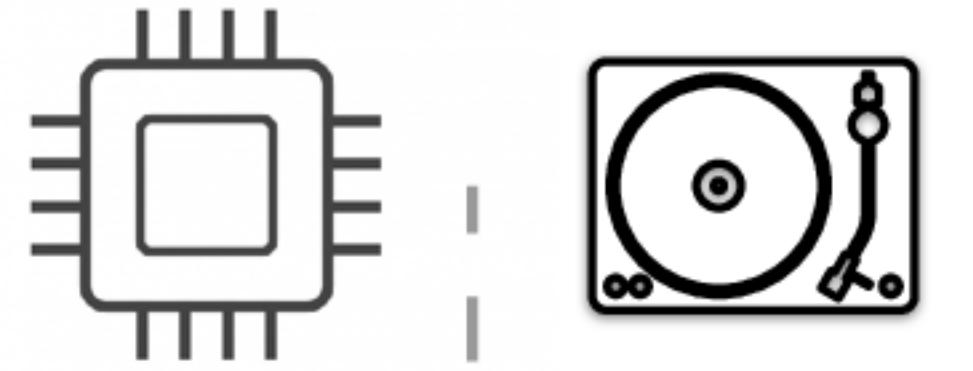
log-structured merge-tree

buffer



log-structured merge-tree

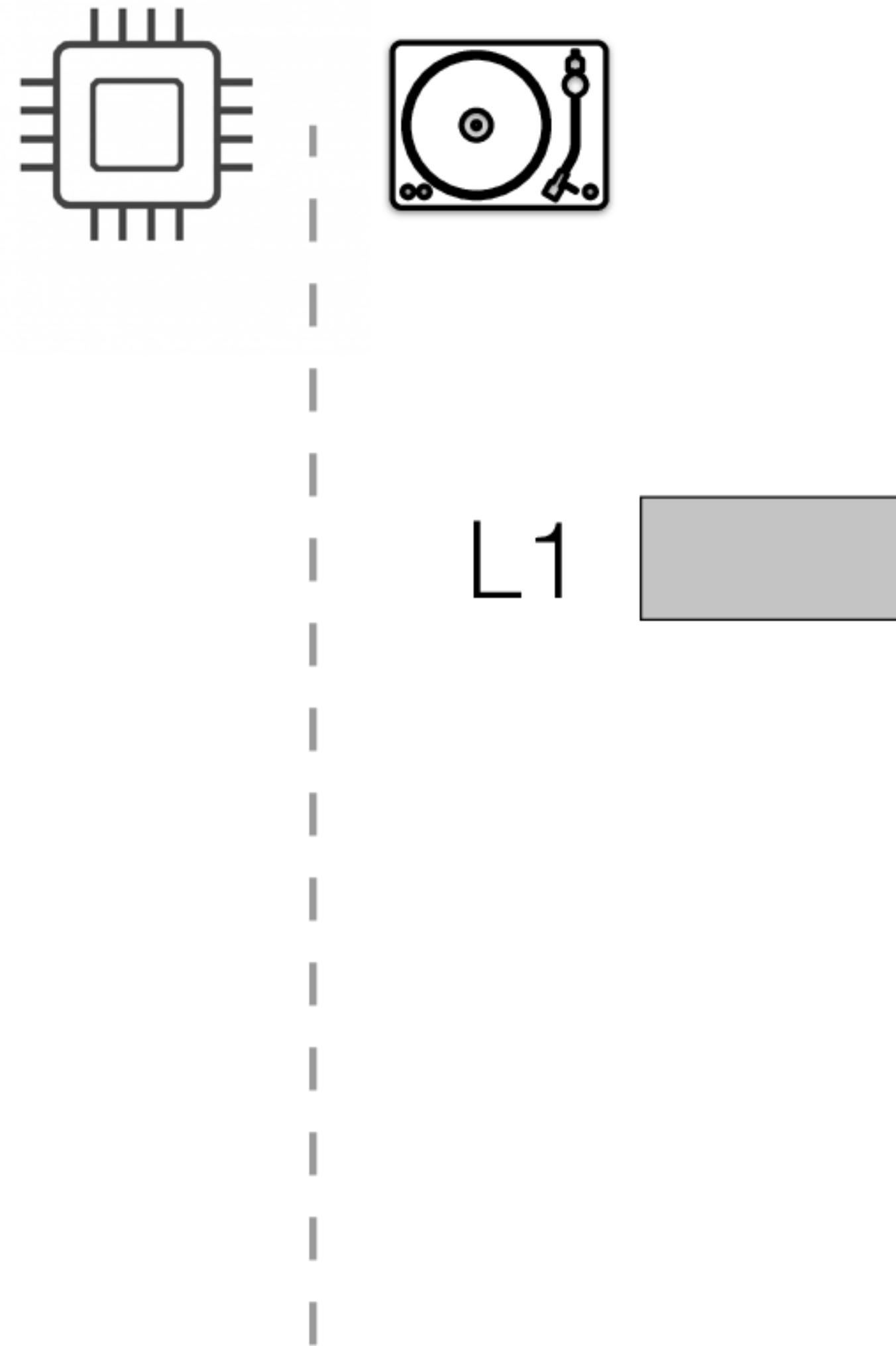
buffer 



L1 

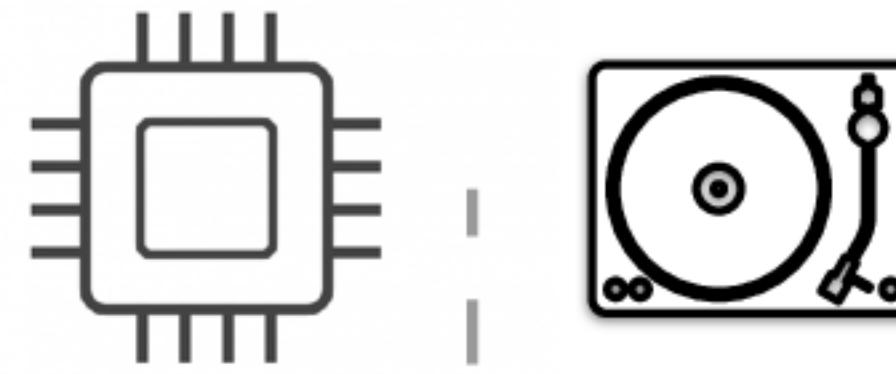
log-structured merge-tree

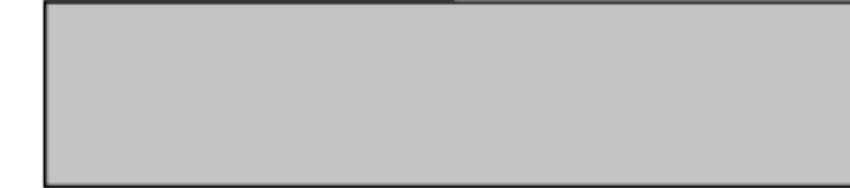
buffer 



log-structured merge-tree

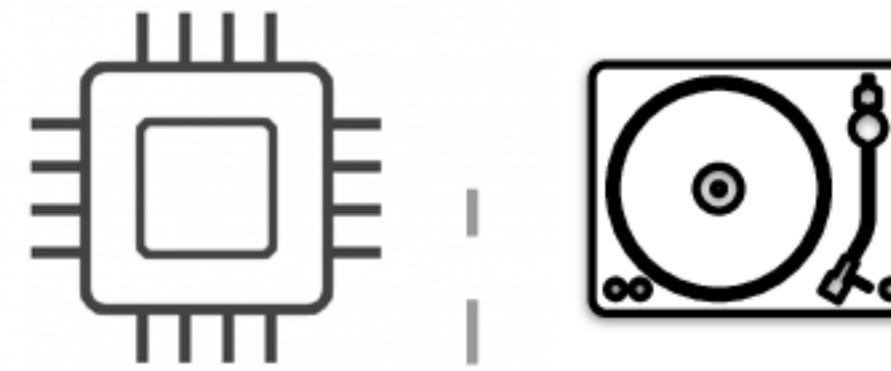
buffer 



L1 

log-structured merge-tree

buffer 



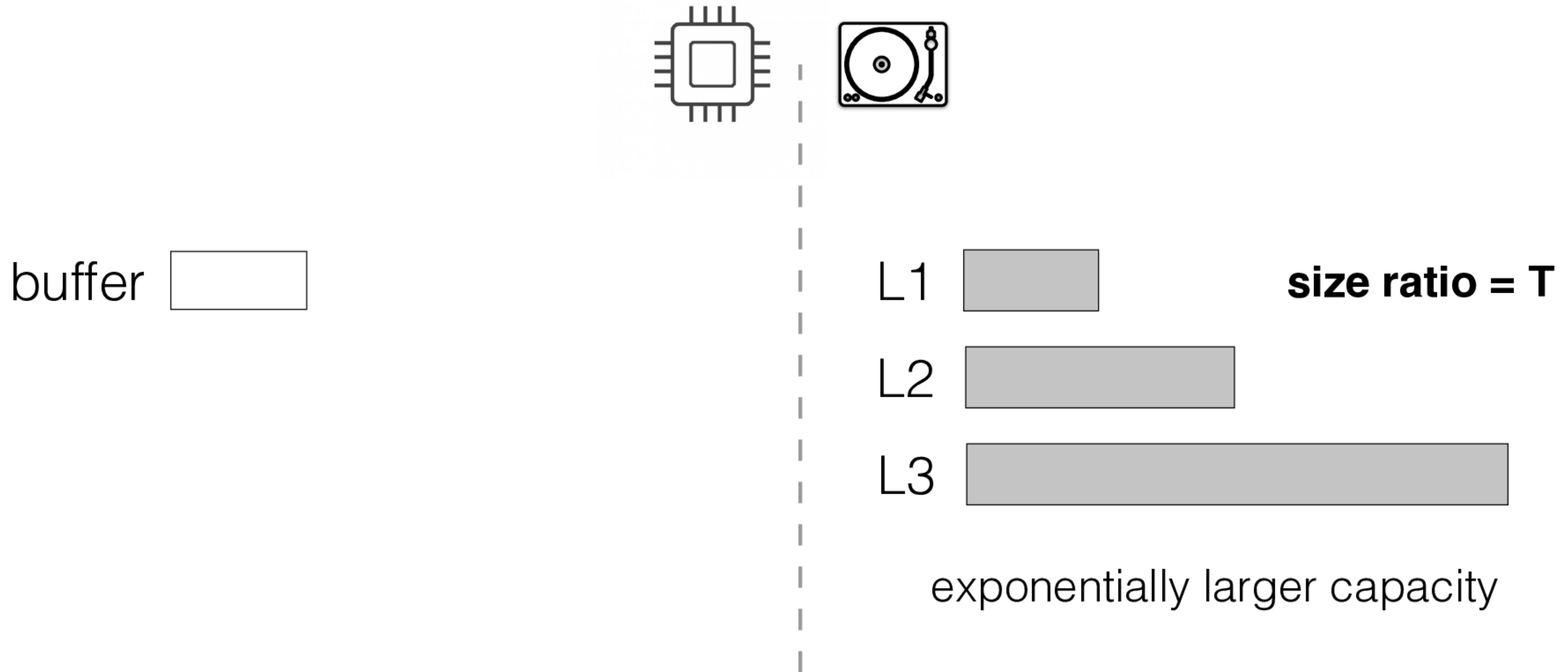
L1

L2



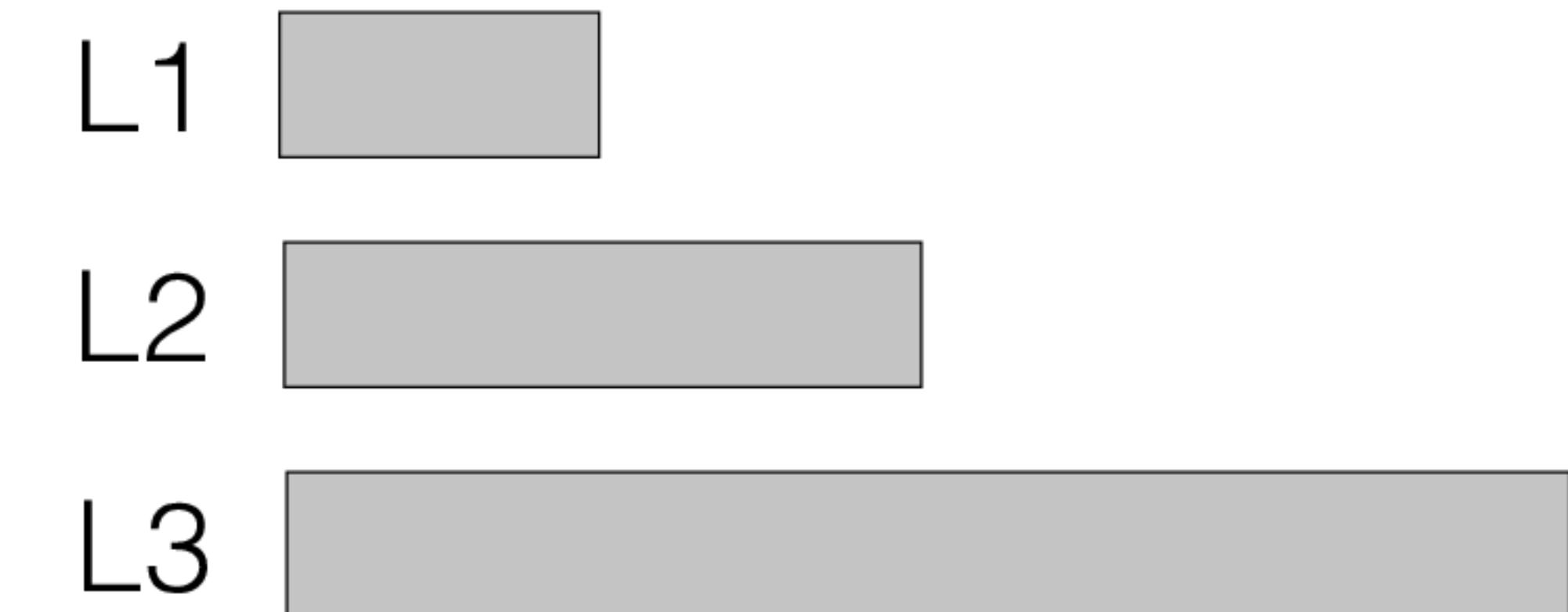
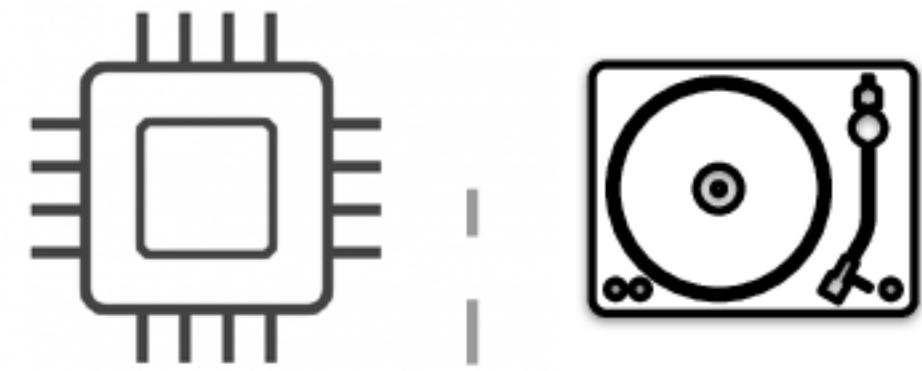
compaction

log-structured merge-tree

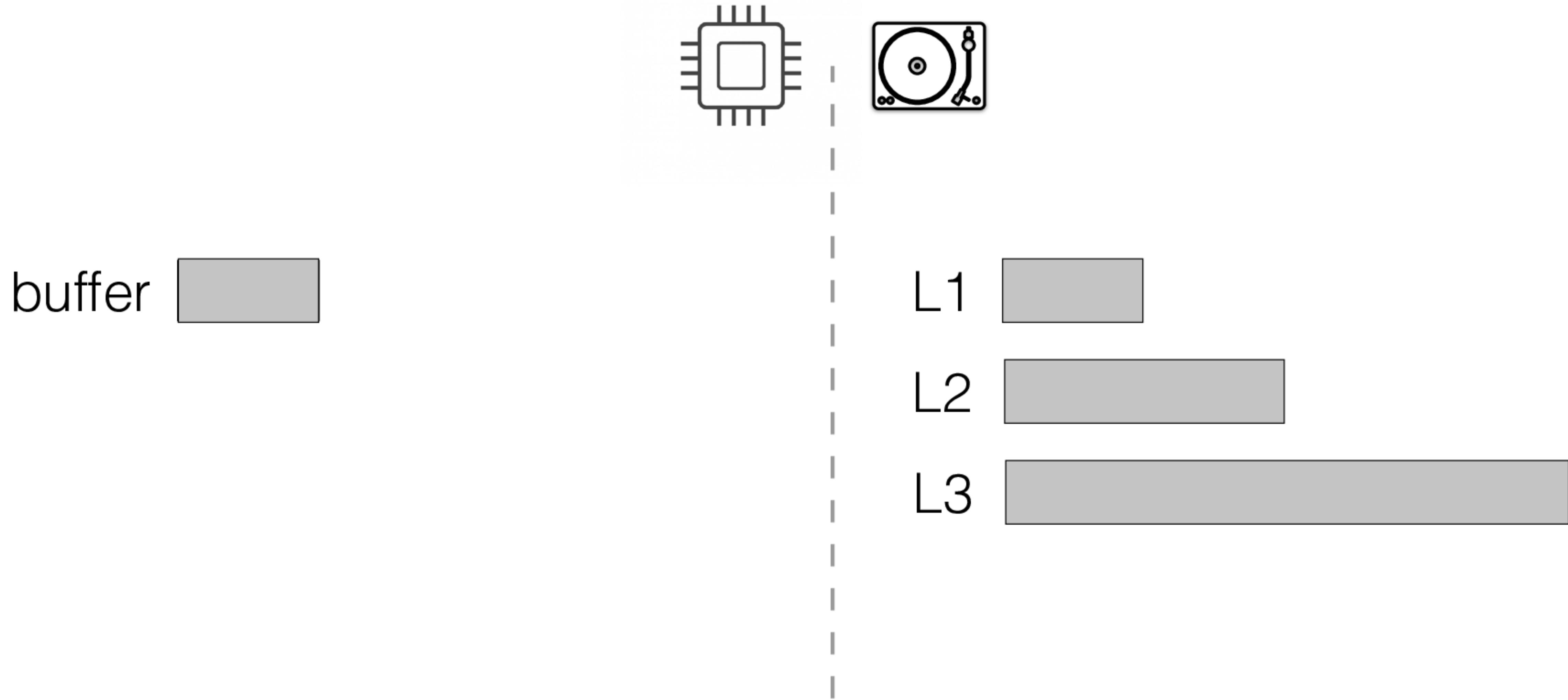


log-structured merge-tree

buffer 

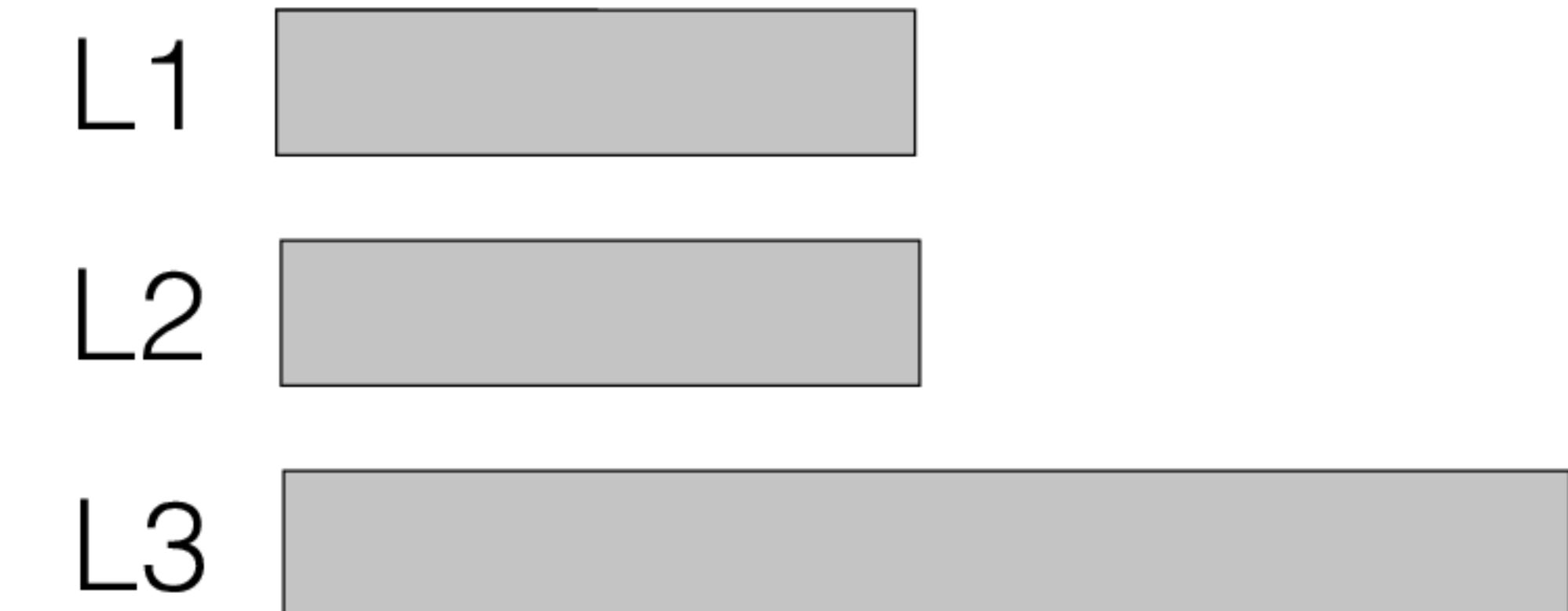
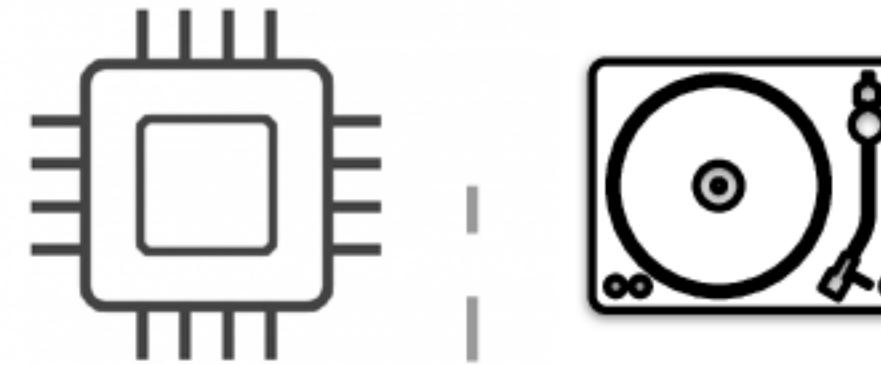


log-structured merge-tree



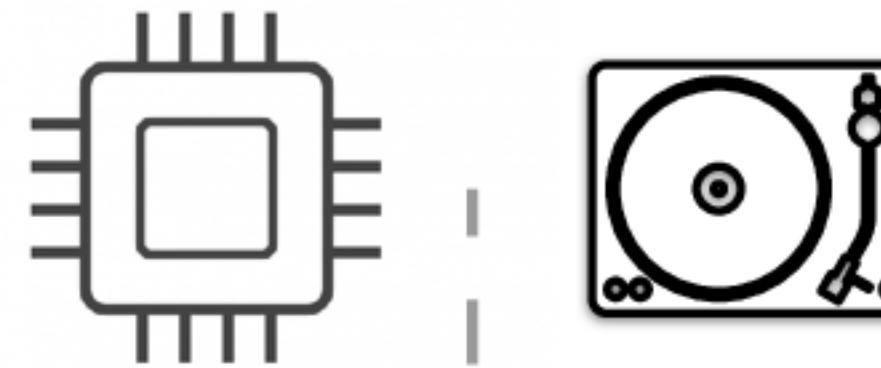
log-structured merge-tree

buffer 



log-structured merge-tree

buffer 



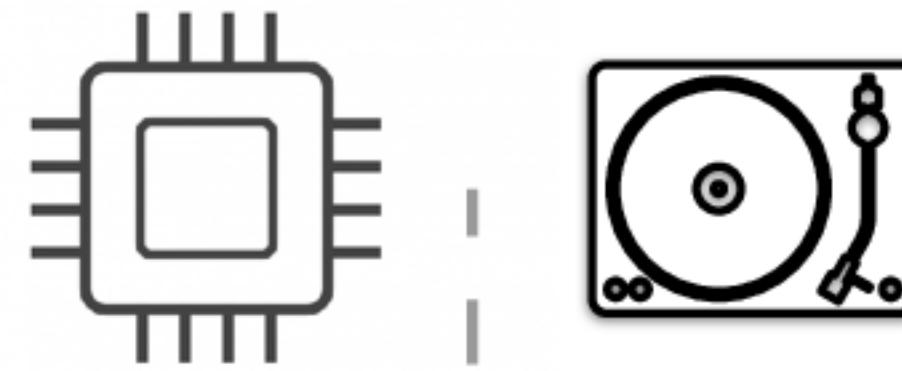
L1

L2 

L3 

log-structured merge-tree

buffer 



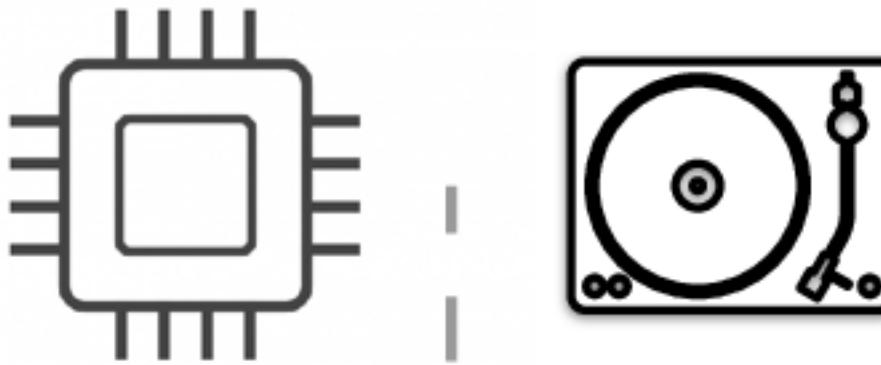
L1

L2

L3 

log-structured merge-tree

buffer 



L1

L2

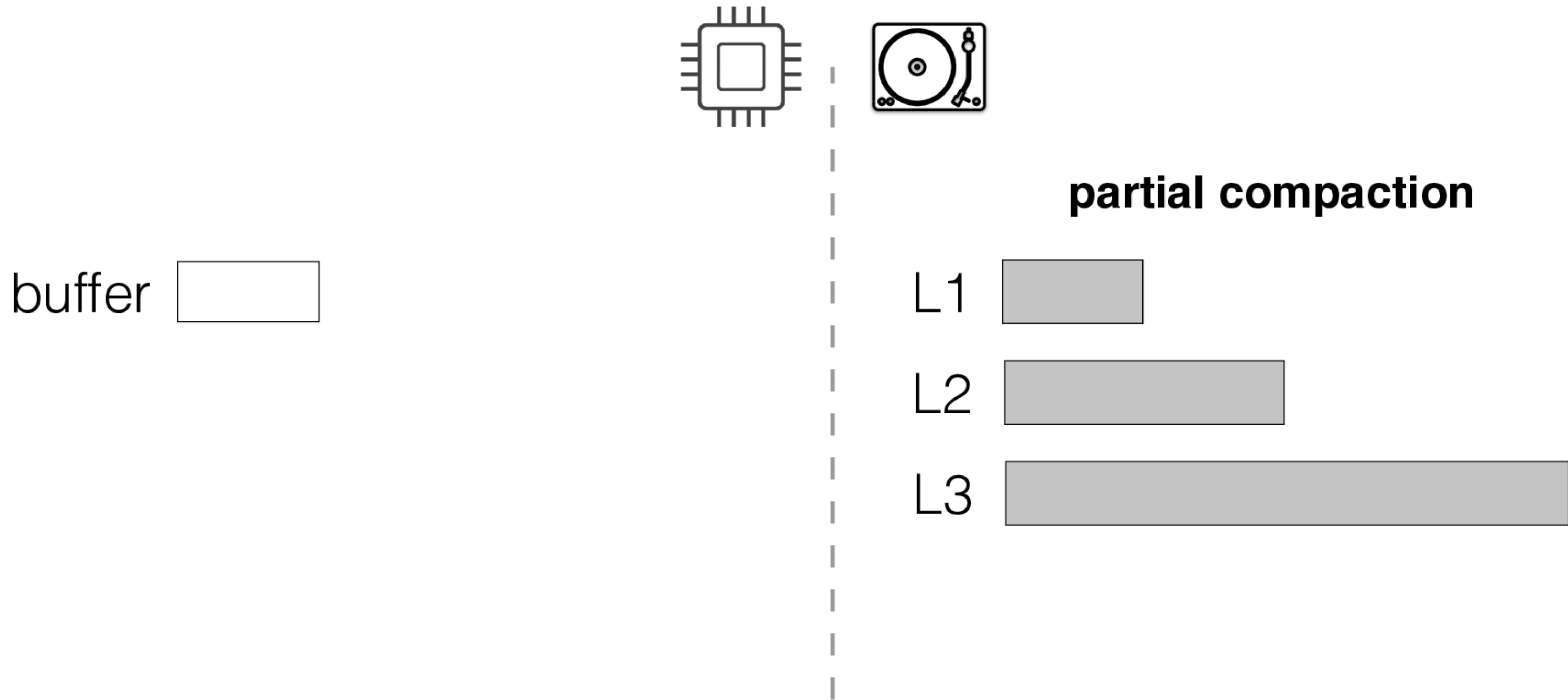
L3

L4

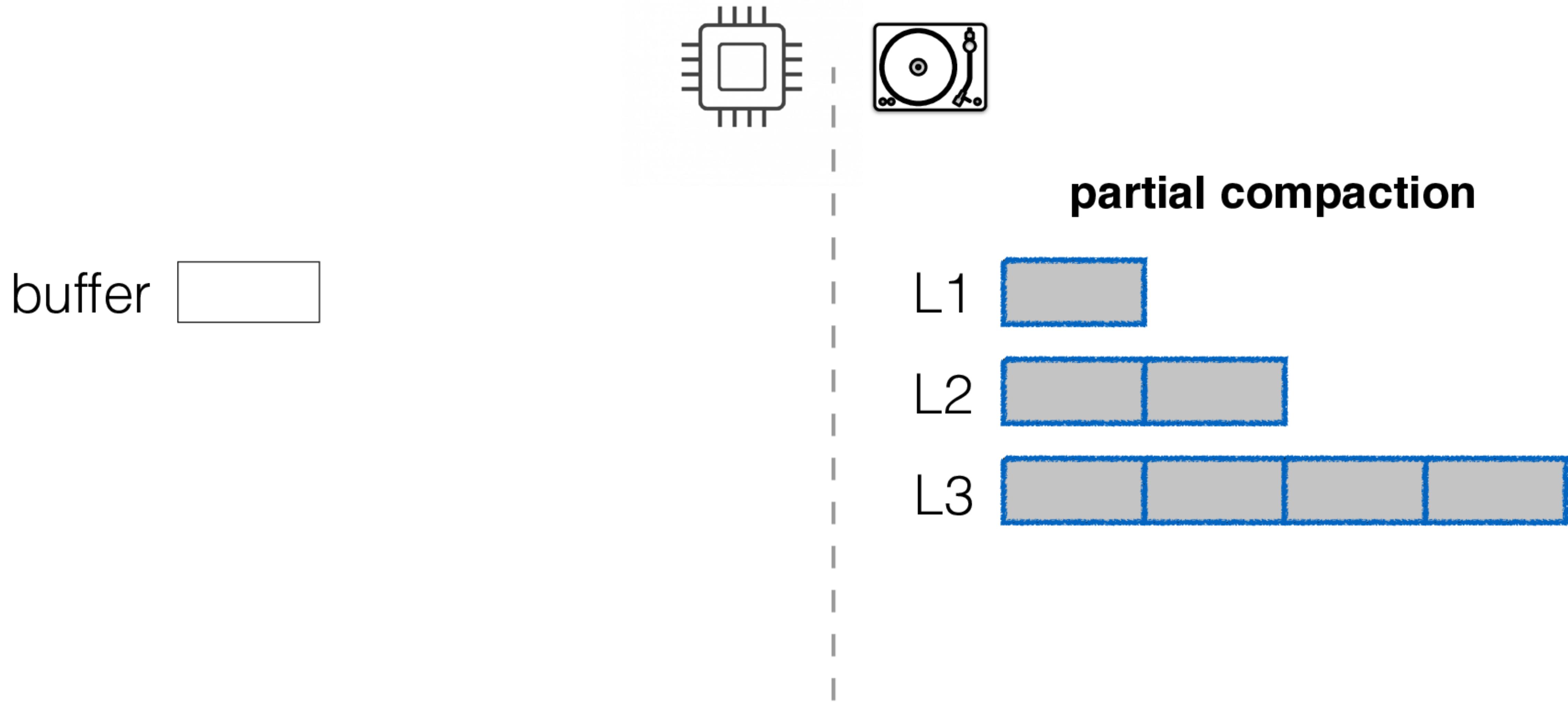
burst of I/Os
prolonged write stalls



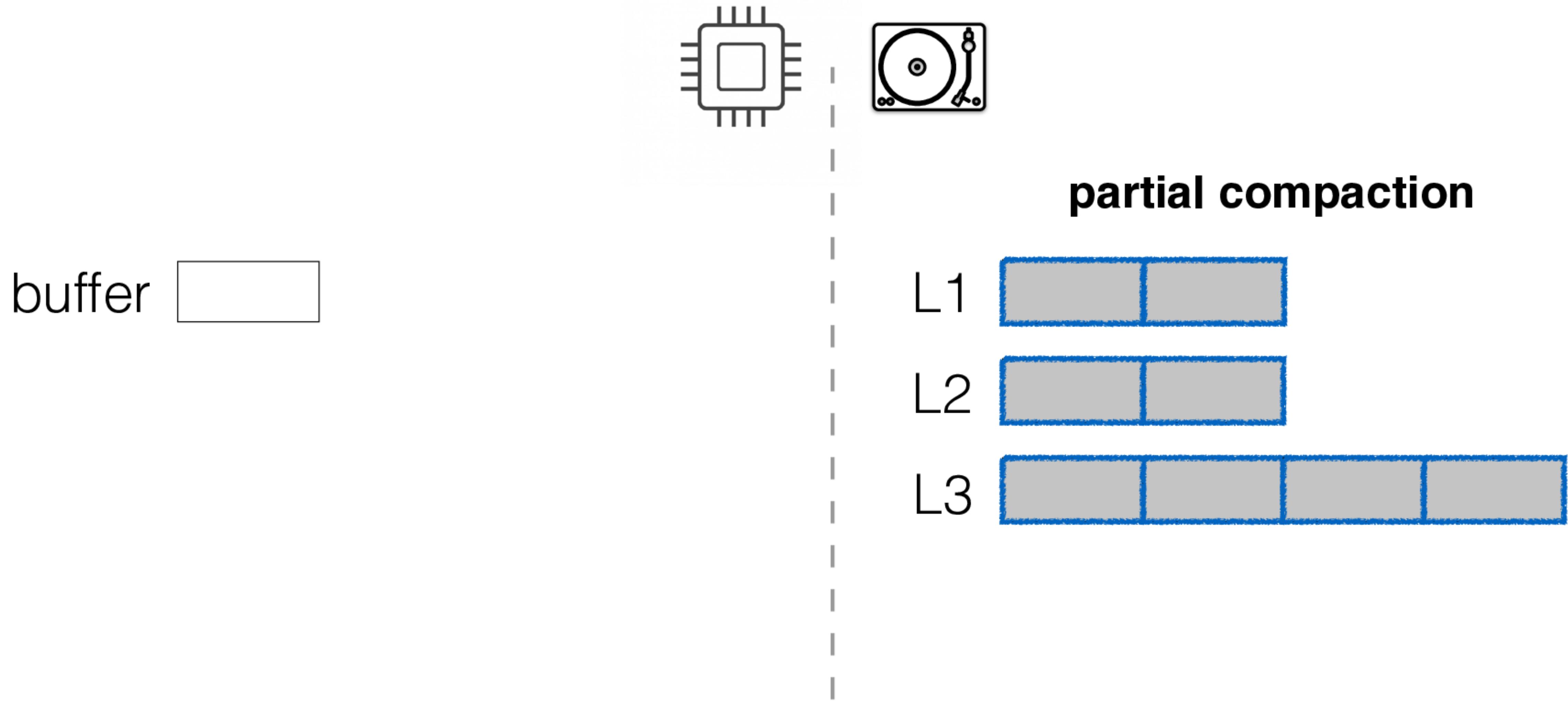
log-structured merge-tree



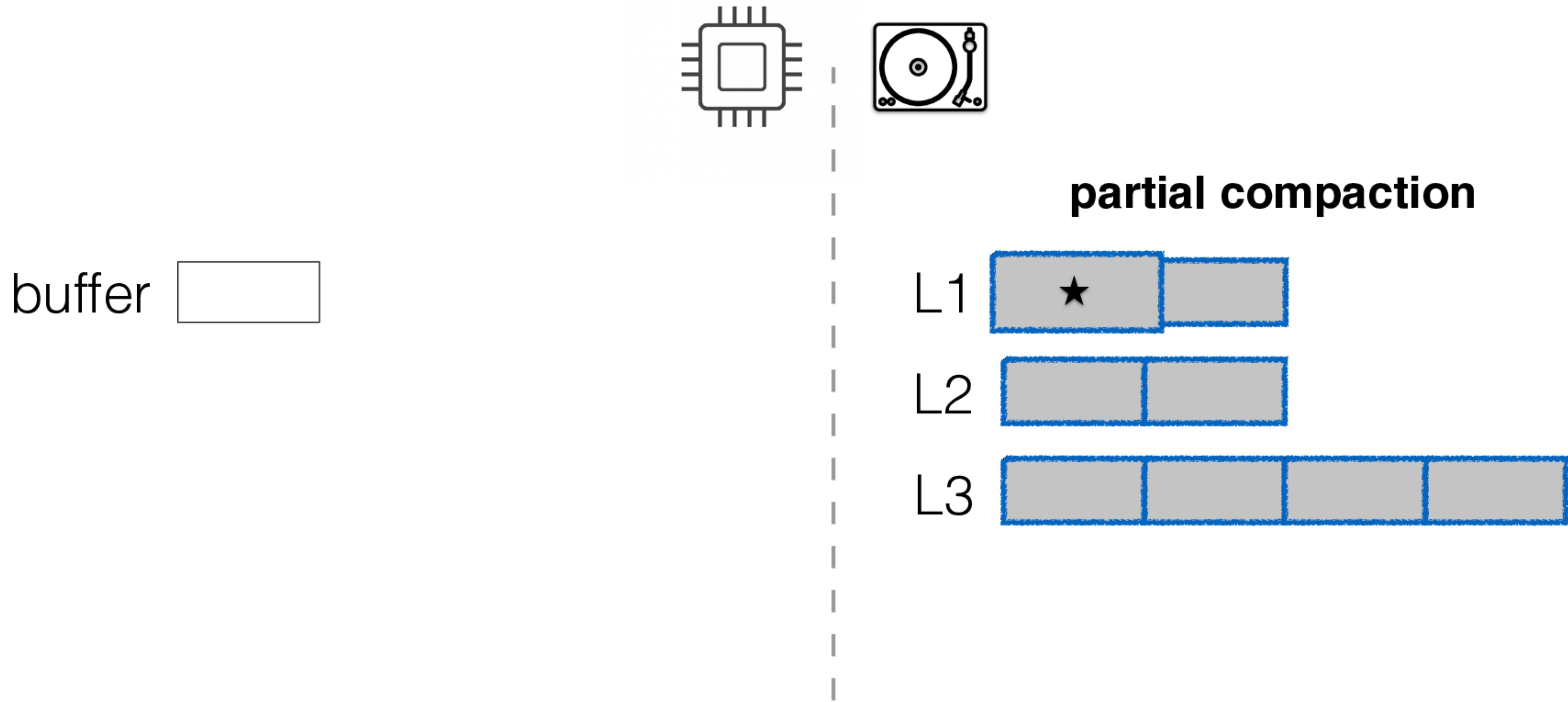
log-structured merge-tree



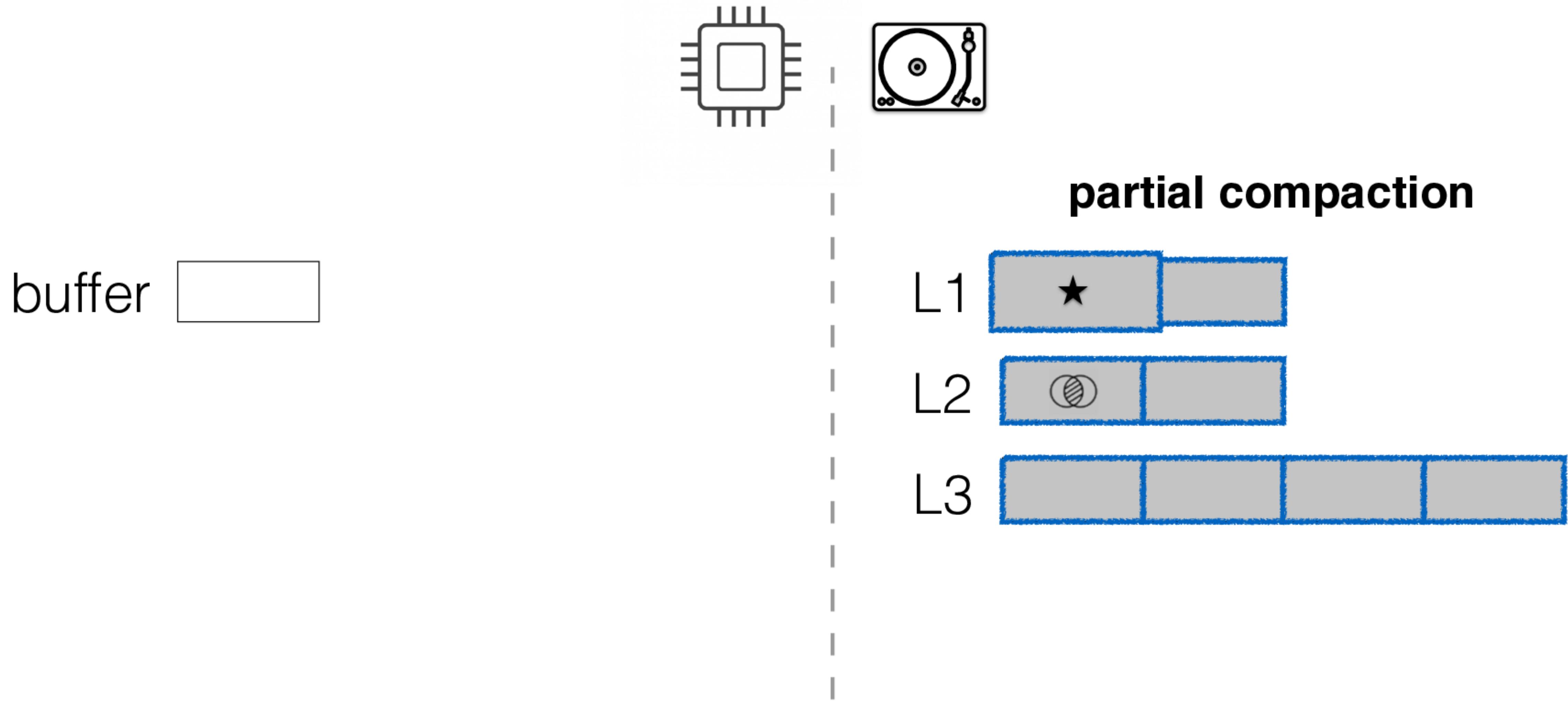
log-structured merge-tree



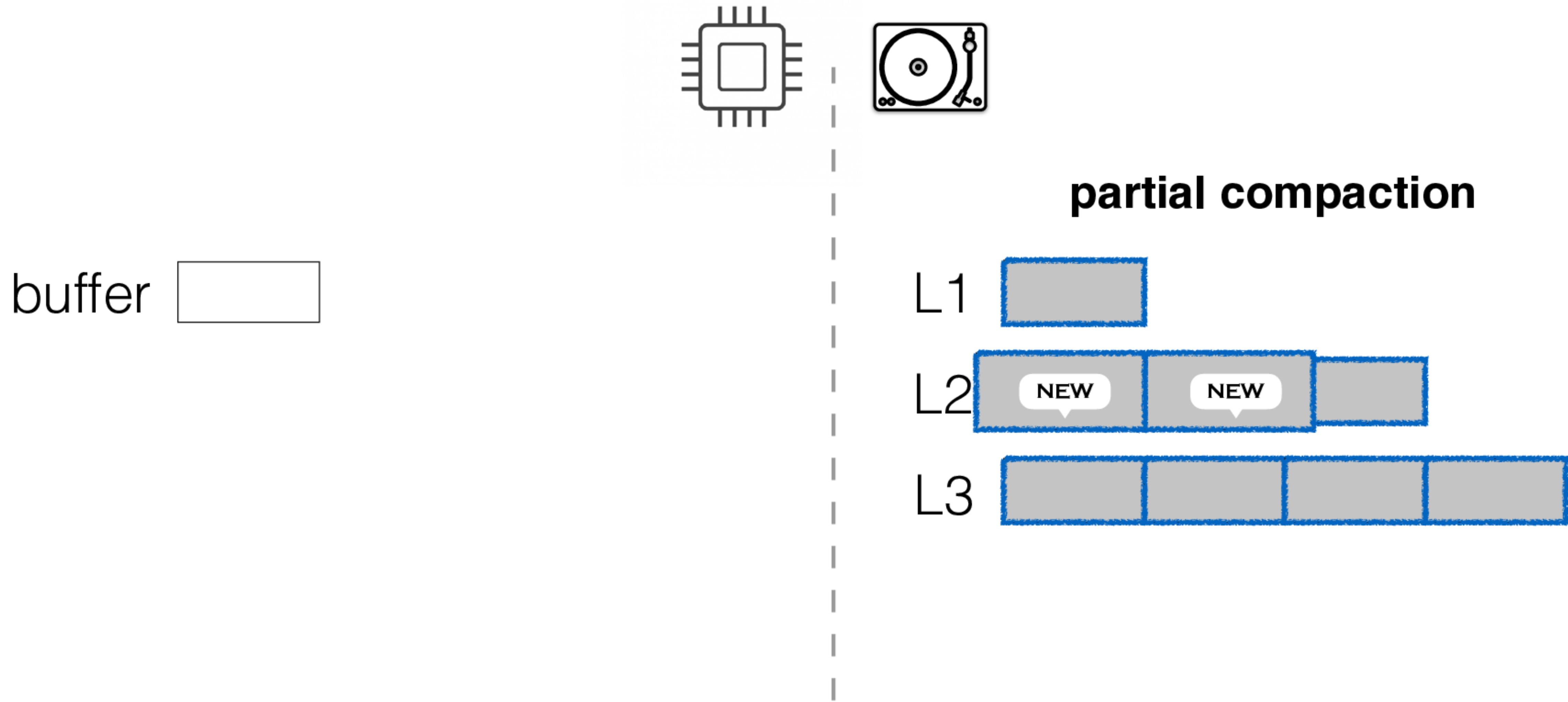
log-structured merge-tree



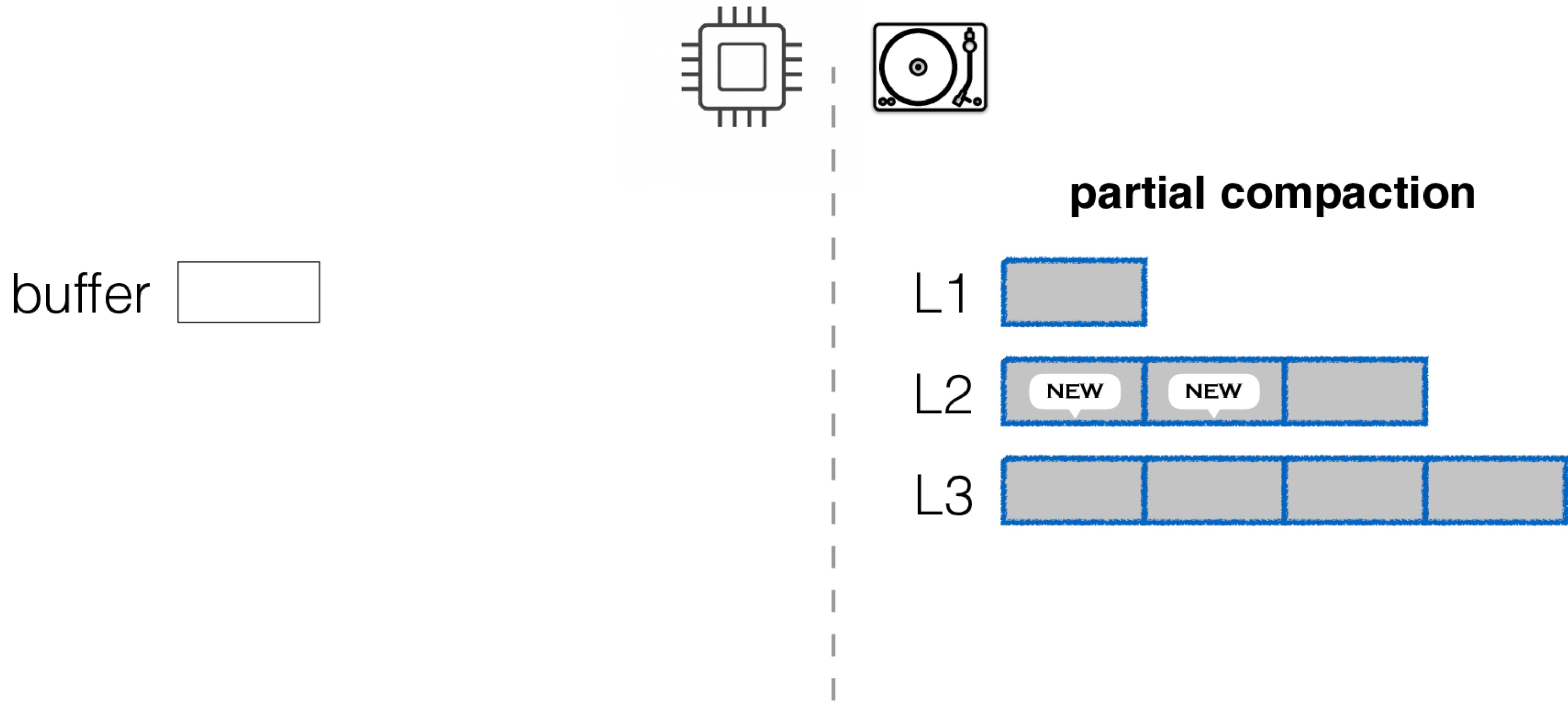
log-structured merge-tree



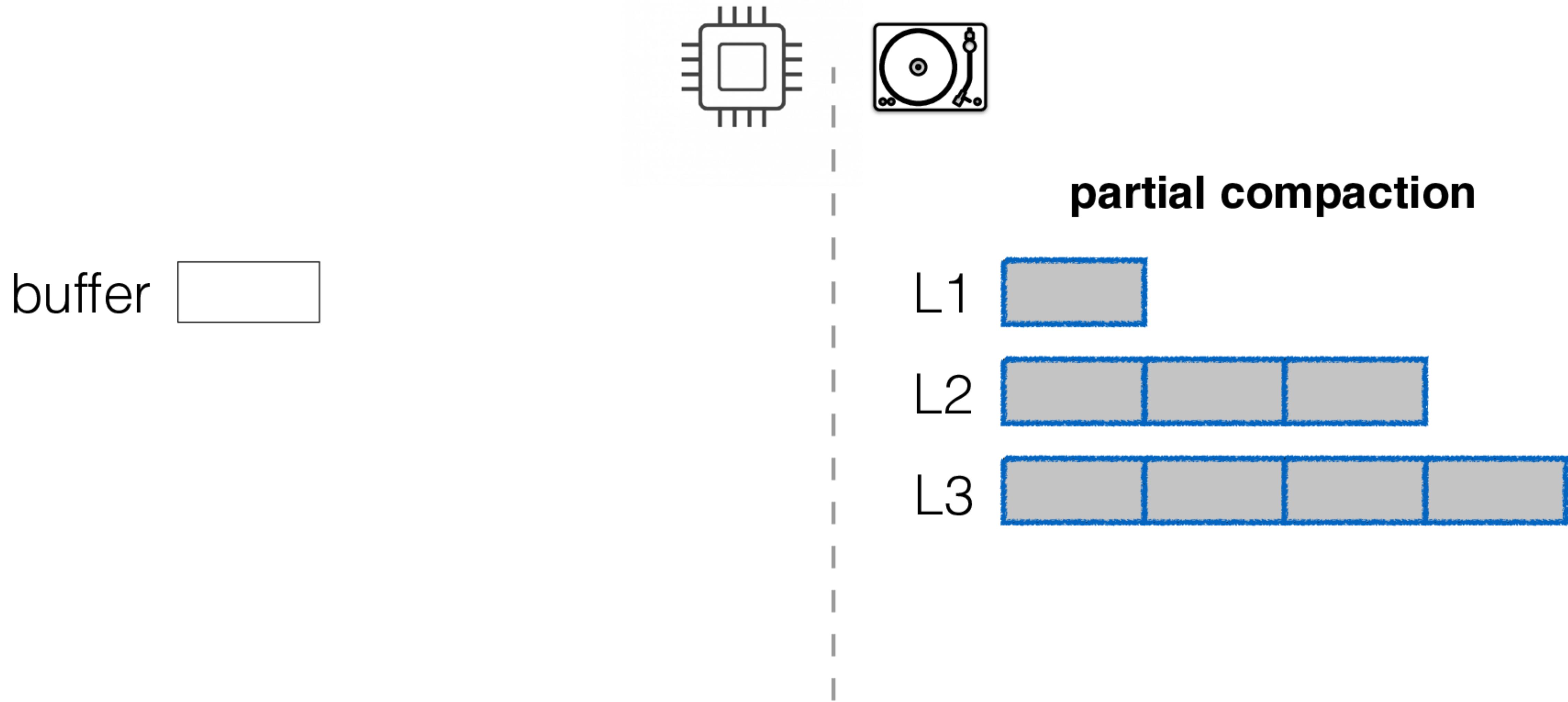
log-structured merge-tree



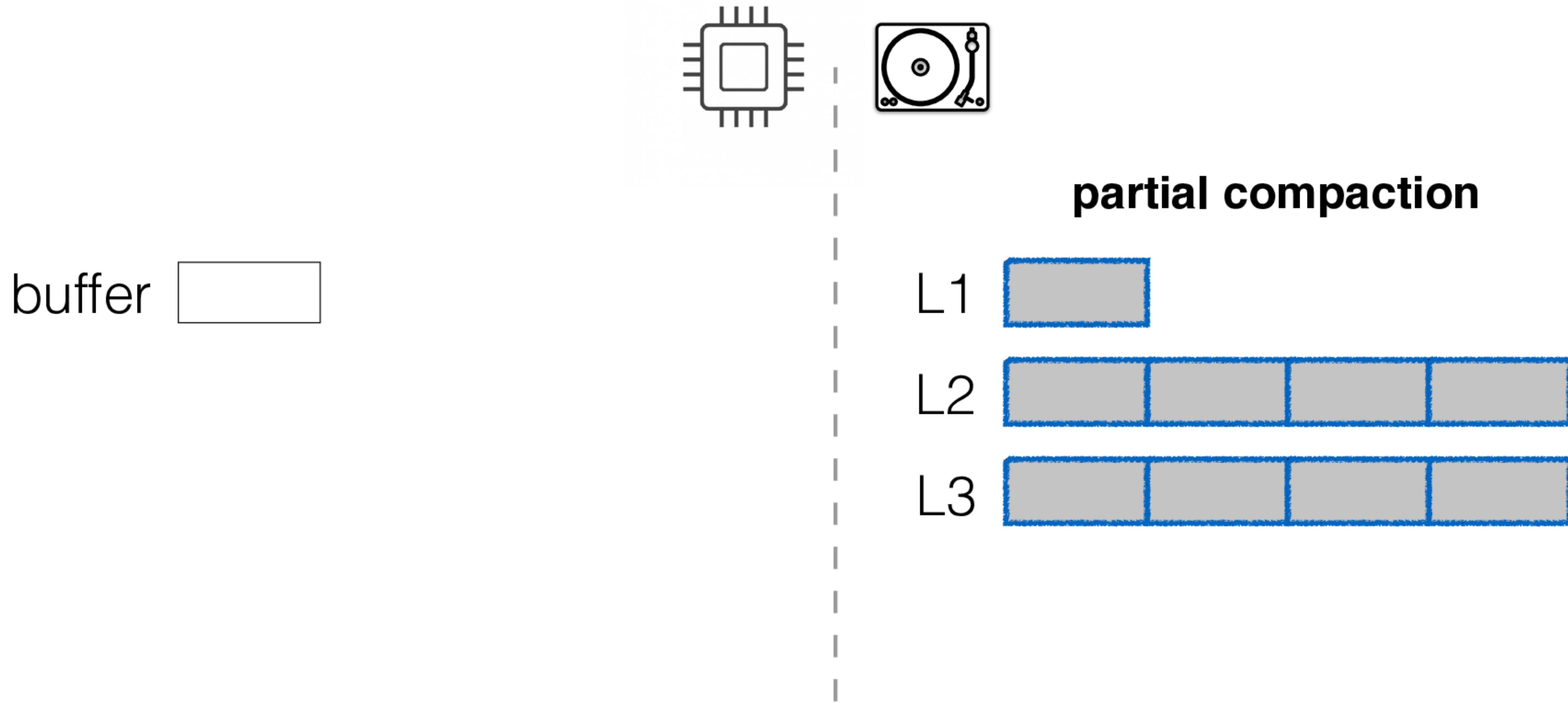
log-structured merge-tree



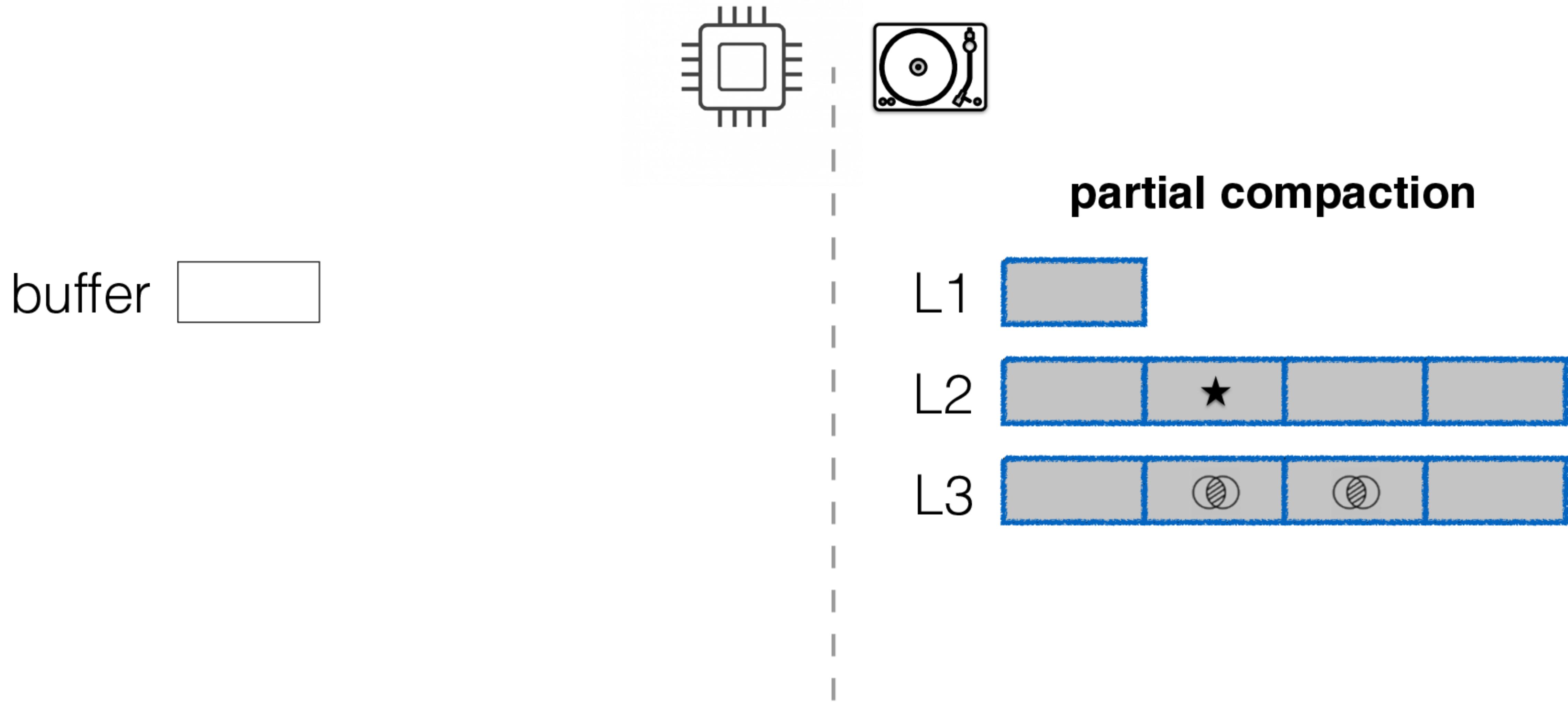
log-structured merge-tree



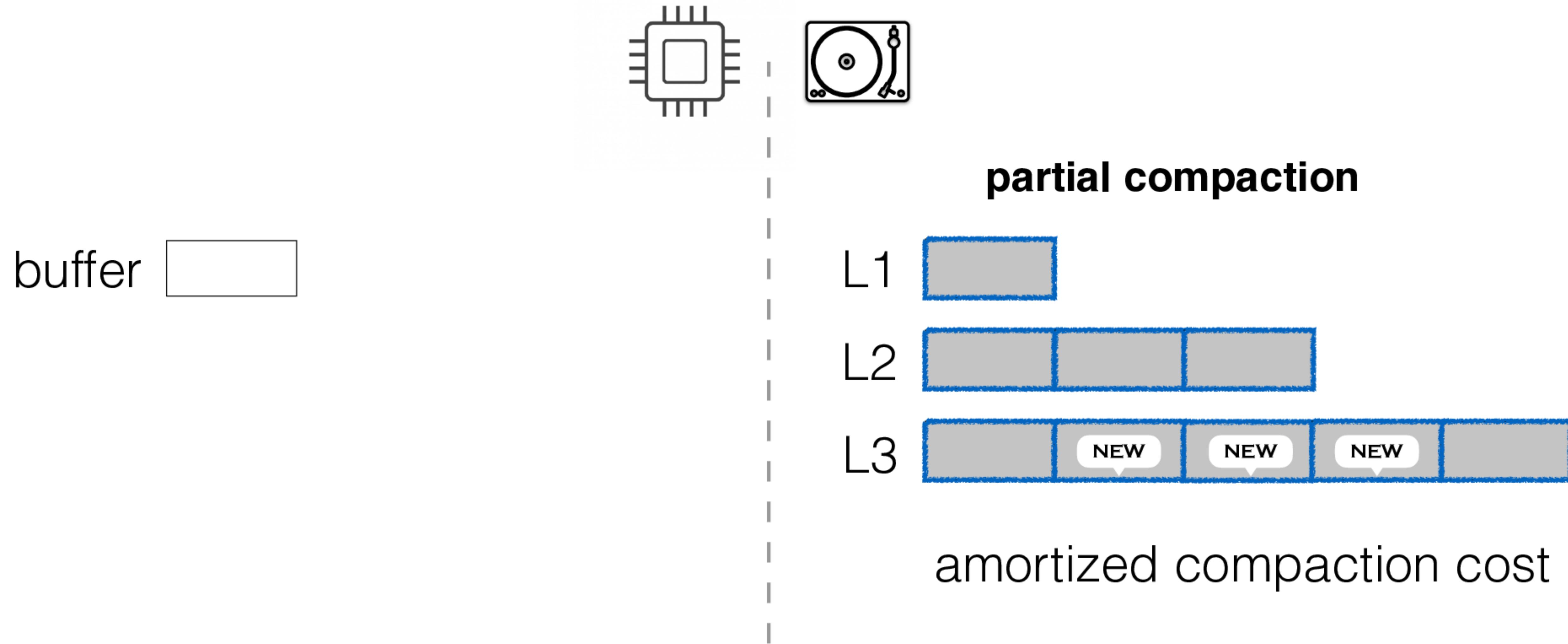
log-structured merge-tree



log-structured merge-tree

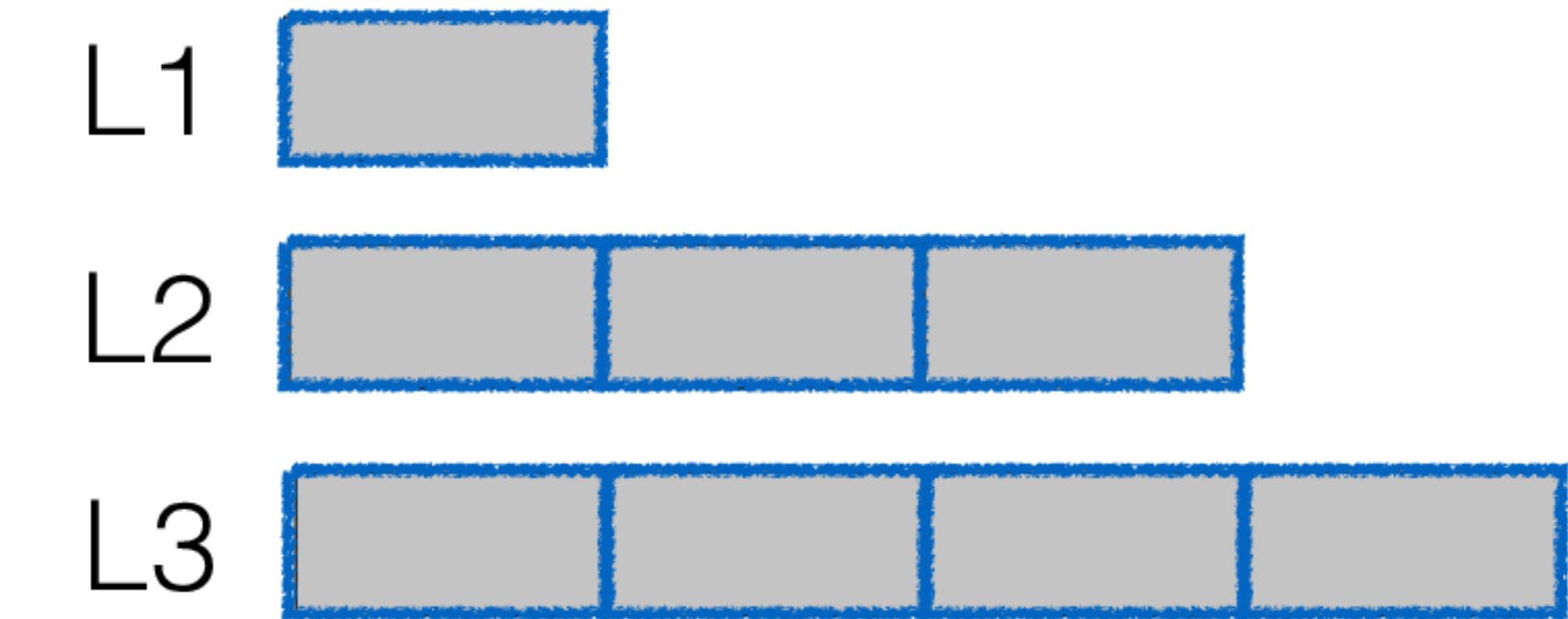
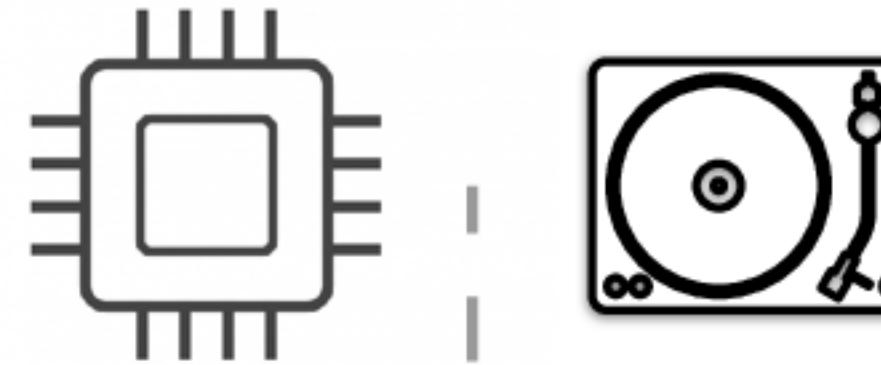


log-structured merge-tree

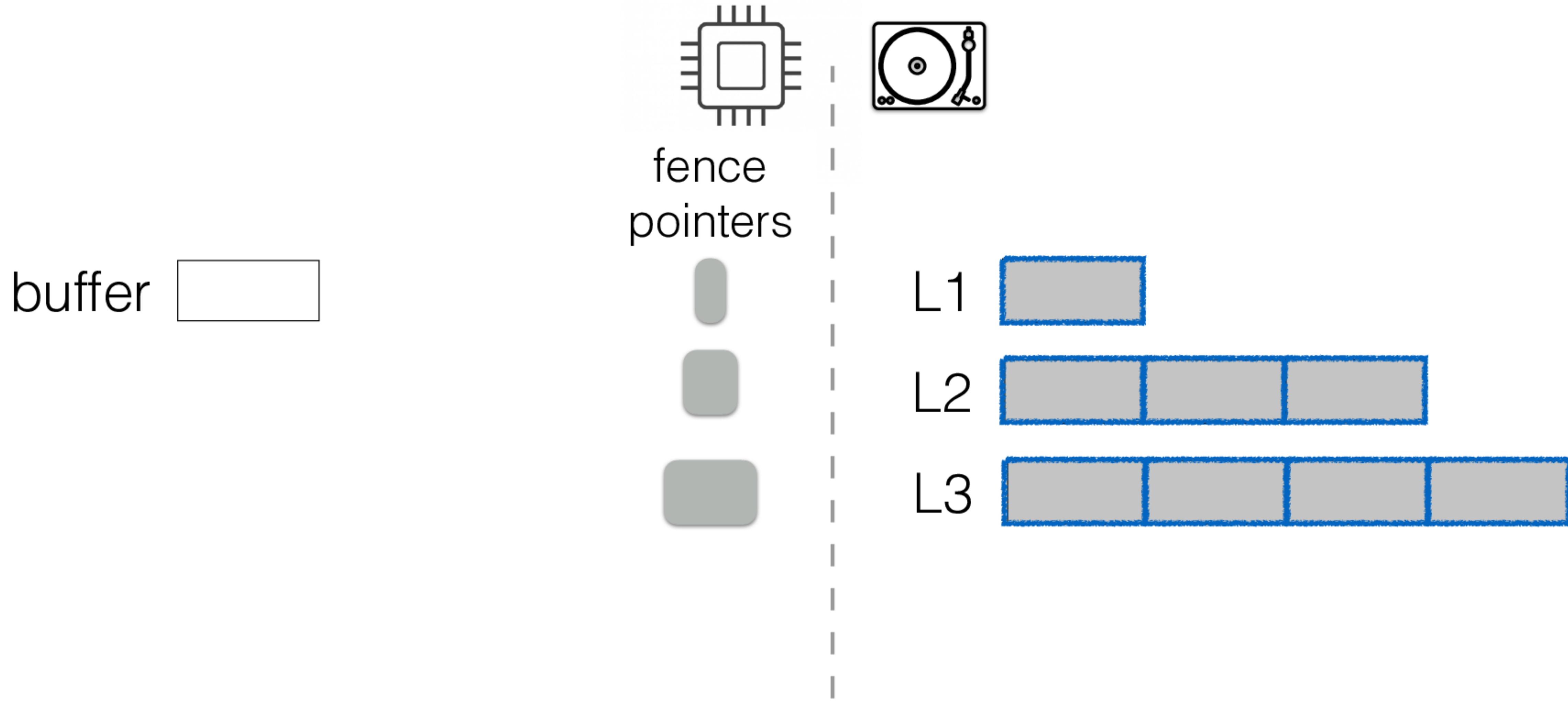


log-structured merge-tree

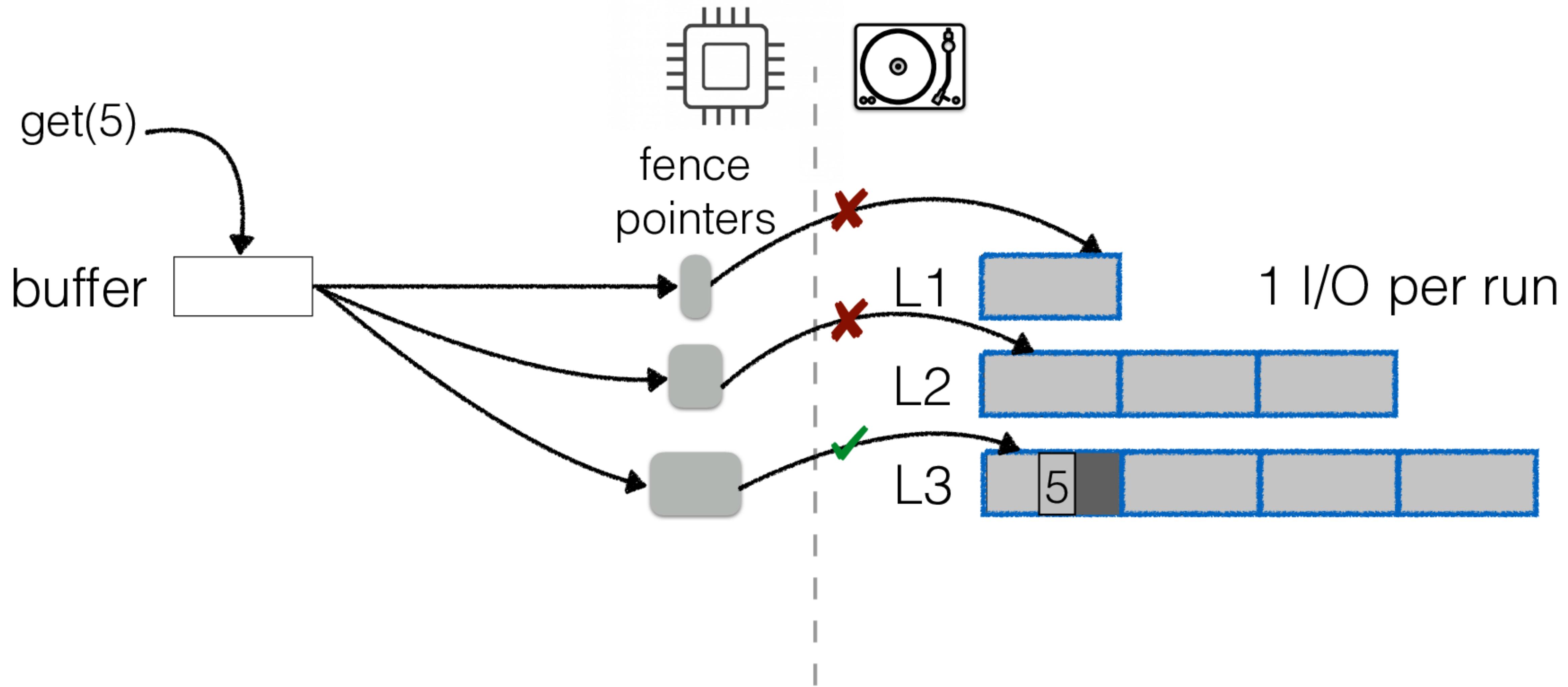
buffer 



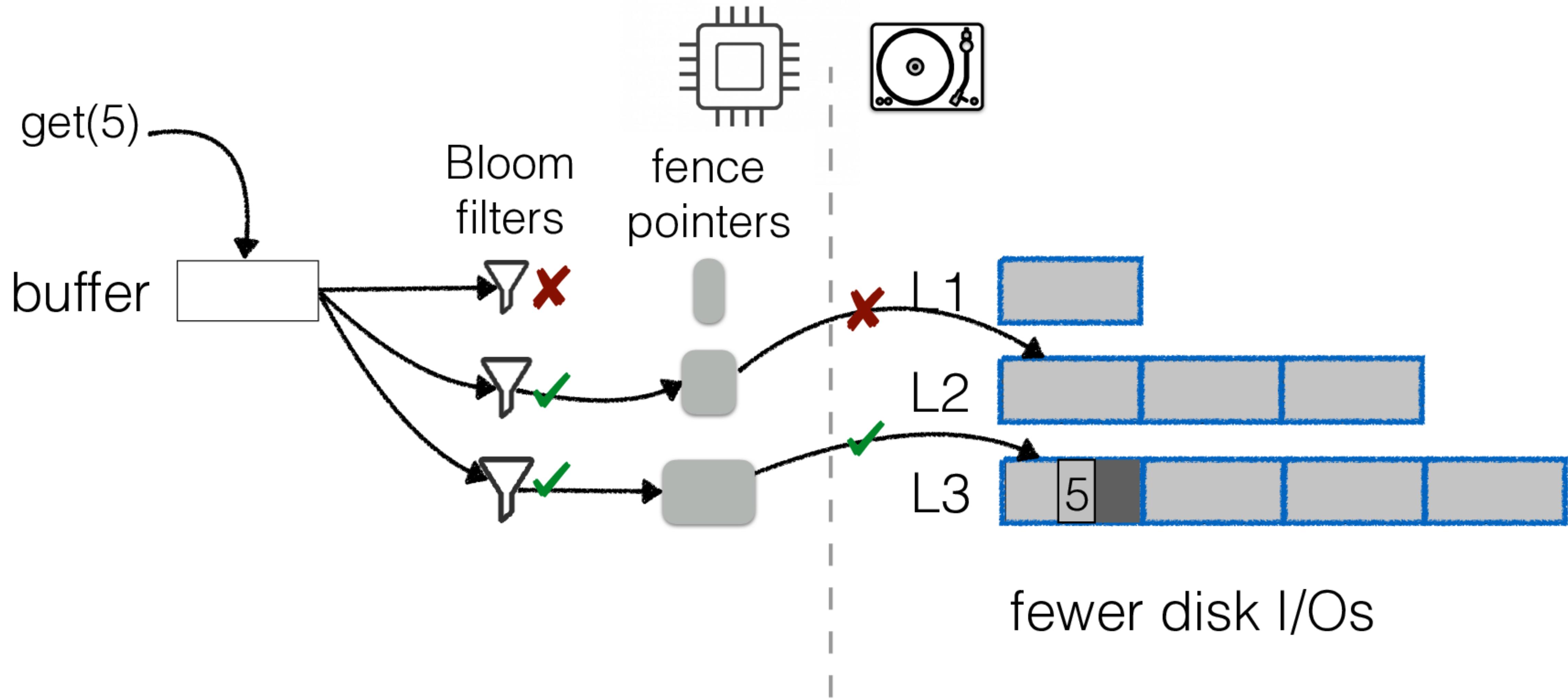
log-structured merge-tree



log-structured merge-tree



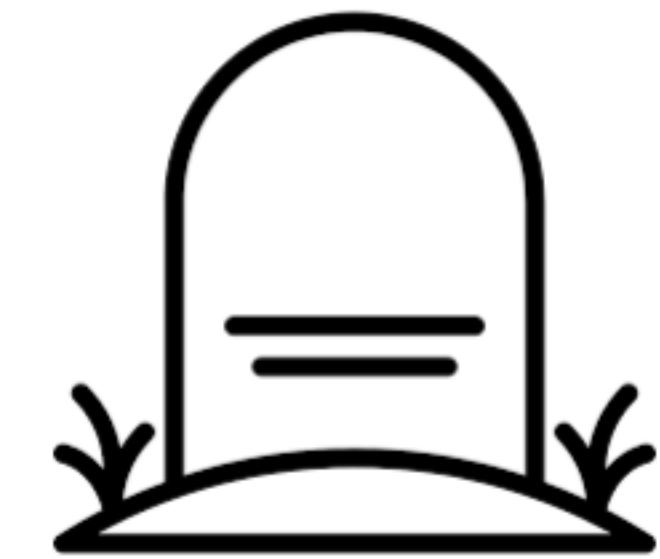
log-structured merge-tree



Now, let's talk about deletes!

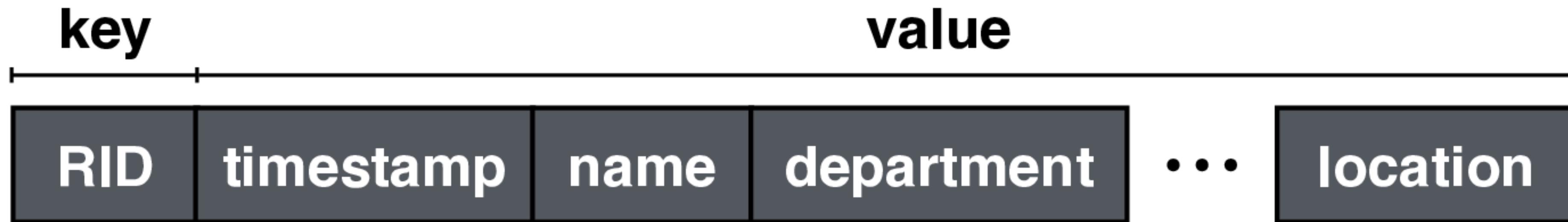
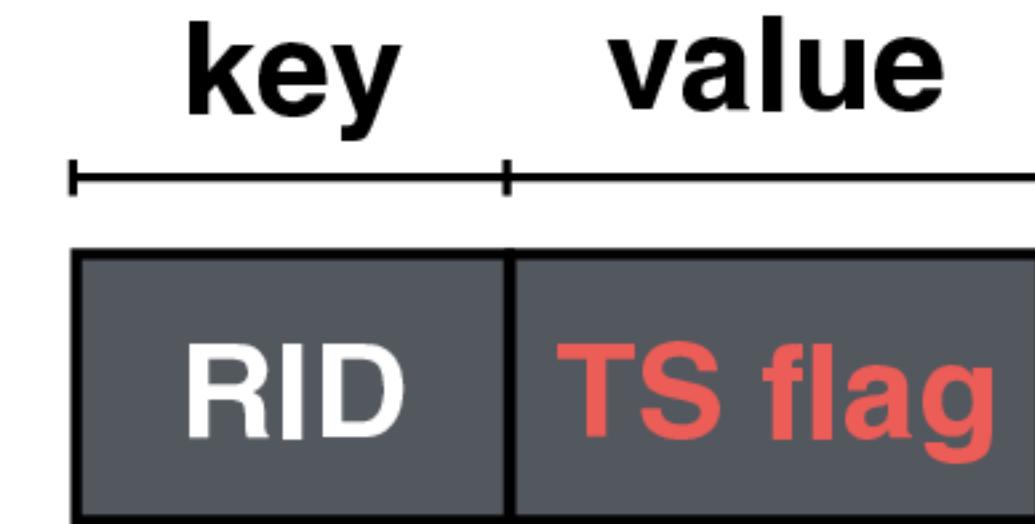
deletes in LSM-tree

delete



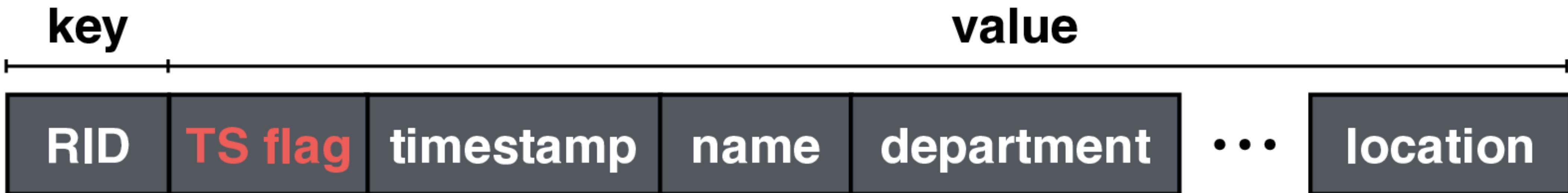
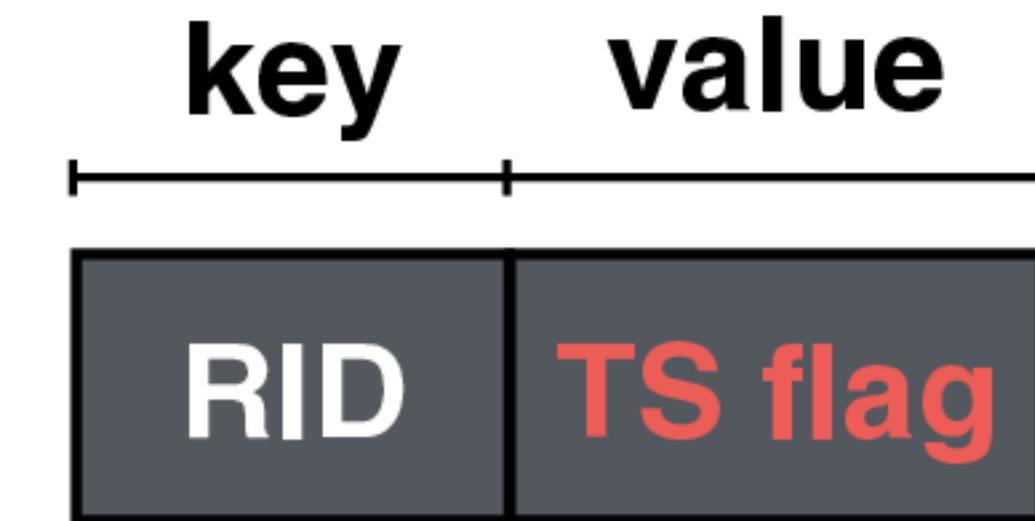
deletes in LSM-tree

delete := insert tombstone

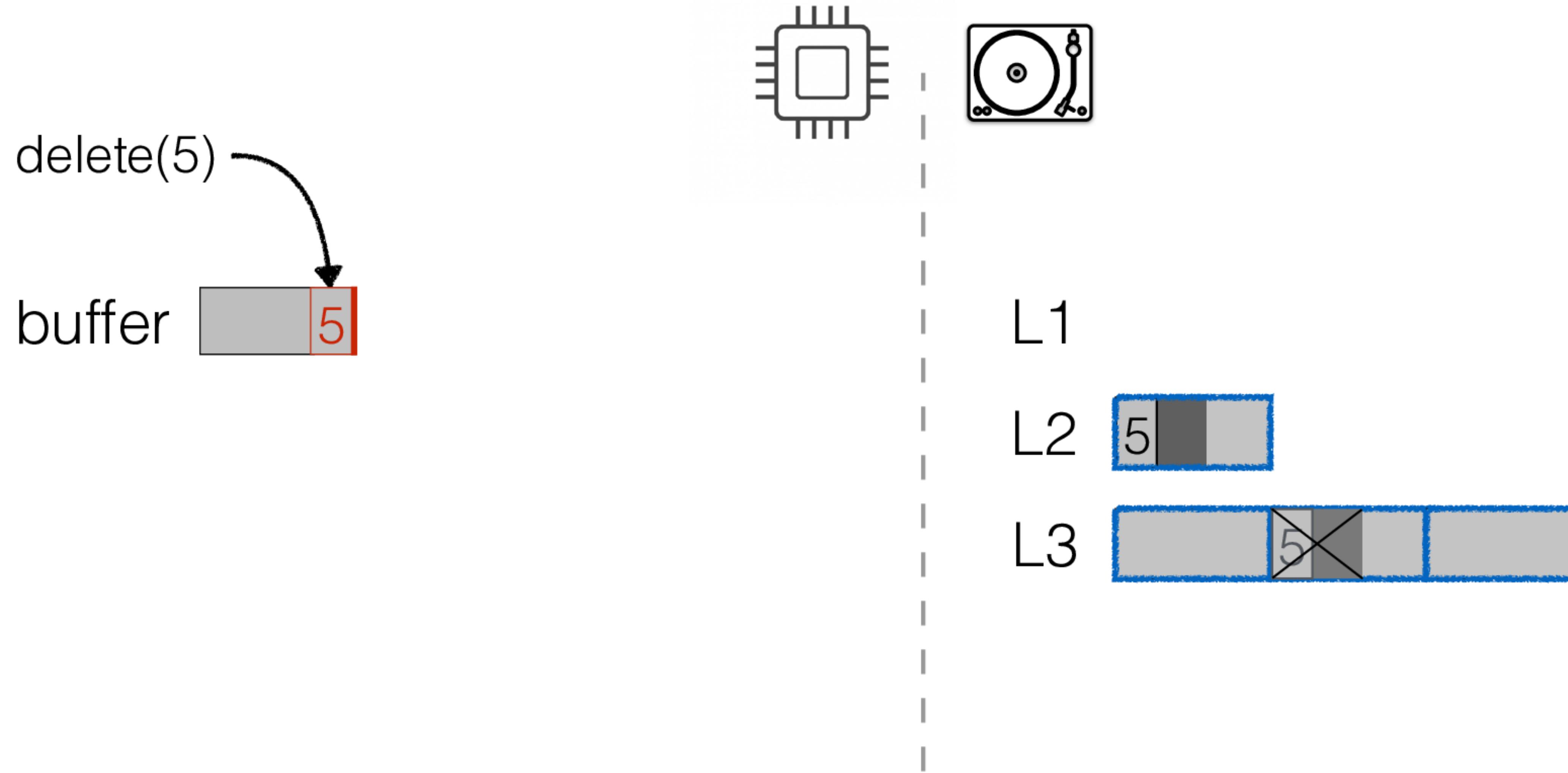


deletes in LSM-tree

delete := insert tombstone

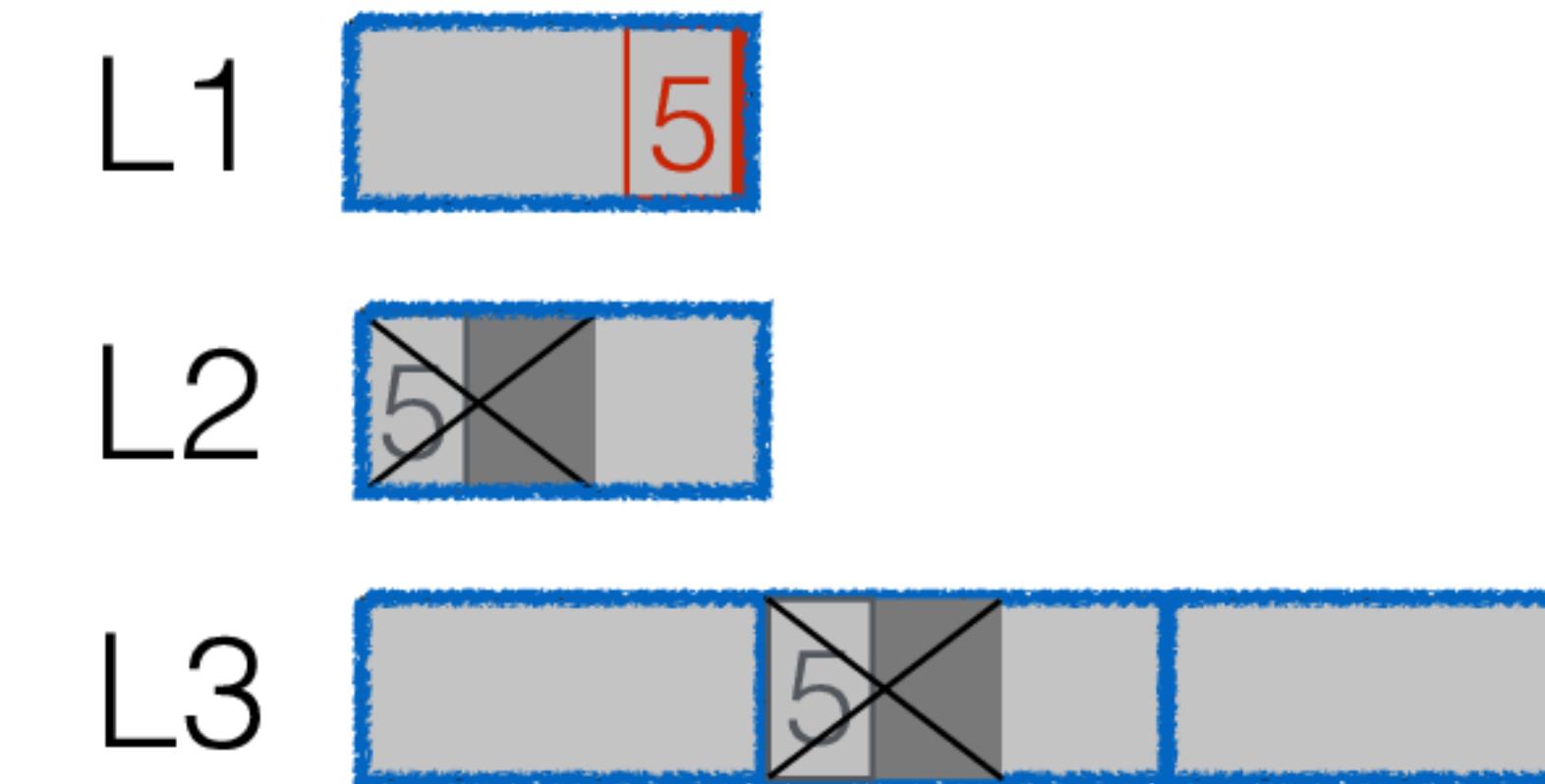
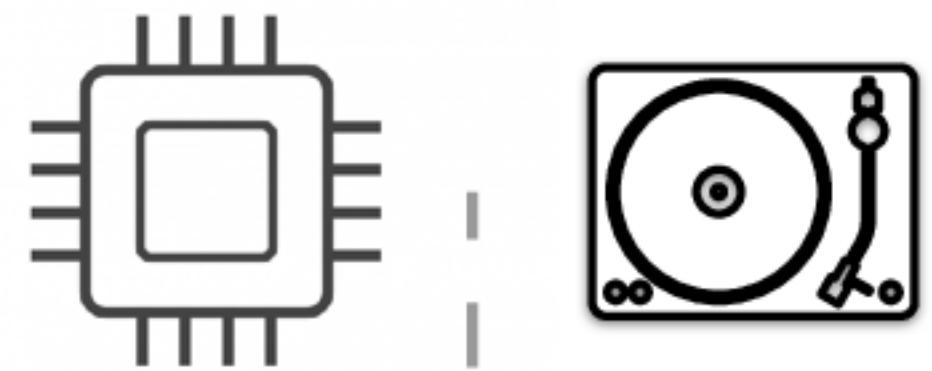


deletes in LSM-tree

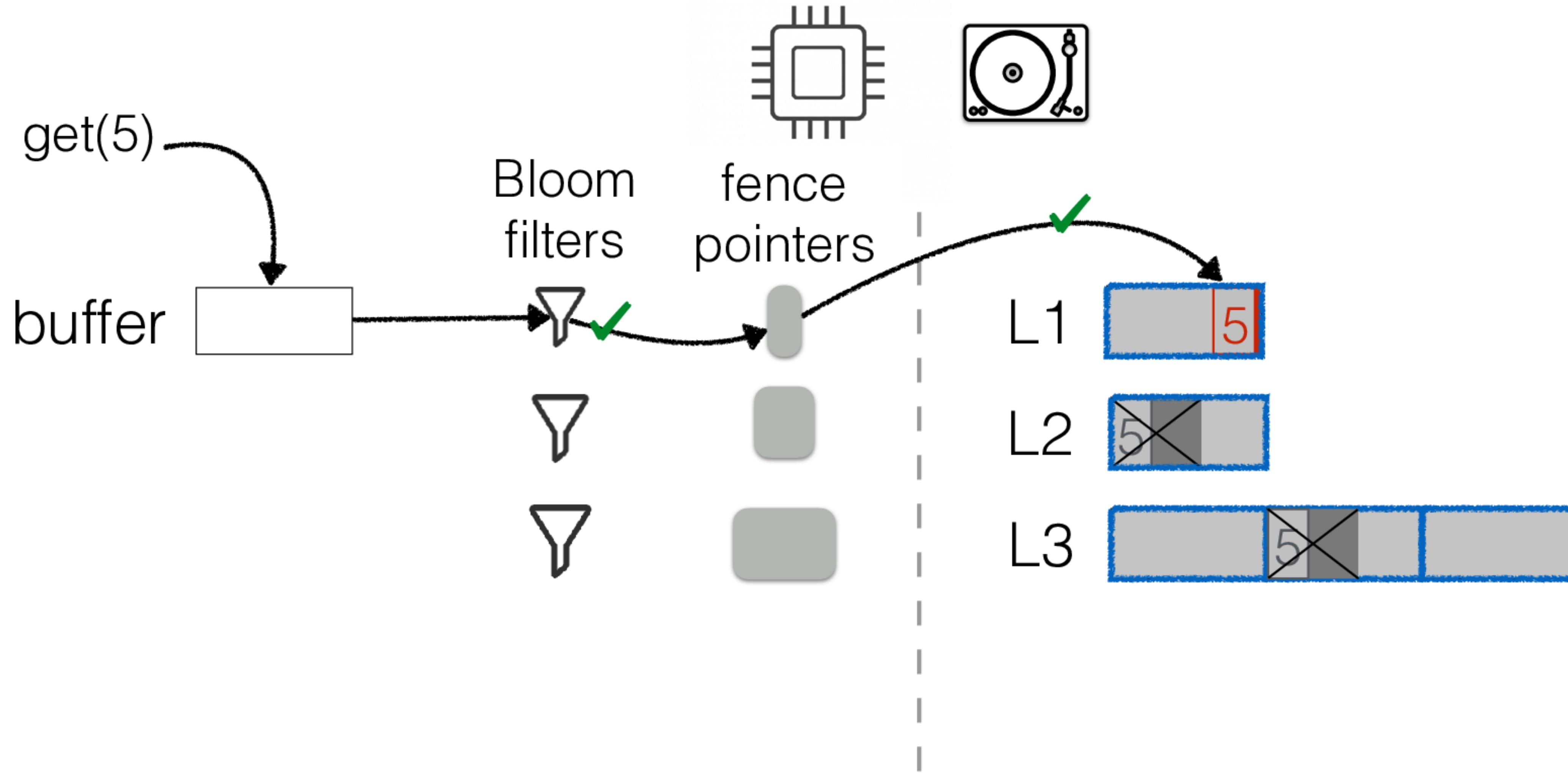


deletes in LSM-tree

buffer 



deletes in LSM-tree



the problems

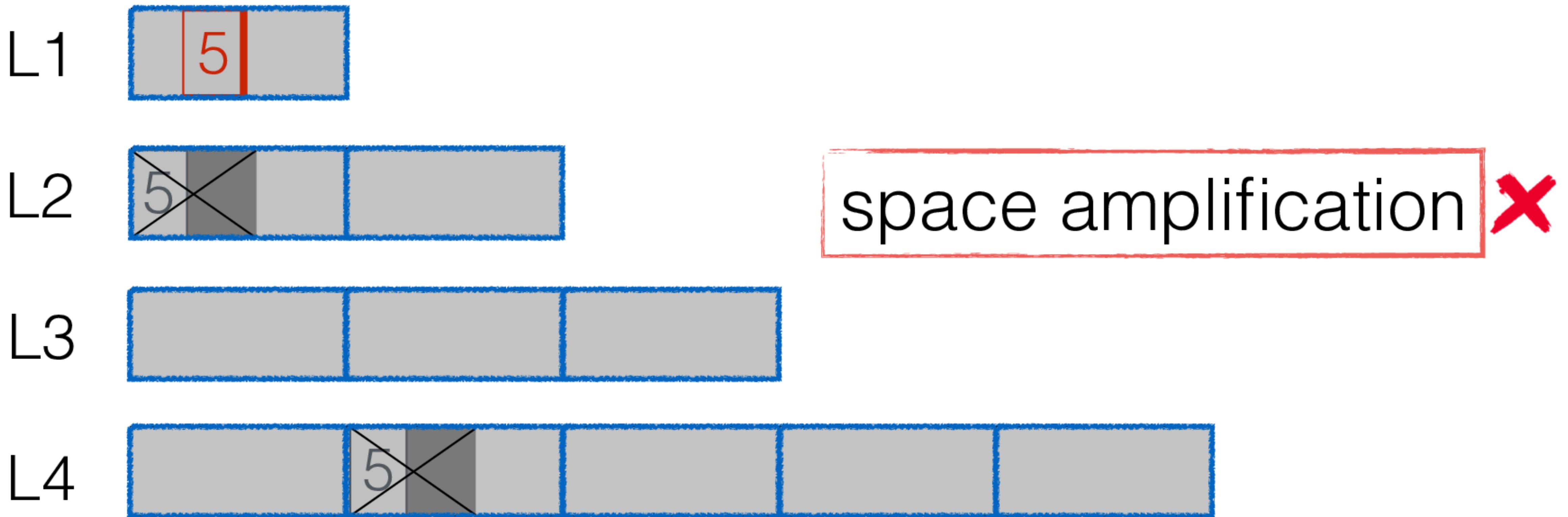


the problems

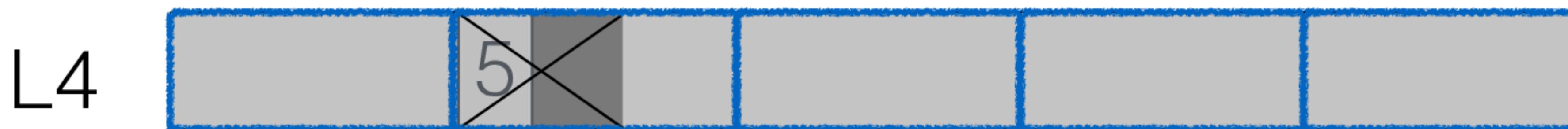


out-of-place deletes

out-of-place deletes

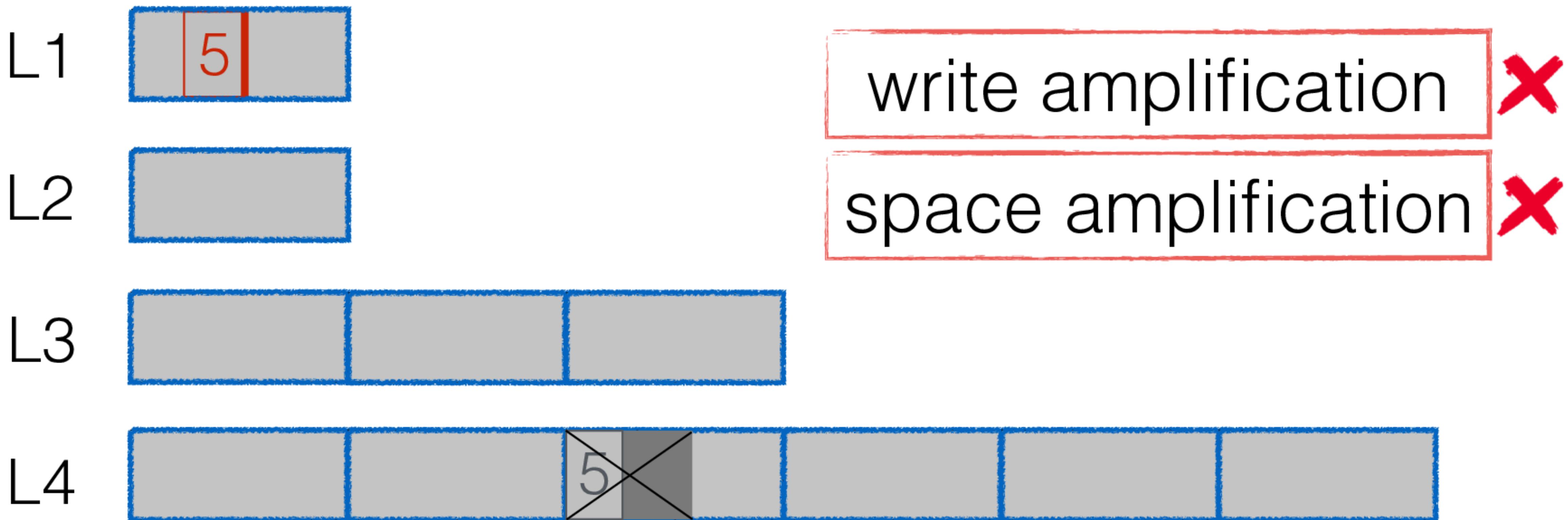


out-of-place deletes



space amplification X

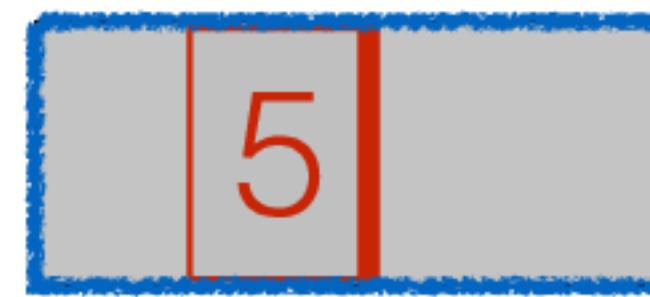
out-of-place deletes



out-of-place deletes

Bloom
filters

▽ L1



▽ L2



▽ L3



▽ L4



write amplification

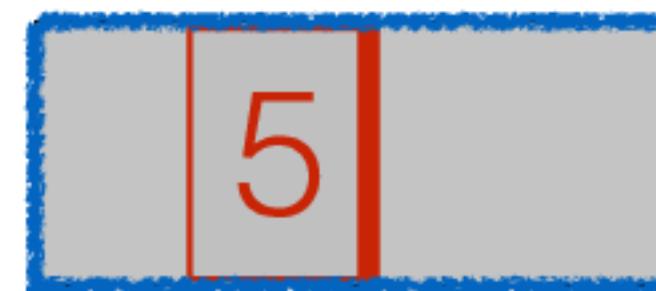
space amplification

out-of-place deletes

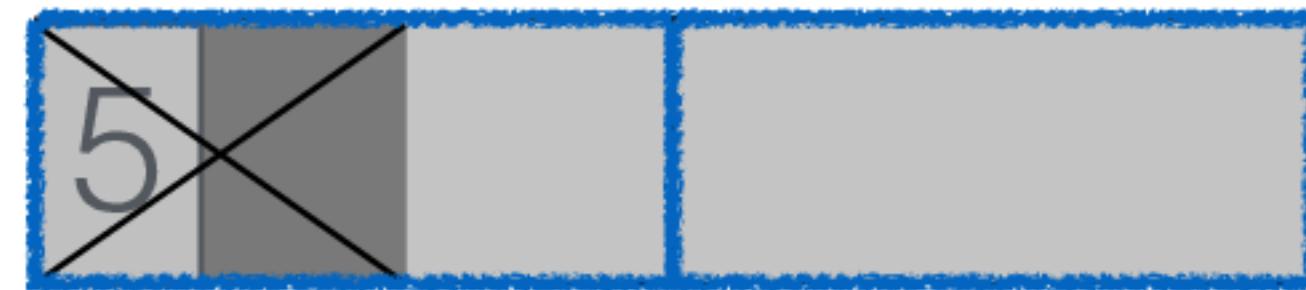
Bloom
filters



L1



L2



L3



L4



poor read perf. X

write amplification X

space amplification X

out-of-place deletes

Bloom
filters



L1



L2



L3



L4



poor read perf.

write amplification

space amplification

the problems

poor read perf.

write amplification

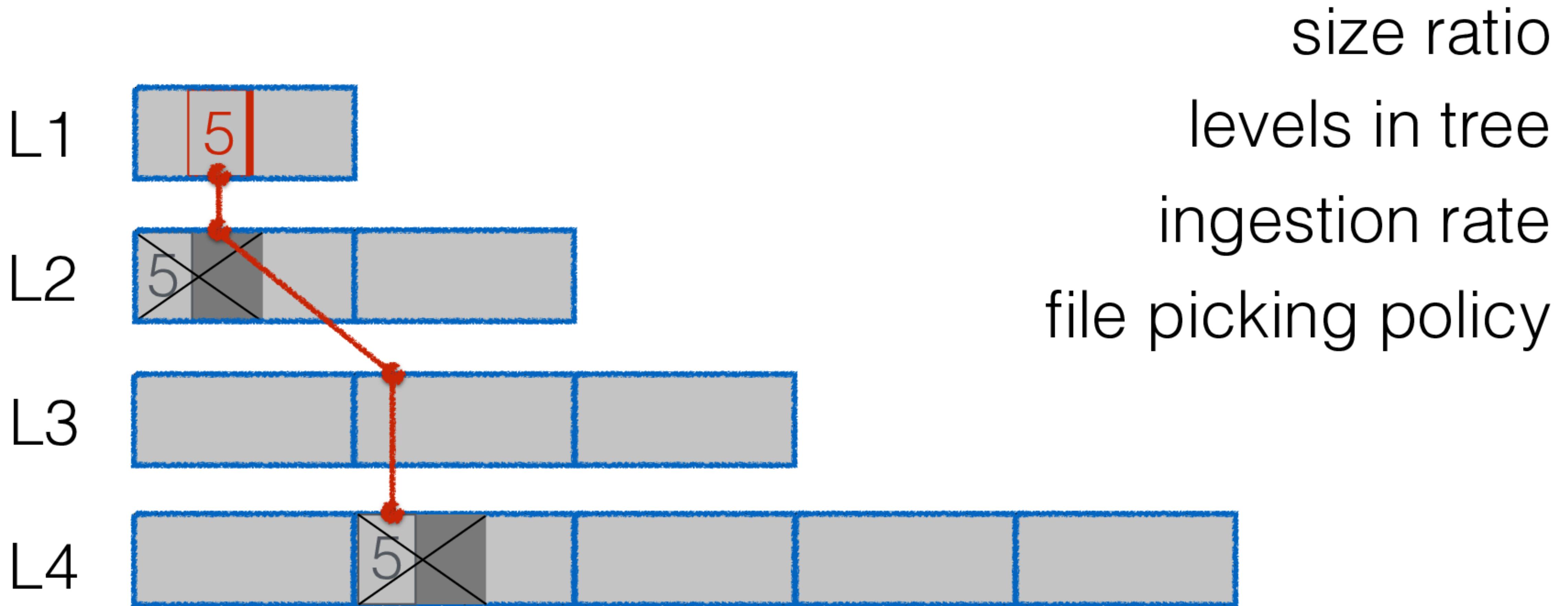
space amplification





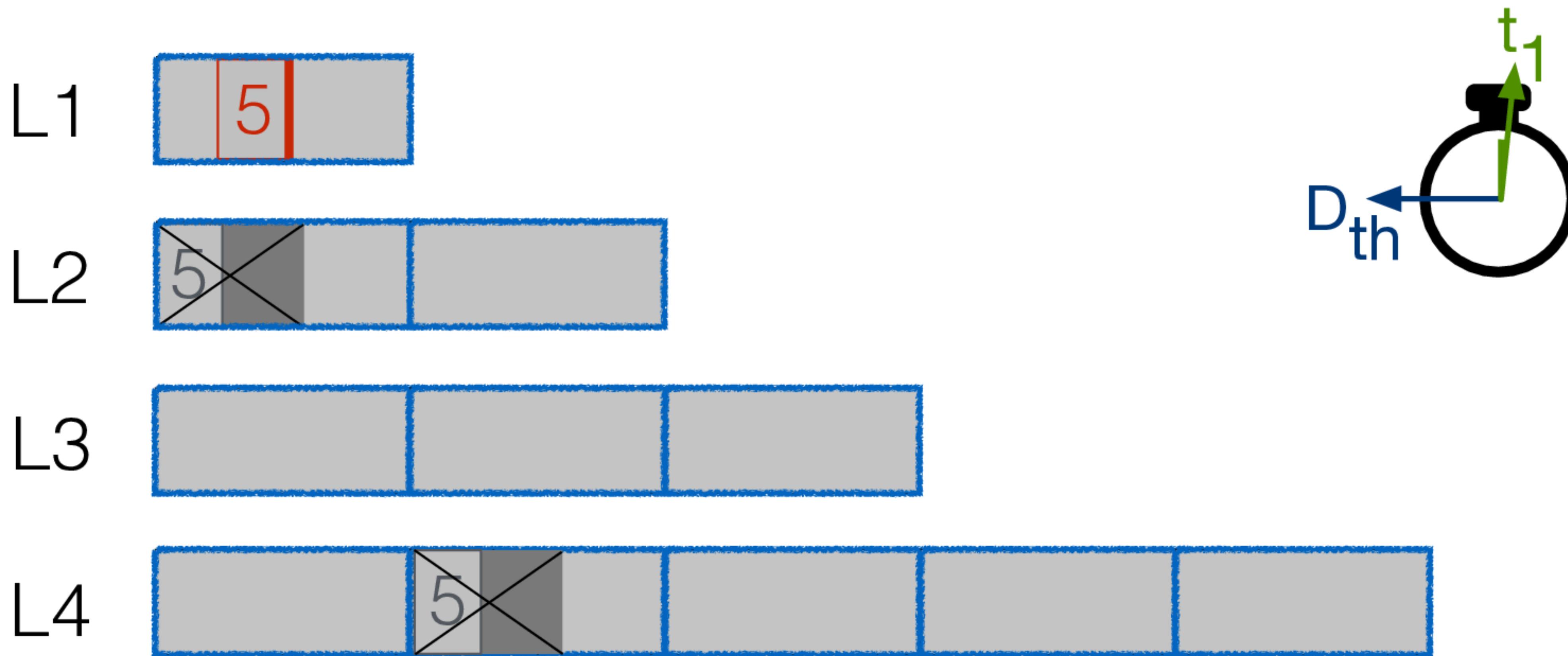
delete persistence latency

delete persistence latency



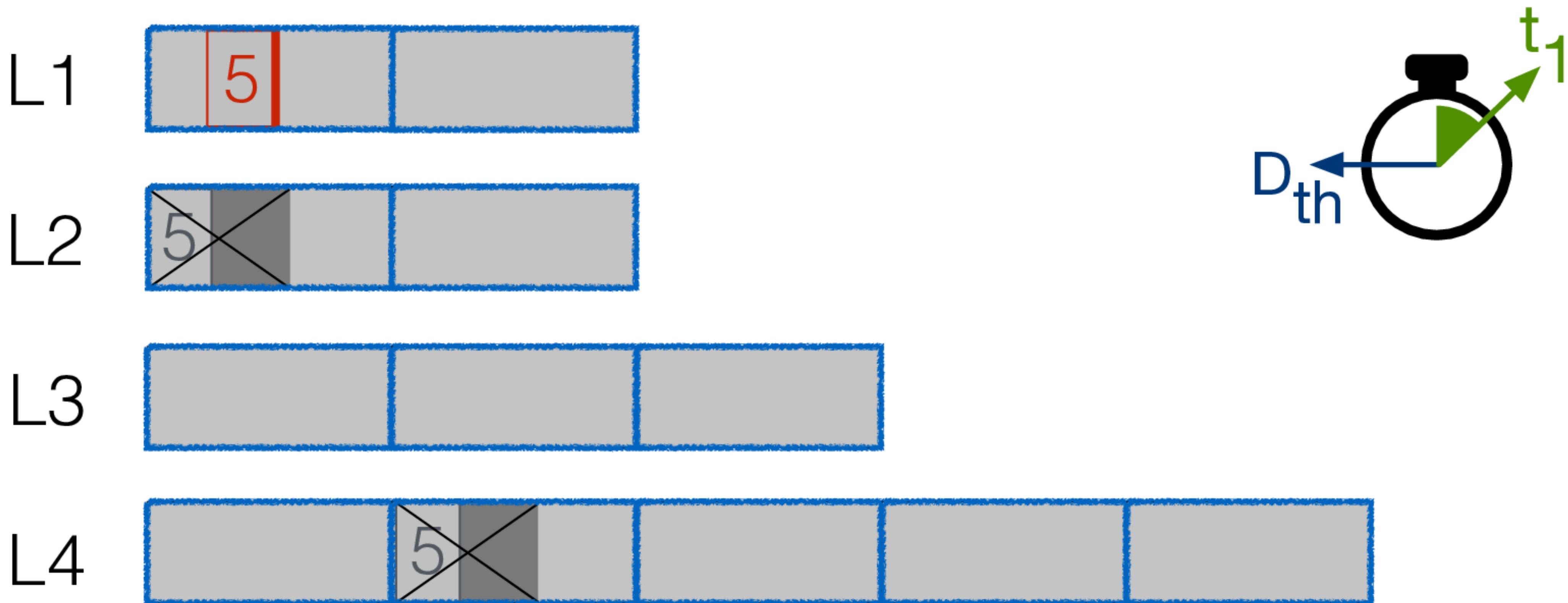
delete persistence latency

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



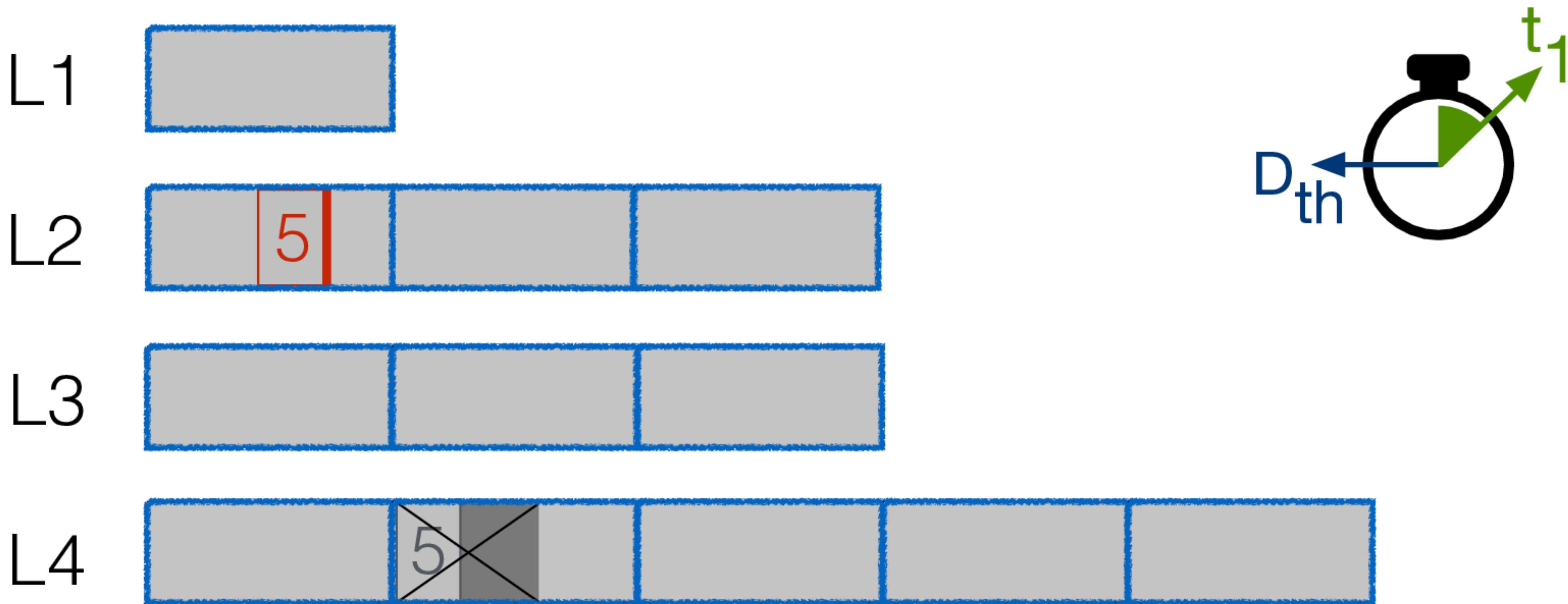
delete persistence latency

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



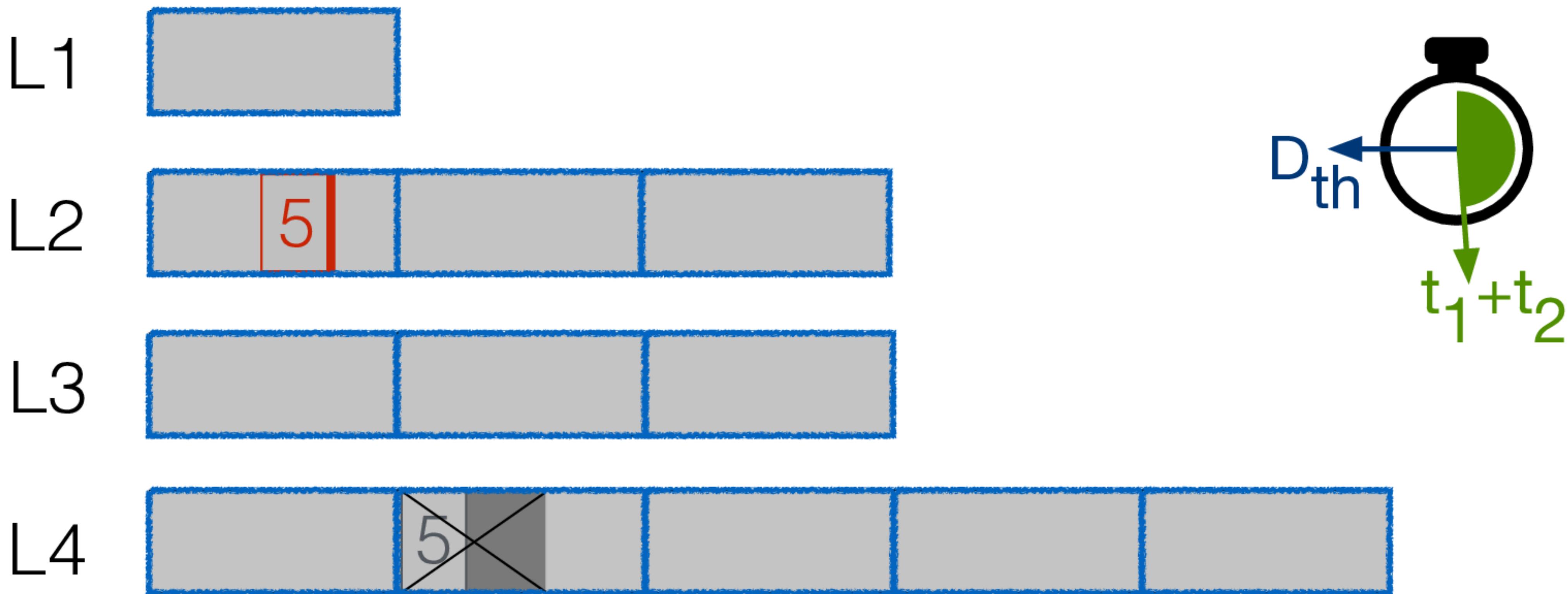
delete persistence latency

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



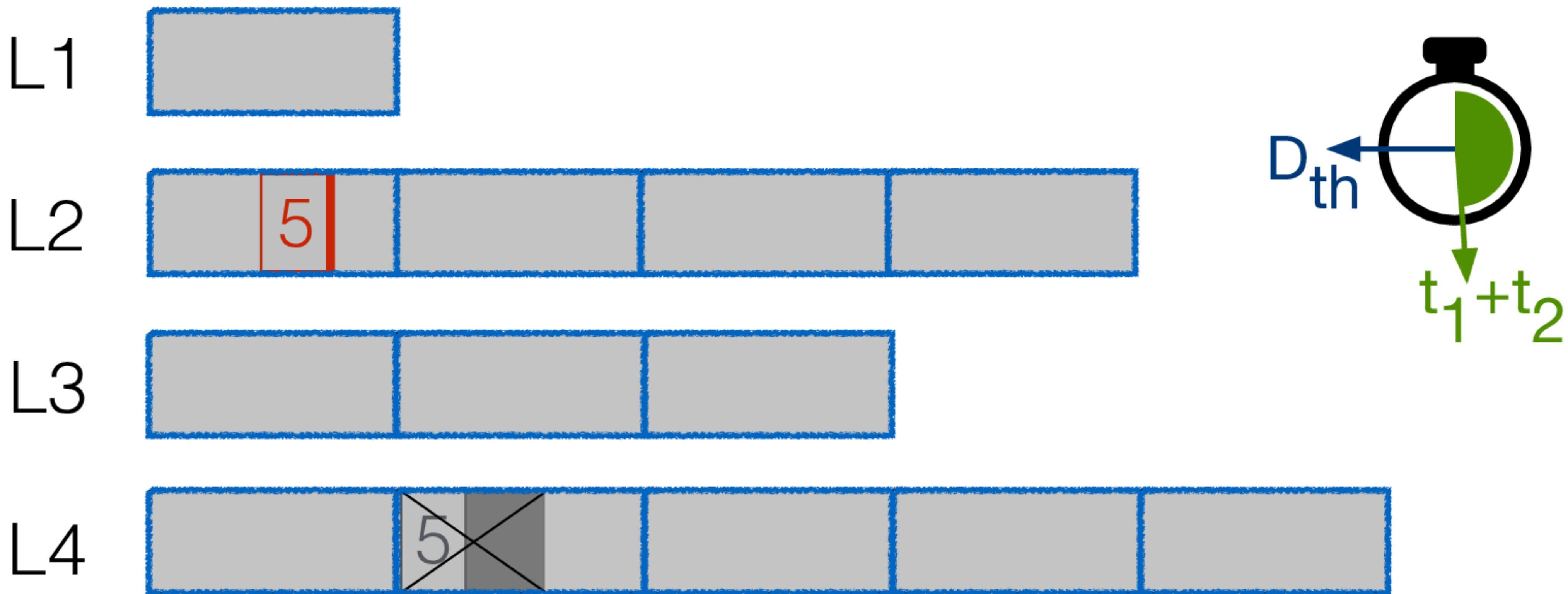
delete persistence latency

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



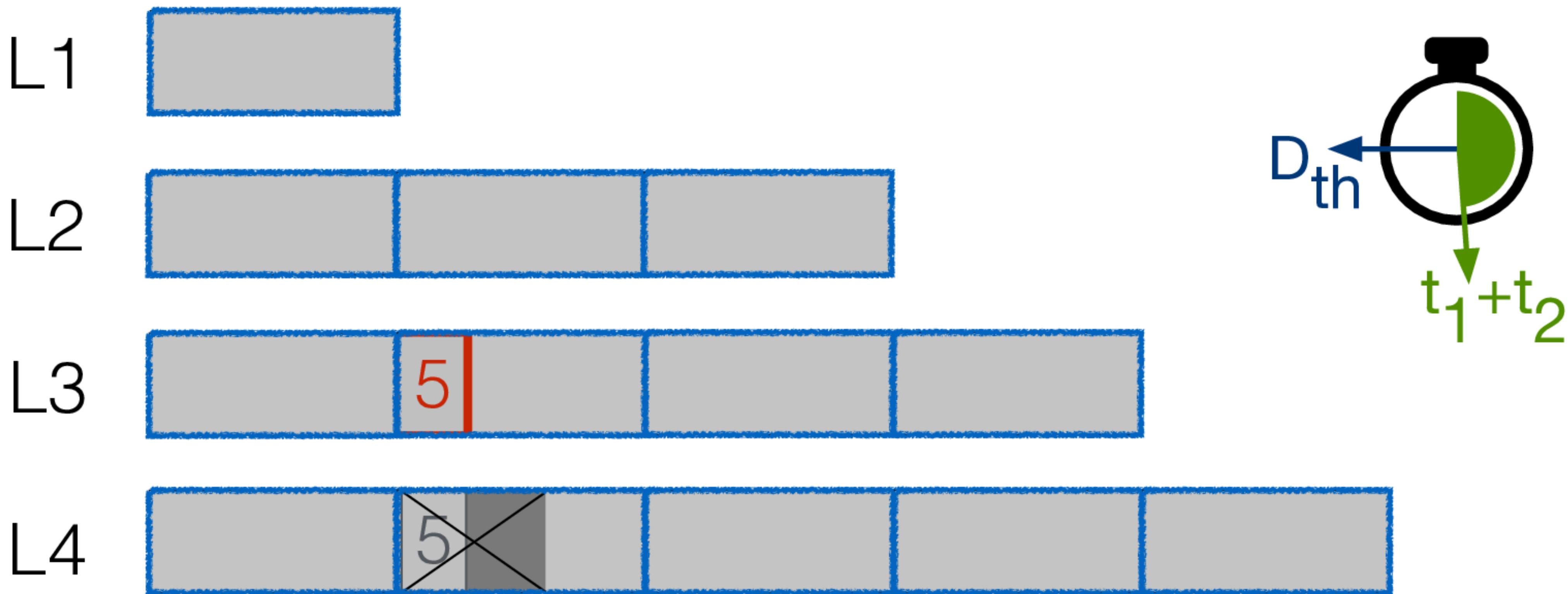
delete persistence latency

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



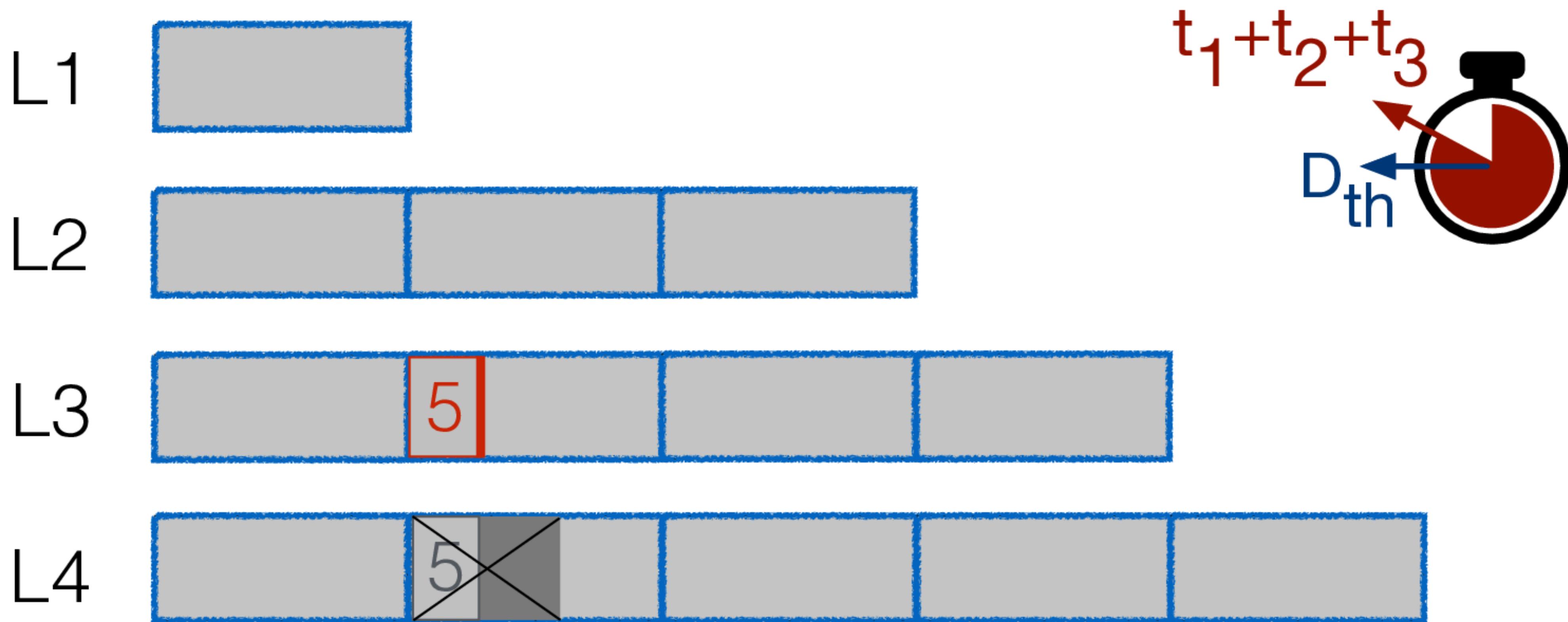
delete persistence latency

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



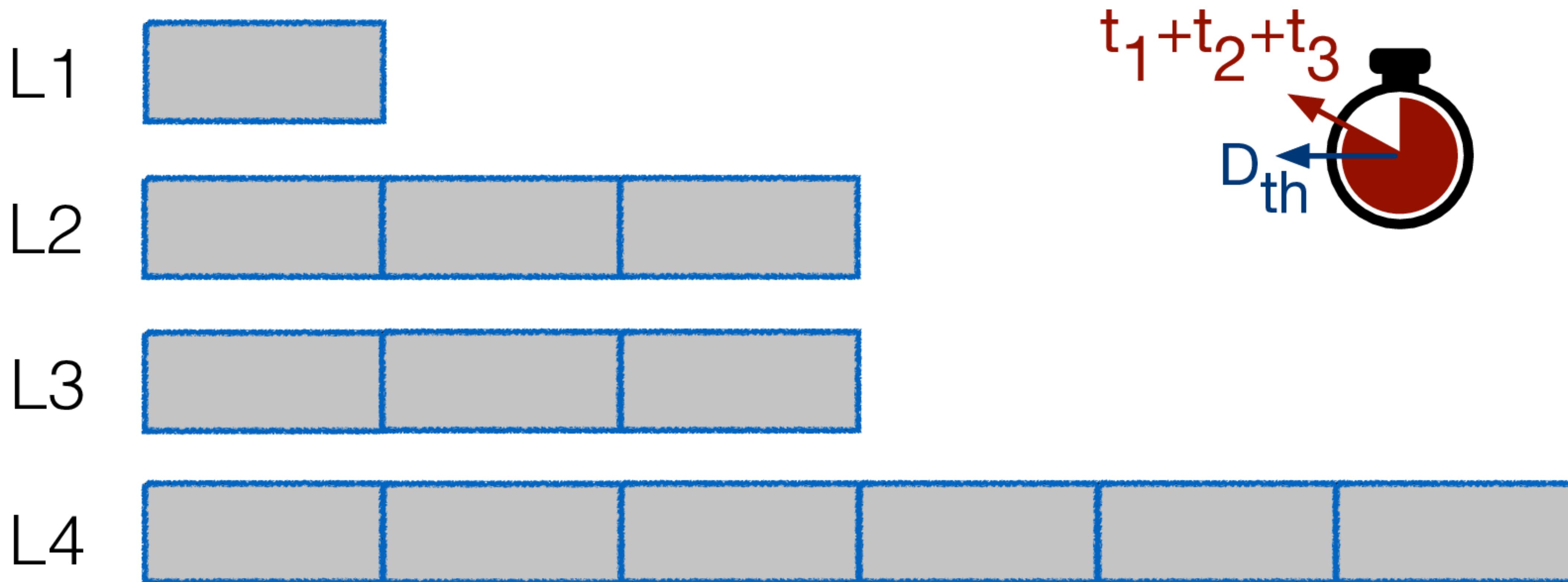
delete persistence latency

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



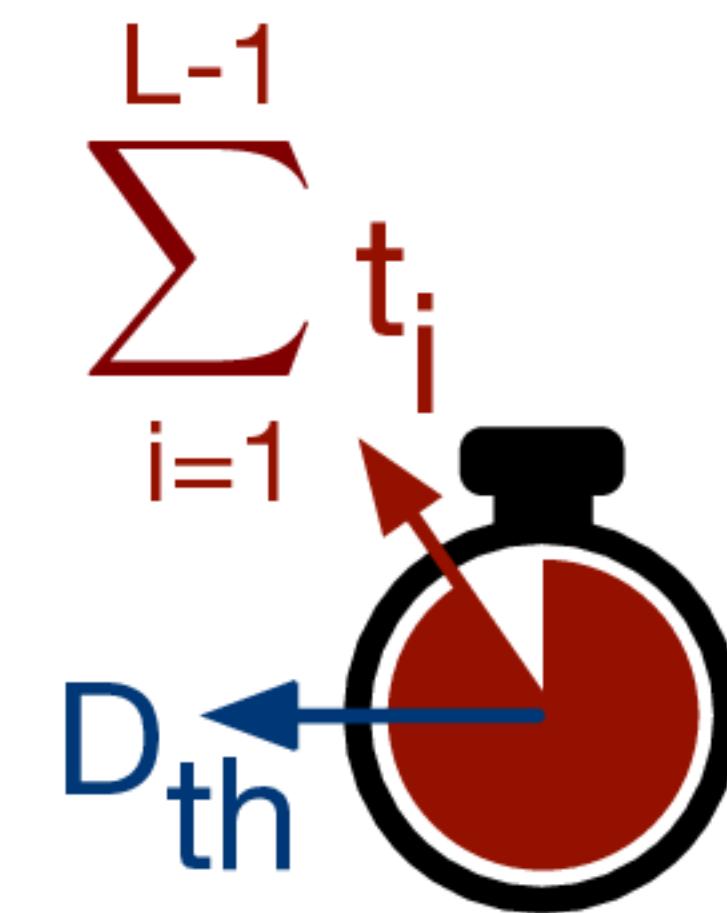
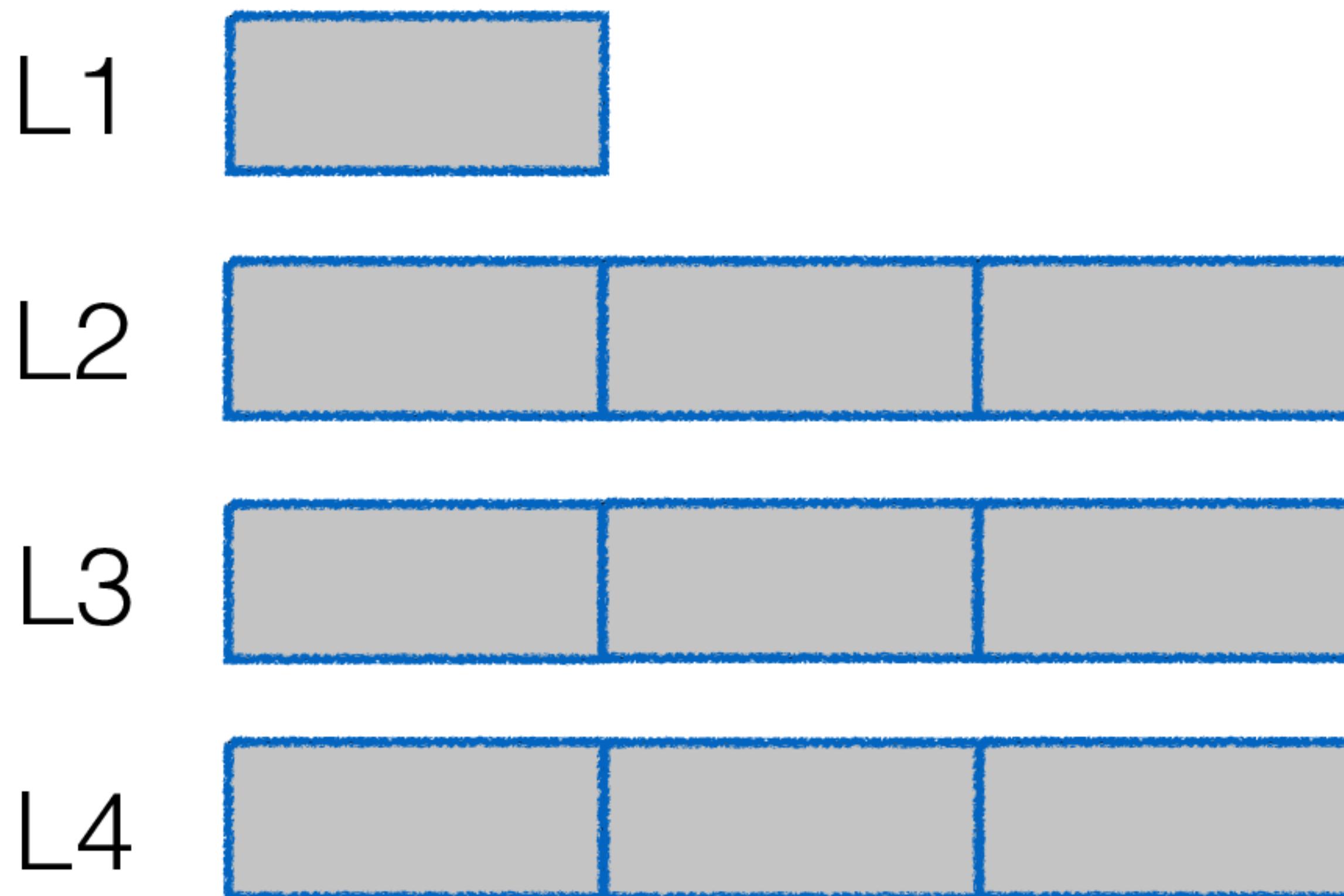
delete persistence latency

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



delete persistence latency

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



unbounded delete
persistence latency

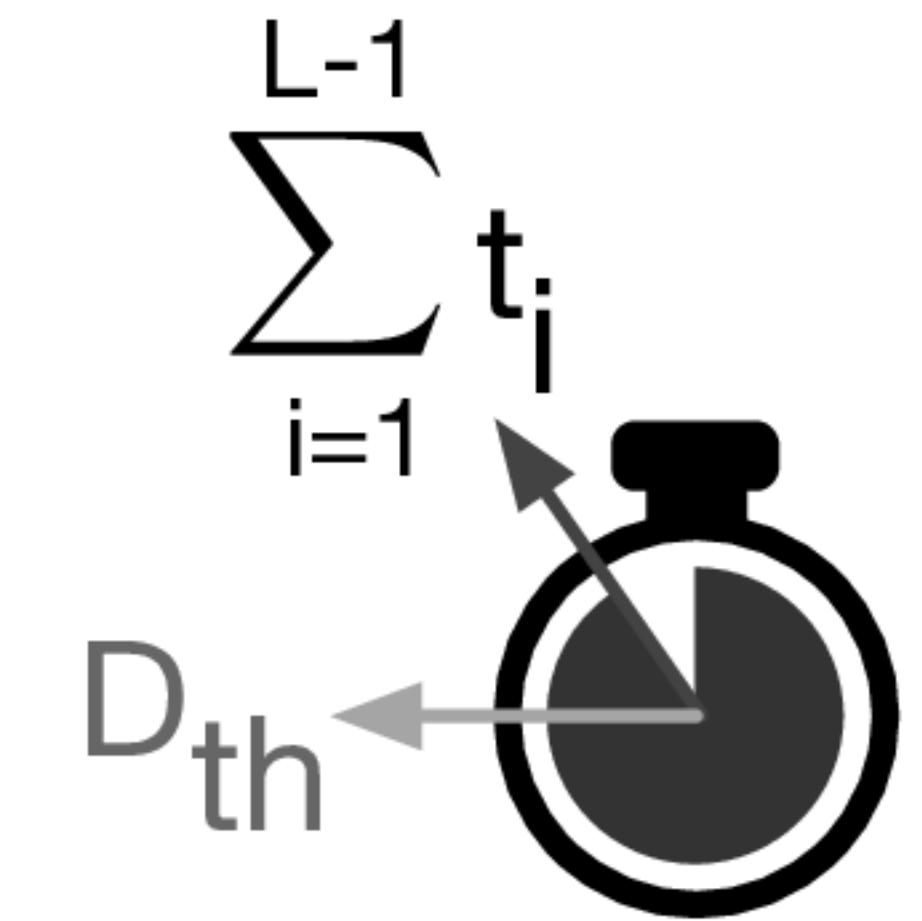
X

the problems

poor read perf.

write amplification

space amplification



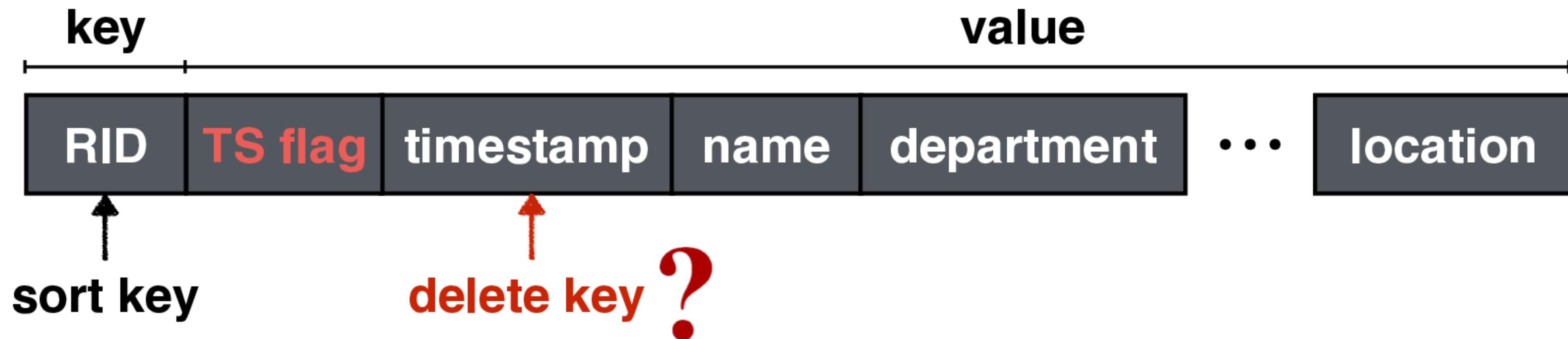
unbounded delete
persistence latency



deletes on a secondary attribute

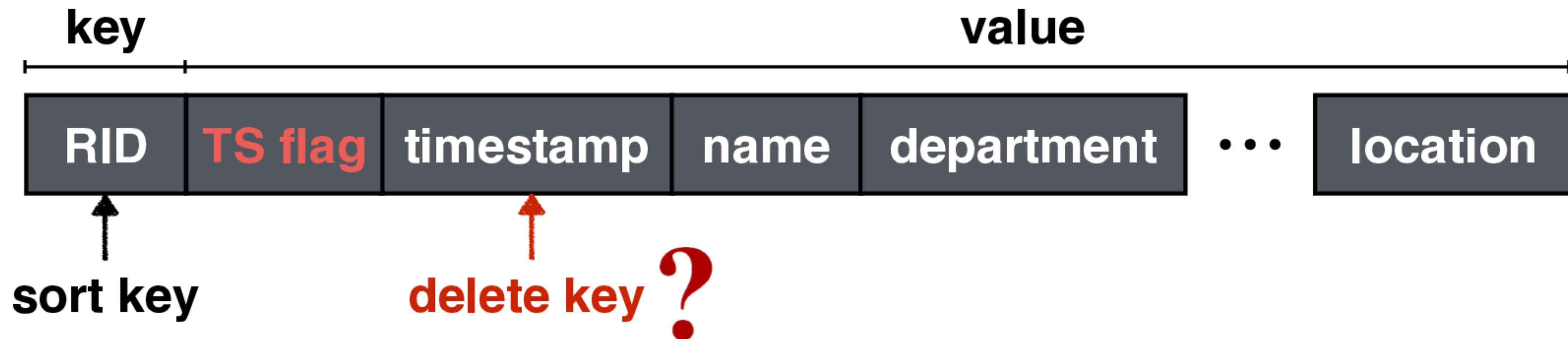
deletes on a secondary attribute

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



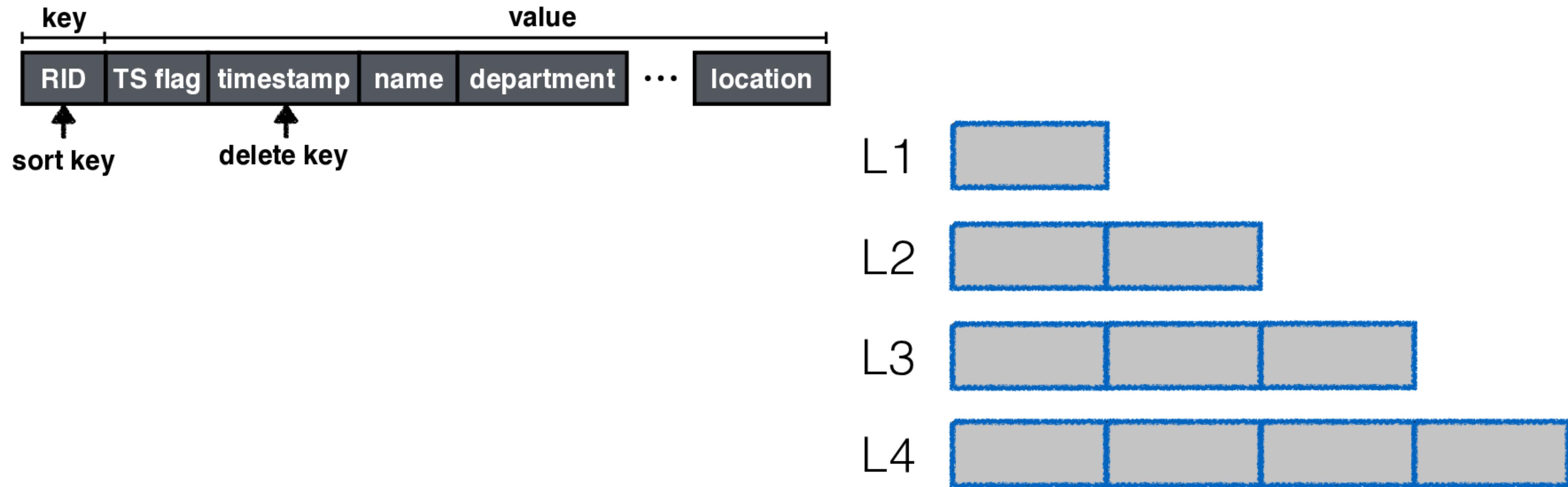
deletes on a secondary attribute

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



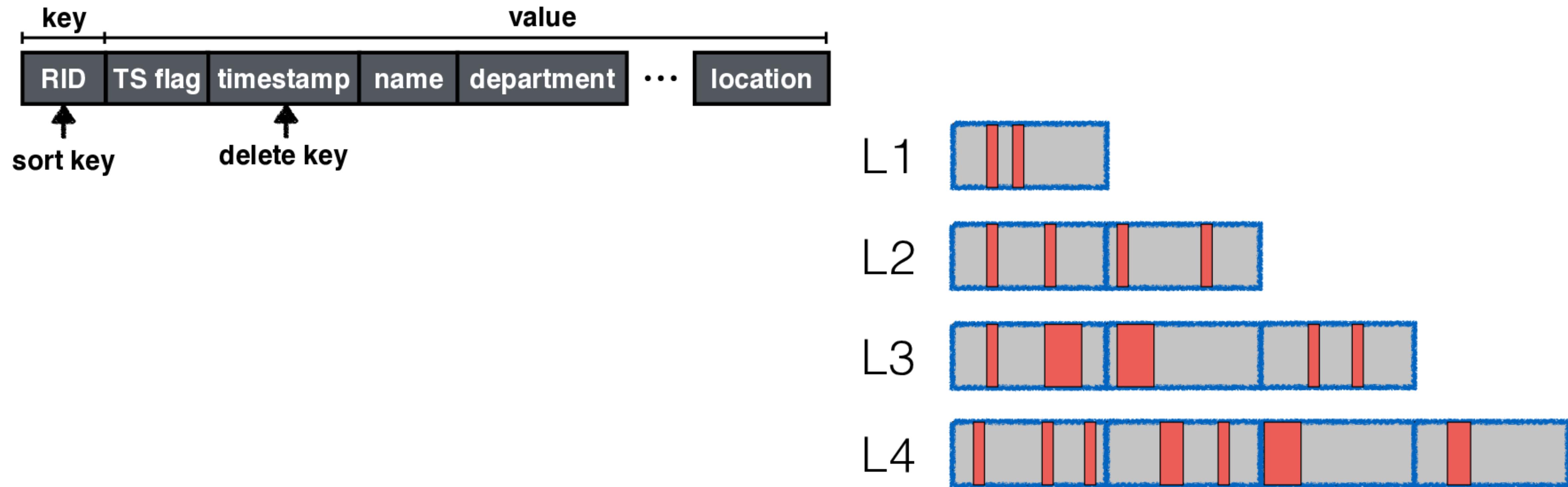
deletes on a secondary attribute

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



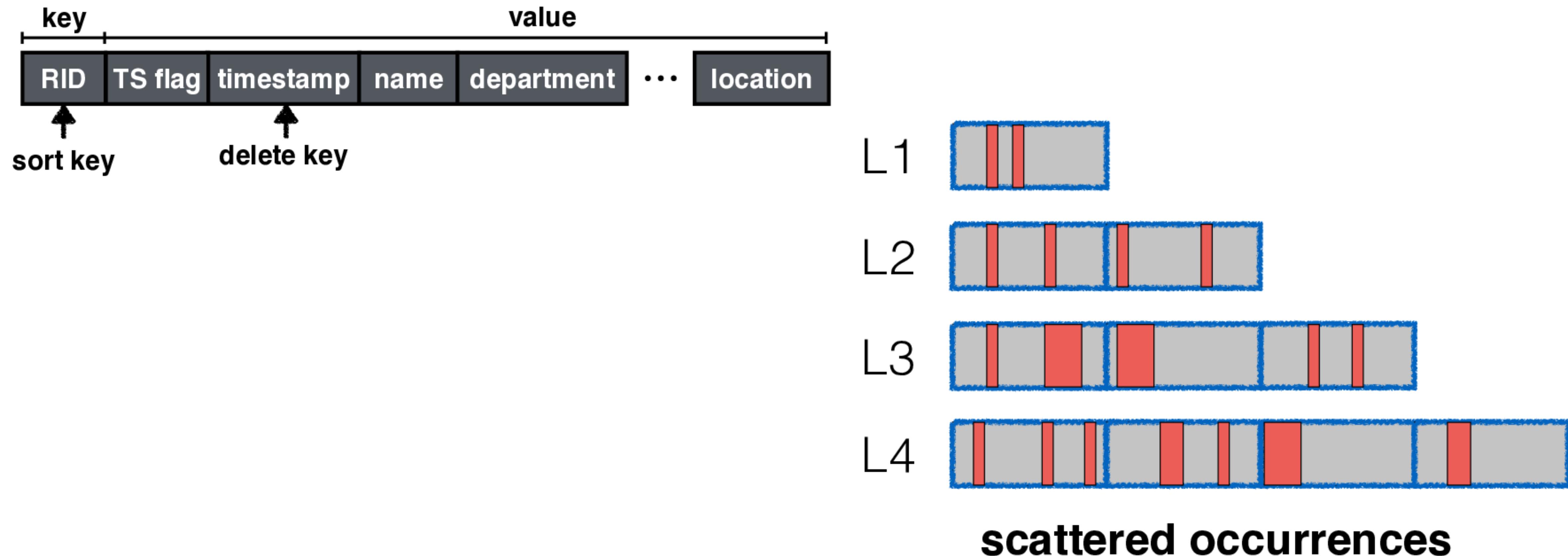
deletes on a secondary attribute

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



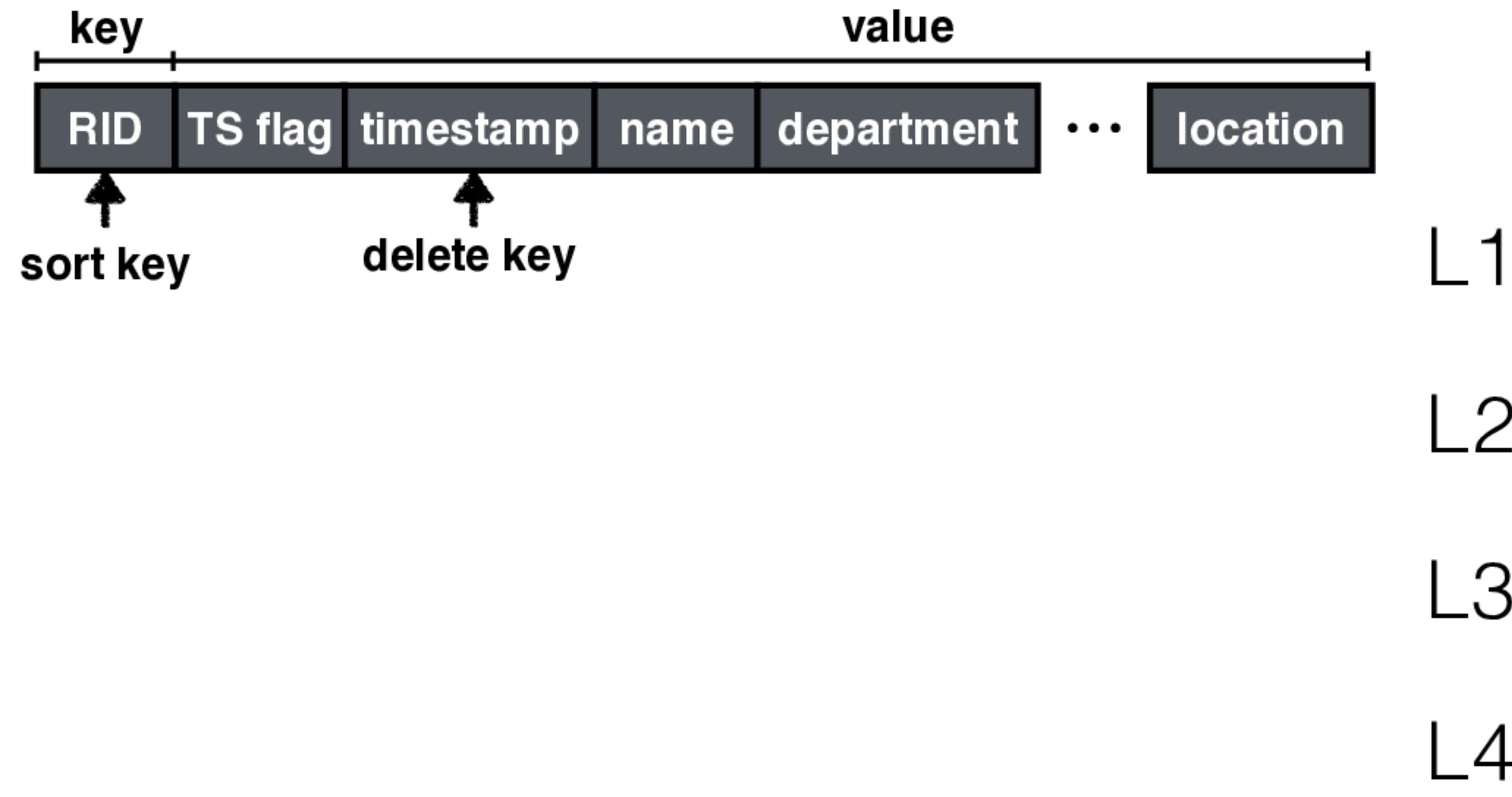
deletes on a secondary attribute

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



deletes on a secondary attribute

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



deletes on a secondary attribute

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



L1

L2

L3

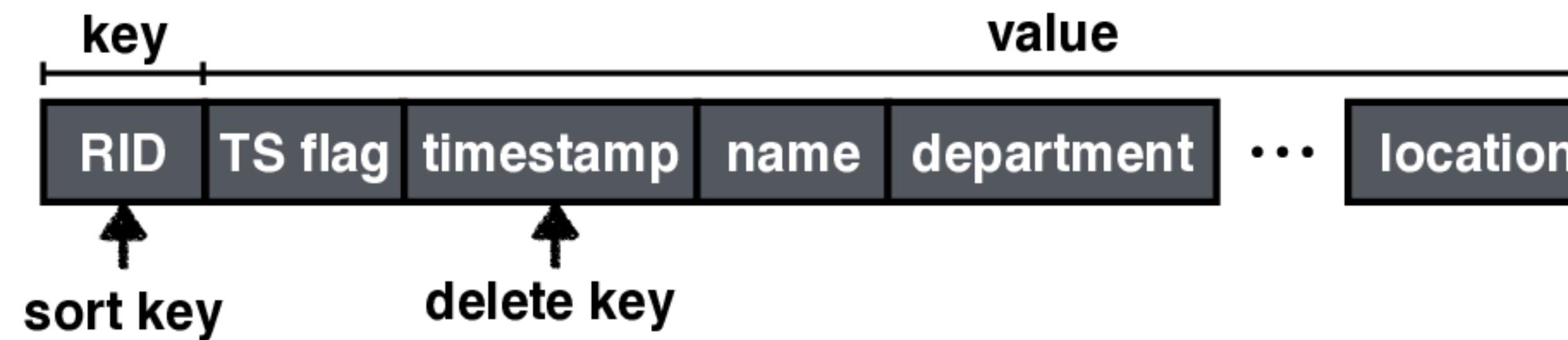
L4

latency spikes X

superfluous I/Os X

deletes on a secondary attribute

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



L1

L2

L3

L4

latency spikes X

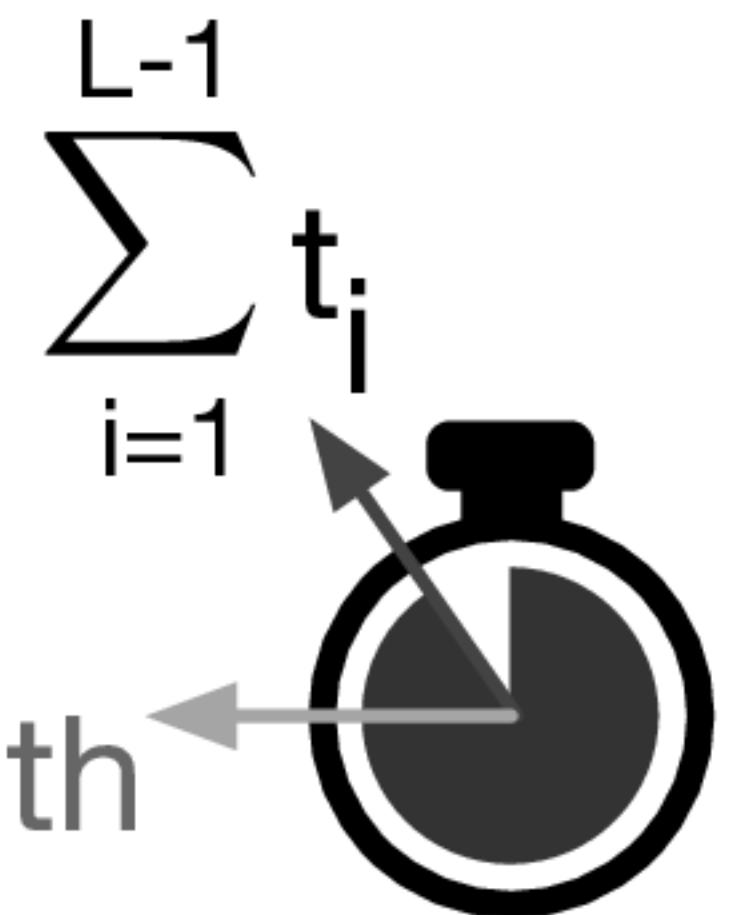
superfluous I/Os X

the problems

poor read perf.

write amplification

space amplification

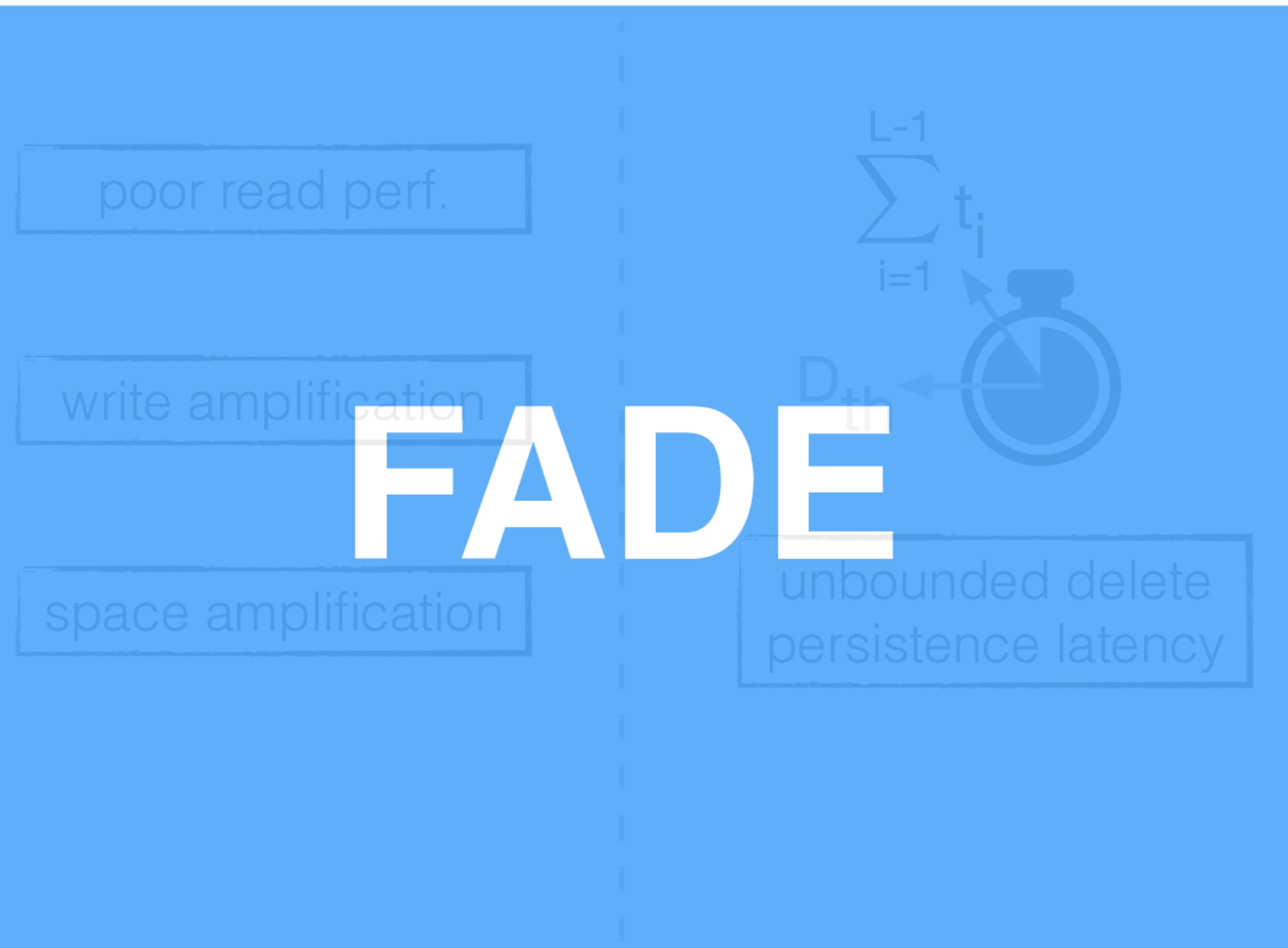


unbounded delete
persistence latency

latency spikes

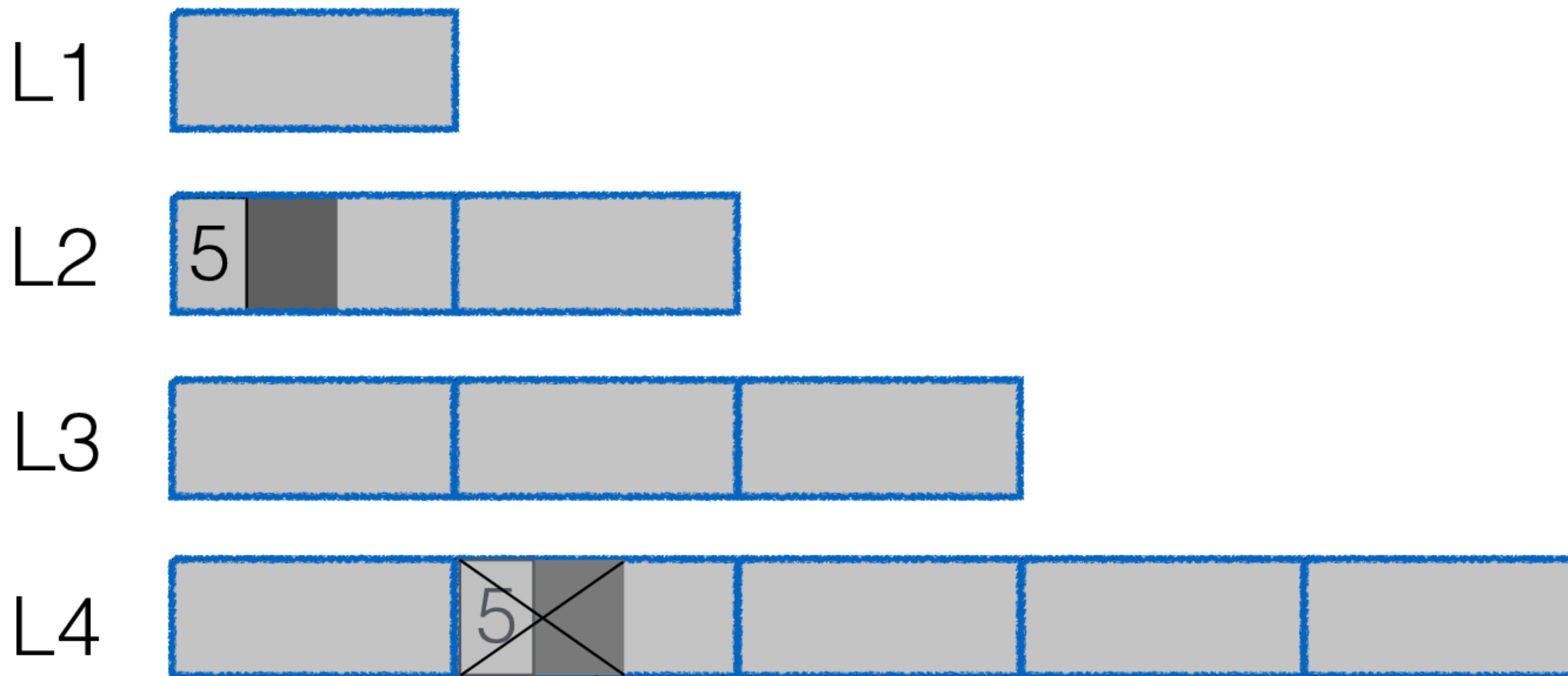
superfluous I/Os

the solution



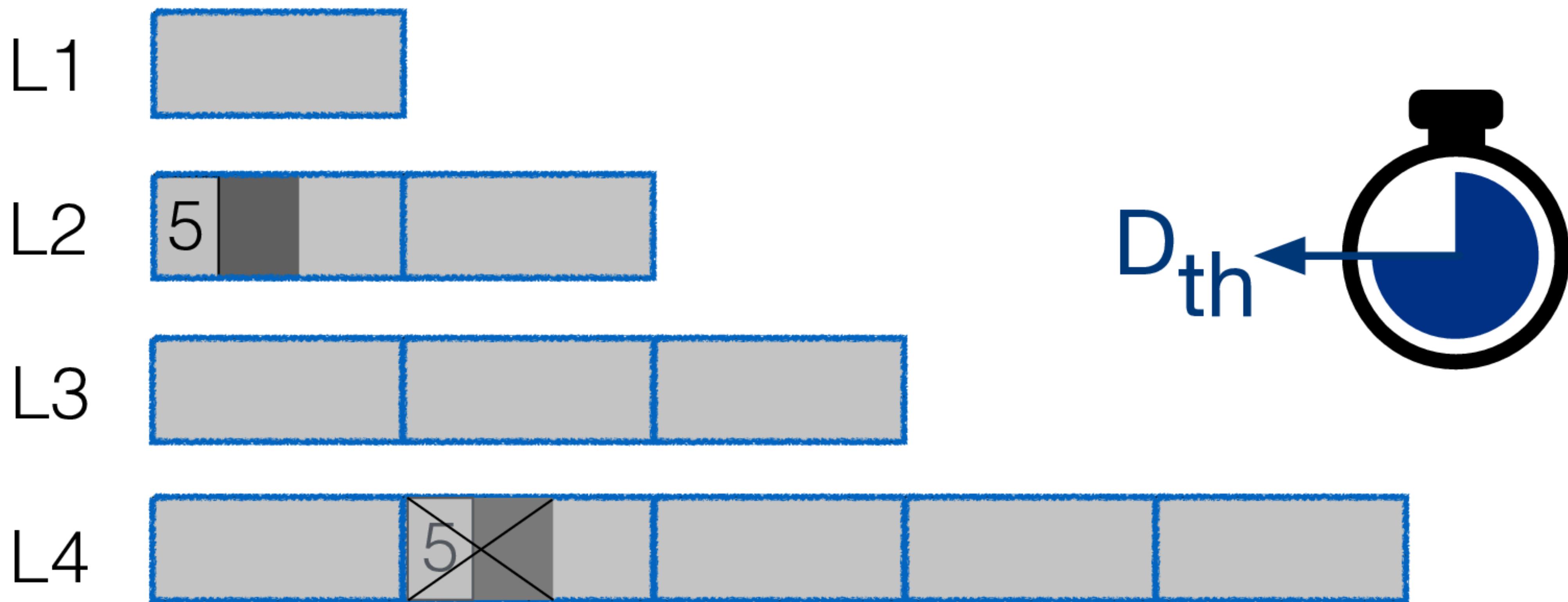
FAst DElete

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



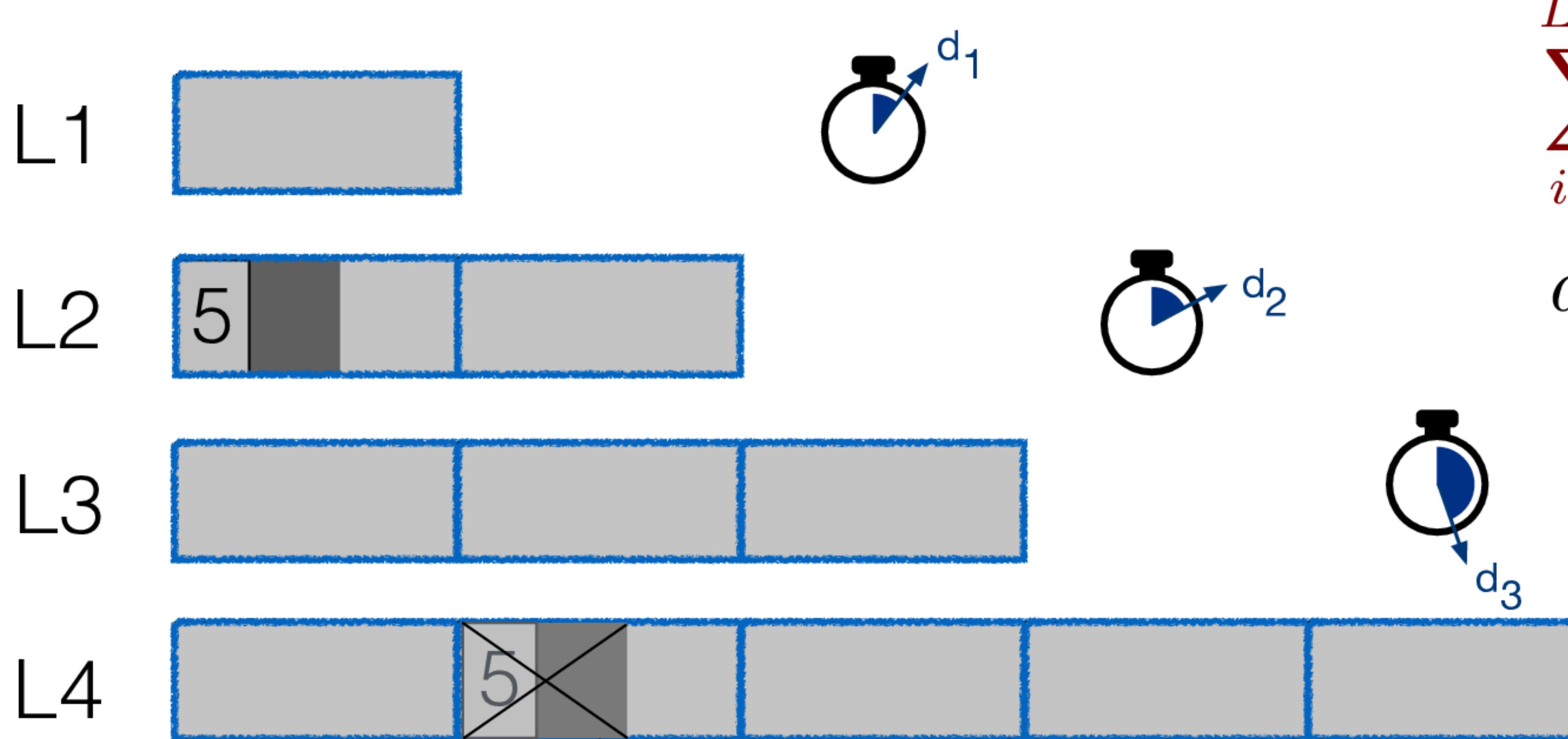
FAst DElete

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



FAst DElete

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



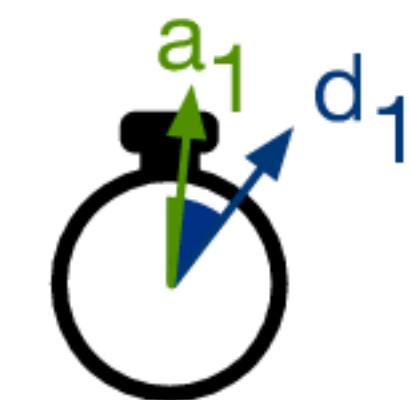
$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

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

FAst DElete

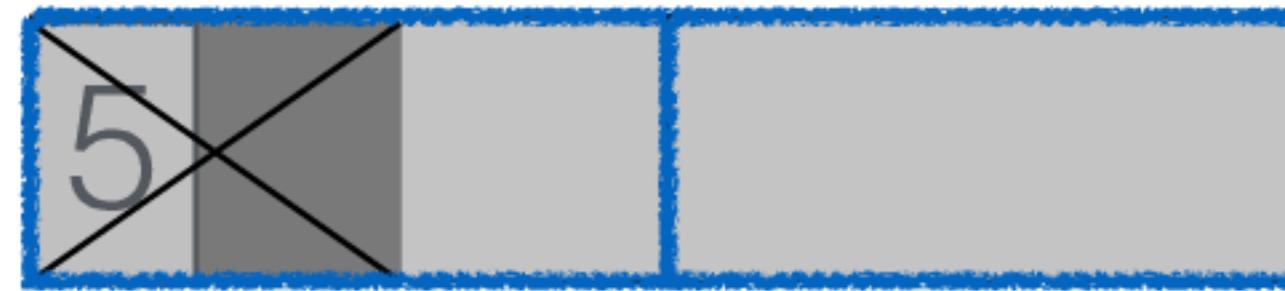
delete(5) within a threshold time: D_{th}

L1



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

L2



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

L3



L4



FAst DElete

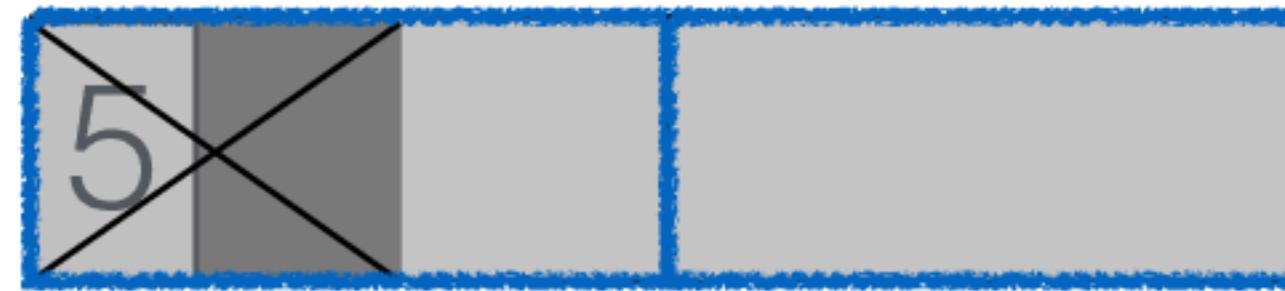
delete(5) within a threshold time: D_{th}

L1



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

L2



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

L3

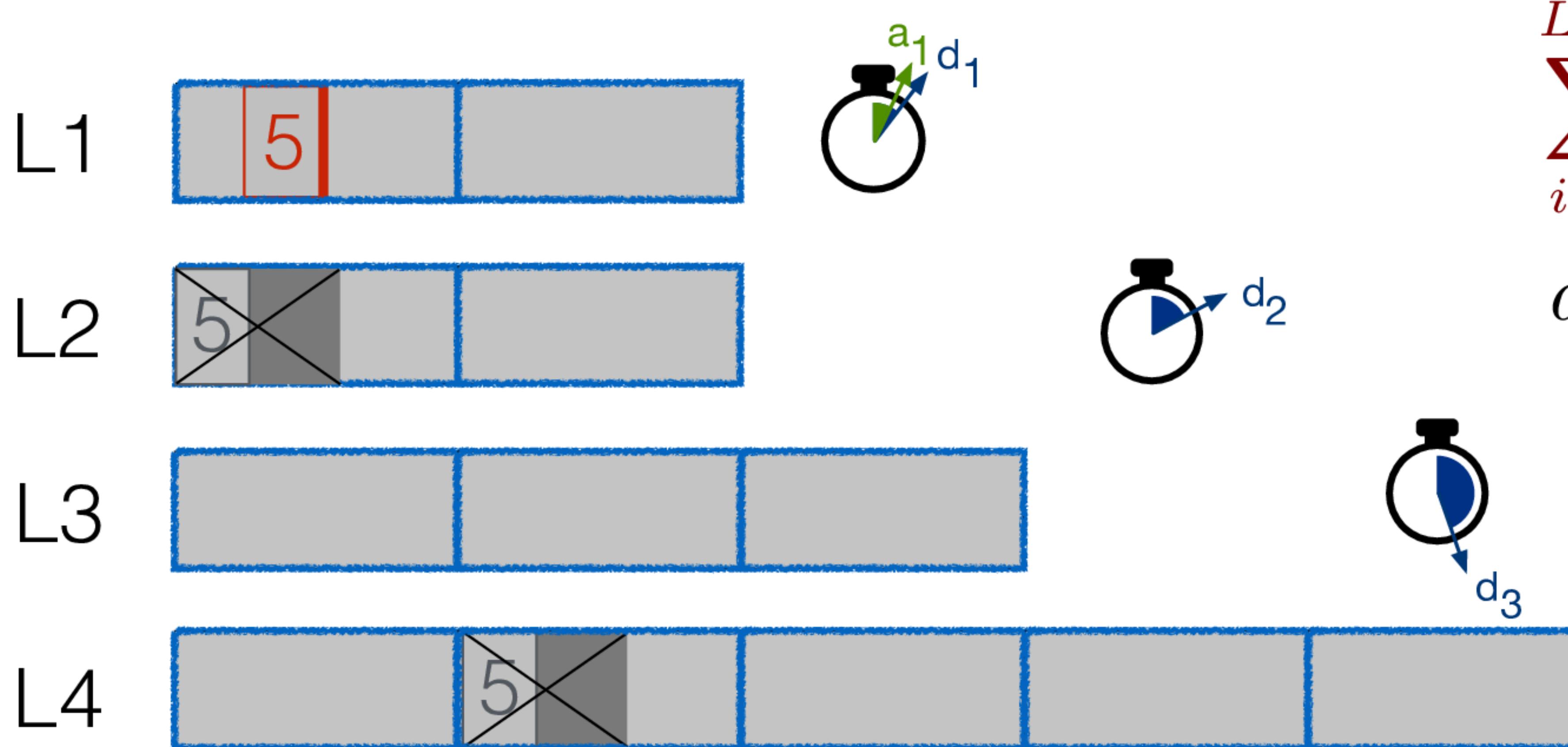


L4



FAst DElete

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

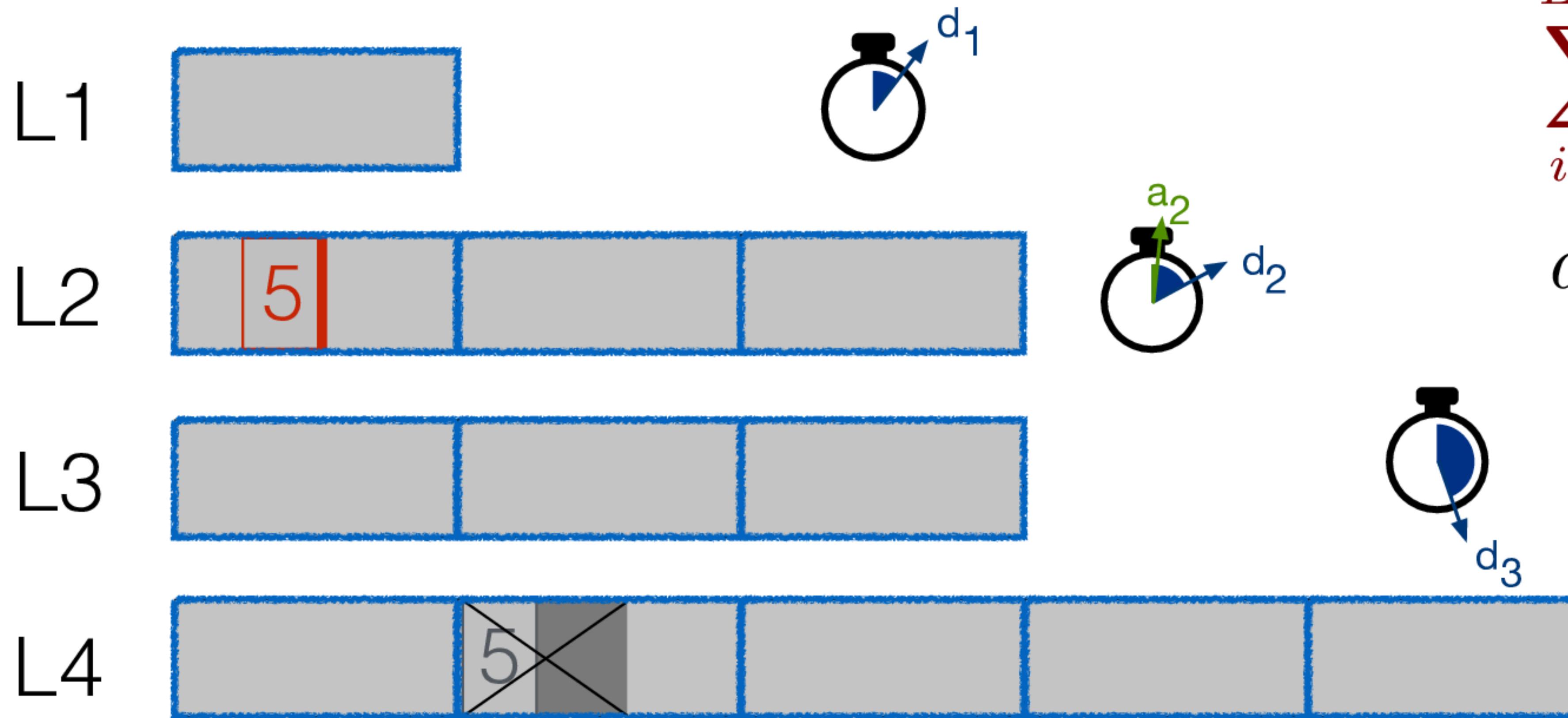


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

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

FAst DElete

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

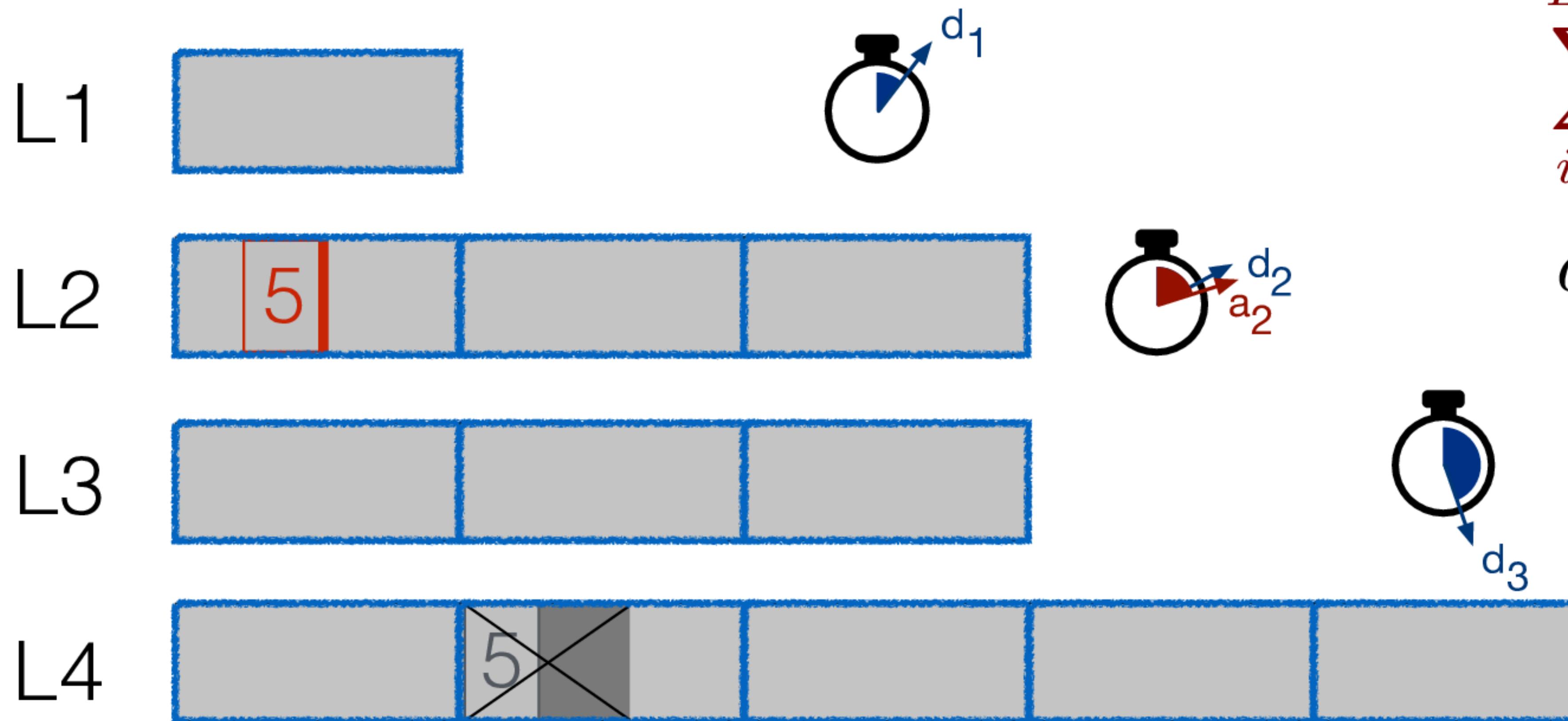


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

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

FAst DElete

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

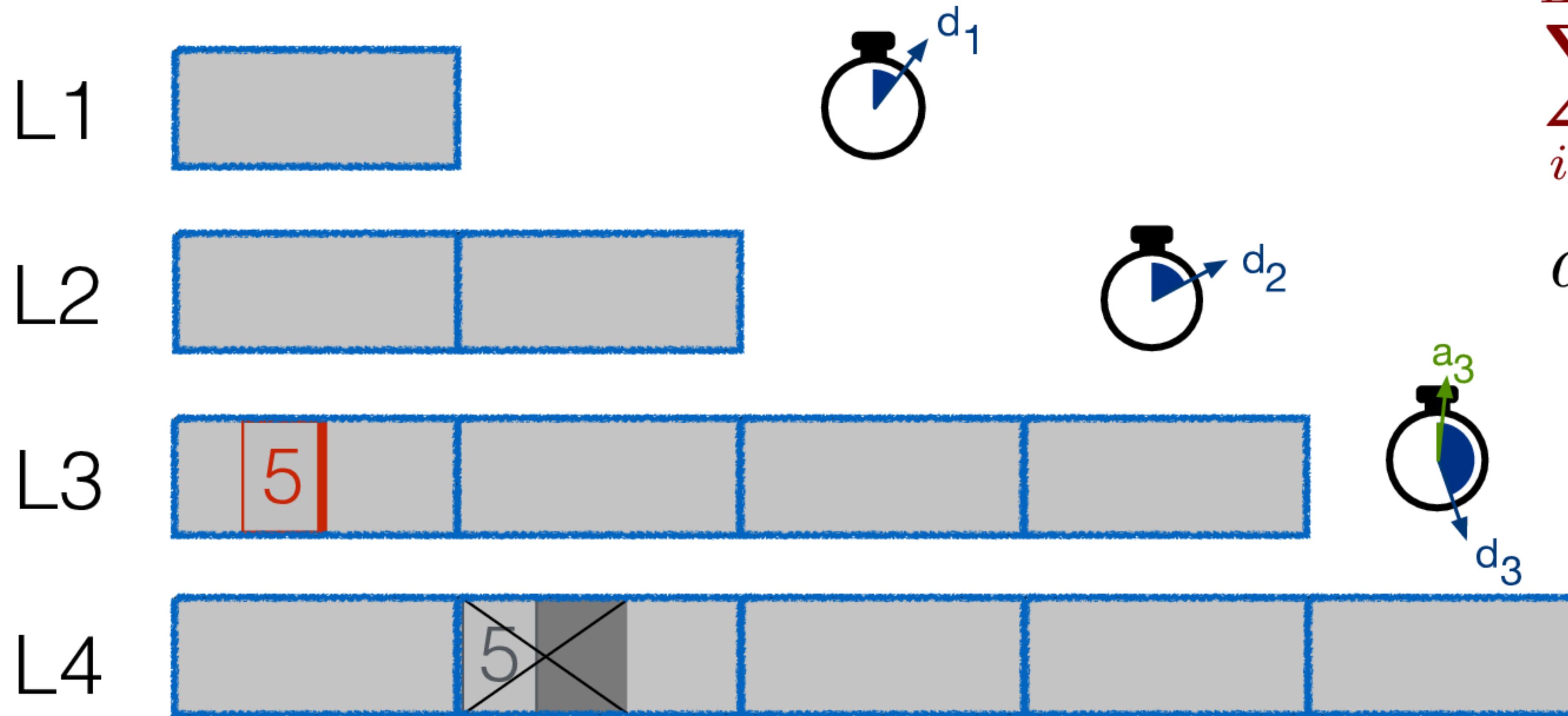


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

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

FAst DElete

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

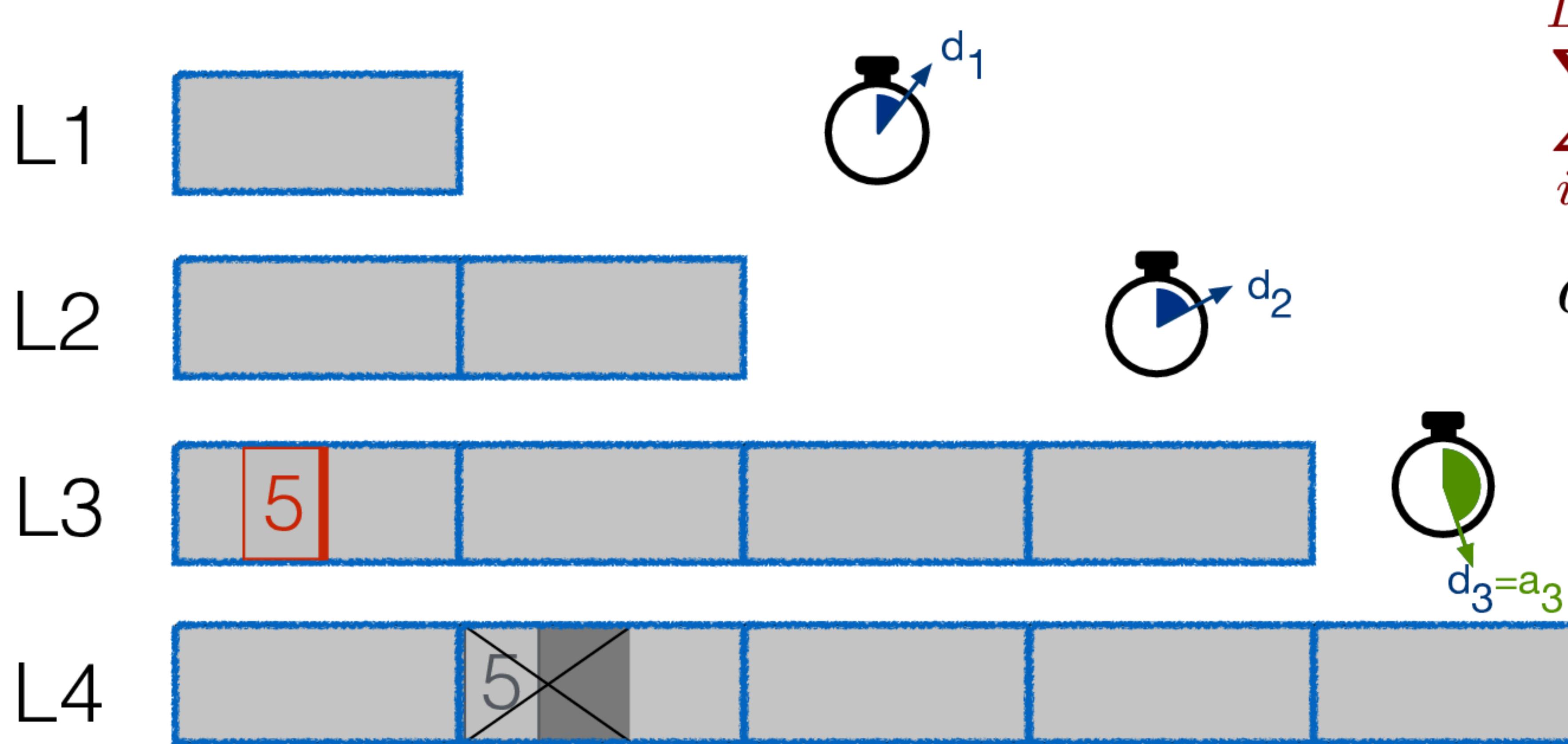


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

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

FAst DElete

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

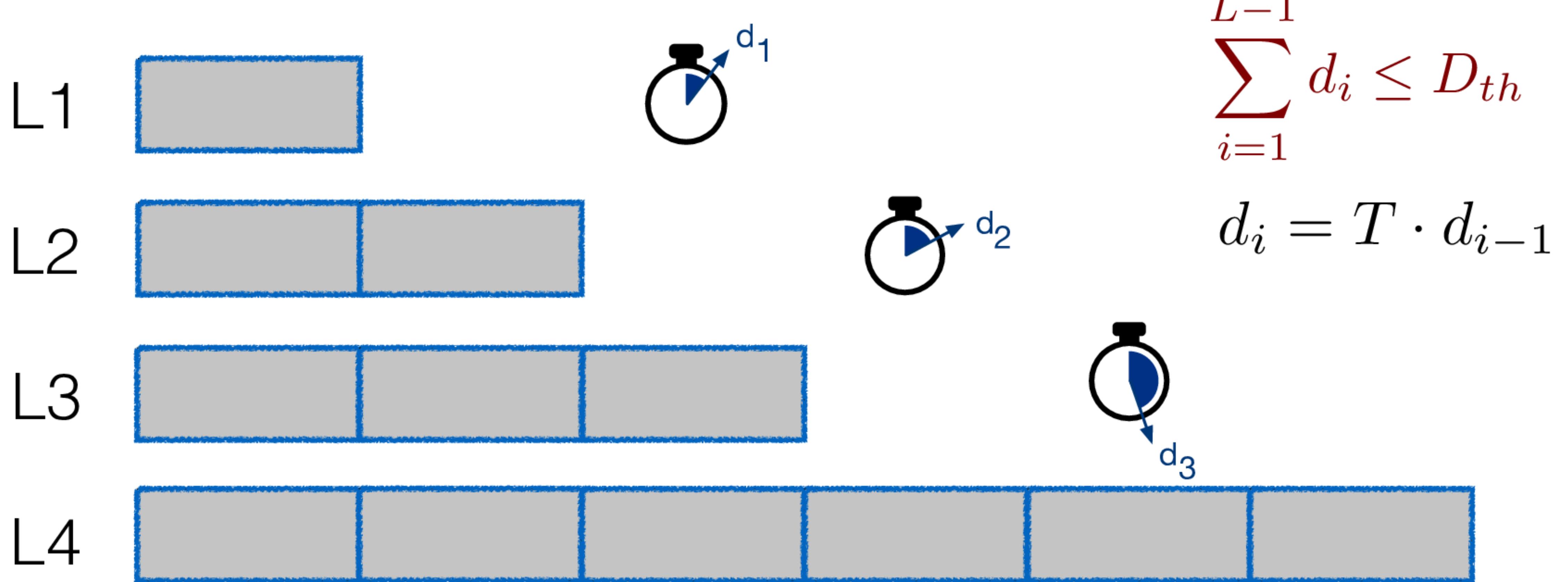


$$\sum_{i=1}^{L-1} d_i \leq D_{th}$$

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

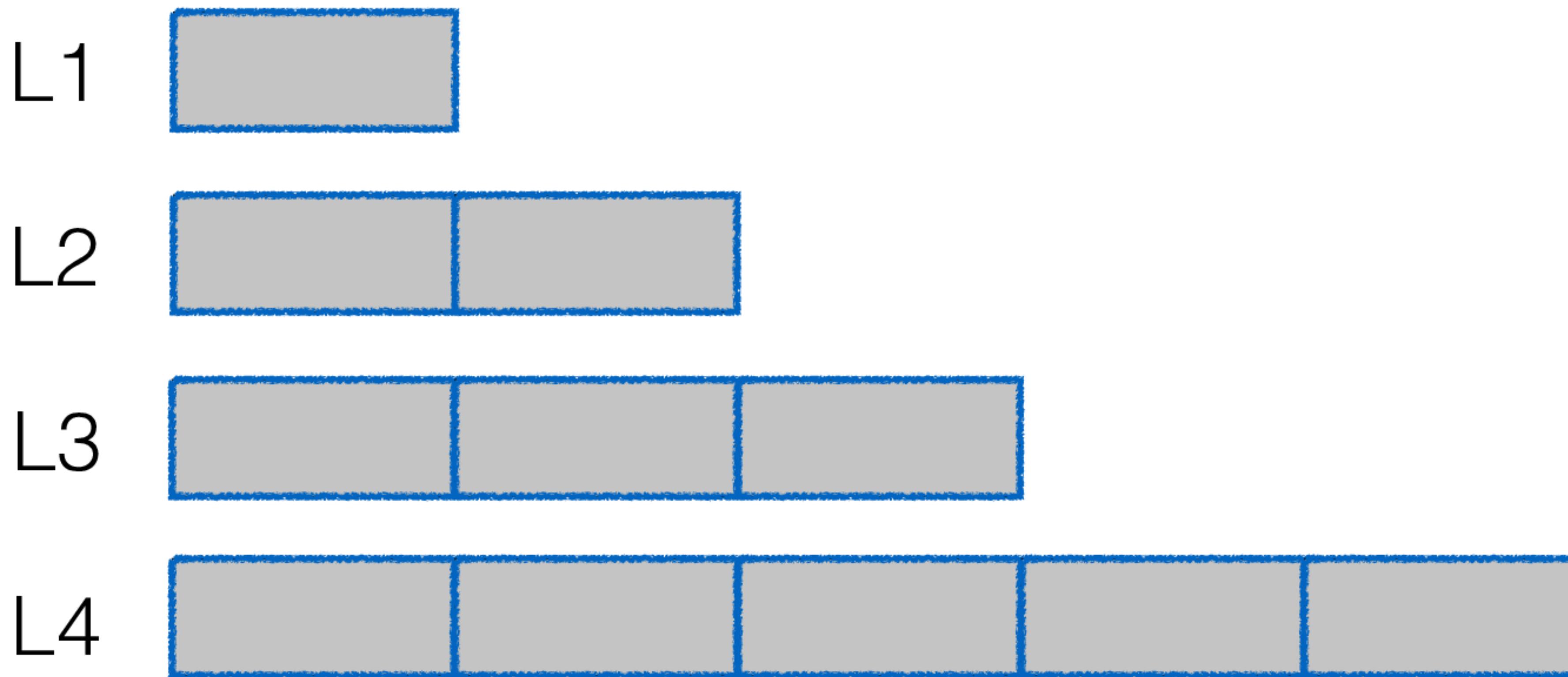
FAst DElete

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



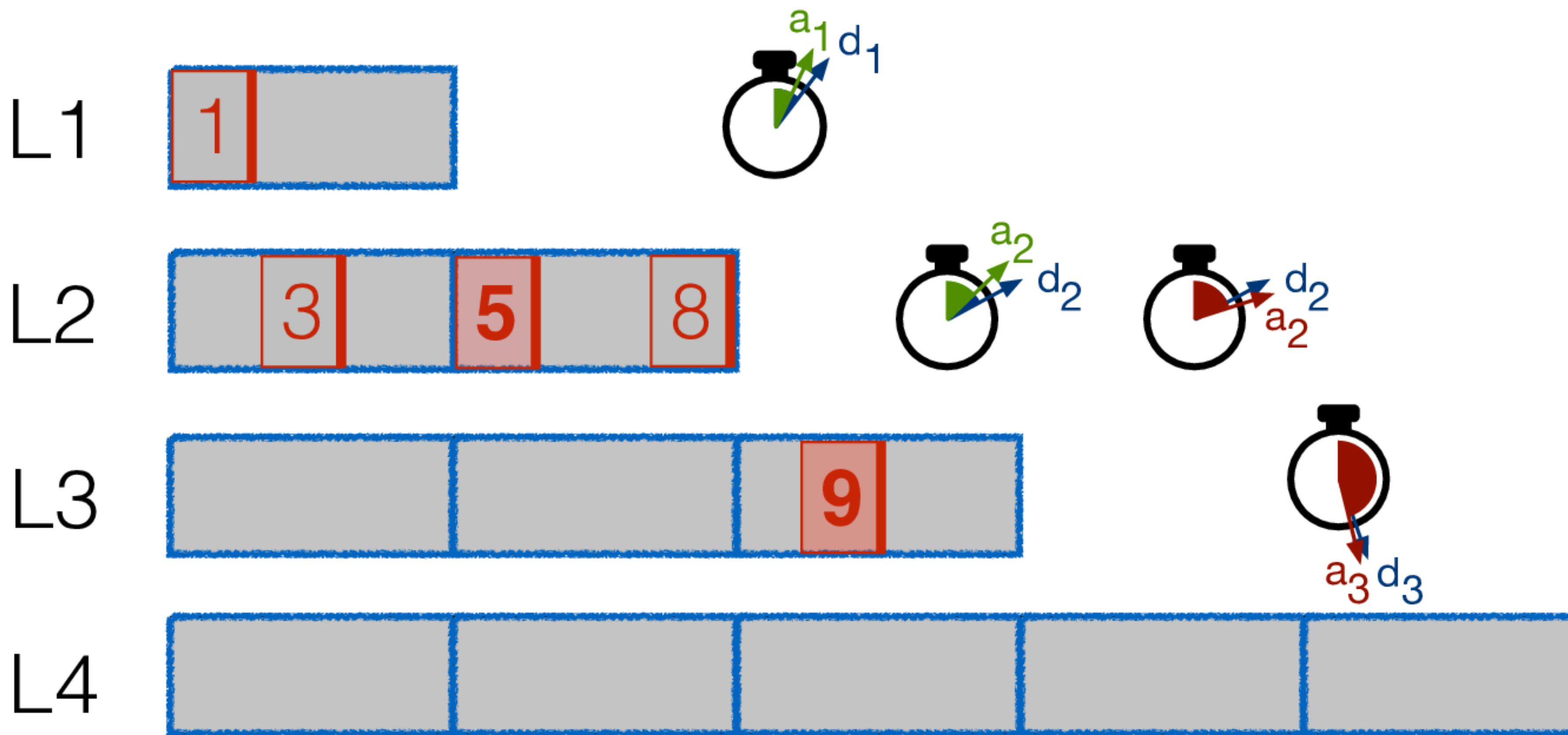
FAst DElete

breaking ties in practical workloads



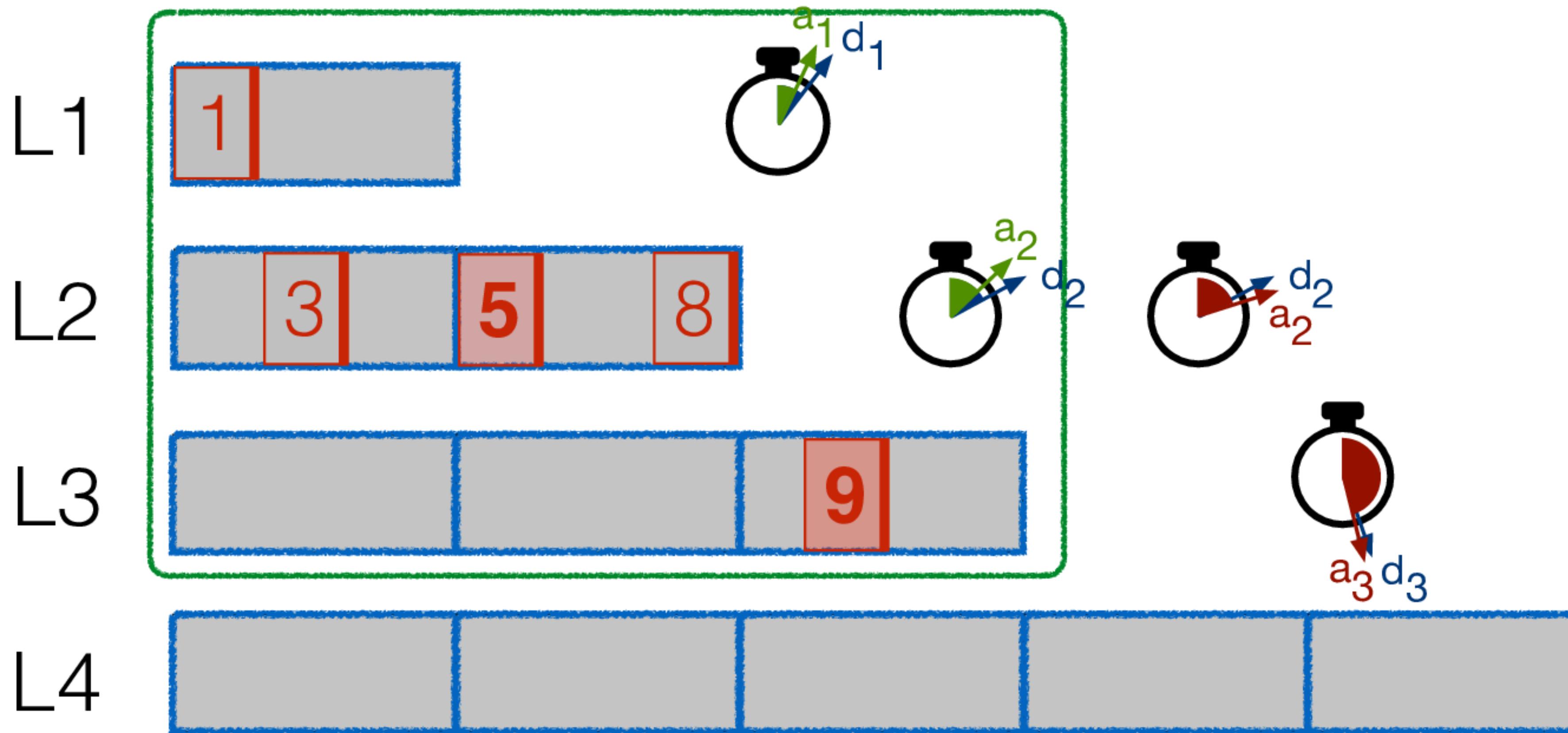
FAst DElete

breaking ties in practical workloads



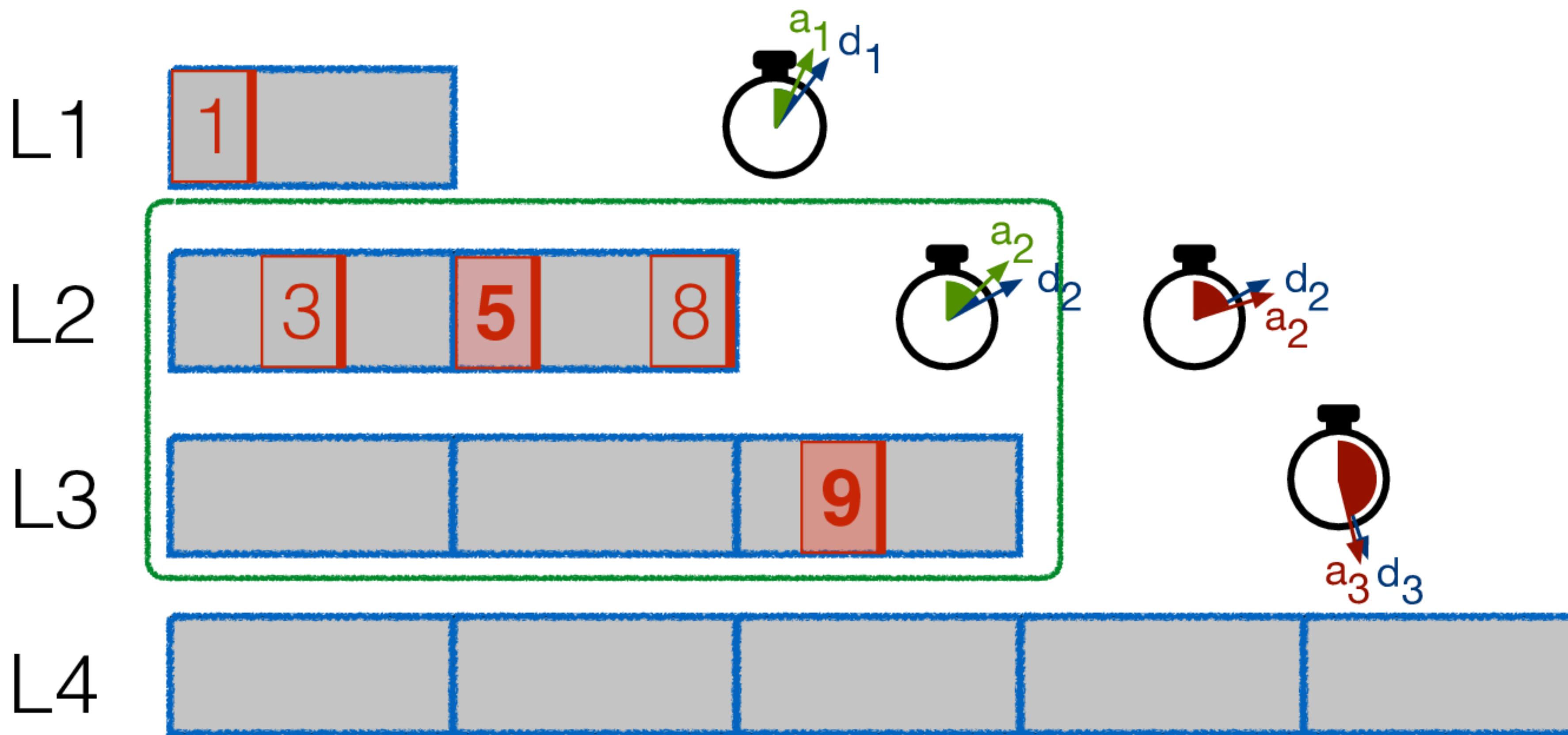
FAst DElete

breaking ties in practical workloads



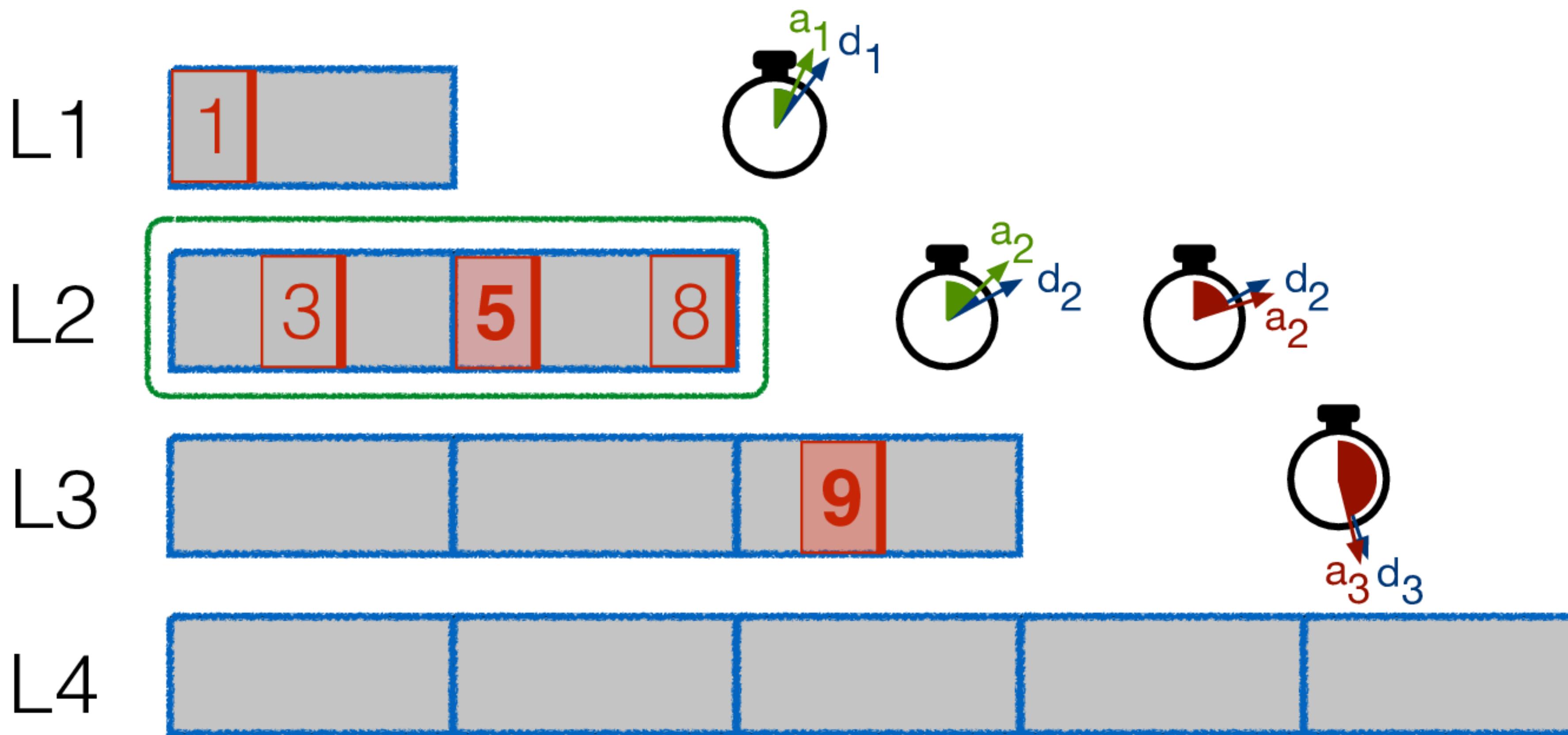
FAst DElete

breaking ties in practical workloads



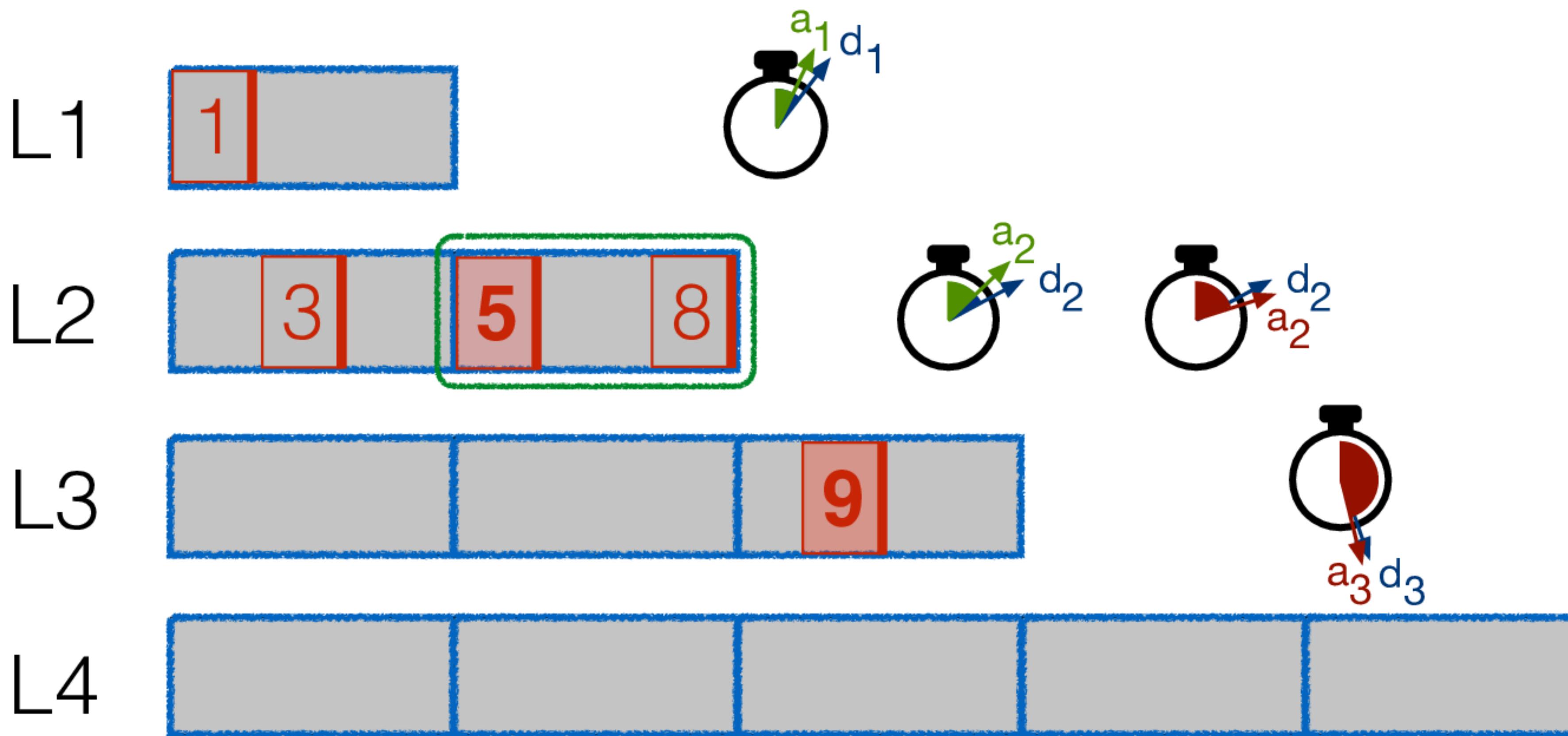
FAst DElete

breaking ties in practical workloads



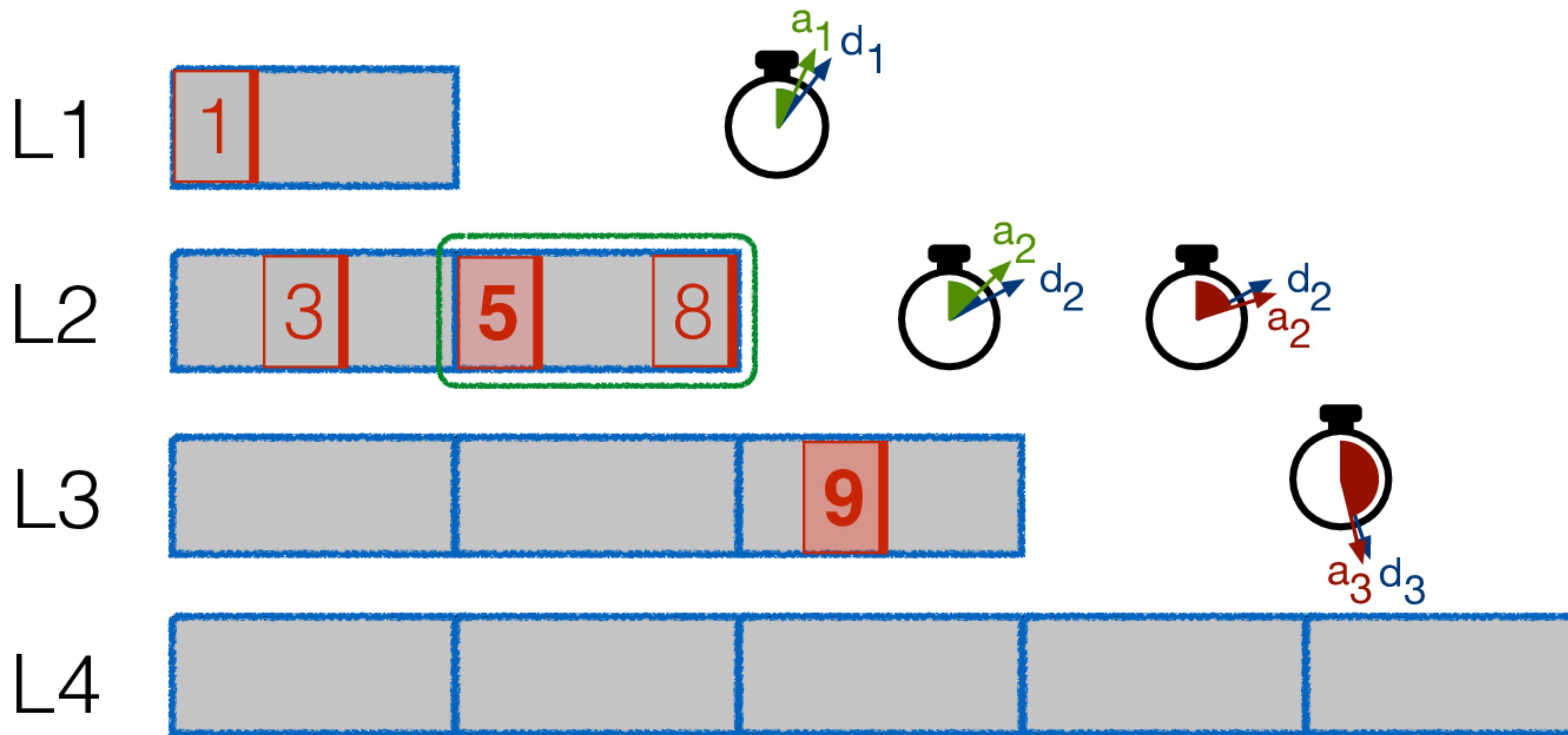
FAst DElete

breaking ties in practical workloads



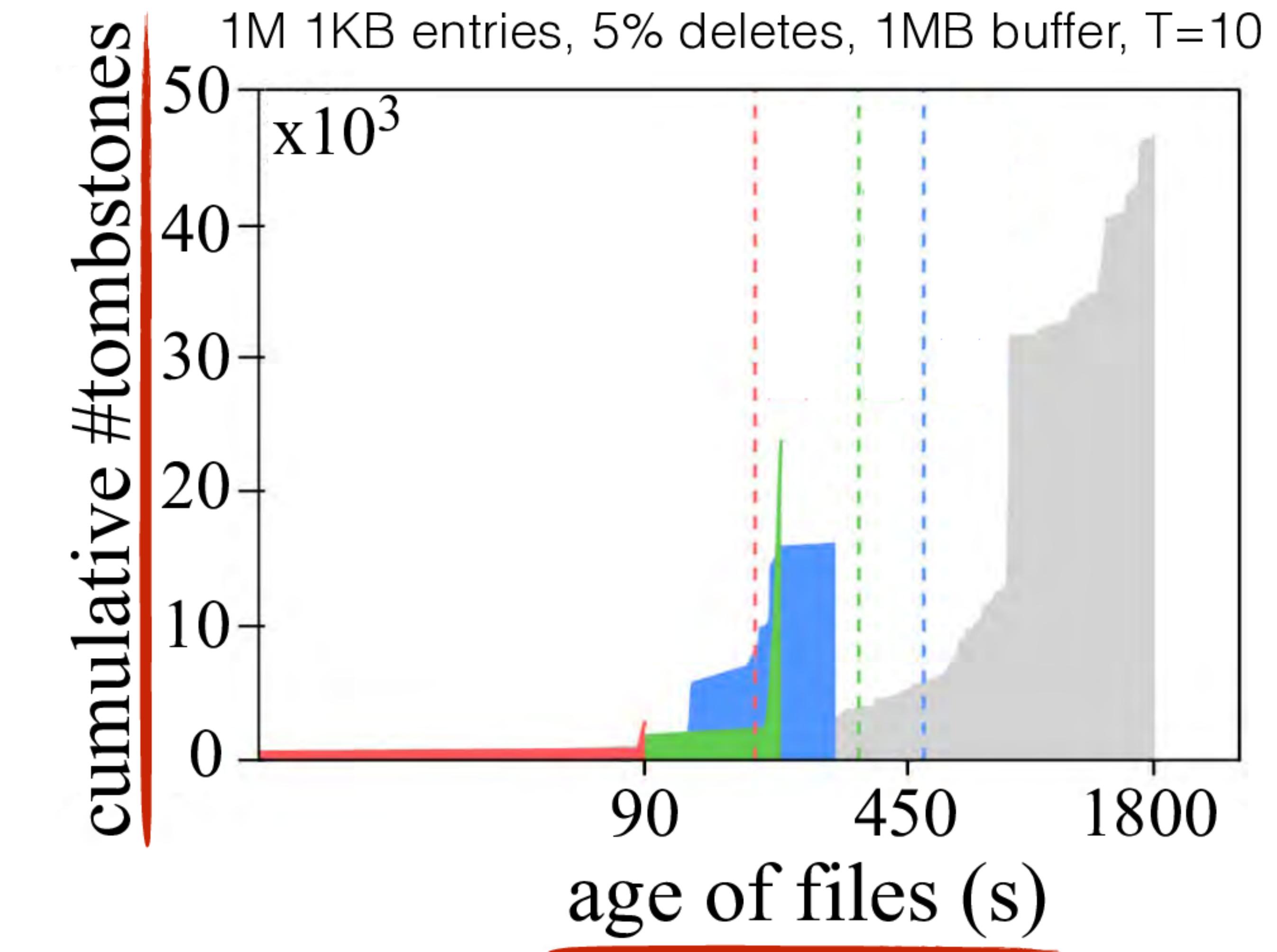
FAst DElete

breaking ties in practical workloads



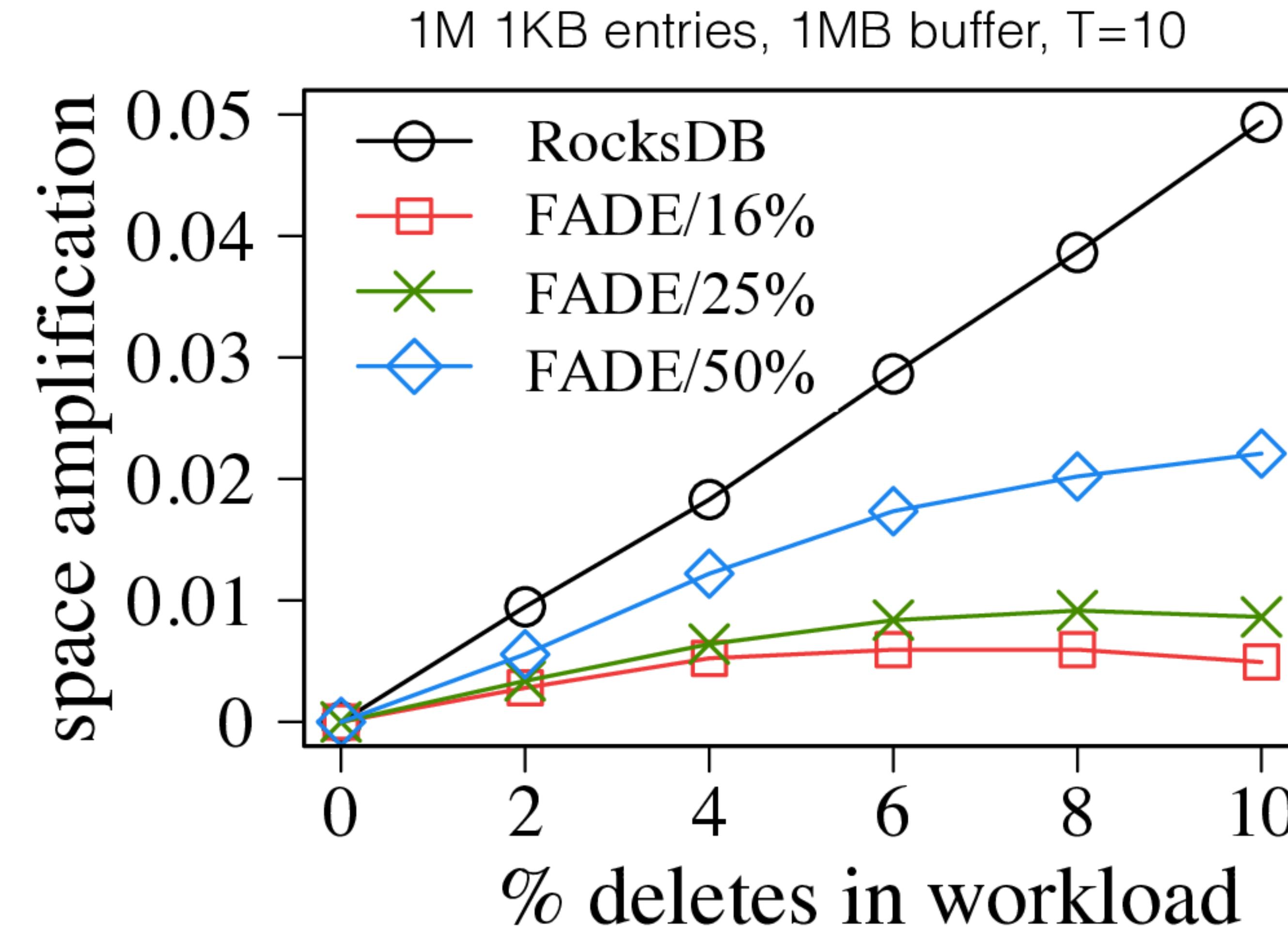
FAst DElete

timely delete persistence
within D_{th}



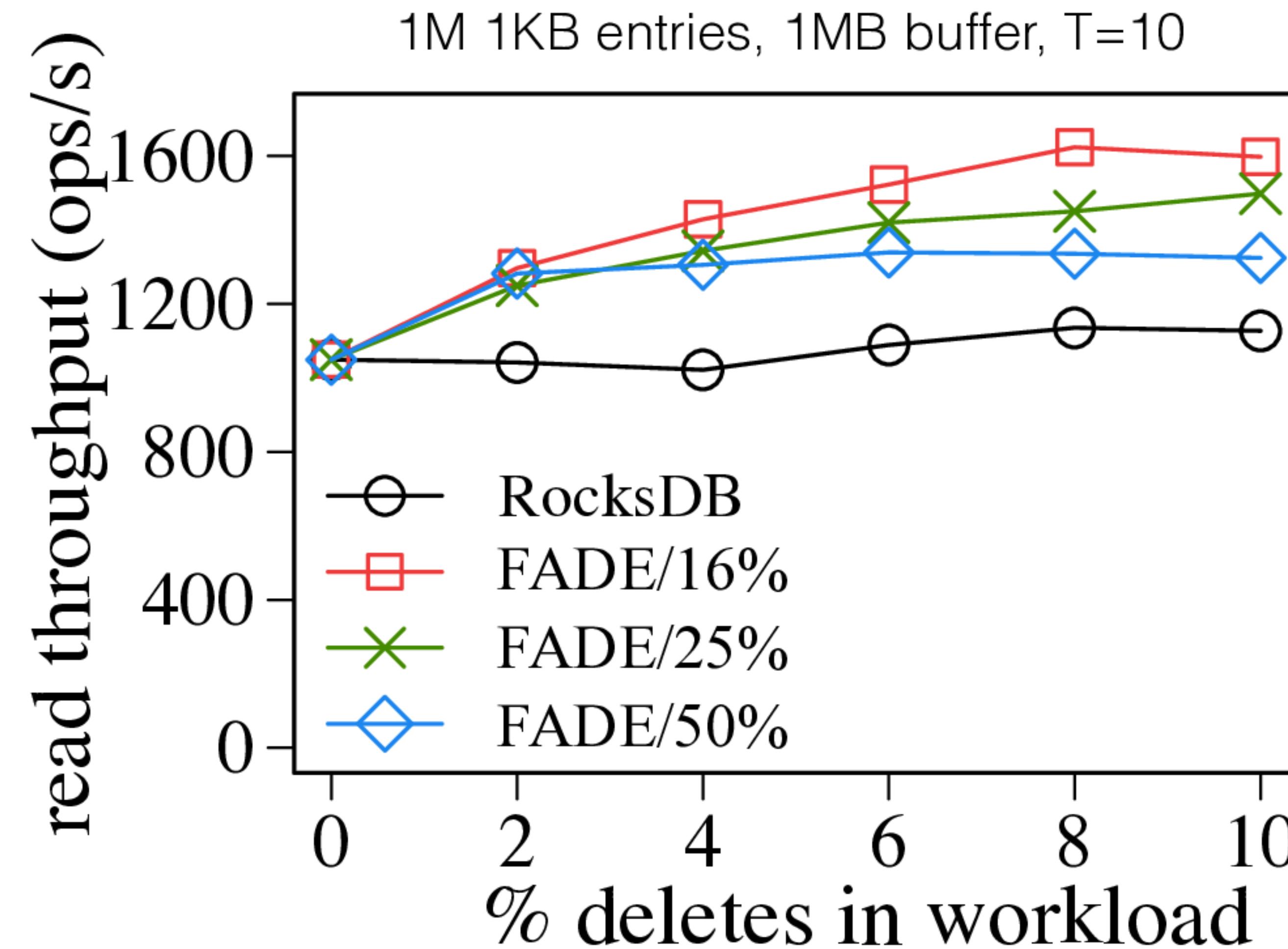
FAst DElete

- reduced space amplification
2.1 - 9.8x ✓
- timely delete persistence
within D_{th} ✓



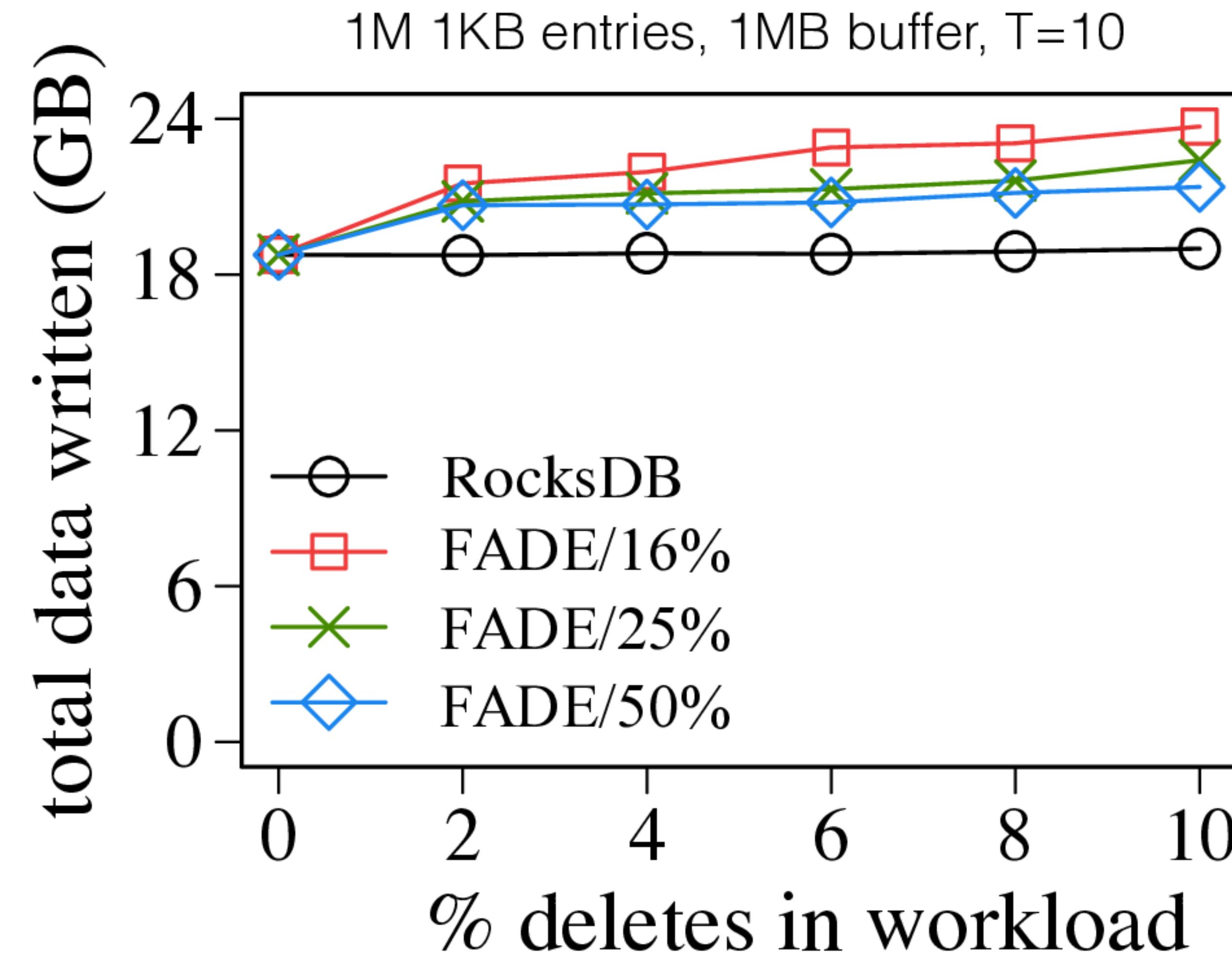
FAst DElete

- improved read performance **1.2 - 1.4x** ✓
- reduced space amplification **2.1 - 9.8x** ✓
- timely delete persistence **within D_{th}** ✓



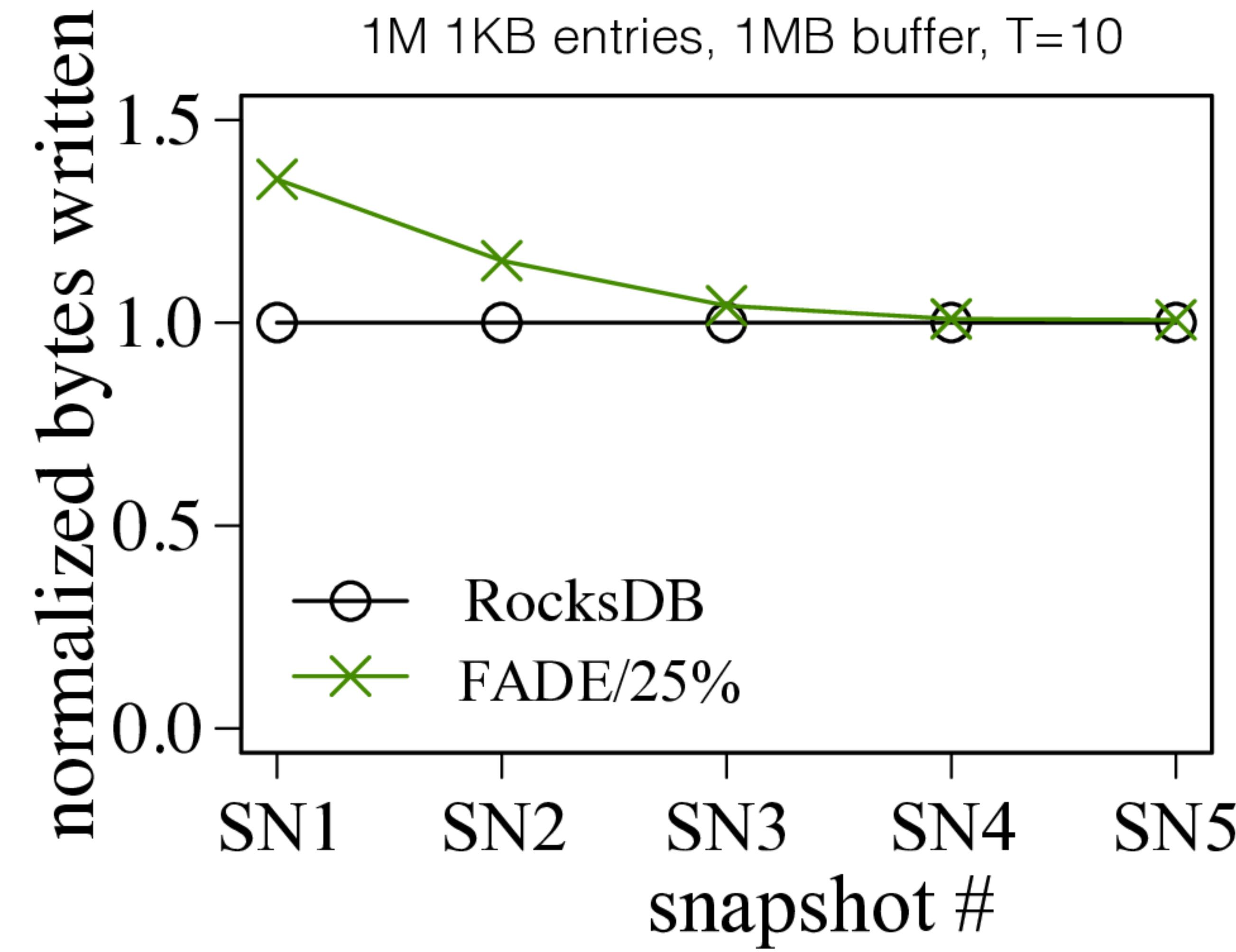
FAst DElete

- higher write amplification
4 - 25% ↑
- improved read performance
1.2 - 1.4x ✓
- reduced space amplification
2.1 - 9.8x ✓
- timely delete persistence
within D_{th} ✓



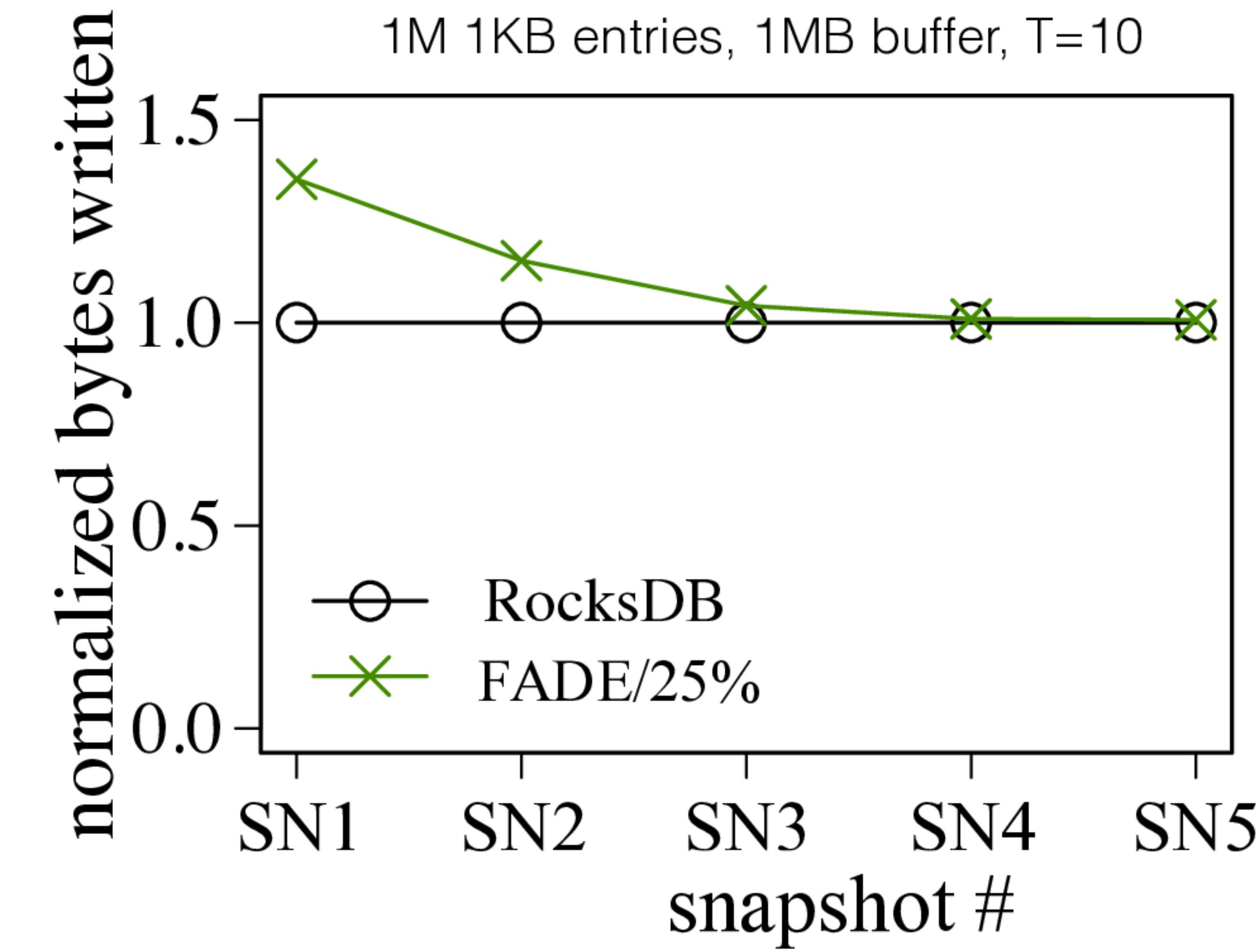
FAst DElete

- higher write amplification
4 - 25% ↑
- improved read performance
1.2 - 1.4x ✓
- reduced space amplification
2.1 - 9.8x ✓
- timely delete persistence
within D_{th} ✓



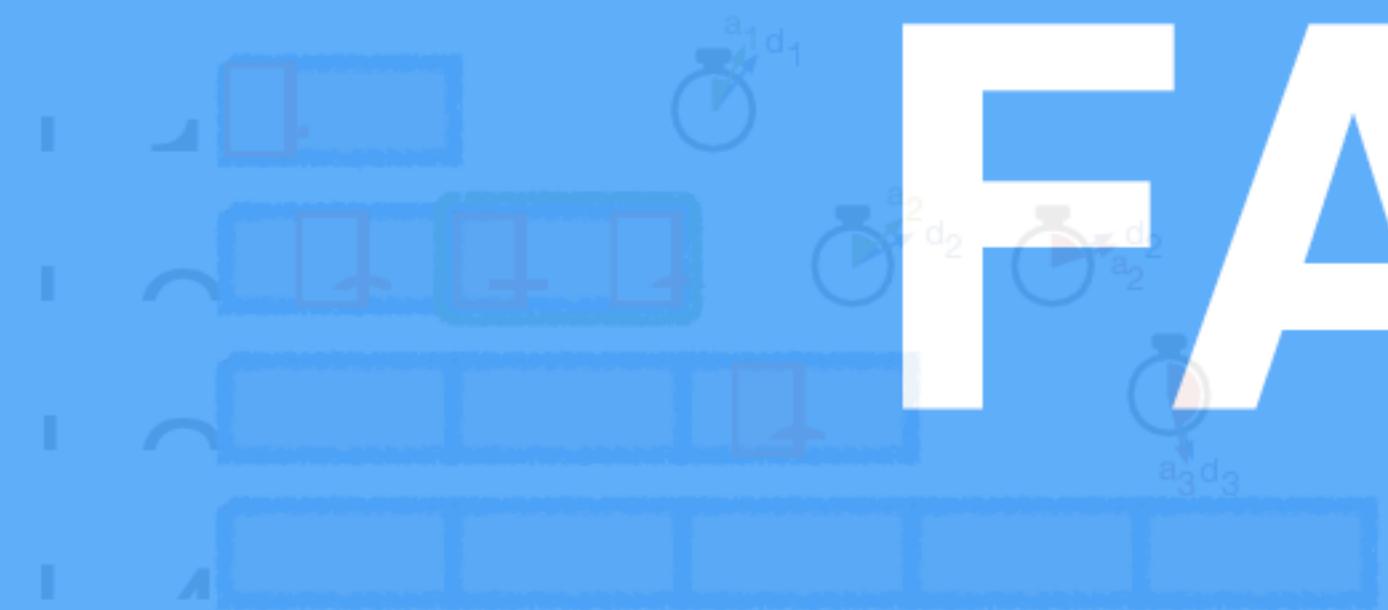
FAst DElete

- higher write amplification
0.7 %
- improved read performance
1.2 - 1.4x
- reduced space amplification
2.1 - 9.8x
- timely delete persistence
within D_{th}



the solution

FAst DElete



FADE

higher write amplification

improved read performance

reduced space amplification

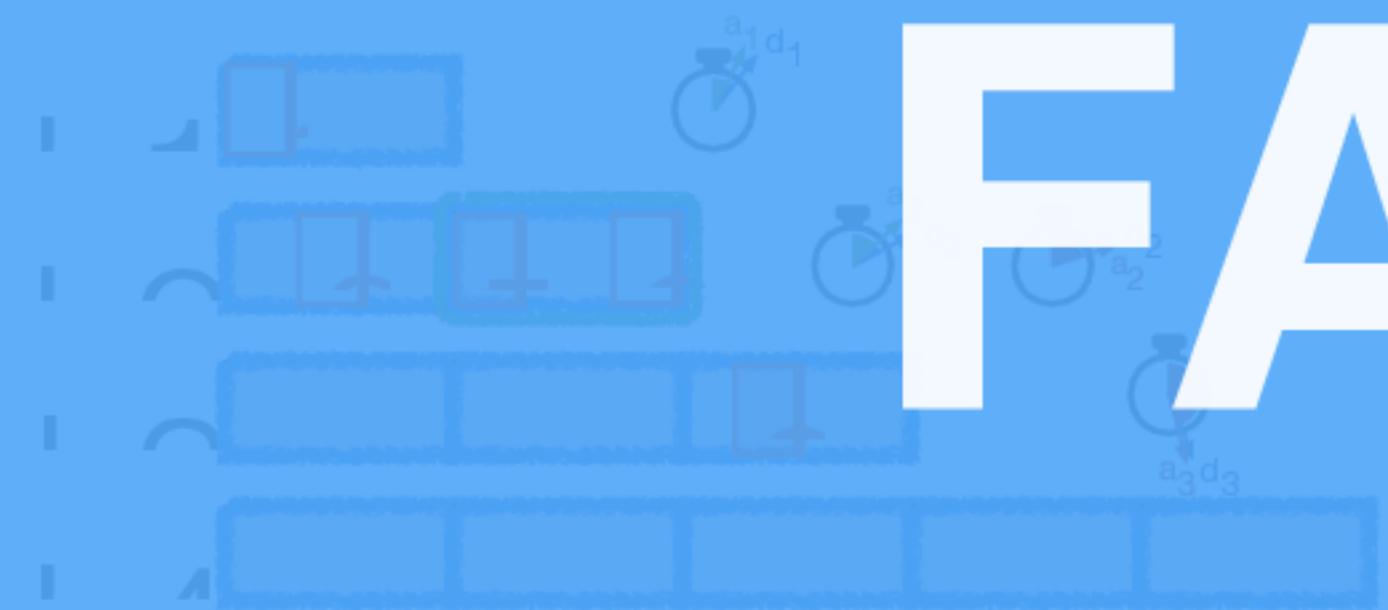
timely delete persistence

latency spikes

KiWi

the solution

FAst DElete



higher write amplification

↑ read performance

reduced space amplification

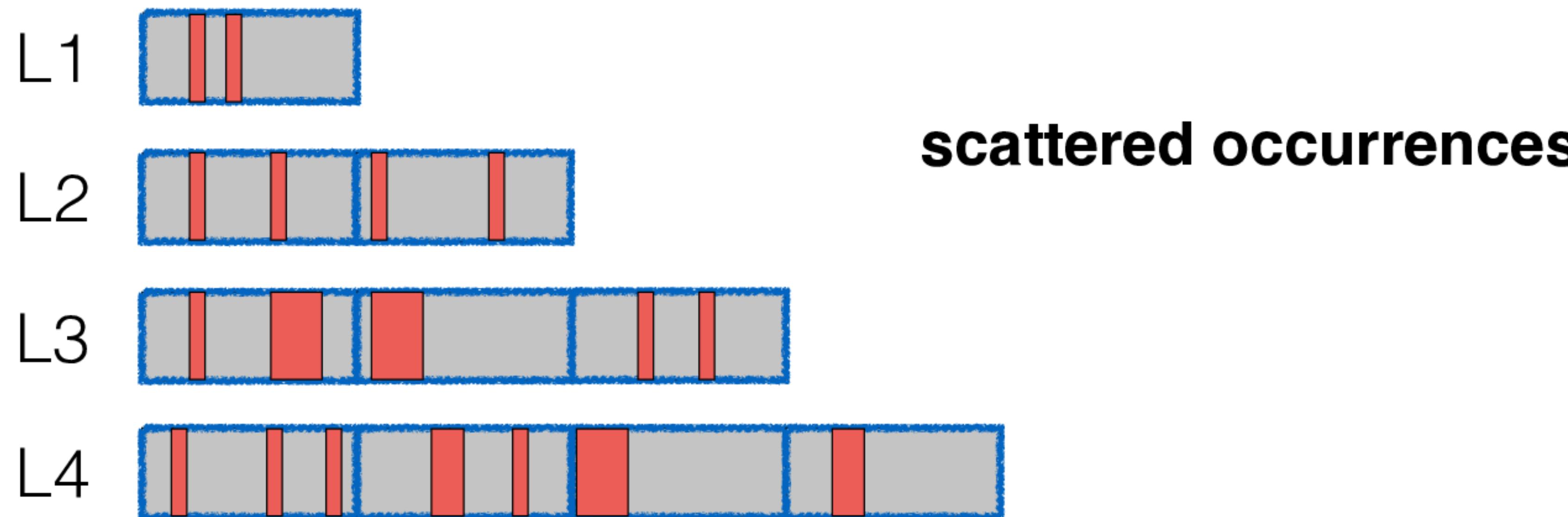
timely delete persistence

latency spikes

KiWi

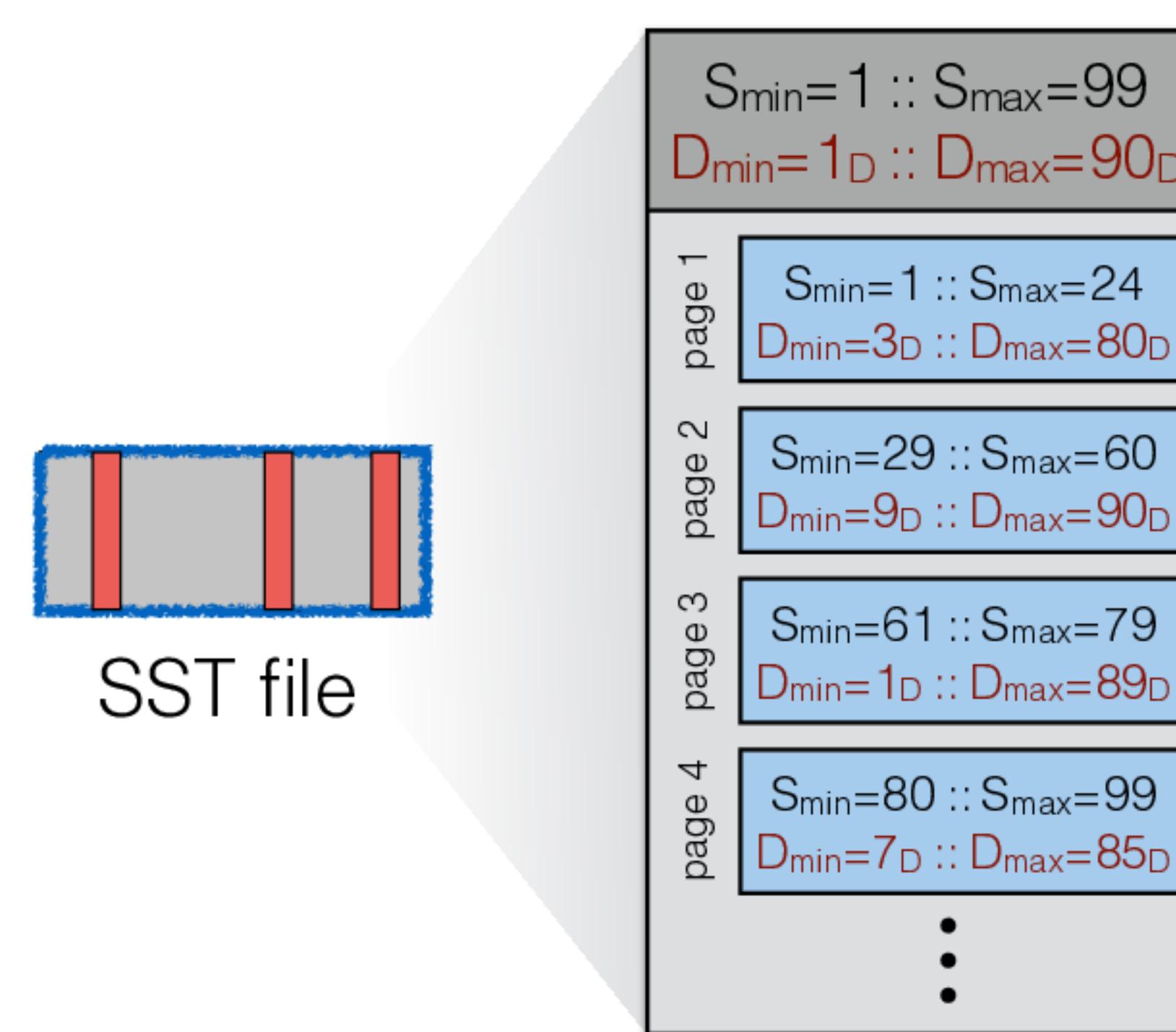
Key Weaving storage layout

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



Key Weaving storage layout

delete all entries with timestamp $\leq 65_D$



SST file

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	

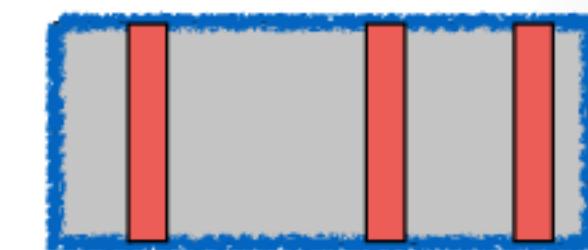
page 2								
29	32	33	40	44	52	56	60	
88 _D	90 _D	28 _D	74 _D	9 _D	76 _D	81 _D	64 _D	

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 $\leq 65_D$



SST file

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

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

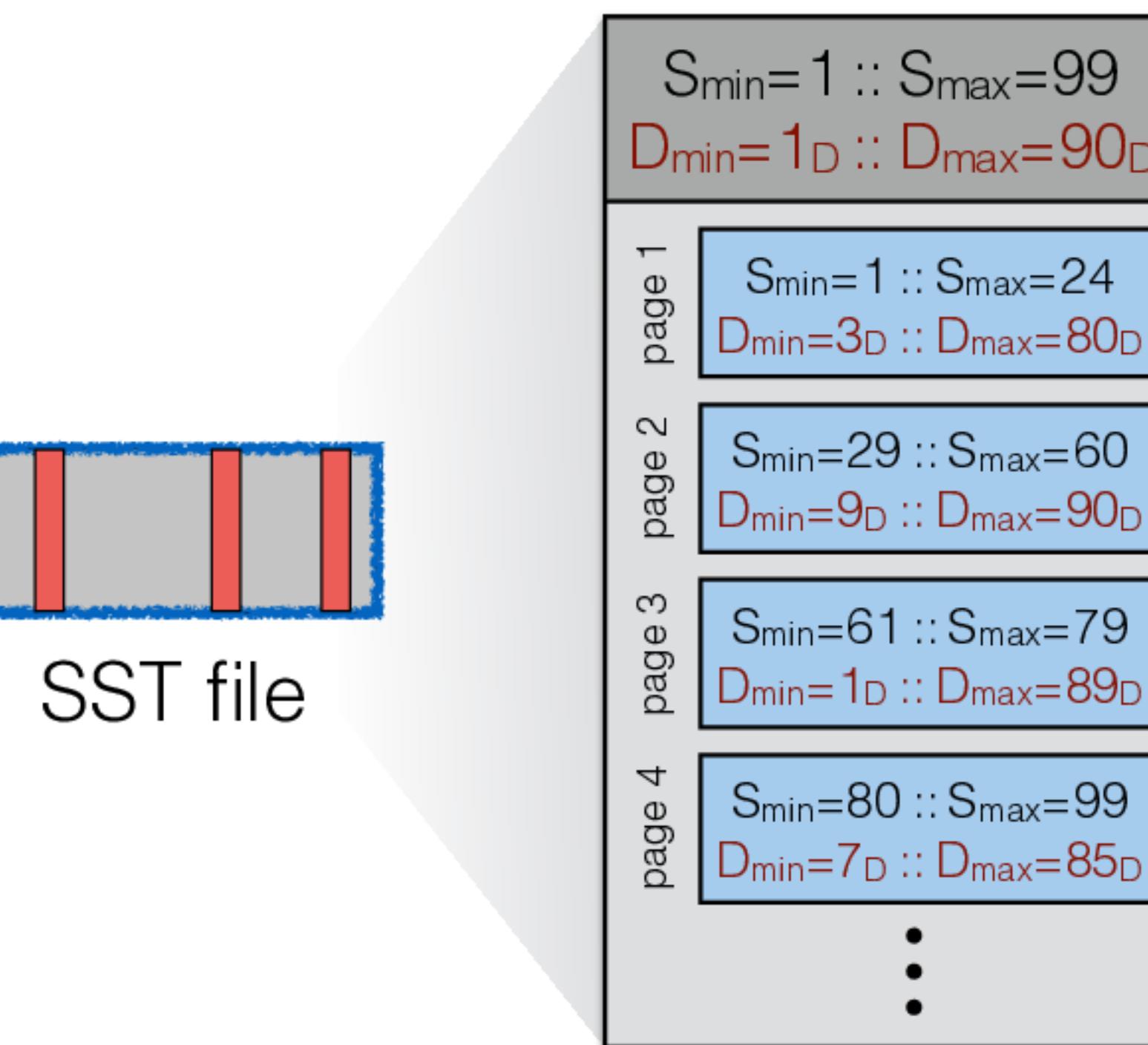
1 I/O

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

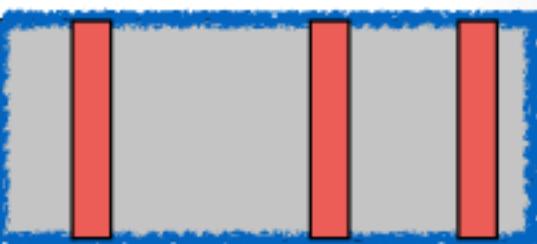
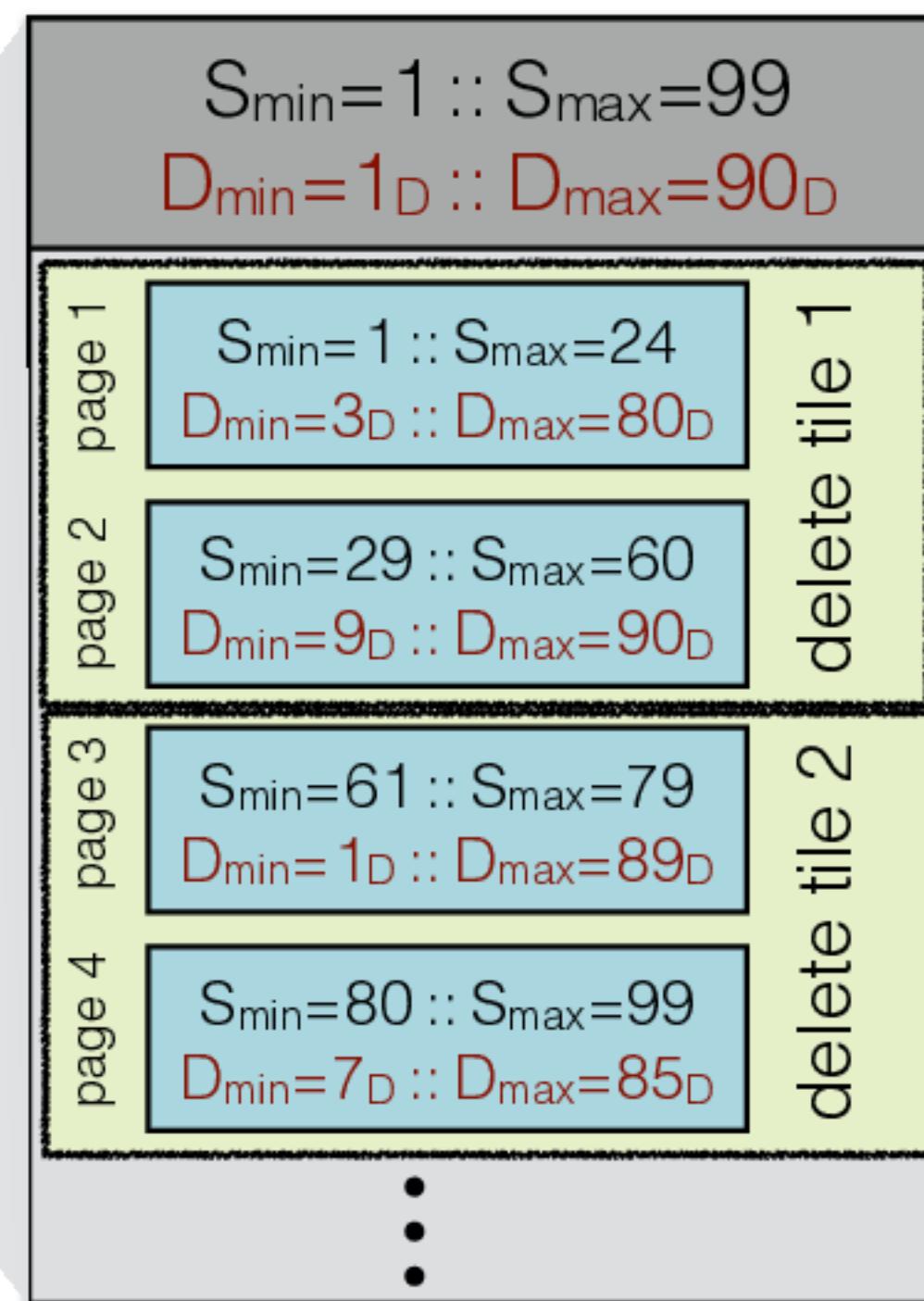
delete all entries with timestamp $\leq 65_D$



SST file

Key Weaving storage layout

delete all entries with timestamp $\leq 65_D$

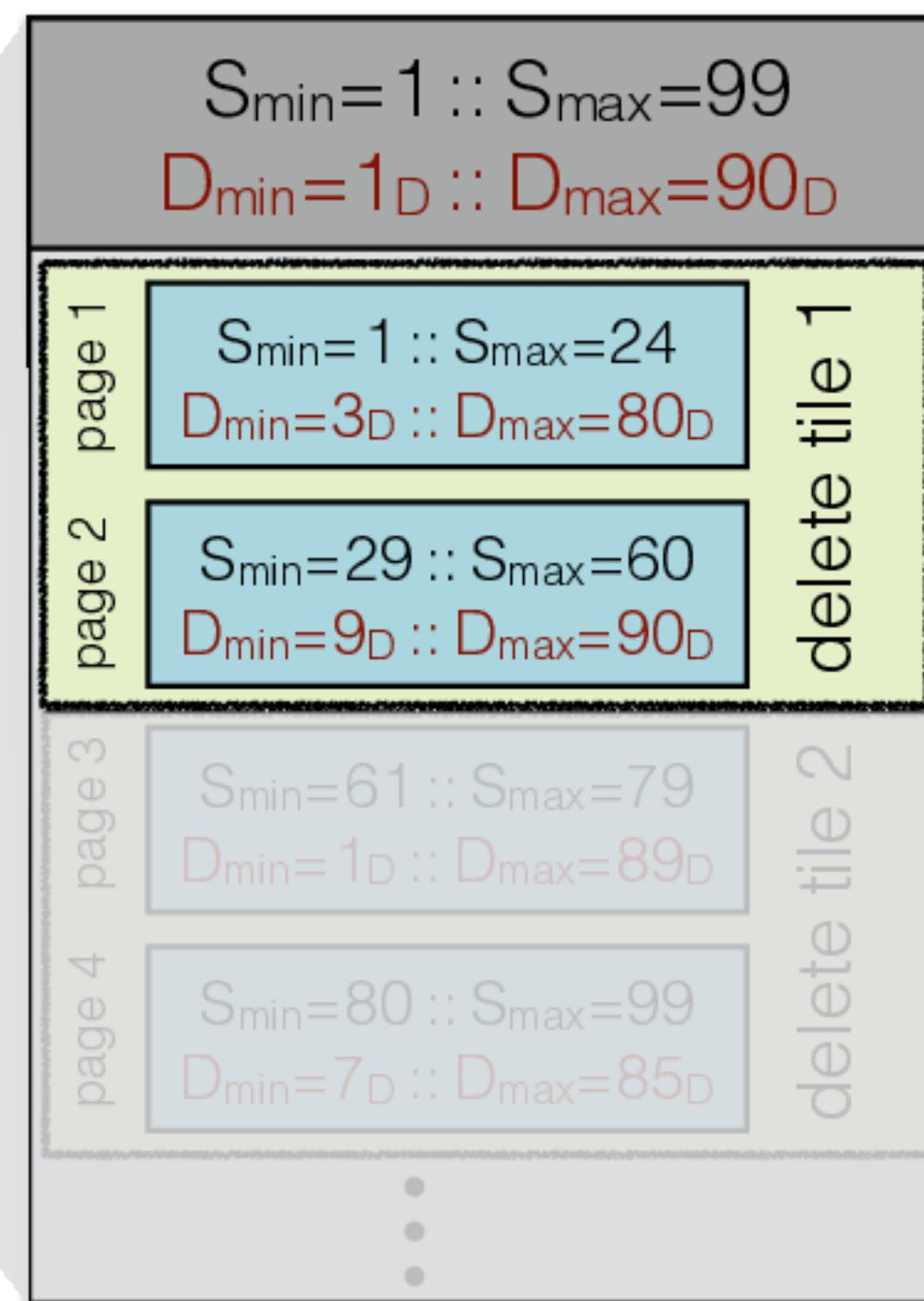


SST file

partitioned on S

Key Weaving storage layout

delete all entries with timestamp $\leq 65_D$

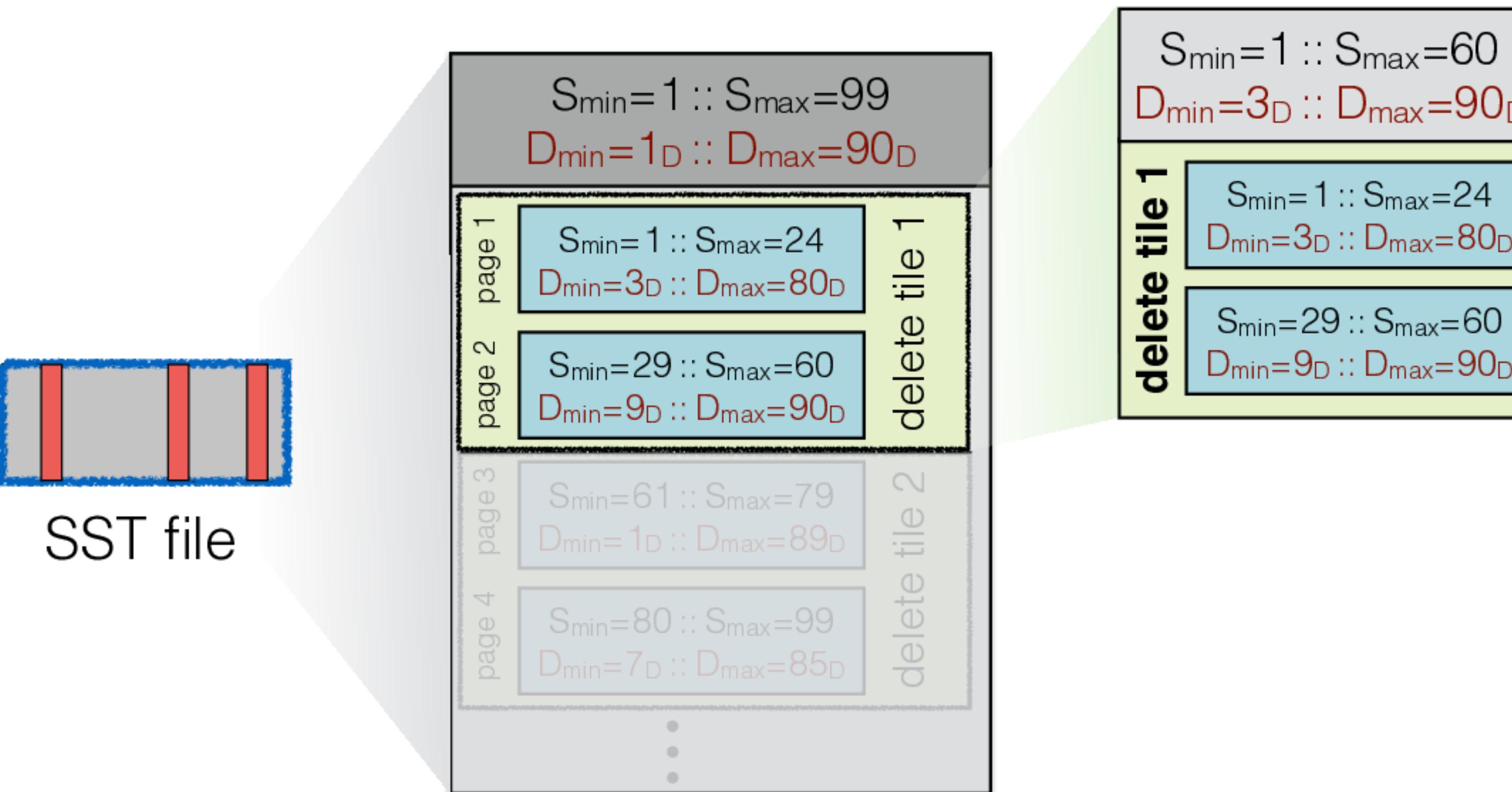


SST file

partitioned on S

Key Weaving storage layout

delete all entries with timestamp $\leq 65_D$

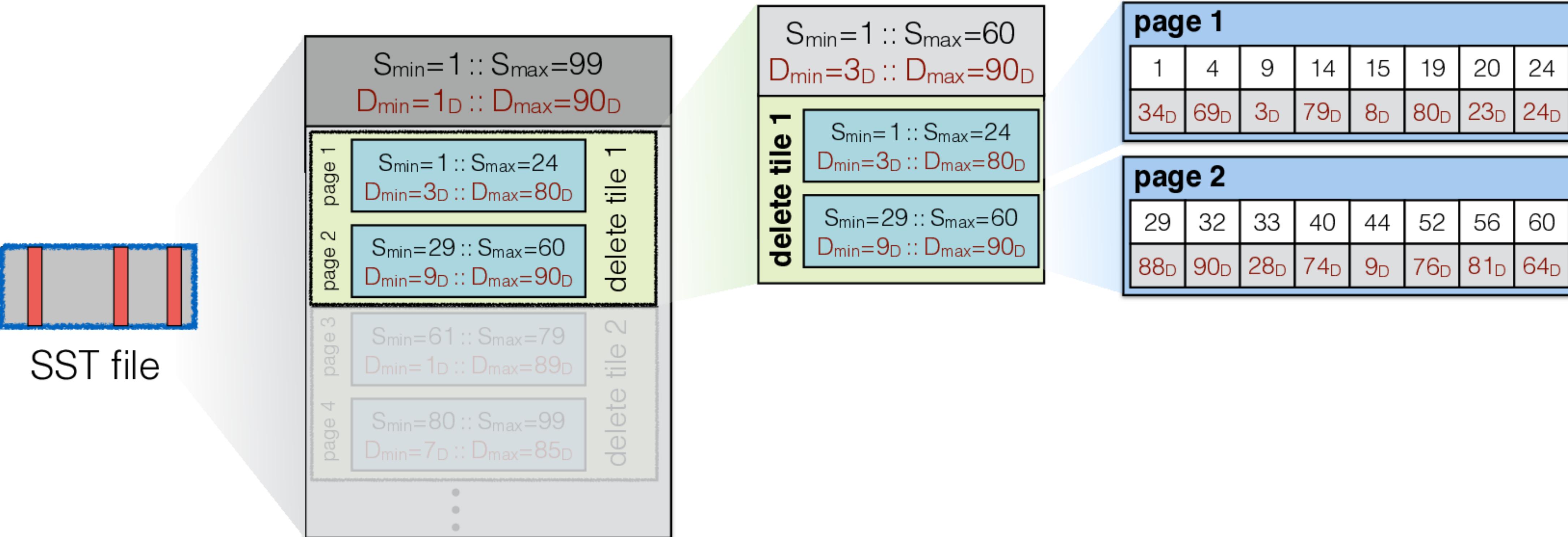


SST file

partitioned on S

Key Weaving storage layout

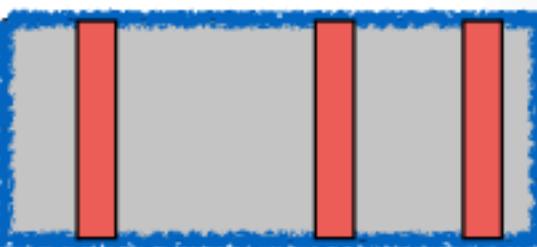
delete all entries with timestamp $\leq 65_D$



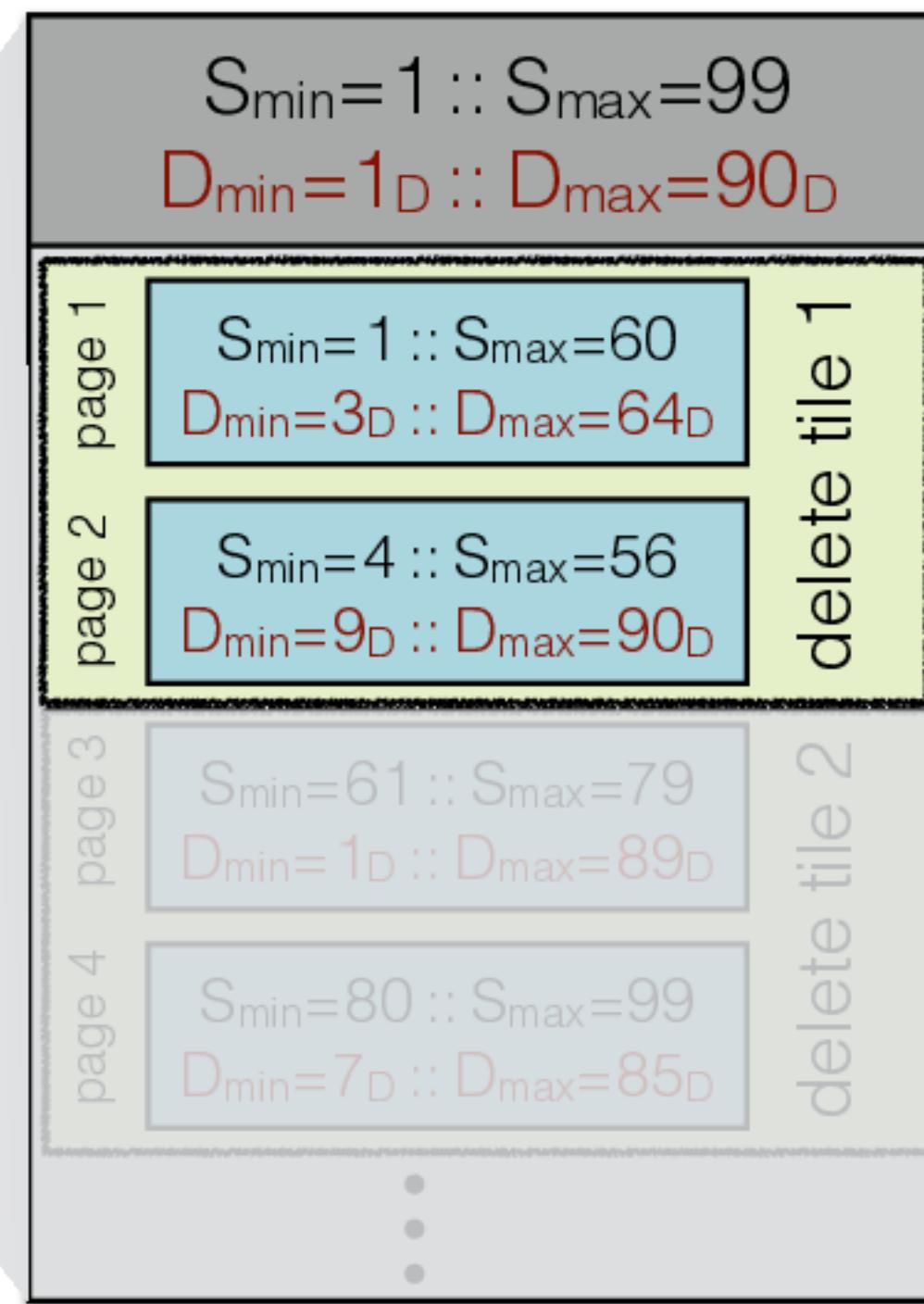
partitioned on S

Key Weaving storage layout

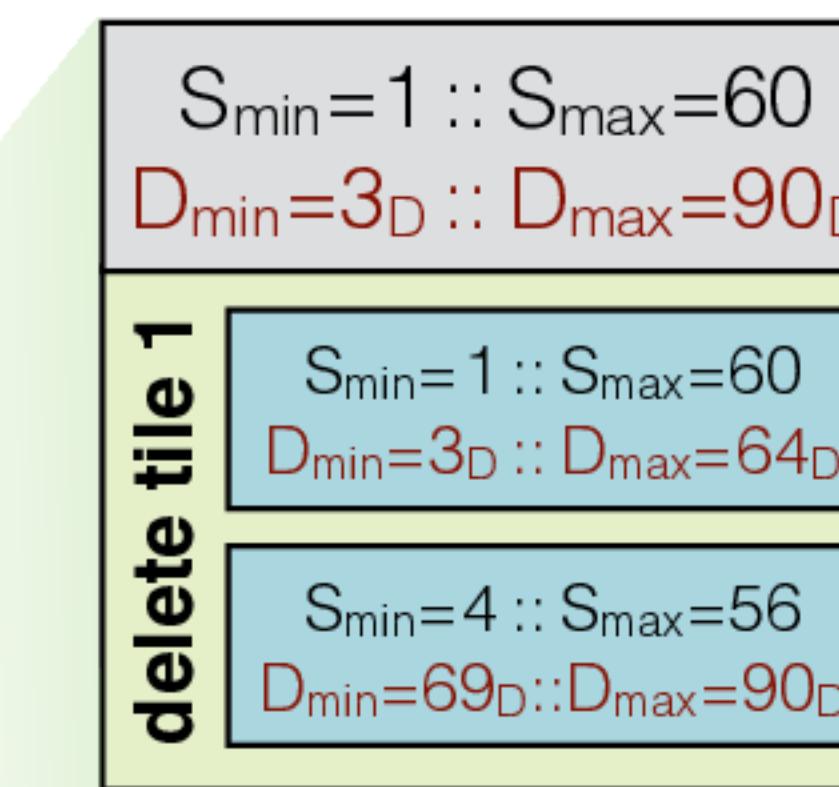
delete all entries with timestamp $\leq 65_D$



SST file



partitioned on S

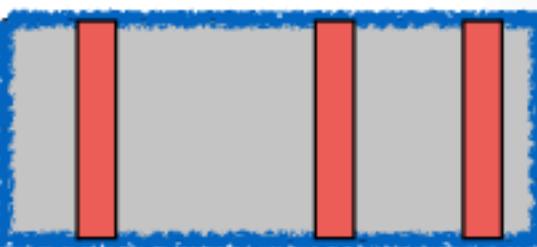


partitioned on D

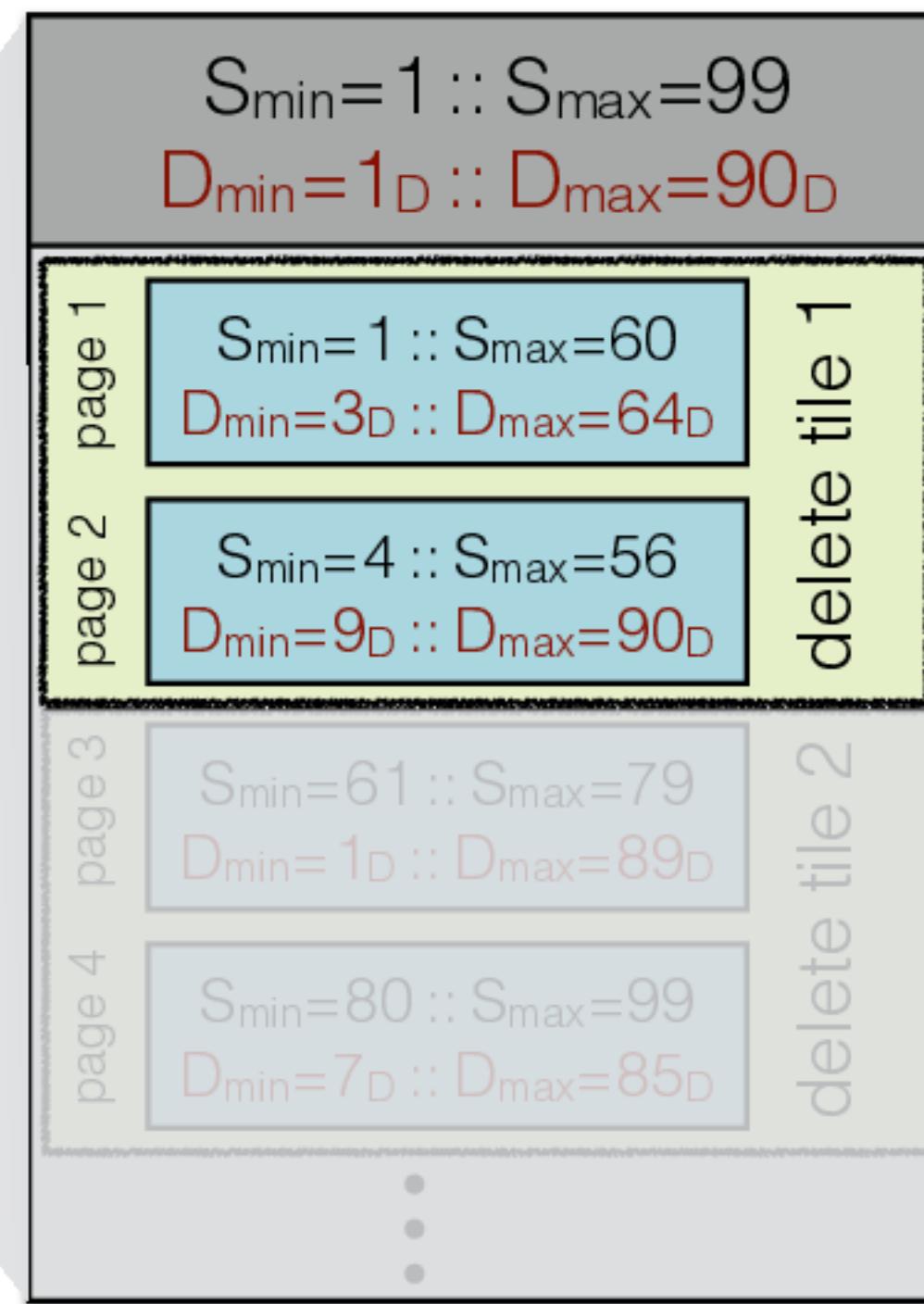
page 1							
9	15	44	20	24	33	1	60
3_D	8_D	9_D	23_D	24_D	28_D	34_D	64_D
page 2							
4	40	52	14	19	56	29	32
69_D	74_D	76_D	79_D	80_D	81_D	88_D	90_D

Key Weaving storage layout

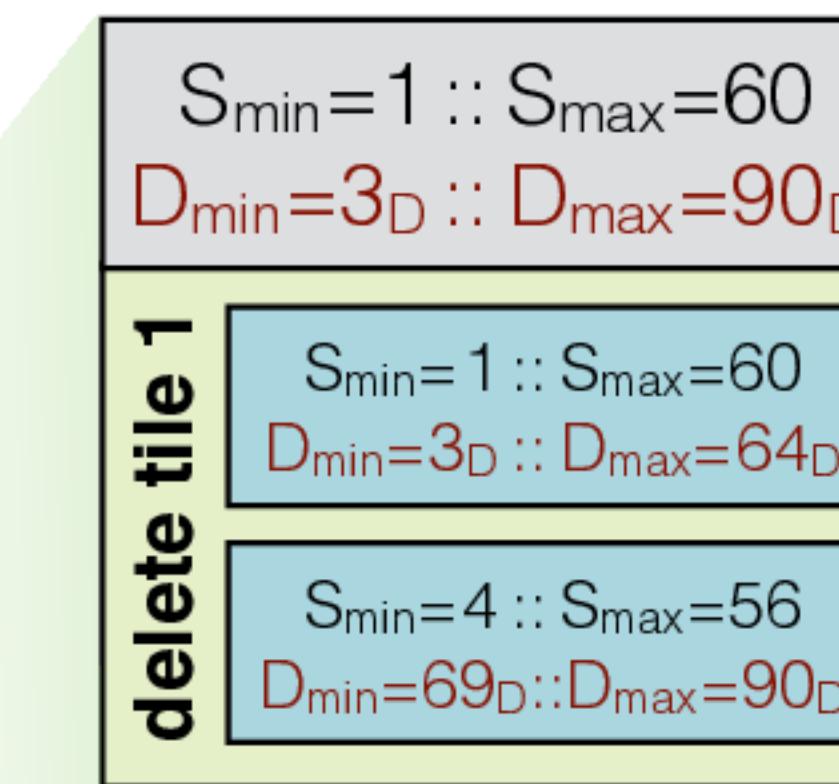
delete all entries with timestamp $\leq 65_D$



SST file



partitioned on S



partitioned on D

Two tables representing pages of data. Page 1 has 8 columns and page 2 has 8 columns. Red numbers indicate deleted entries.

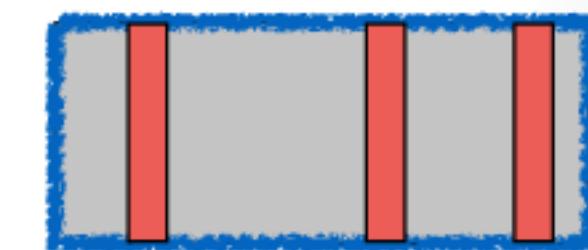
page 1							
9	15	44	20	24	33	1	60
3_D	8_D	9_D	23_D	24_D	28_D	34_D	64_D

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

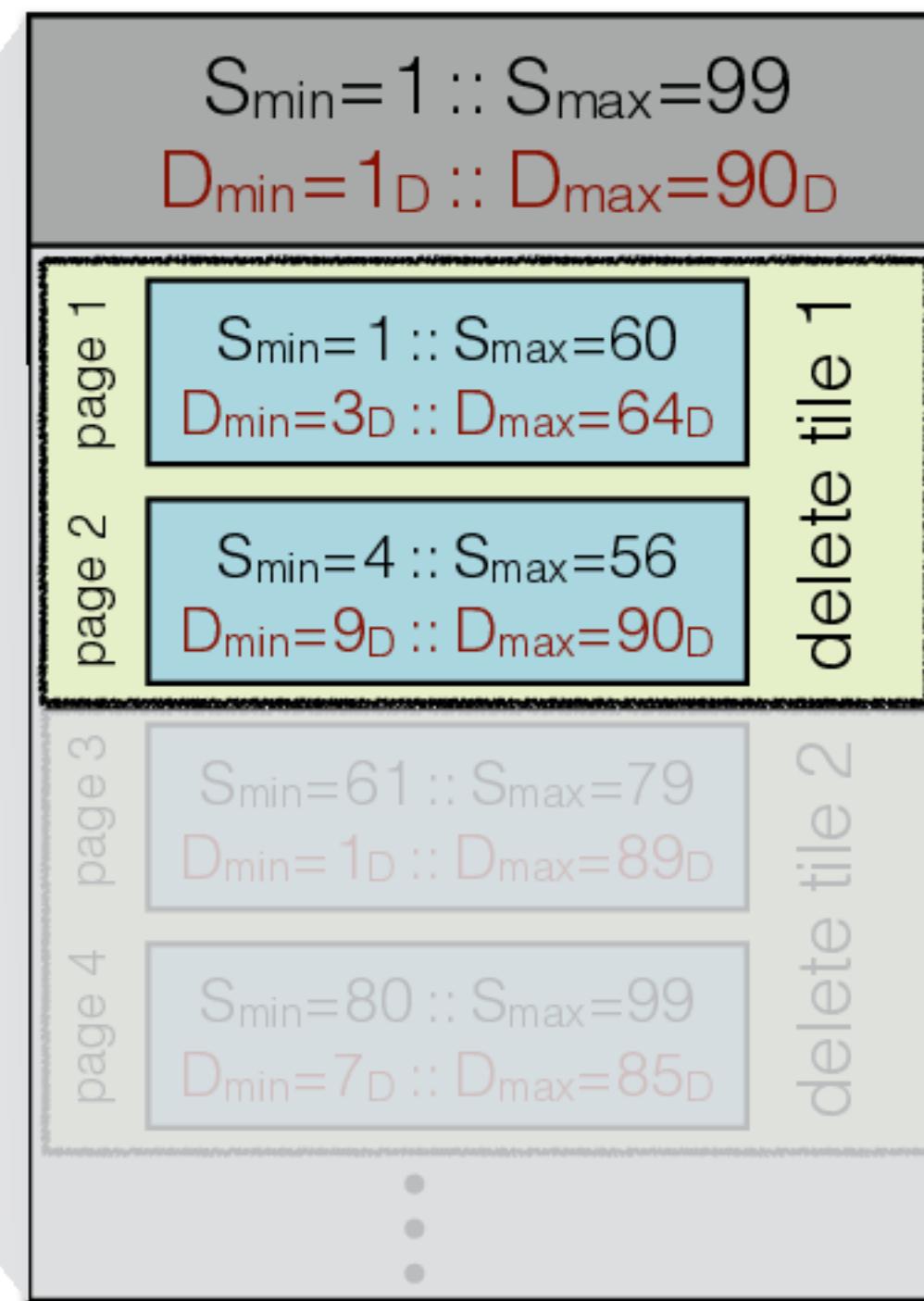
drop page

Key Weaving storage layout

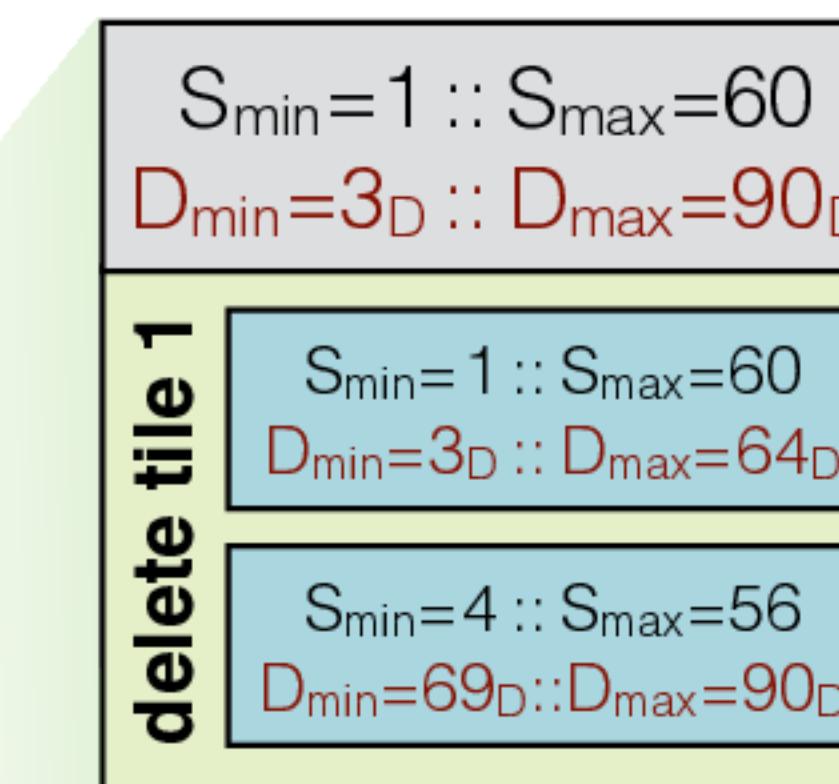
delete all entries with timestamp $\leq 65_D$



SST file



partitioned on S



partitioned on D

Two tables representing pages from the SST file. **page 1** contains entries: 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** contains entries: 4, 14, 19, 29, 32, 40, 52, 56, 69_D, 79_D, 80_D, 88_D, 90_D, 74_D, 76_D, 81_D. The text "drop page" is written vertically next to the tables.

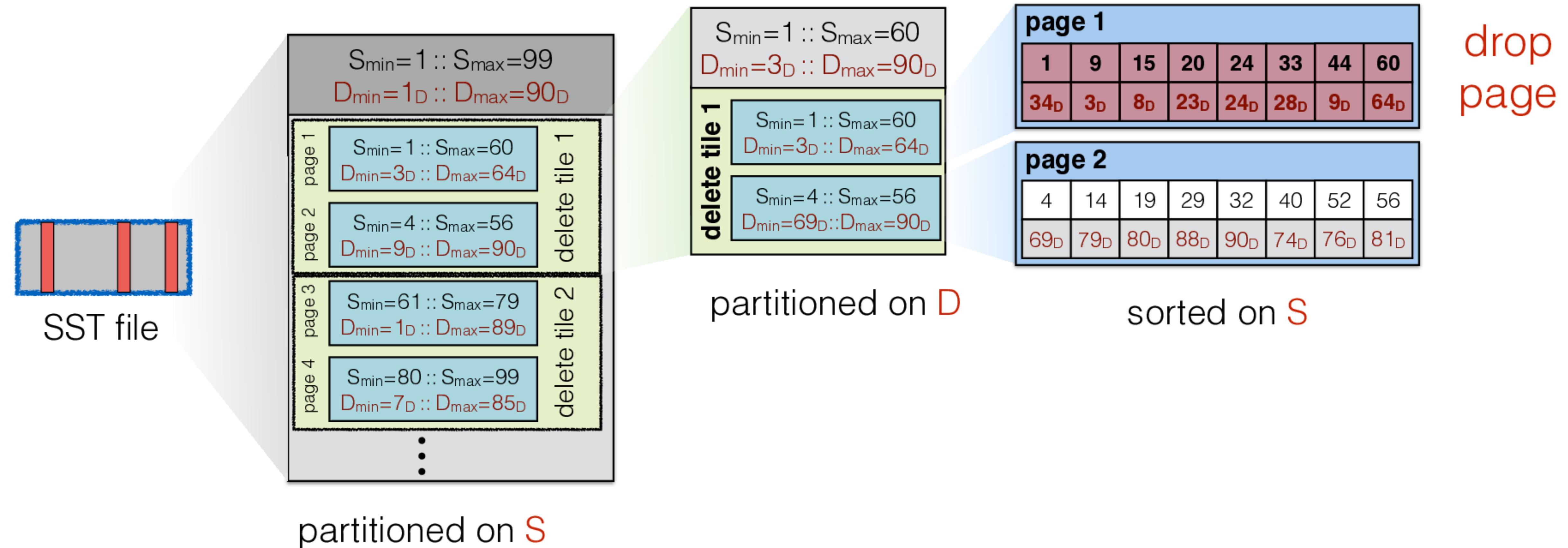
page 1							
1	9	15	20	24	33	44	60
34 _D	3 _D	8 _D	23 _D	24 _D	28 _D	9 _D	64 _D
page 2							
4	14	19	29	32	40	52	56
69 _D	79 _D	80 _D	88 _D	90 _D	74 _D	76 _D	81 _D

sorted on S

drop
page

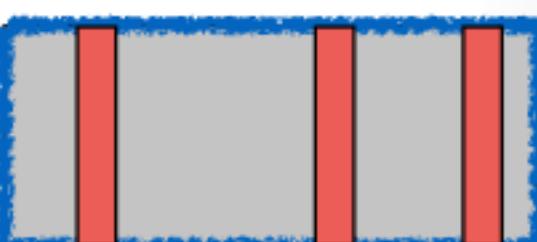
Key Weaving storage layout

delete all entries with timestamp $\leq 65\text{D}$

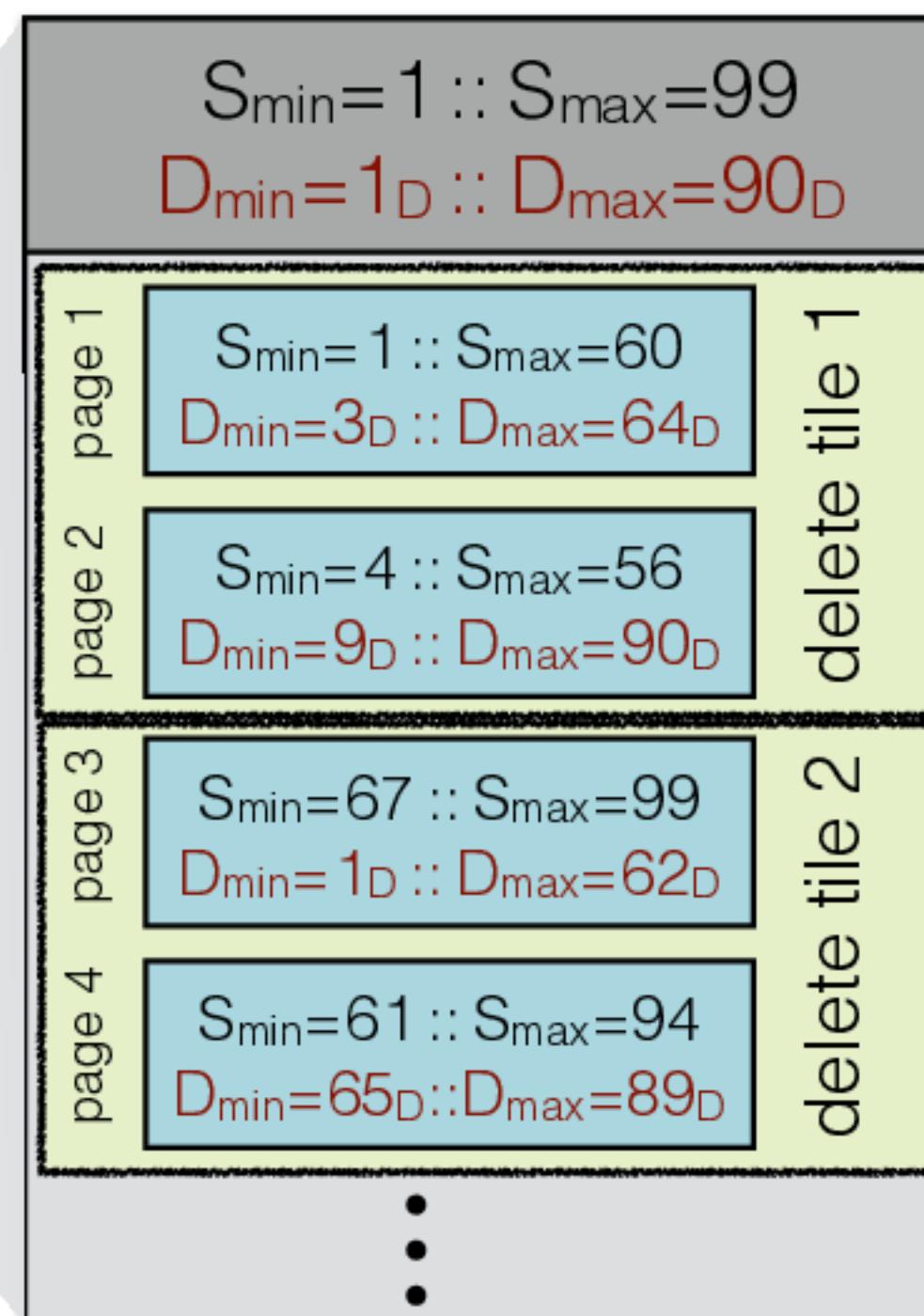


Key Weaving storage layout

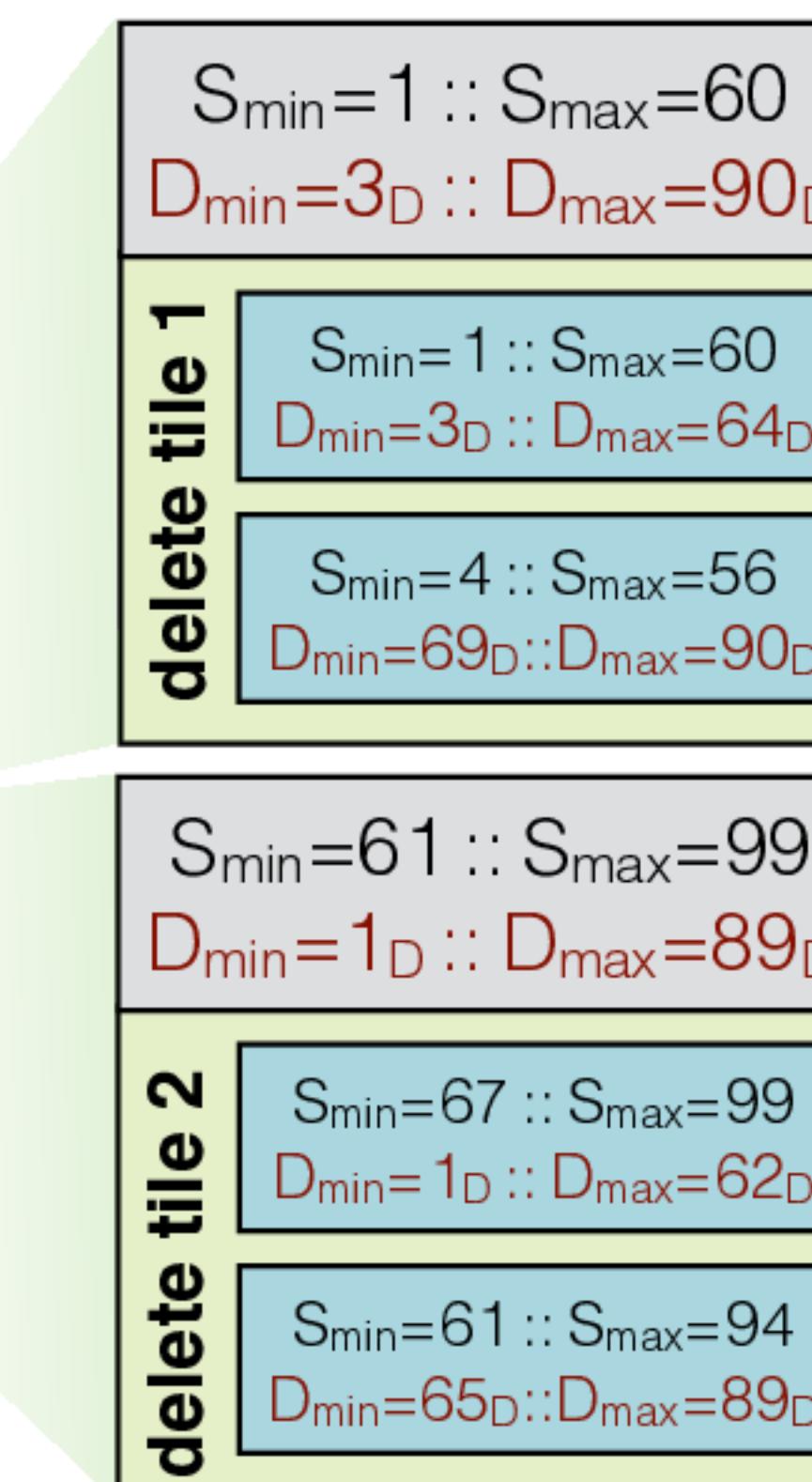
delete all entries with timestamp $\leq 65_D$



SST file



partitioned on S



partitioned on D

The storage layout consists of four pages:

- page 1:** Contains values 1 through 60. Red boxes highlight values 34, 3, 8, 23, 24, 28, 9, and 64, which are greater than or equal to 65.
- page 2:** Contains values 4 through 56. Red boxes highlight values 79, 80, 88, 90, 74, 76, and 81, which are greater than or equal to 65.
- page 3:** Contains values 67 through 99. Red boxes highlight values 12, 41, 62, 7, 25, 59, and 19, which are greater than or equal to 65.
- page 4:** Contains values 61 through 94. Red boxes highlight values 78, 65, 70, and 85, which are greater than or equal to 65.

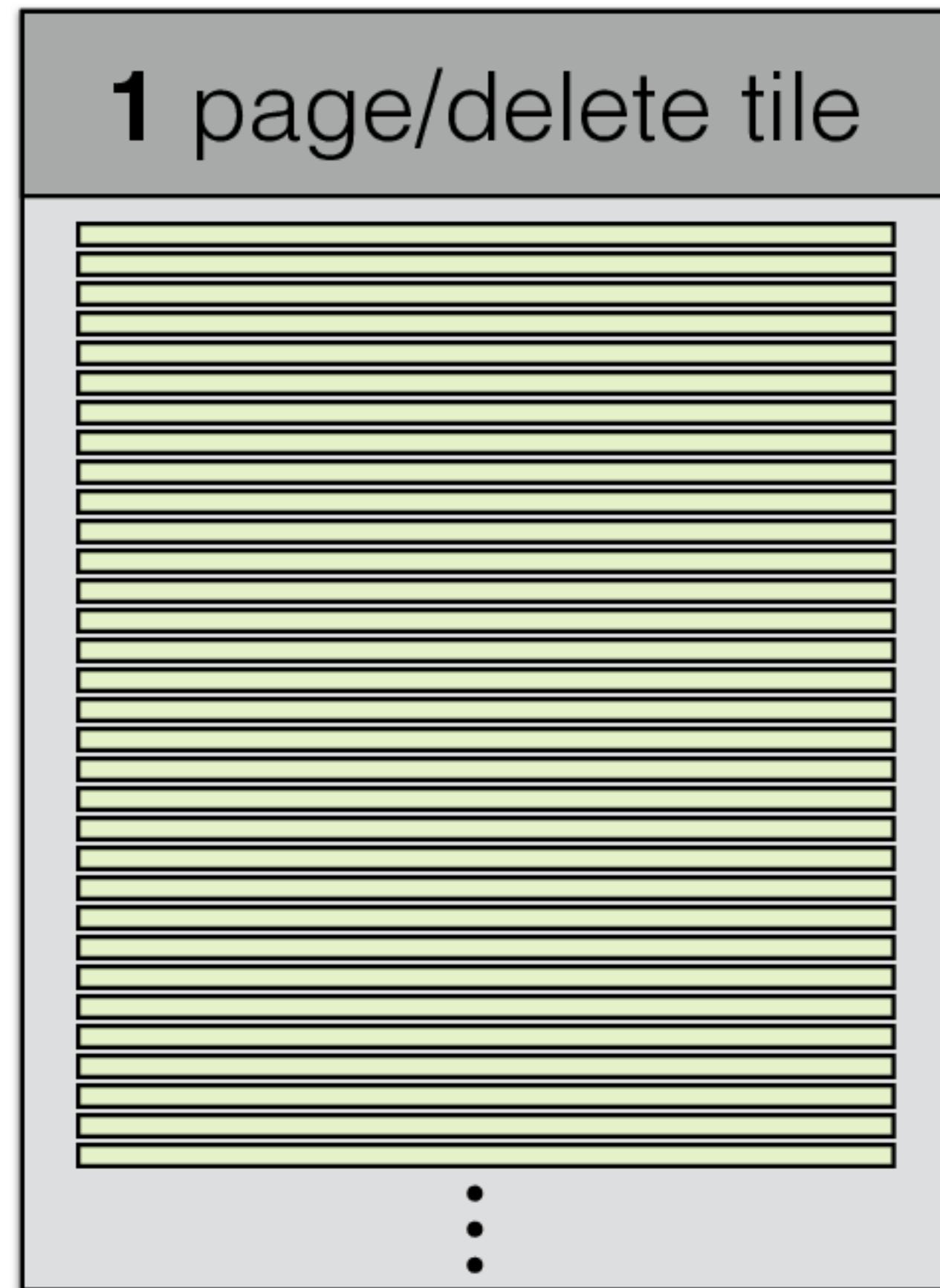
sorted on S

drop
page

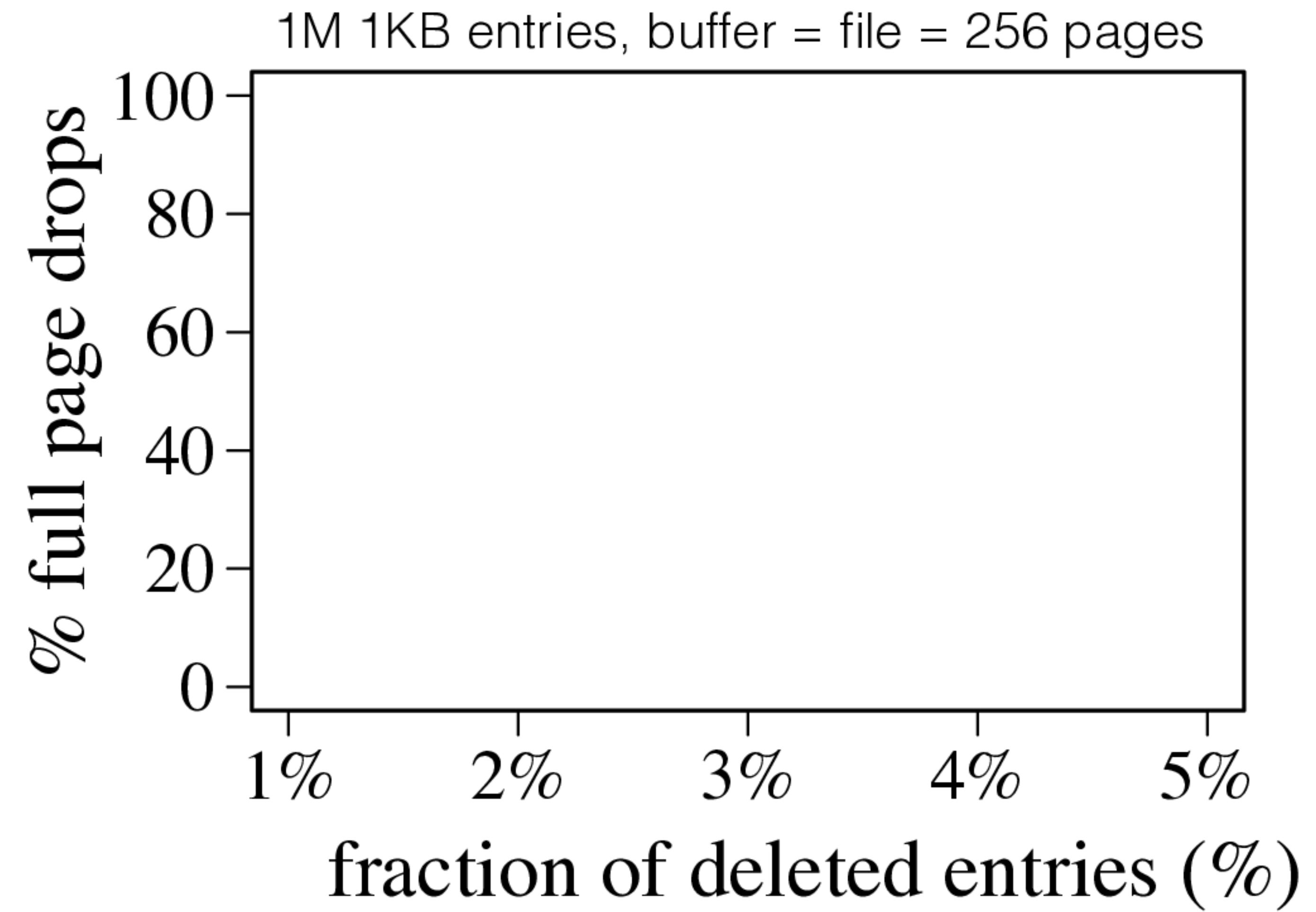
drop
page

1 I/O

Key Weaving storage layout



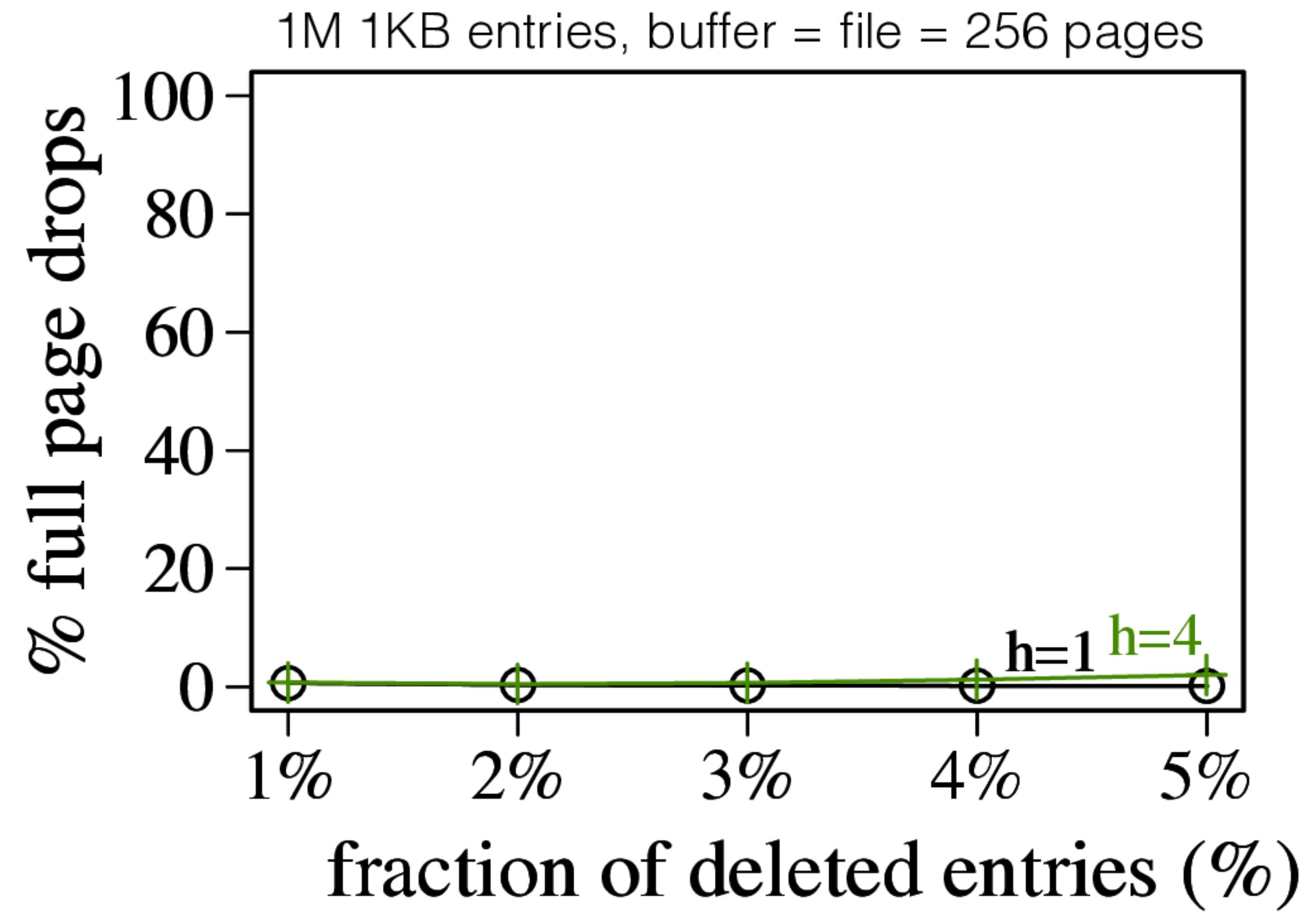
Internals of an SST file in [KiWi](#)



Key Weaving storage layout



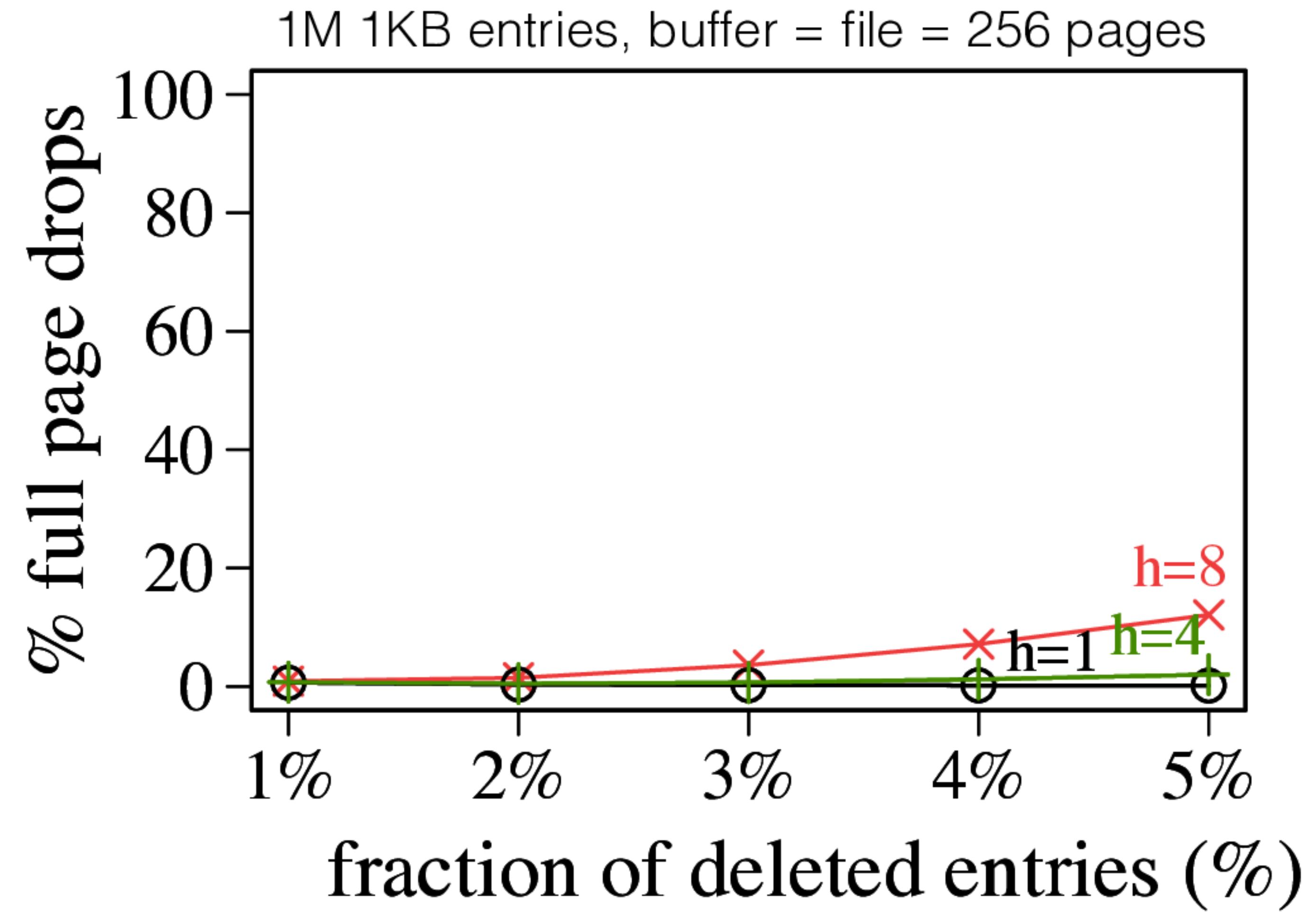
Internals of an SST file in [KiWi](#)



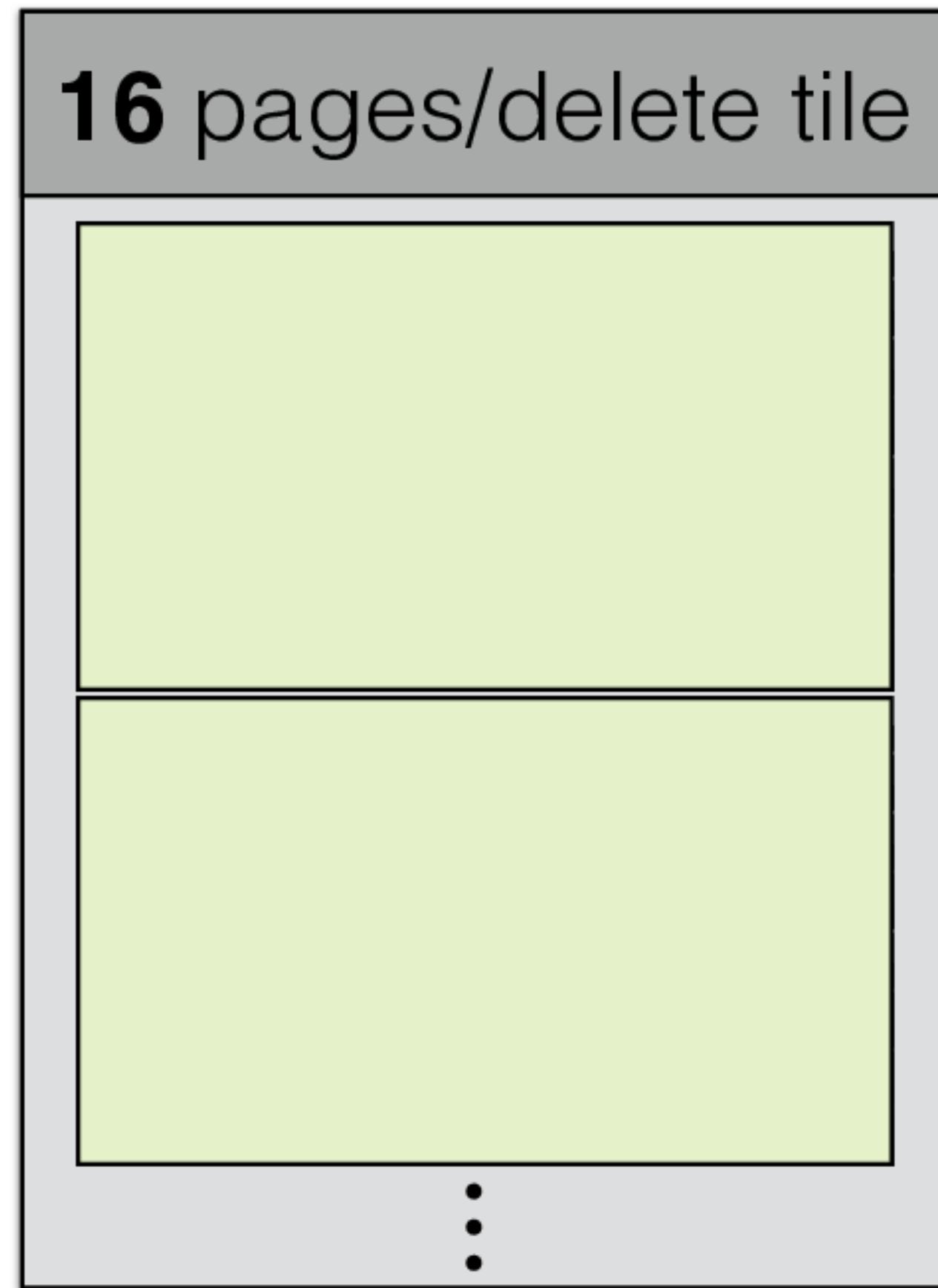
Key Weaving storage layout



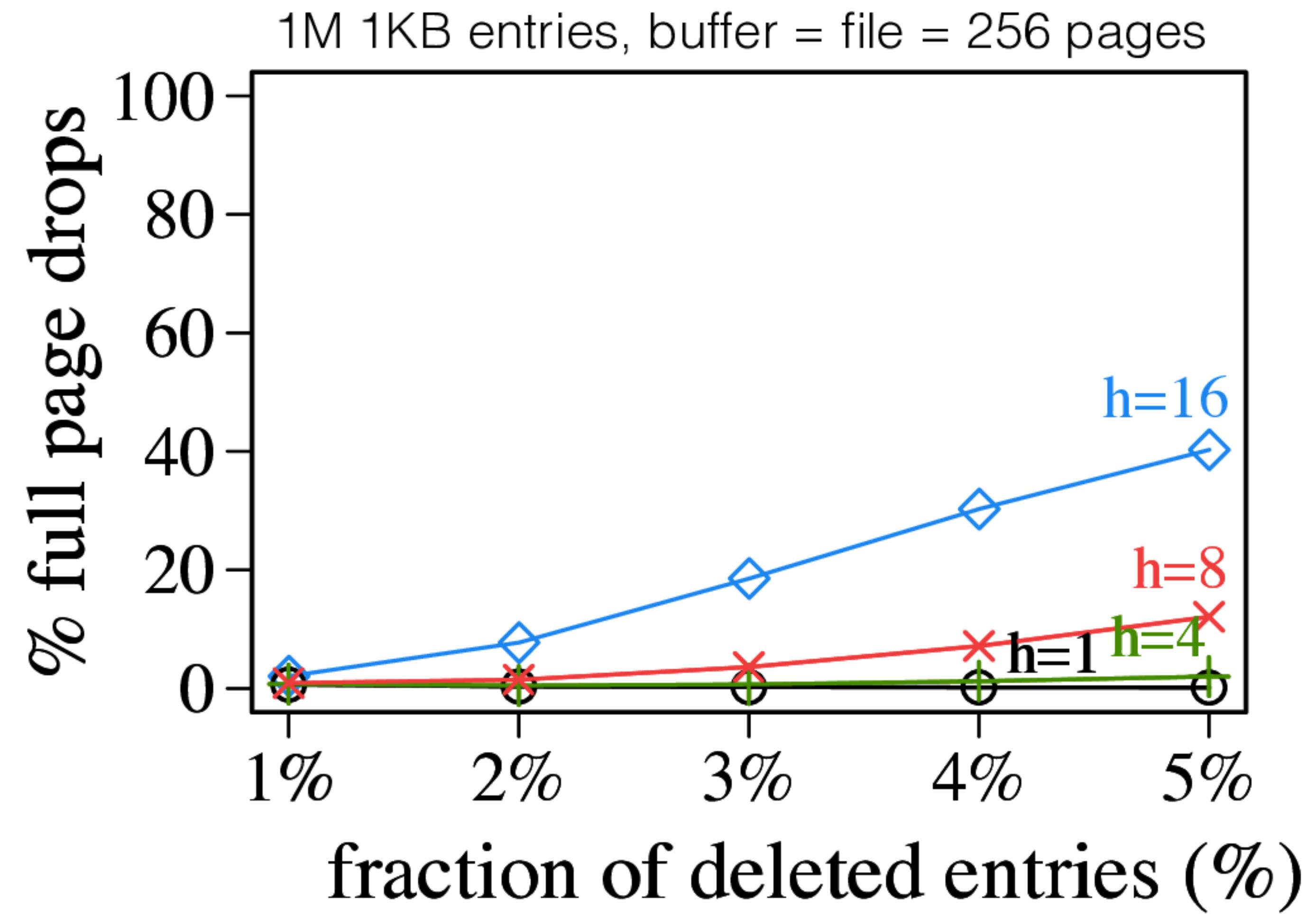
Internals of an SST file in [KiWi](#)



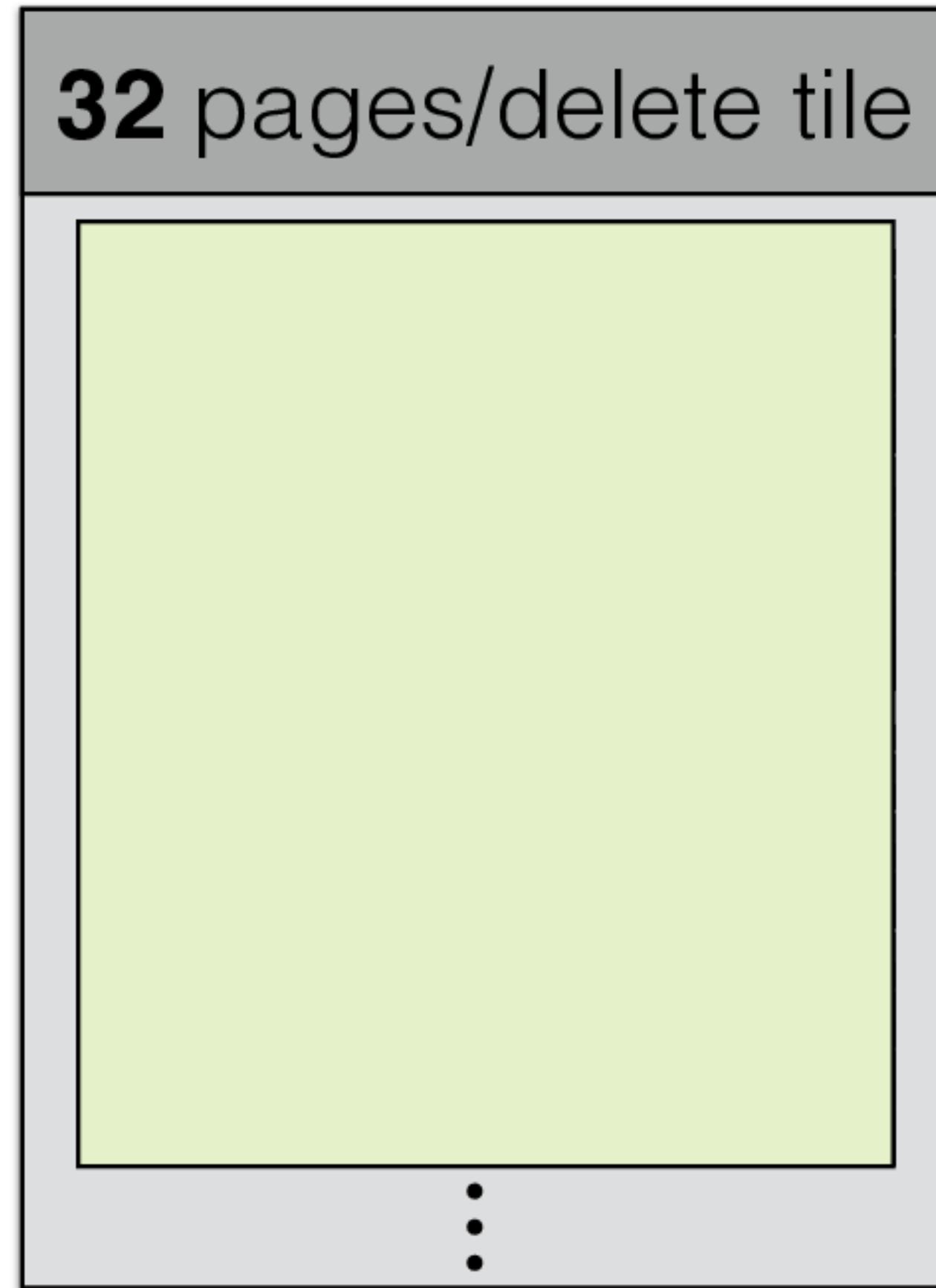
Key Weaving storage layout



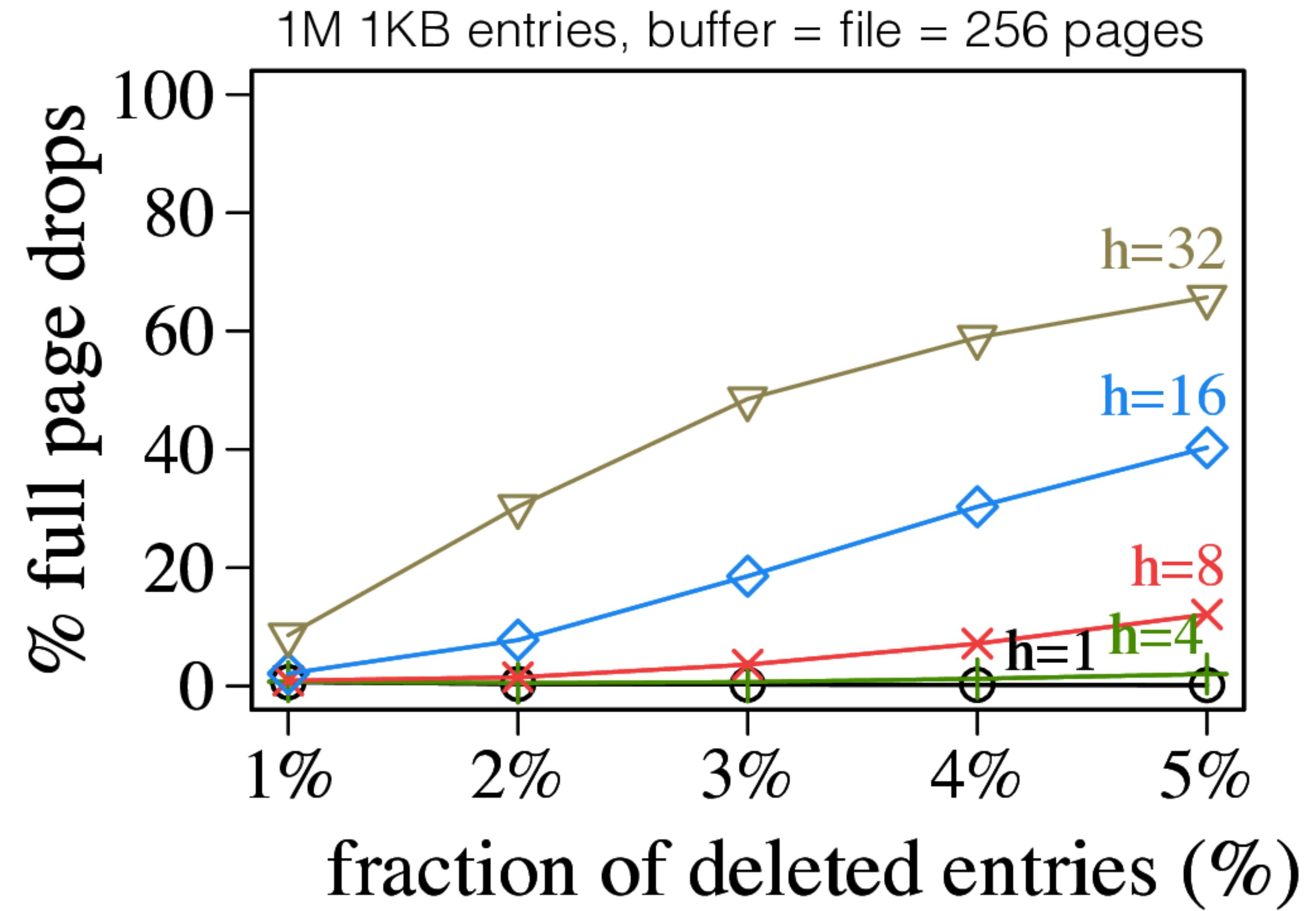
Internals of an SST file in **KiWi**



Key Weaving storage layout

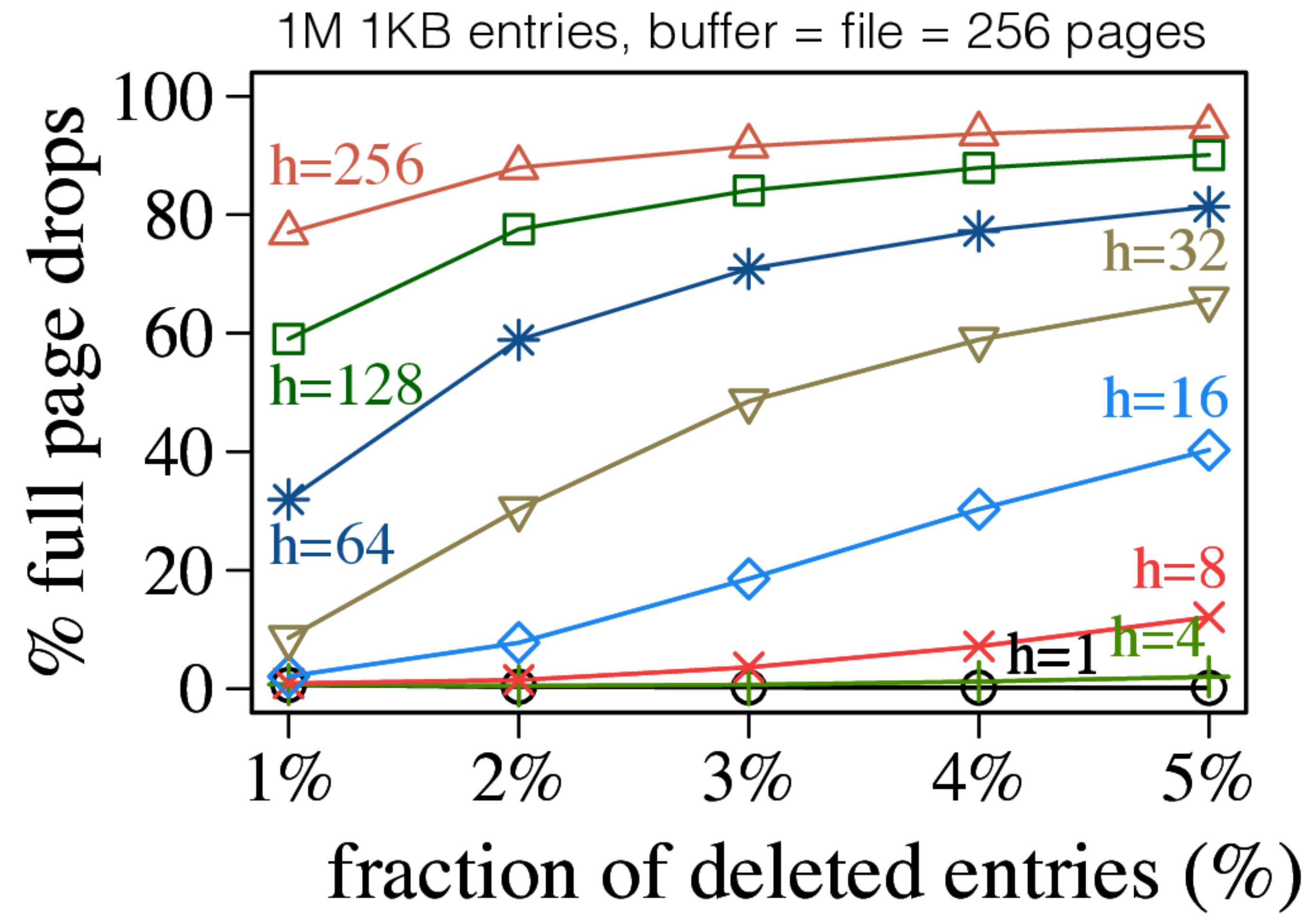


Internals of an SST file in **KiWi**



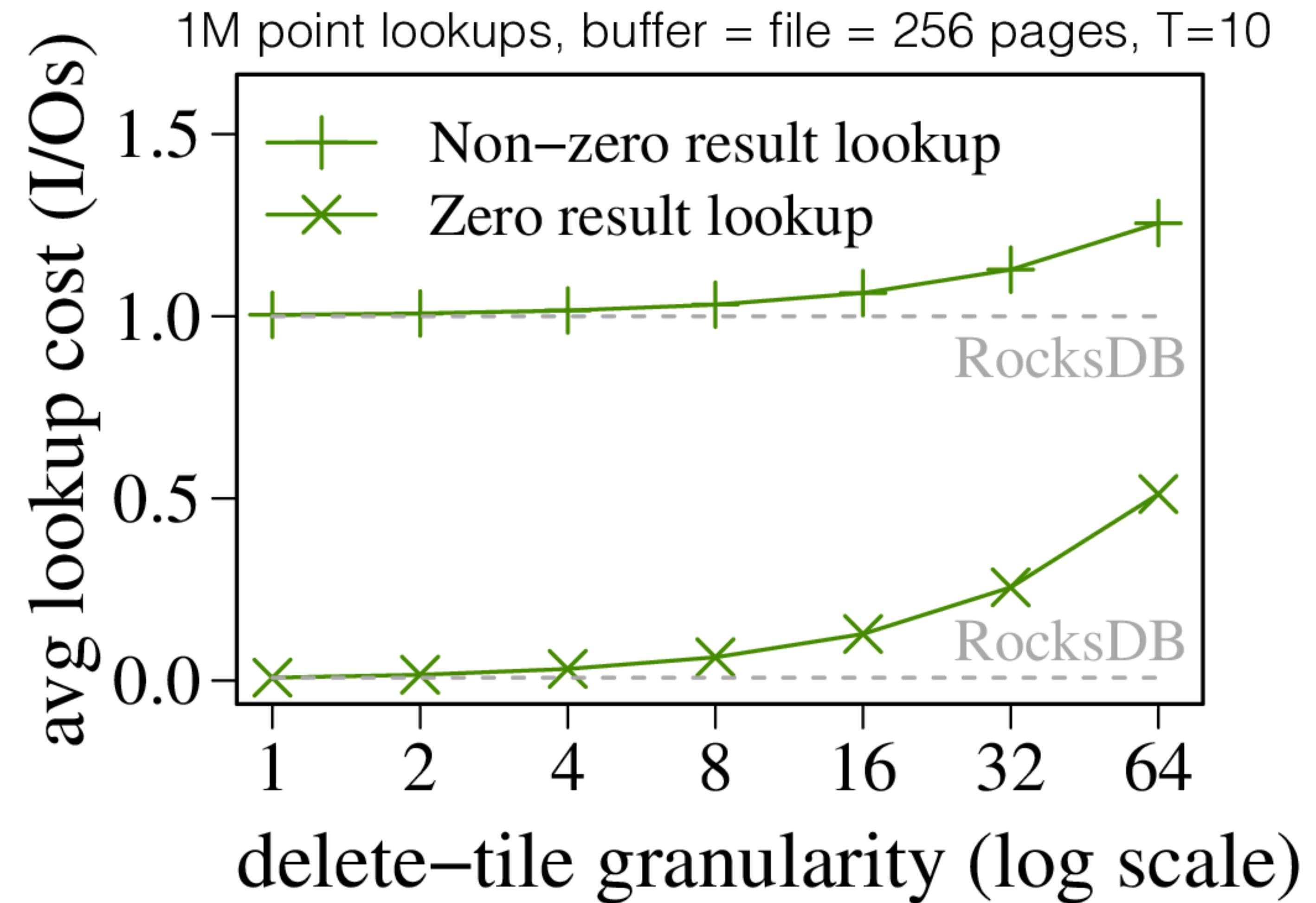
Key Weaving storage layout

- reduced latency spikes ✓
- full page drops reduces superfluous I/Os ✓



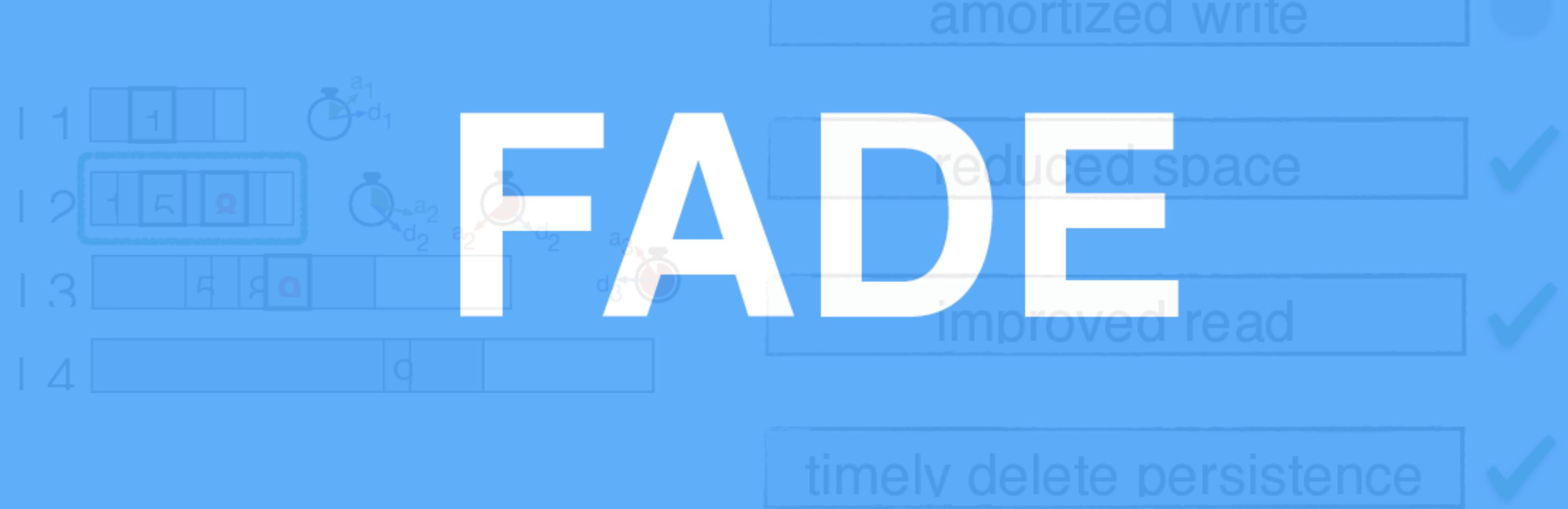
Key Weaving storage layout

- higher lookup cost ↑
- reduced latency spikes ✓
- full page drops reduces superfluous I/Os ✓



the solution

FAst DElete



higher lookup cost

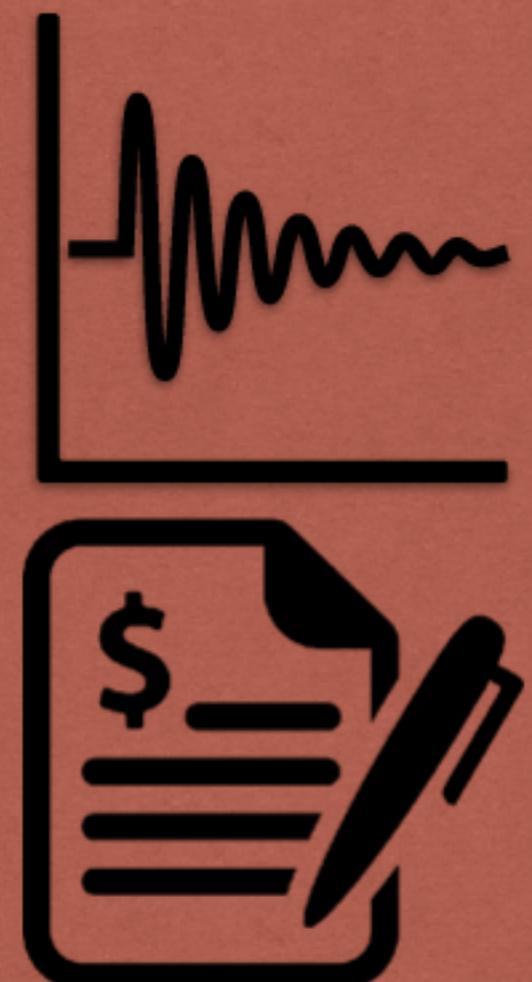
KiWi

full page drops reduces
superfluous I/Os

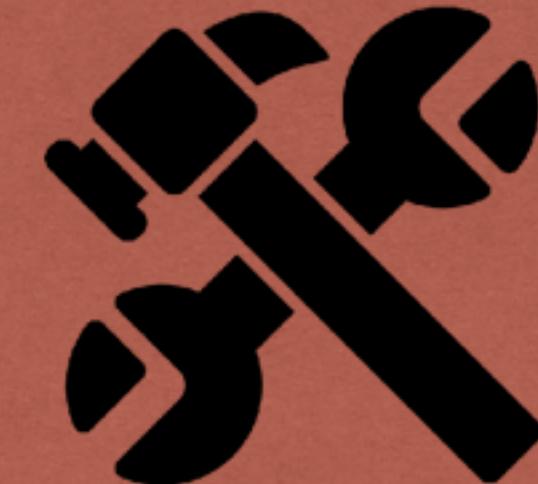
the solution

FADE

KiWi



Lethe



delete tile size



lookup
cost

secondary range
delete cost



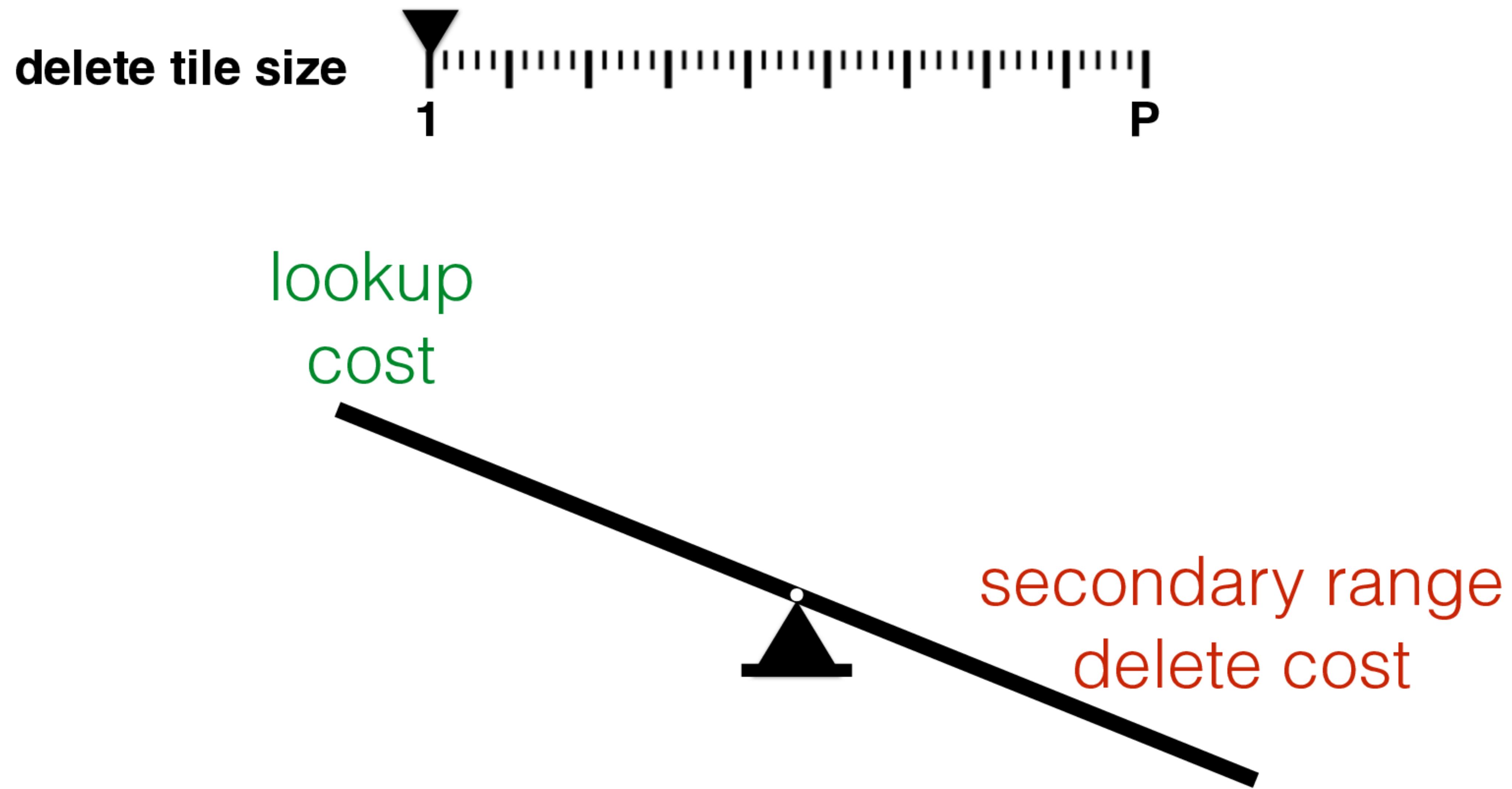
delete tile size



lookup
cost

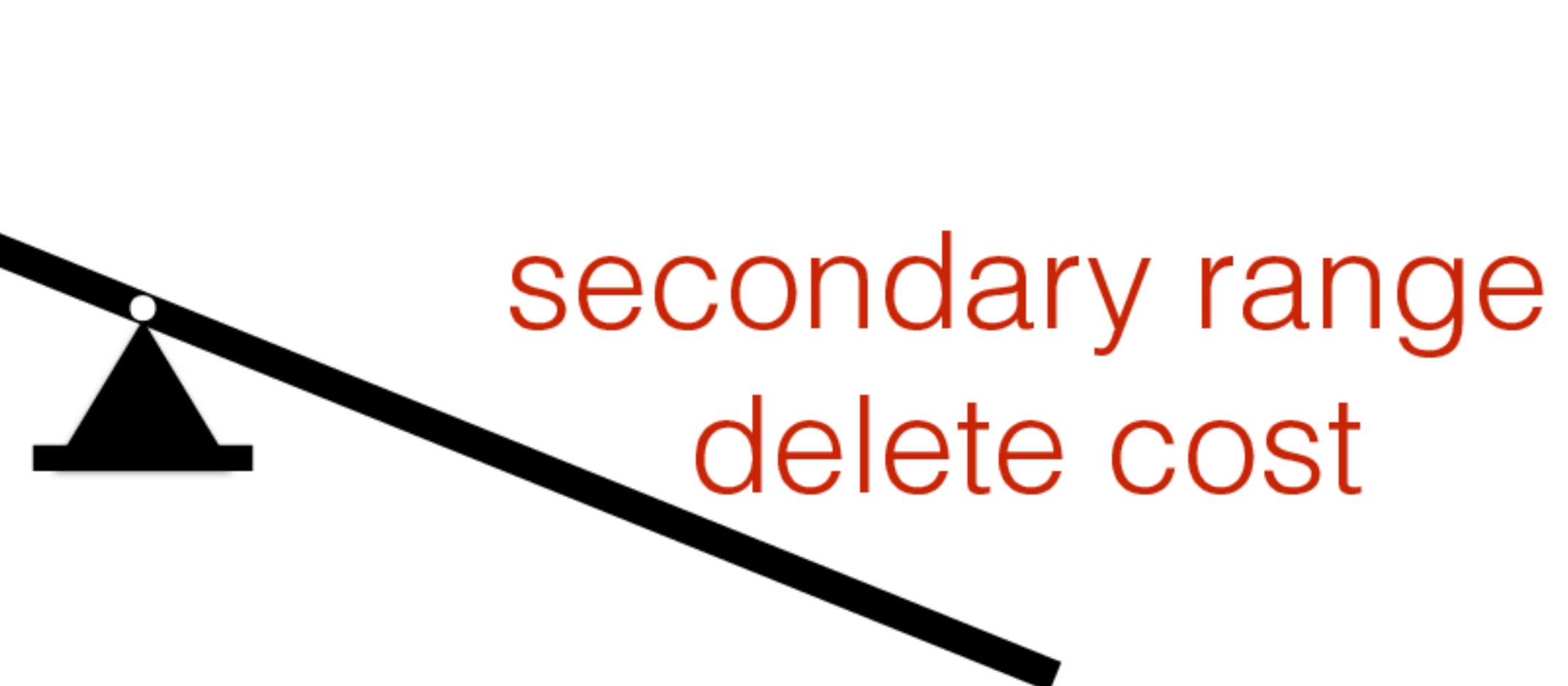
secondary range
delete cost







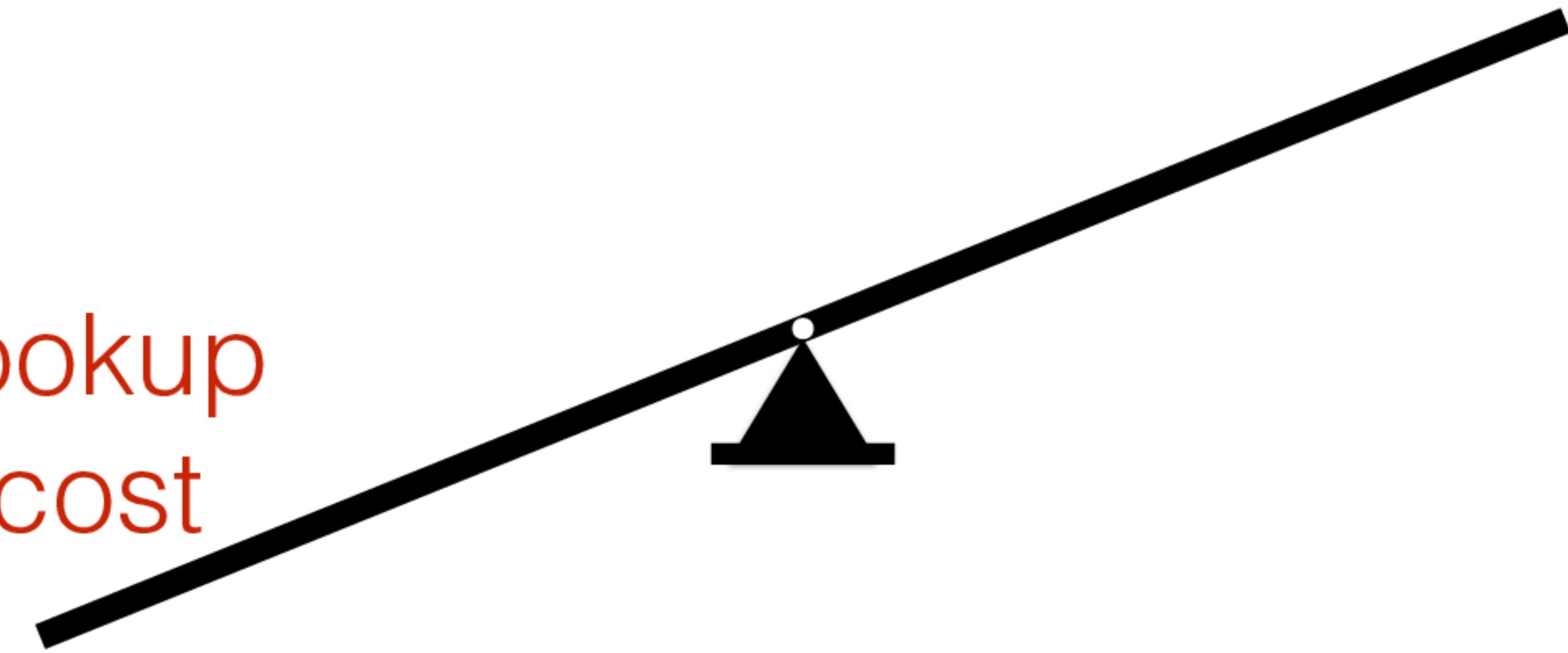
lookup
cost





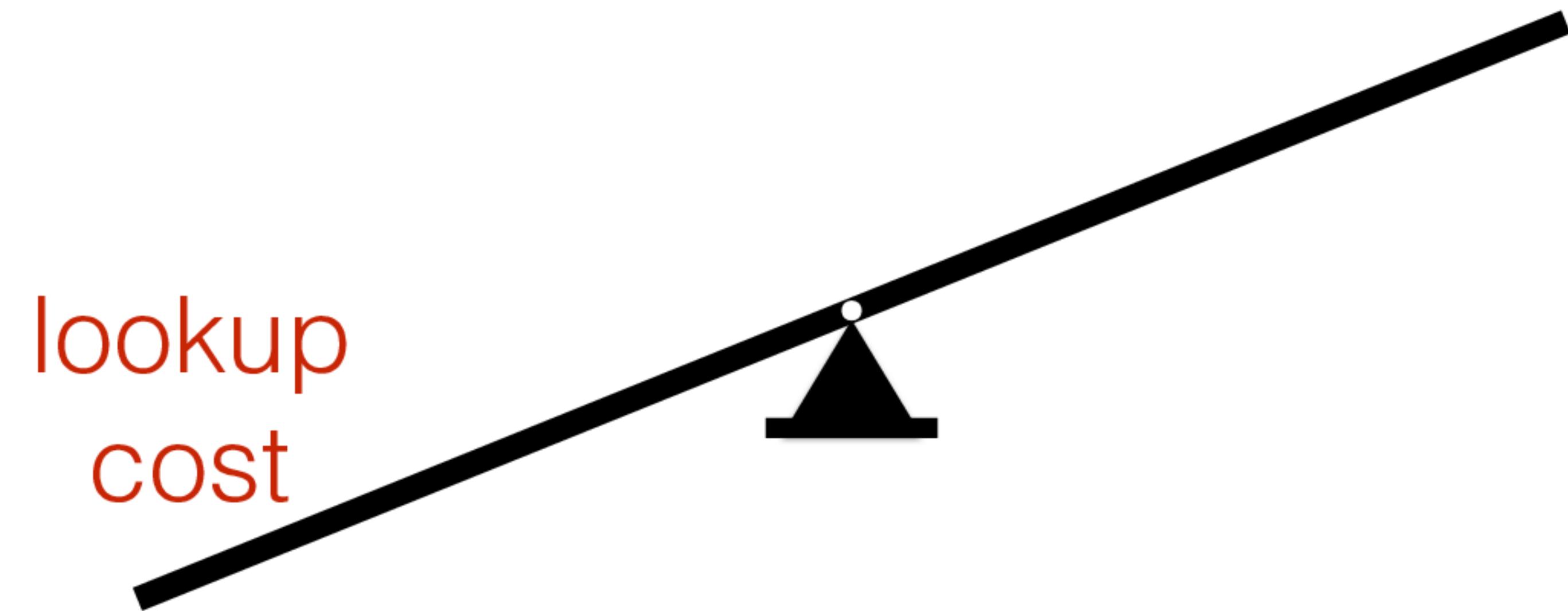
secondary range
delete cost

lookup
cost





secondary range
delete cost



delete tile size



lookup
cost

secondary range
delete cost

delete tile size



lookup
cost

secondary range
delete cost

delete tile size



lookup
cost

secondary range
delete cost



delete tile size



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/