Constructing and *Analyzing* the **LSM Compaction Design Space**

Subhadeep Sarkar Dimitris Staratzis

Zichen Zhu Manos Athanassoulis

BOSTON UNIVERSITY

DiSC lab

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

LSM-tree

LSM-tree

WT

levelDB

SCYLLA

APACHE HBASE

cassandra

tarantool

RocksDB

Bigtable

DynamoDB

accumulo™

SQLite

riak

QuasarDB

influxdb

2021

LSM-tree

APACHE HBASE

WT

levelDB

SCYLLA

cassandra

tarantool

RocksDB

Bigtable

DynamoDB

accumulo™

SQLite

riak

QuasarDB

influxdb

Why **LSM** ?

fast writes

competitive reads

good space utilization

fast writes

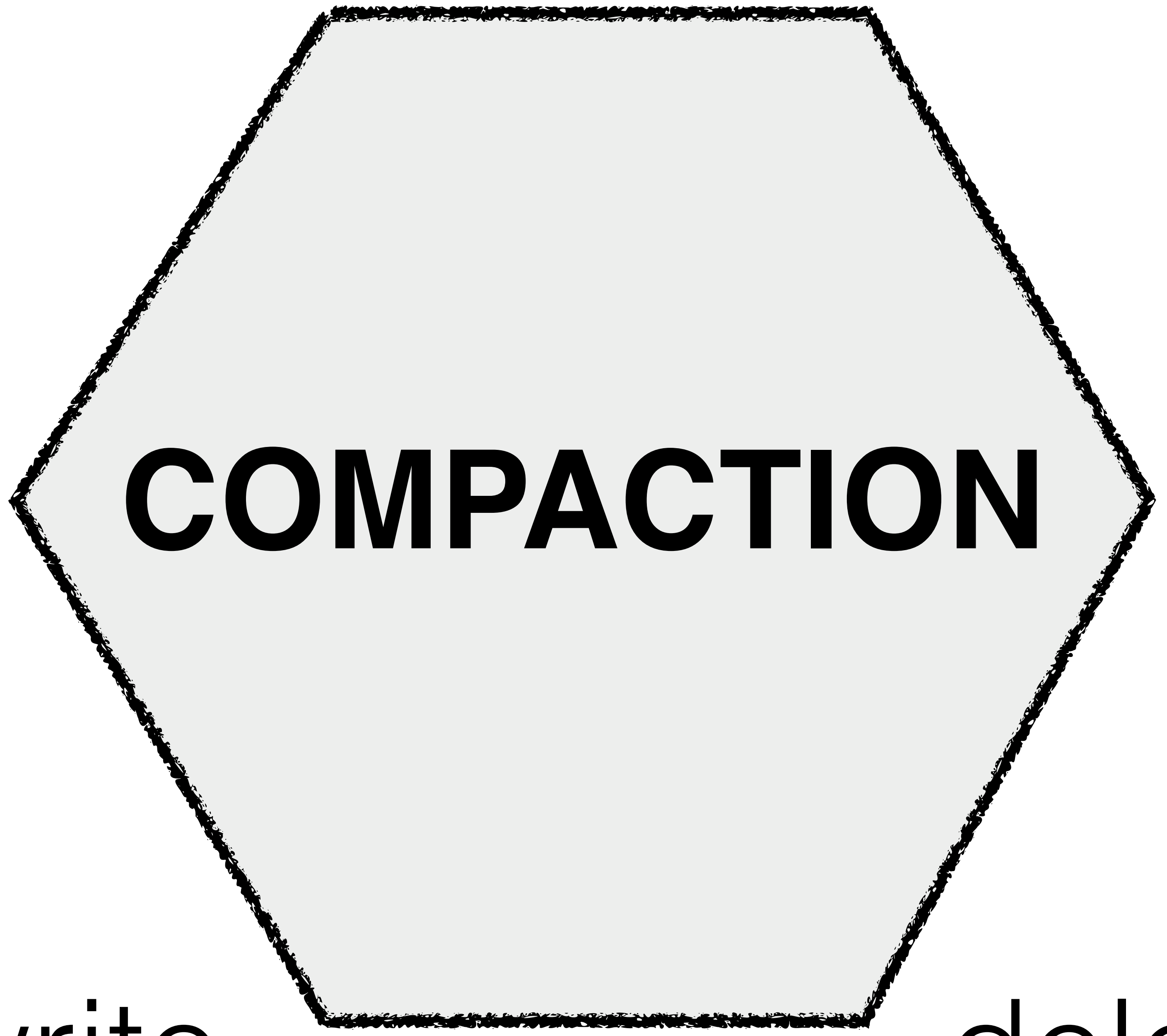
competitive reads

good space utilization

COMPACTION

space amplification

write performance



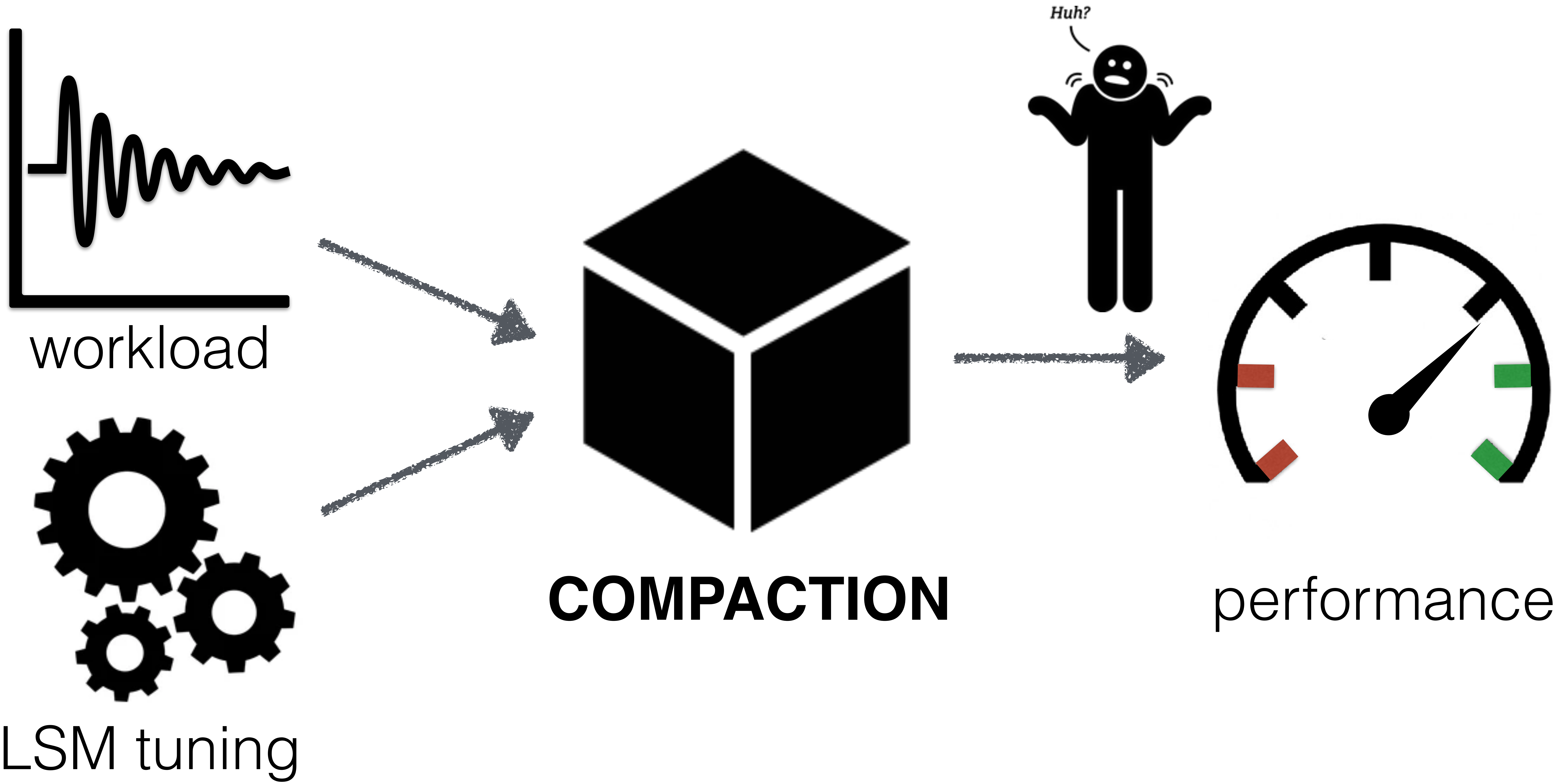
lookup performance

COMPACTION

scan performance

write amplification

delete performance



workload

LSM tuning

COMPACTION

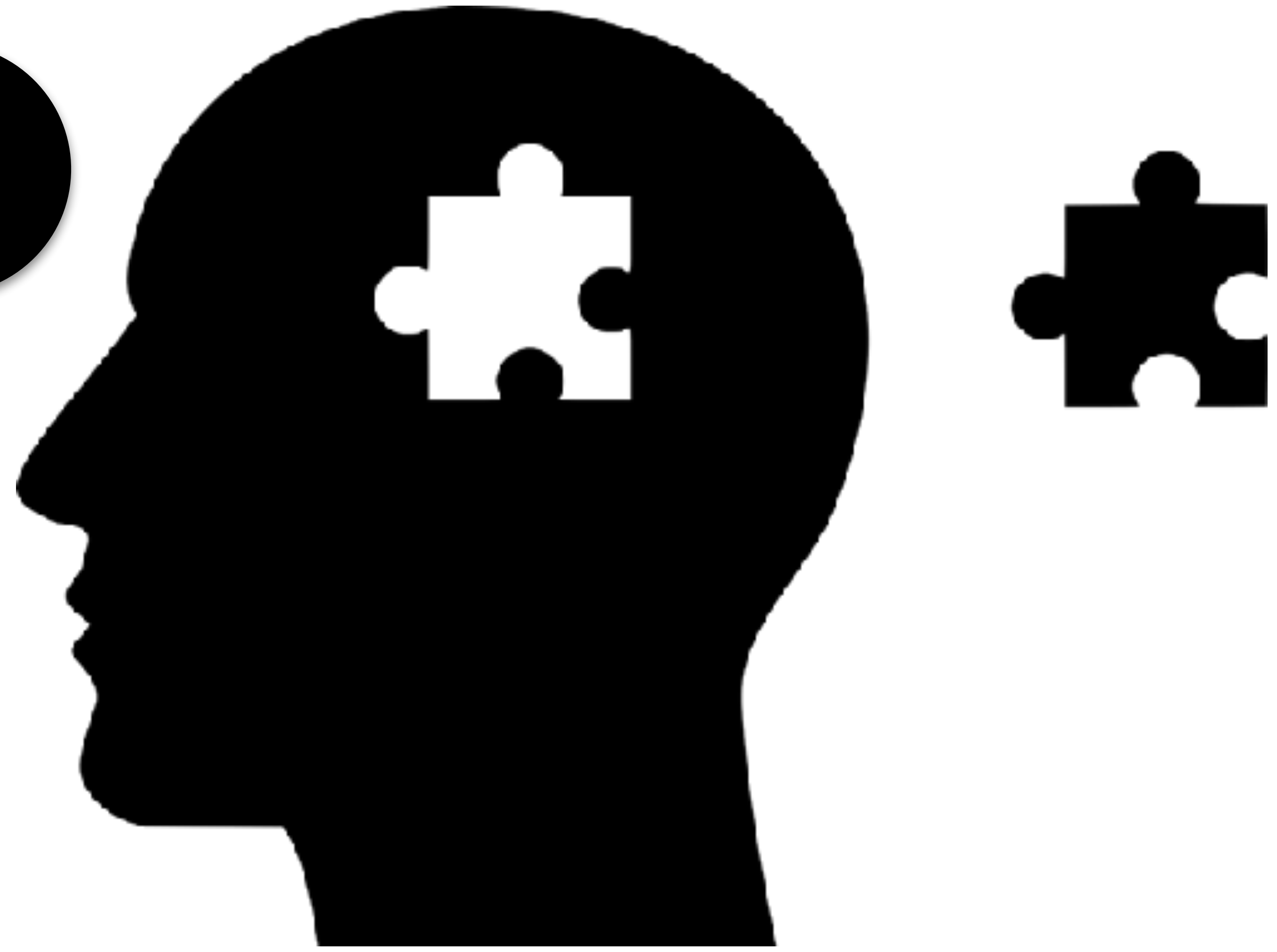
performance

Our **Goal**

❶

Roadmap to pick compactions

❷



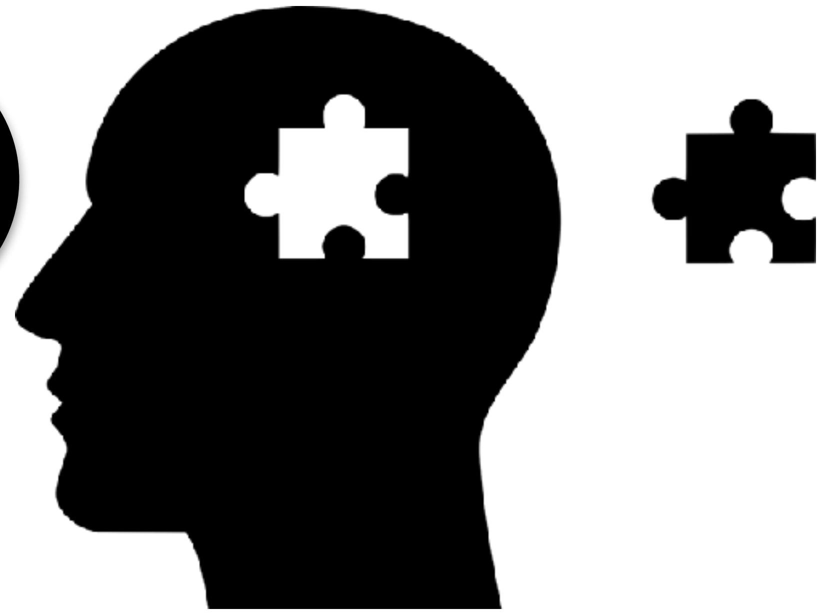
Answer to complex design questions

Our **Goal**

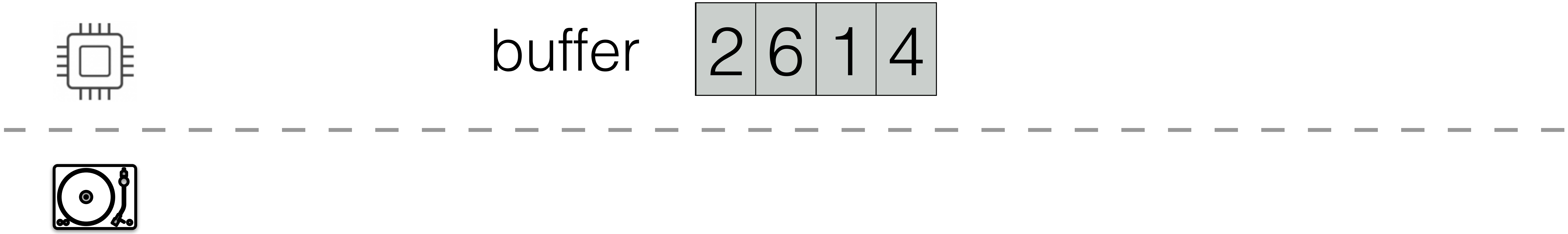
❶

break the black box

❷

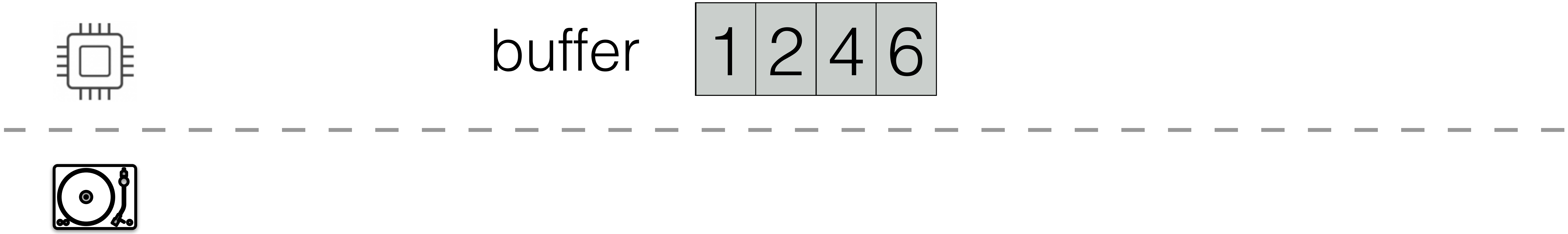


learn from 2000+ experiments



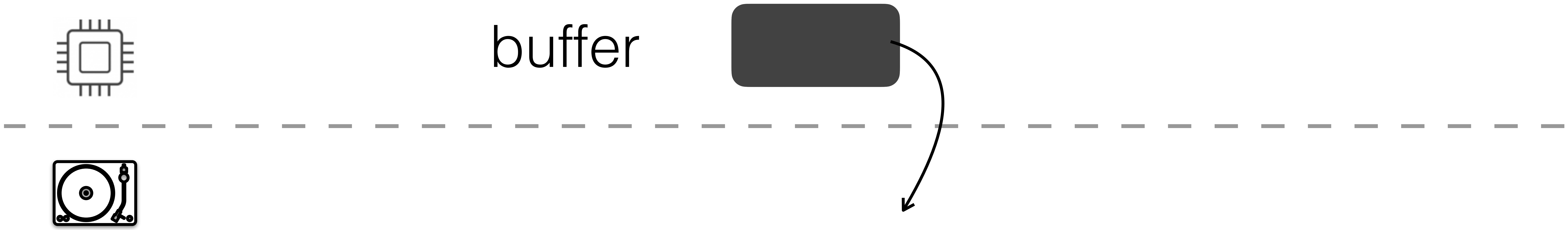
buffer

2	6	1	4

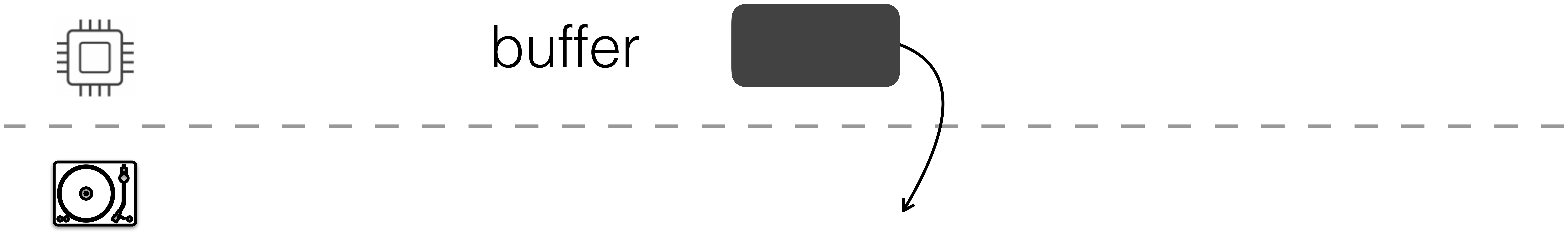


buffer

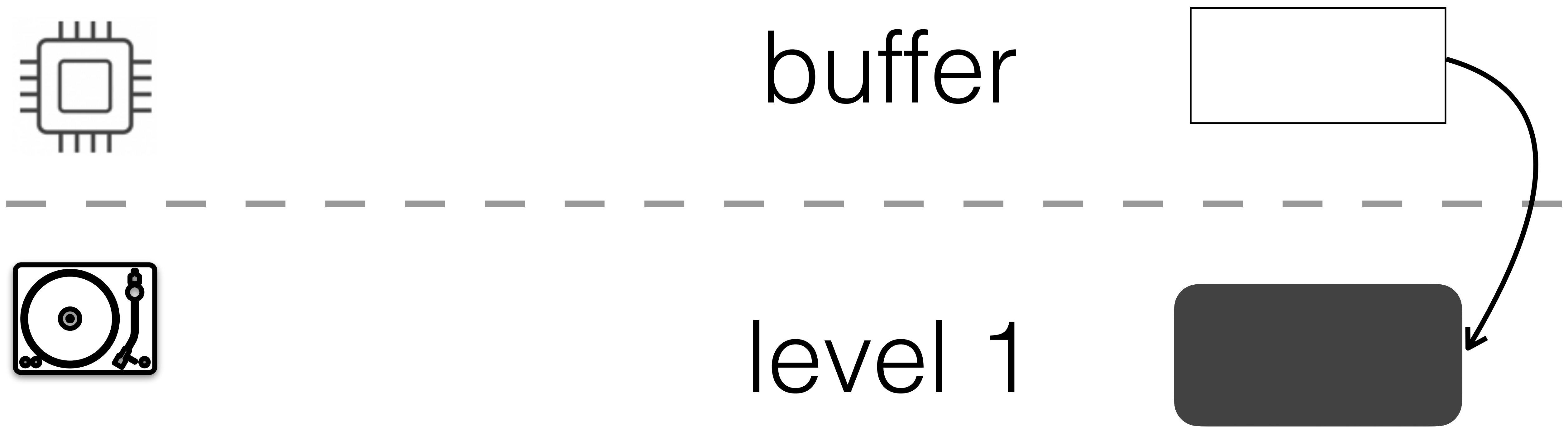
1	2	4	6



buffer

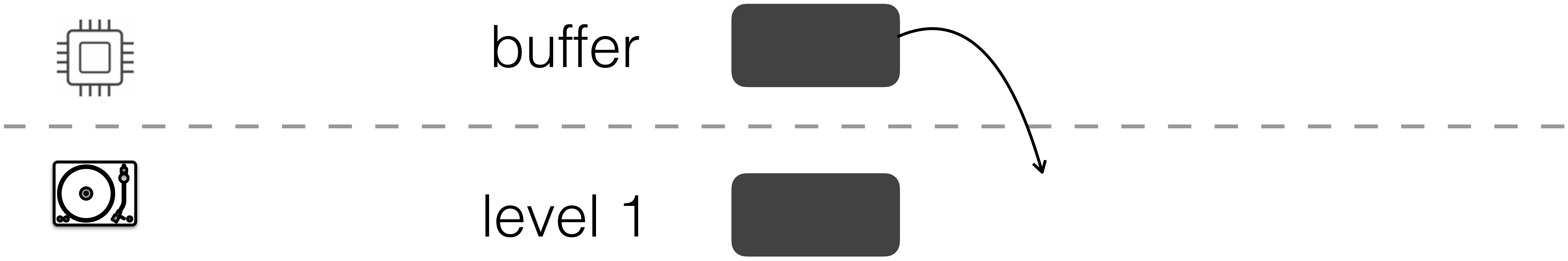


buffer



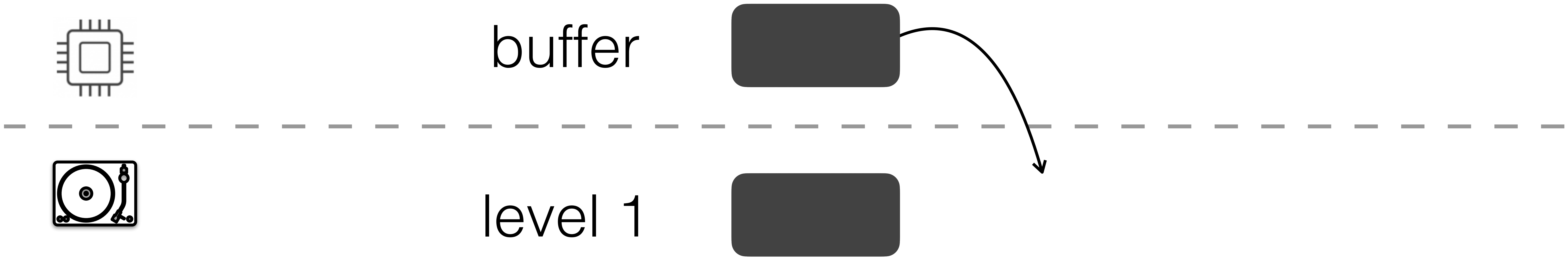
buffer

level 1

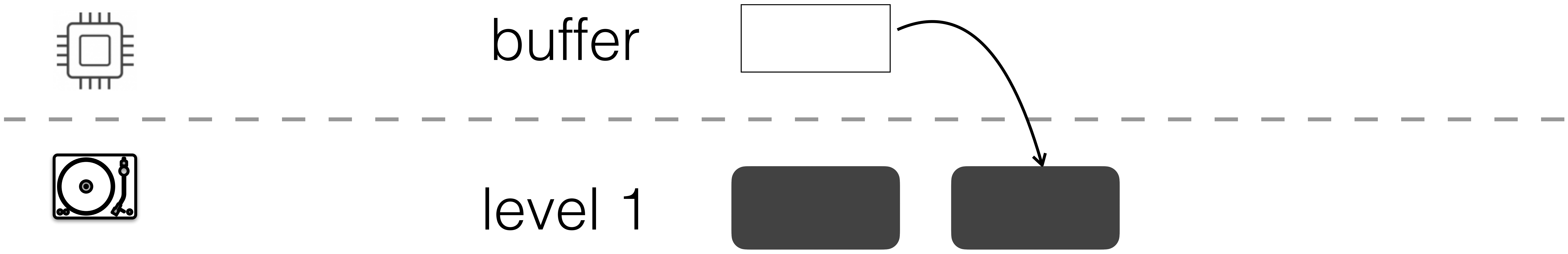


buffer

level 1

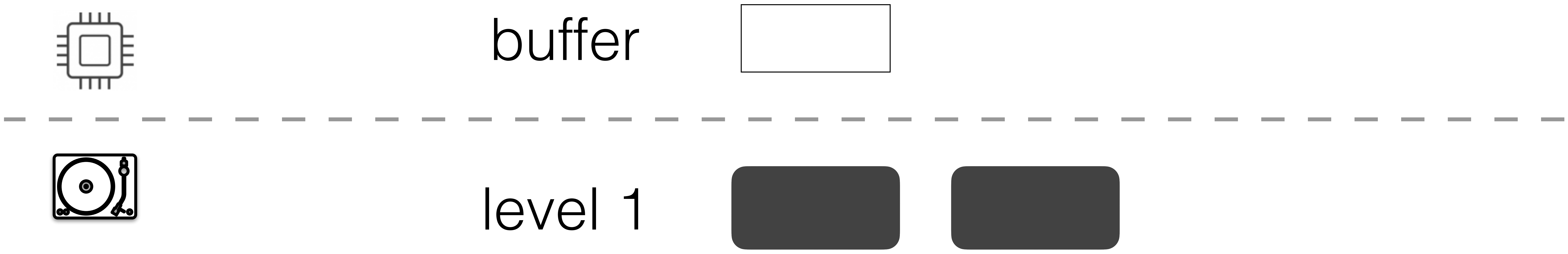


buffer
level 1



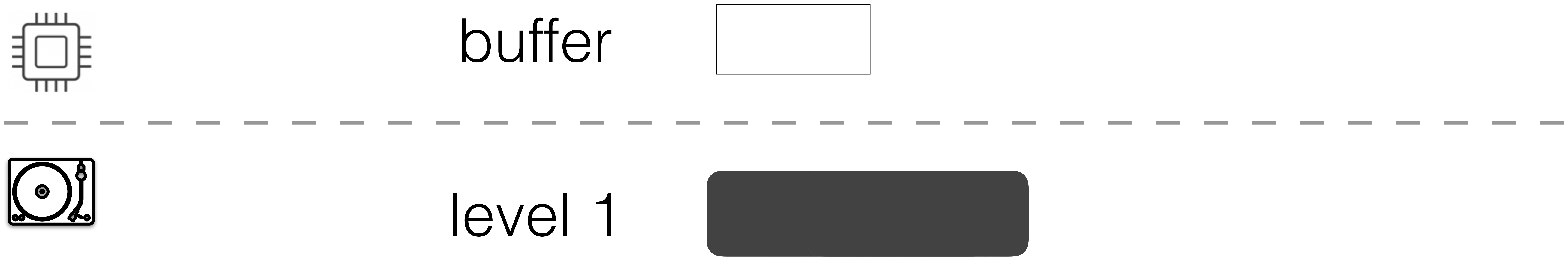
buffer

level 1



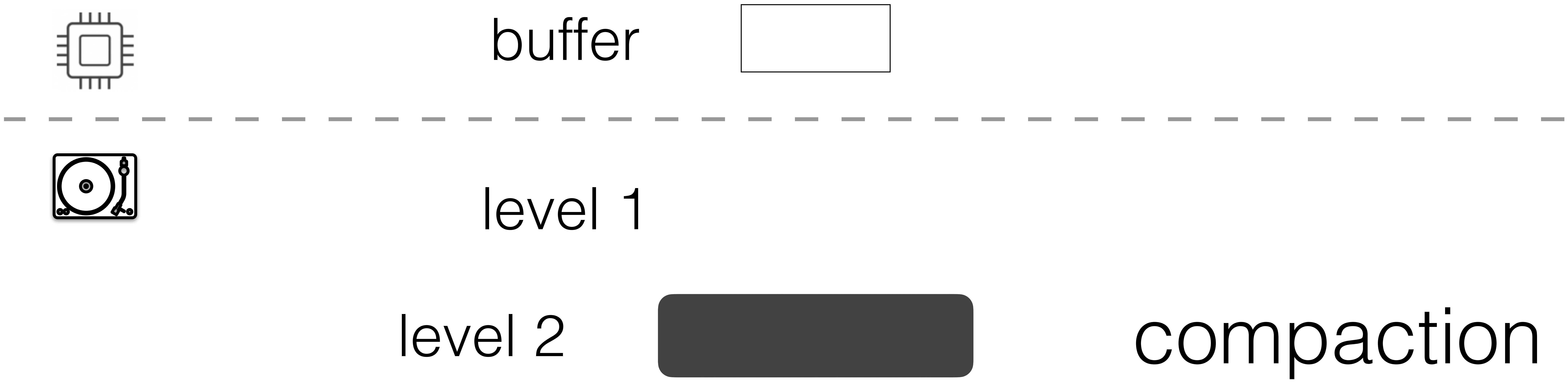
buffer

level 1



buffer

level 1

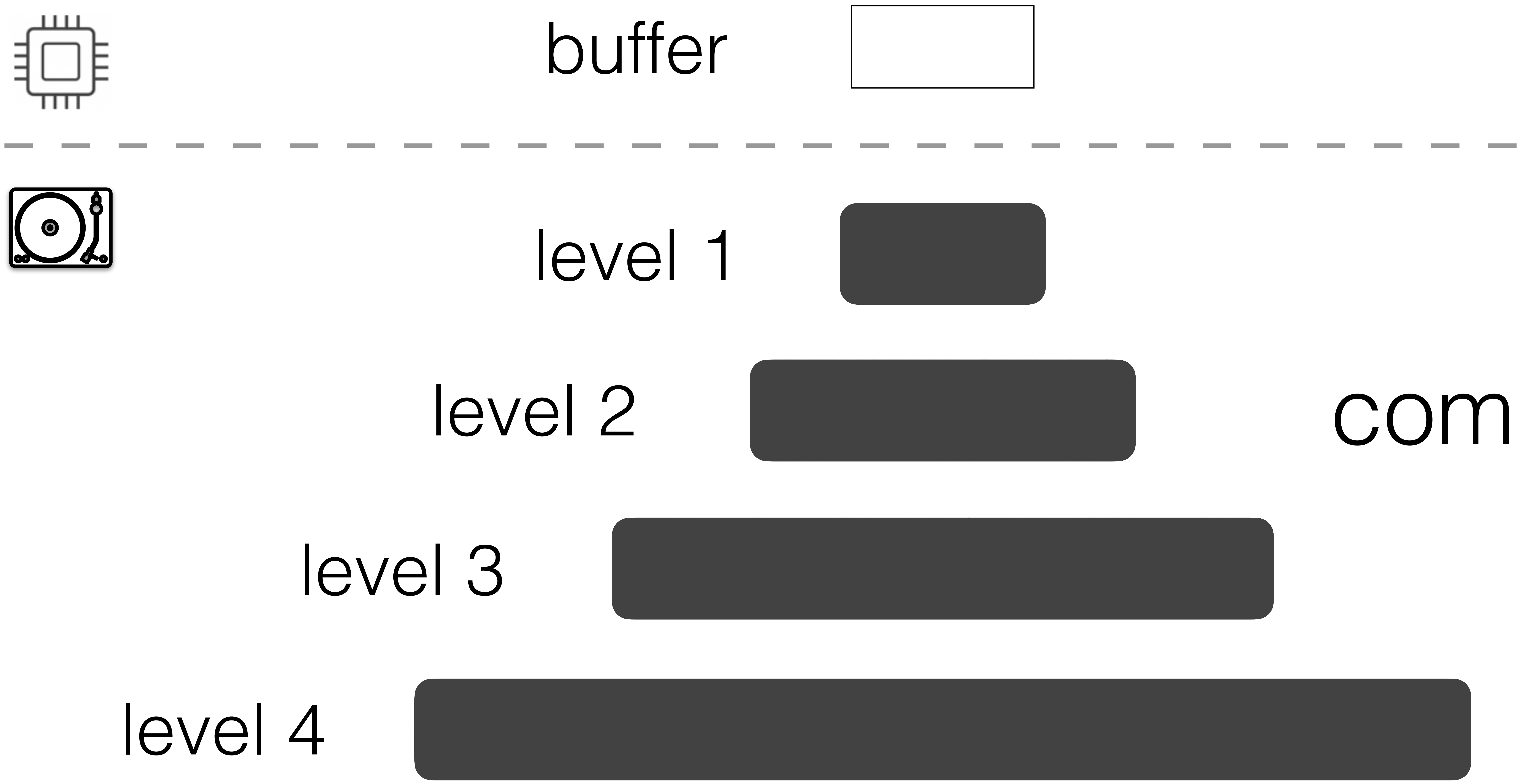


buffer

level 1

level 2

compaction



buffer

level 1

level 2

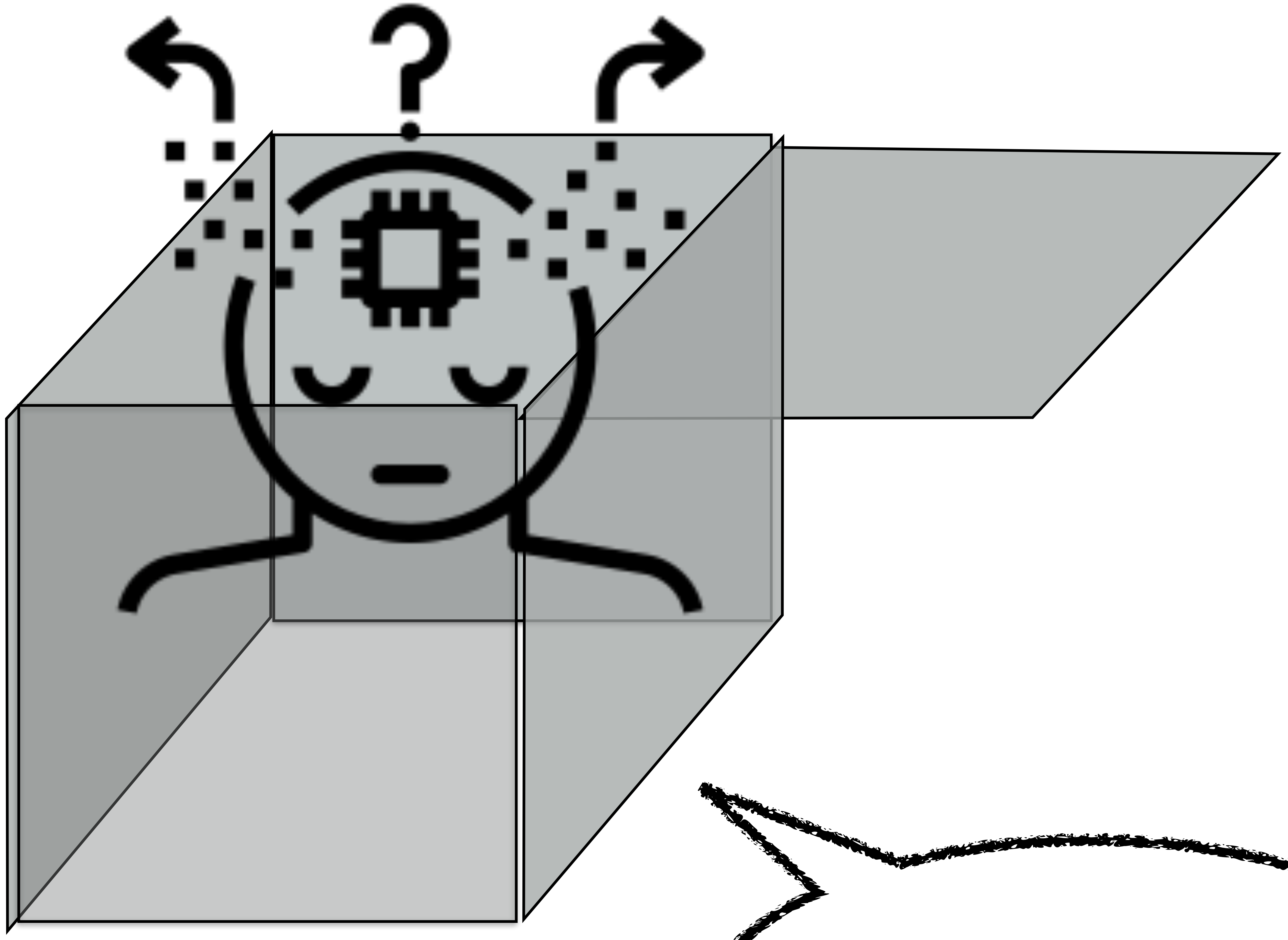
level 3

level 4

compaction

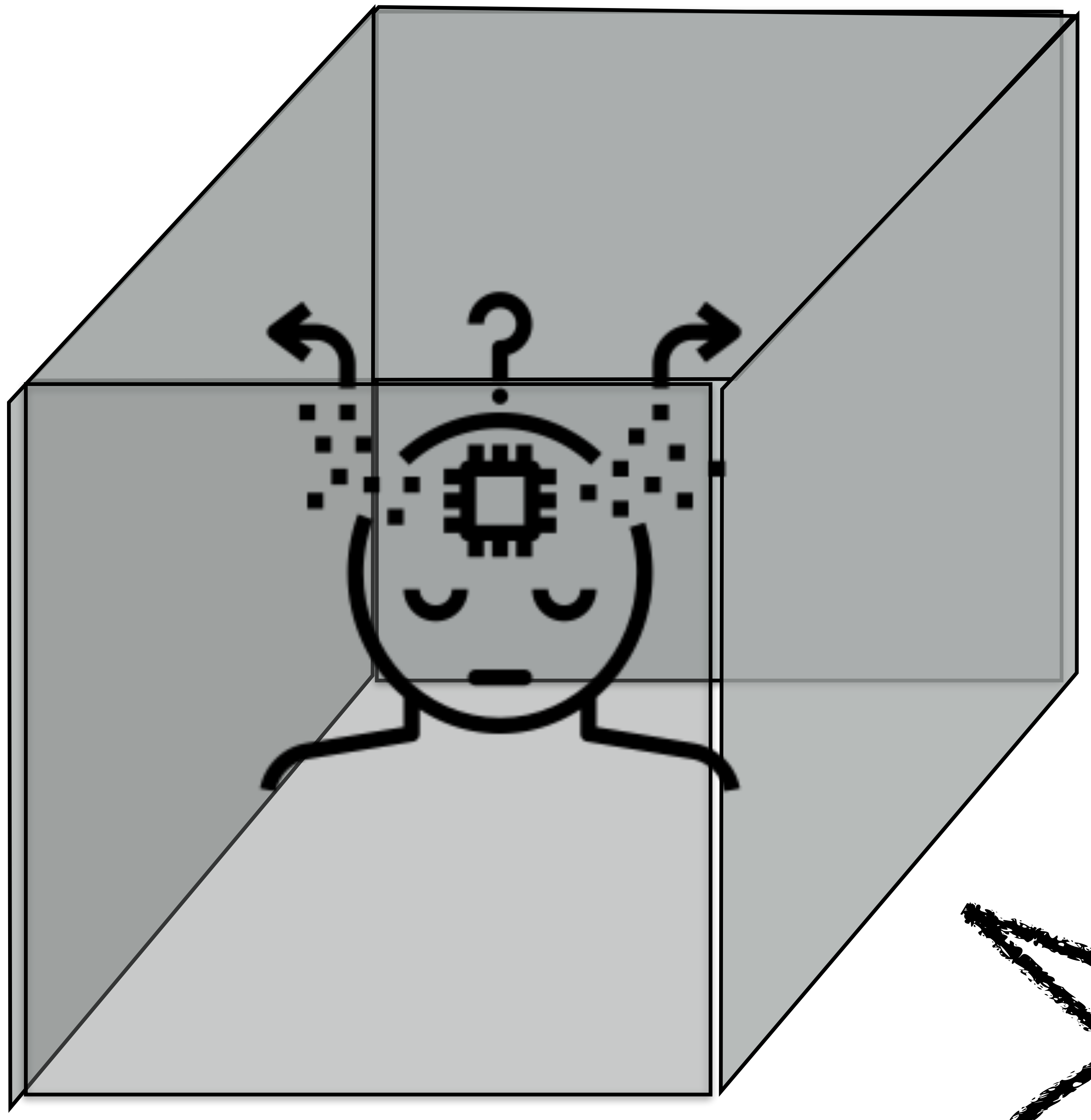
compaction?

How many runs per level?

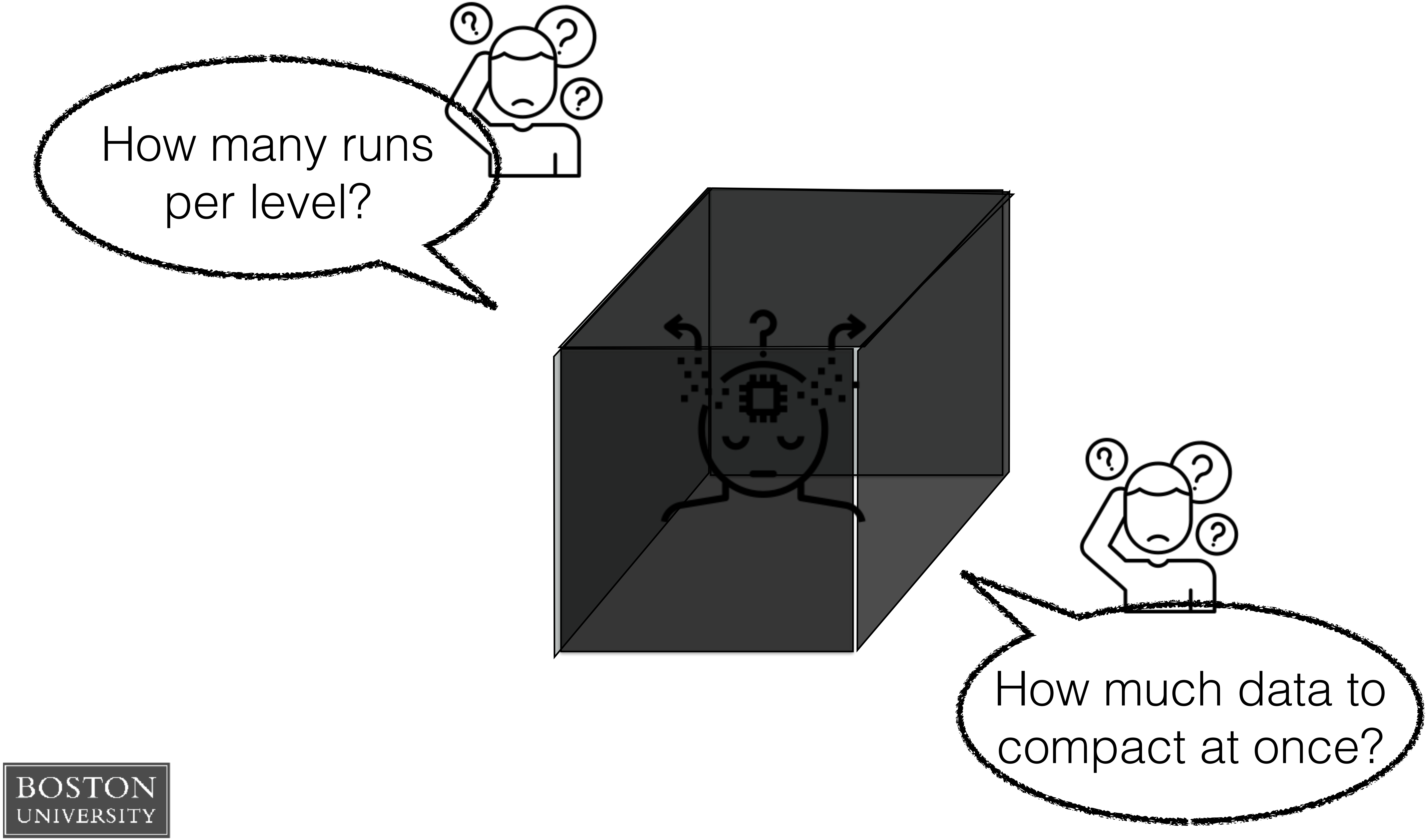


How much data to compact at once?

How many runs per level?



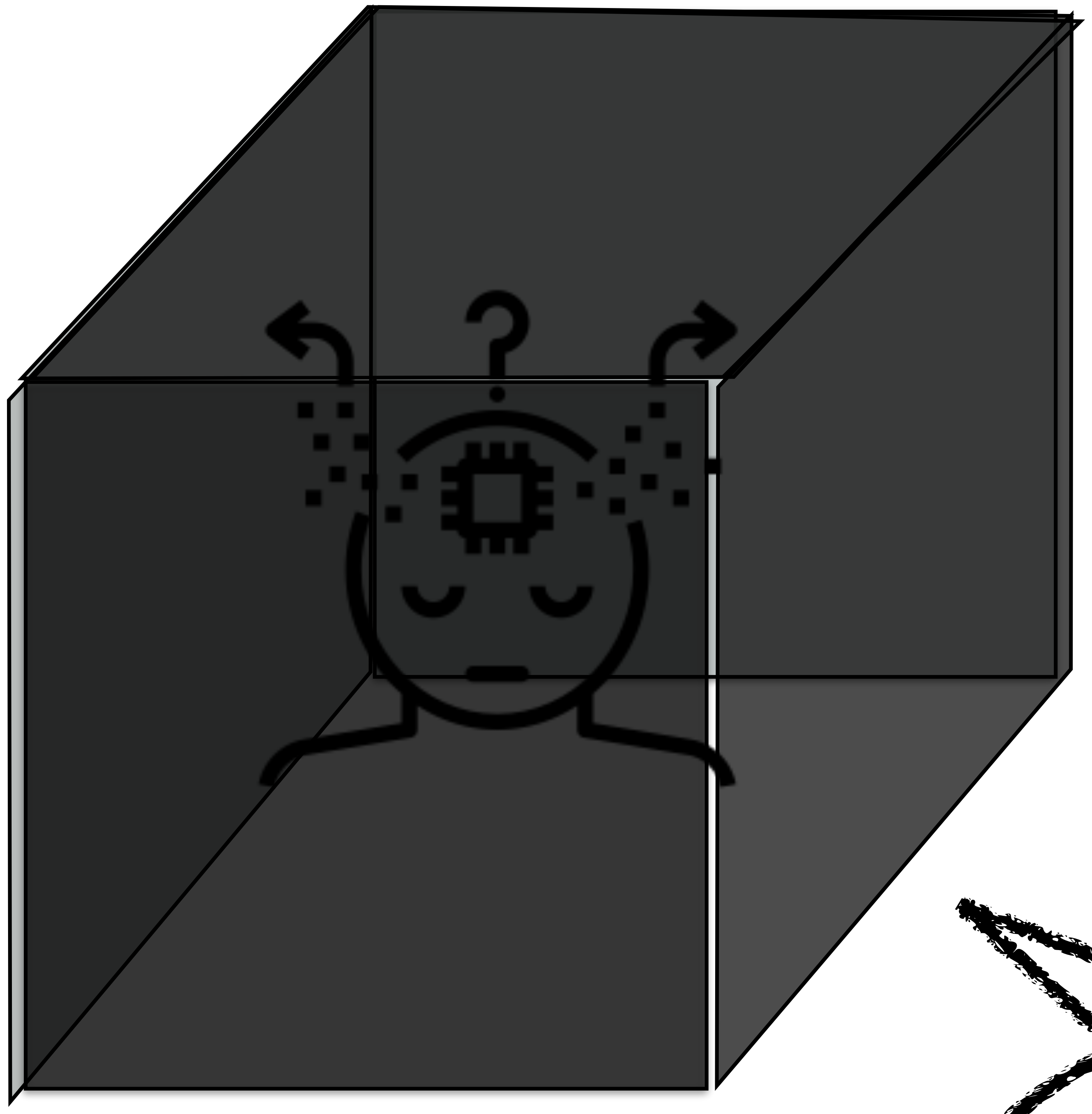
How much data to compact at once?



How many runs per level?

How much data to compact at once?

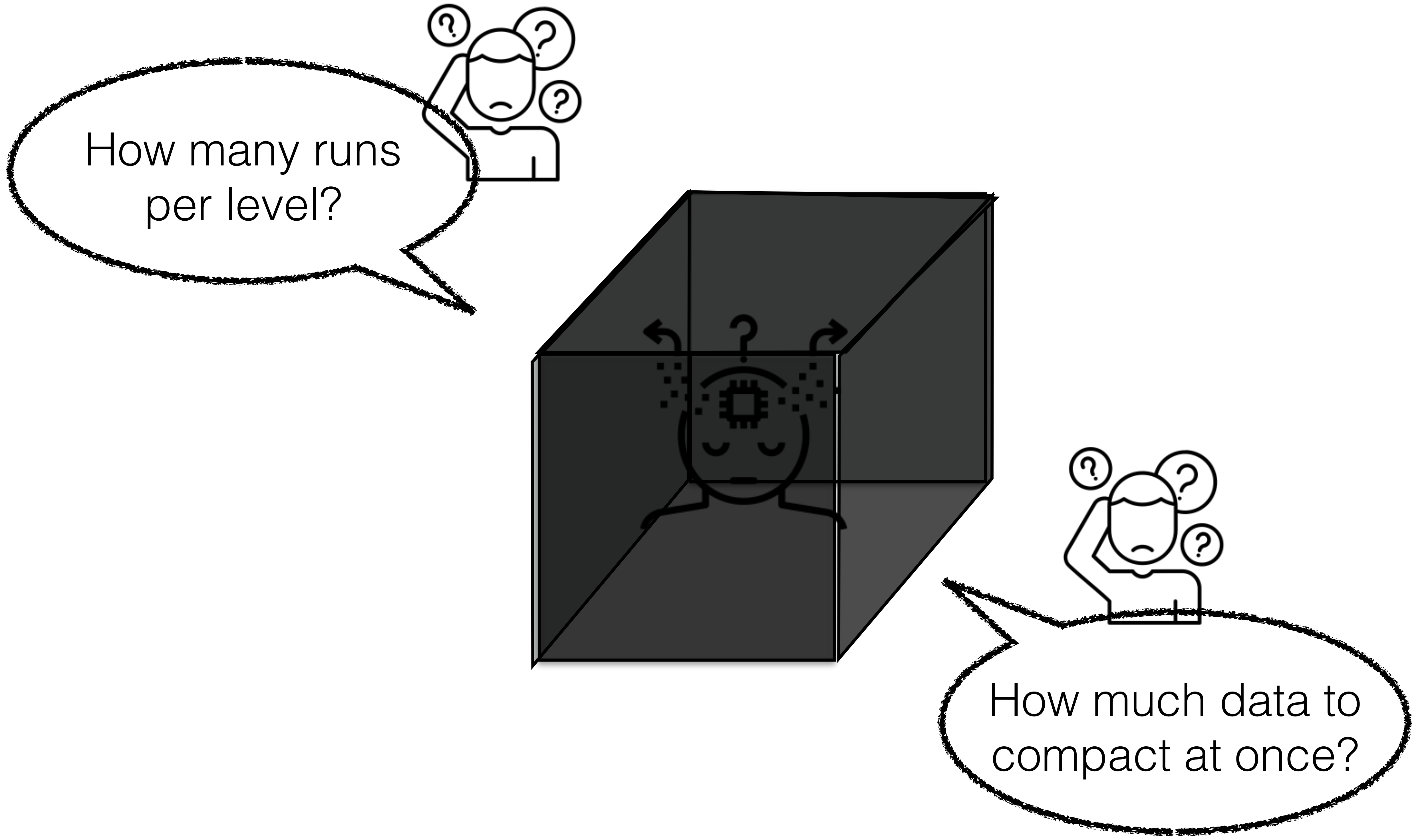
How many runs per level?



How much data to compact at once?

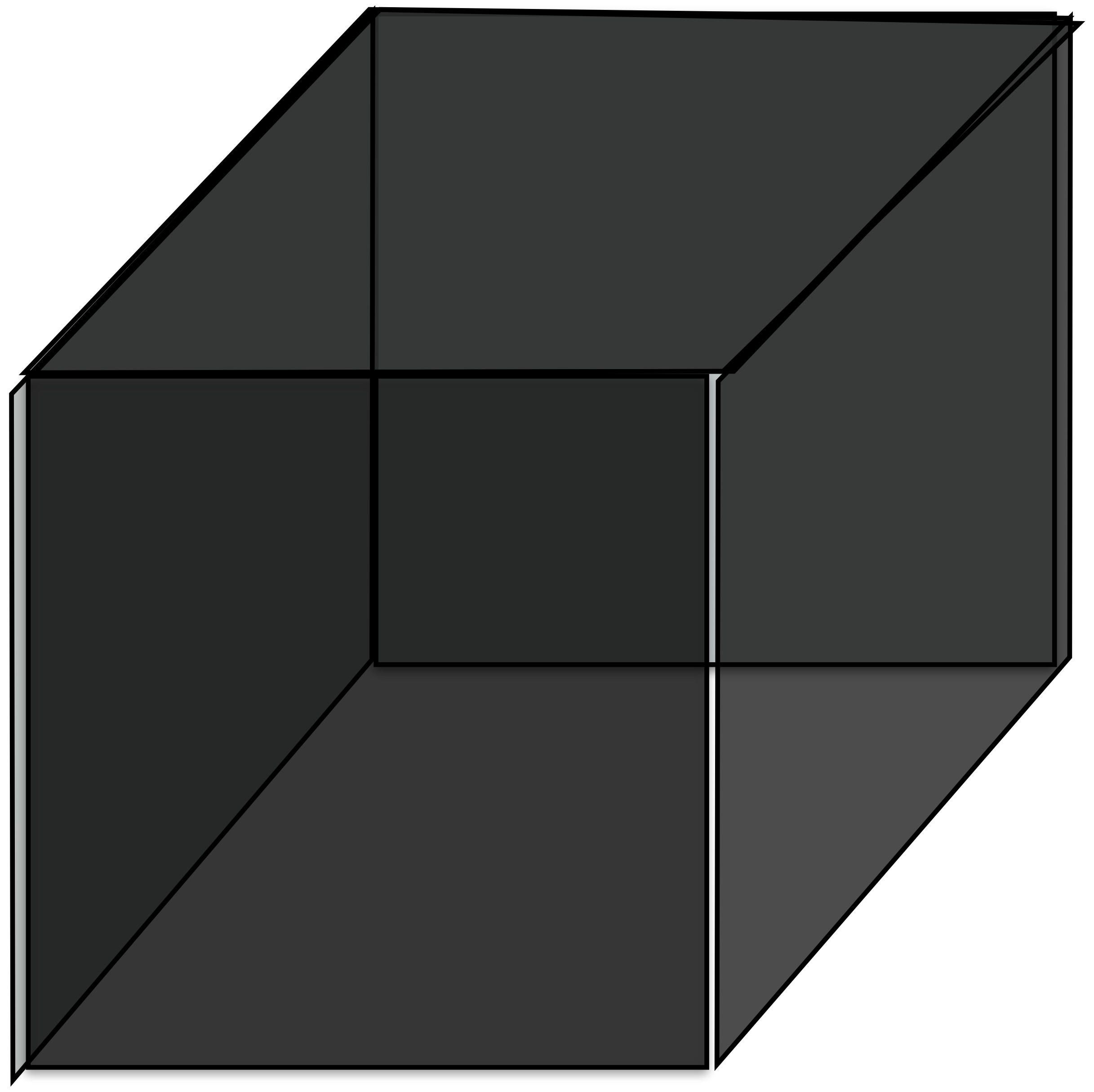
What are the **design choices**?

How does a choice **affect performance**?



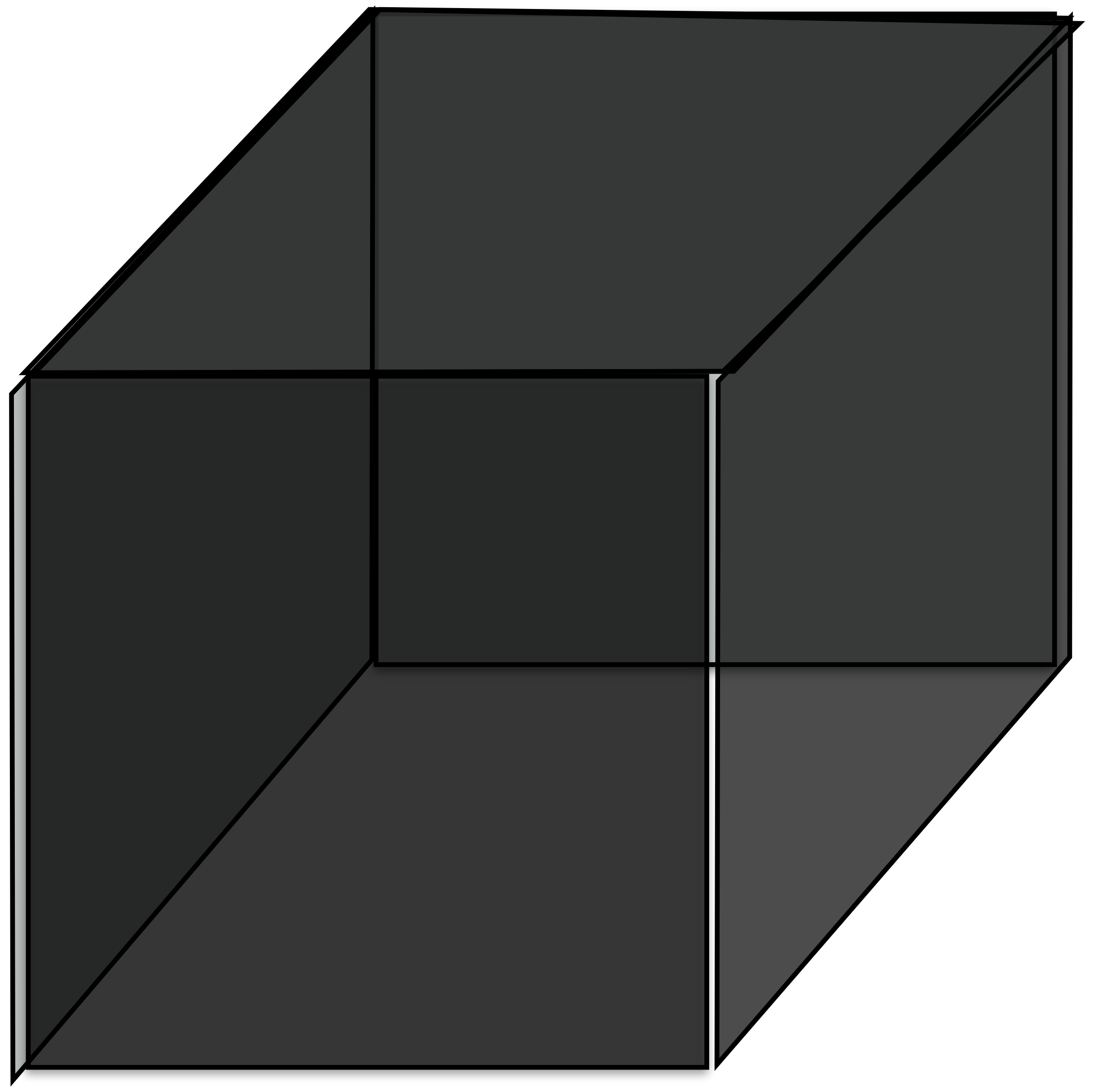
What are the **design choices**?

How does a choice
affect performance?



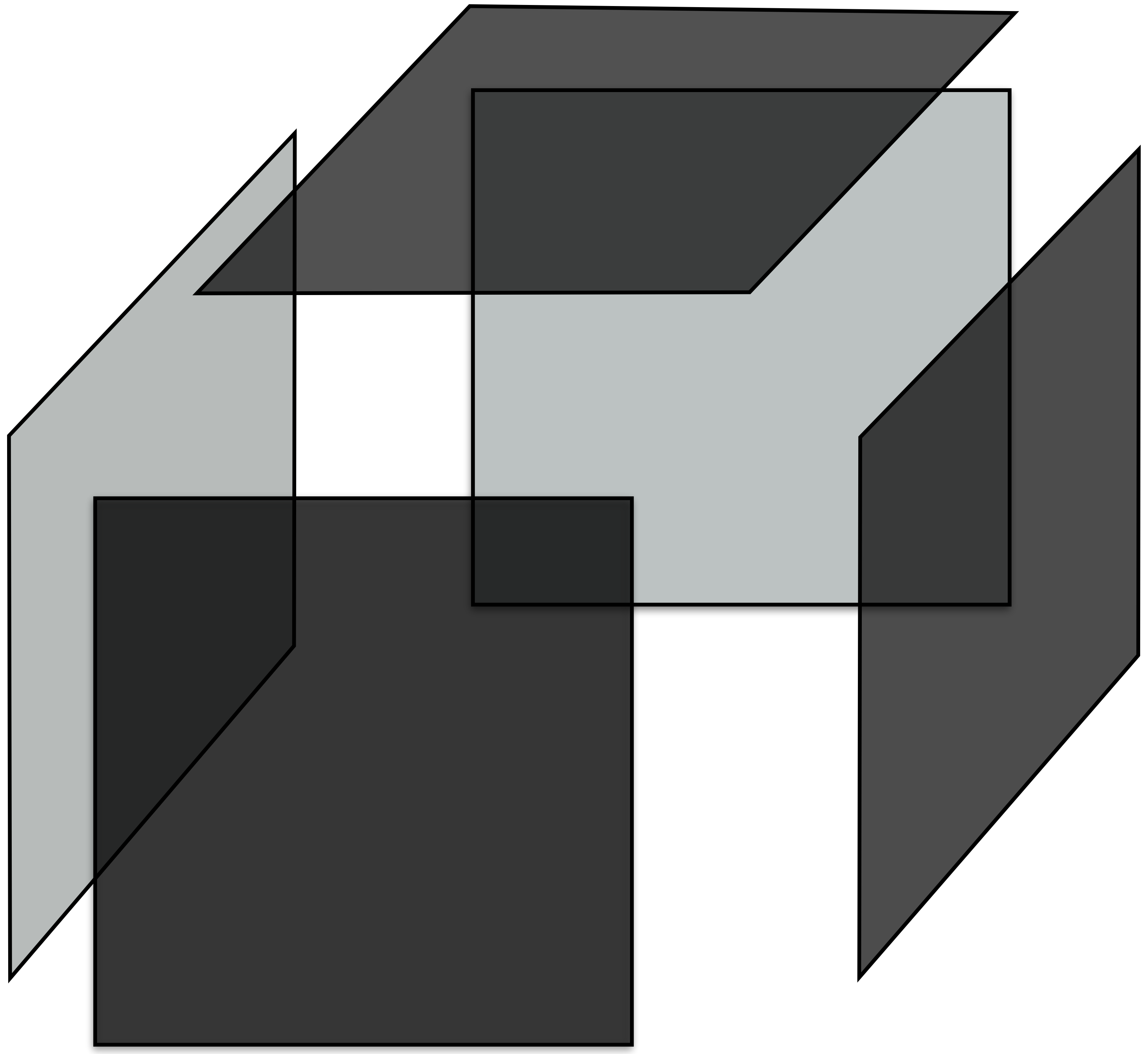
What are the **design choices**?

How does a choice
affect performance?



What are the **design choices**?

How does a choice
affect performance?



1. **How** to organize the data on device?

2. **How much** data to move at-a-time?

3. **Which** block of data to be moved?

4. **When** to re-organize the data layout?

Data Layout	1	**How** to organize the data on device?
Compaction granularity	2	**How much** data to move at-a-time?
Data movement policy	3	**Which** block of data to be moved?
Trigger	4	**When** to re-organize the data layout?

1 Data Layout

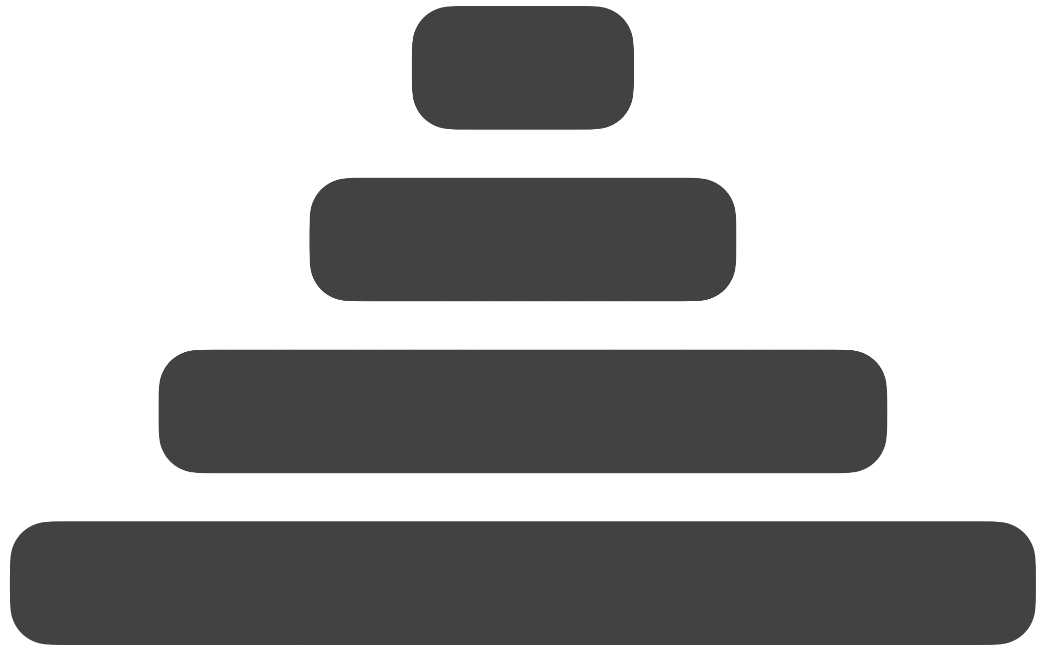
number of runs per level

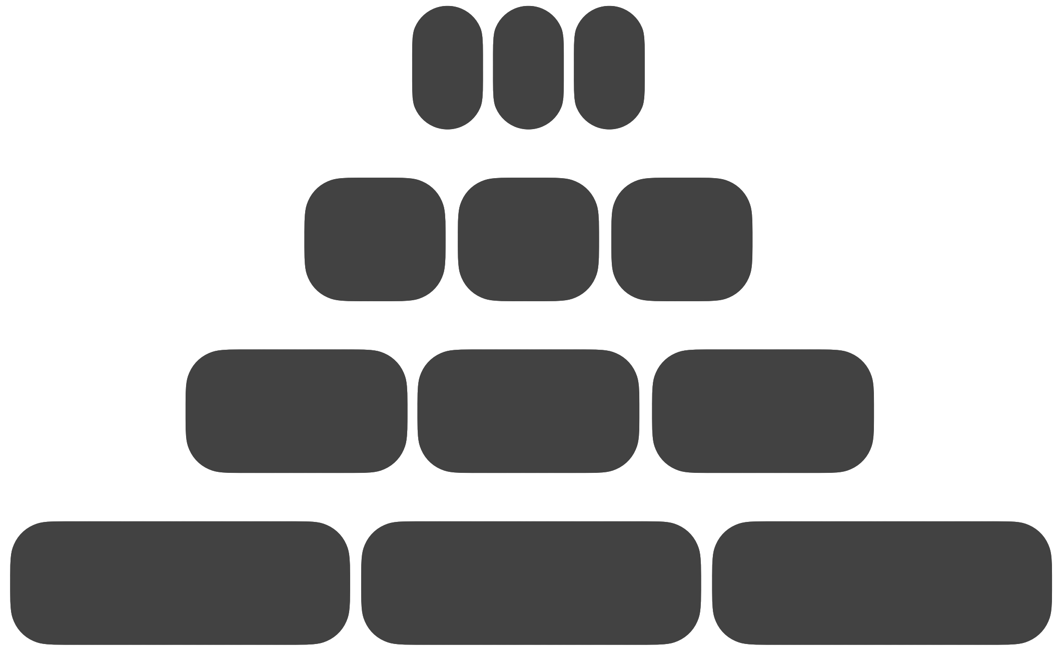
Data Layout

number of runs per level

leveling

[eager]



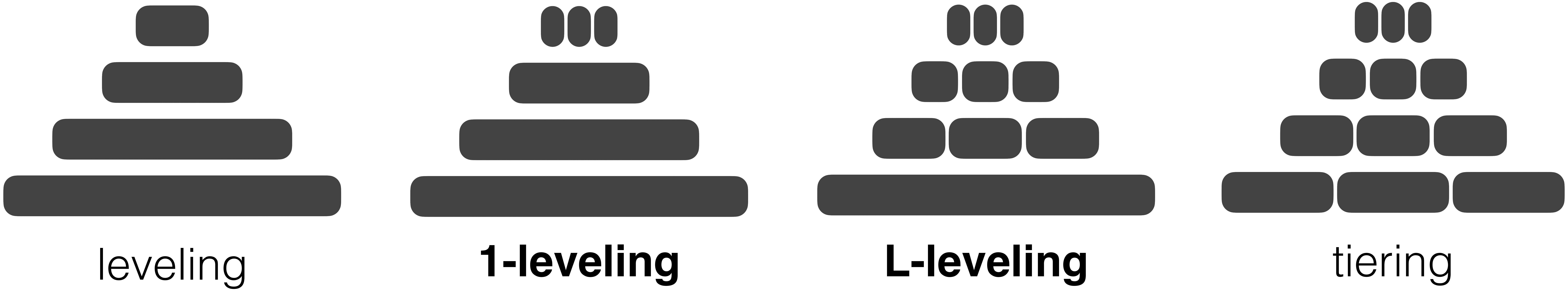


tiering

[lazy]

Data Layout

number of runs per level

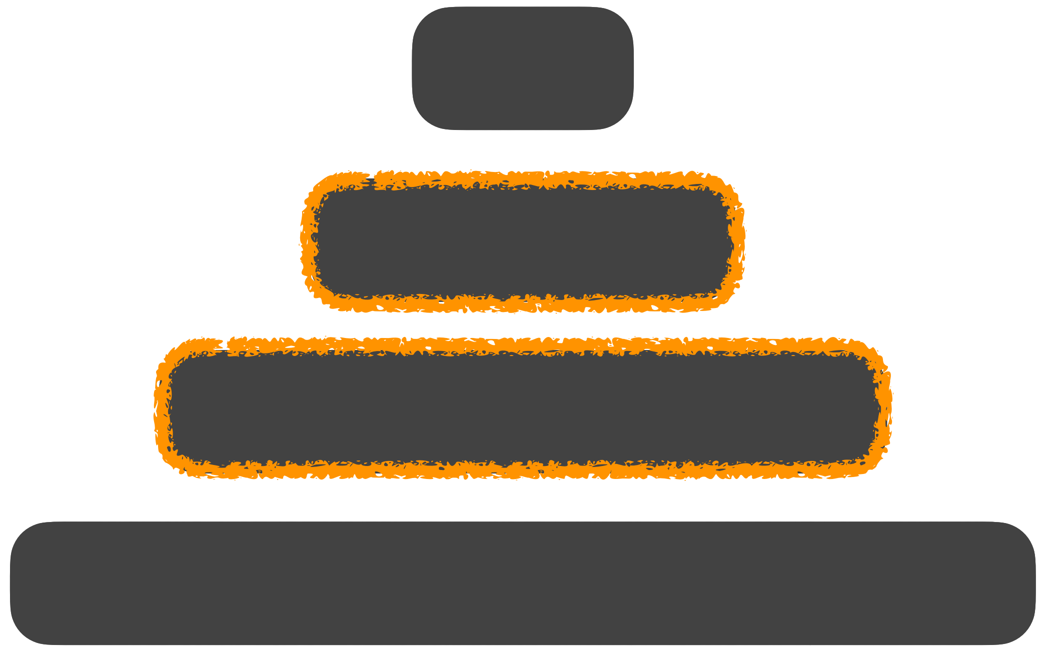


2 Compaction Granularity

data moved per compaction

2 Compaction Granularity

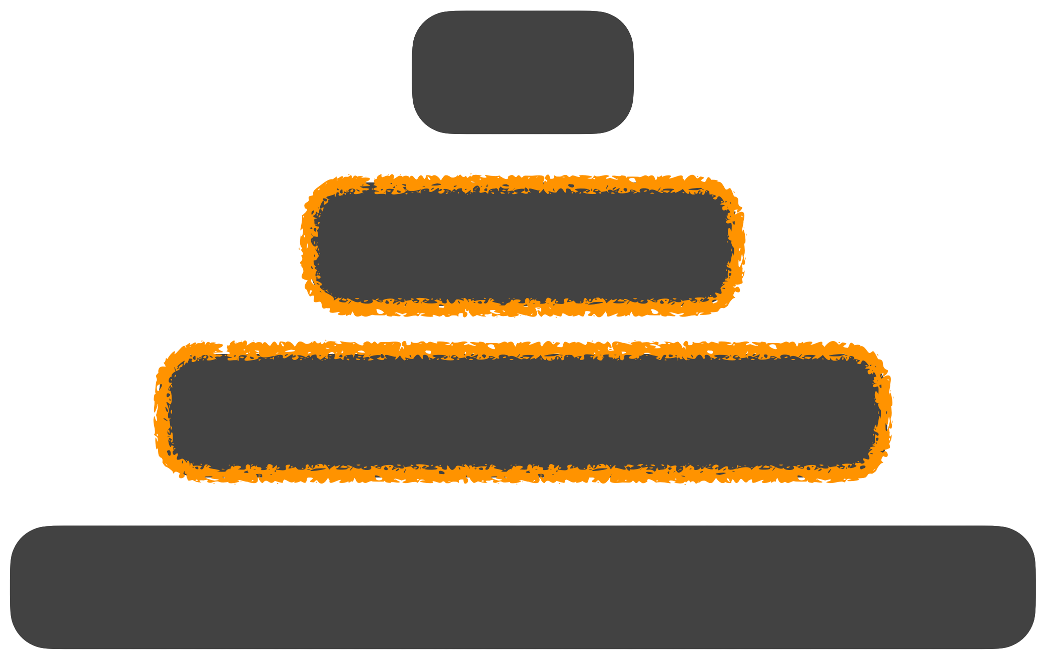
data moved per compaction



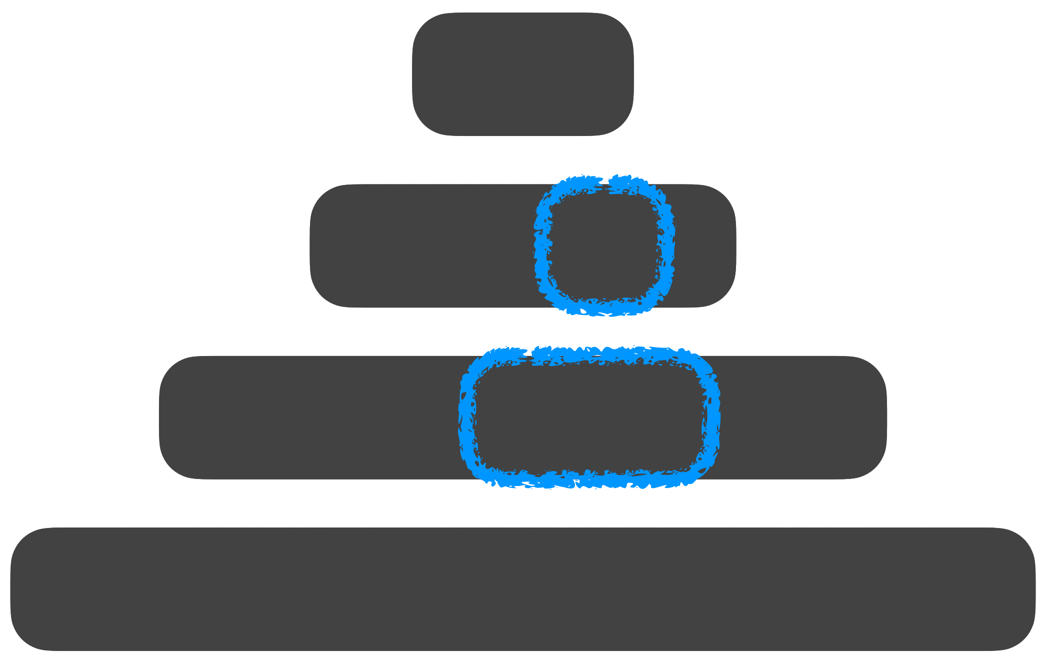
levels

2 Compaction Granularity

data moved per compaction



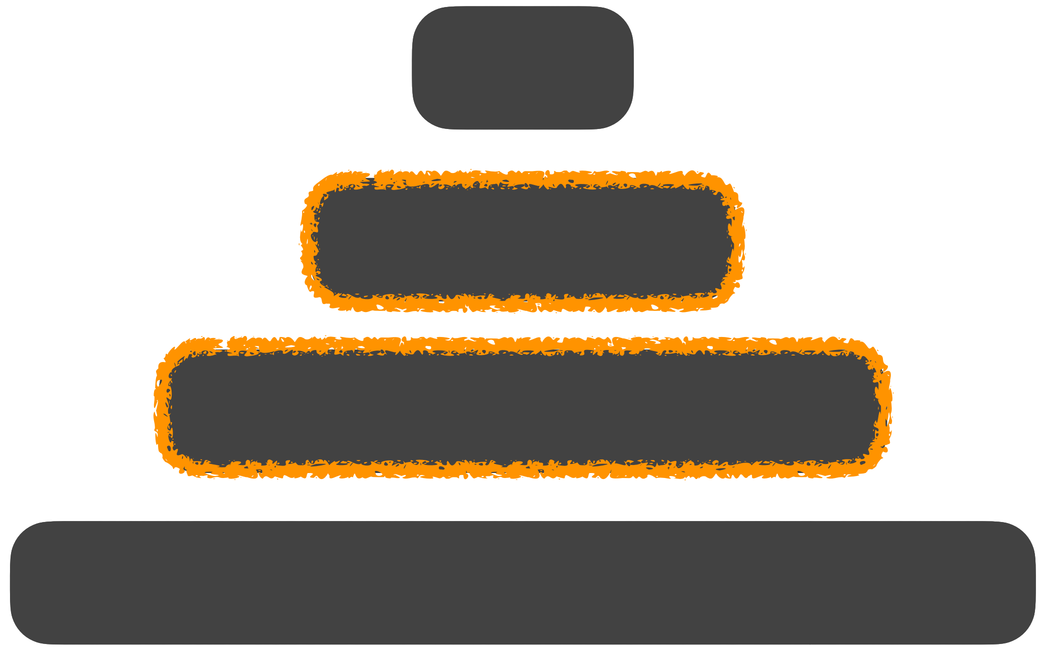
levels



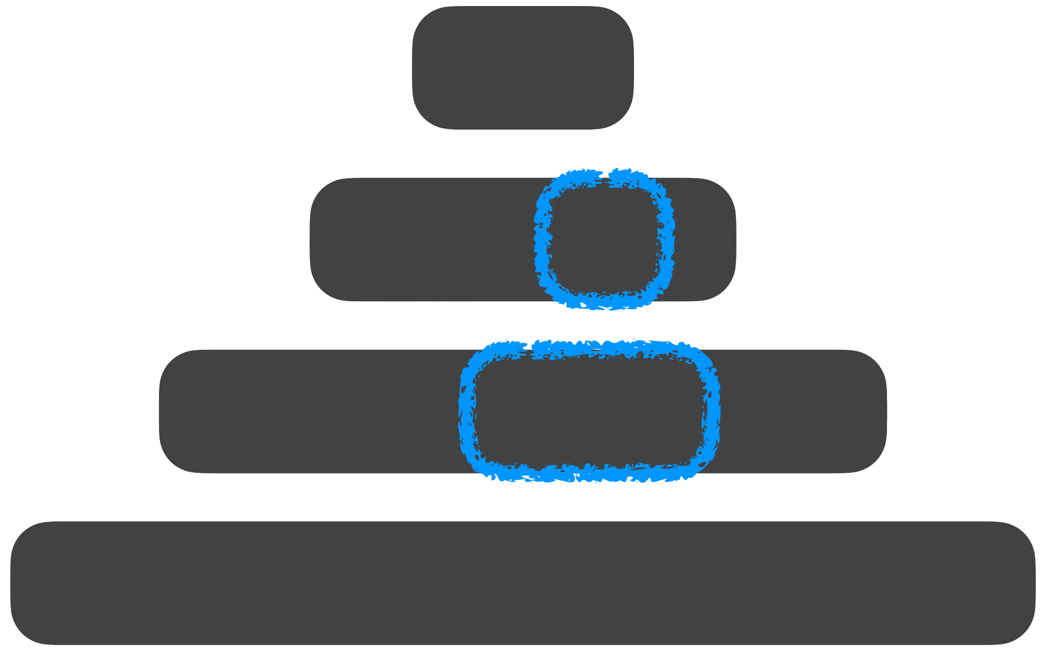
files

2 Compaction Granularity

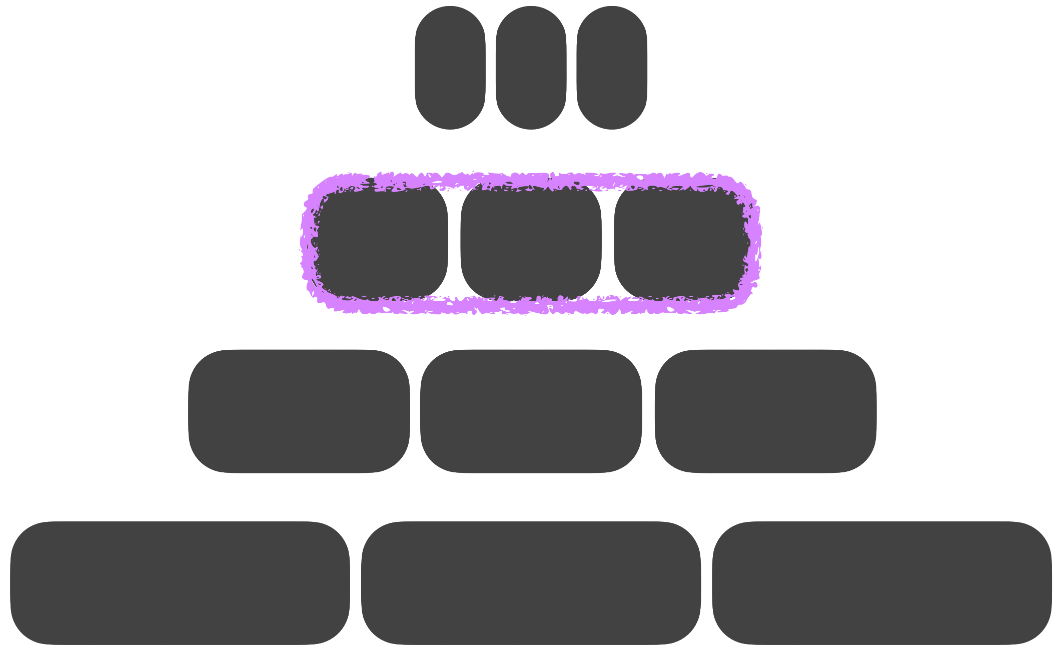
data moved per compaction



levels



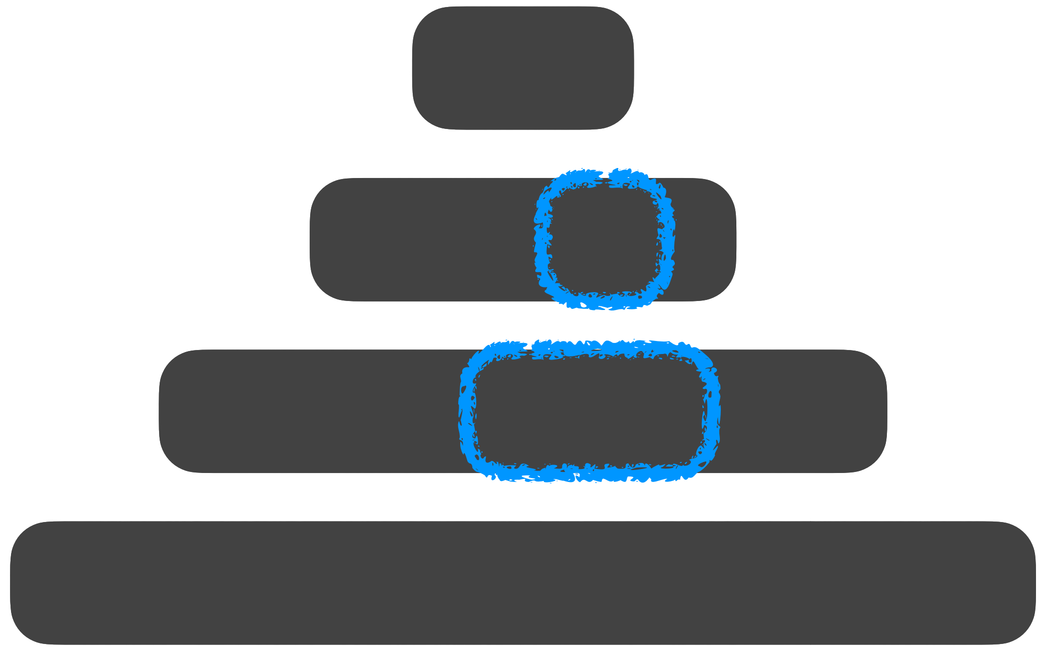
files



sorted runs in a level

3 Data Movement Policy

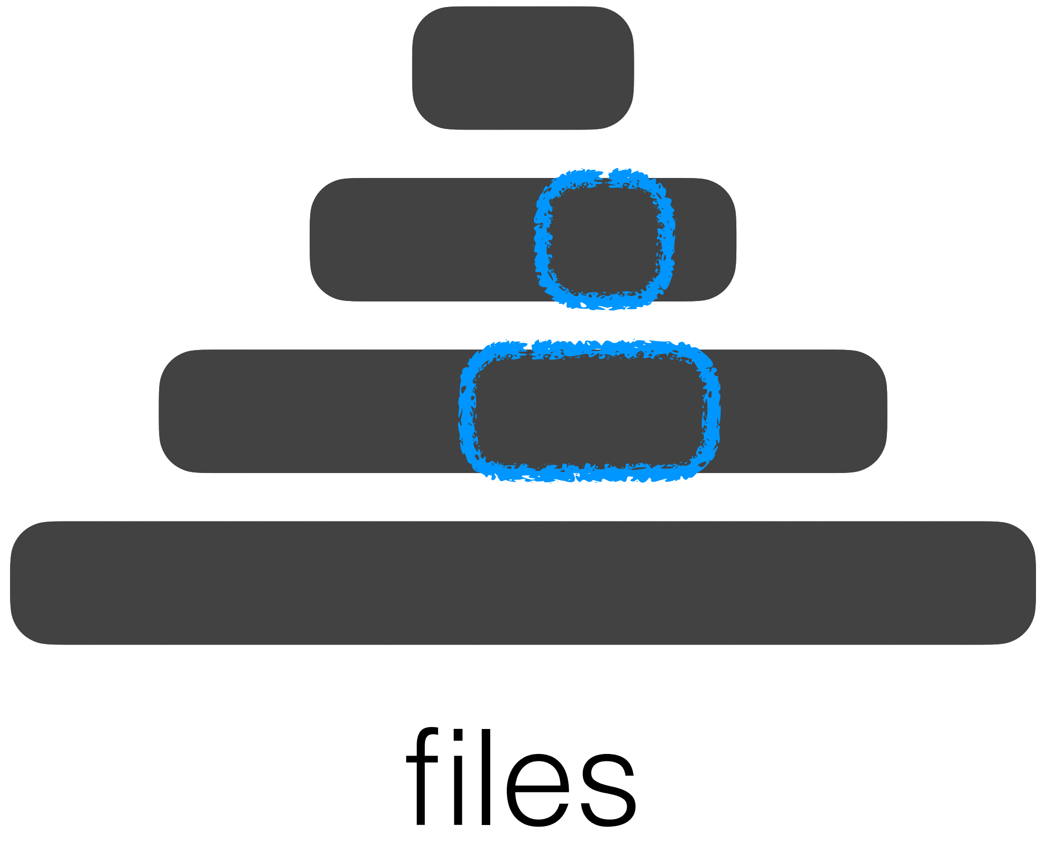
which data to compact



files

3 Data Movement Policy

which data to compact



round-robin

minimum **overlap with parent** level

file with most **tombstones**

coldest file

4 Compaction Trigger

invoking the compaction routine

level **saturation**

4 Compaction Trigger

invoking the compaction routine

level **saturation**

4 Compaction Trigger

invoking the compaction routine

level **saturation**

number of **sorted runs**

age of a file

space amplification

1 Data Layout

2 Compaction Granularity

3 Data Movement Policy

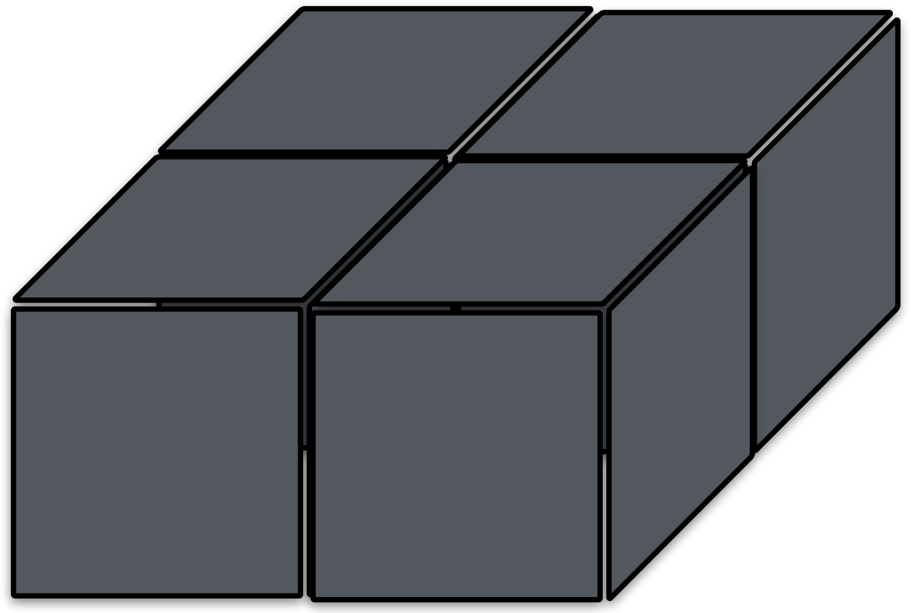
4 Compaction Trigger

Data Layout

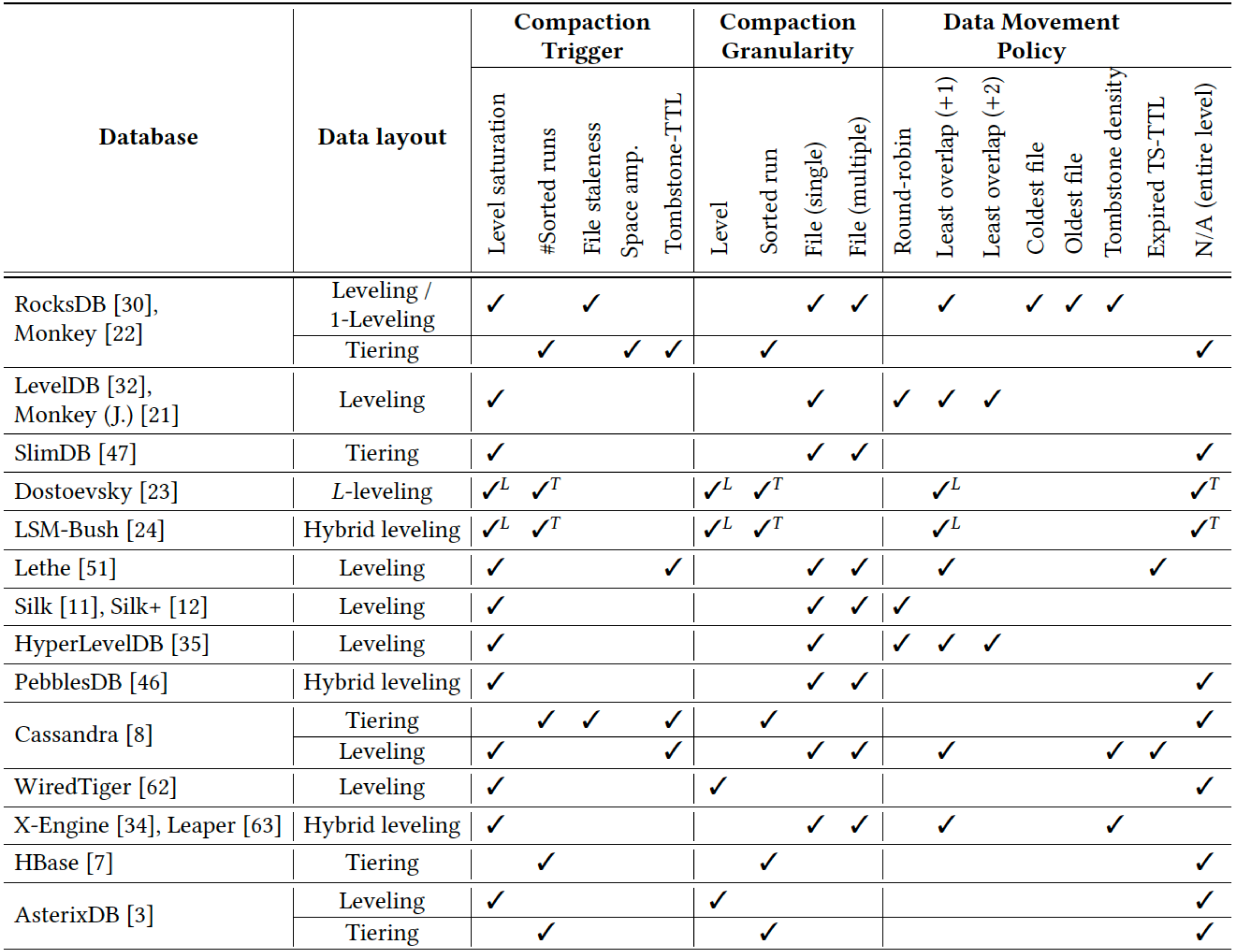
Compaction Granularity

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		✓					✓										✓

Blueprint for **Experiments**

10 compaction strategies

primitives

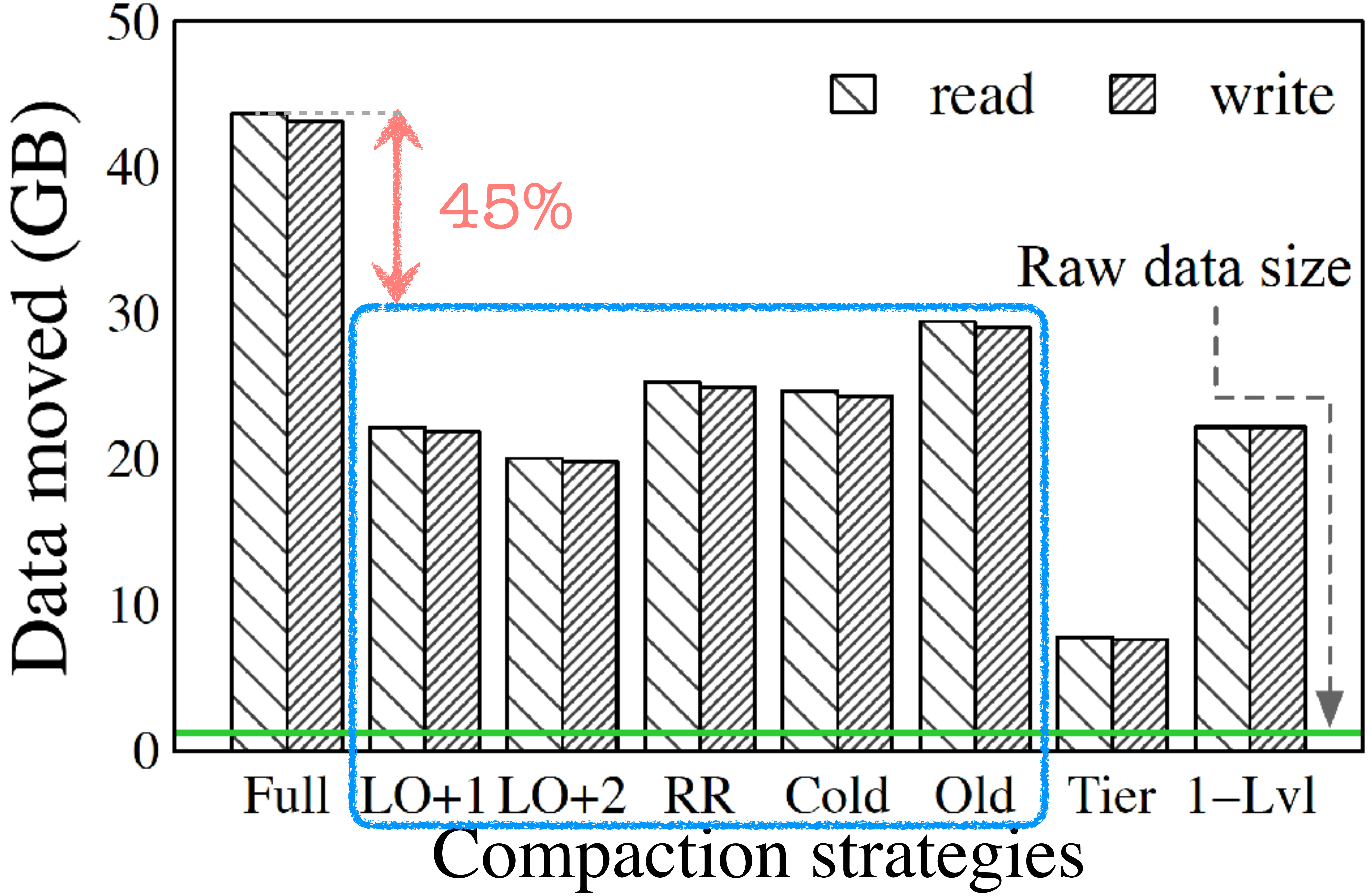
workloads

[distribution
+
composition]

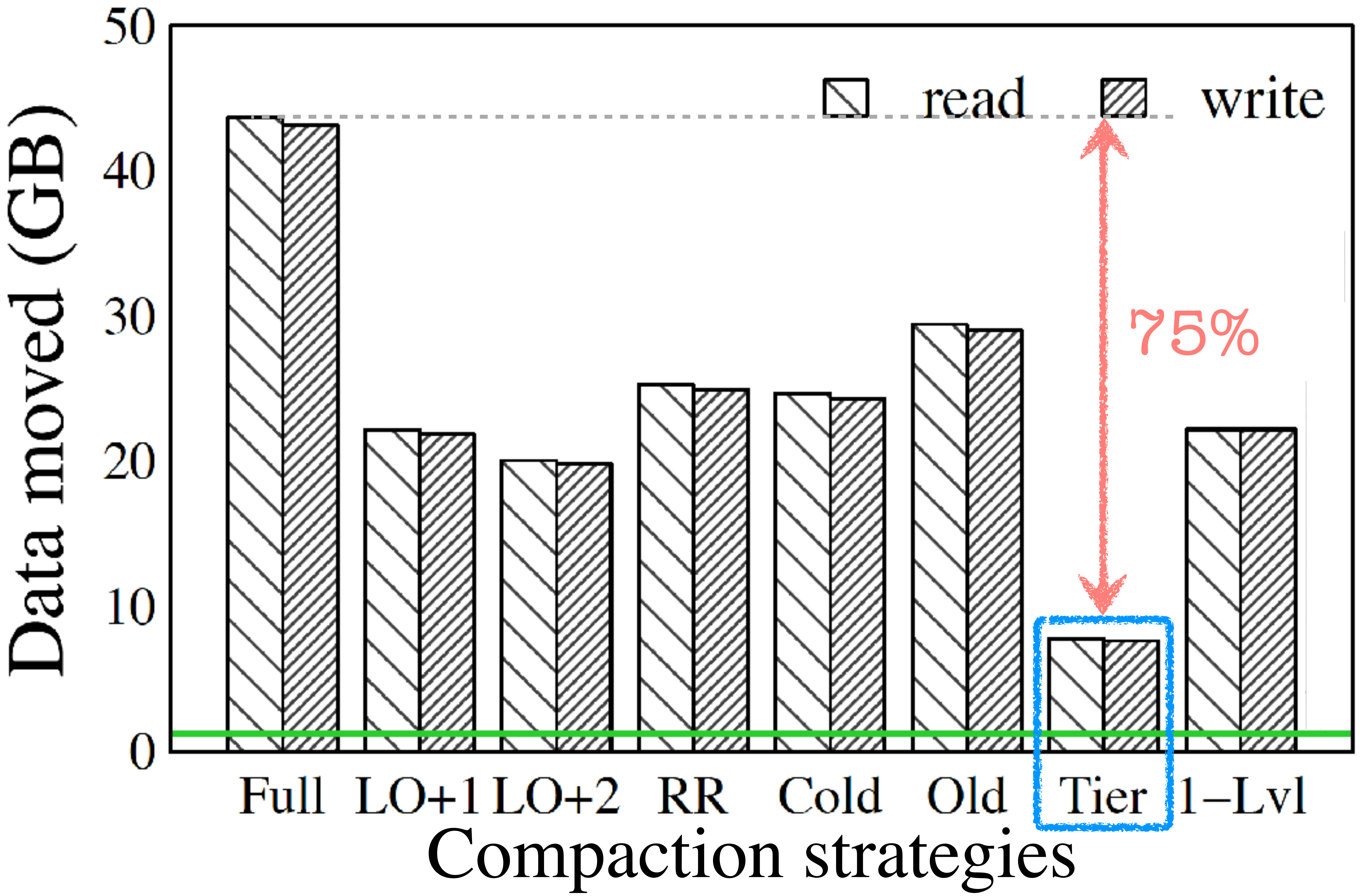
LSM tuning

612 metrics

Compacting data at smaller granularity reduces data movement.

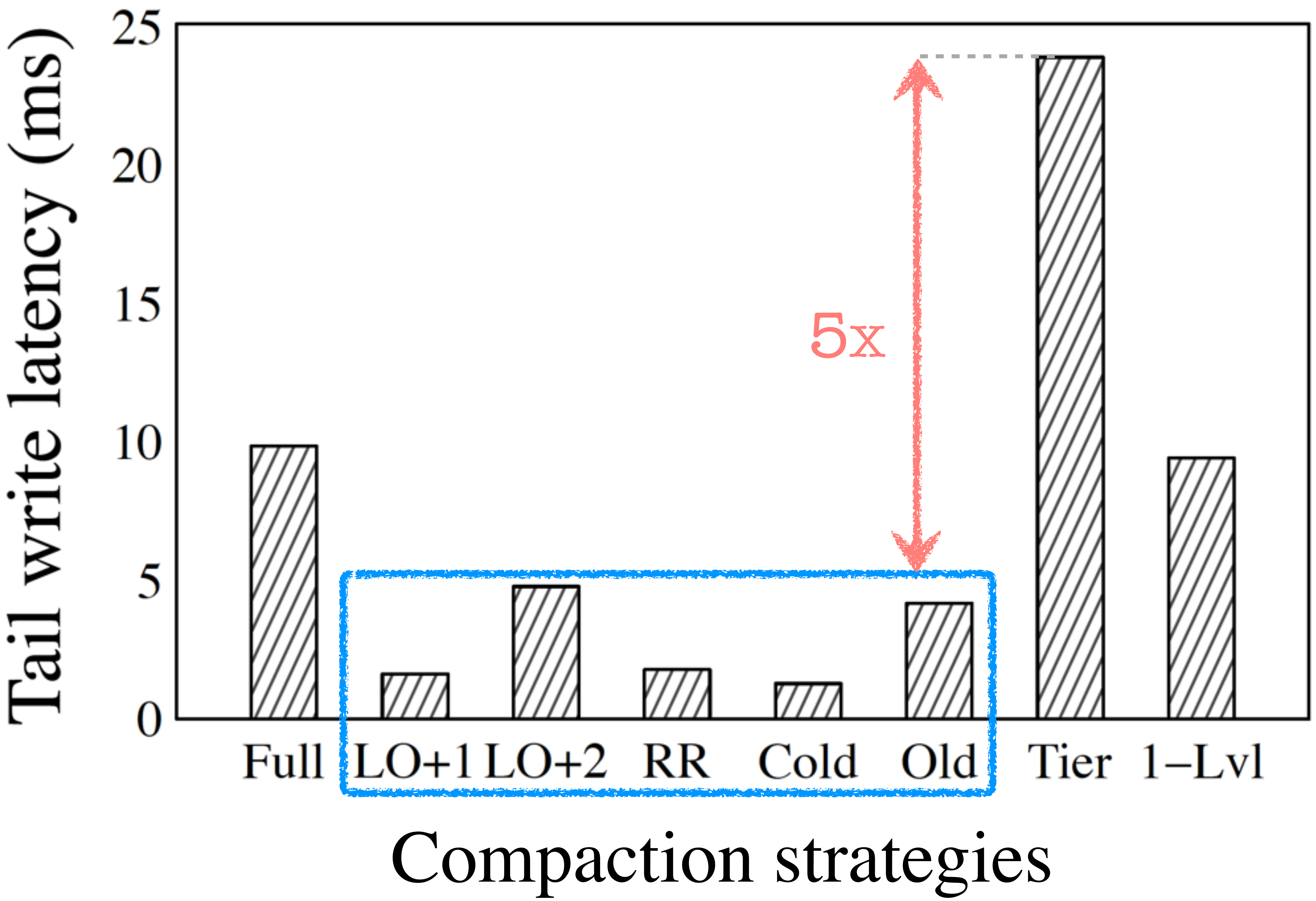


Compacting data at smaller granularity reduces data movement.



Compacting data at smaller granularity reduces data movement.

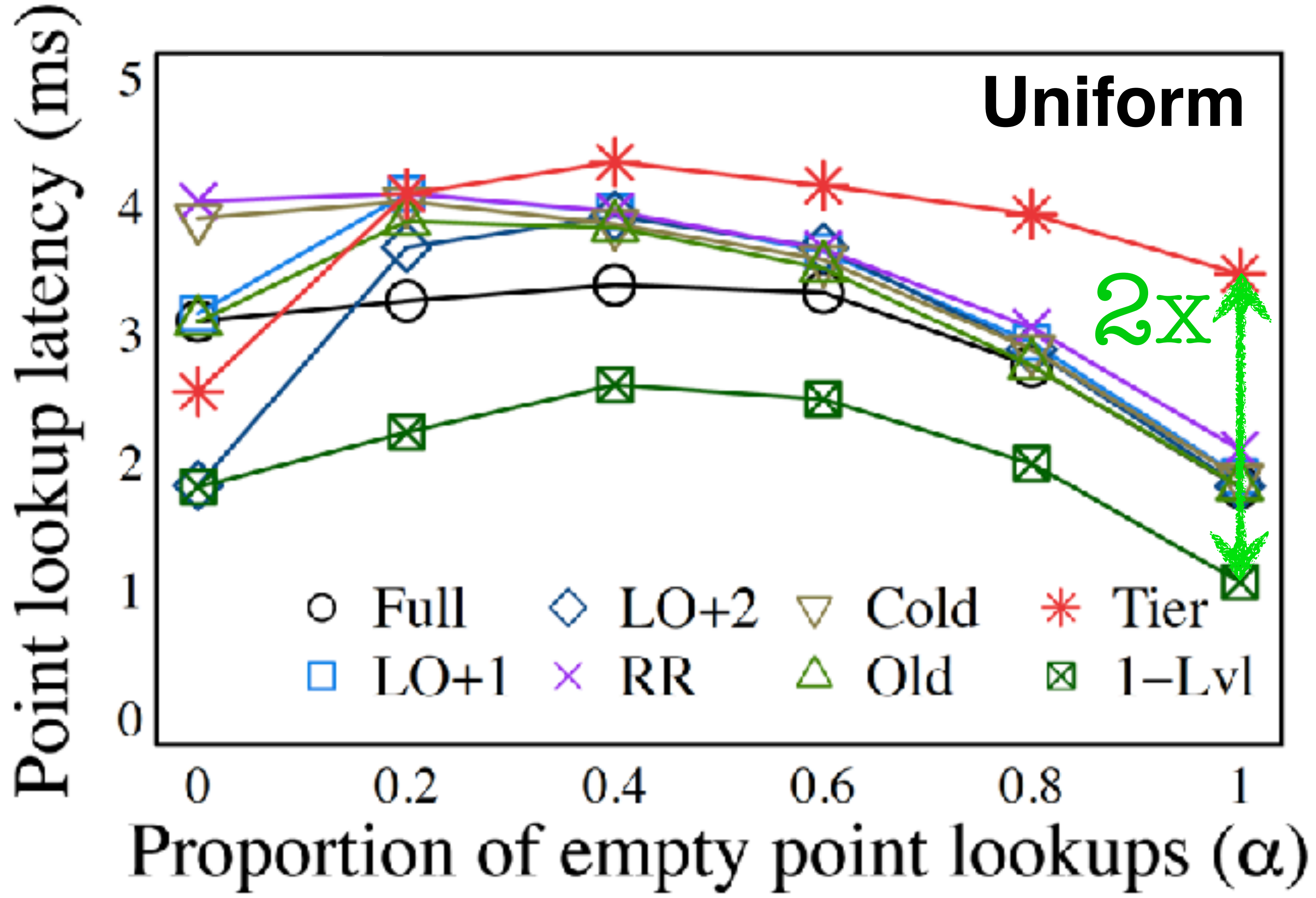
Tiered data layout has the highest write throughput but also the highest tail write latency.

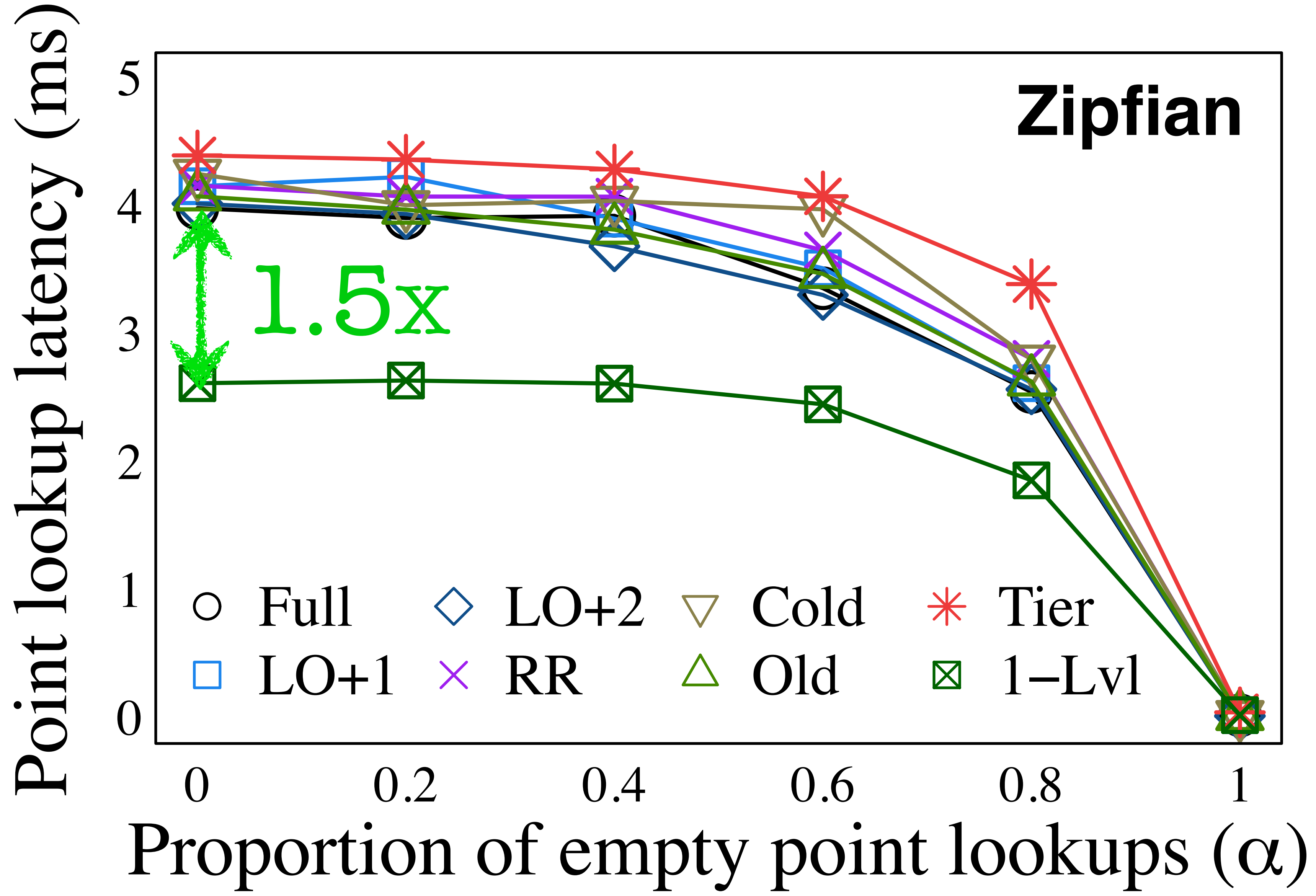


Compacting data at smaller granularity reduces data movement.

Tiered data layout has the highest write throughput but also the highest tail write latency.

Hybrid data layouts dominate point lookup performance.



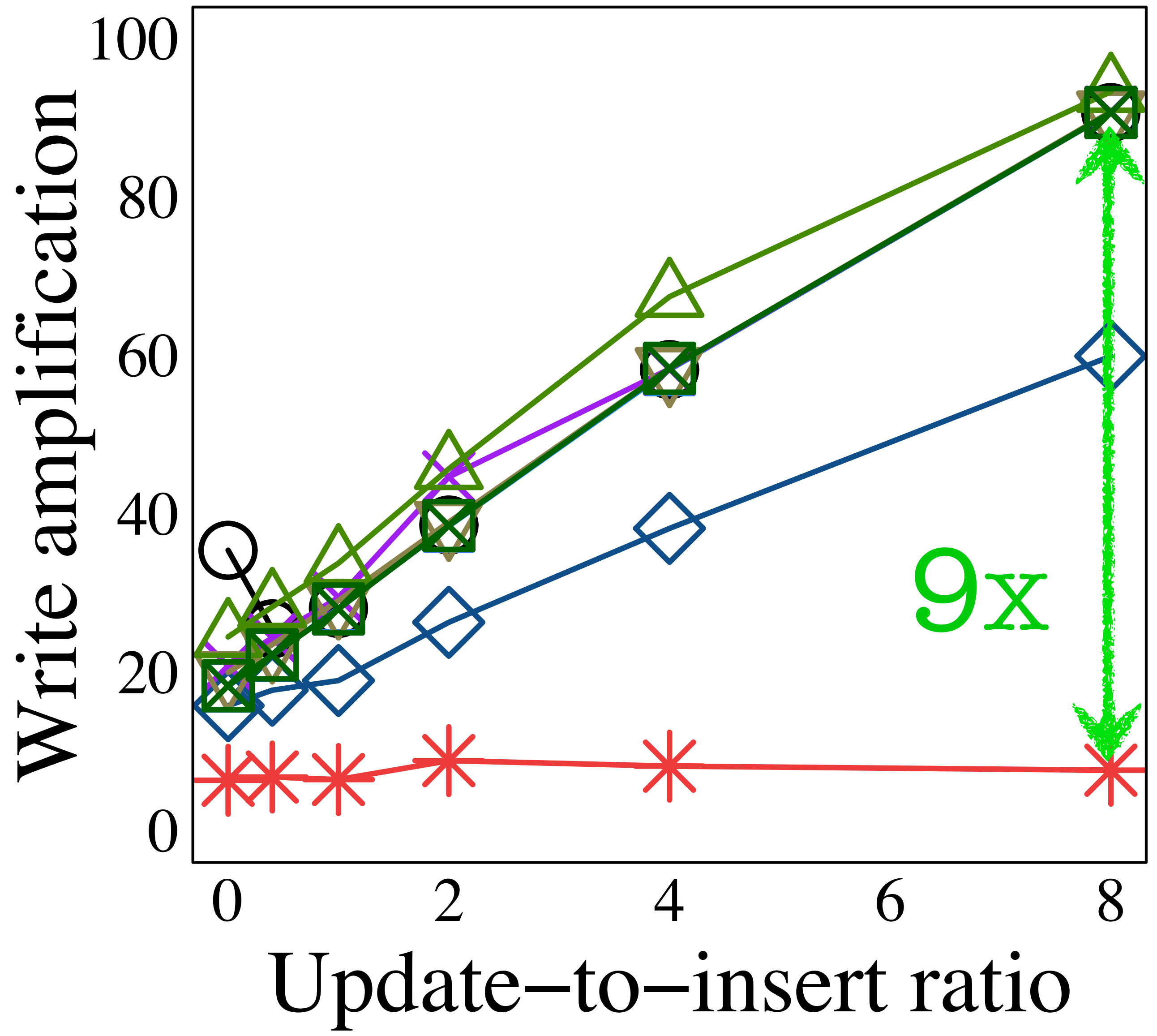


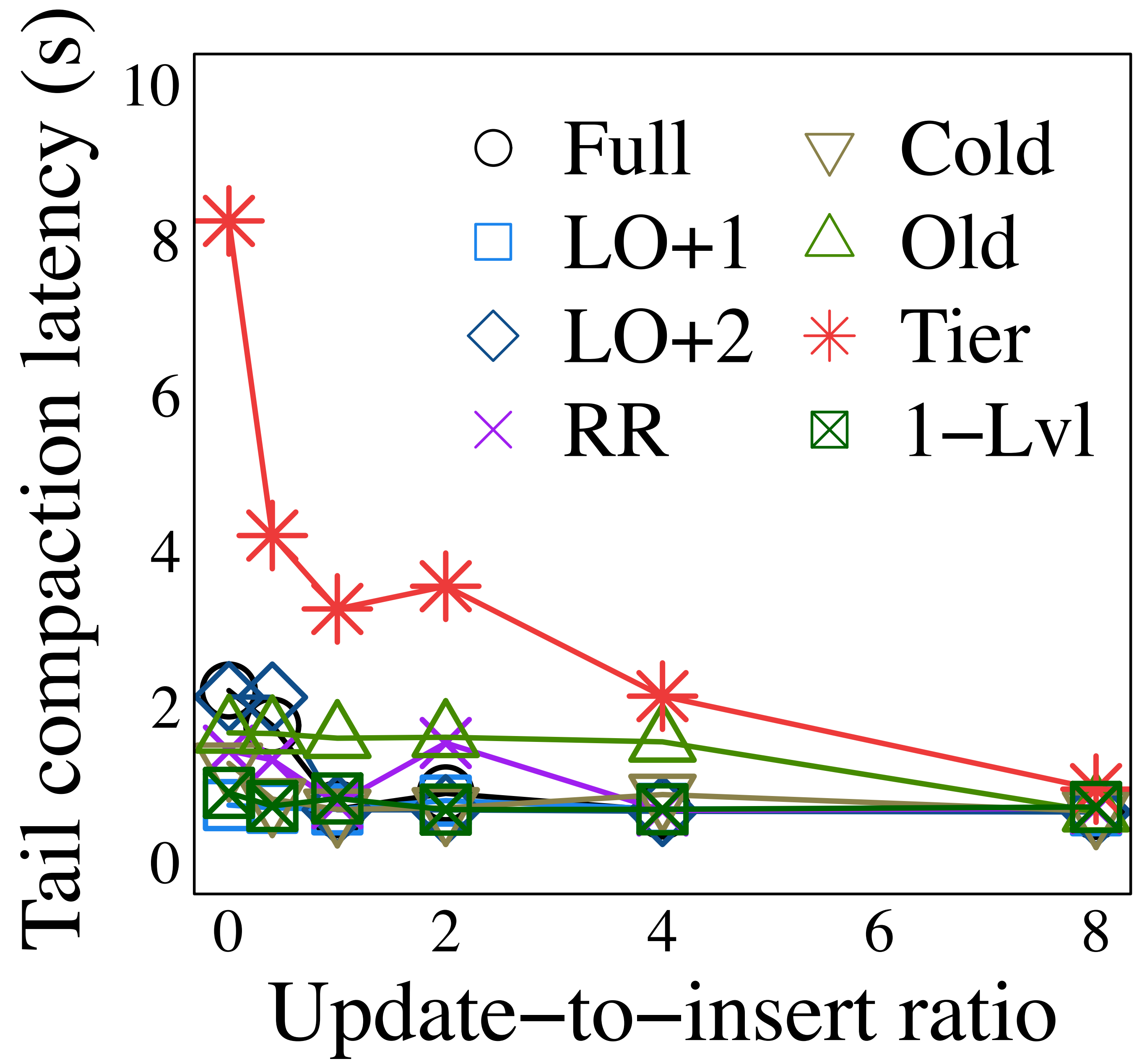
Compacting data at smaller granularity reduces data movement.

For update-intensive workloads, tiering dominates the performance space.

Tiered data layout has the highest write throughput but also the highest tail write latency.

Hybrid data layouts dominate point lookup performance.





Compacting data at smaller granularity reduces data movement.

Tiered data layout has the highest write throughput but also the highest tail write latency.

Hybrid data layouts dominate point lookup performance.

For update-intensive workloads, tiering dominates the performance space.

The relative benefits of compaction strategies are marginally affected by LSM-tuning.

BOSTON UNIVERSITY

Summary

Compaction is **key to LSM-performance**.

Compaction as first-order **design primitives**.

Guidelines to design and tuning through experiments.

BOSTON UNIVERSITY

Thank You!