CS 561: ***Data Systems Architecture***

Class 5

Log-Structured Merge (LSM) Trees

Dr. Subhadeep Sarkar

BOSTON UNIVERSITY

DiSC lab

https://bu-disc.github.io/CS561/

Updates: Logistics

First **technical question** is due on **02/07**.

First **review** is due on **02/14**.

Deadline for **Project 0** extended till **02/05**.

The first **student presentation** is in two weeks (on **02/14**)!

Select the paper you will work on for your **presentation** (groups of 3-4)

A week before the presentation, discuss the slides with me in OH.

LSM-tree

LSM-tree

NoSQL

relational

time-series

2023

LSM-tree

NoSQL

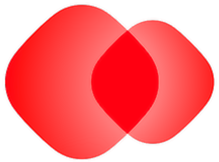

RocksDB


levelDB


SCYLLA


DynamoDB


cassandra


tarantool


Bigtable


APACHE HBASE


riak


accumulo


SQLite

relational


influxdb


QuasarDB

time-series

2023

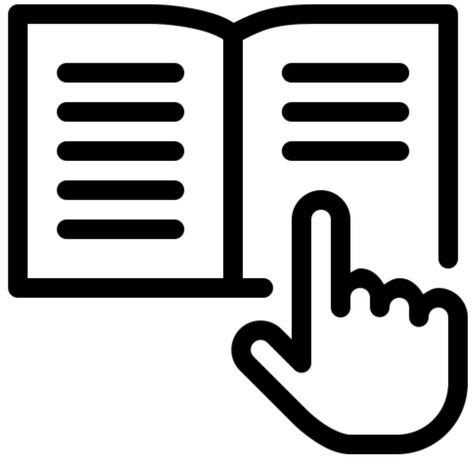
BOSTON UNIVERSITY

Why **LSM** ?

fast ingestion

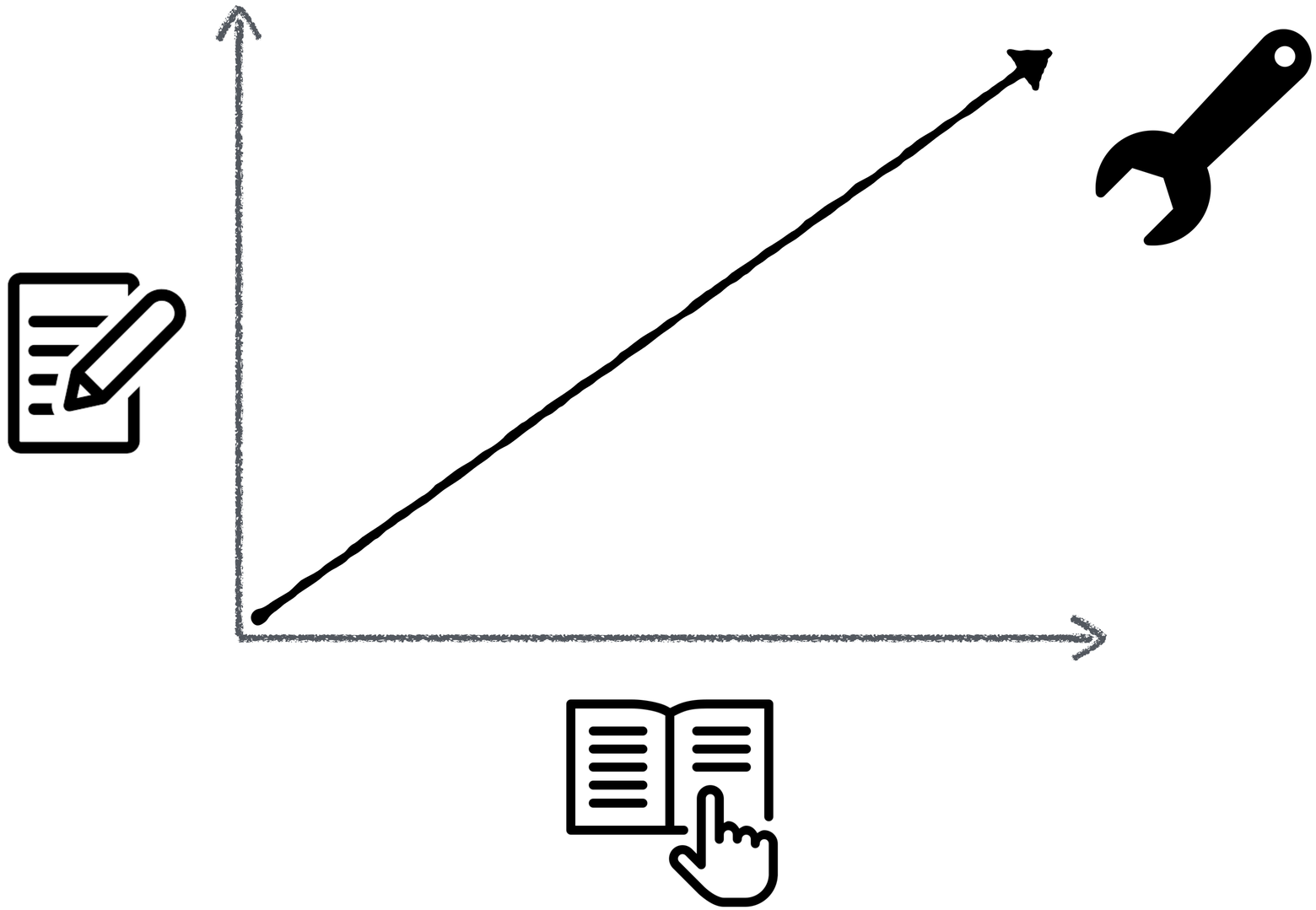
Why **LSM** ?

fast ingestion



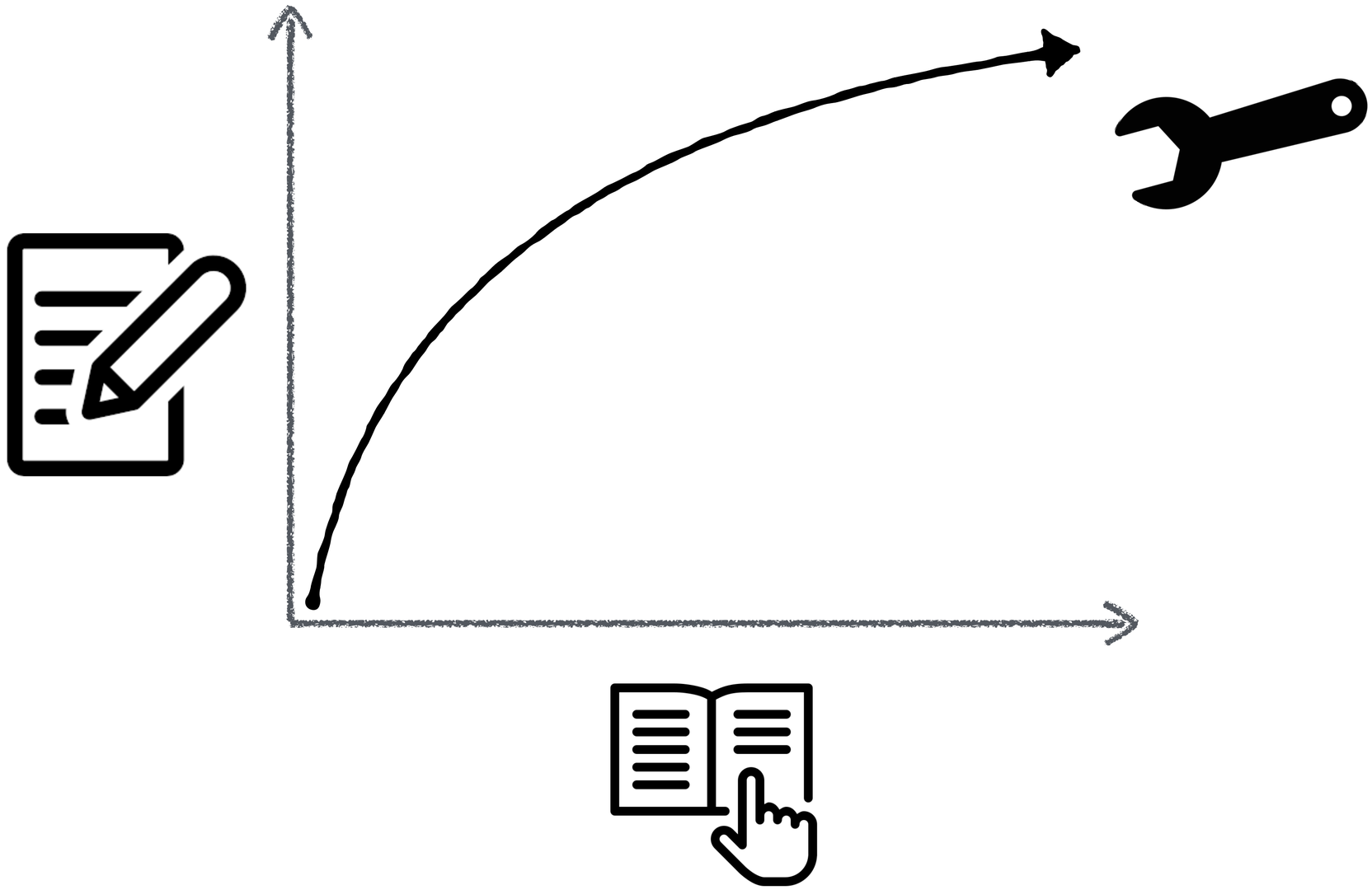
competitive reads

Why **LSM** ?



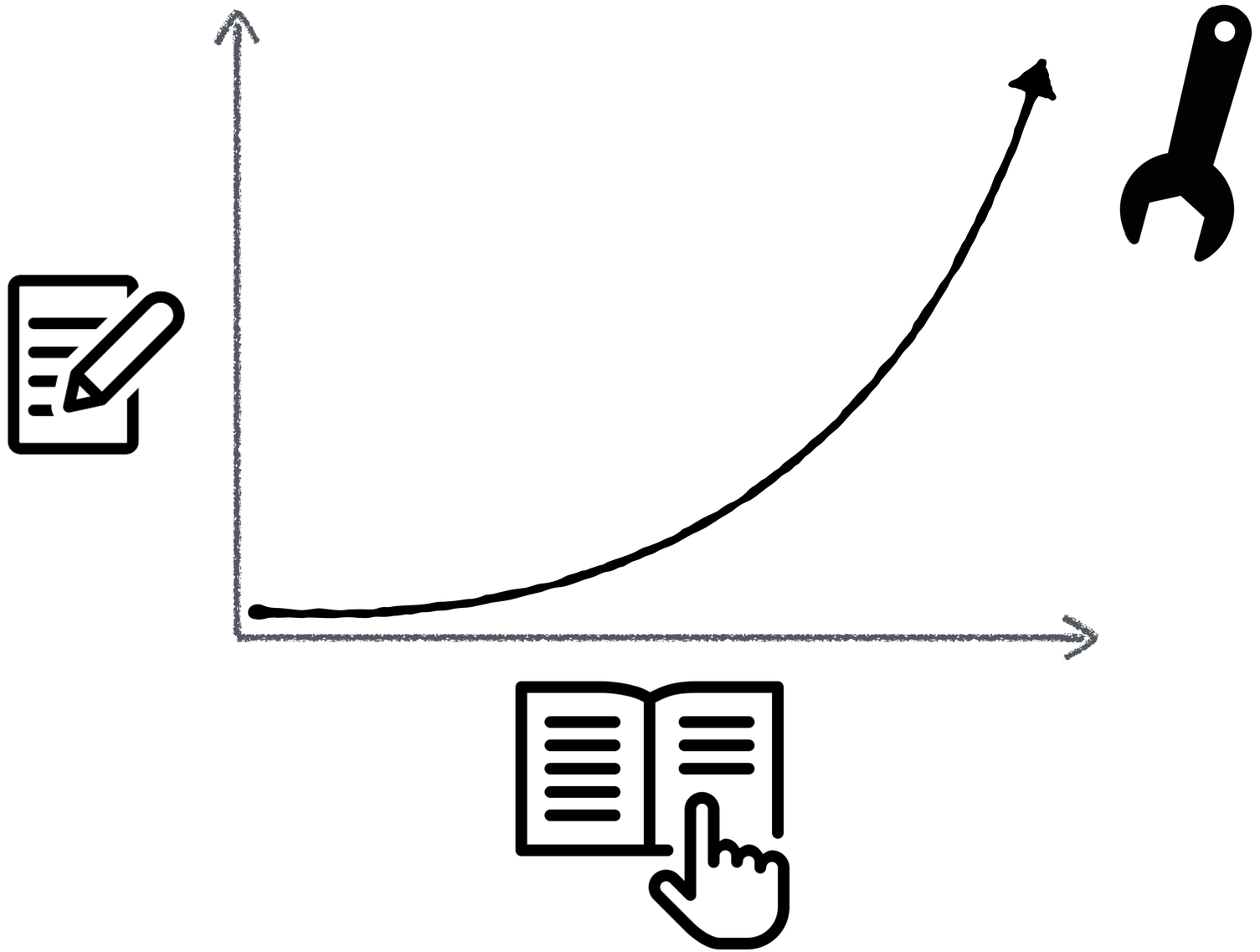
BOSTON UNIVERSITY

Why **LSM** ?

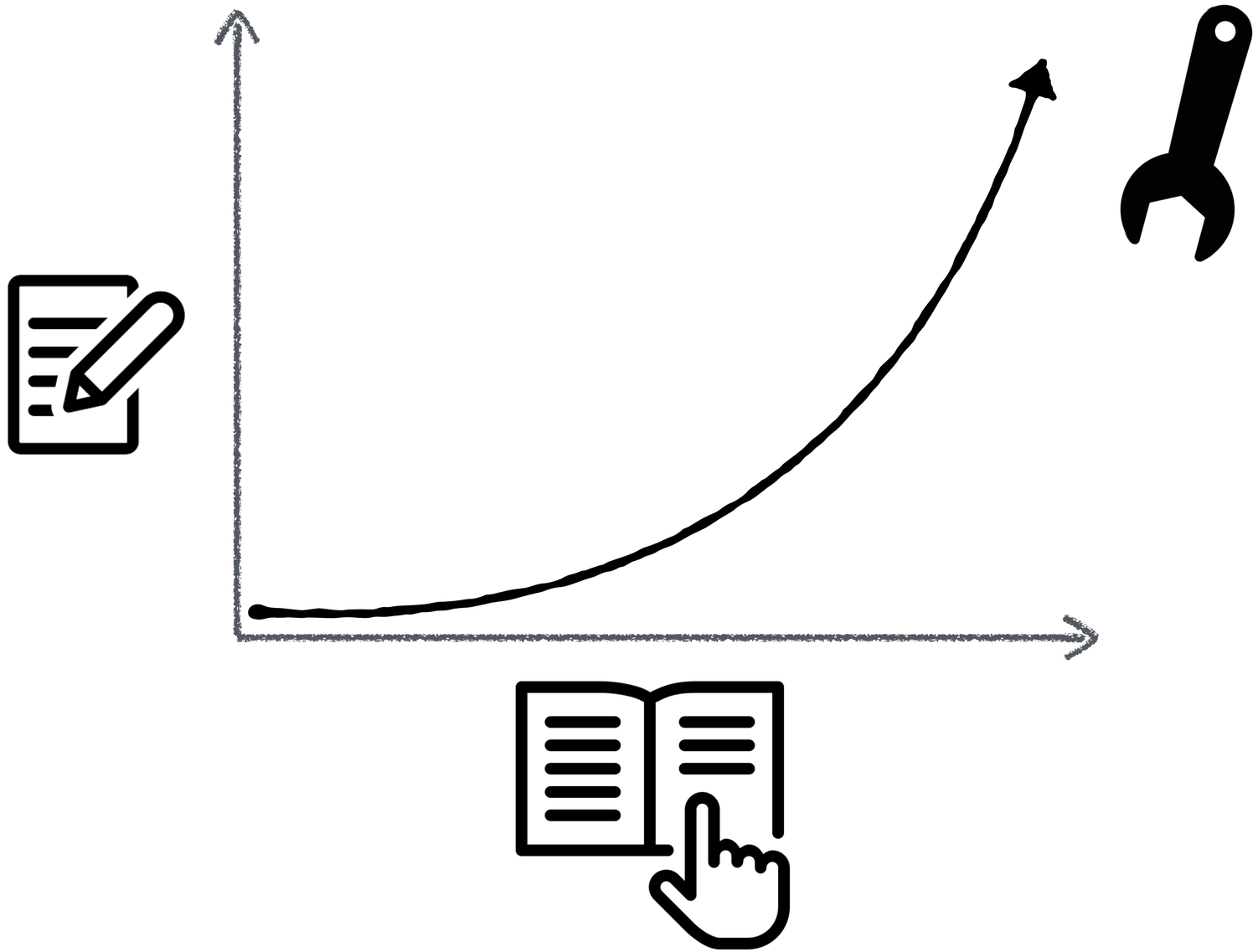


BOSTON UNIVERSITY

Why **LSM** ?



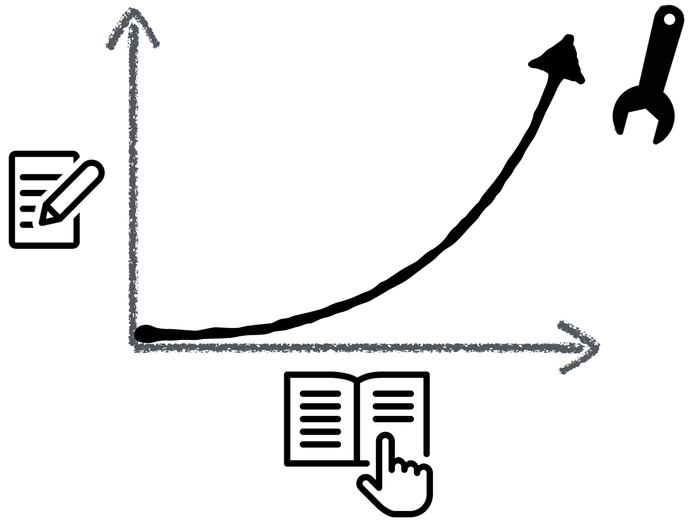
Why **LSM** ?



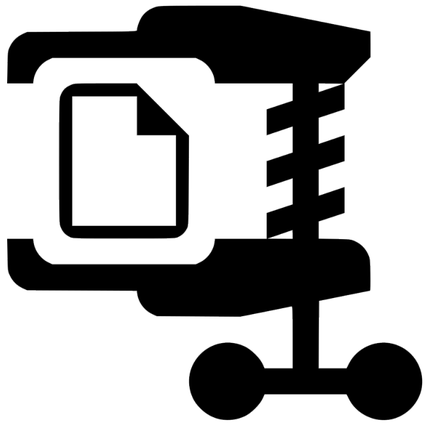
BOSTON UNIVERSITY

Why **LSM** ?

fast writes

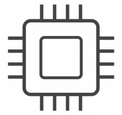


tunable read-write performance

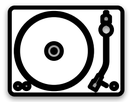


good space utilization

LSM **Basics**



buffer



level 1

level 2

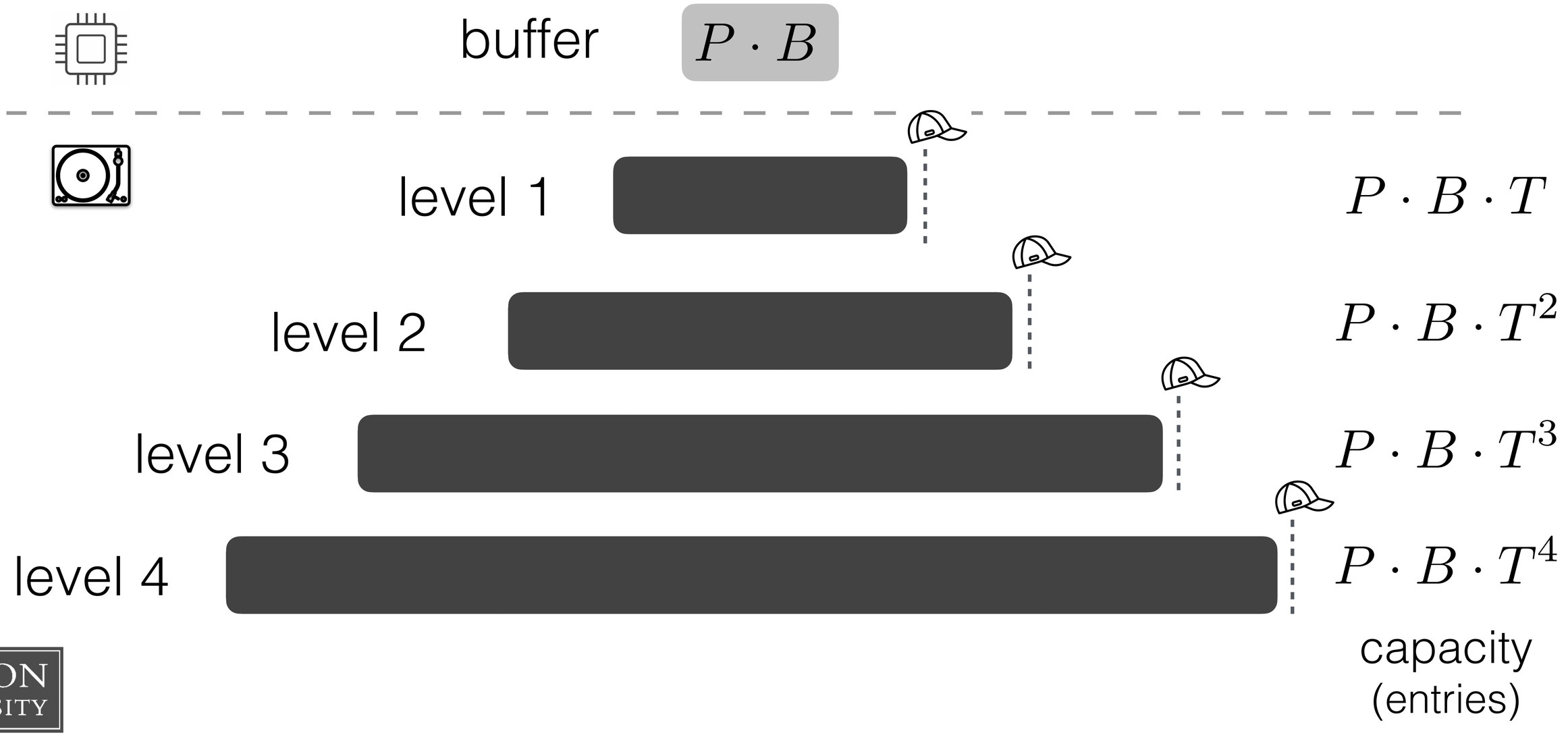
level 3

level 4

size ratio: T

P: pages in buffer
B: entries/page
L: #levels
T: size ratio

LSM **Basics**



LSM Operating **Principles**

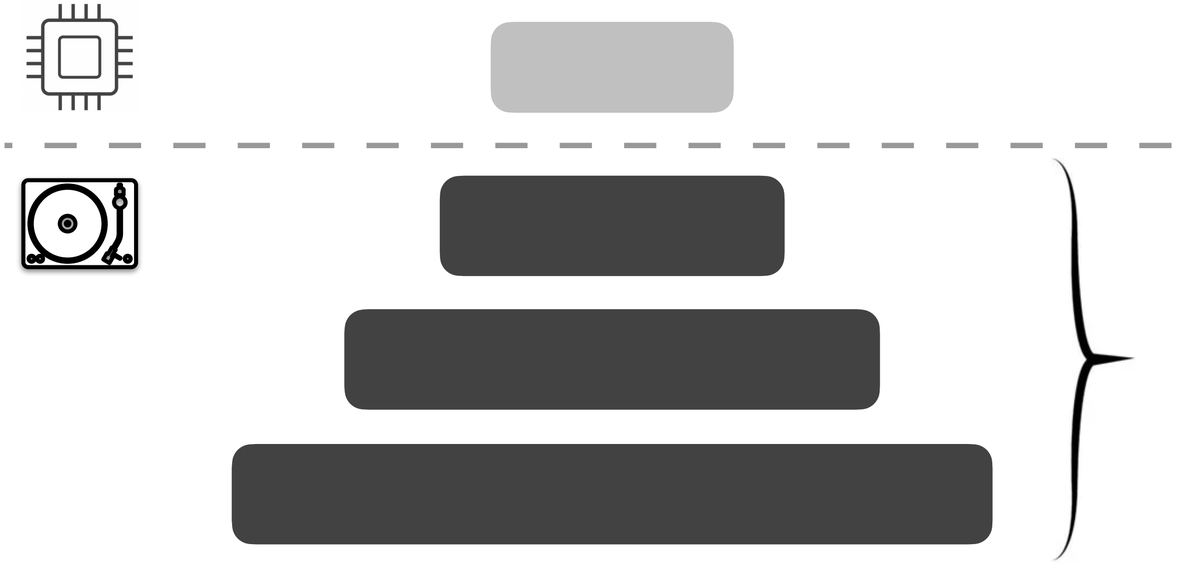
Buffering ingestion

Immutable files on storage

Out-of-place updates & deletes

Periodic data layout reorganization

L: #levels
T: size ratio



most data
on storage

if $T = 10$ & $L = 4$

99.9% on storage

How does the storage layer affect ingestion?

Data **Layout**

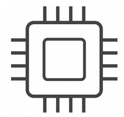
Classical LSM design: **leveling**

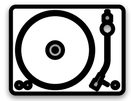
[eager merging]

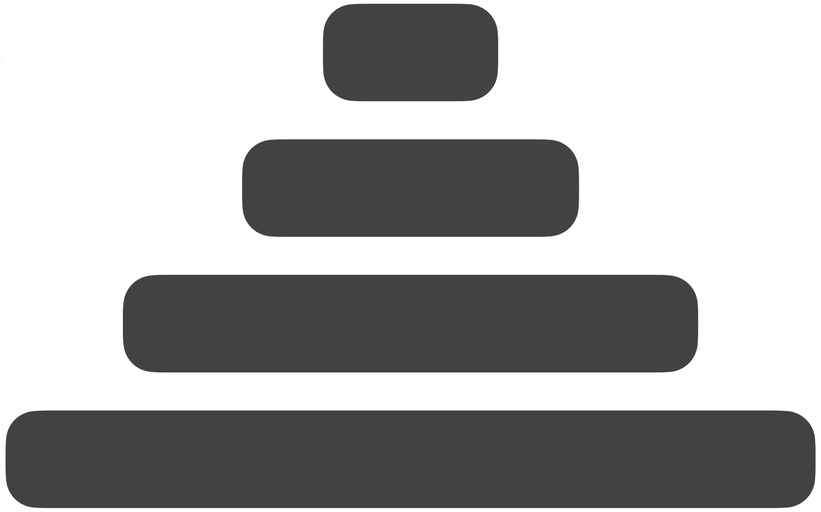
BOSTON UNIVERSITY

Data **Layout**

leveling [eager]







1 run per level

- good read performance
- good space amplification
- high write amplification

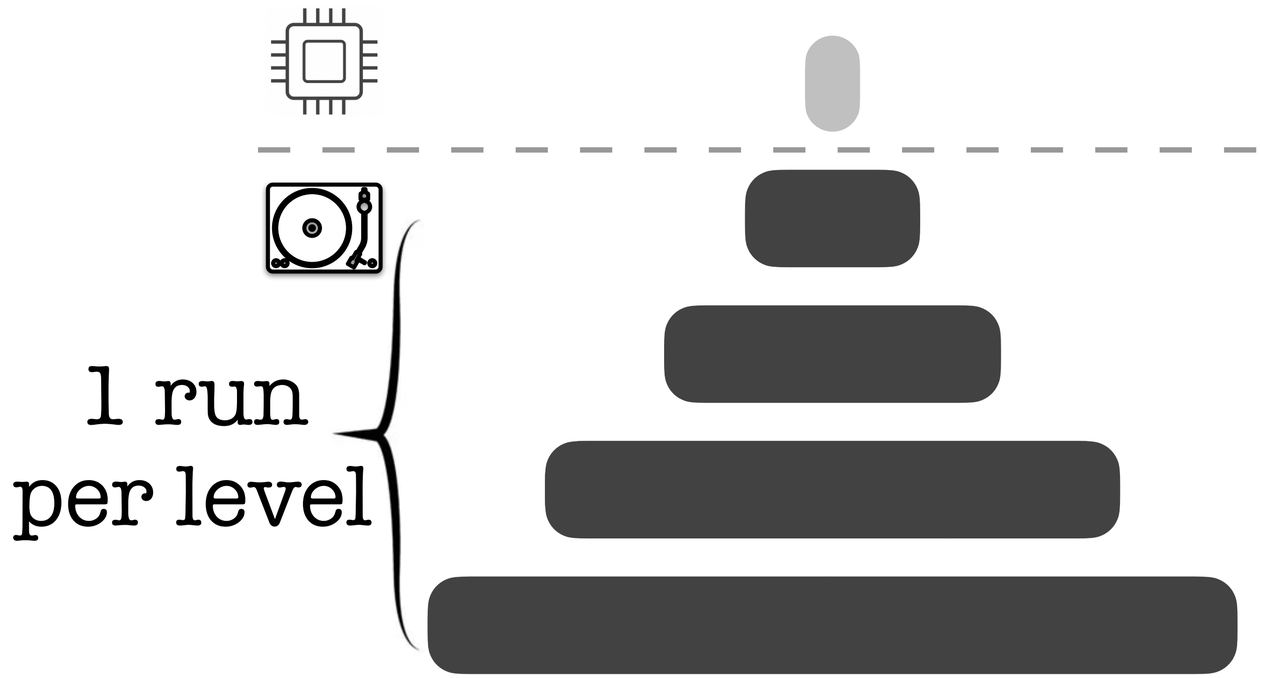
Seems like heaven!

BOSTON UNIVERSITY

Data **Layout**

leveling [eager]

tiering [lazy]



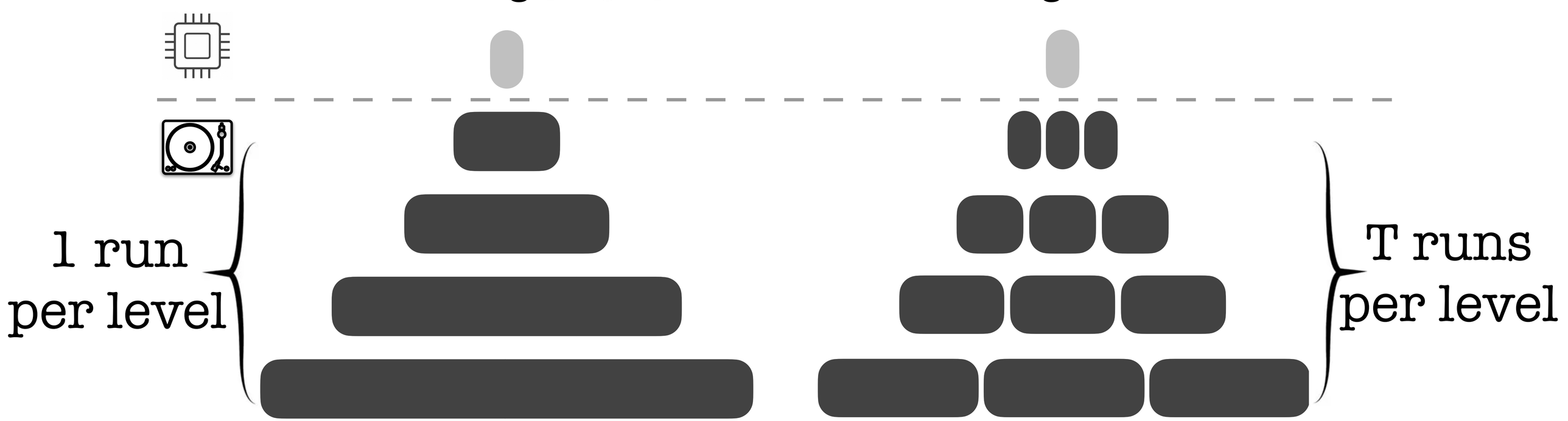
- good read performance
- good space amplification
- high write amplification

BOSTON UNIVERSITY

Data **Layout**

leveling [eager]

tiering [lazy]



- good read performance
- good space amplification
- high write amplification

- good ingestion performance

Limitations?

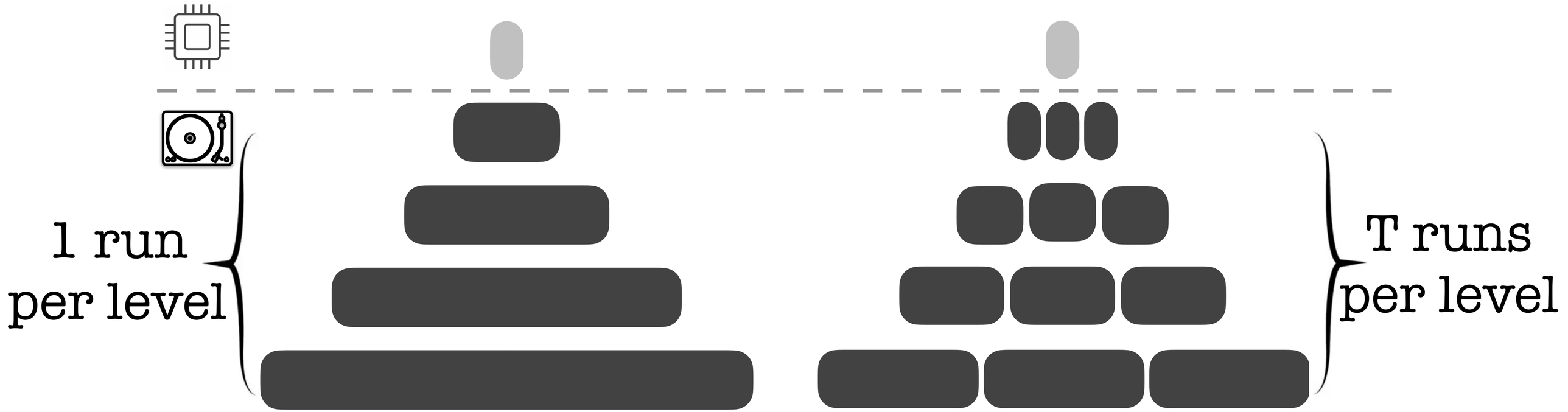
BOSTON UNIVERSITY

P: pages in buffer
B: entries/page
L: #levels
T: size ratio
N: #entries
ϕ : FPR of BF

Data **Layout**

leveling [eager]

tiering [lazy]



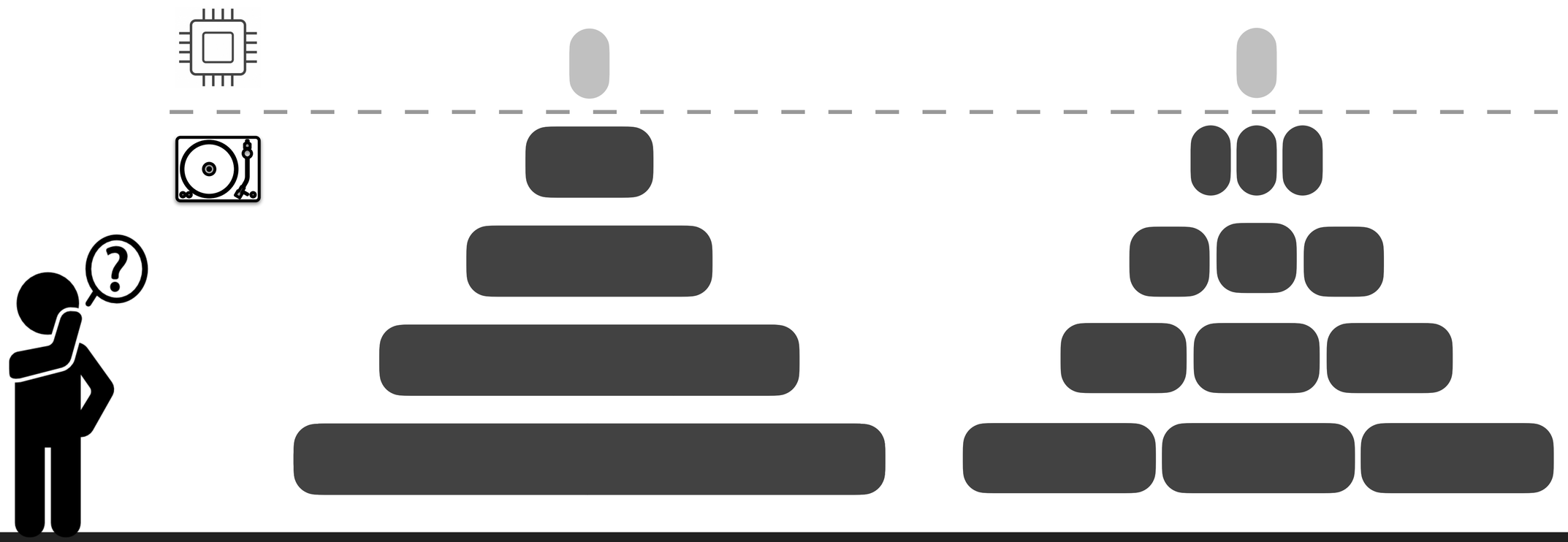
	leveling [eager]	**tiering** [lazy]
Read cost:	$\mathcal{O}(L \cdot \phi)$	$\mathcal{O}(T \cdot L \cdot \phi)$
Write cost:	$\mathcal{O}(T \cdot L/B)$	$\mathcal{O}(L/B)$
SA:	$\mathcal{O}(1/T)$	$\mathcal{O}(T)$

BOSTON UNIVERSITY

Data **Layout**

leveling [eager]

tiering [lazy]

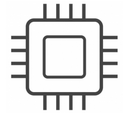


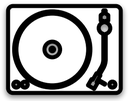
What happens if T becomes too large?

Data **Layout**

leveling [eager]

tiering [lazy]



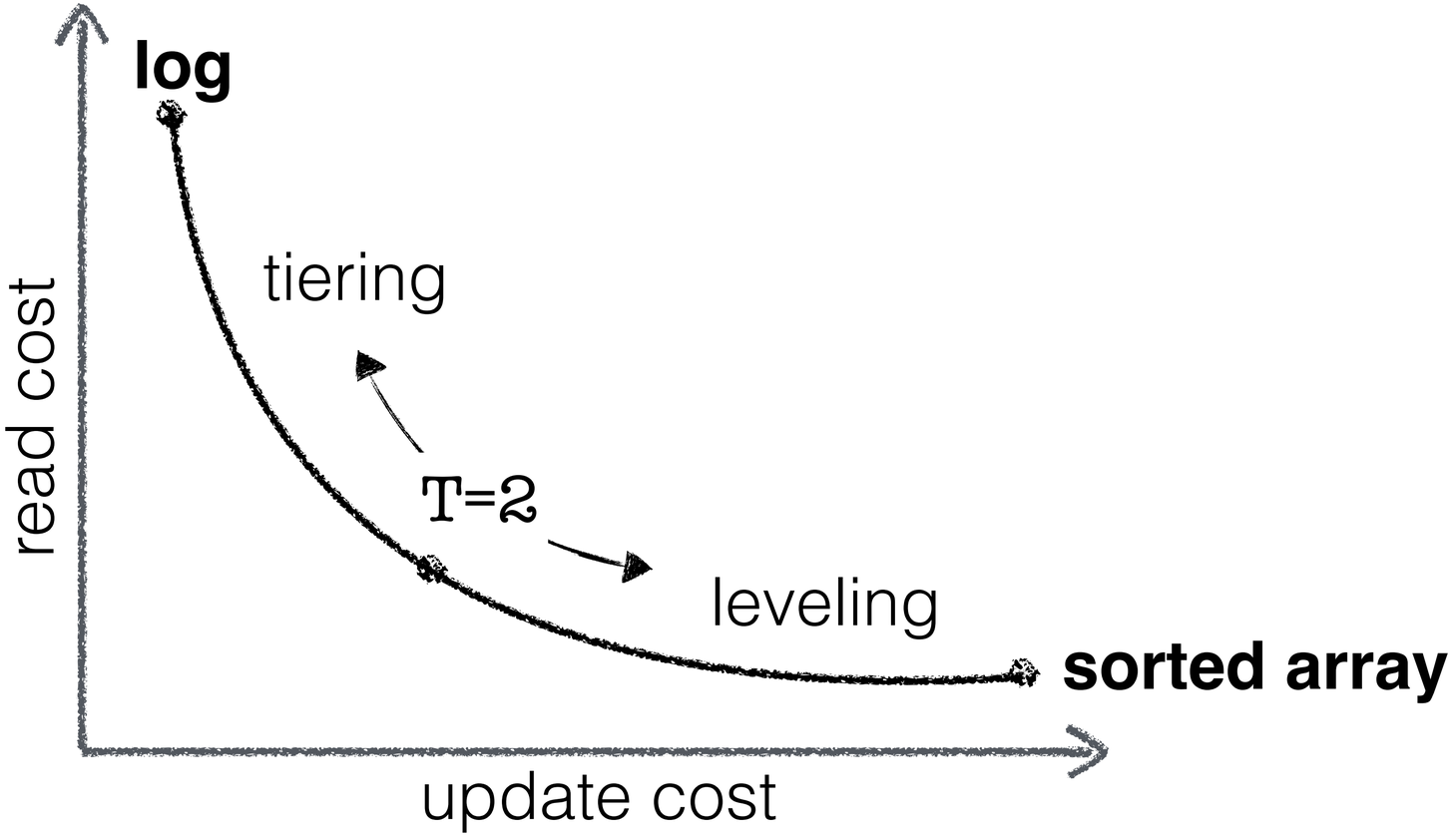


sorted array

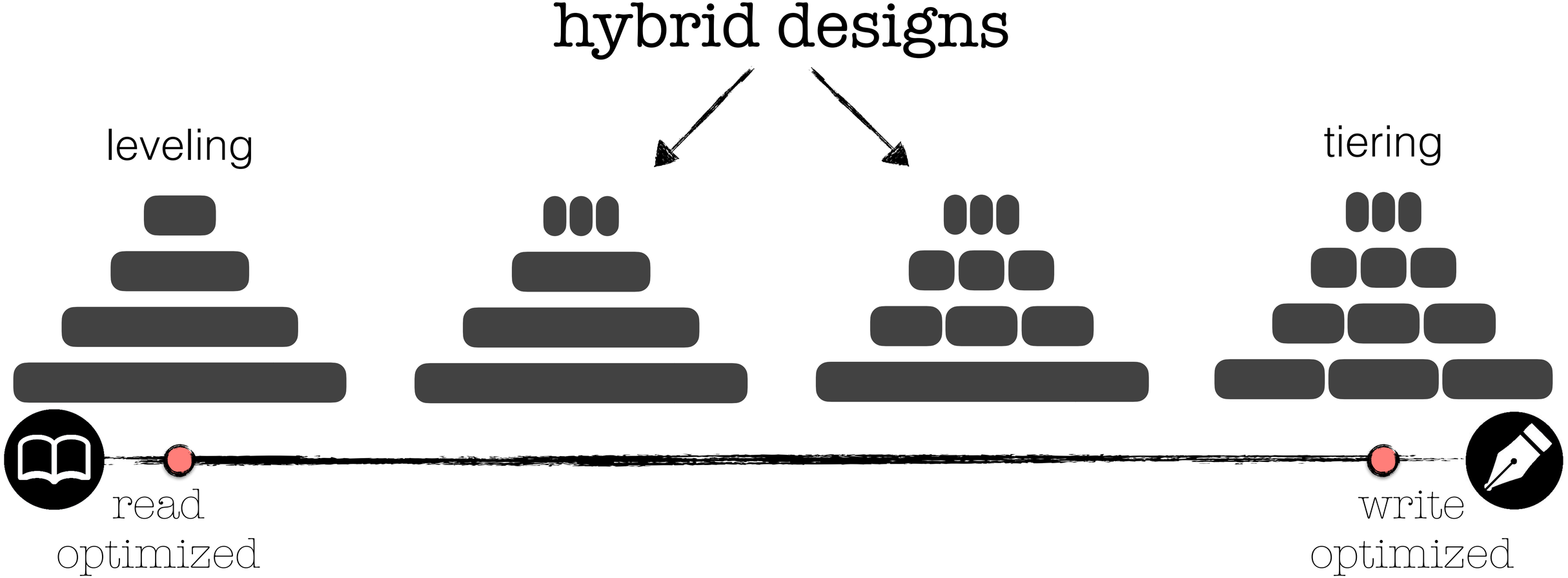
log

T: size ratio

Data **Layout**

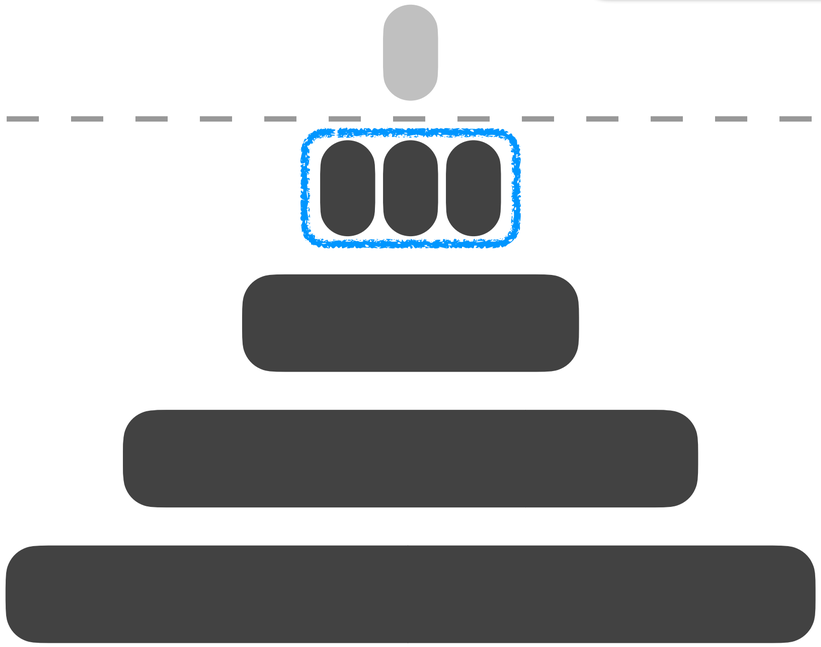


Data **Layout**


hybrid designs
leveling
tiering
read
optimized
write
optimized

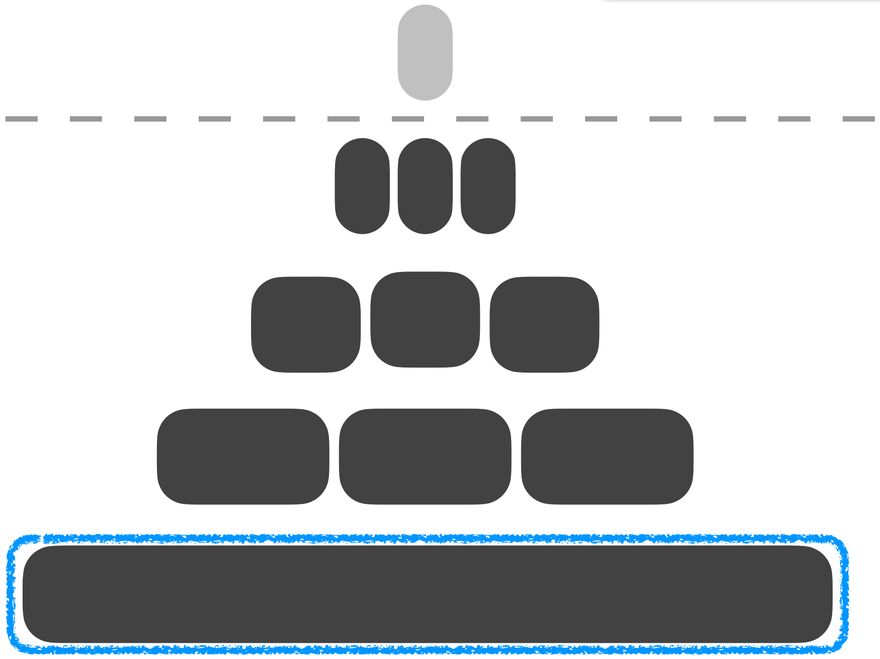
Data **Layout**

1-leveling RocksDB20



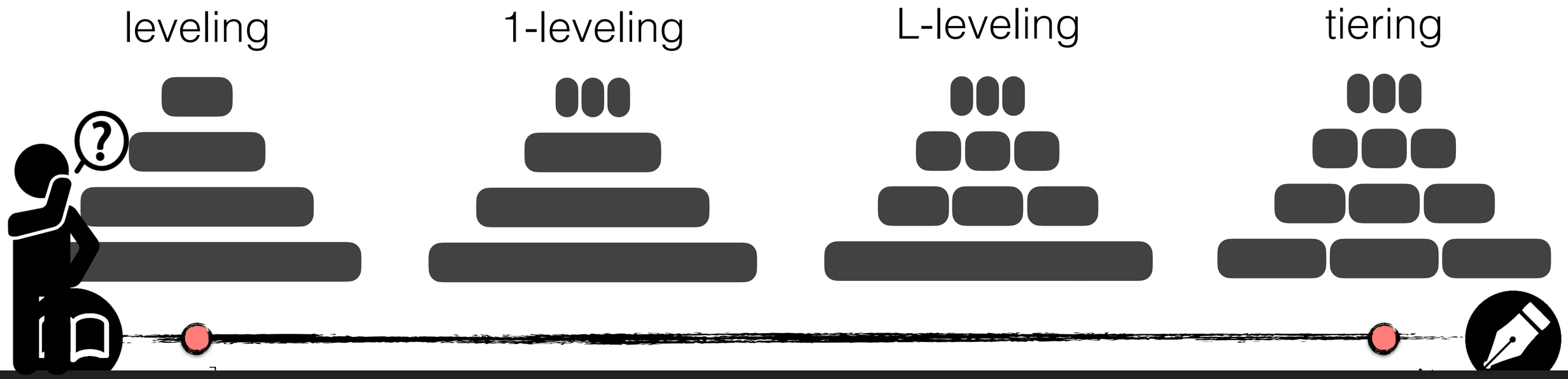
- fewer write stalls
- increased block cache hits

L-leveling DayanSIGMOD18

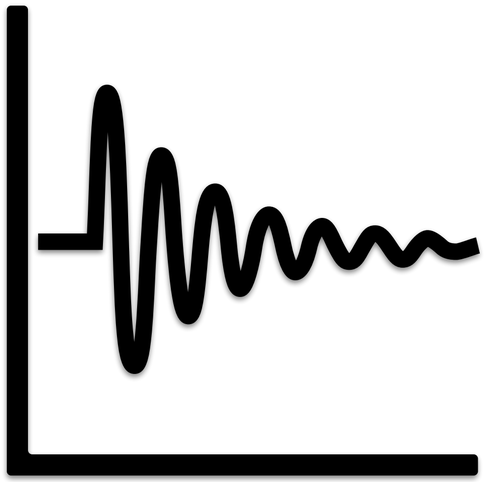


- low write amplification
- better read performance

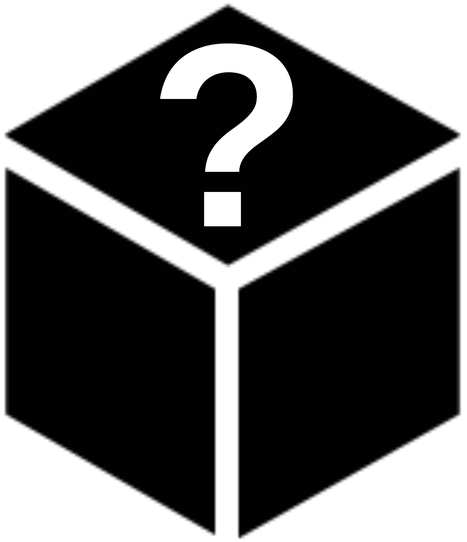
Data **Layout**



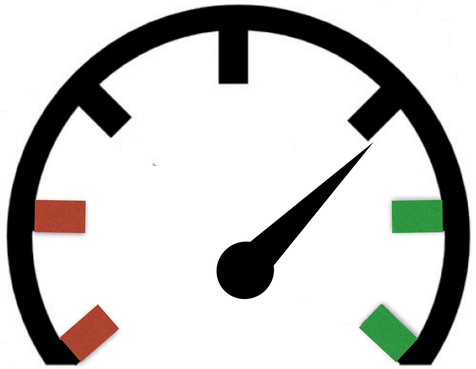
So, how do we reason about the data layout?



workload

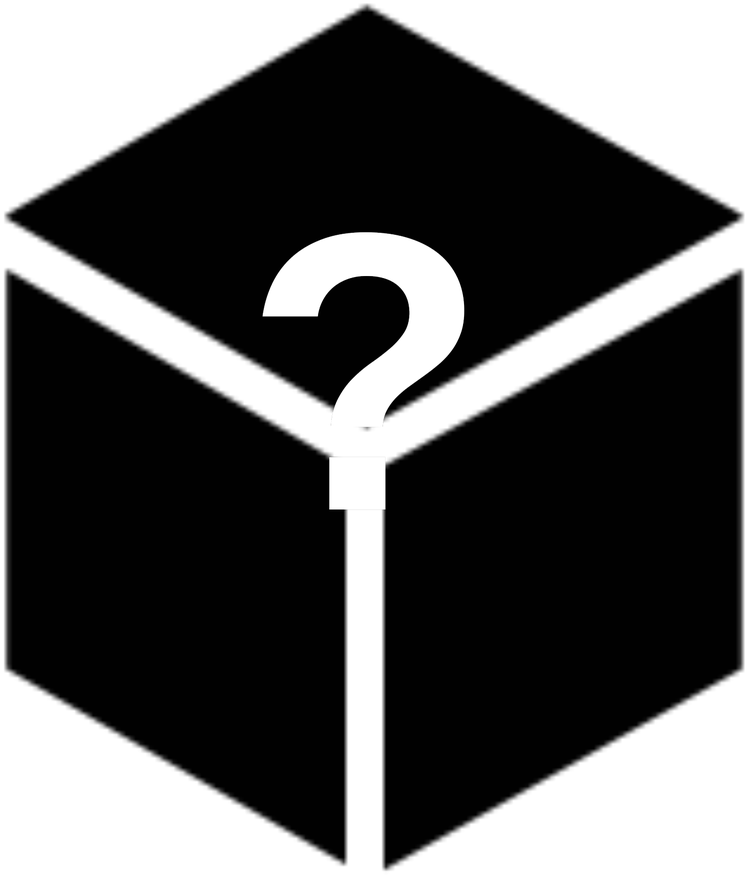


data layout



performance

BOSTON UNIVERSITY



Compaction

black box

How to organize the data on device?

How much data to move at-a-time?

Which block of data to be moved?

When to re-organize the data layout?

Data Layout

How to organize the data on device? ✓

Compaction Granularity

How much data to move at-a-time?

Data Movement Policy

Which block of data to be moved?

Compaction Trigger

When to re-organize the data layout?

Compaction **Granularity**

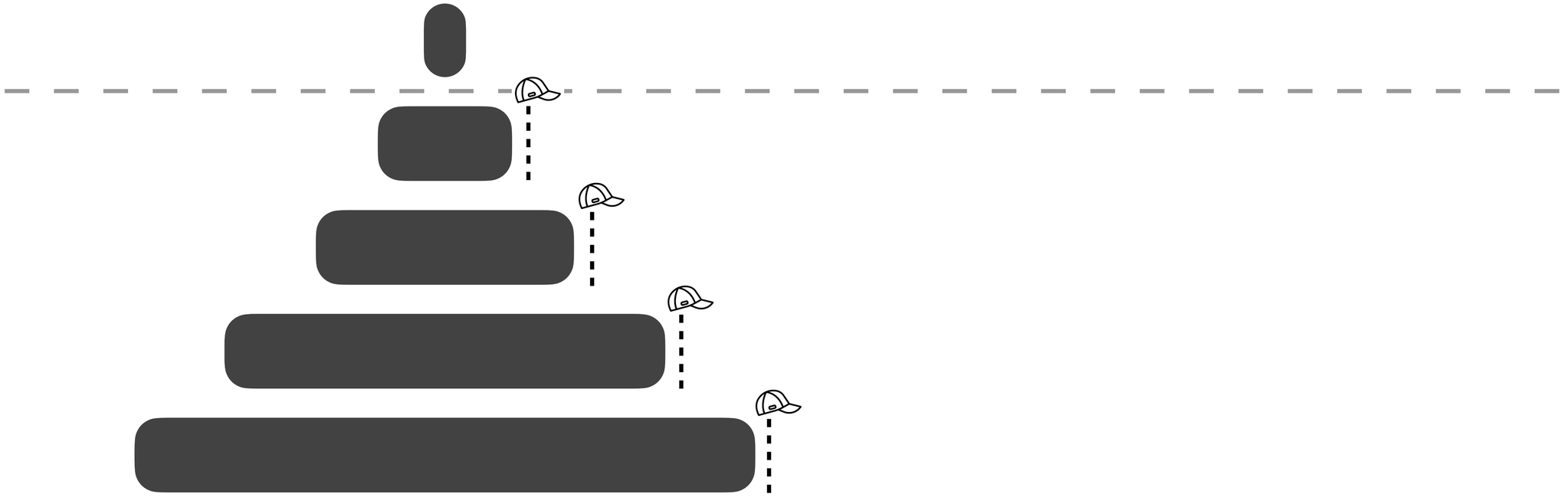
data moved per compaction

consecutive levels

AsterixDB

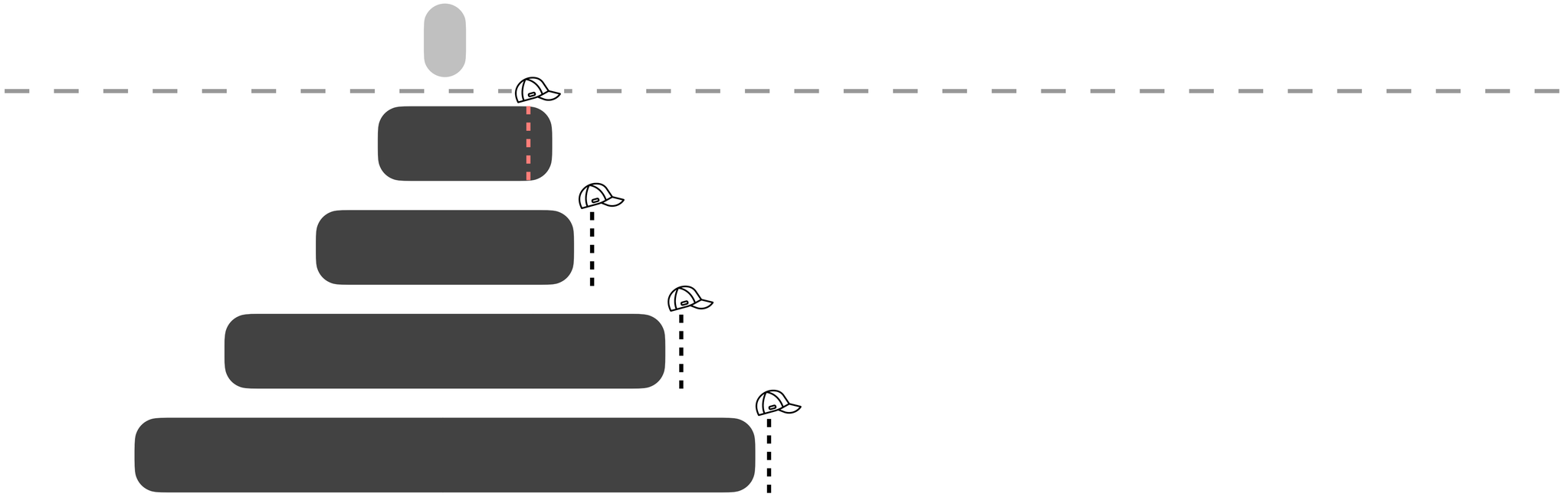
Compaction **Granularity**

data moved per compaction



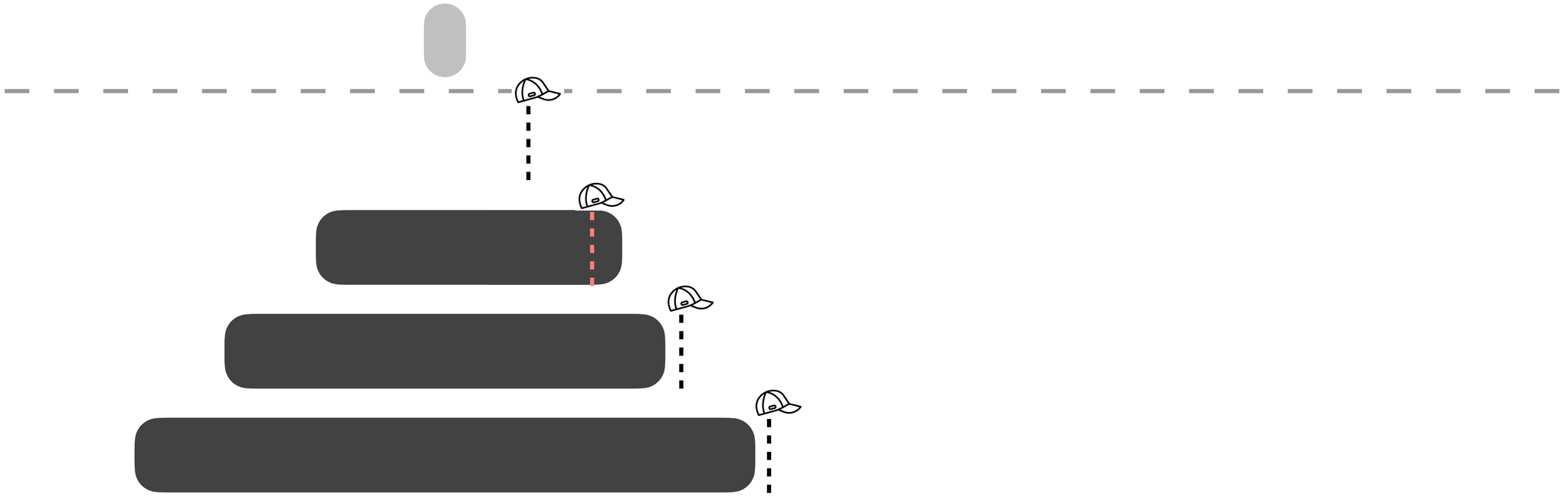
Compaction **Granularity**

data moved per compaction



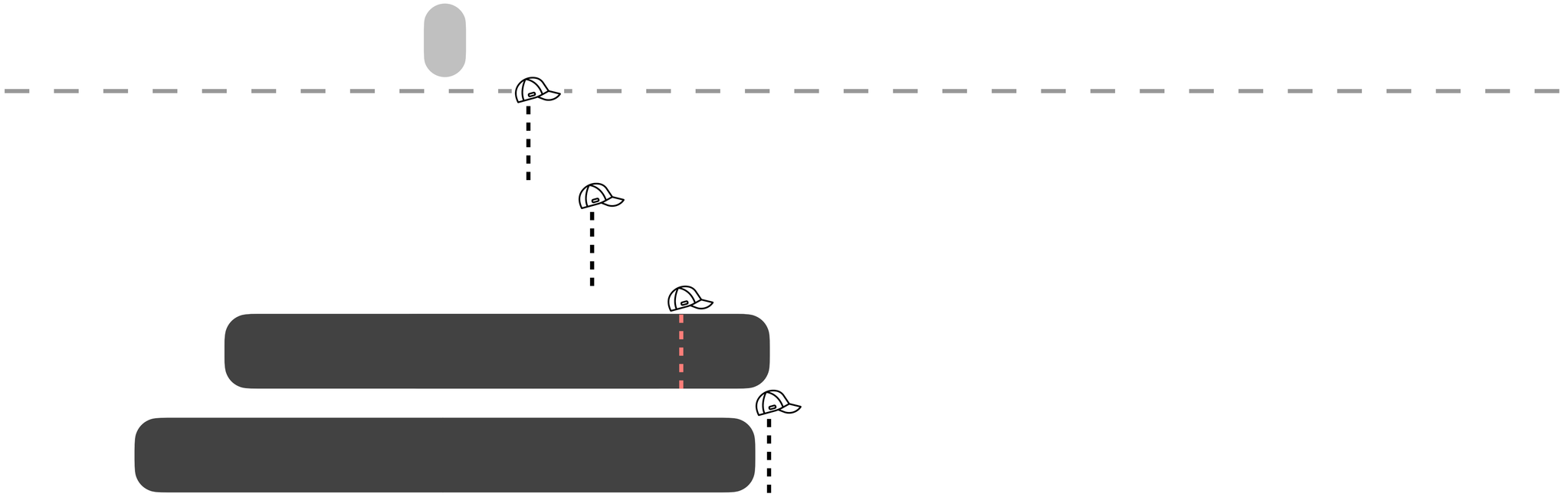
Compaction **Granularity**

data moved per compaction



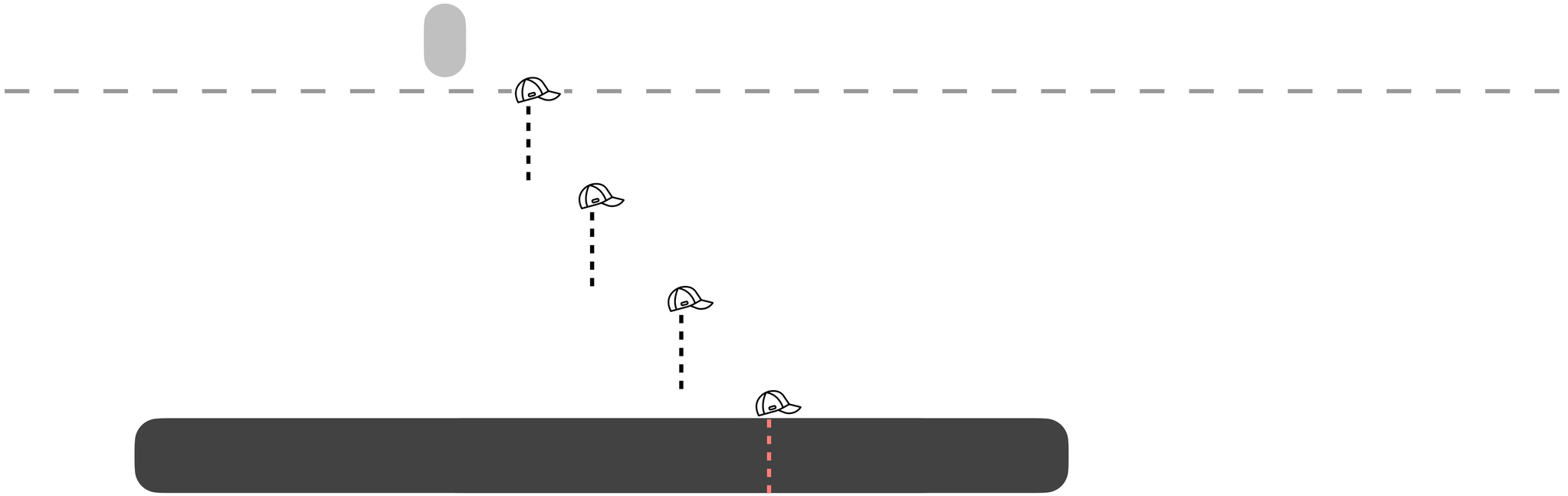
Compaction **Granularity**

data moved per compaction



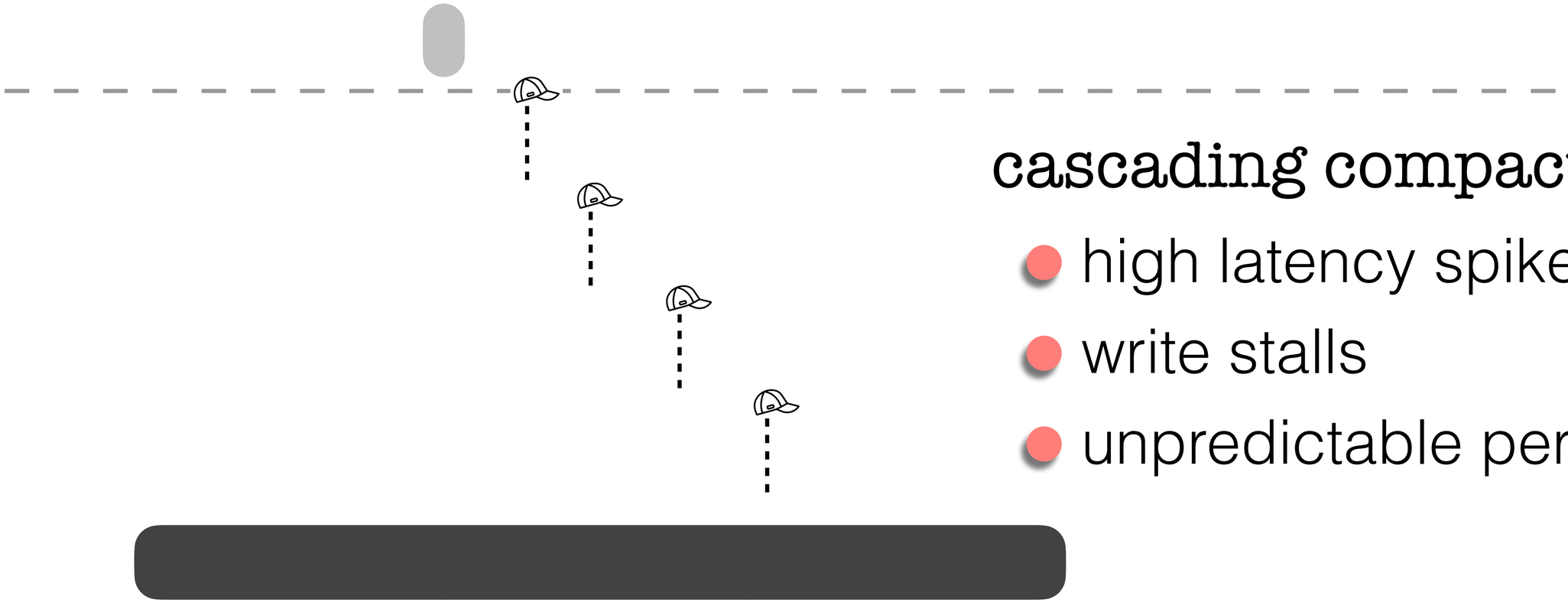
Compaction **Granularity**

data moved per compaction



Compaction **Granularity**

data moved per compaction



cascading compactions

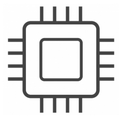
- high latency spikes
- write stalls
- unpredictable perf.

Compaction **Granularity**

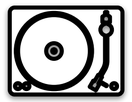
data moved per compaction

partial compaction

granularity: **files**



buffer



level 1

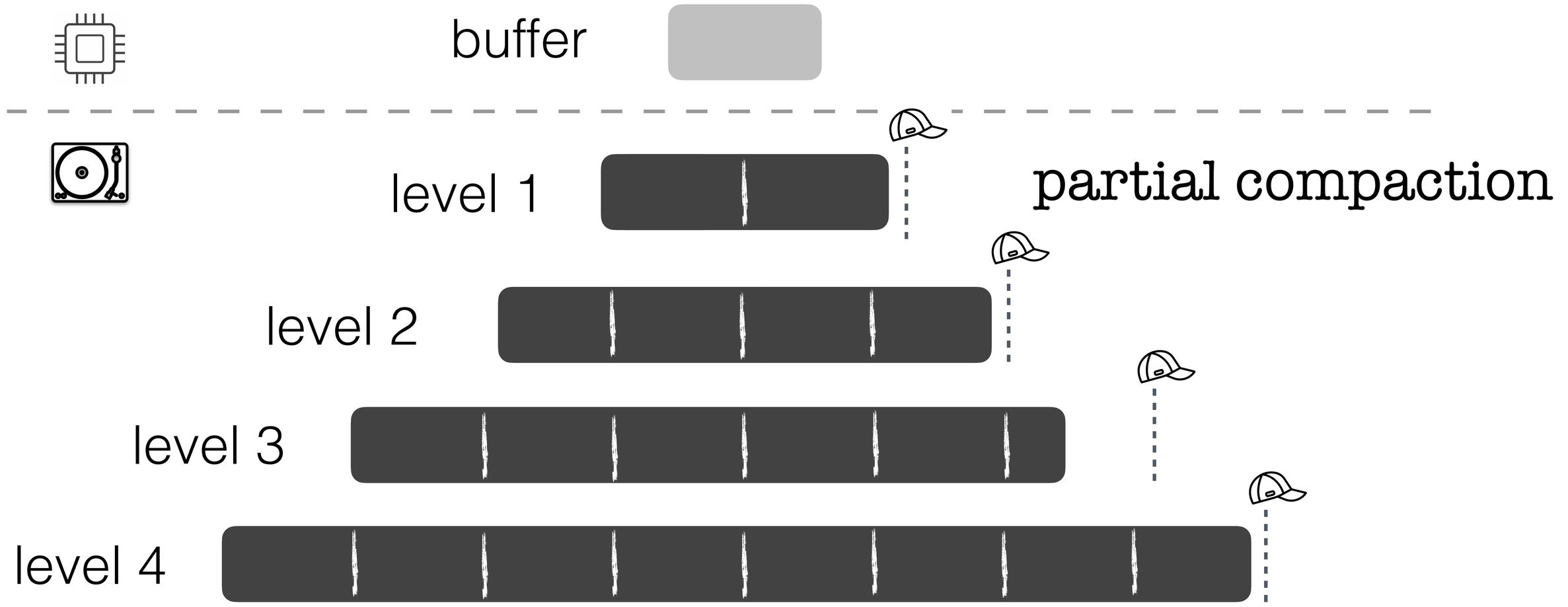
partial compaction

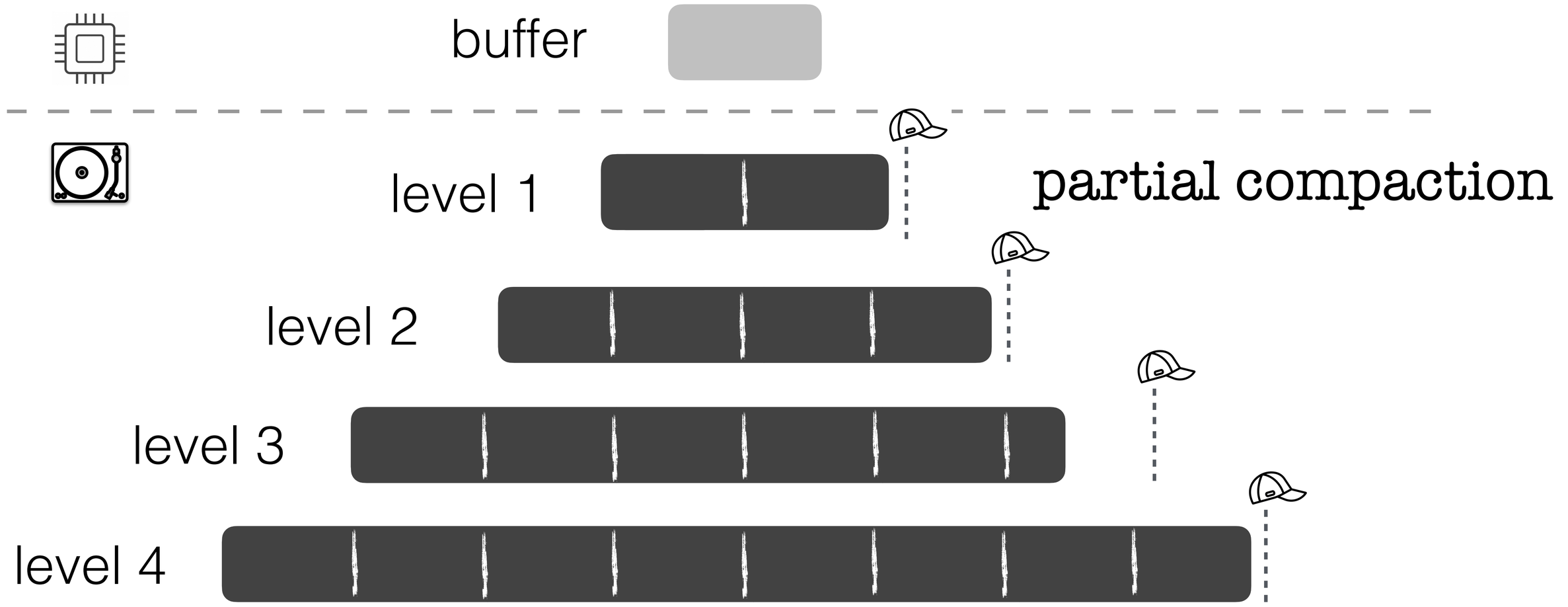
level 2

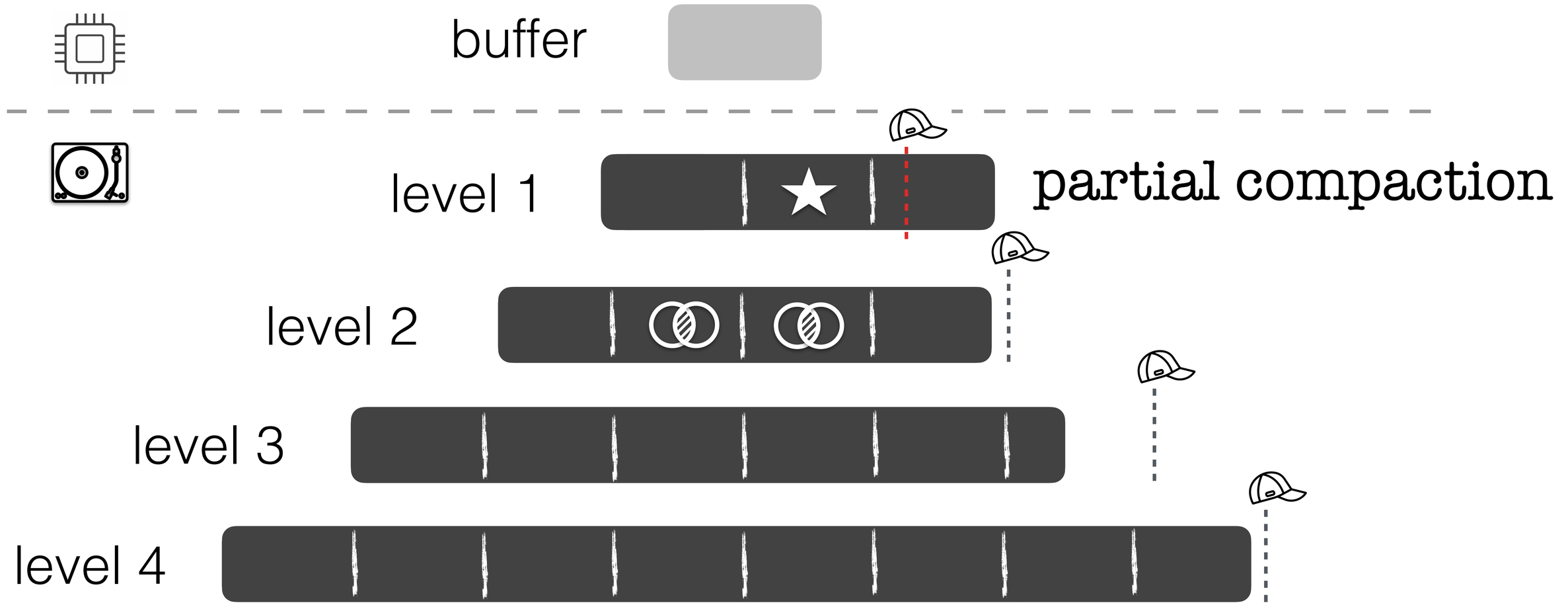
level 3

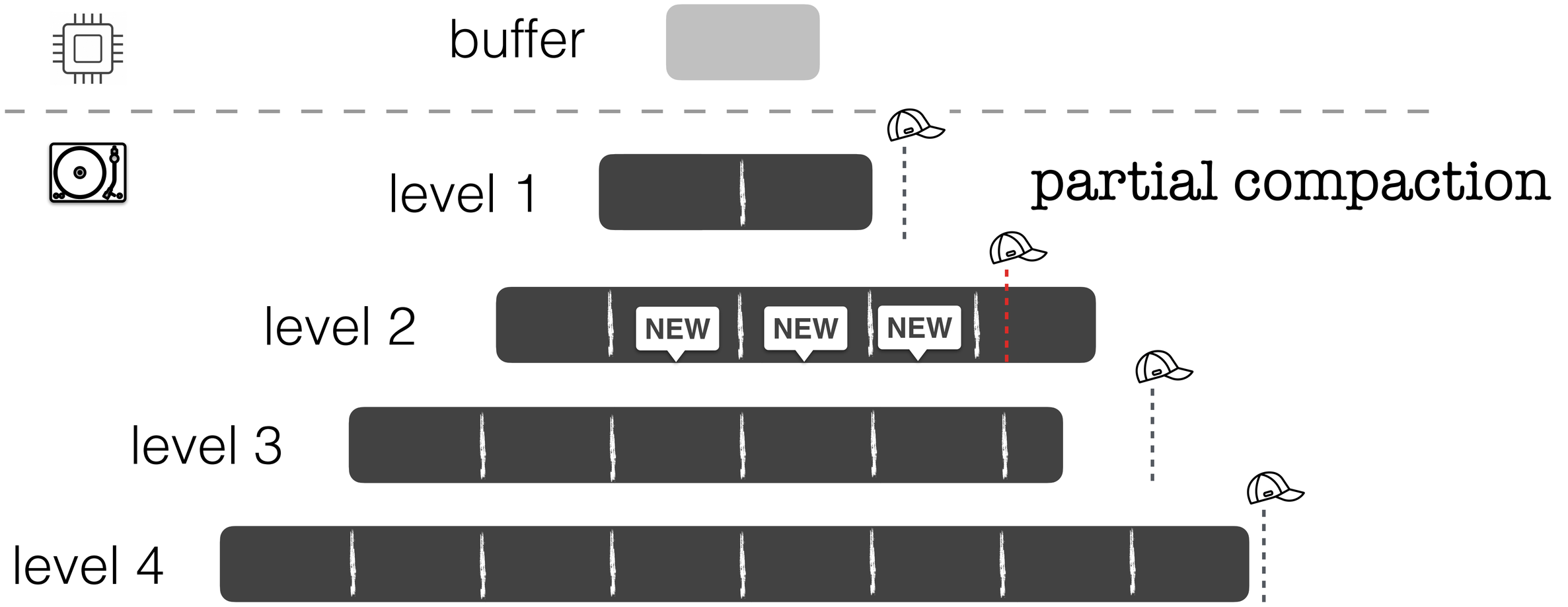
level 4

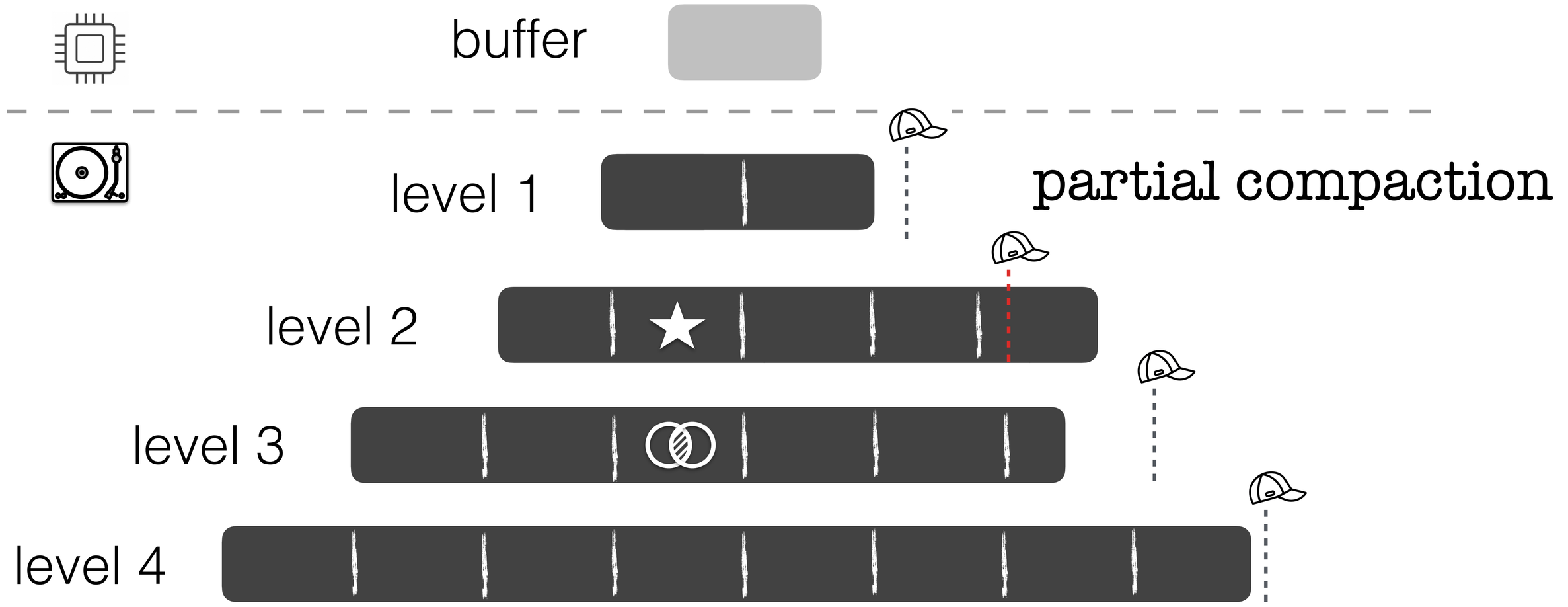
BOSTON UNIVERSITY

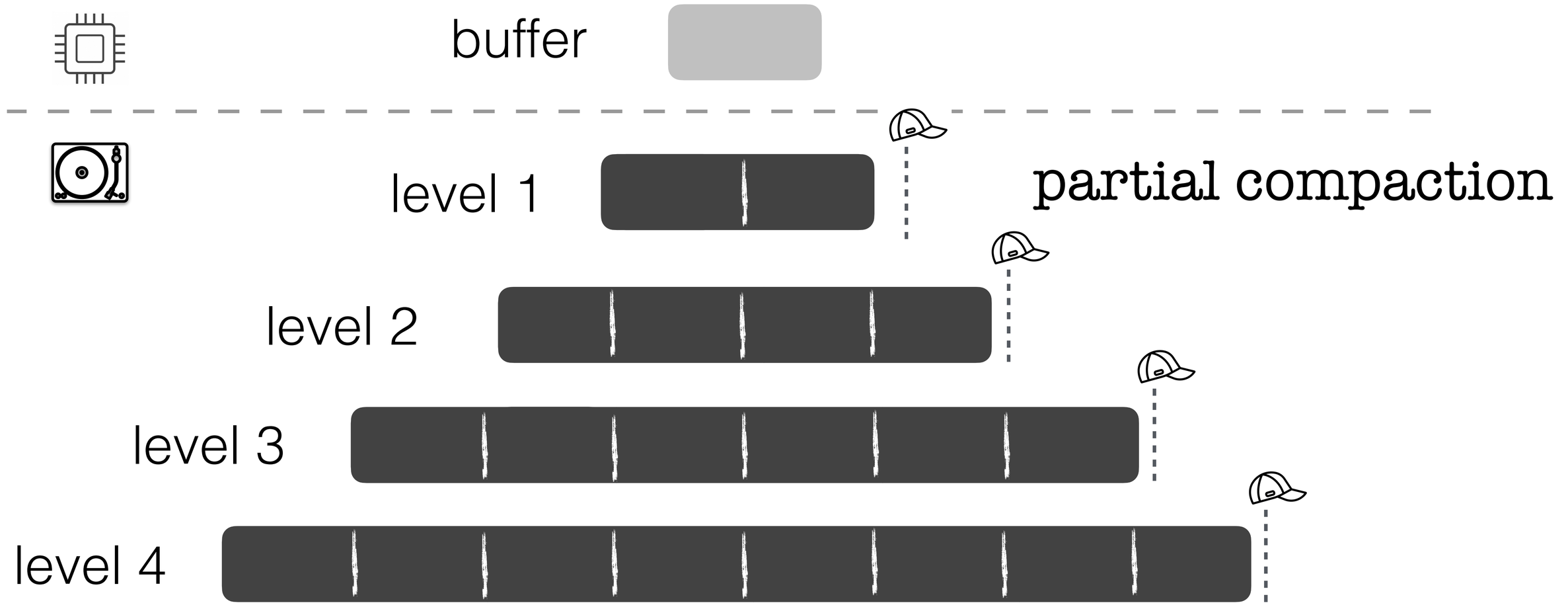





buffer
level 1
partial compaction
level 2
level 3
level 4

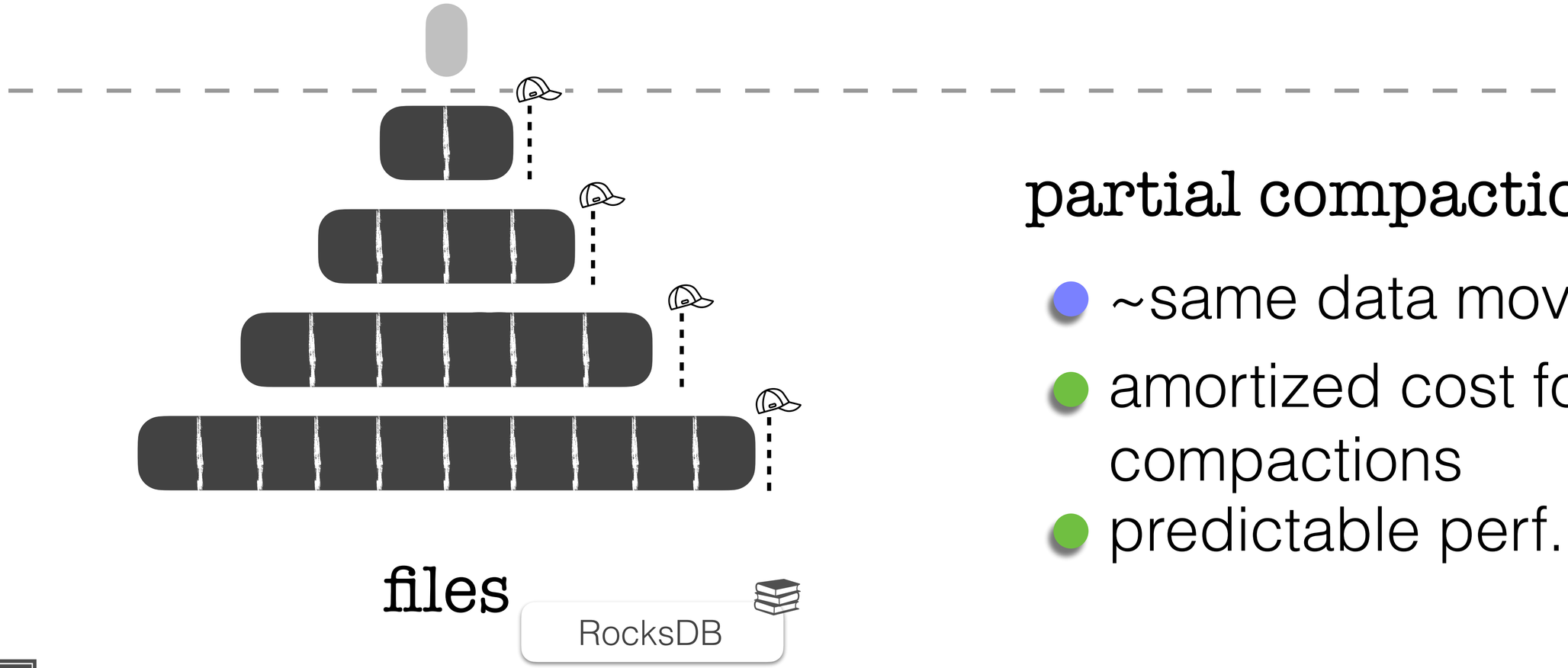

buffer
level 1
partial compaction
level 2
NEW
NEW
NEW
level 3
level 4


buffer
level 1
partial compaction
level 2
level 3
level 4


buffer
level 1
partial compaction
level 2
level 3
level 4

Compaction **Granularity**

data moved per compaction

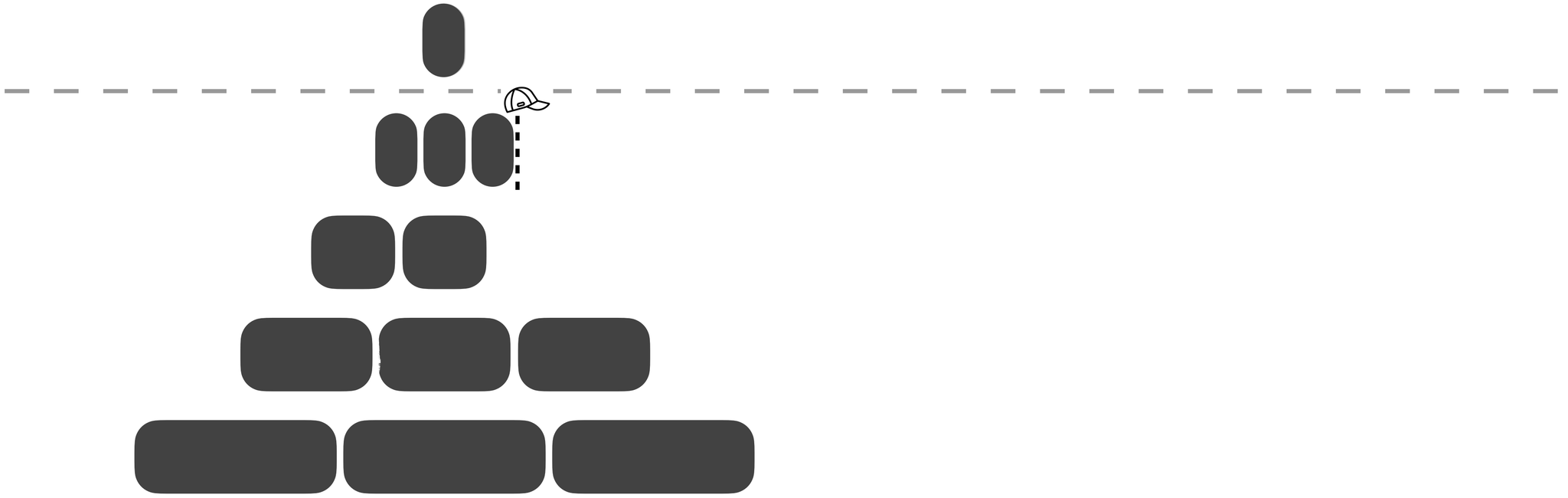


partial compaction

- ~same data movement
- amortized cost for compactions
- predictable perf.

Compaction **Granularity**

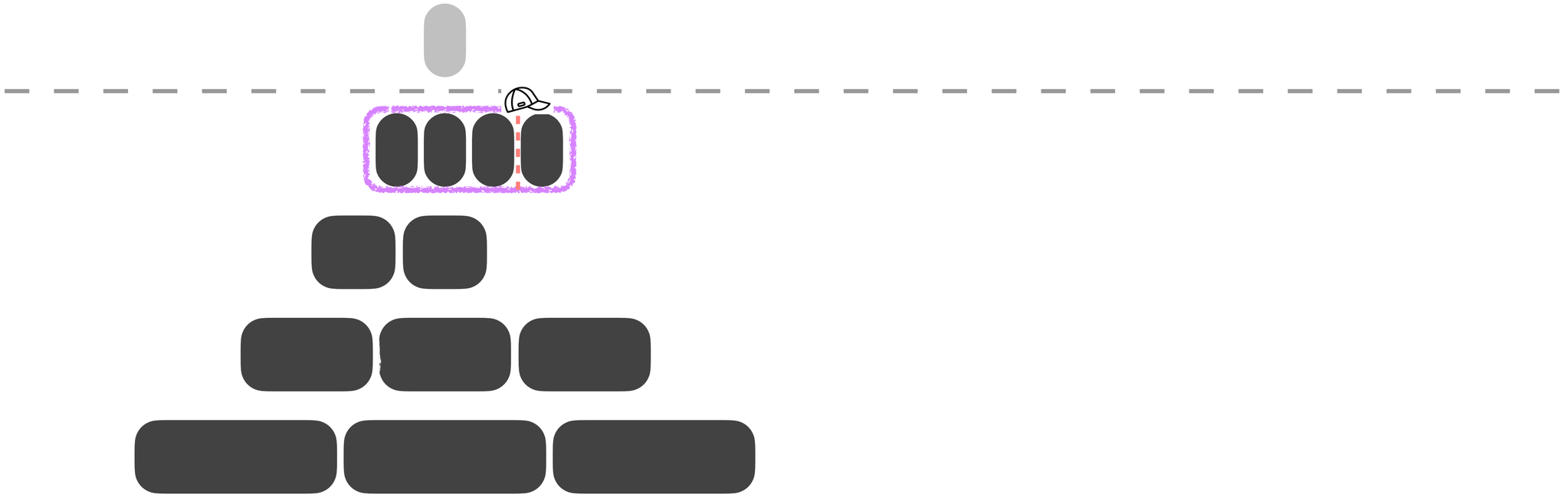
data moved per compaction



sorted runs in a level

Compaction **Granularity**

data moved per compaction

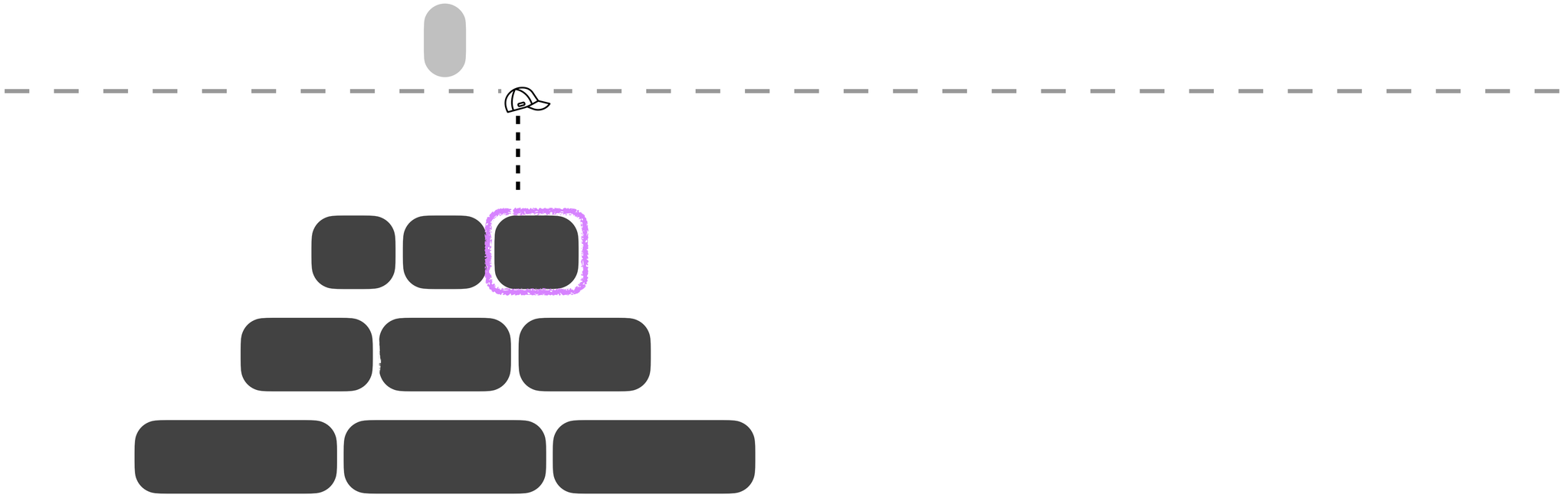


sorted runs in a level

BOSTON UNIVERSITY

Compaction **Granularity**

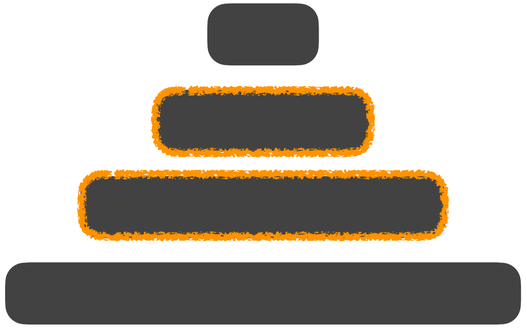
data moved per compaction



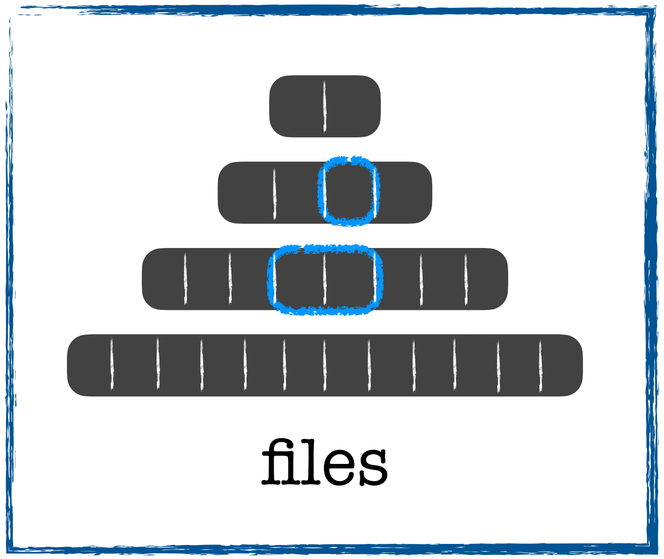
sorted runs in a level

Compaction **Granularity**

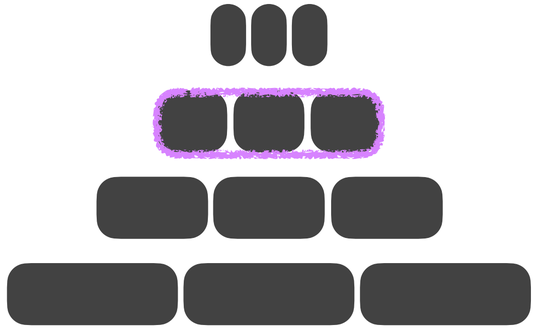
data moved per compaction



levels



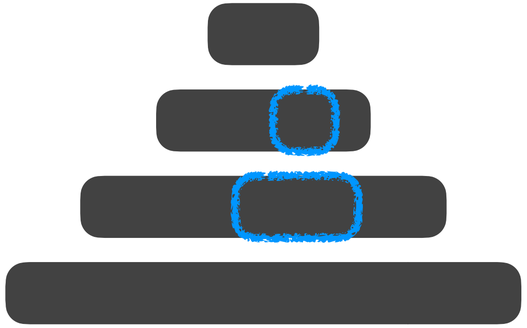
files



sorted runs in a level

Data Movement Policy

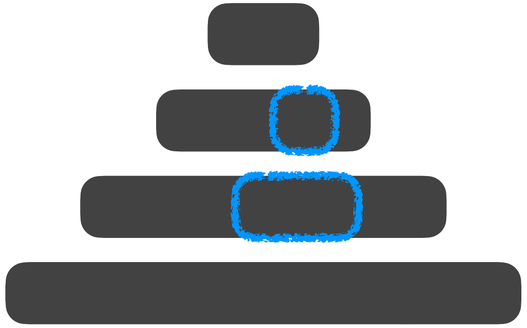
which data to compact



files

Data Movement Policy

which data to compact



files

round-robin !

minimum **overlap with parent** level WA

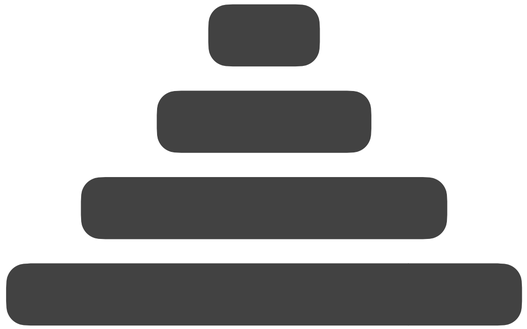
file with most **tombstones** SA

coldest file PQ

Compaction **Trigger**

invoking the compaction routine

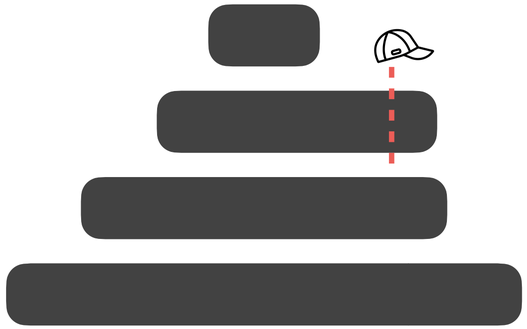
level **saturation**



Compaction **Trigger**

invoking the compaction routine

level **saturation**

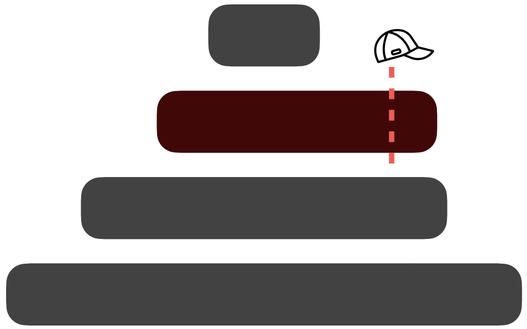


BOSTON UNIVERSITY

Compaction **Trigger**

invoking the compaction routine

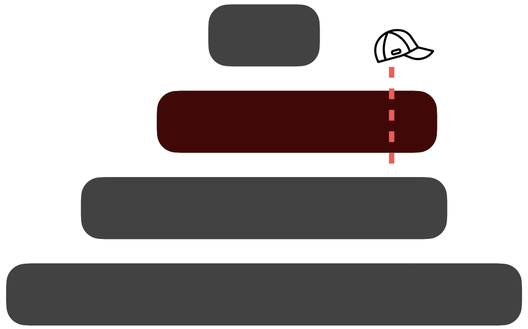
level **saturation**



BOSTON UNIVERSITY

Compaction **Trigger**

invoking the compaction routine



level **saturation**

number of **sorted runs**

space amplification

SA

age of a file

De

Data Layout

Compaction Granularity

Data Movement Policy

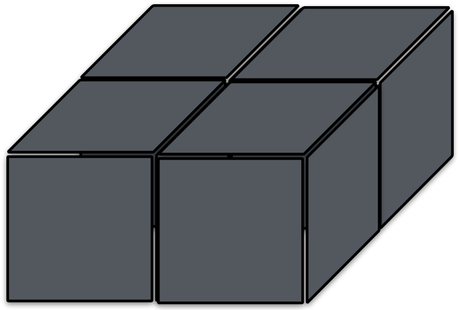
Compaction Trigger

Data Layout

Compaction Granularity

Data Movement Policy

Compaction Trigger



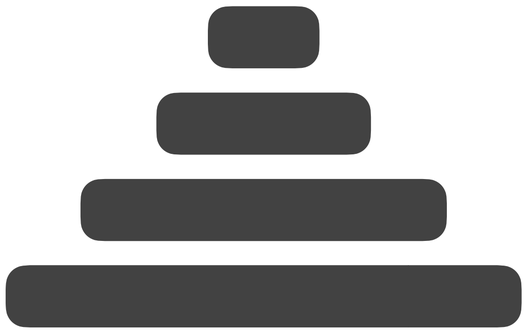
***Any* Compaction Algorithm**

Database	Data layout	Compaction Trigger					Compaction Granularity				Data Movement Policy							
		Level saturation	#Sorted runs	File staleness	Space amp.	Tombstone-TTL	Level	Sorted run	File (single)	File (multiple)	Round-robin	Least overlap (+1)	Least overlap (+2)	Coldest file	Oldest file	Tombstone density	Expired TS-TTL	N/A (entire level)
RocksDB [30], Monkey [22]	Leveling / 1-Leveling	✓		✓					✓	✓		✓		✓	✓	✓		
	Tiering		✓		✓	✓		✓										✓
LevelDB [32], Monkey (J.) [21]	Leveling	✓							✓		✓	✓	✓					
SlimDB [47]	Tiering	✓							✓	✓								✓
Dostoevsky [23]	*L*-leveling	$✓^{L}$	$✓^{T}$				$✓^{L}$	$✓^{T}$				$✓^{L}$						$✓^{T}$
LSM-Bush [24]	Hybrid leveling	$✓^{L}$	$✓^{T}$				$✓^{L}$	$✓^{T}$				$✓^{L}$						$✓^{T}$
Lethe [51]	Leveling	✓				✓			✓	✓		✓					✓	
Silk [11], Silk+ [12]	Leveling	✓							✓	✓	✓							
HyperLevelDB [35]	Leveling	✓							✓		✓	✓	✓					
PebblesDB [46]	Hybrid leveling	✓							✓	✓								✓
Cassandra [8]	Tiering		✓	✓		✓		✓										✓
	Leveling	✓				✓			✓	✓		✓				✓	✓	
WiredTiger [62]	Leveling	✓					✓											✓
X-Engine [34], Leaper [63]	Hybrid leveling	✓							✓	✓		✓				✓		
HBase [7]	Tiering		✓					✓										✓
AsterixDB [3]	Leveling	✓					✓											✓
	Tiering		✓					✓										✓

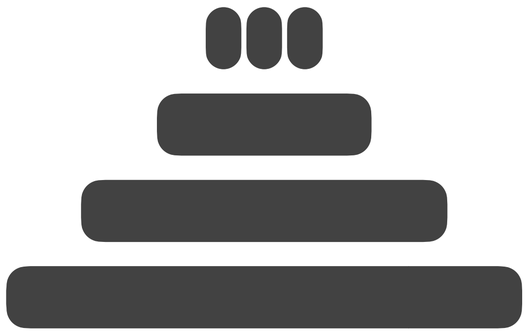
BOSTON UNIVERSITY

Storage Layer **Design Continuum**

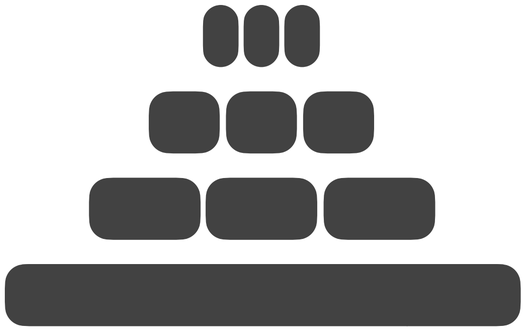
leveling



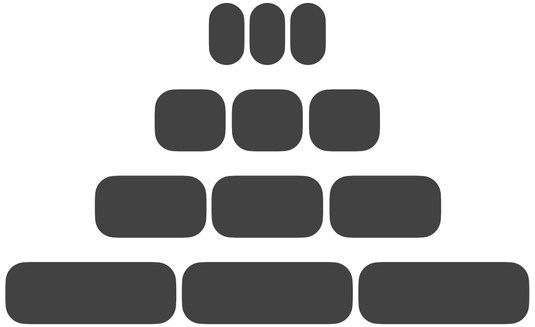
1-leveling



L-leveling



tiering

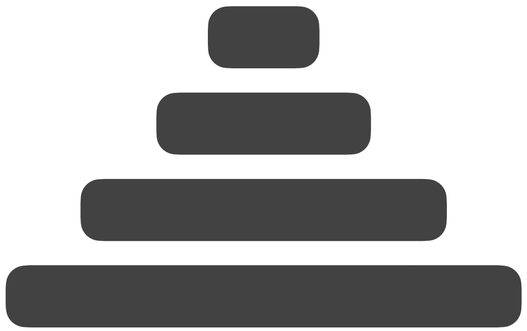


Any design can be defined by the tuple-set: $(\mathtt{T}, \mathtt{i})$

Storage Layer **Design Continuum**

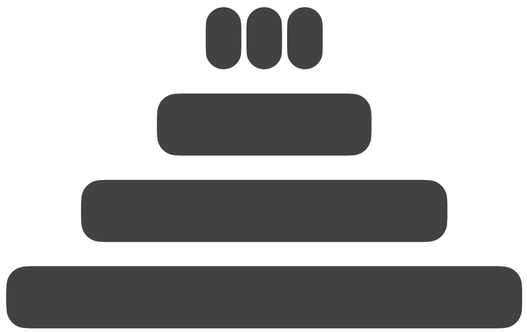
leveling

(T, 0)



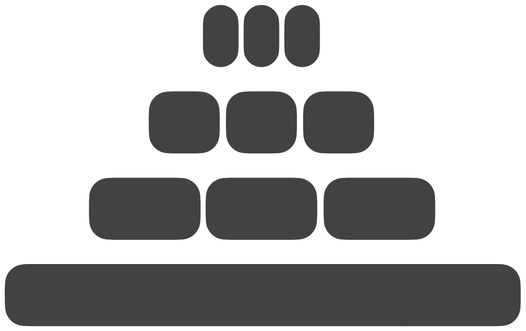
1-leveling

(T, 1)



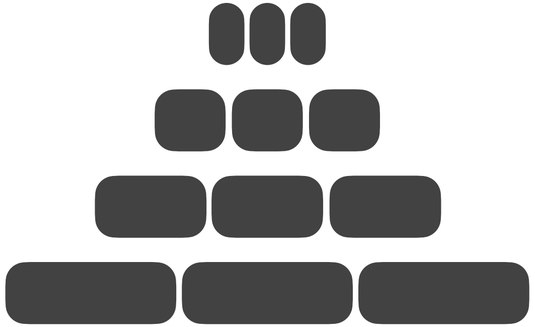
L-leveling

(T, L)



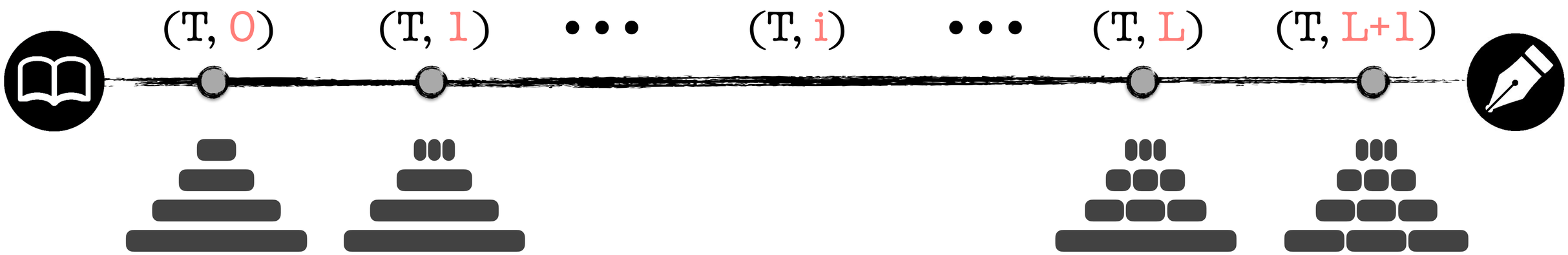
tiering

(T, L+1)

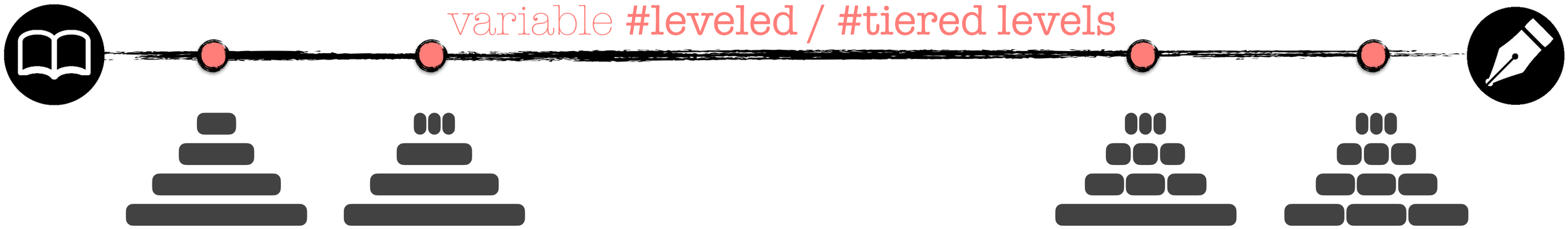


Any design can be defined by the tuple-set: (T, i)

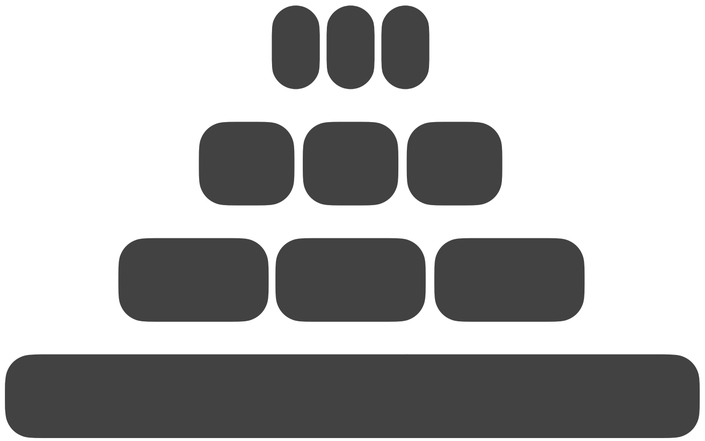
Storage Layer **Design Continuum**



Storage Layer **Design Continuum**



Storage Layer **Design Continuum**



size ratio

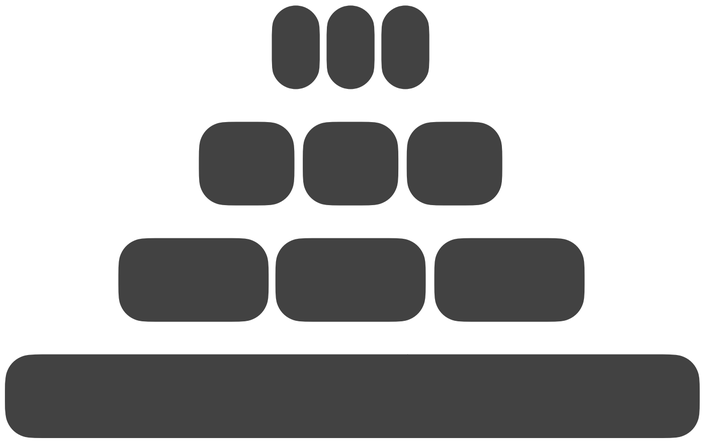
T

T

T

T

Storage Layer **Design Continuum**



size ratio

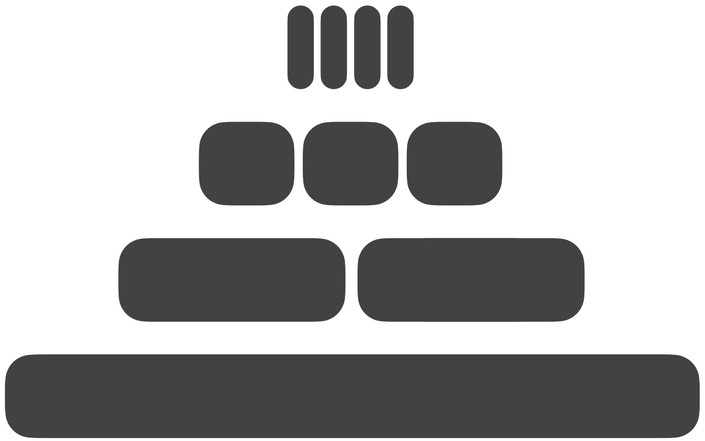
T

T

T

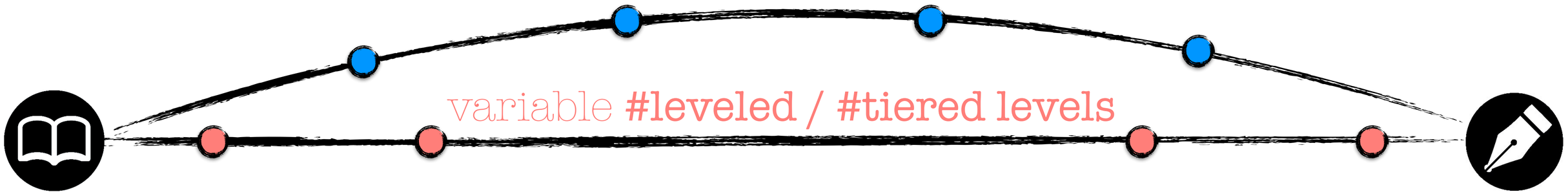
T

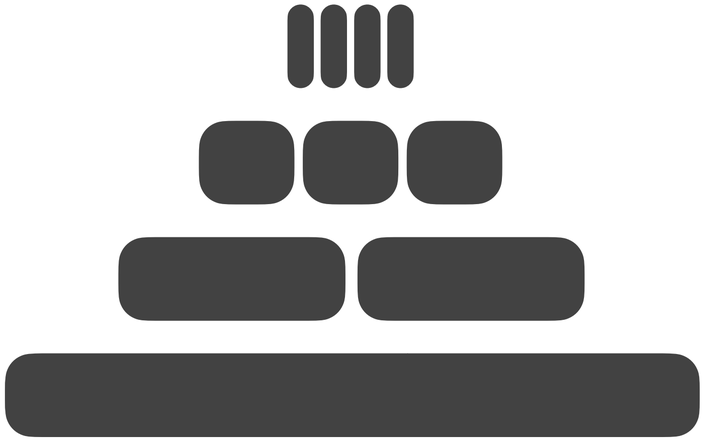
Storage Layer **Design Continuum**



size ratio	#runs
T	4
T	3
T	2
T	1

Storage Layer **Design Continuum**

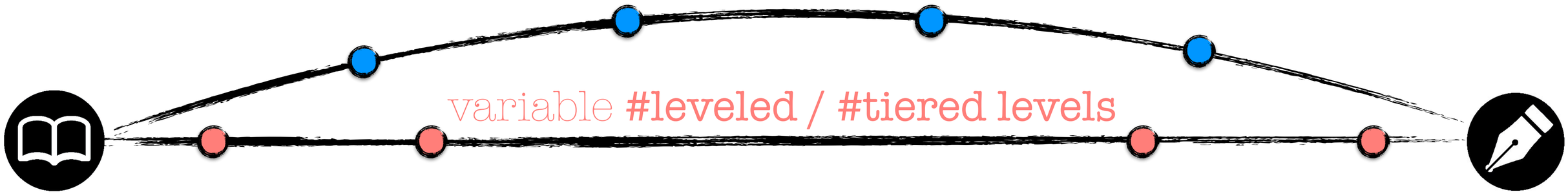




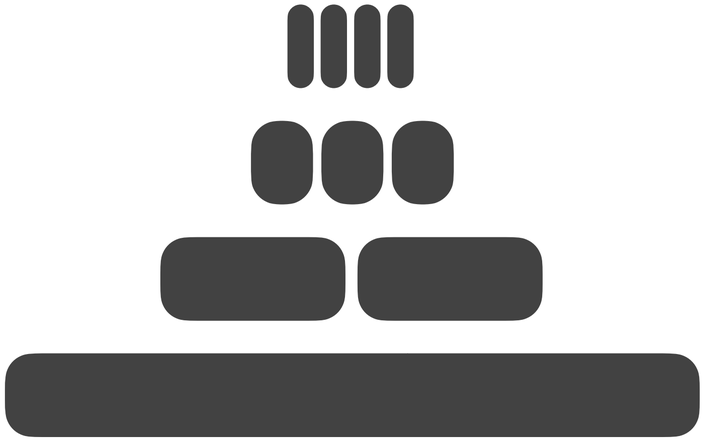
size ratio	#runs
T	4
T	3
T	2
T	1

Storage Layer **Design Continuum**

variable **#runs/level**

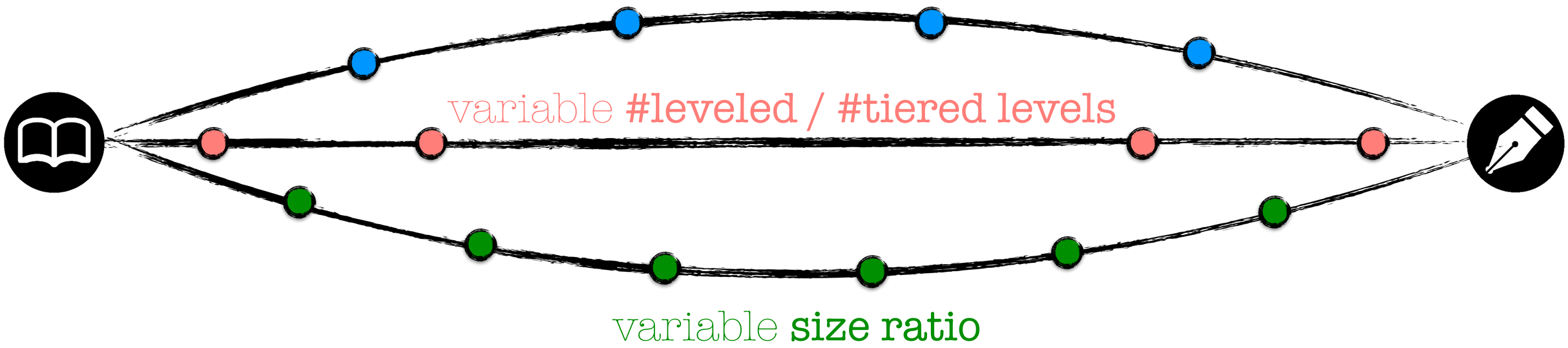


variable **#leveled / #tiered levels**

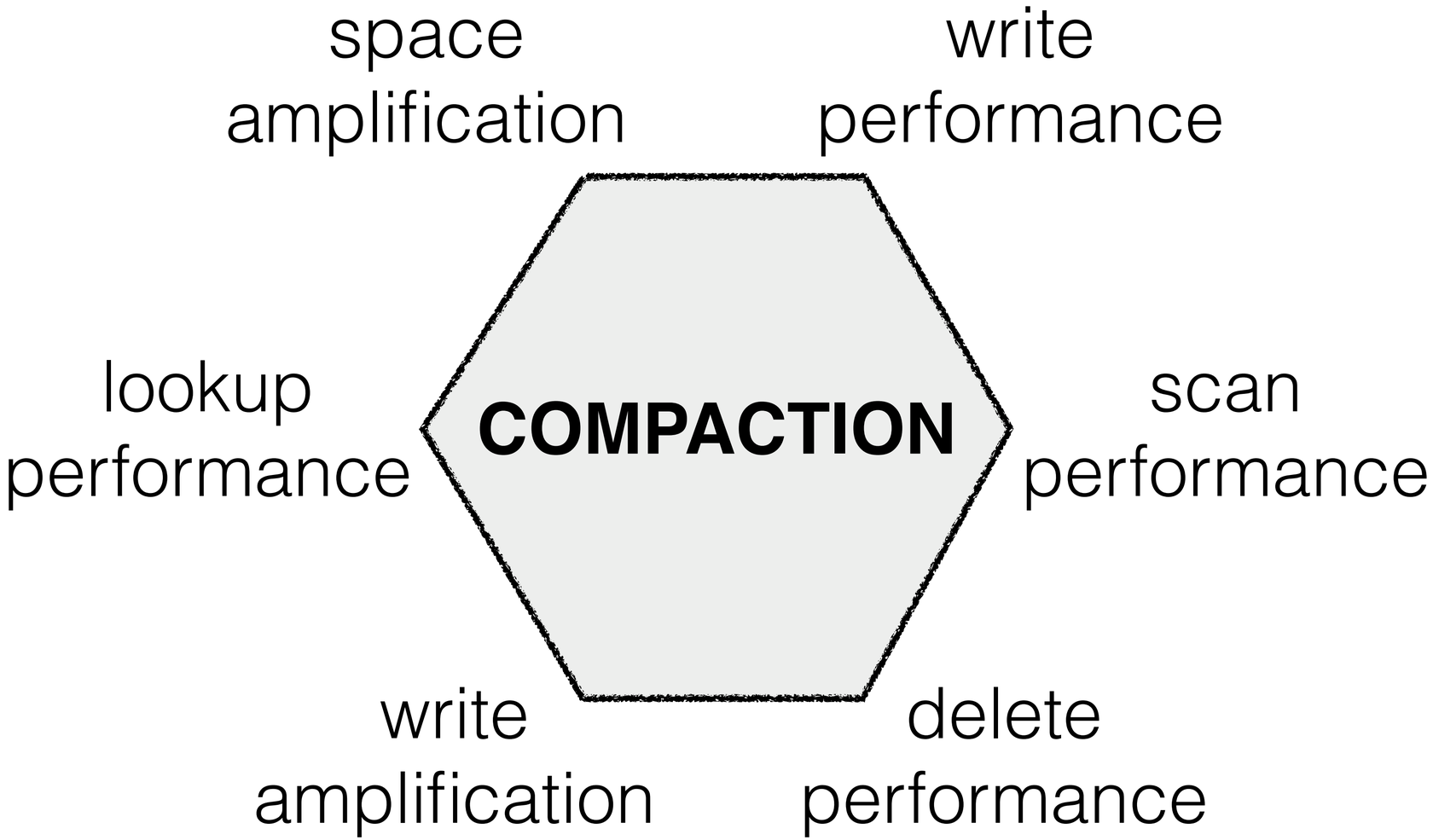


size ratio	#runs
2	4
2.5	3
3	2
4	1

Storage Layer **Design Continuum**

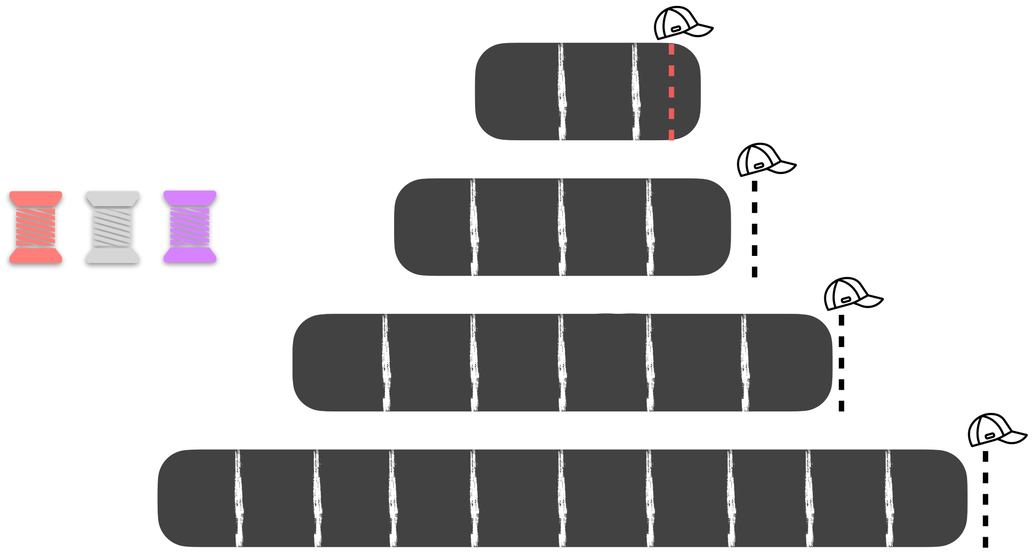


The LSM storage layer design continuum


space
amplification
write
performance
lookup
performance
COMPACTION
scan
performance
write
amplification
delete
performance

Optimizing **Compactions**

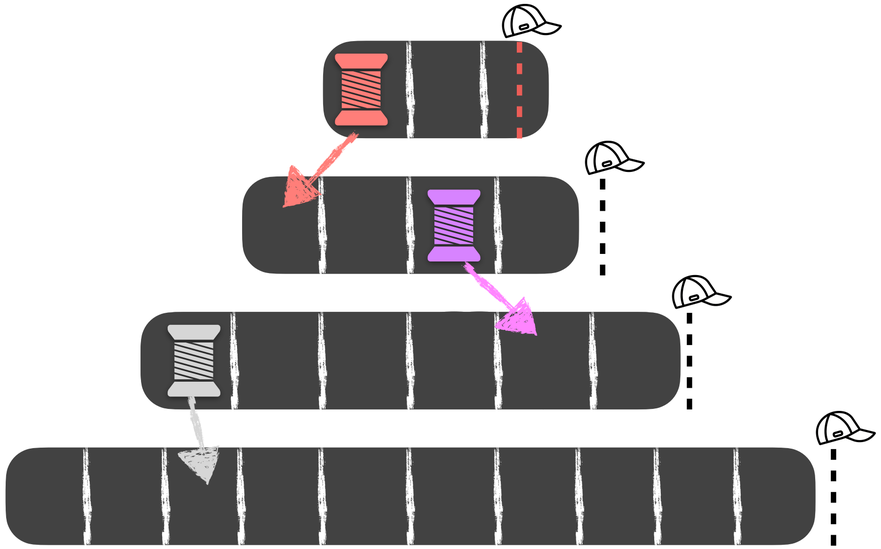
Background Compactions



BOSTON UNIVERSITY

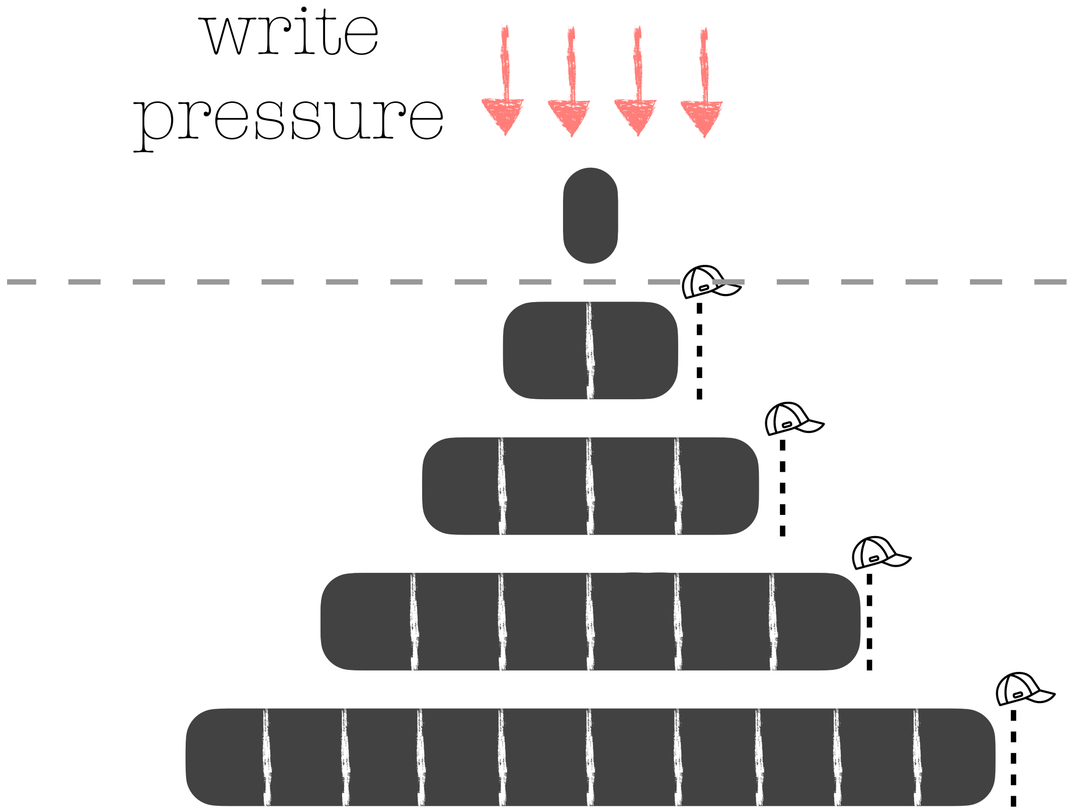
Optimizing **Compactions**

Background Compactions



- non-blocking reads/writes
- improves write throughput

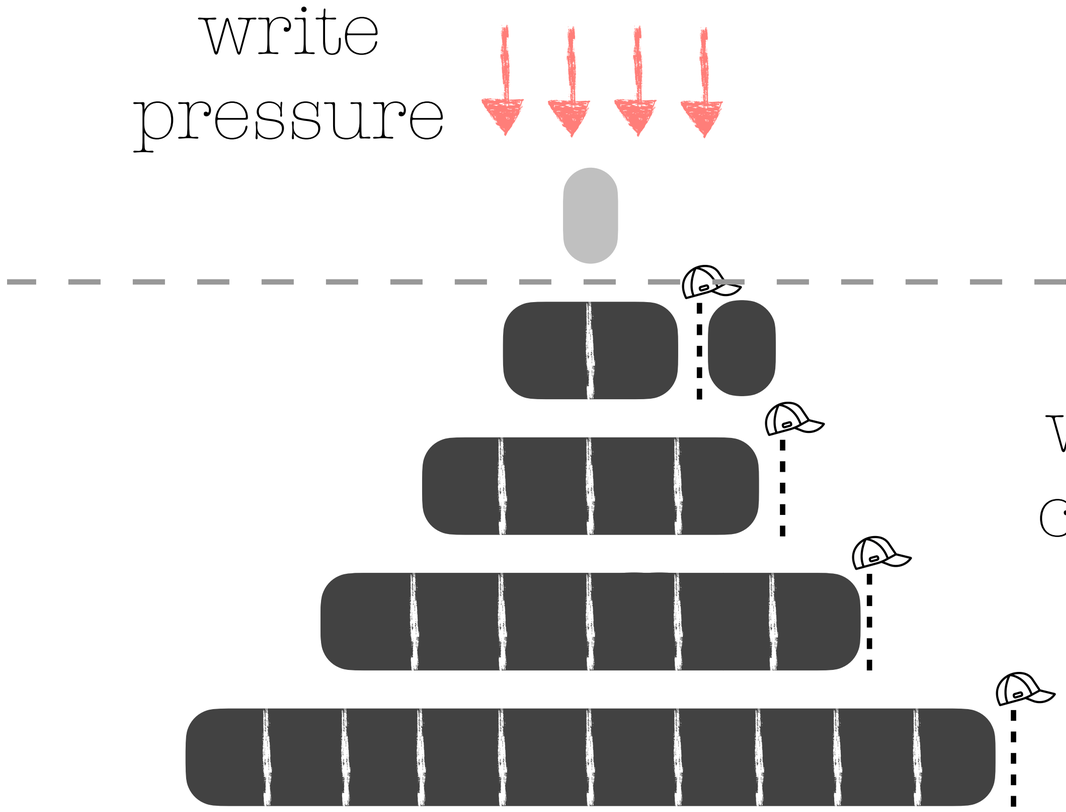
Optimizing **Compactions**



Background Compactions

Compaction Priority

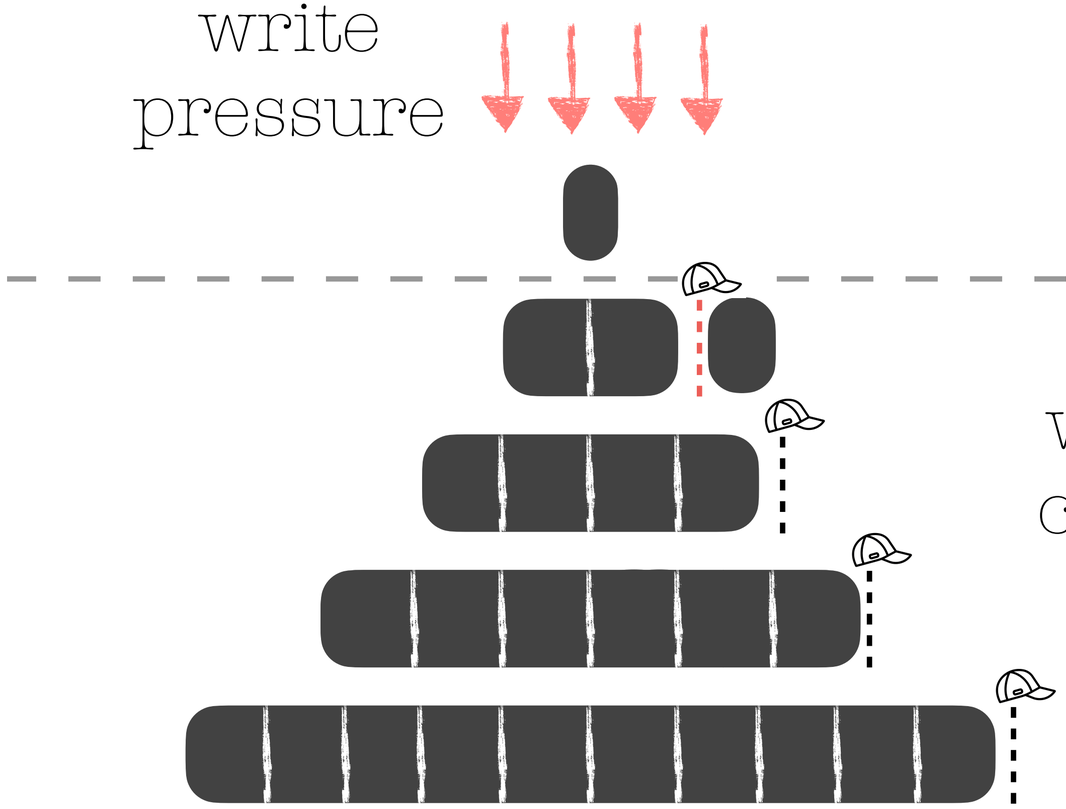
Optimizing **Compactions**



Background Compactions

Compaction Priority

Optimizing **Compactions**

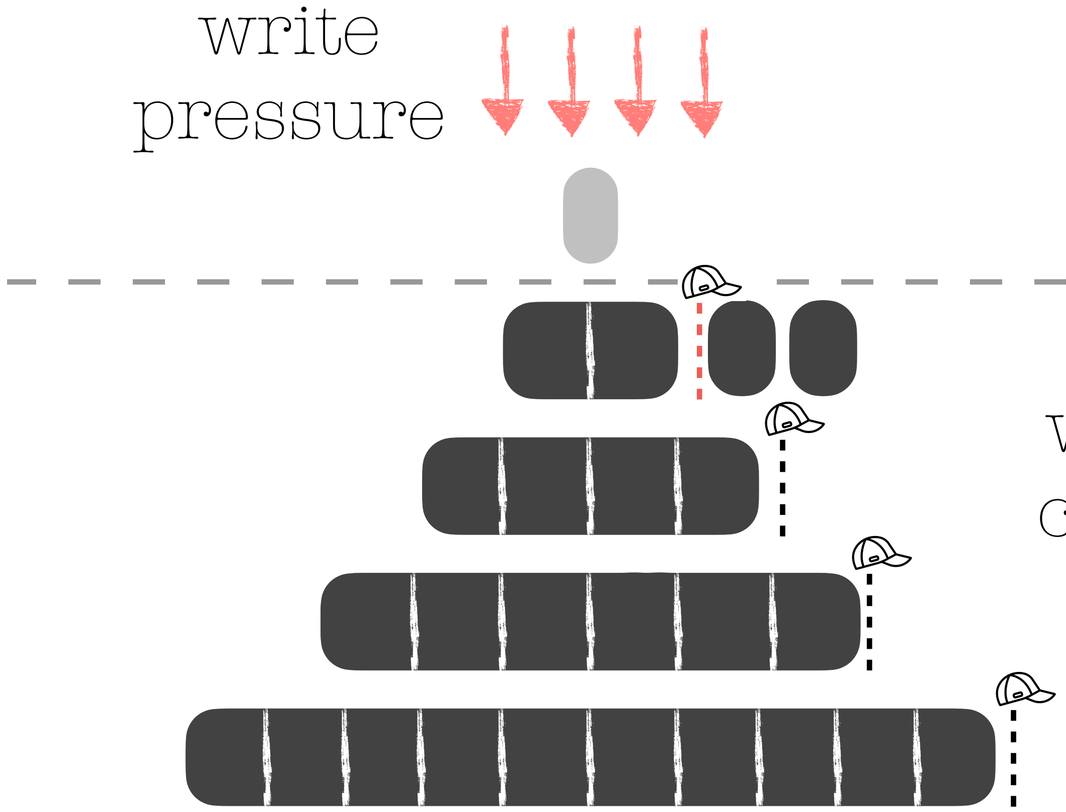


prioritize
writes over
compaction

Background
Compactions

Compaction
Priority

Optimizing **Compactions**

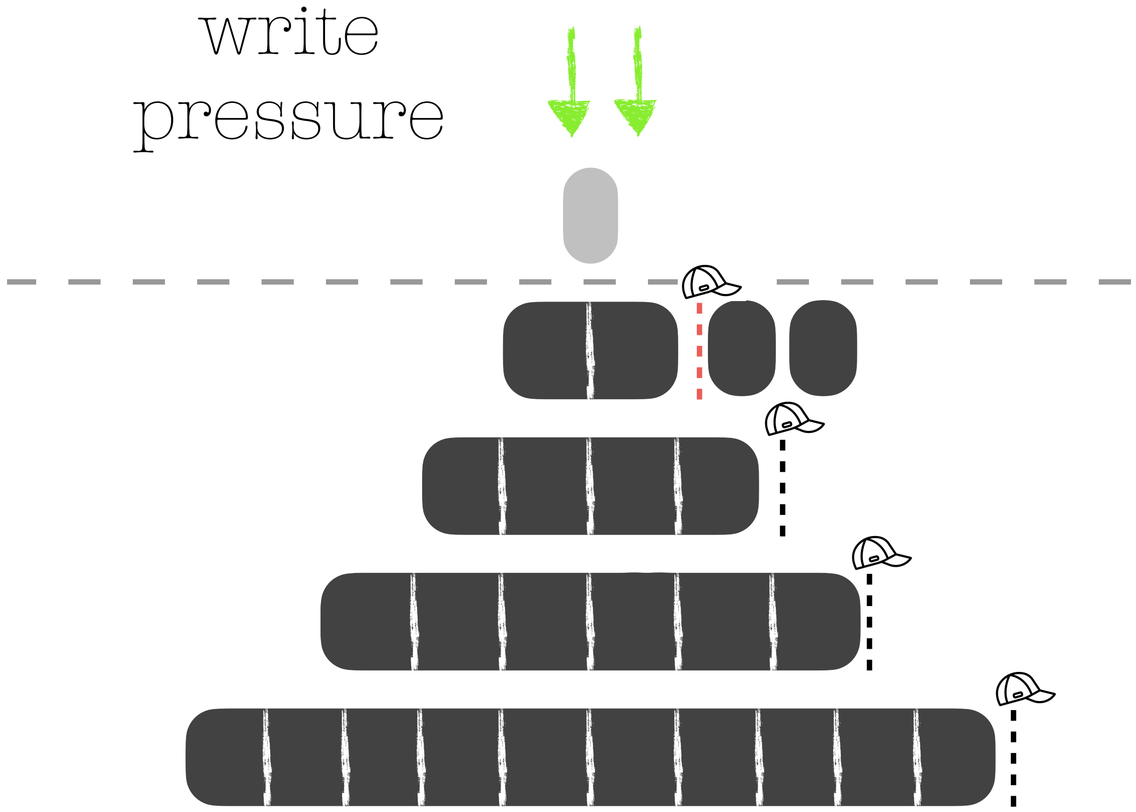


prioritize
writes over
compaction

Background
Compactions

Compaction
Priority

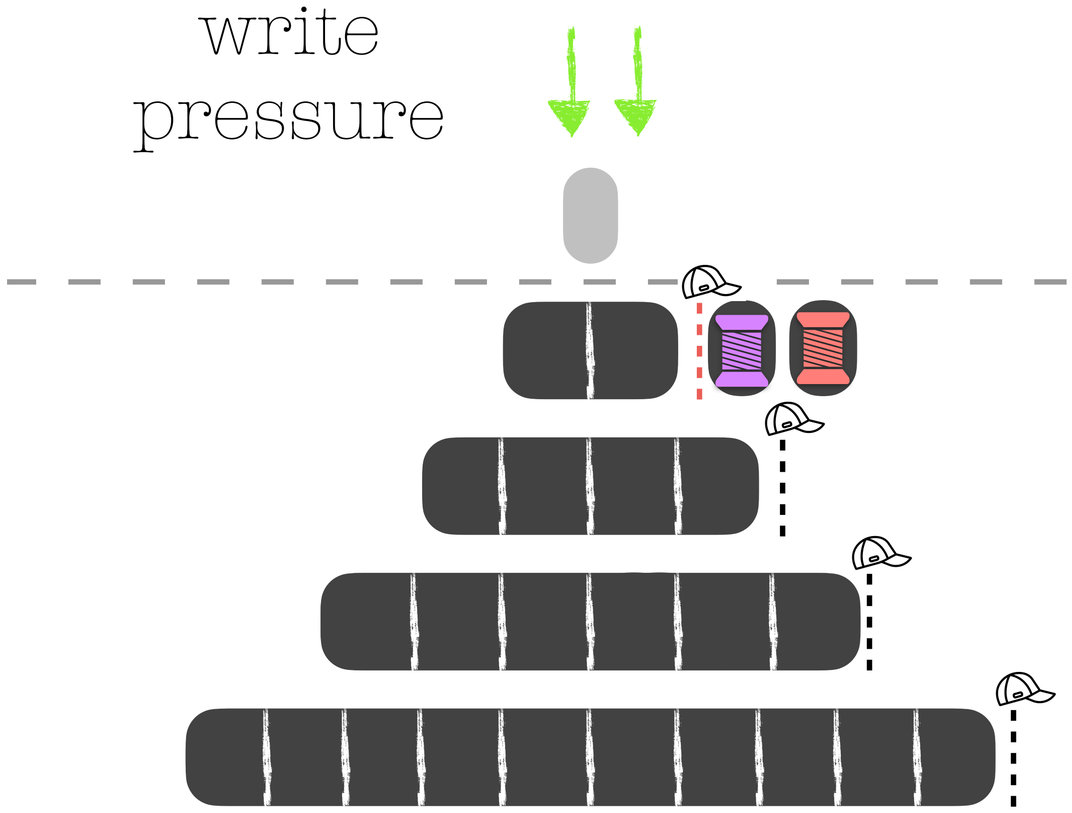
Optimizing **Compactions**



Background Compactions

Compaction Priority

Optimizing **Compactions**

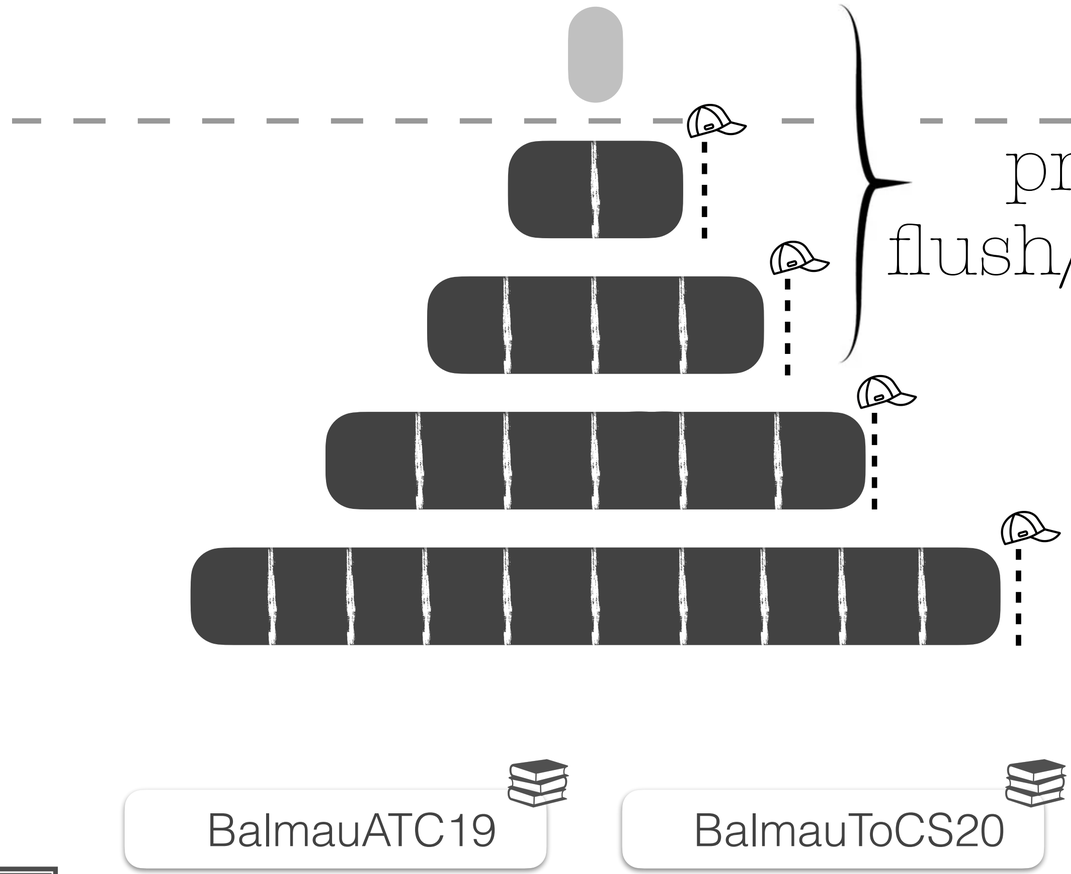


Background Compactions

Compaction Priority

- sustain heavy write bursts
- tree becomes out of shape

Optimizing **Compactions**



Background Compactions

Compaction Priority

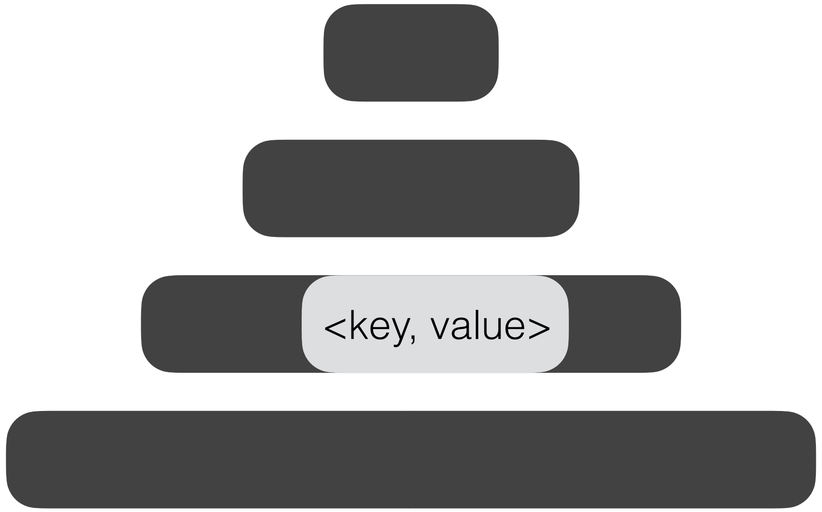
I/O Scheduler

- eliminates write stalls
- no unnecessary high-priority compactions in lower levels

BalmauATC19

BalmauToCS20

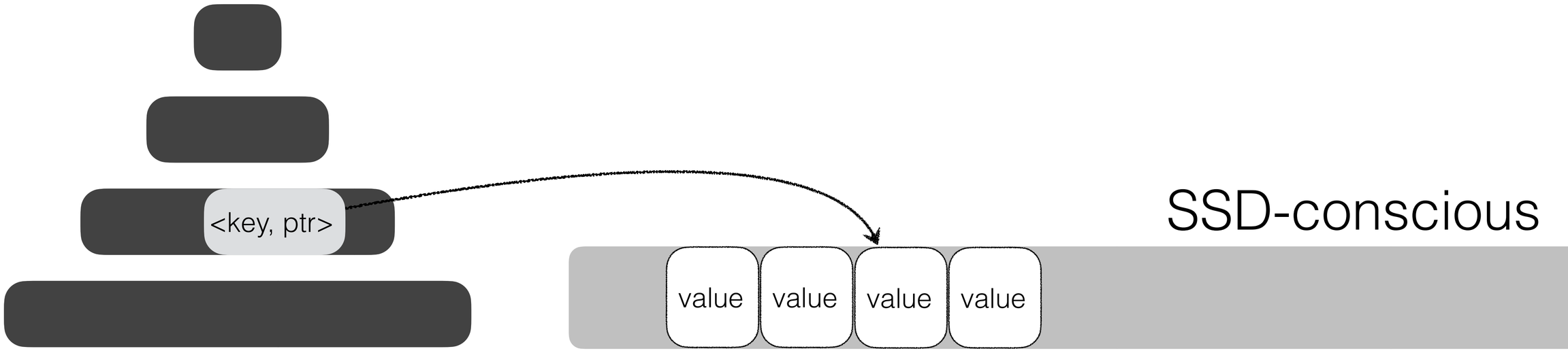
Data Placement Variations



key-value separation

LuFAST16

Data Placement Variations



key-value separation

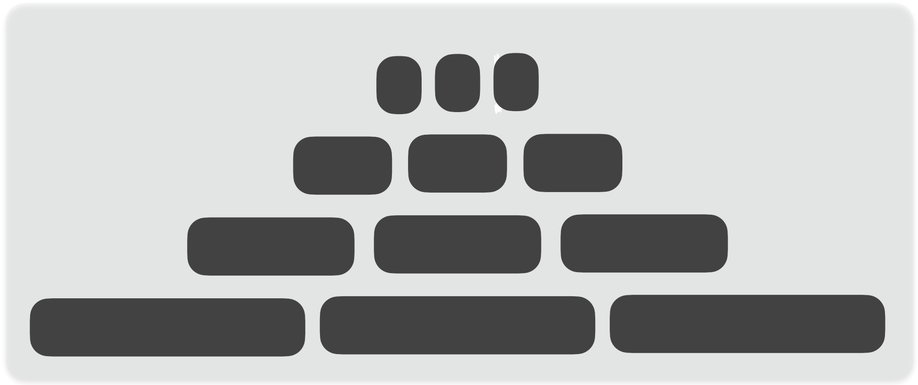
LuFAST16

- reduced write amplification
- better read performance

Data Placement Variations

partitioning / sharding

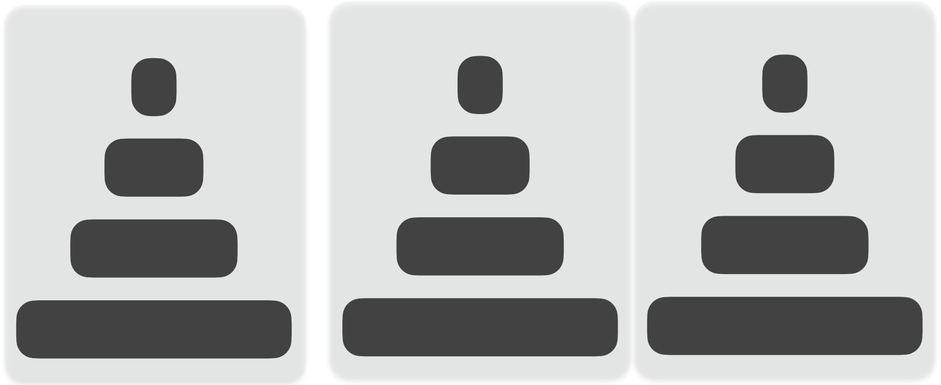
Data Placement Variations



storage

partitioning

RajuSOSP17



storage-1 storage-2 storage-3

sharding

HuangSIGMOD21

- improved ingestion throughput
- reduced write amplification

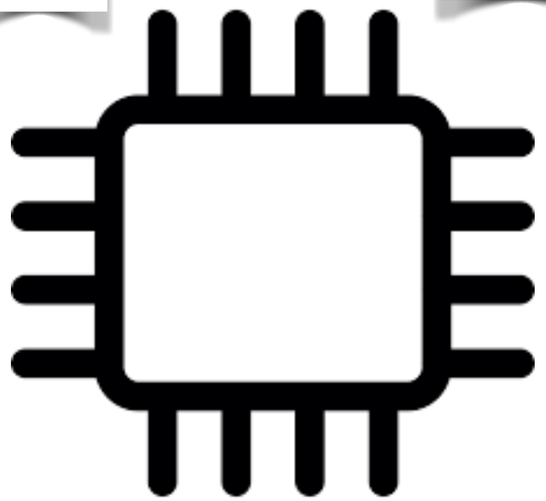
BOSTON UNIVERSITY

Summary: **Ingestion Optimization**

data structure

size

flush strategy



main memory design elements

compaction design space

tuning compactions

data layout on storage

key-value separation

data partitioning

data sharding

hardware-conscious designs

BOSTON UNIVERSITY

No Labs on 02/03

Stay
Warm

CS 561: ***Data Systems Architecture***

Class 5

Log-Structured Merge (LSM) Trees

Dr. Subhadeep Sarkar

BOSTON UNIVERSITY

DiSC lab

https://bu-disc.github.io/CS561/