CS 265

Stratos Idreos

BIG DATA SYSTEMS

NoSQL | Neural Networks | SQL | Graph | Data Science

piazza is active now

video access active

what you should be doing?

READING

Readings for this week (and systems project)

The Log-Structured Merge-Bush & the Wacky Continuum. Niv Dayan, Stratos Idreos. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 2019

Dostoevsky: Better Space-Time Trade-Offs for LSM-Tree Based Key-Value Stores via Adaptive Removal of Superfluous Merging. Niv Dayan, Stratos Idreos. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 2018

Optimal Bloom Filters and Adaptive Merging for LSM-Trees. Niv Dayan, Manos Athanassoulis, Stratos Idreos. ACM Transactions on Database Systems, Dec 2018

Monkey: Optimal Navigable Key-Value Store. Niv Dayan, Manos Athanassoulis, Stratos Idreos. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 2017

DASlab @ Harvard SEAS

how to prepare slides

no bullets 2 colors big text images animation for examples

how to prepare slides

no bullets 2 colors big text images animation for examples

story

one message per slide connection from slide to slide

how to prepare slides

no bullets 2 colors big text images animation for examples

story

one message per slide connection from slide to slide

1) prepare slides, 2) meet with a TF and Stratos the week before

How many and which structures are possible?

DESIGN SPACE

Can we compute performance w/o coding?

COST SYNTHESIS

more in depth discussion on NoSQL storage + design optimizations

(systems project and related research projects)

first steps in constructing a design space

NoSQL Key-value Stores

RocksDB Google BigTable mongoDB SQLite in

amazon DynamoDB cassandra APACHE HBASE

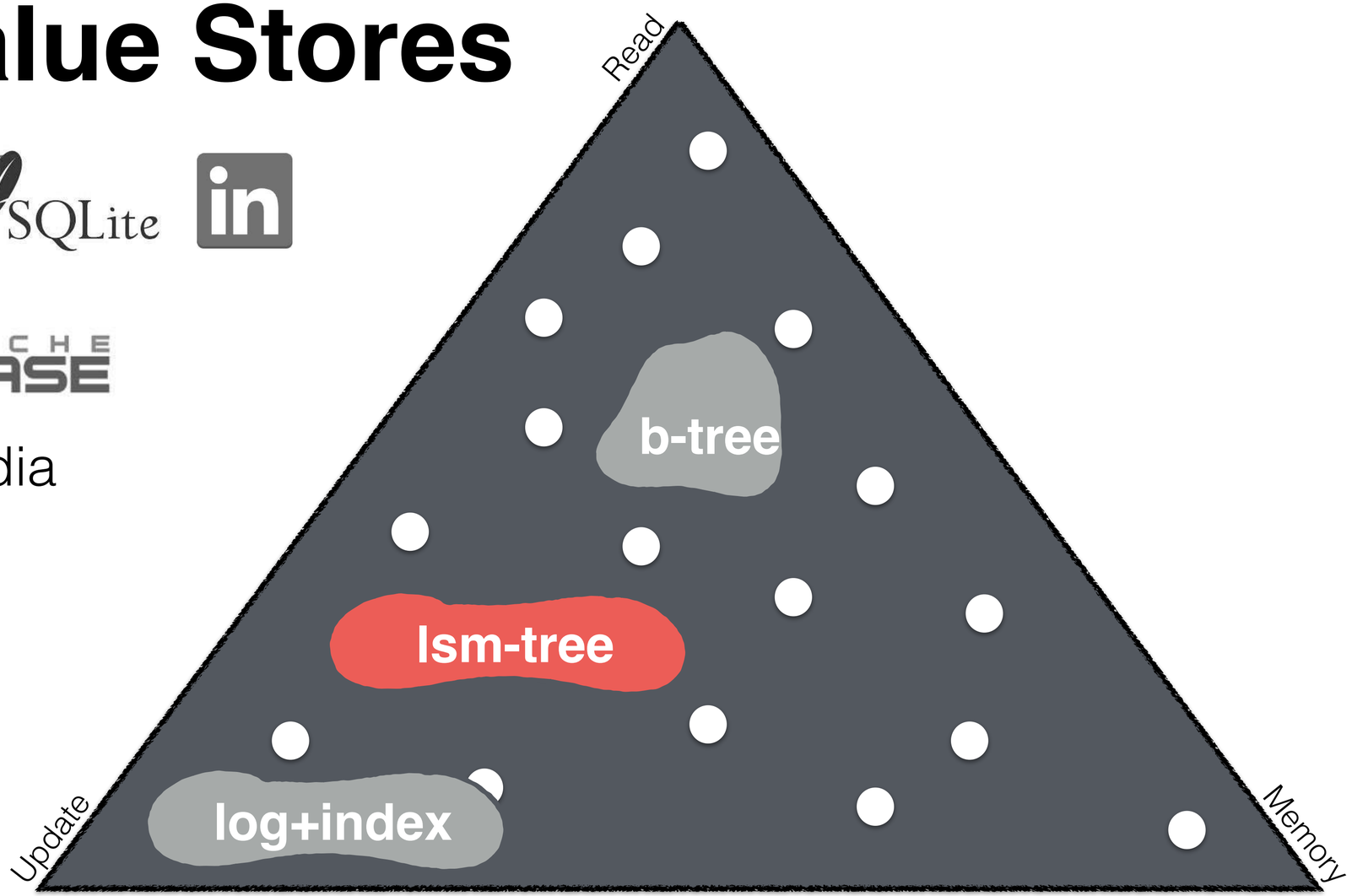
machine learning social media

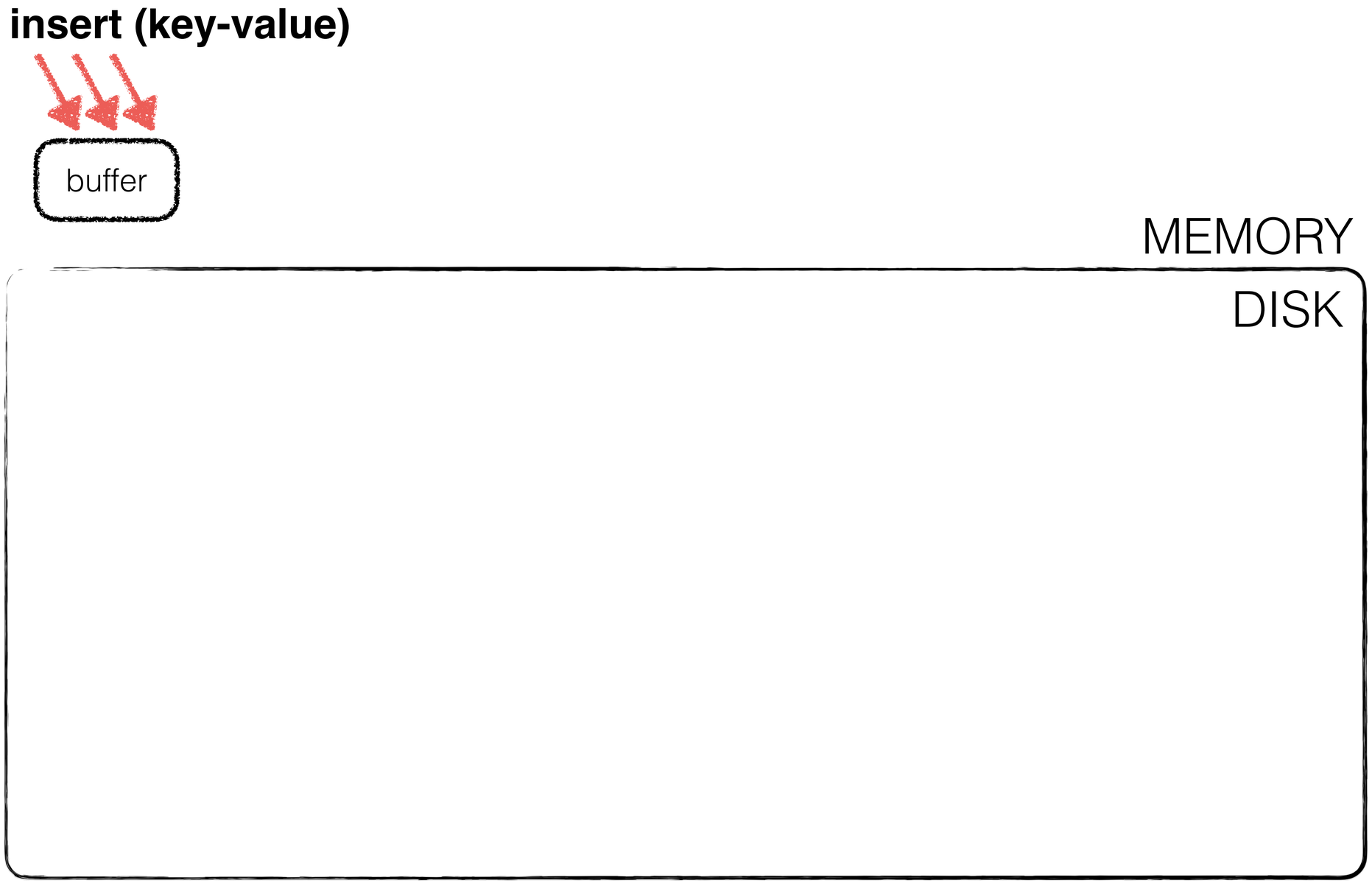
smart homes web browsers

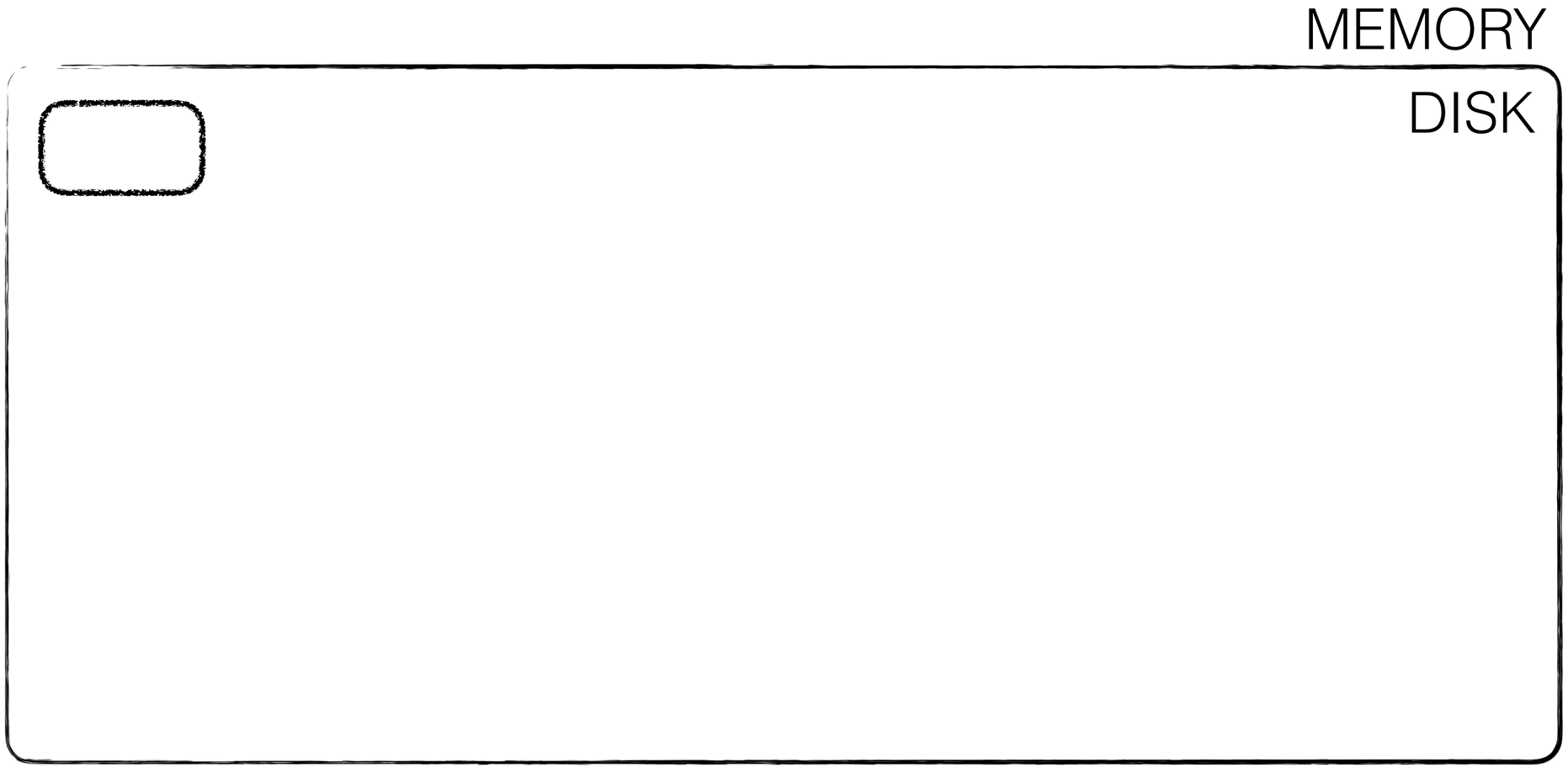
phones web-based apps

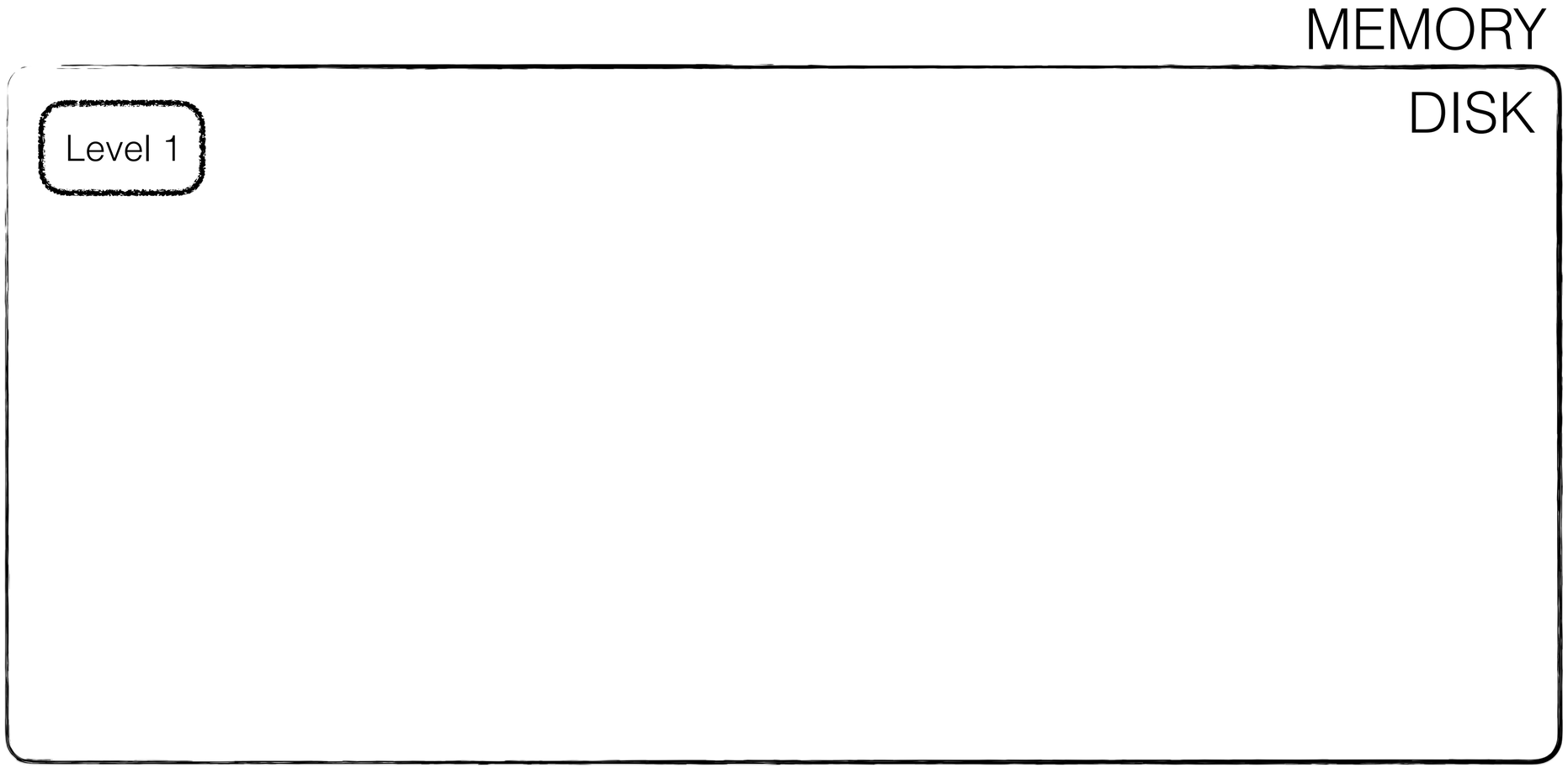
security health devices

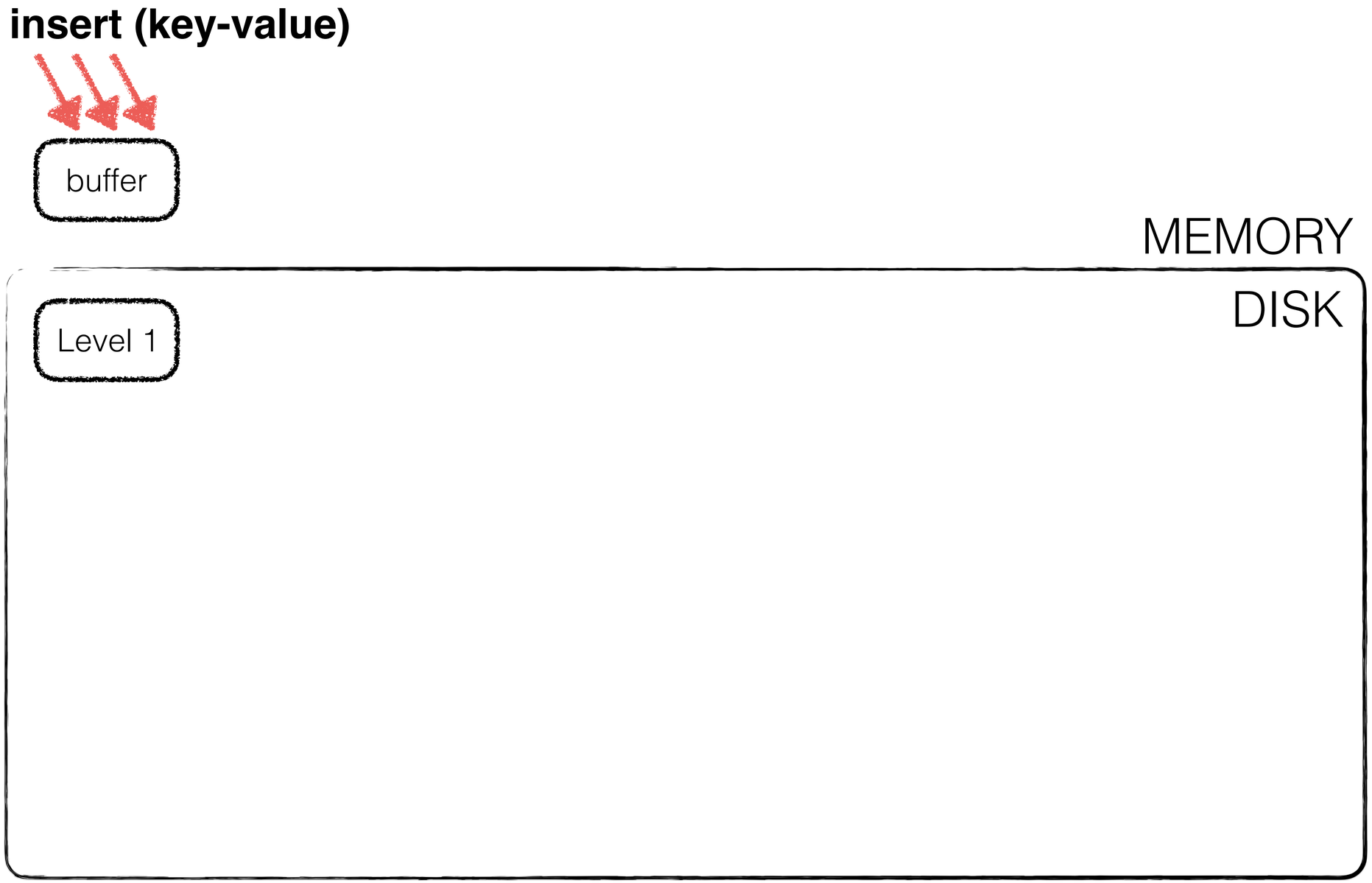
graphs analytics

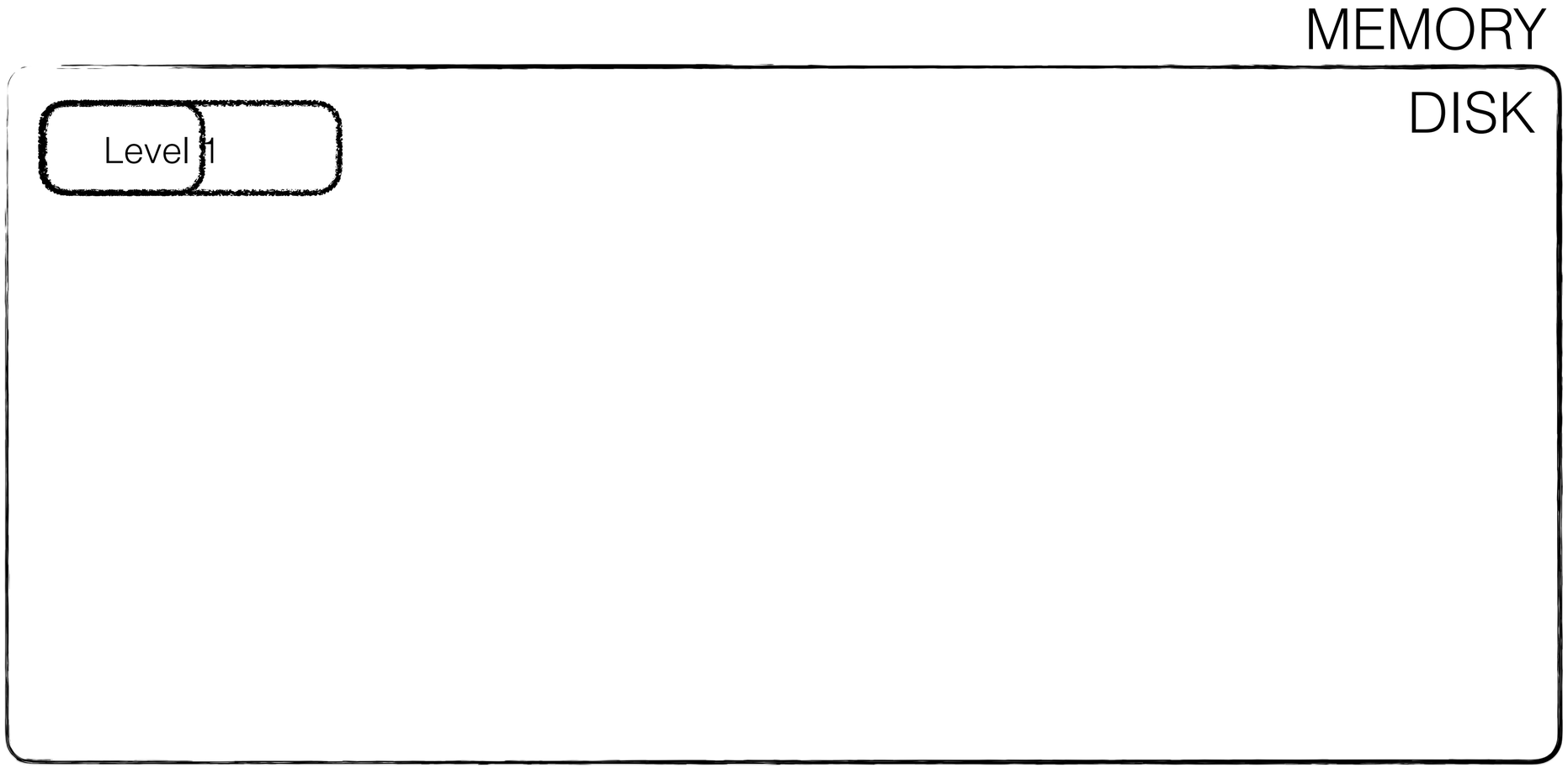


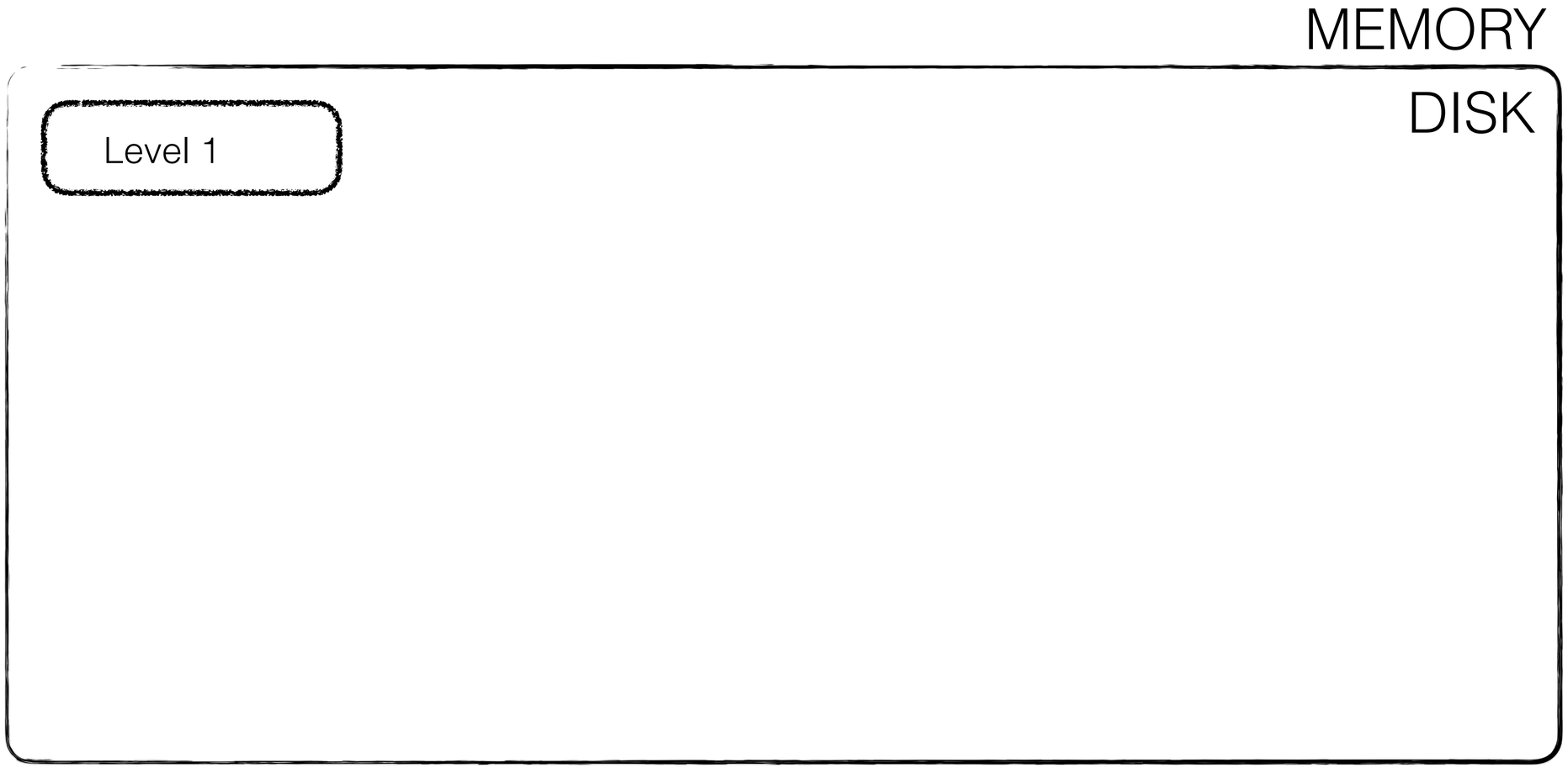

insert (key-value)
buffer
MEMORY
DISK

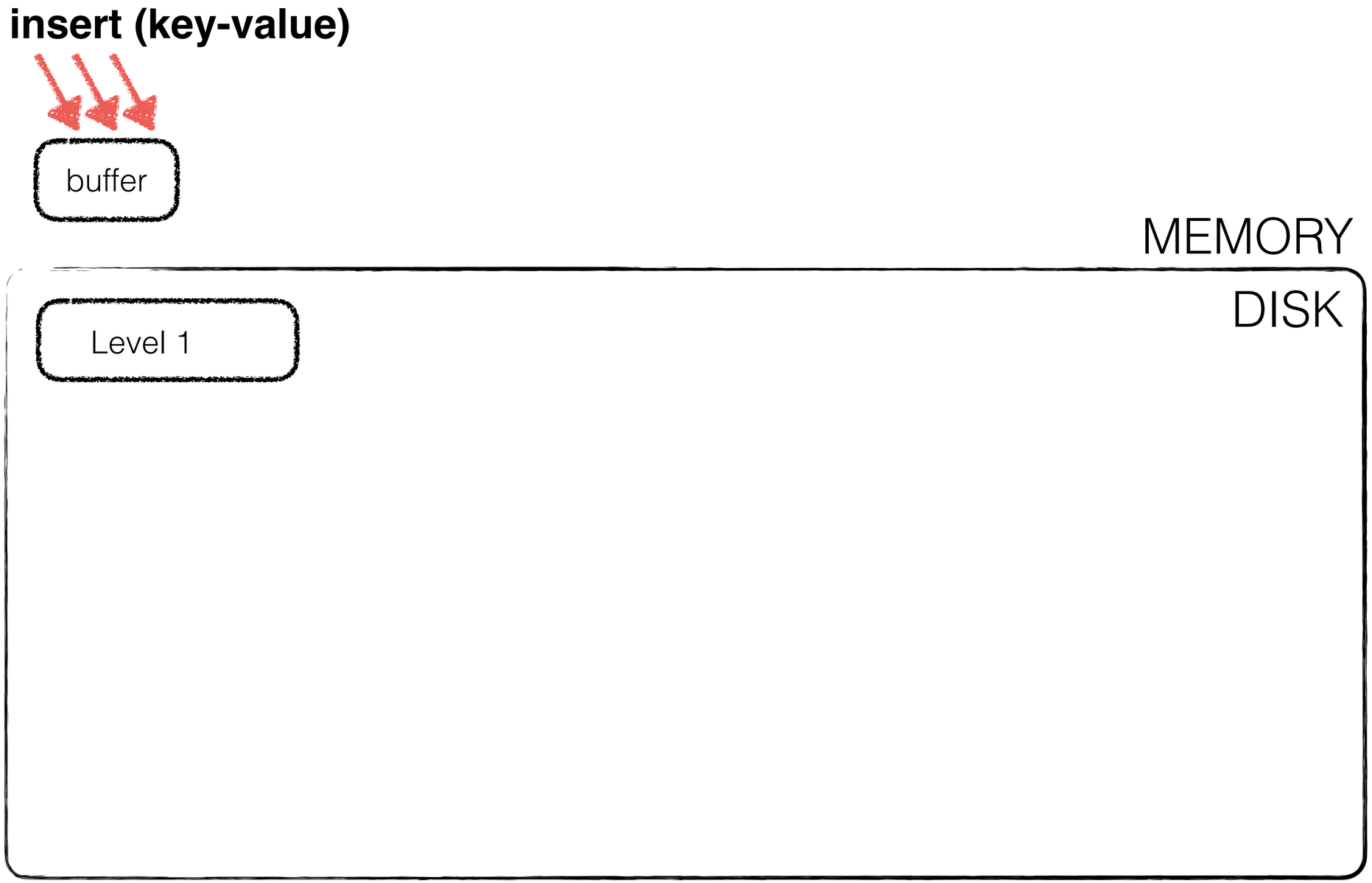

MEMORY
DISK

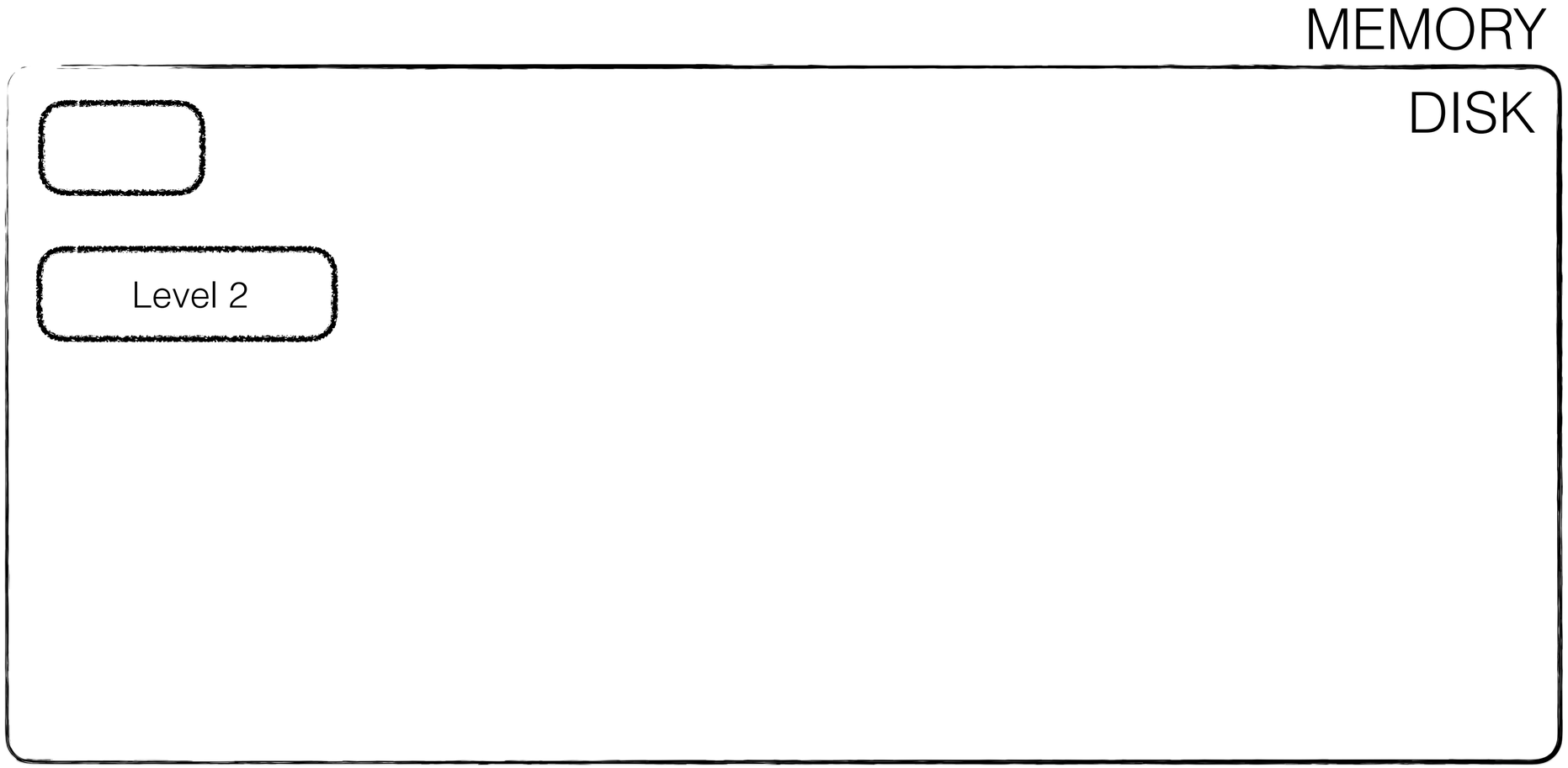

MEMORY
DISK
Level 1

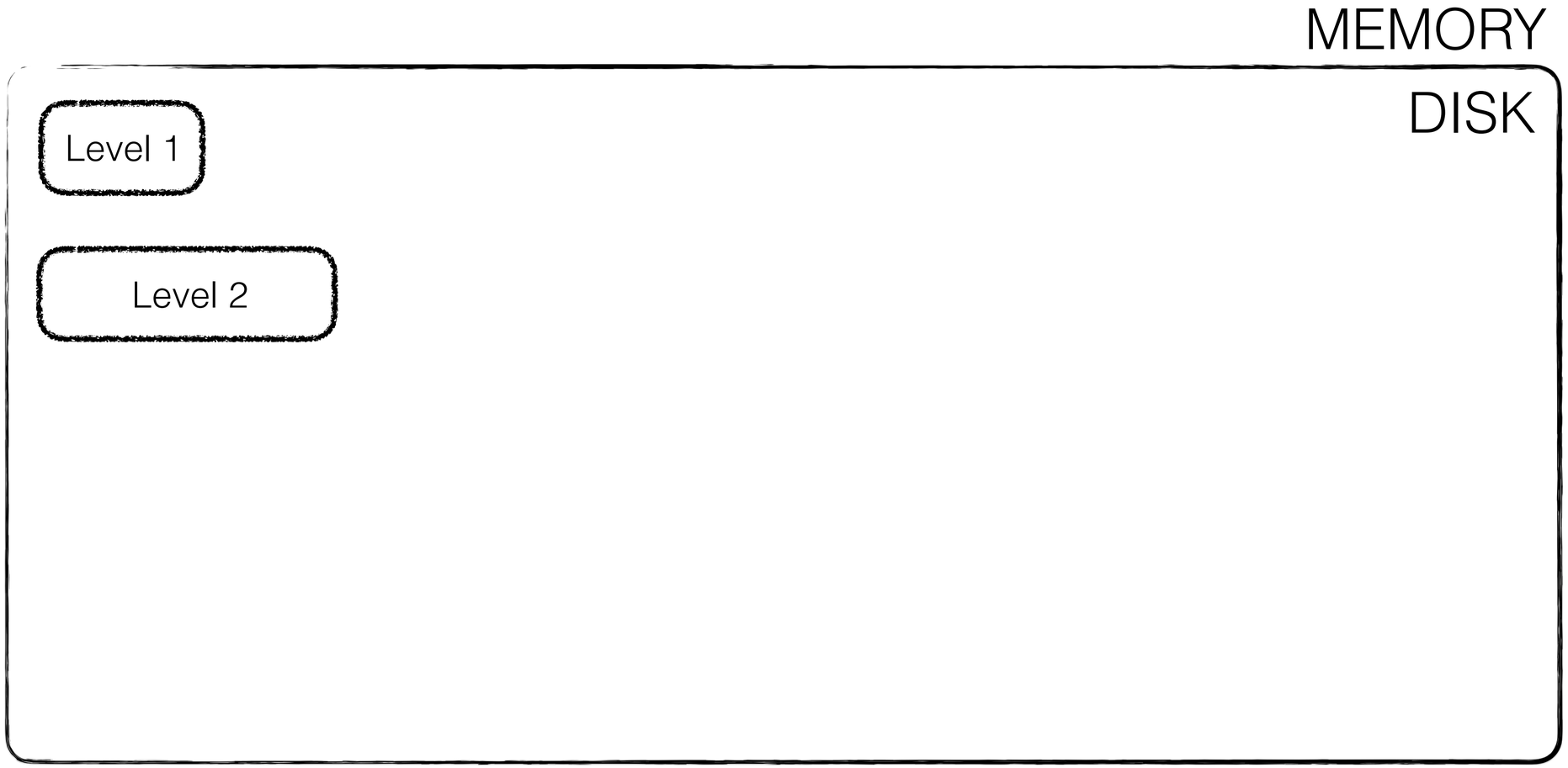

insert (key-value)
buffer
MEMORY
DISK
Level 1

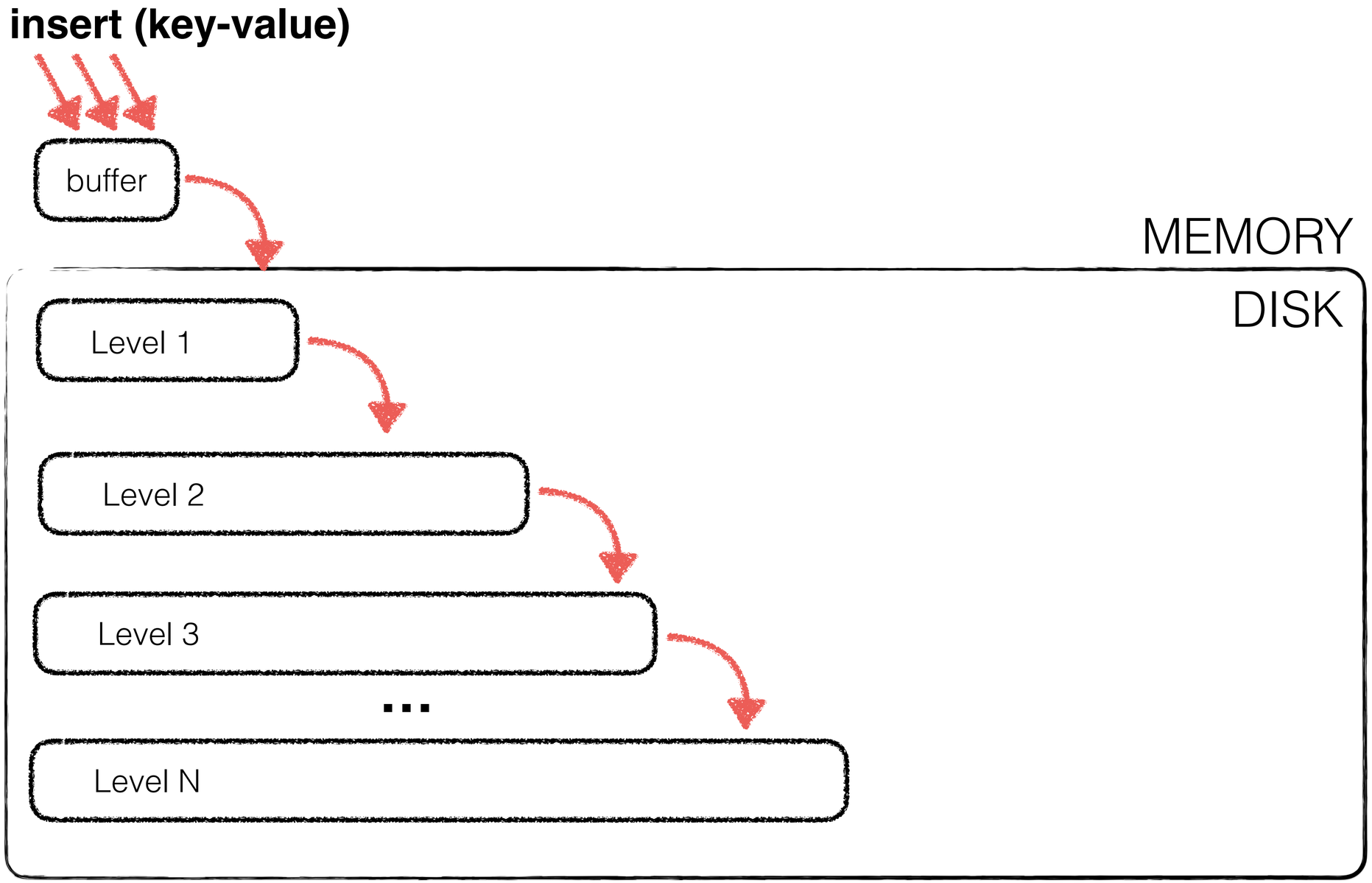

MEMORY
DISK
Level 1

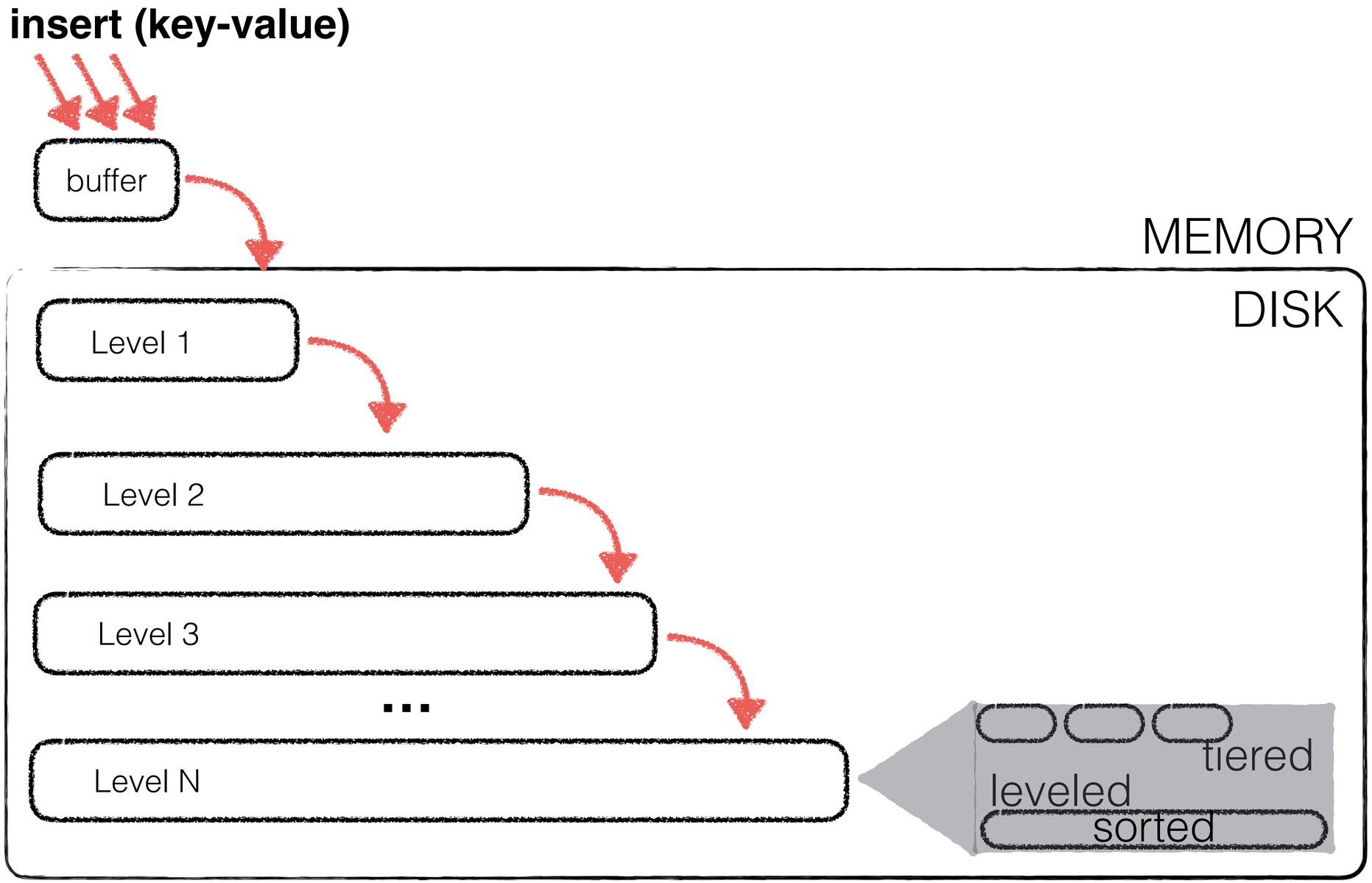

MEMORY
DISK
Level 1

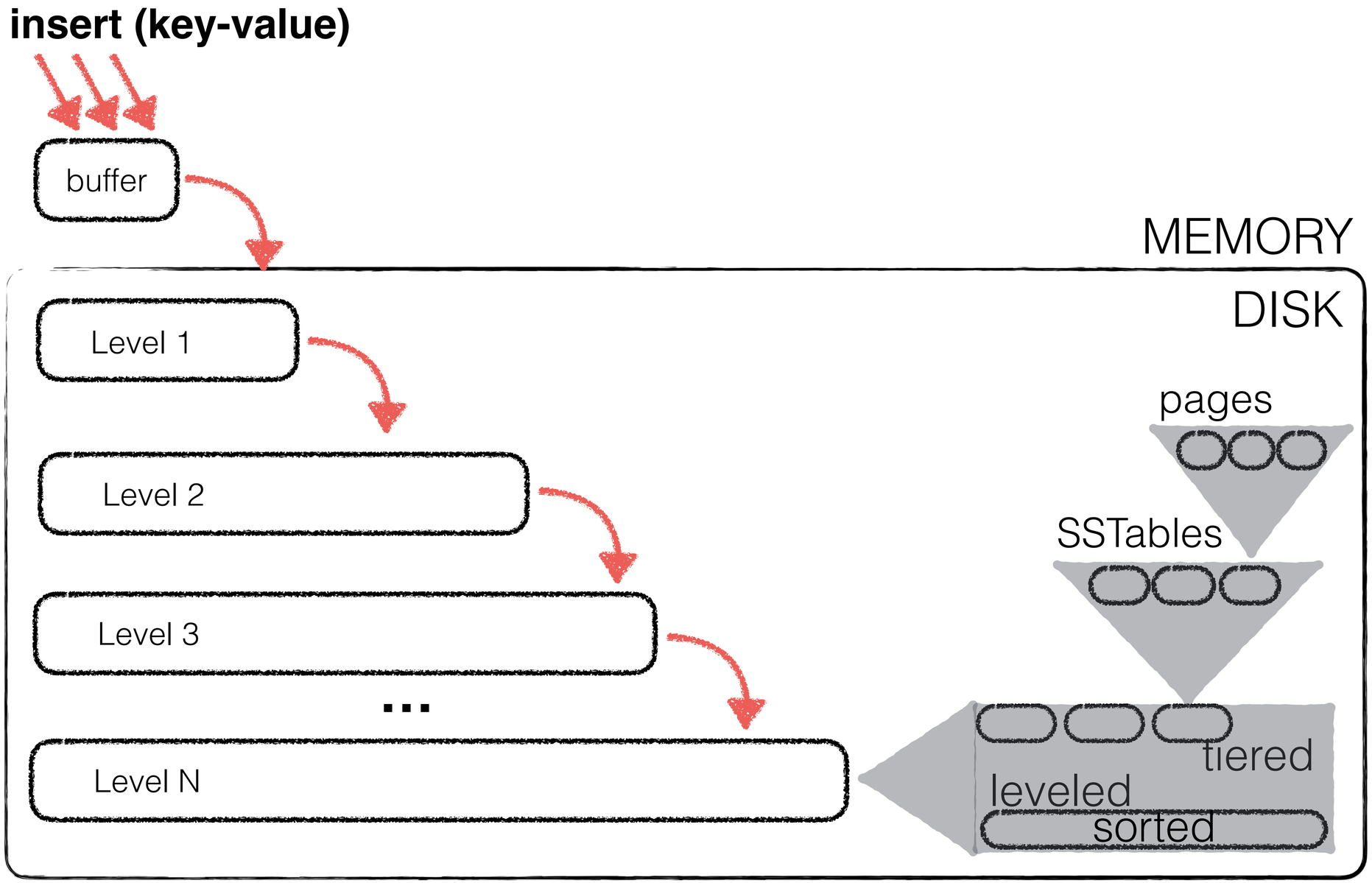

insert (key-value)
buffer
MEMORY
DISK
Level 1

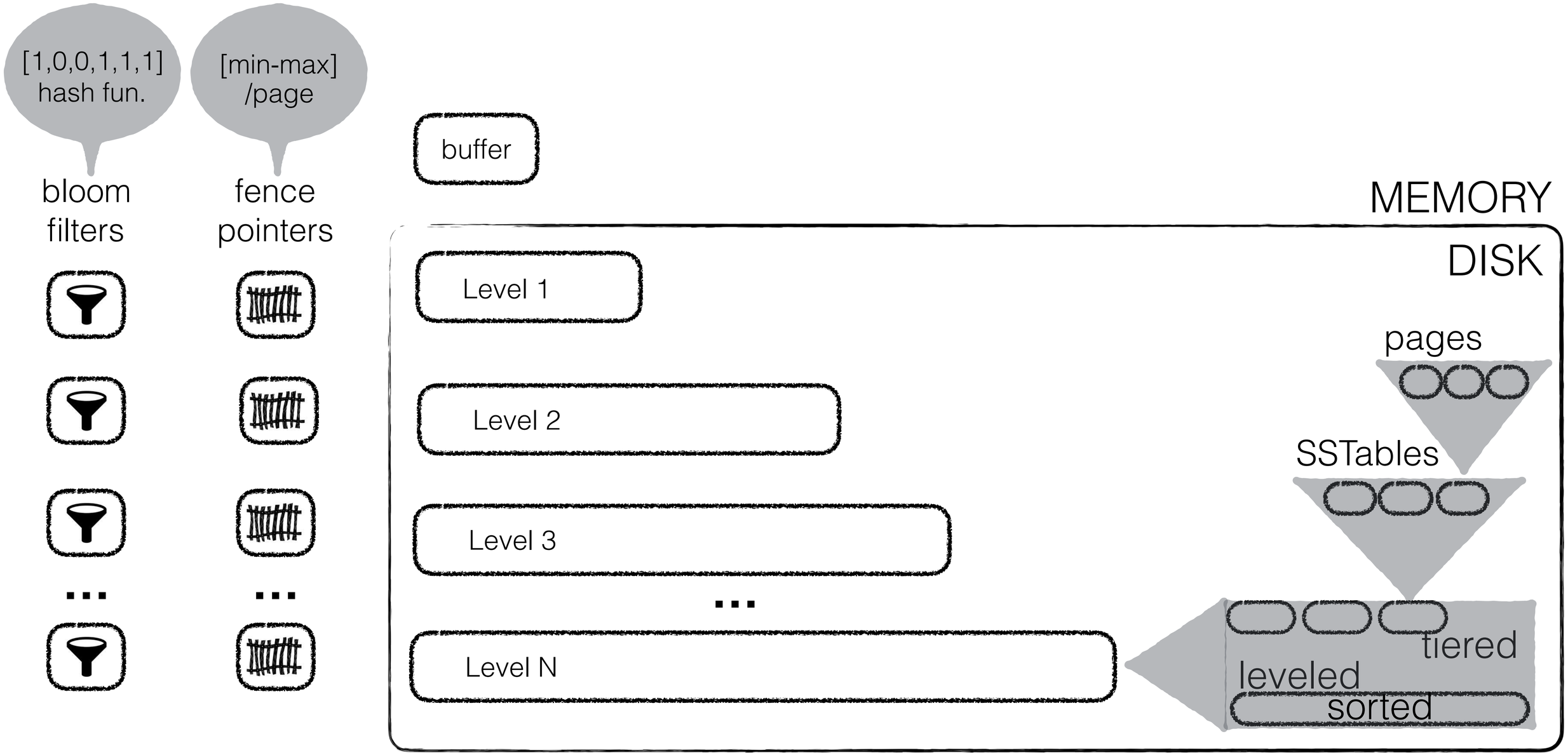

MEMORY
DISK
Level 2

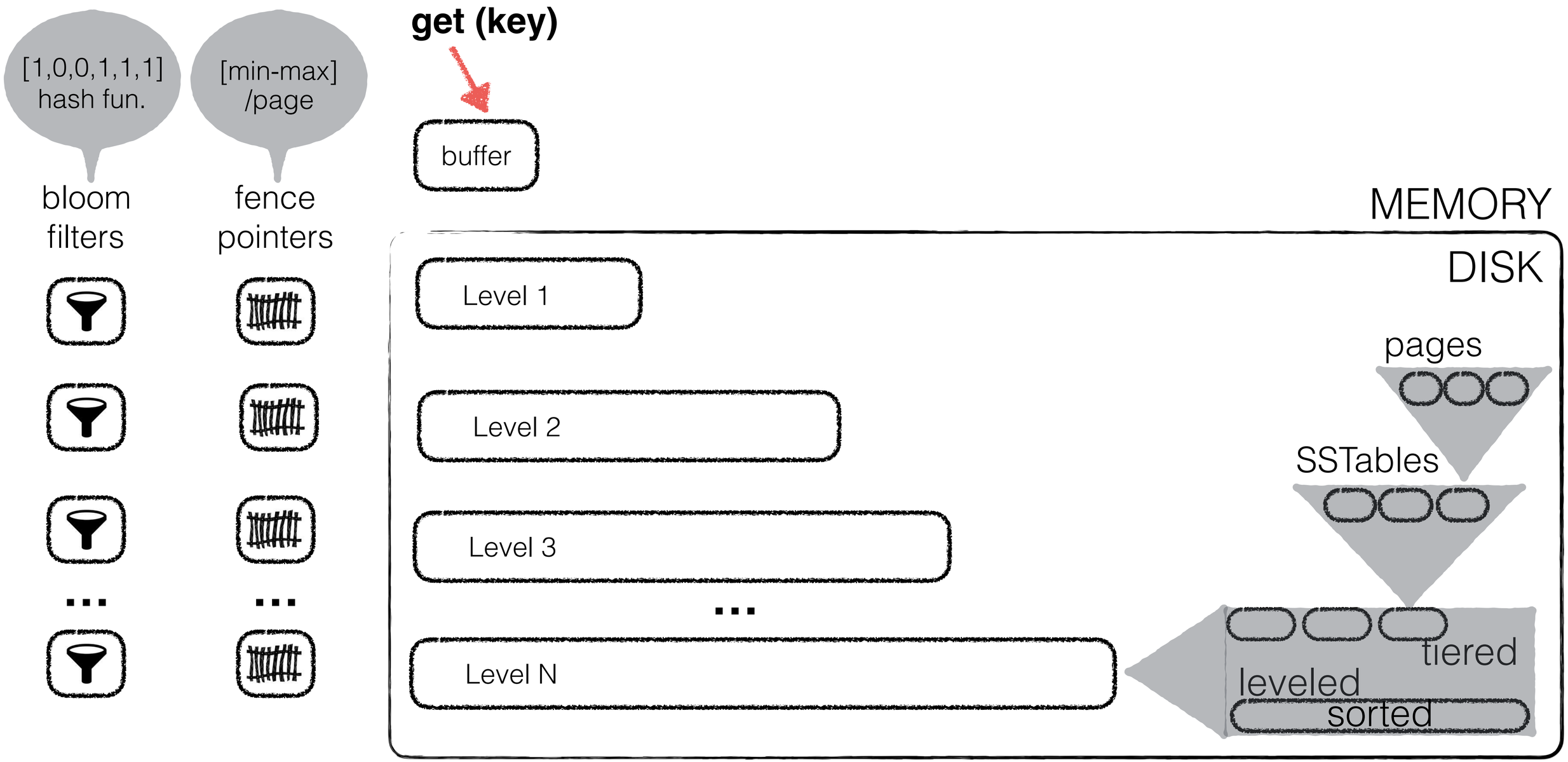

MEMORY
DISK
Level 1
Level 2

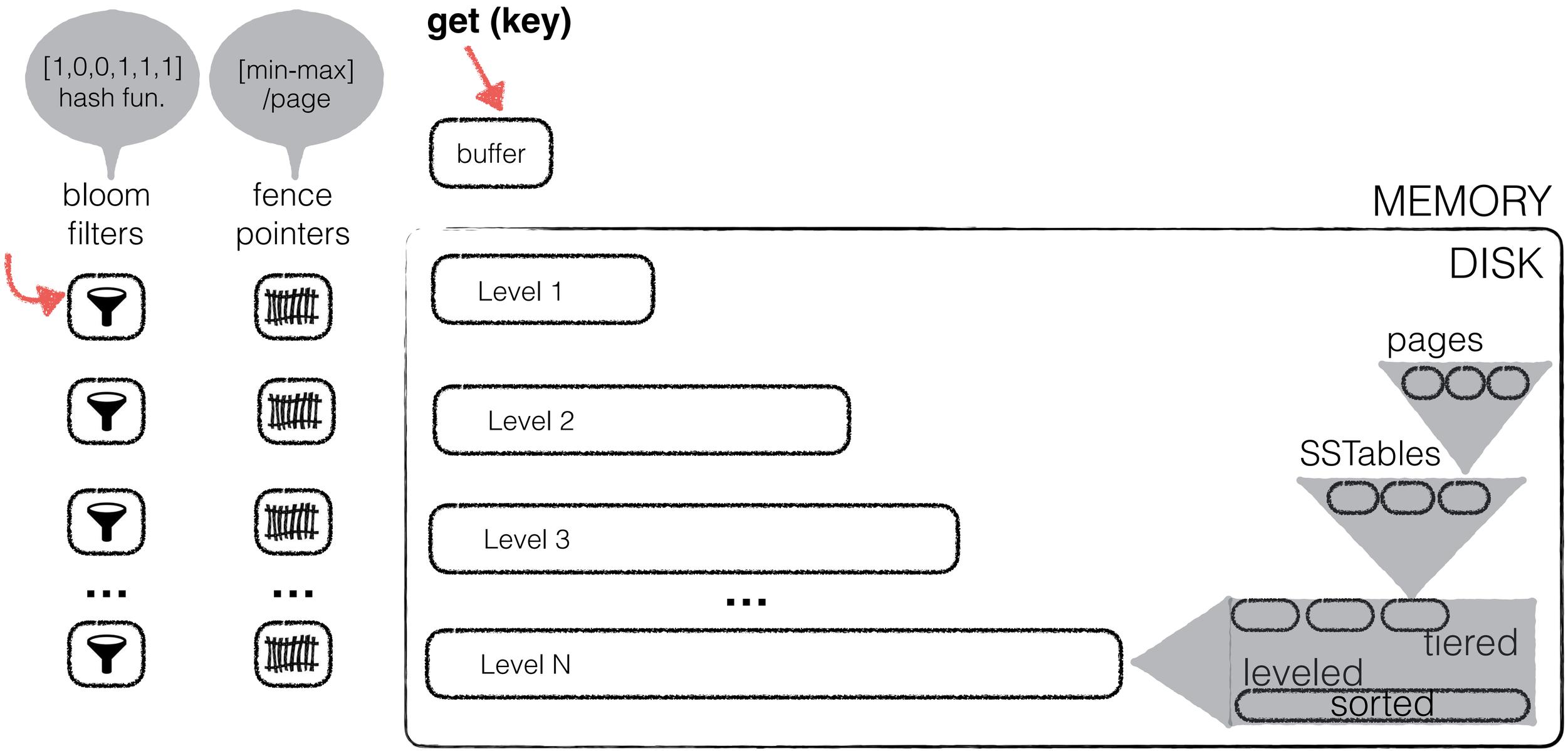

insert (key-value)
buffer
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N

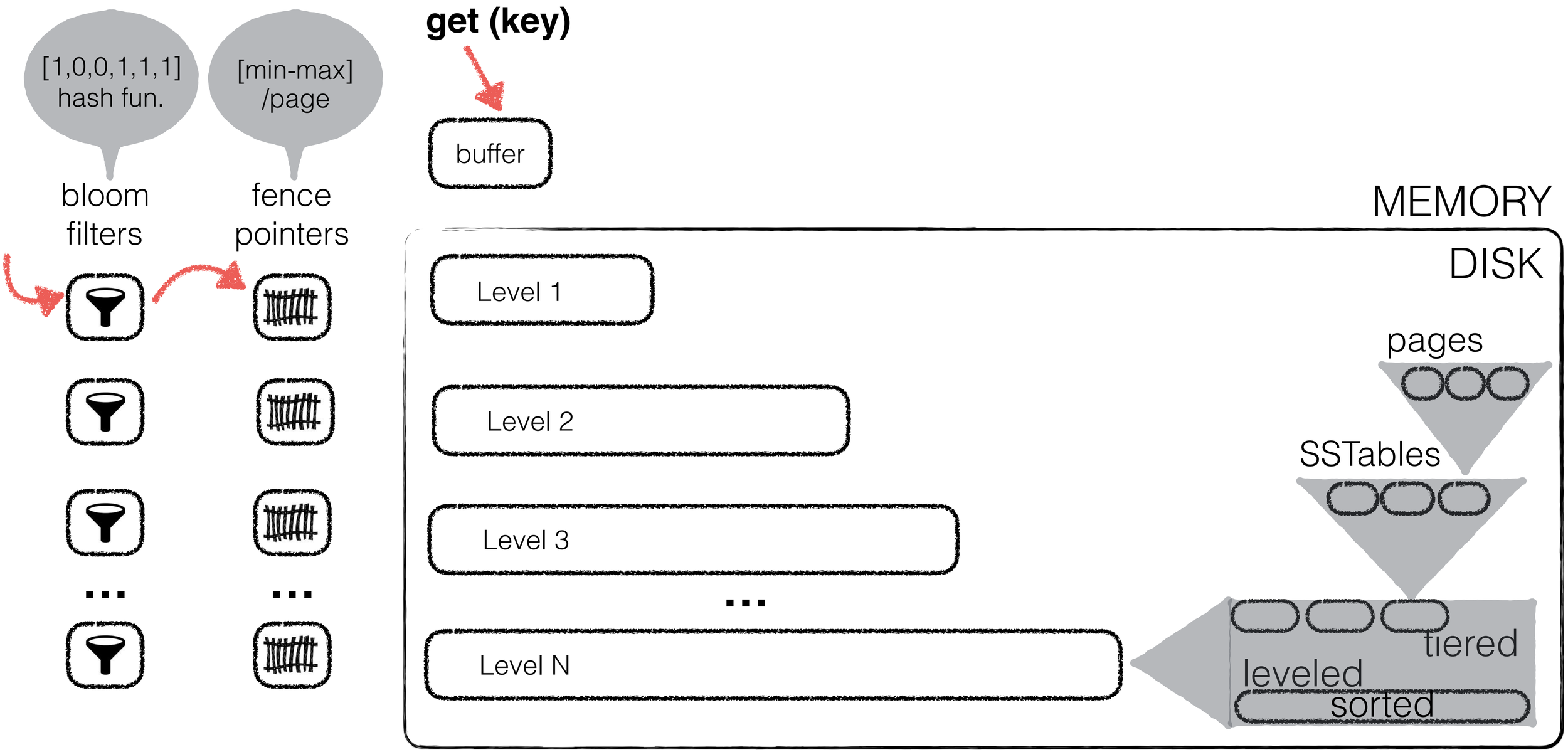

insert (key-value)
buffer
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N
tiered
leveled
sorted

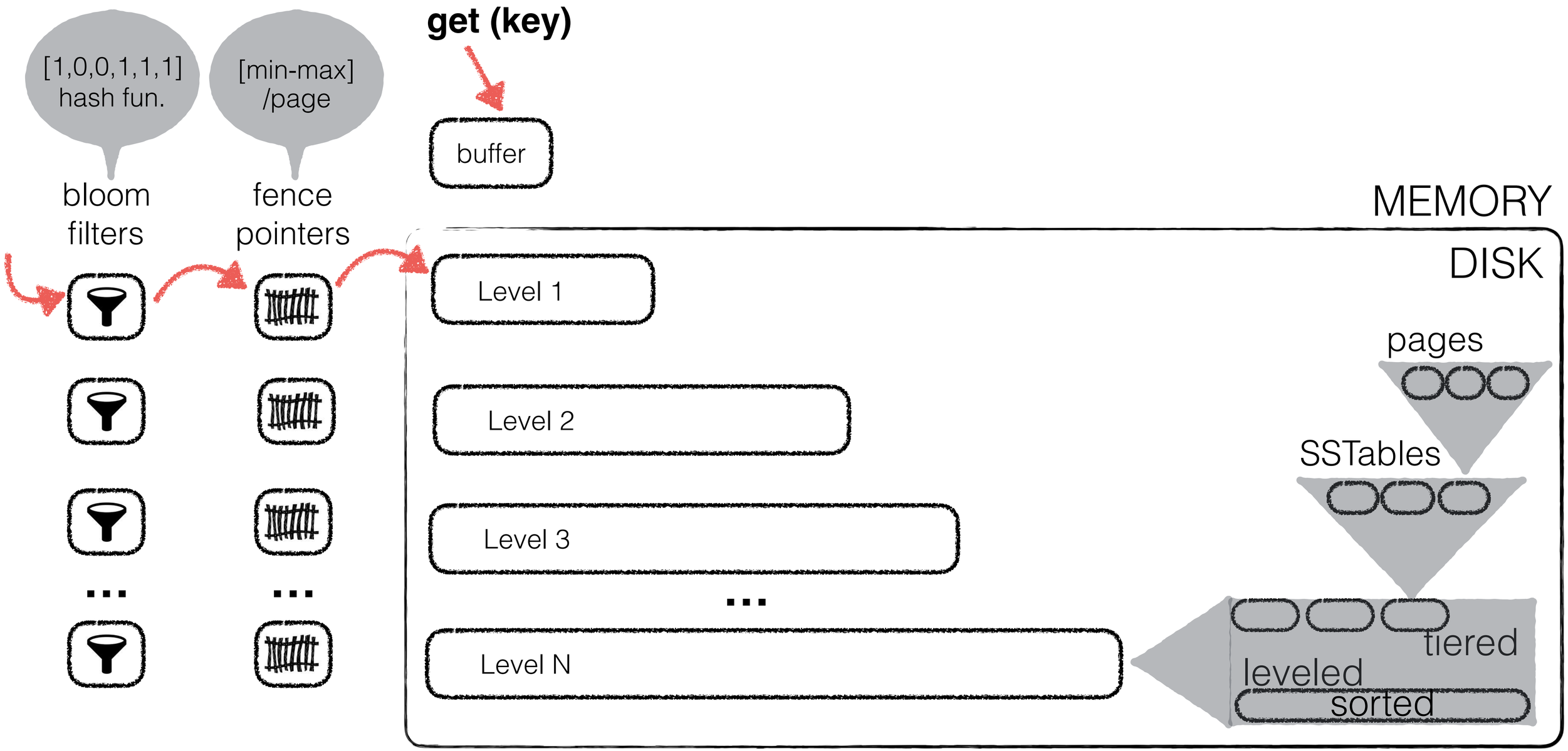

insert (key-value)
buffer
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N
pages
SSTables
tiered
leveled
sorted

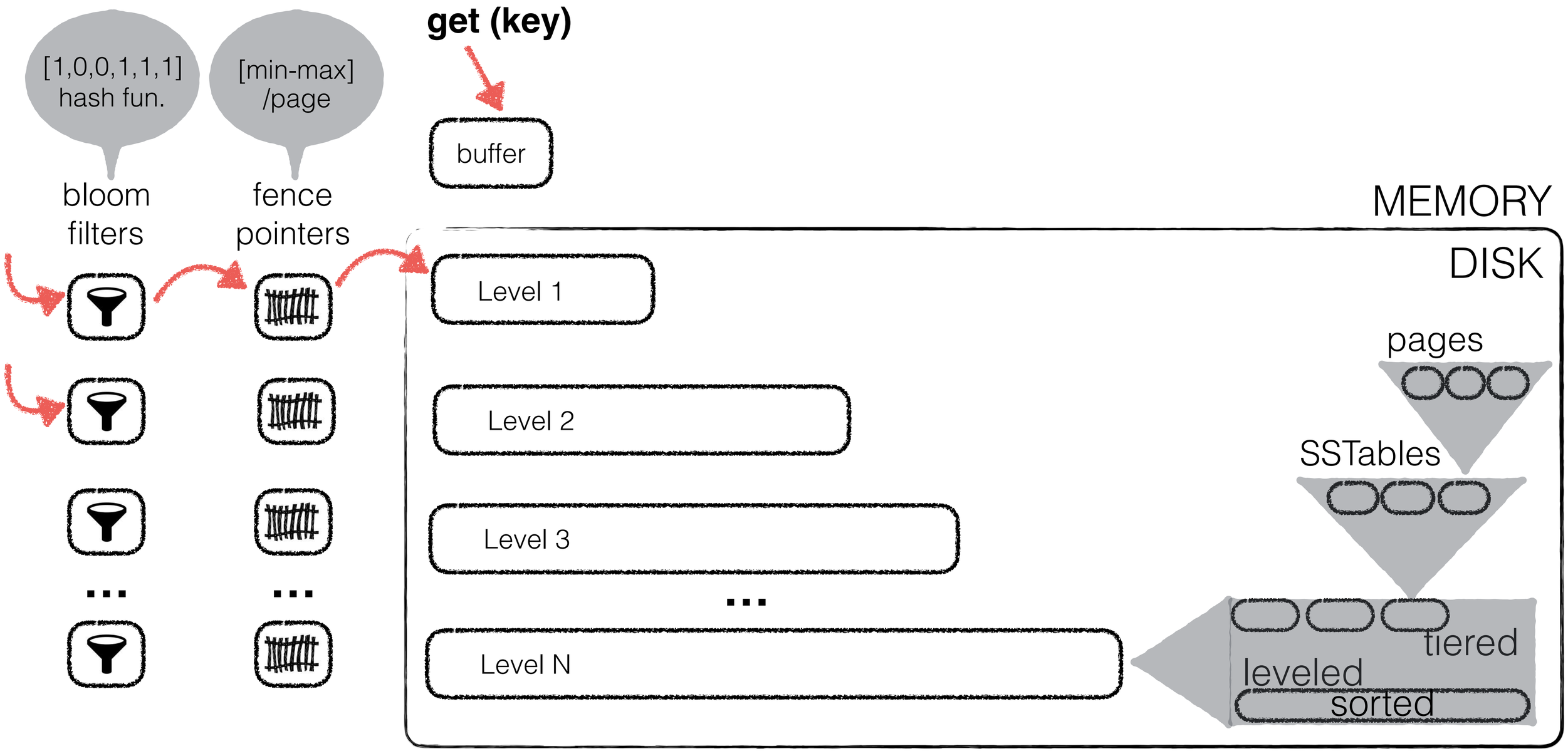

[1,0,0,1,1,1]
hash fun.
[min-max]
/page
bloom
filters
fence
pointers
buffer
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N
pages
SSTables
tiered
leveled
sorted

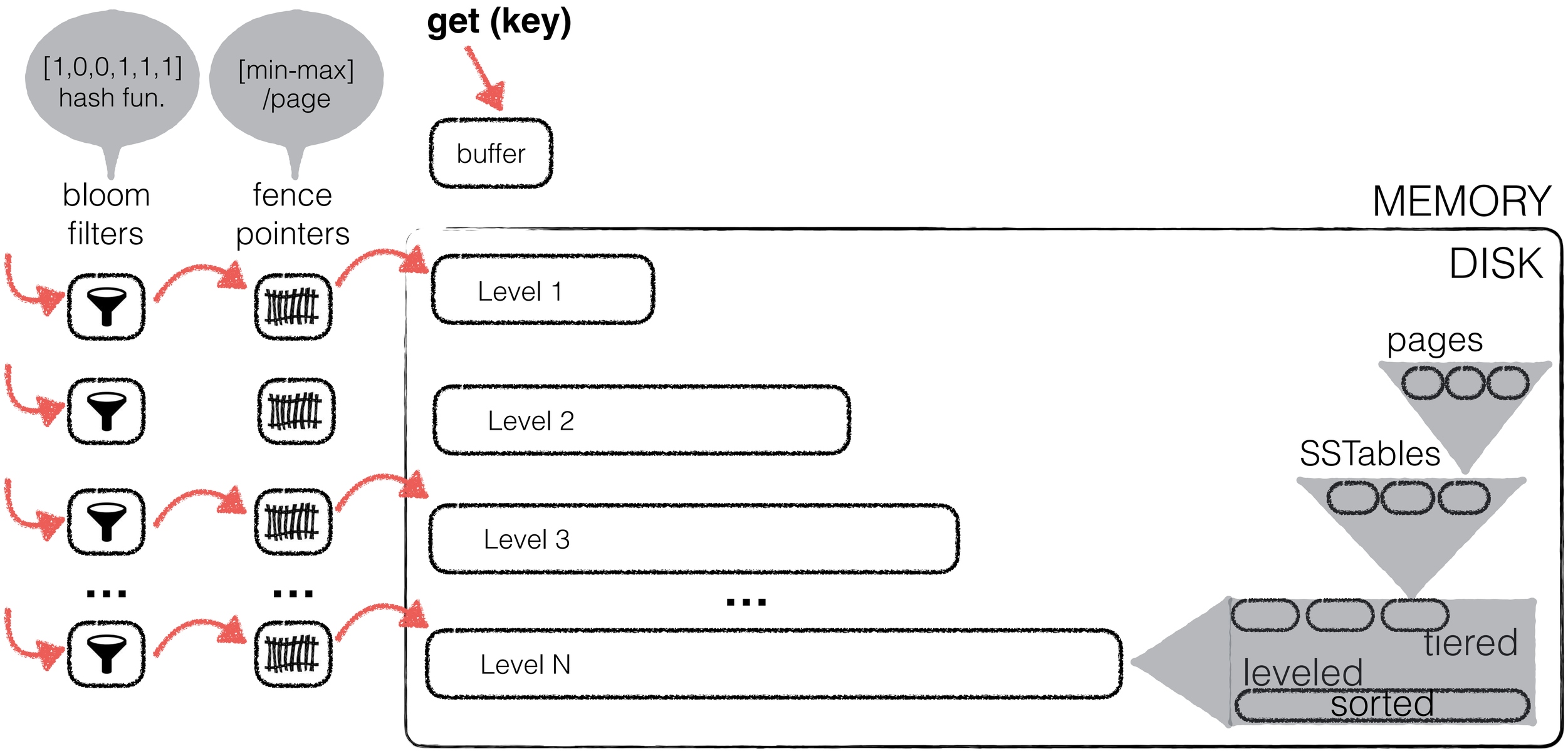

[1,0,0,1,1,1]
hash fun.
[min-max]
/page
bloom
filters
fence
pointers
get (key)
buffer
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N
pages
SSTables
tiered
leveled
sorted

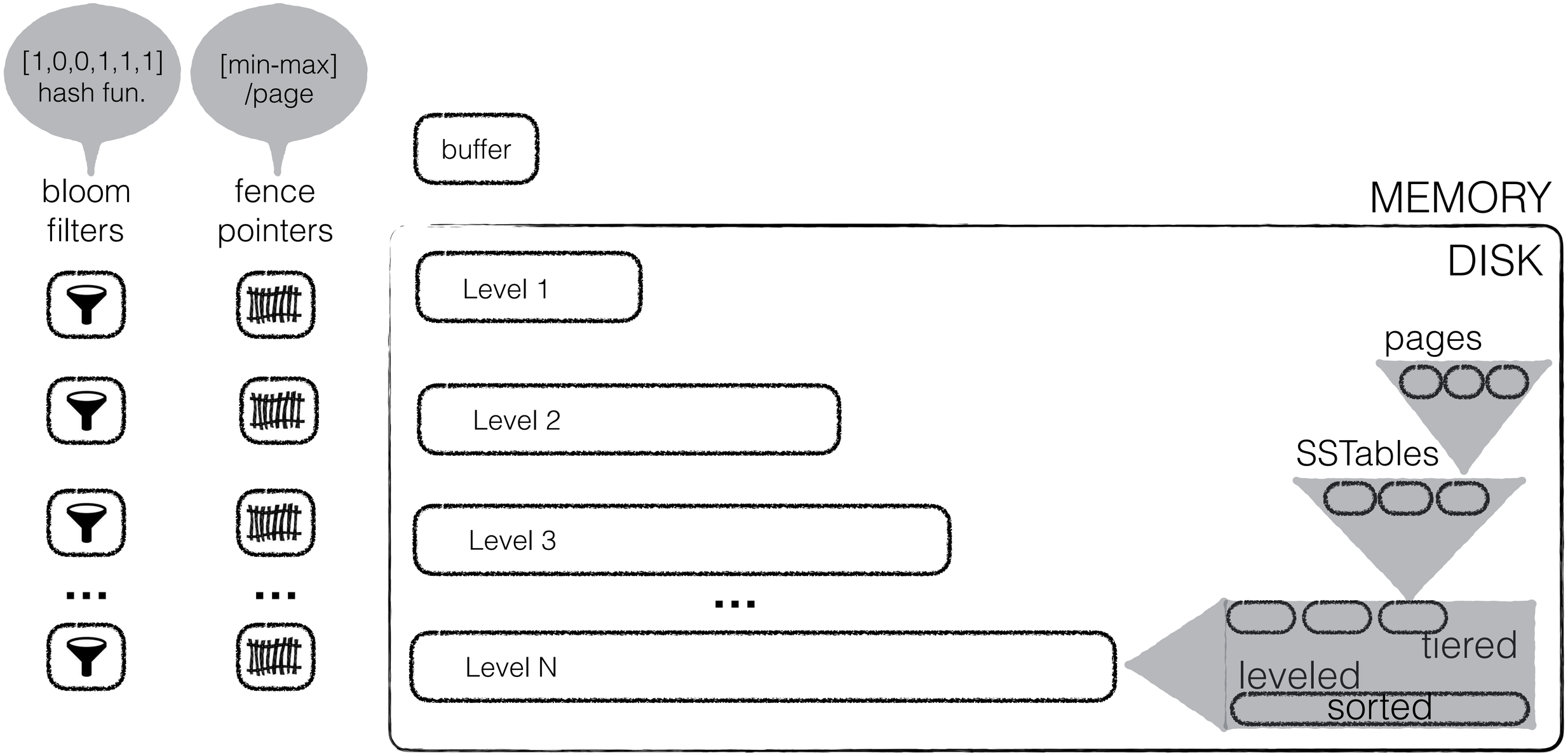

[1,0,0,1,1,1]
hash fun.
[min-max]
/page
bloom
filters
fence
pointers
get (key)
buffer
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N
pages
SSTables
tiered
leveled
sorted

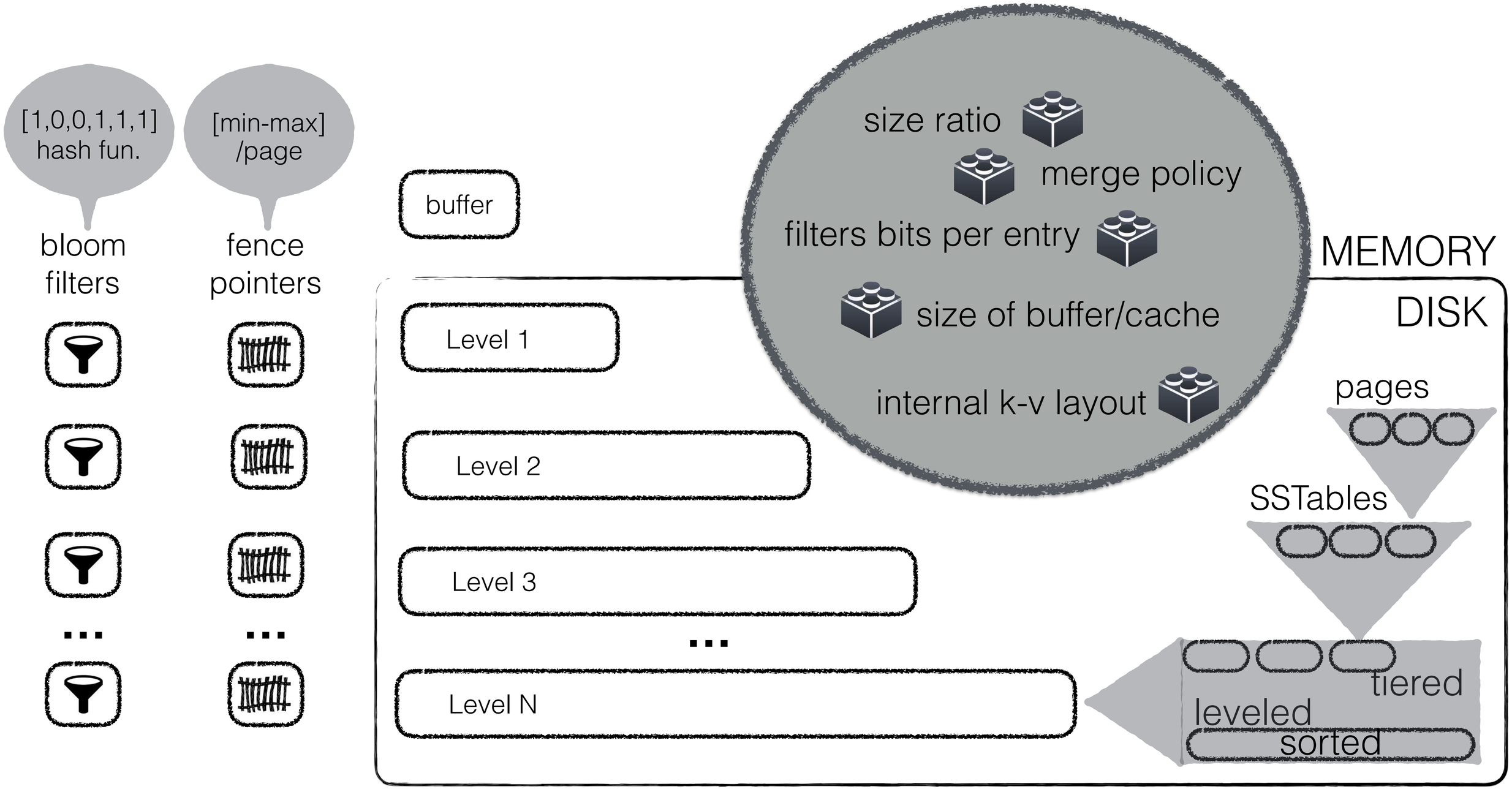

[1,0,0,1,1,1]
hash fun.
[min-max]
/page
bloom
filters
fence
pointers
...
...
get (key)
buffer
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N
pages
SSTables
tiered
leveled
sorted

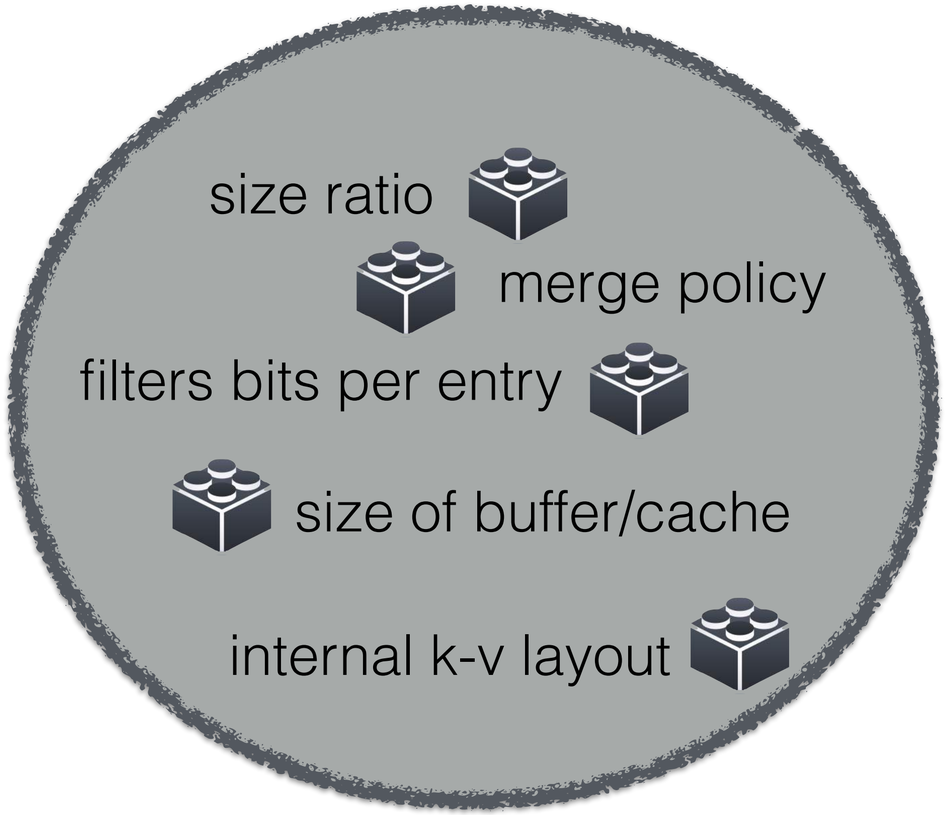



get (key)
[1,0,0,1,1,1]
hash fun.
[min-max]
/page
buffer
bloom
filters
fence
pointers
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N
pages
SSTables
tiered
leveled
sorted

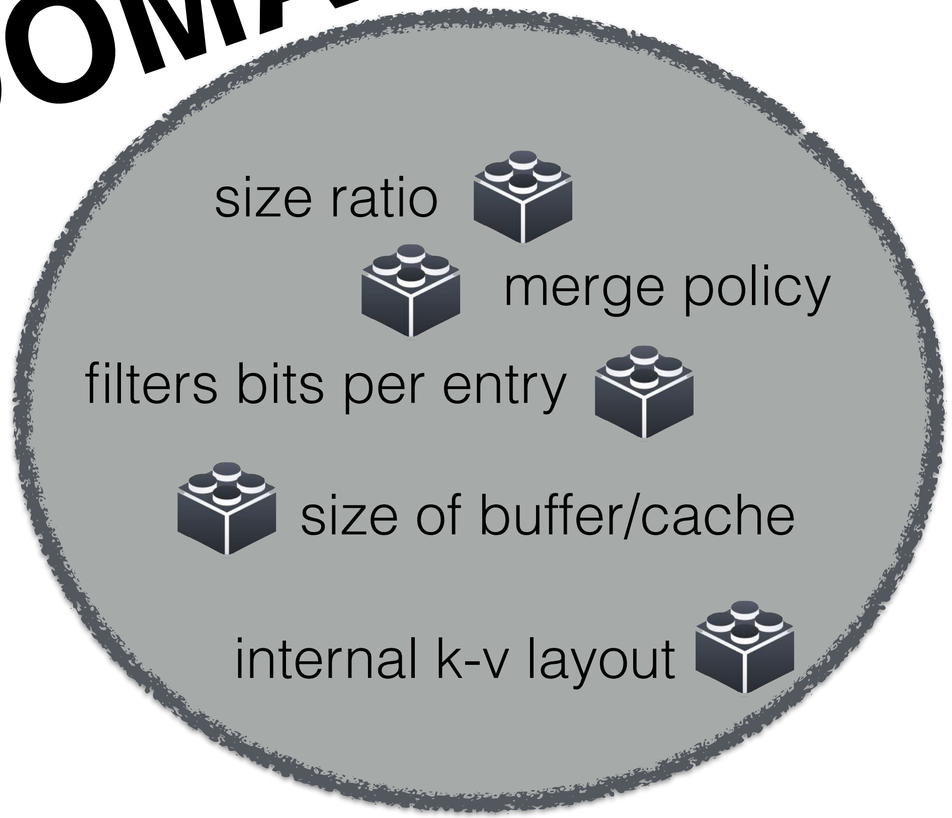

get (key)
[1,0,0,1,1,1]
hash fun.
[min-max]
/page
buffer
bloom
filters
fence
pointers
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N
pages
SSTables
tiered
leveled
sorted


[1,0,0,1,1,1]
hash fun.
[min-max]
/page
bloom
filters
fence
pointers
buffer
MEMORY
DISK
Level 1
Level 2
Level 3
...
Level N
pages
SSTables
tiered
leveled
sorted

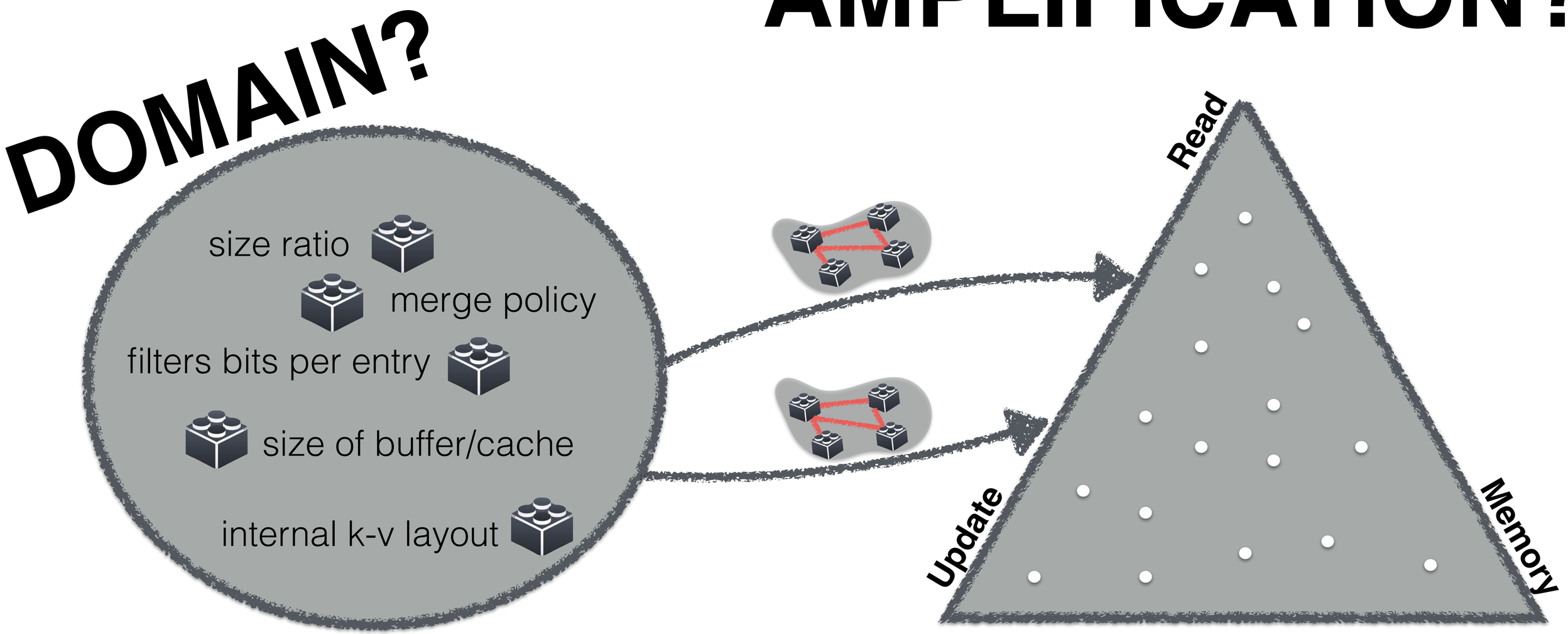



size ratio
merge policy
filters bits per entry
size of buffer/cache
internal k-v layout

DOMAIN?


size ratio
merge policy
filters bits per entry
size of buffer/cache
internal k-v layout

AMPLIFICATION?



merging

writes

reads

when we do more

merging

writes

reads

when we do more

merging

writes ↑

↓ **reads**

merging

Tiering
write-optimized

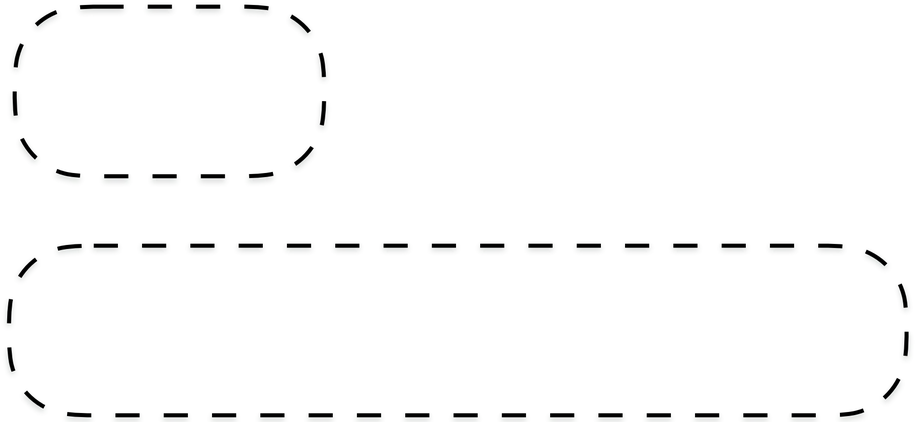
cassandra

Leveling
read-optimized

RocksDB

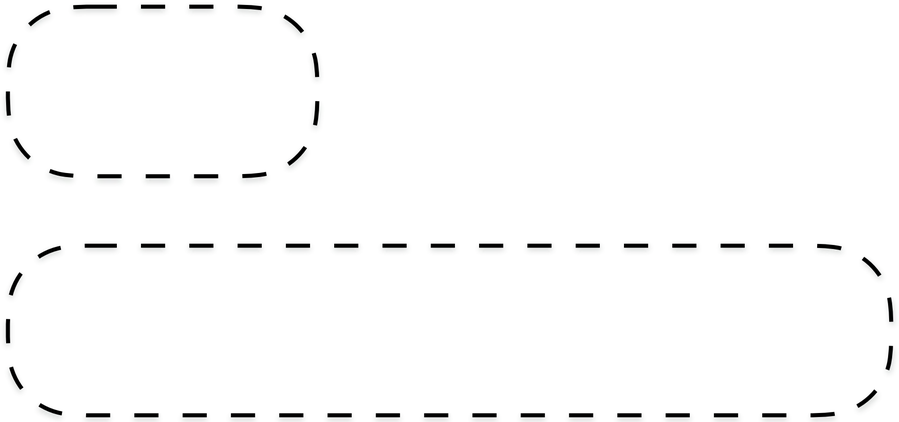
Tiering

write-optimized



Leveling

read-optimized

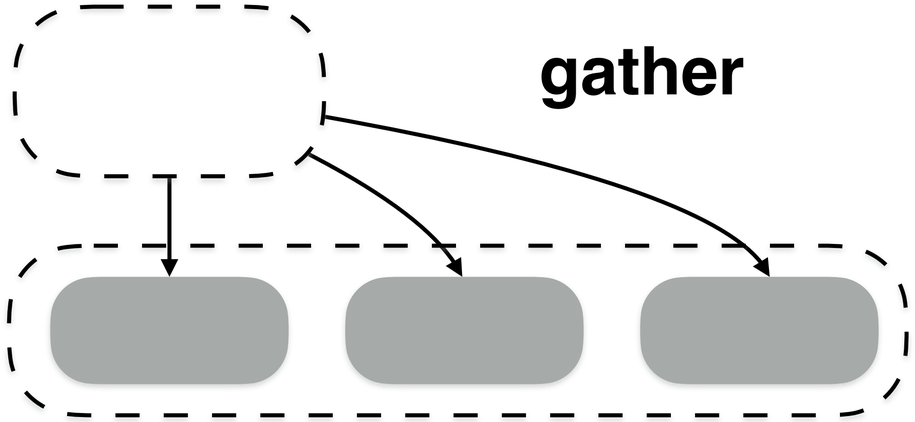


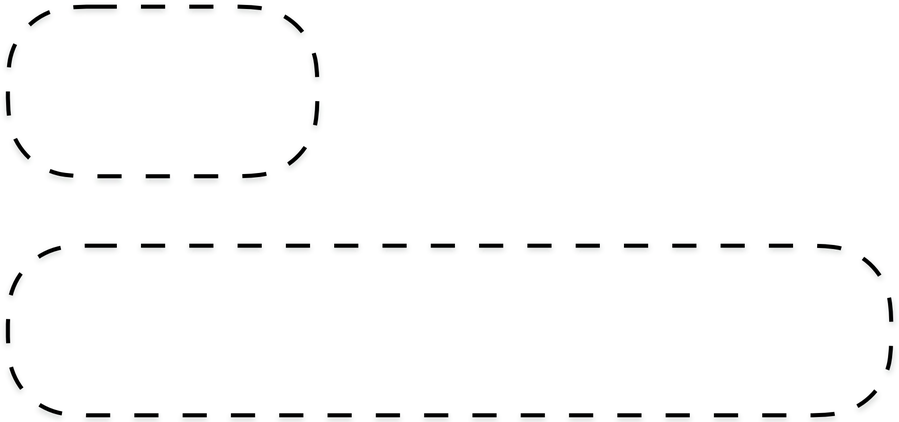
Tiering

write-optimized

Leveling

read-optimized



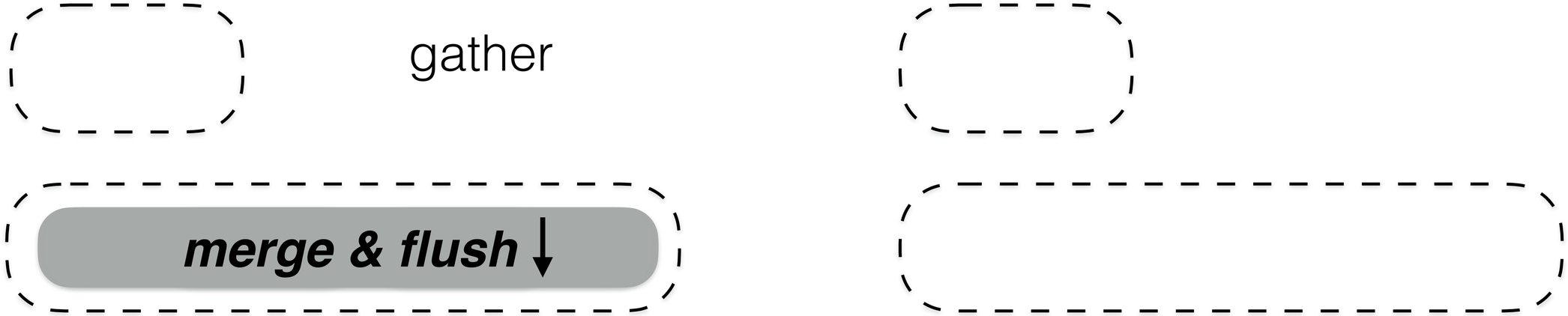


Tiering

write-optimized

Leveling

read-optimized

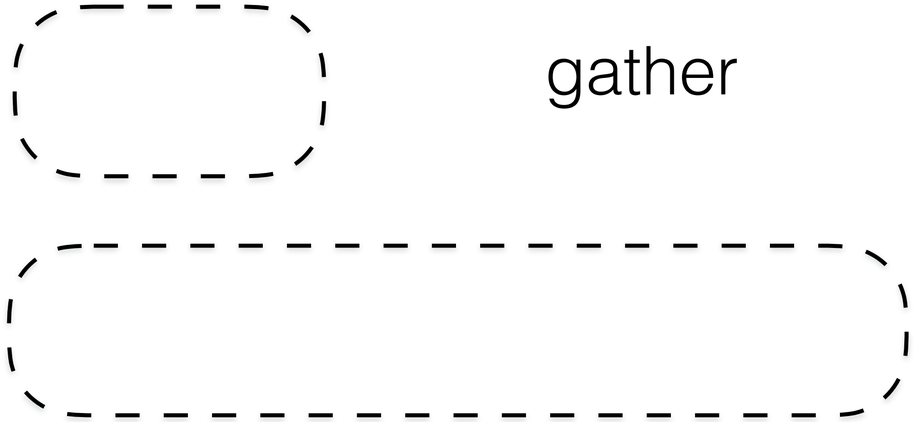


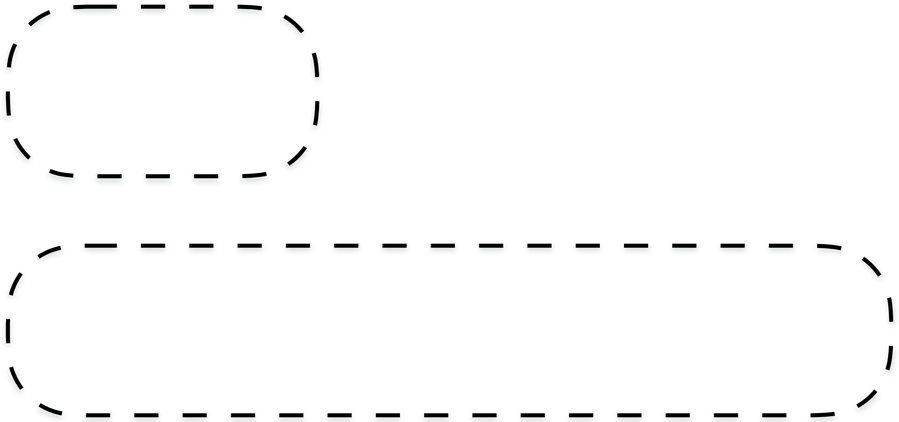
Tiering

write-optimized

Leveling

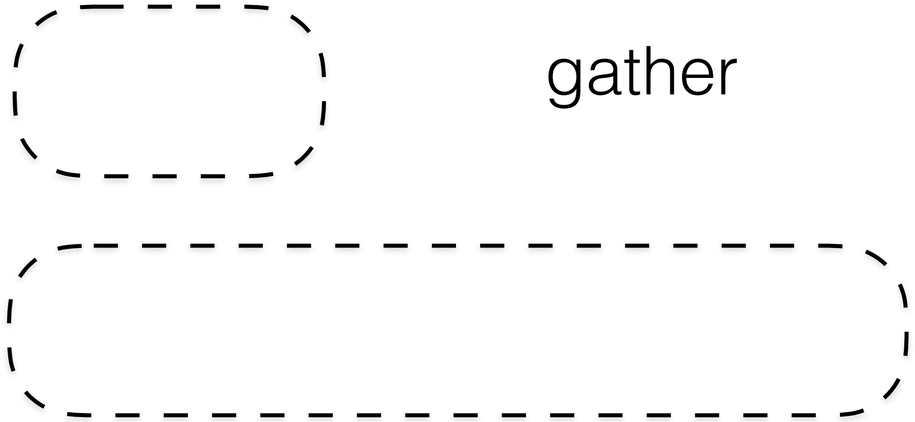
read-optimized





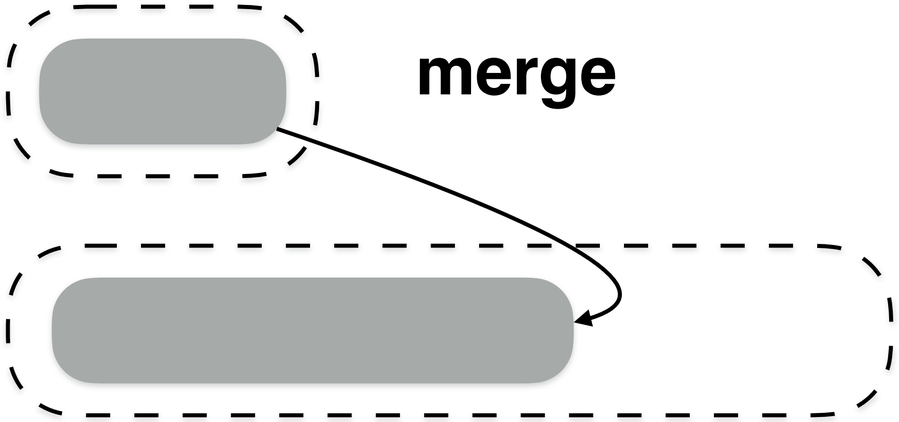
Tiering

write-optimized



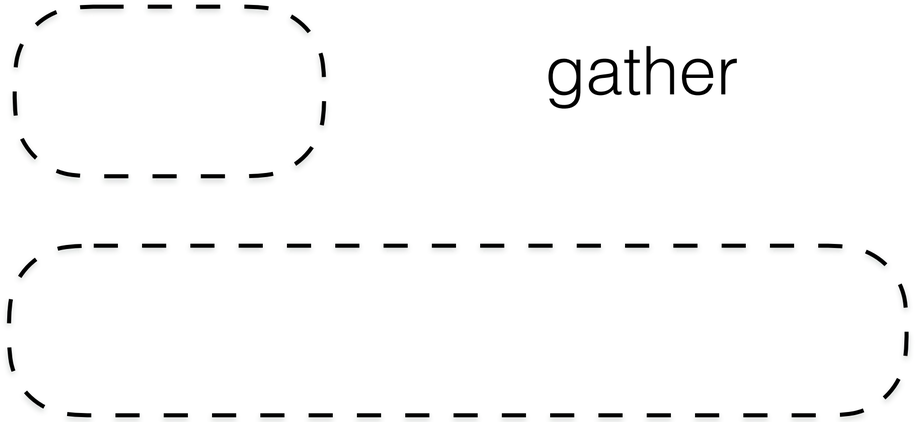
Leveling

read-optimized



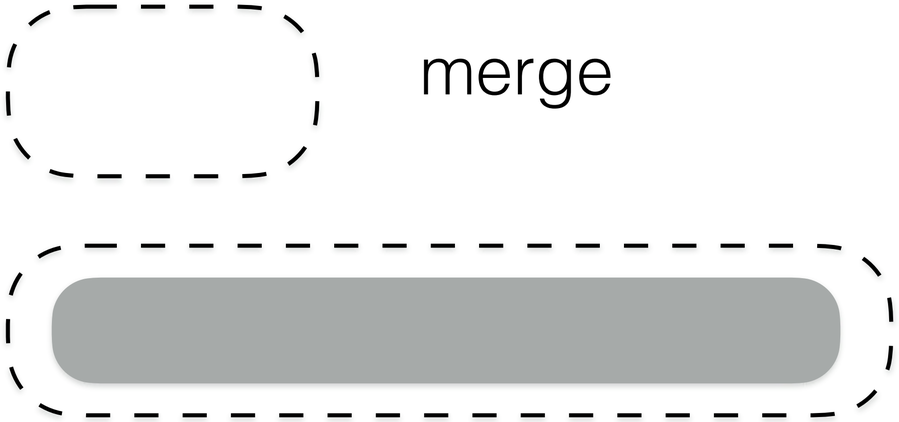
Tiering

write-optimized



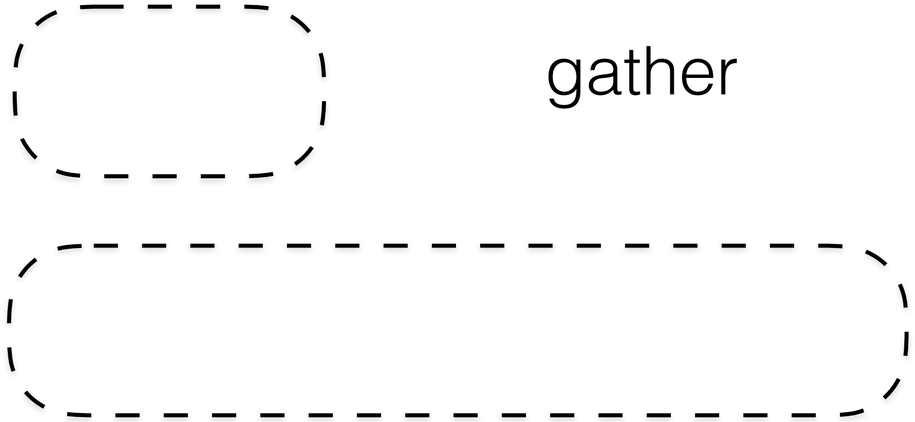
Leveling

read-optimized



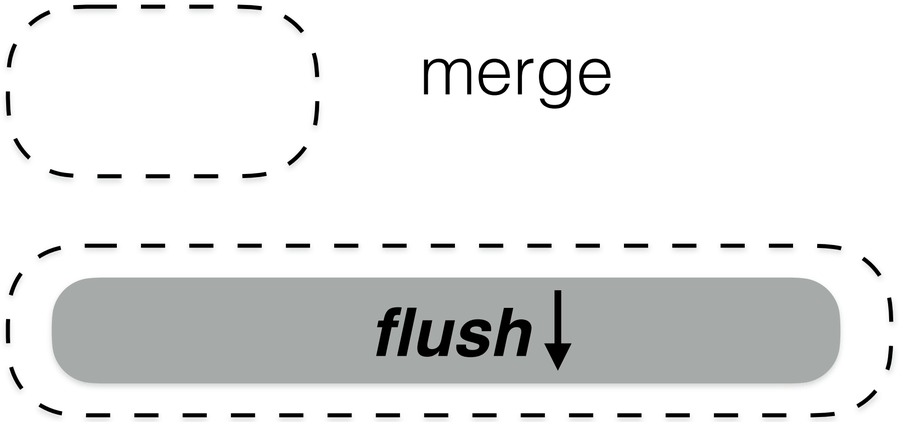
Tiering

write-optimized



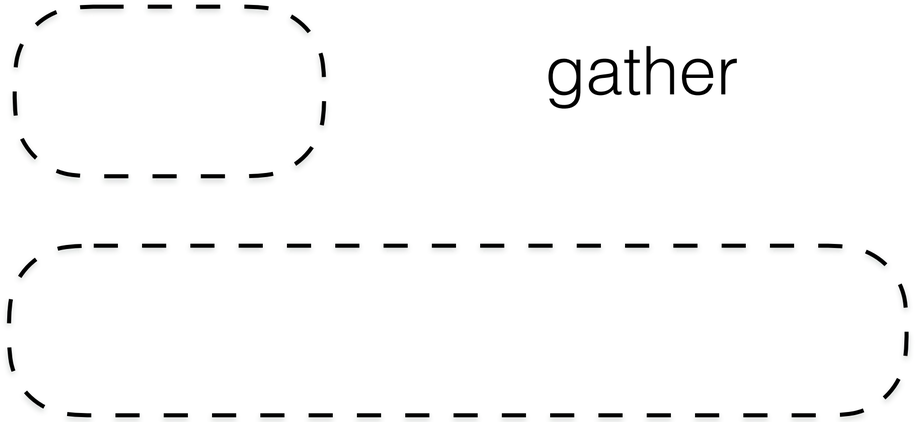
Leveling

read-optimized



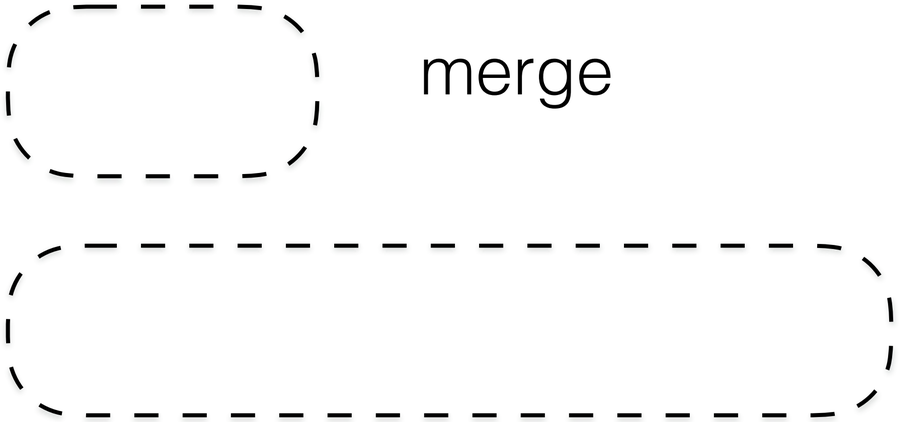
Tiering

write-optimized



Leveling

read-optimized



Tiering

write-optimized

Leveling

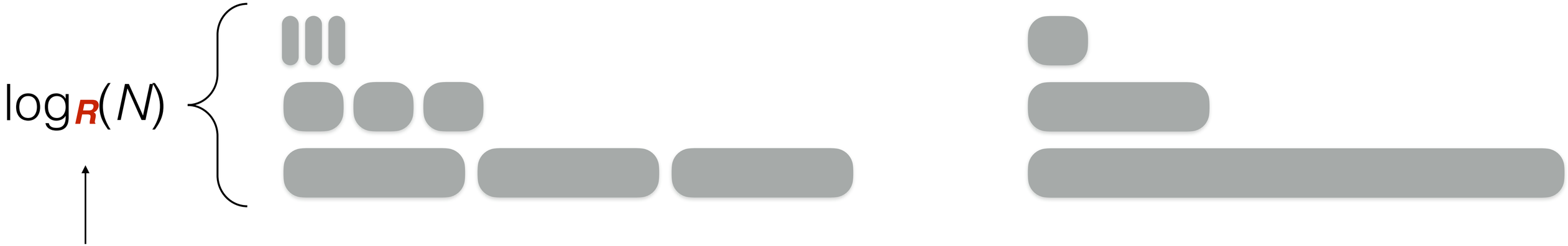
read-optimized

Tiering

write-optimized

Leveling

read-optimized



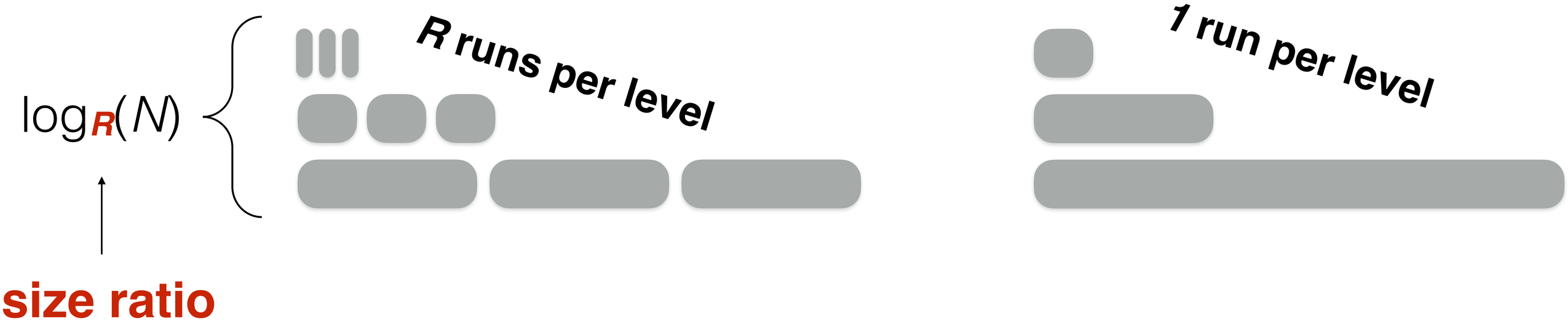
size ratio

Tiering

write-optimized

Leveling

read-optimized

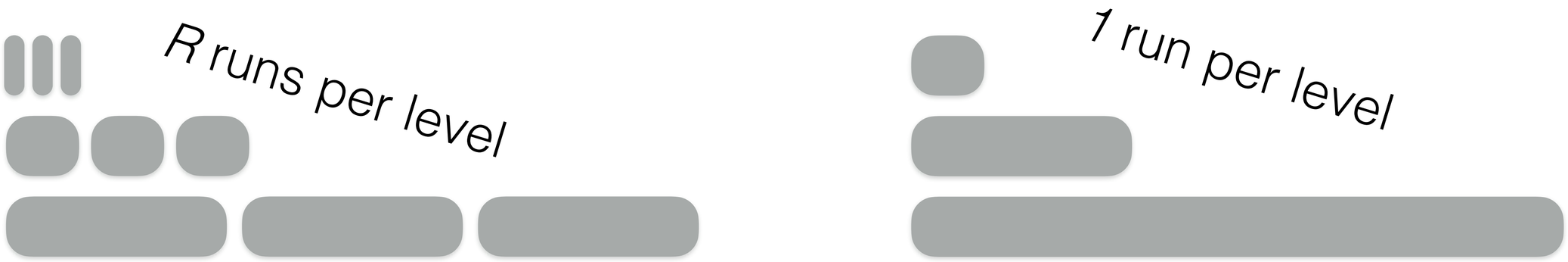


Tiering

write-optimized

Leveling

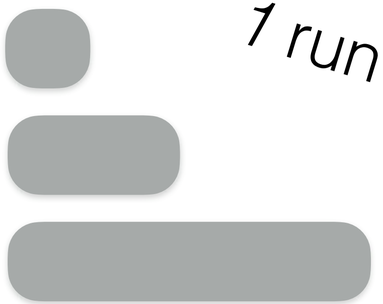
read-optimized



size ratio *R*

Tiering

write-optimized



1 run per level

Leveling

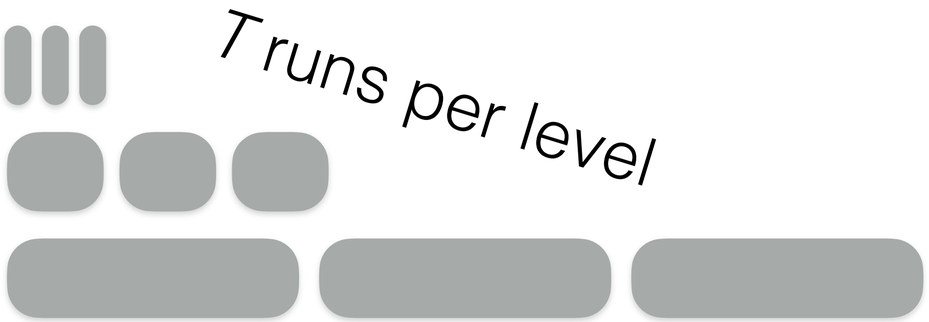
read-optimized

1 run per level

size ratio *R*

Tiering

write-optimized



Leveling

read-optimized

size ratio *R*

Tiering

write-optimized

O(*N*) runs per level

Leveling

read-optimized

1 run per level

size ratio *R* ⌃

Tiering

write-optimized

Leveling

read-optimized

O(*N*) runs per level

log

1 run per level

size ratio *R* ⌃

Tiering

write-optimized

O(*N*) runs per level

log

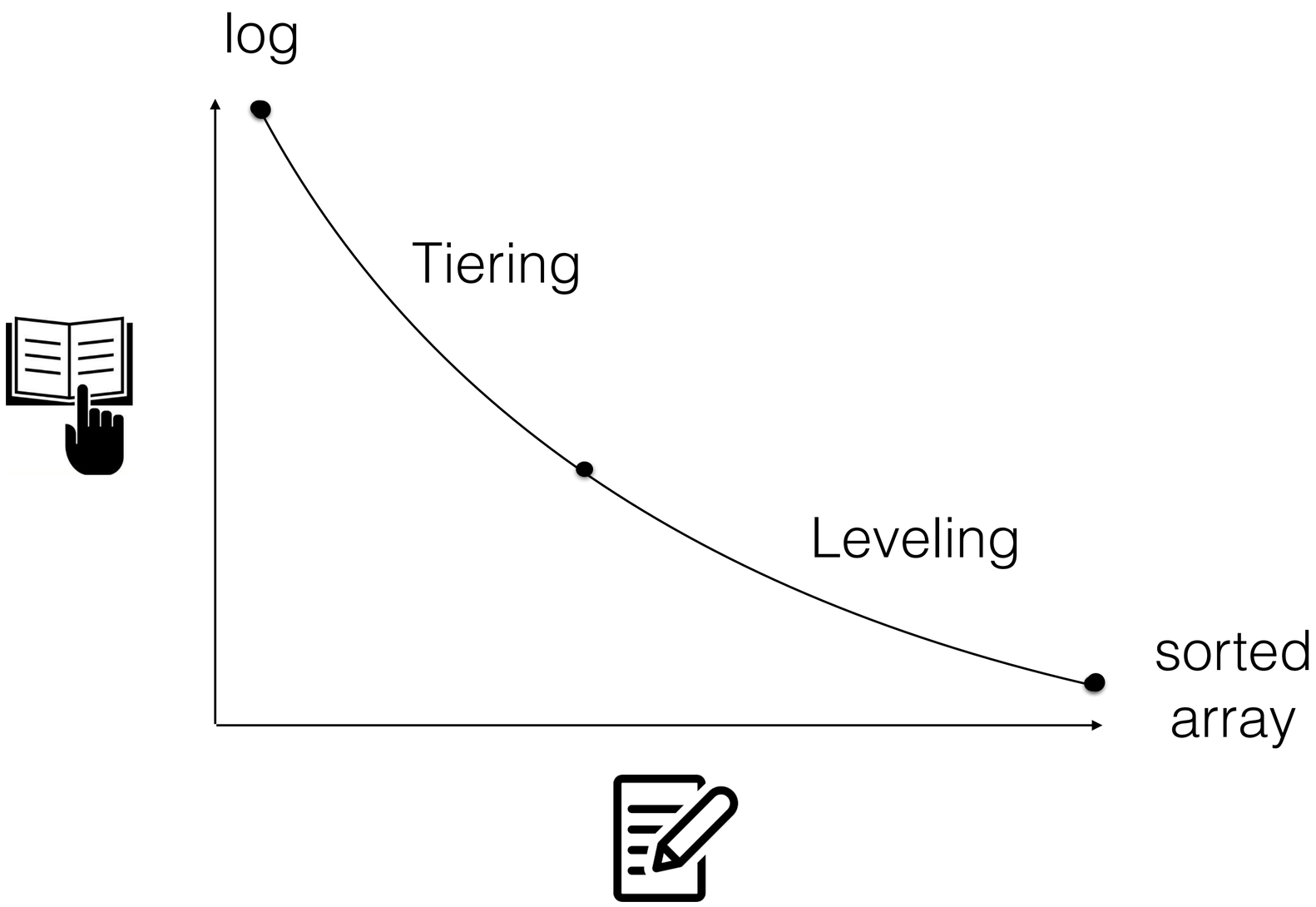
Leveling

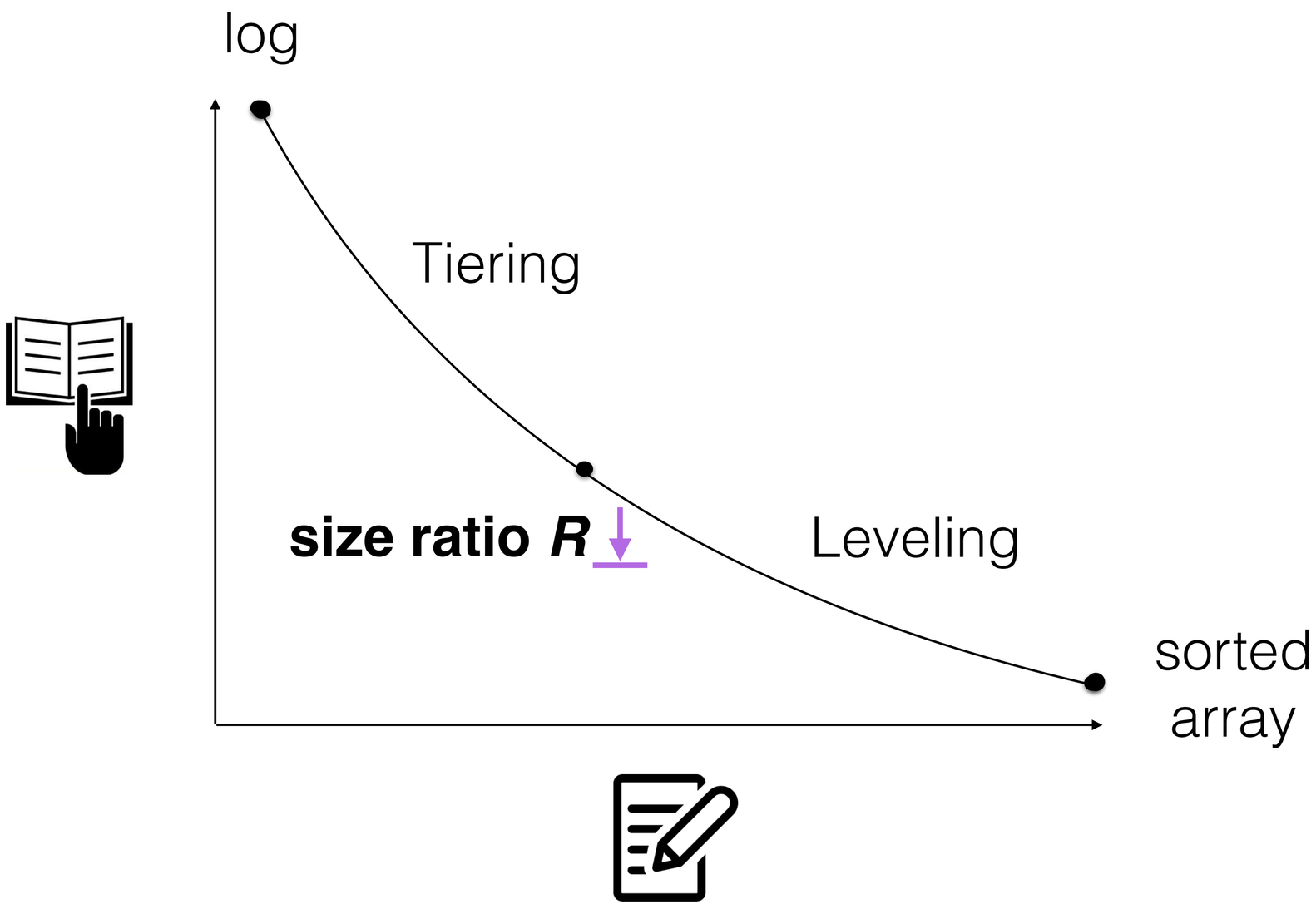
read-optimized

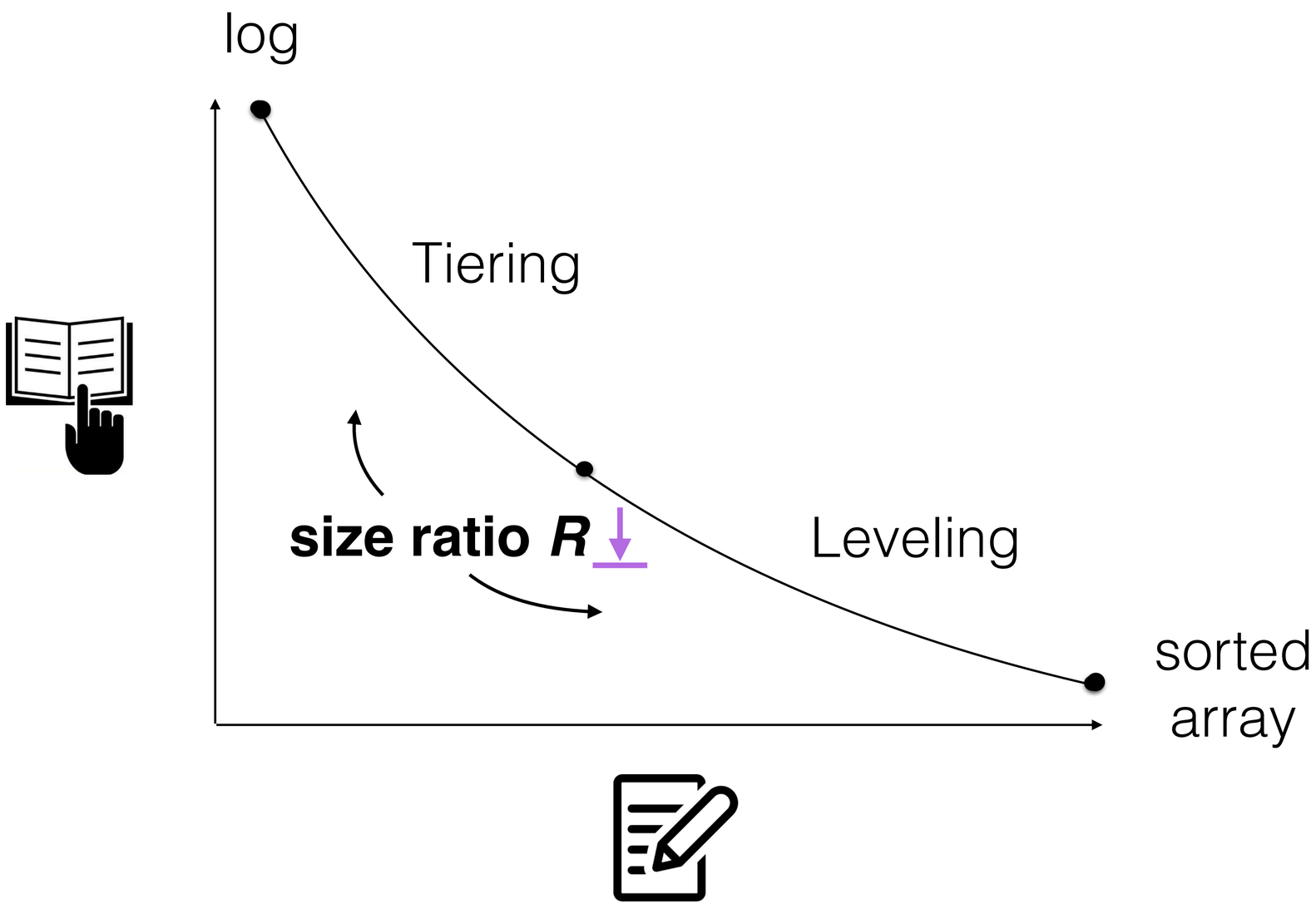
1 run per level

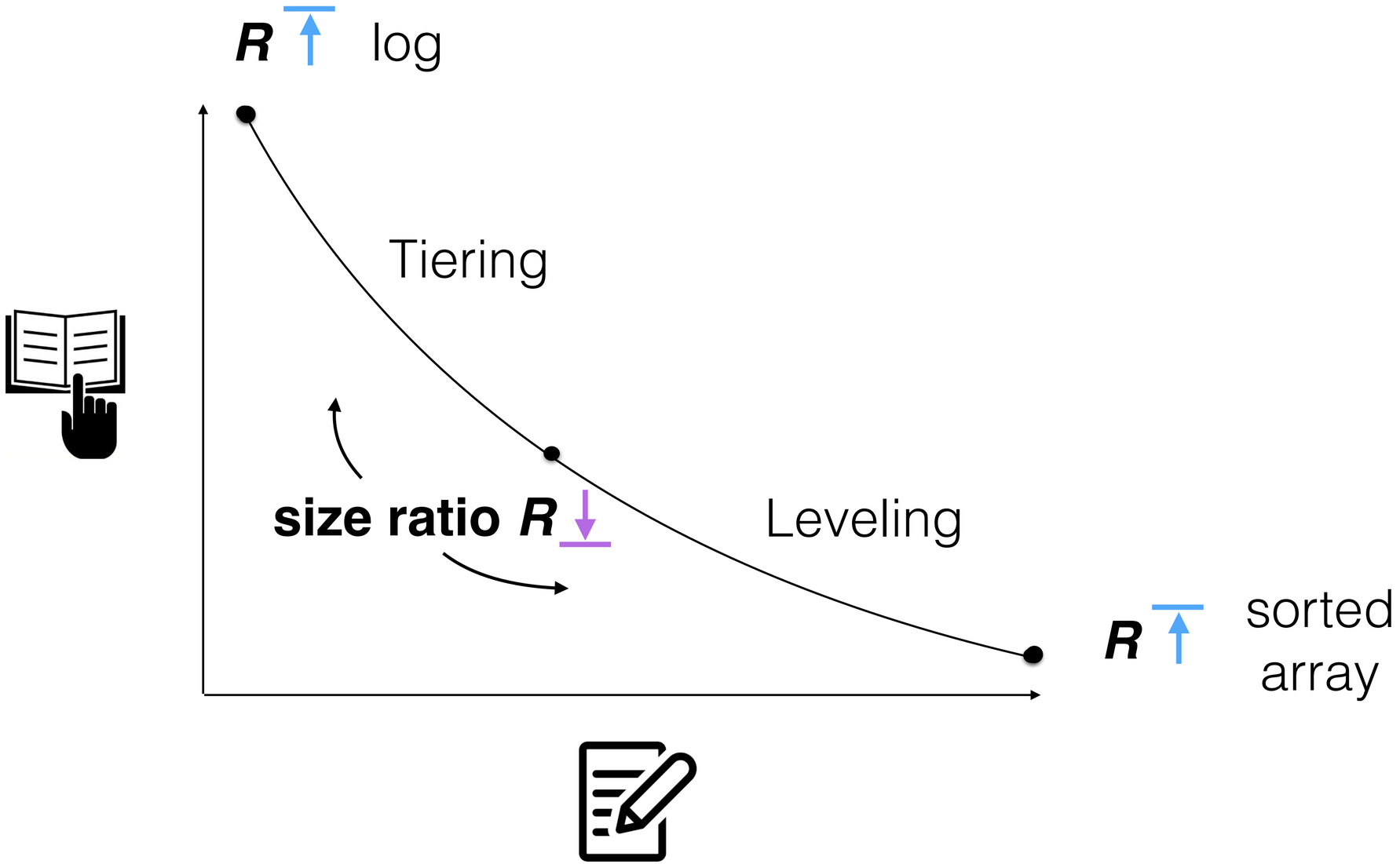
sorted array

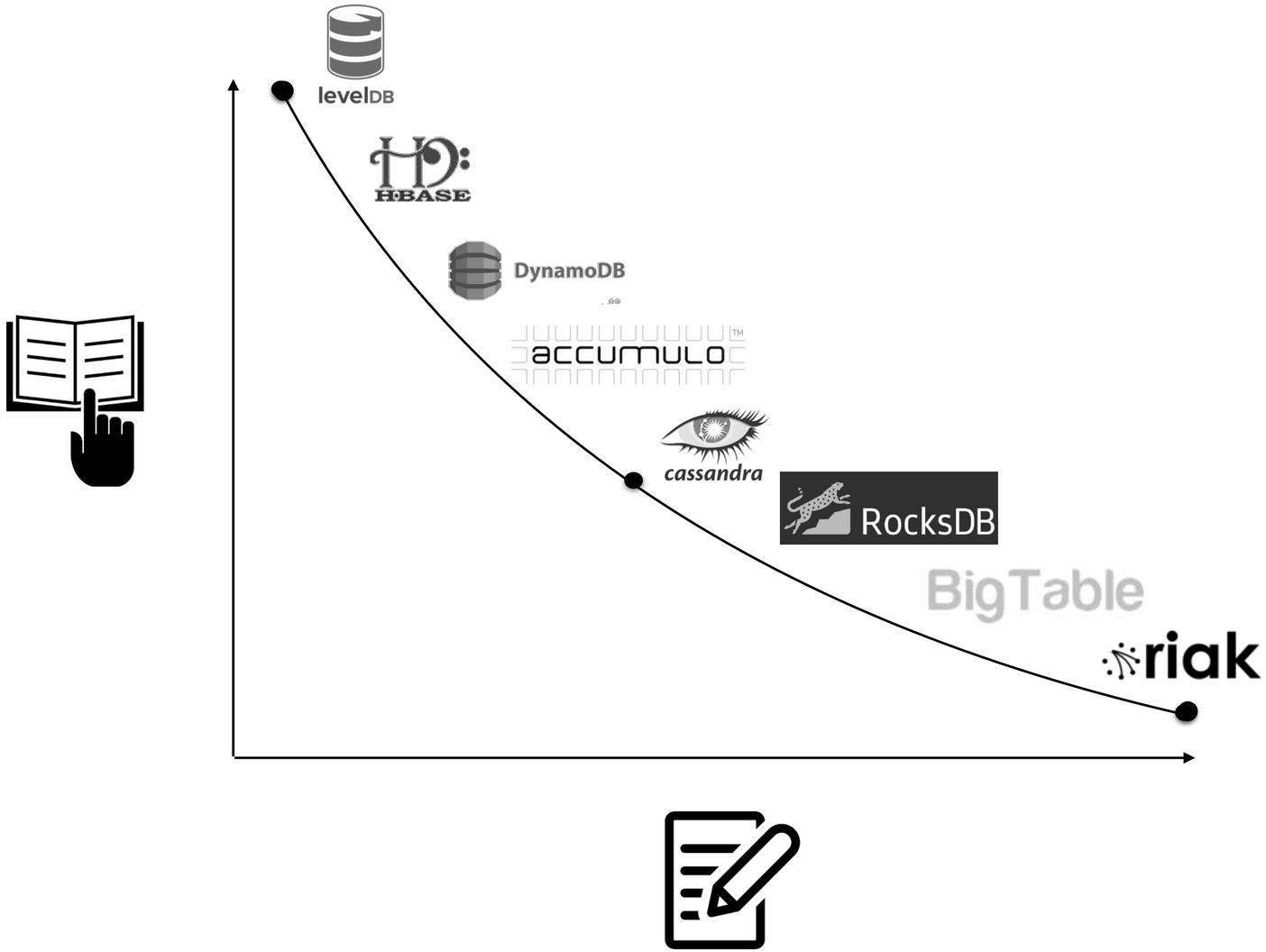
size ratio *R* ⌃

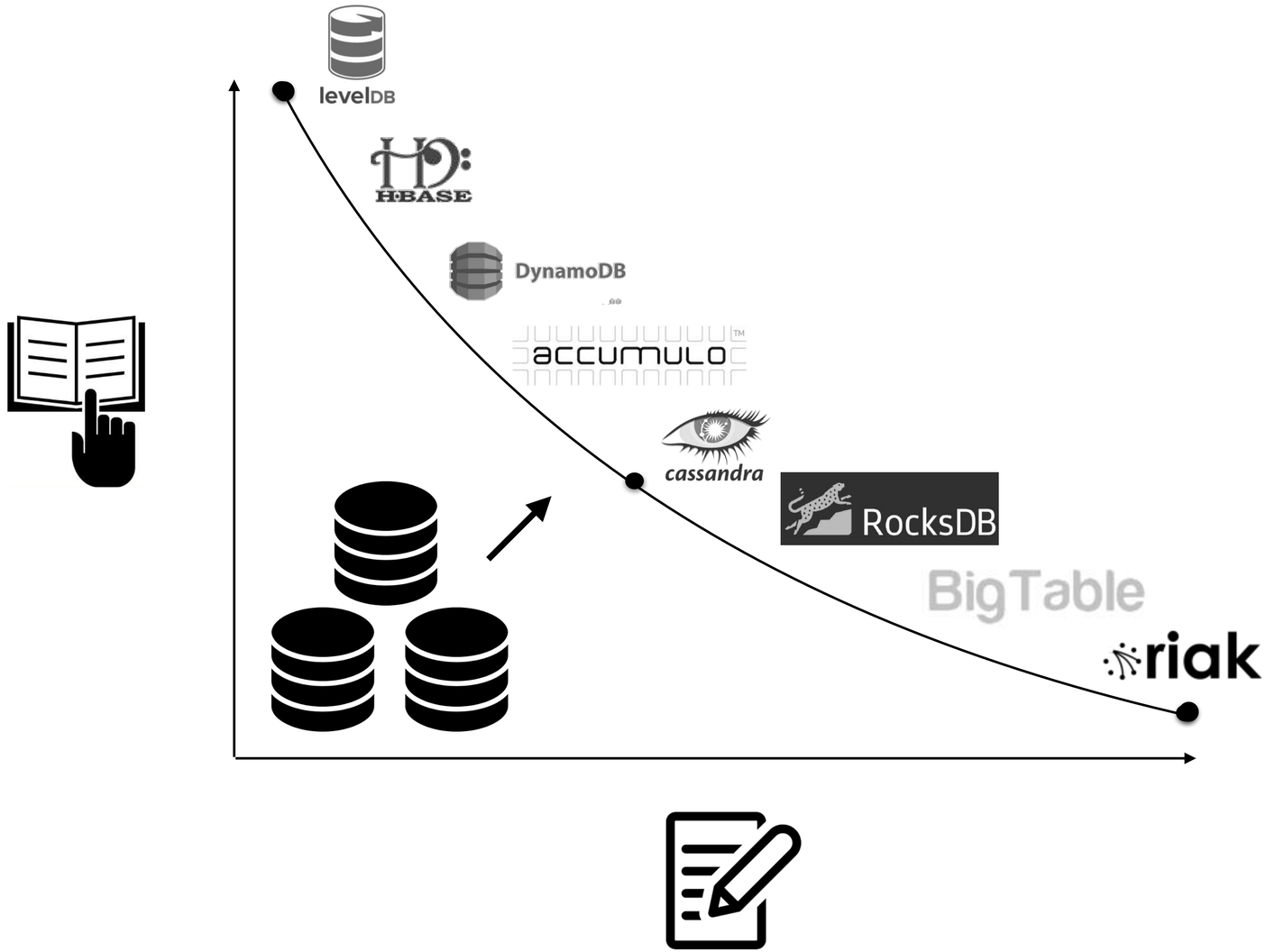

log
Tiering
Leveling
sorted
array


log
Tiering
size ratio R
Leveling
sorted
array

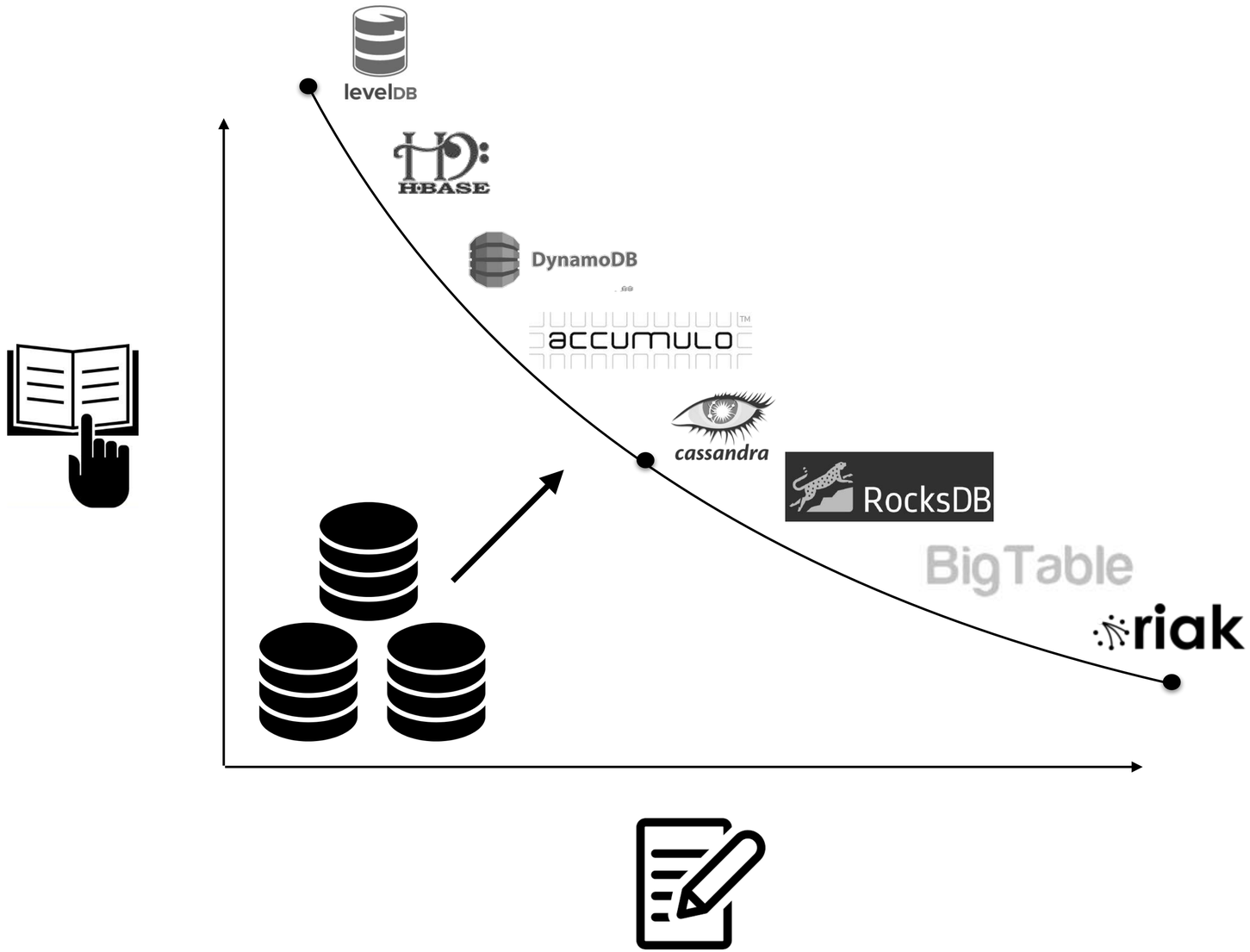

log
Tiering
size ratio R
Leveling
sorted
array


R
log
Tiering
size ratio R
Leveling
R
sorted
array


levelDB
HBASE
DynamoDB
accumulo
cassandra
RocksDB
BigTable
riak

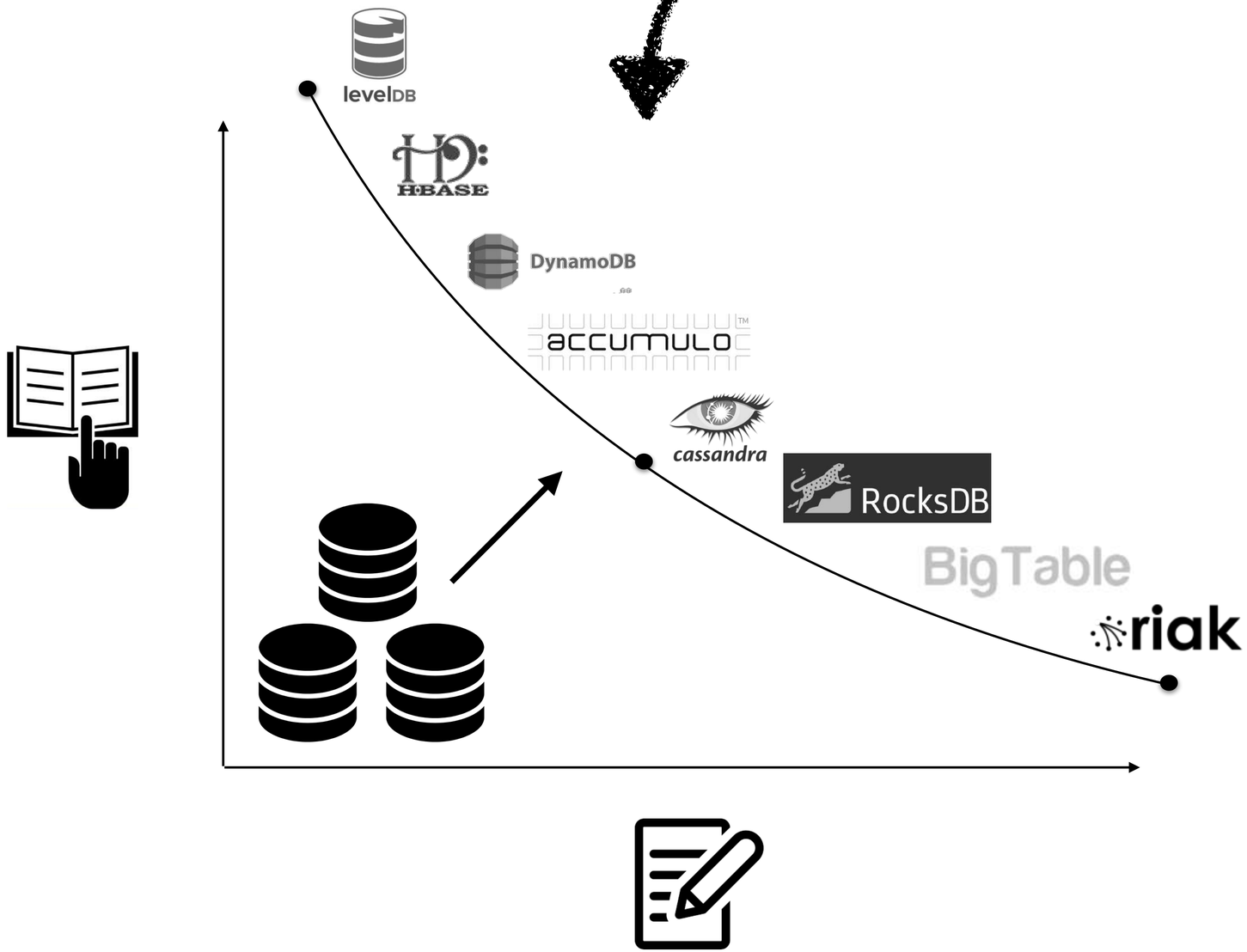


what happens as we collect more data?

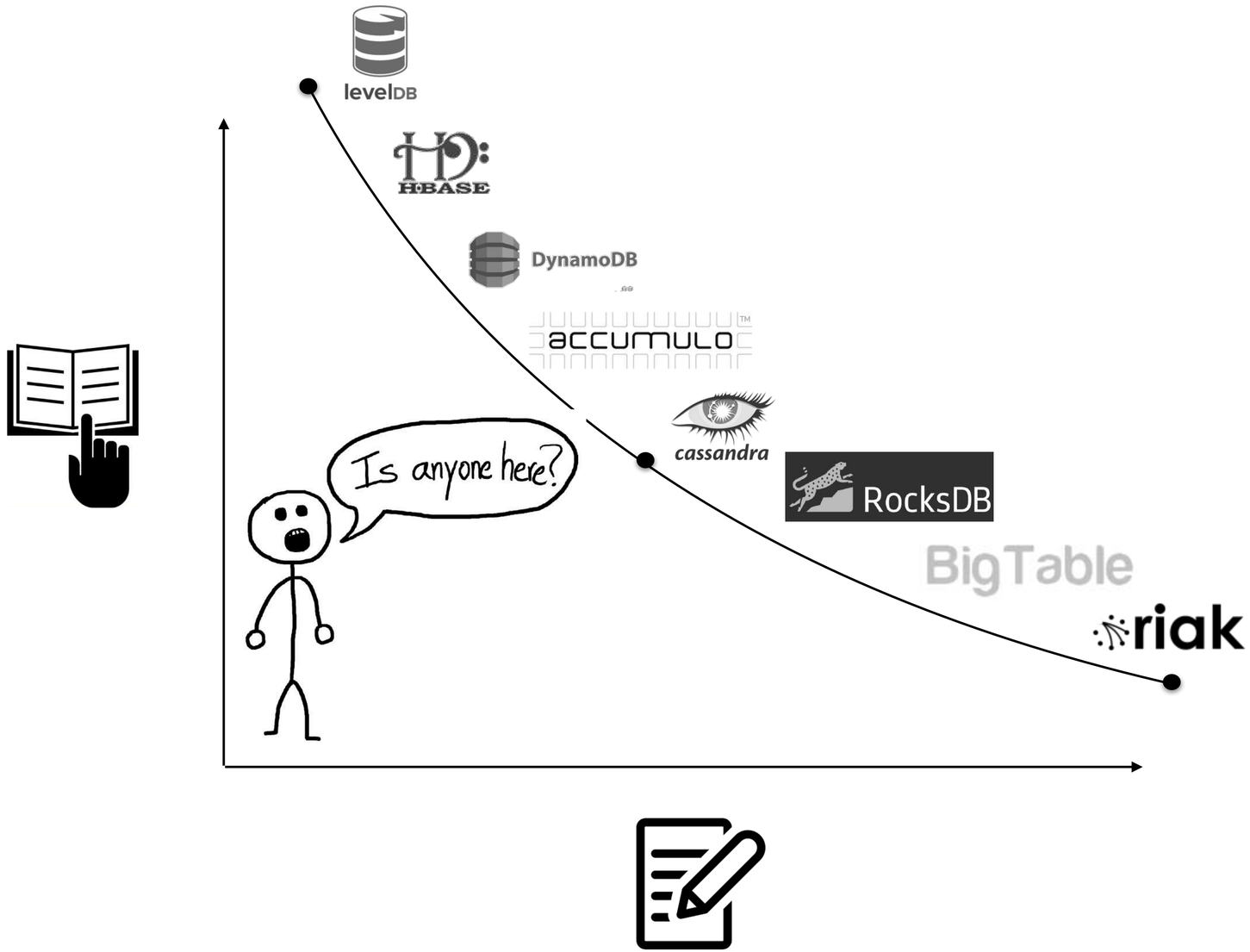


what happens as we collect more data?

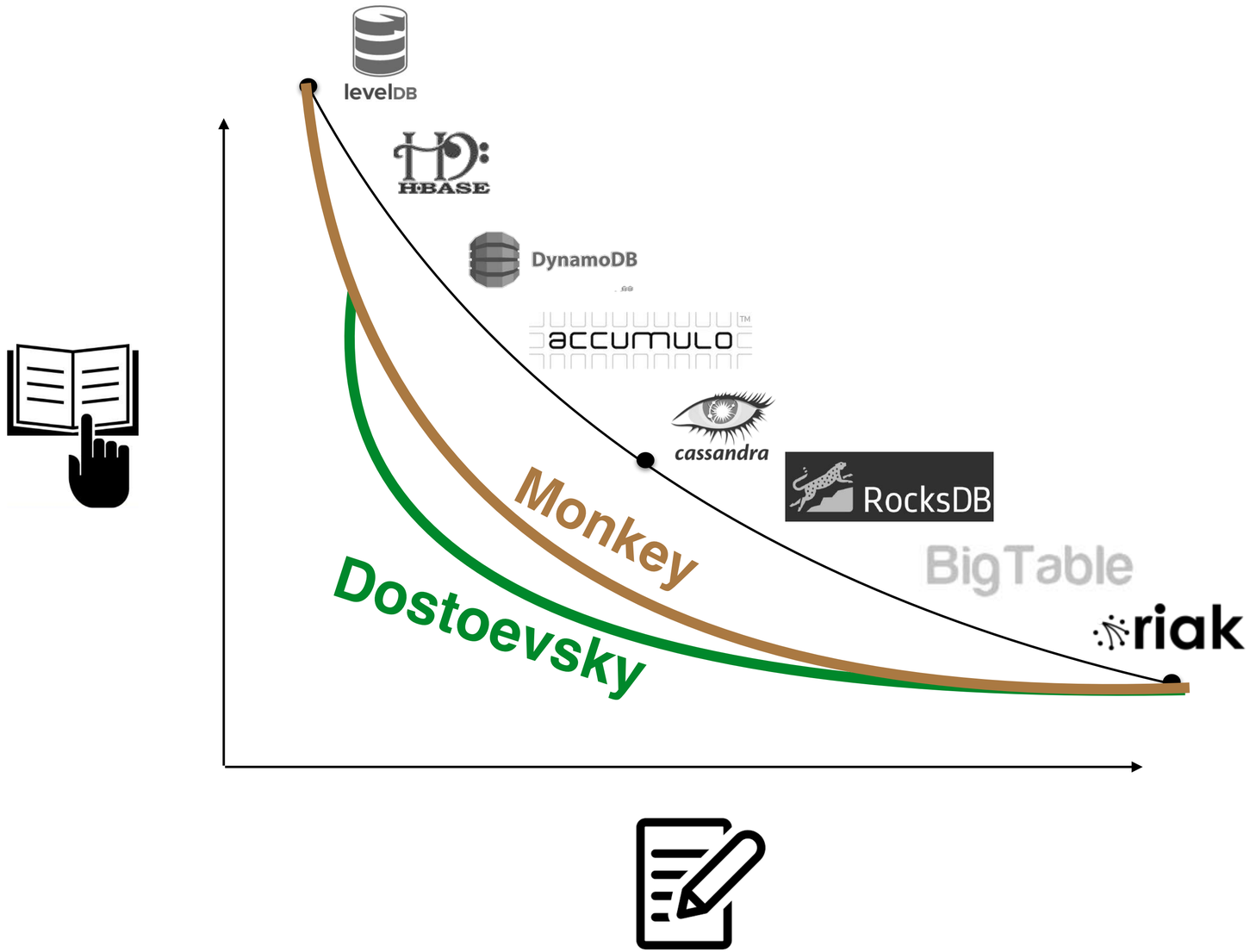
both reads and writes get worse!



what happens as we collect more data?



what happens as we collect more data?



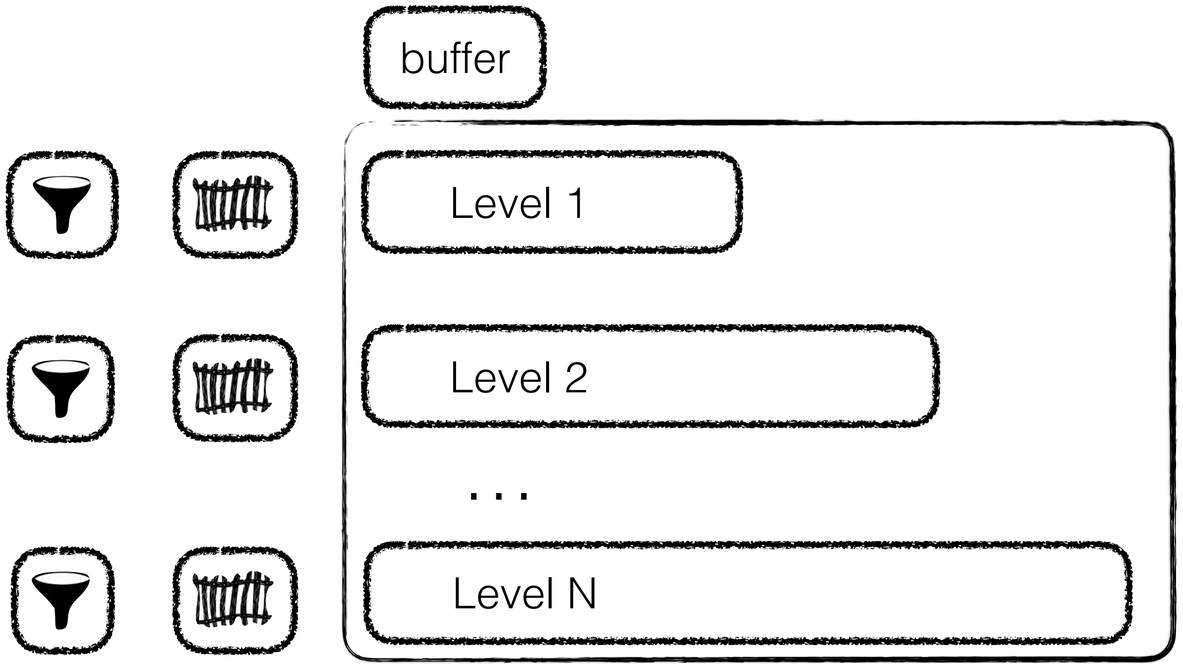
what happens as we collect more data?

BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: **O**ptimal **N**avigable **Key**-Value Store @SIGMOD2017

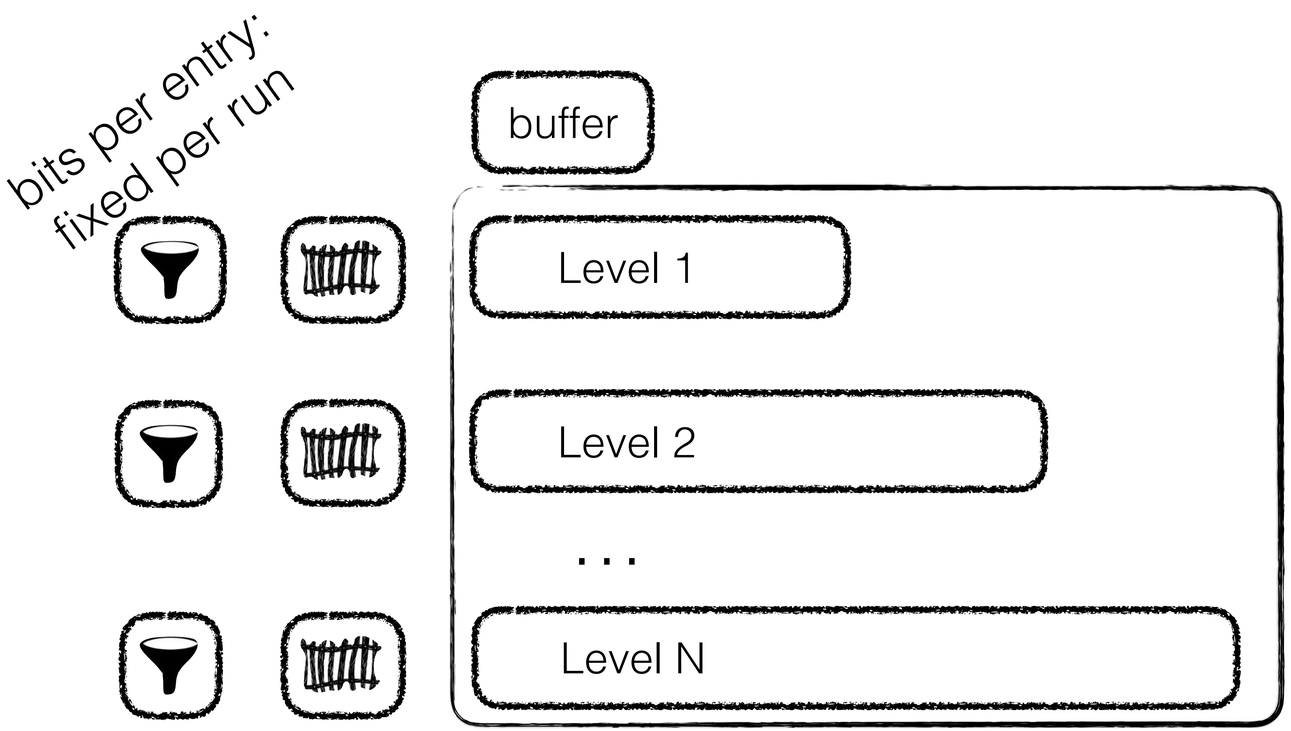
BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: **O**ptimal **N**avigable **Key**-Value Store @SIGMOD2017



BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: **O**ptimal **N**avigable **Key**-Value Store @SIGMOD2017

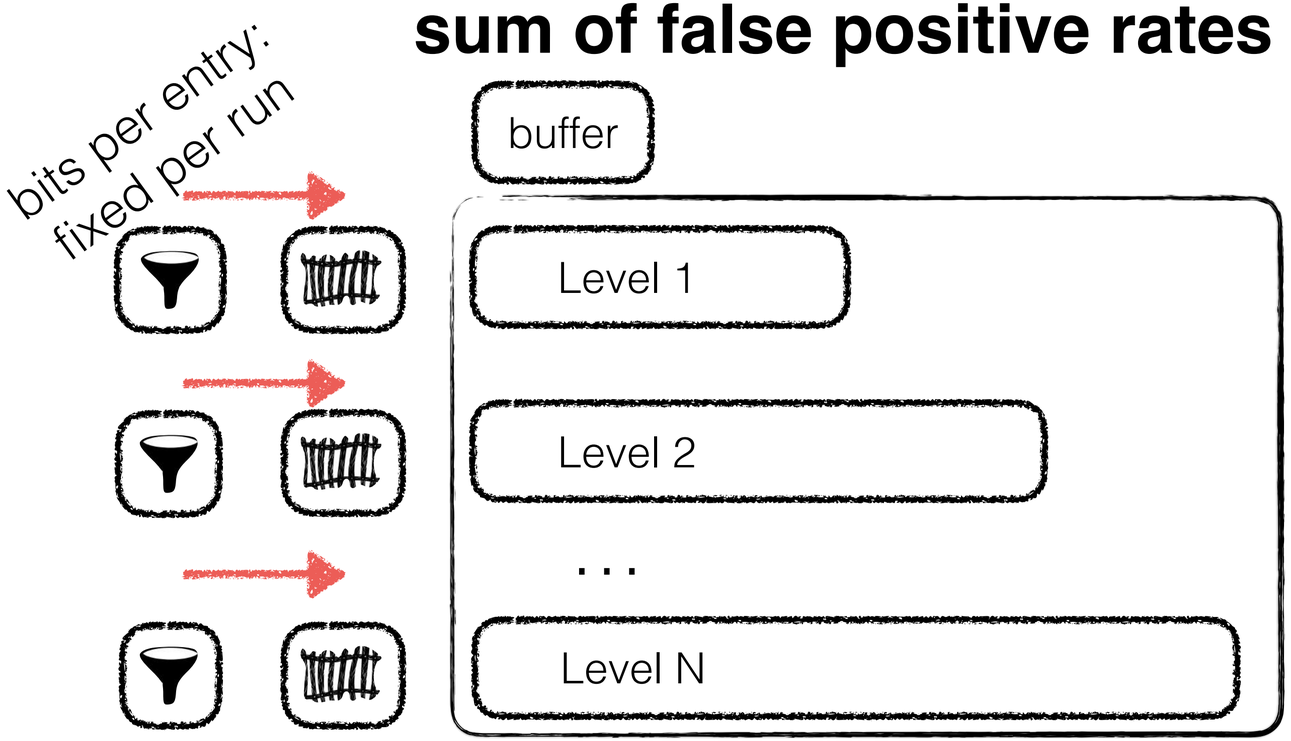


DASlab
@ Harvard SEAS

BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: **O**ptimal **N**avigable **Key**-Value Store @SIGMOD2017

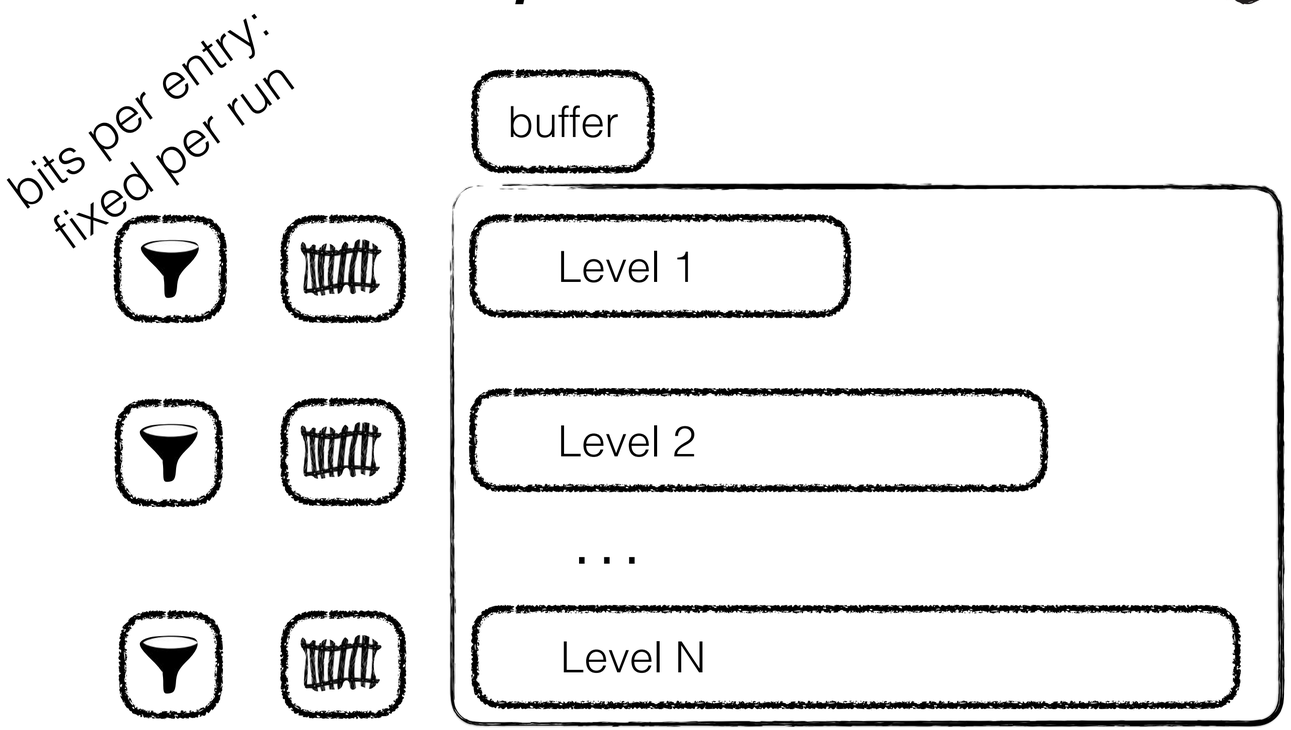
worst lookup cost:
sum of false positive rates



BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: **O**ptimal **N**avigable **Key**-Value Store @SIGMOD2017

the same memory budget
is more impactful at smaller levels

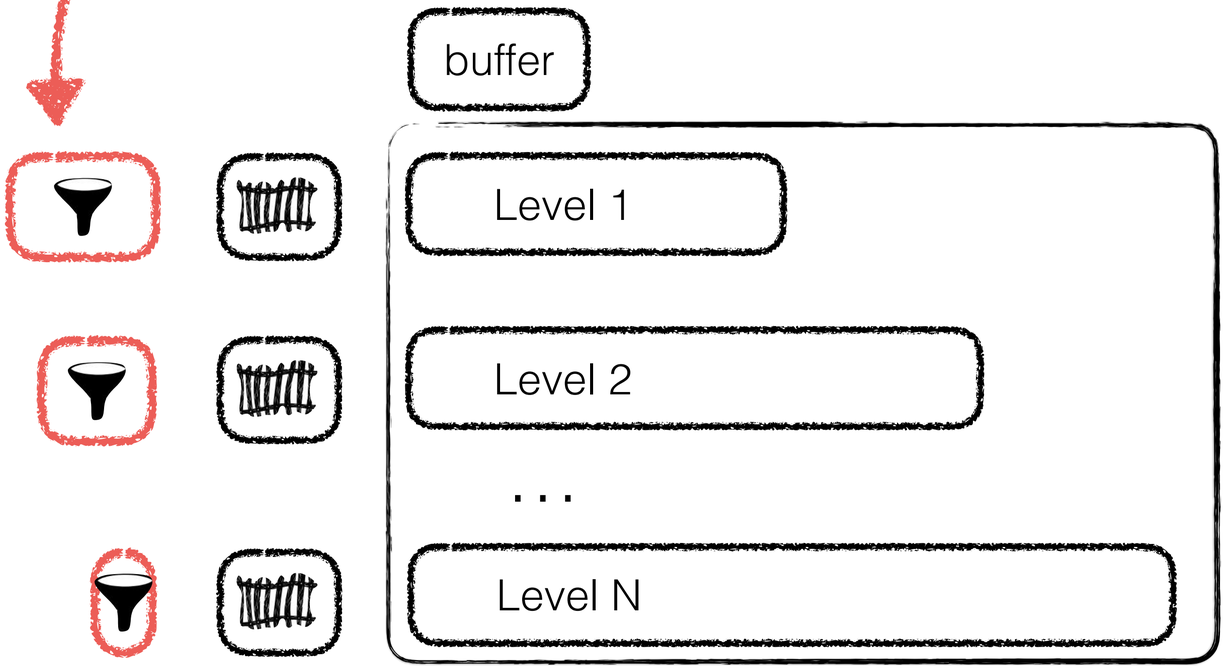


DASlab
@ Harvard SEAS

BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: **O**ptimal **N**avigable **Key**-Value Store @SIGMOD2017

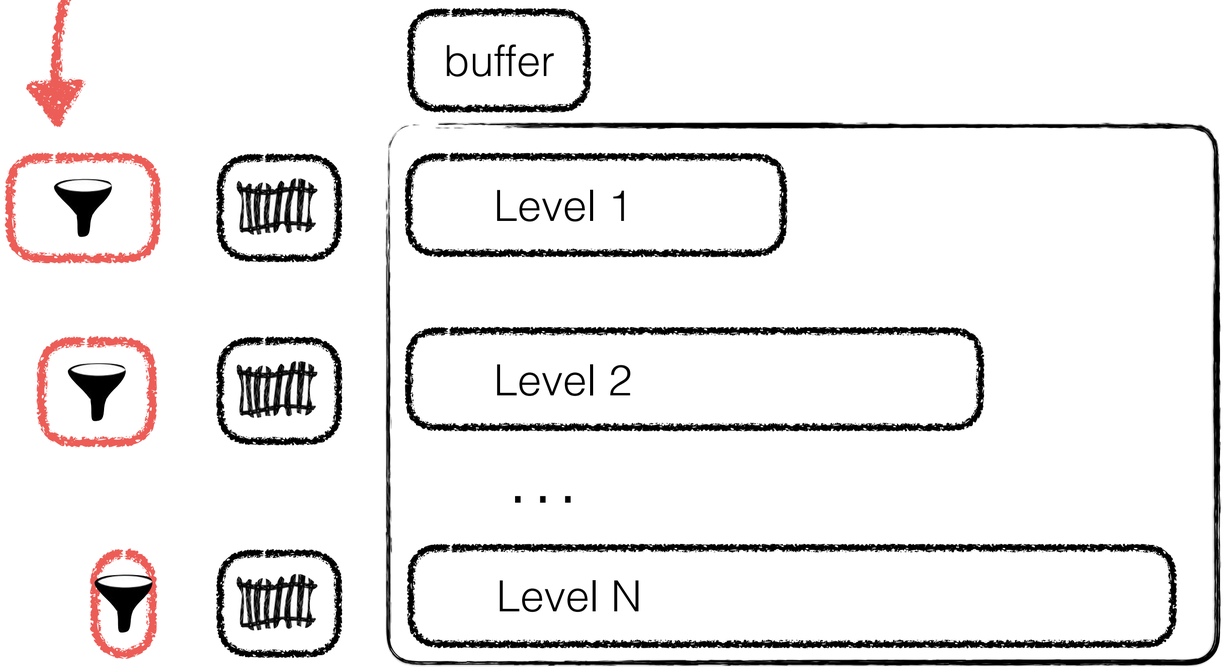
the same memory budget
is more impactful at smaller levels

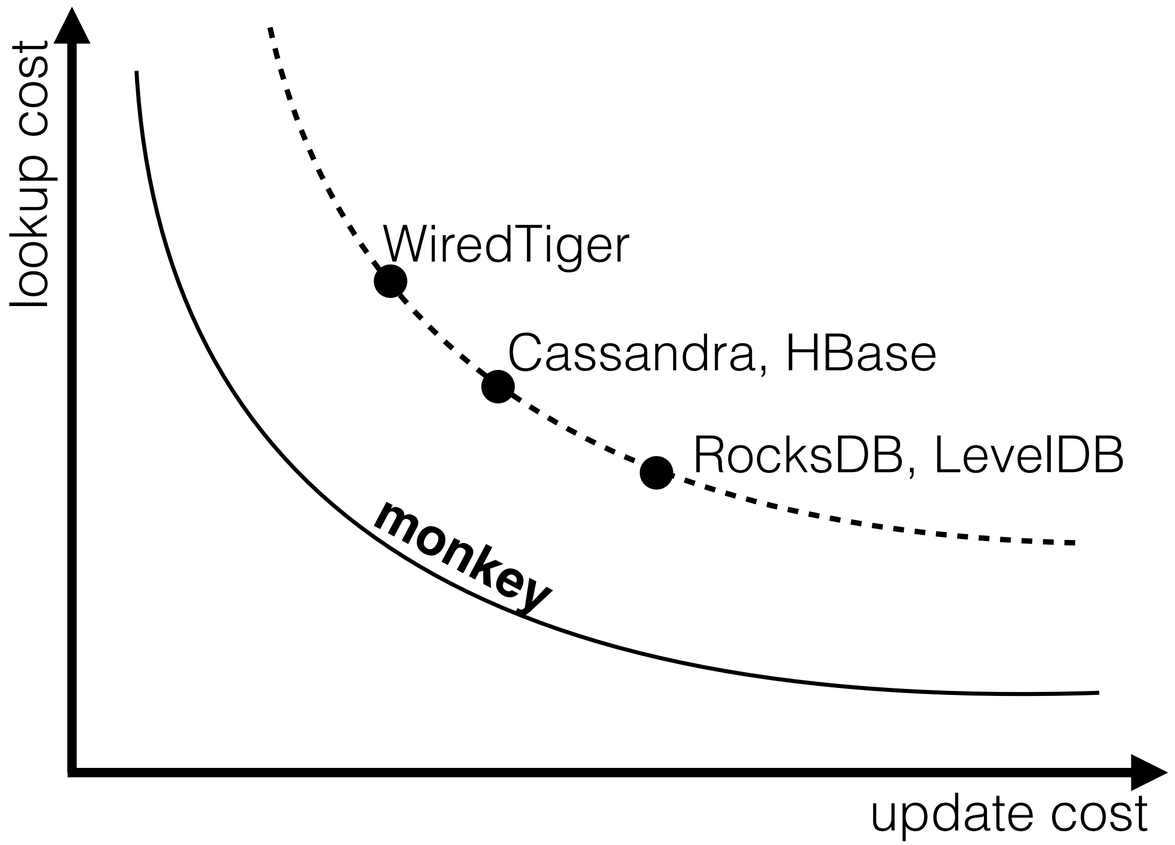


BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: **O**ptimal **N**avigable **Key**-Value Store @SIGMOD2017

the same memory budget is more impactful at smaller levels



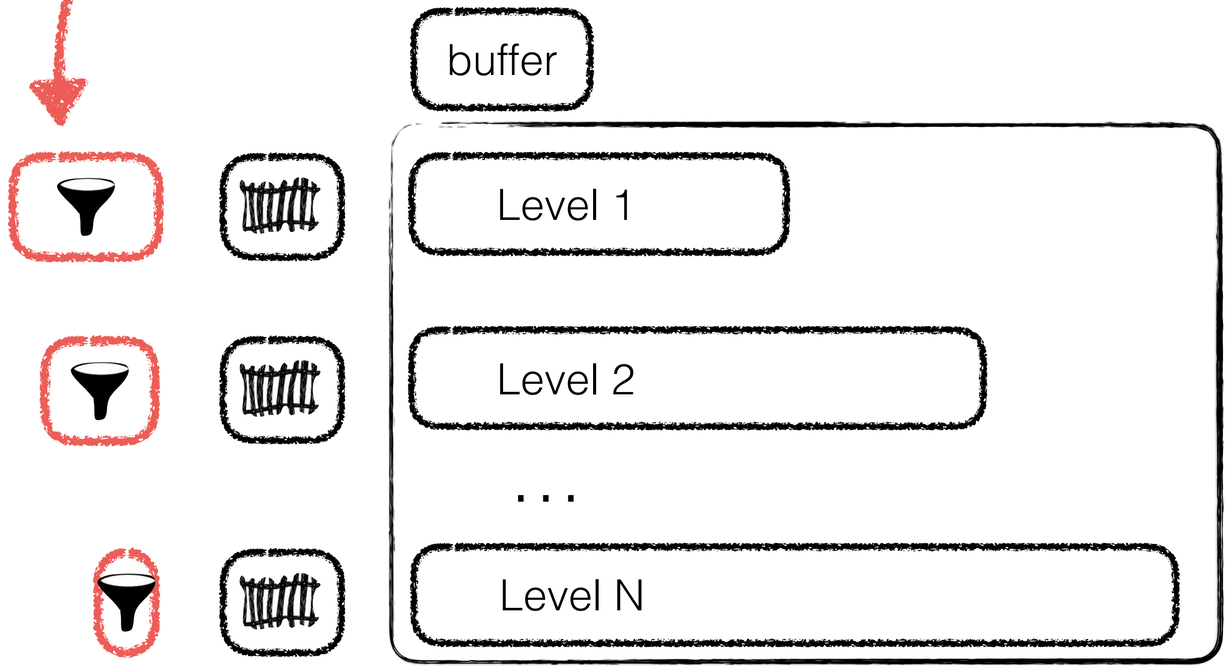


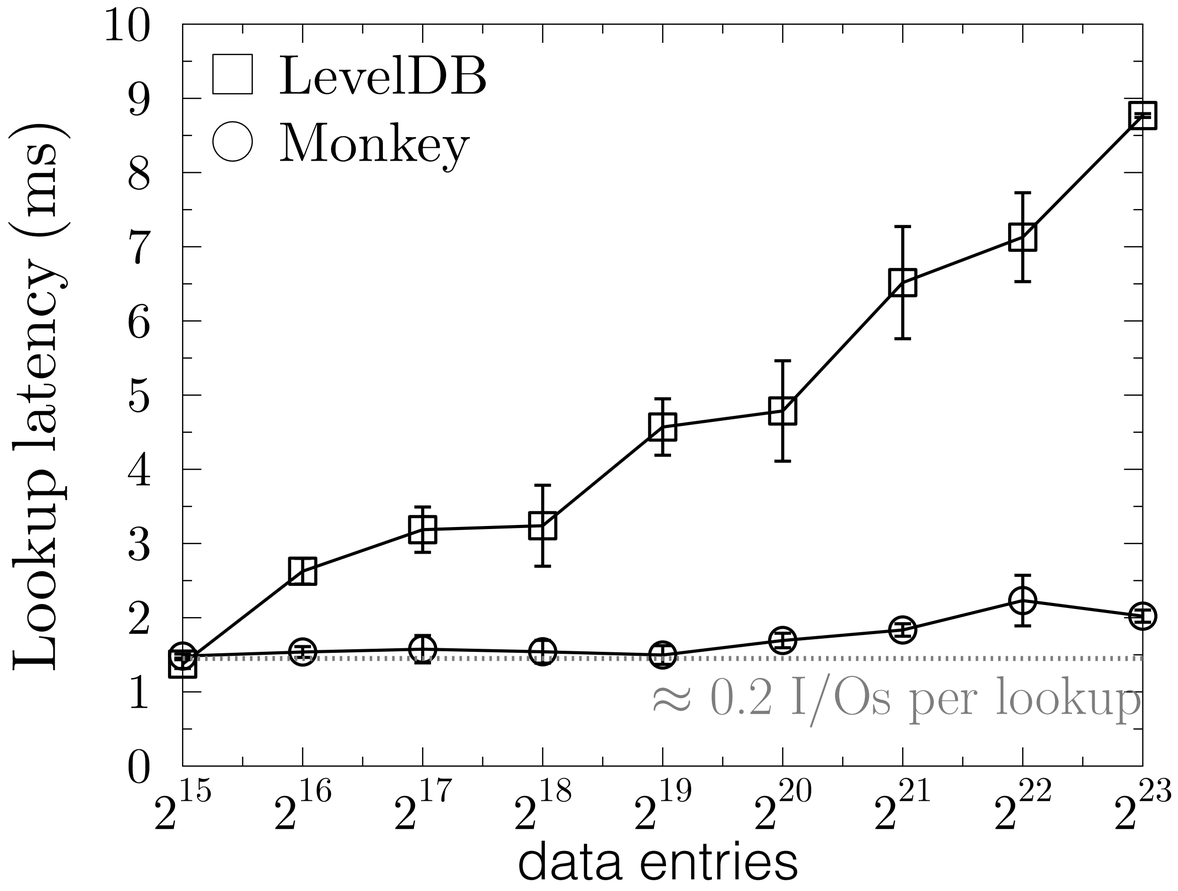
DASlab @ Harvard SEAS

BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: **O**ptimal **N**avigable **Key**-Value Store @SIGMOD2017

the same memory budget is more impactful at smaller levels





uniform, zero result, point queries, entry size=1KB

DASlab @ Harvard SEAS

MERGE POLICY: SHOULD BE TUNED

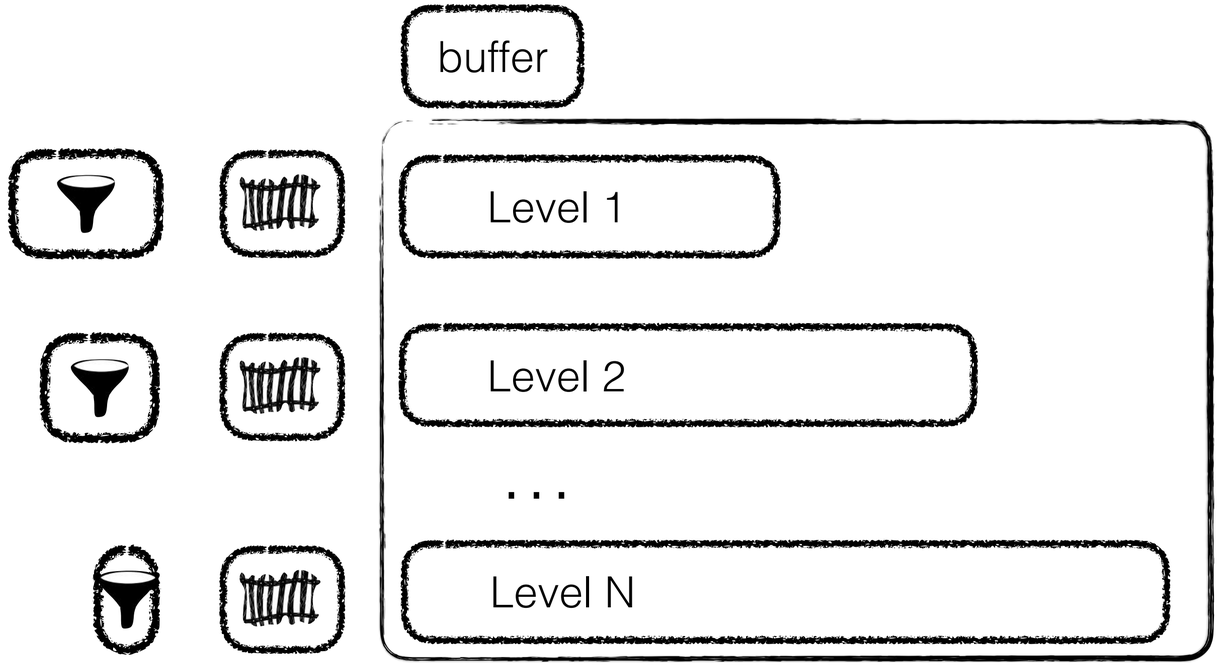
@SIGMOD2018

Dostoevsky: **S**pace-**T**ime **O**ptimized **Ev**olvable **S**calable **K**e**y**-Value Store

MERGE POLICY: SHOULD BE TUNED

@SIGMOD2018

Dostoevsky: **S**pace-**T**ime **O**ptimized **Ev**olvable **S**calable **K**e**y**-Value Store

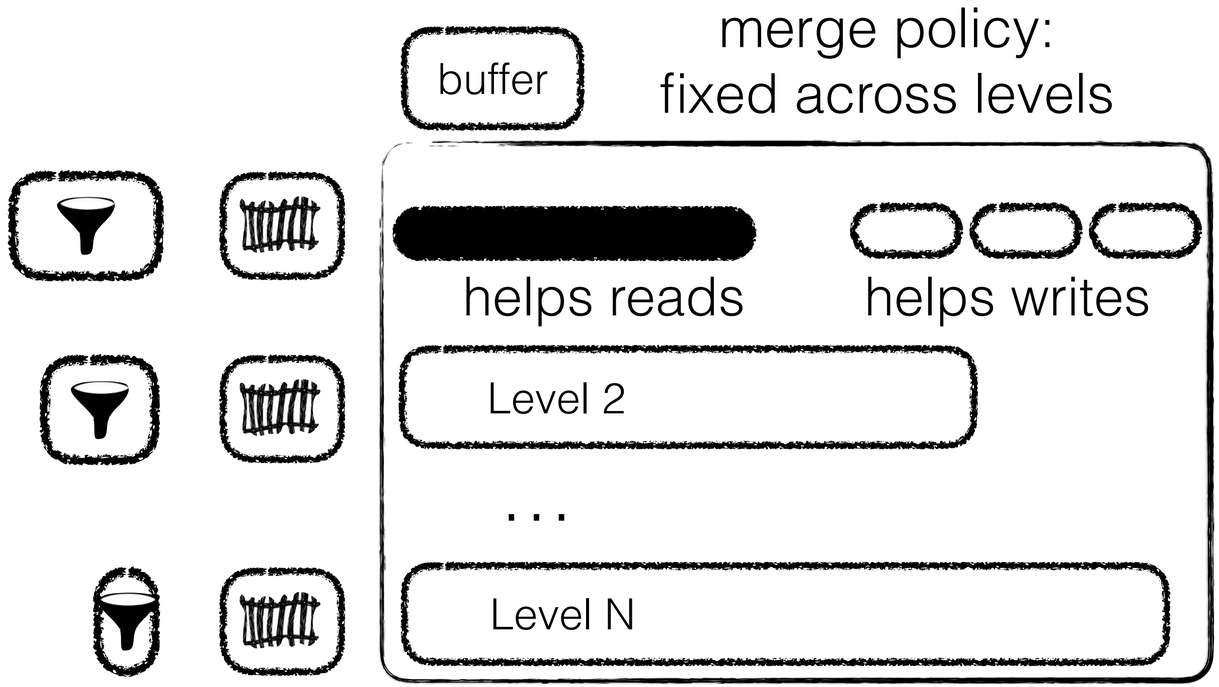


DASlab
@ Harvard SEAS

MERGE POLICY: SHOULD BE TUNED

@SIGMOD2018

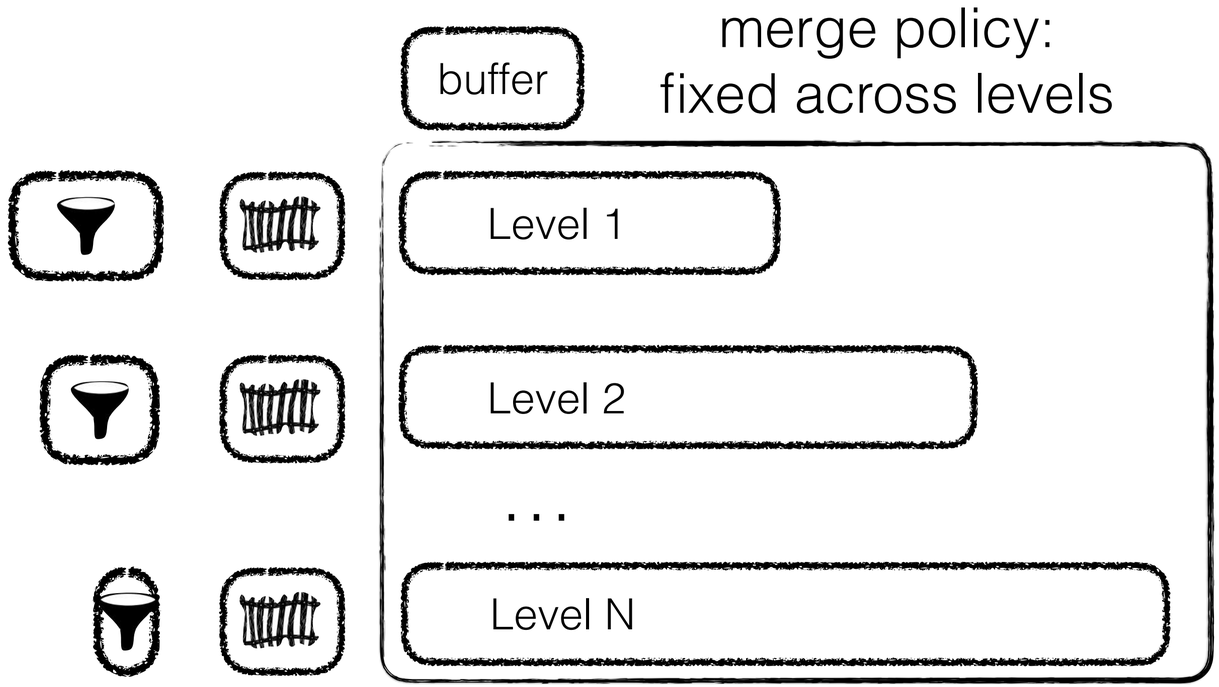
Dostoevsky: **S**pace-**T**ime **O**ptimized **Ev**olvable **S**calable **K**e**y**-Value Store



MERGE POLICY: SHOULD BE TUNED

@SIGMOD2018

Dostoevsky: **S**pace-**T**ime **O**ptimized **Ev**olvable **S**calable **Ke**y-Value Store



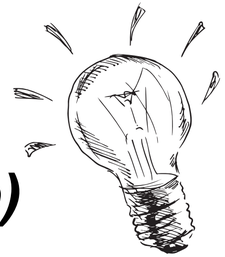
DASlab
@ Harvard SEAS

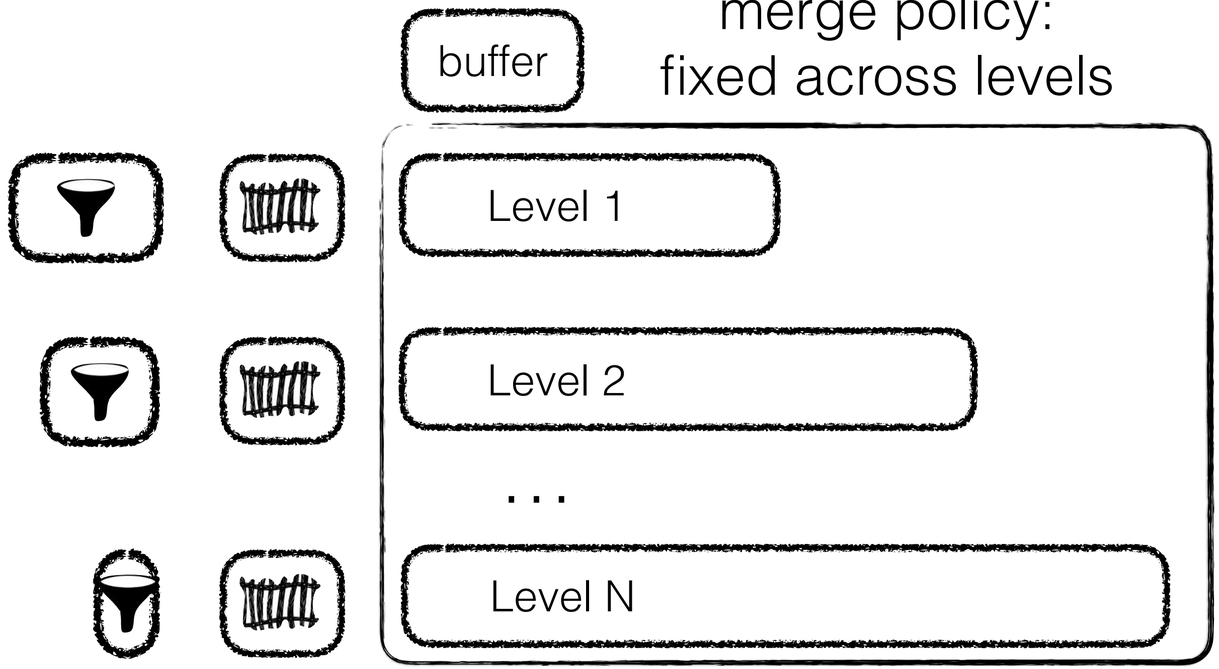
MERGE POLICY: SHOULD BE TUNED

@SIGMOD2018

Dostoevsky: **S**pace-**T**ime **O**ptimized **Ev**olvable **S**calable **K**e**y**-Value Store

merging small levels does not help that much (point,range,space)



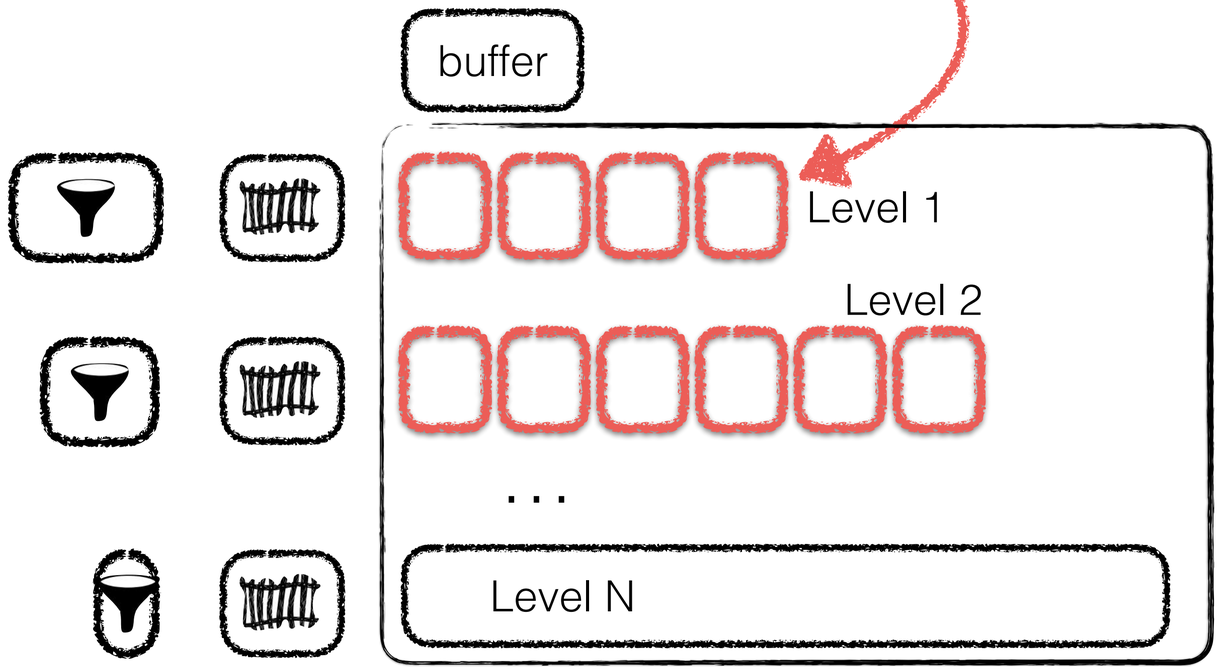


MERGE POLICY: SHOULD BE TUNED

@SIGMOD2018

Dostoevsky: **S**pace-**T**ime **O**ptimized **Ev**olvable **S**calable **K**e**y**-Value Store

merging small levels does not help that much (point,range,space)

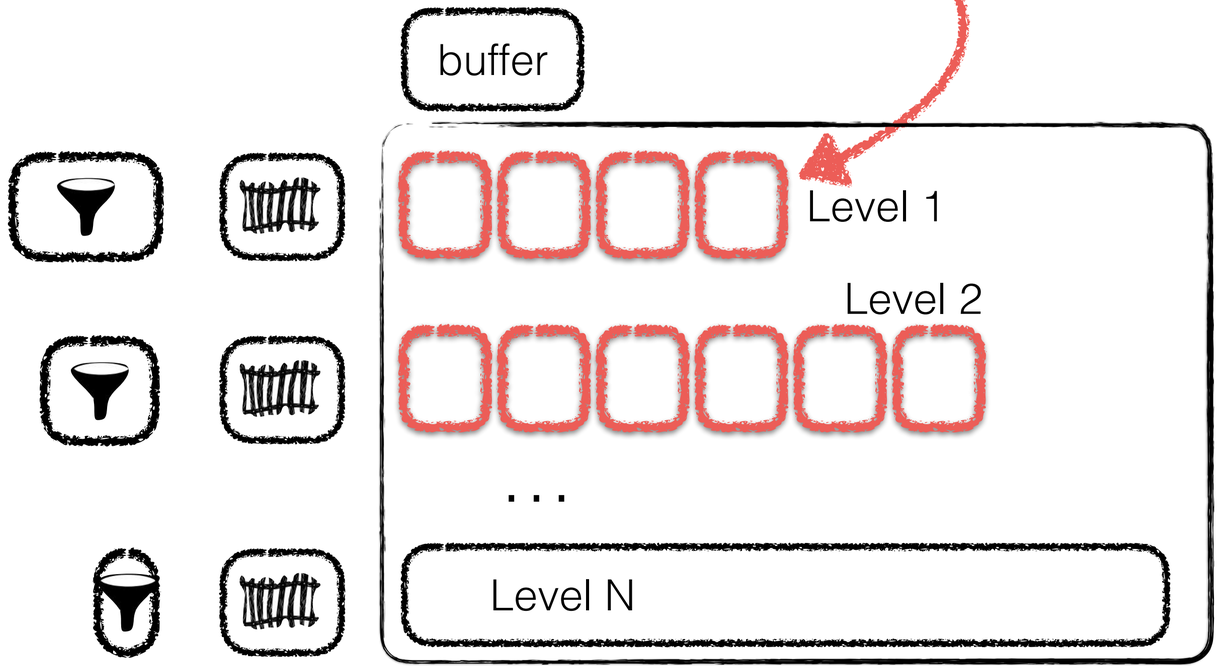


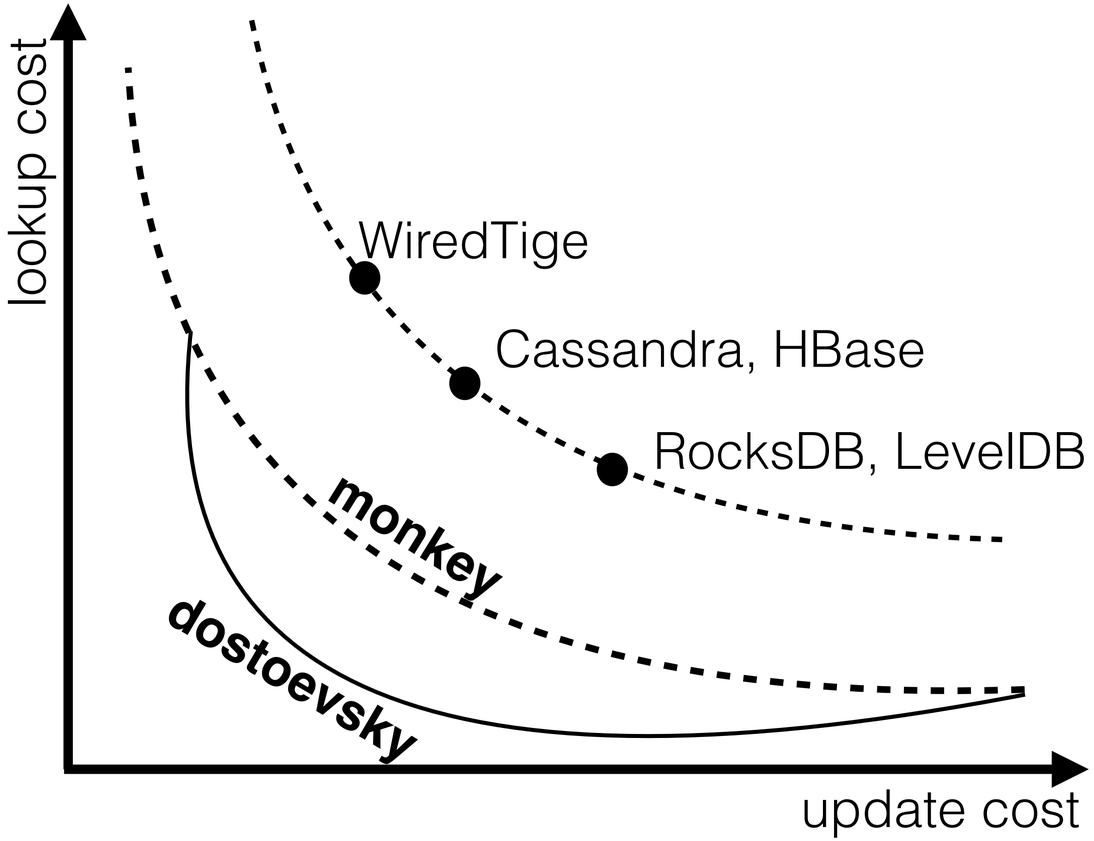
MERGE POLICY: SHOULD BE TUNED

@SIGMOD2018

Dostoevsky: **S**pace-**T**ime **O**ptimized **Ev**olvable **S**calable **K**ey-Value Store

merging small levels does not help that much (point,range,space)





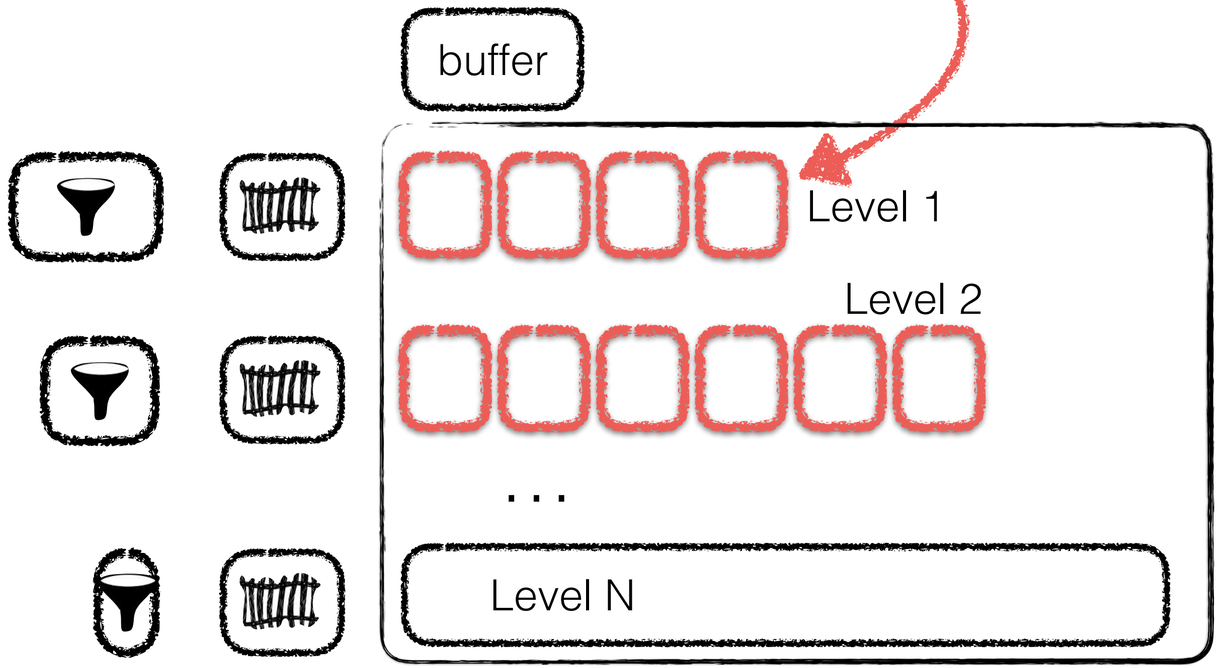
DASlab @ Harvard SEAS

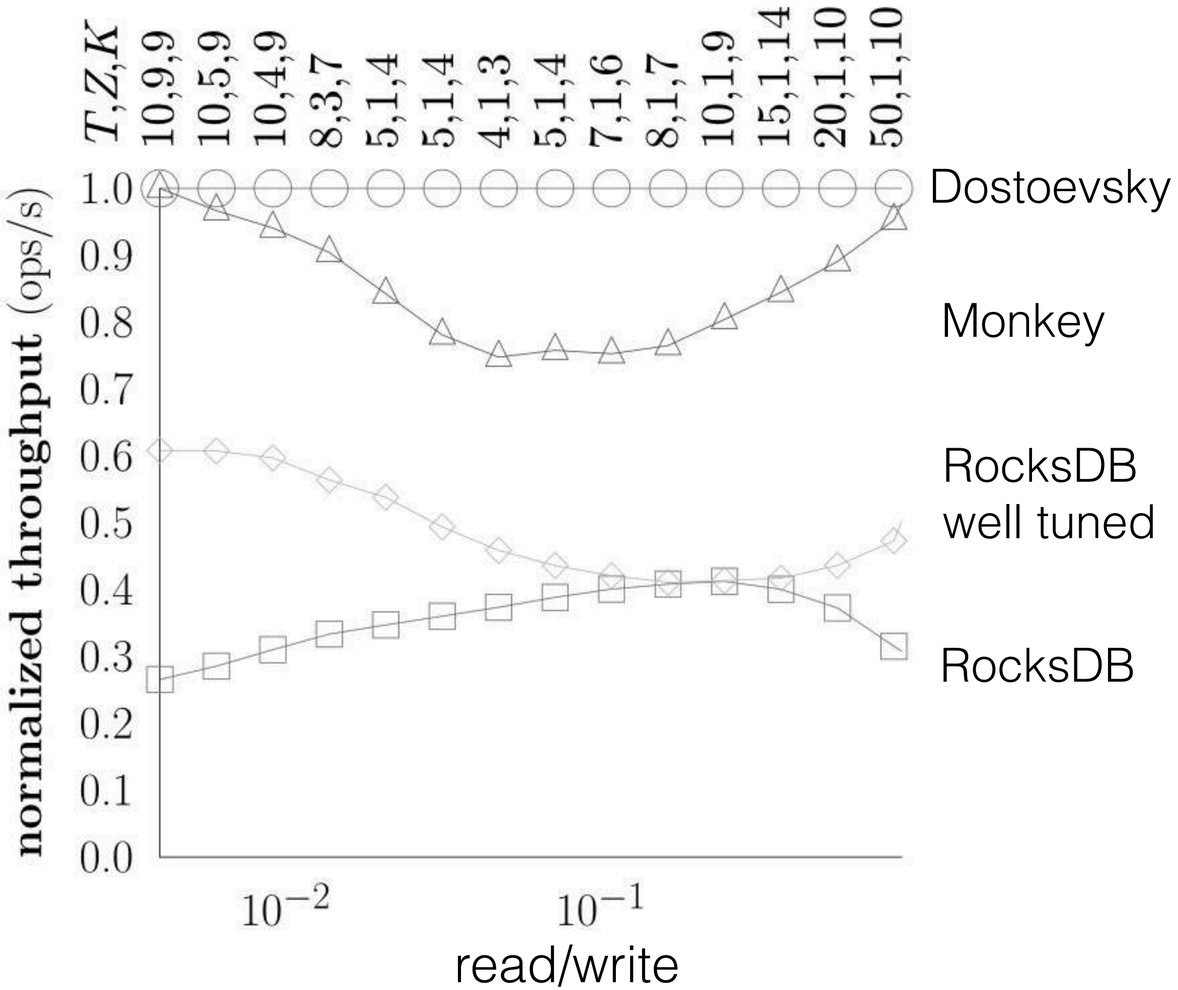
MERGE POLICY: SHOULD BE TUNED

@SIGMOD2018

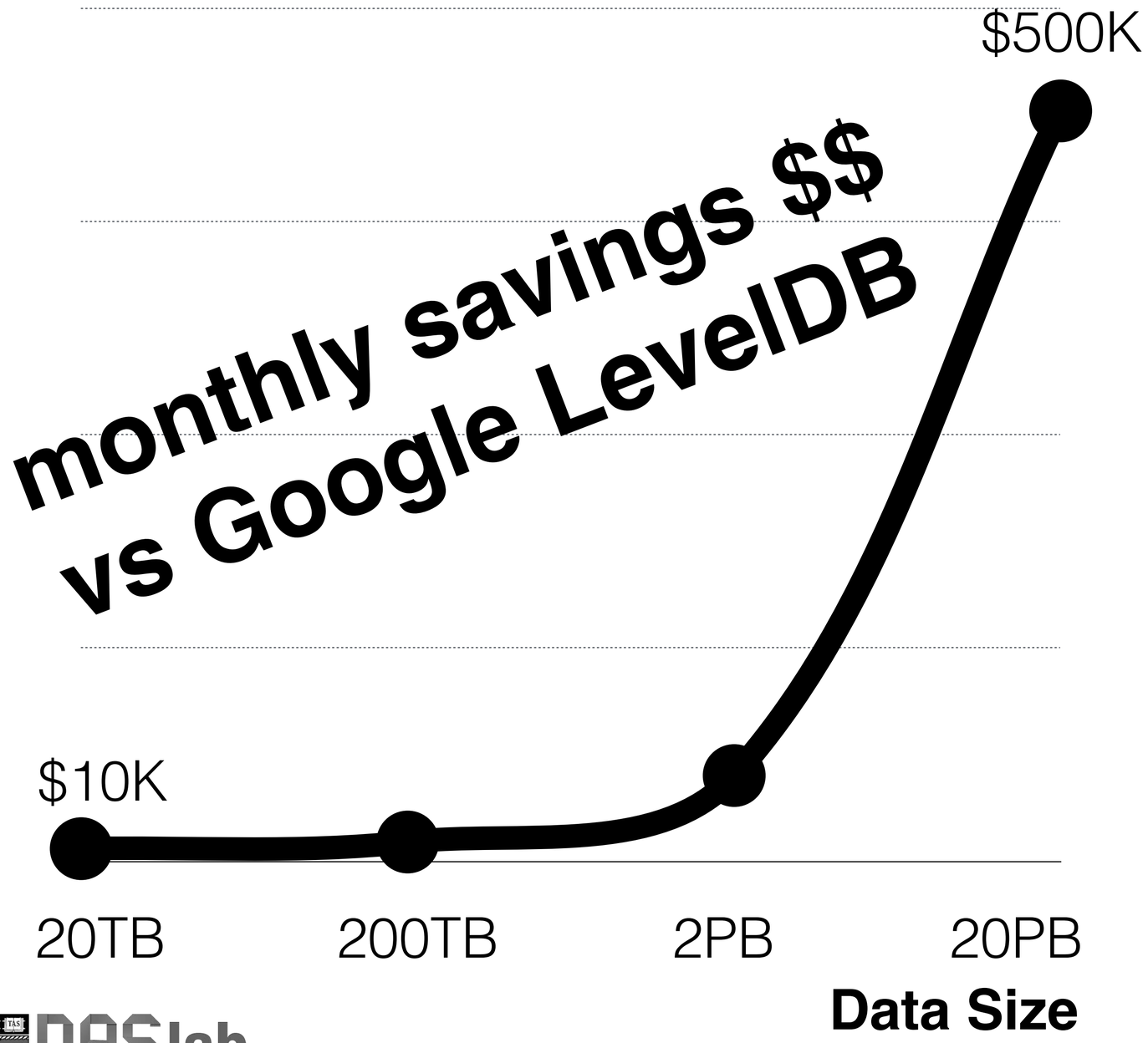
Dostoevsky: **S**pace-**T**ime **O**ptimized **Ev**olvable **S**calable **K**ey-Value Store

merging small levels does not help that much (point,range,space)



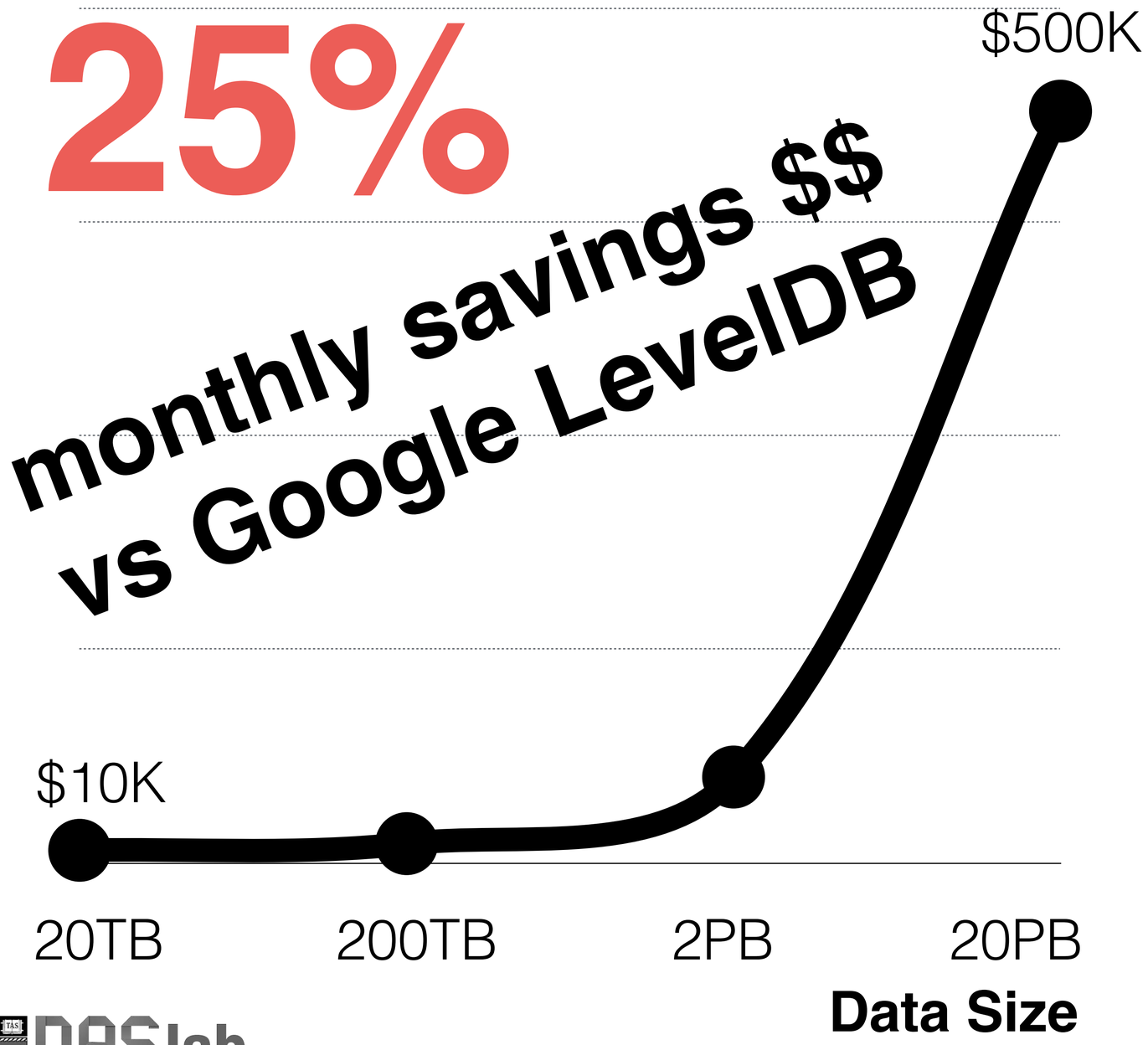


DASlab @ Harvard SEAS



DASlab @ Harvard SEAS

Amazon Cloud (North America)

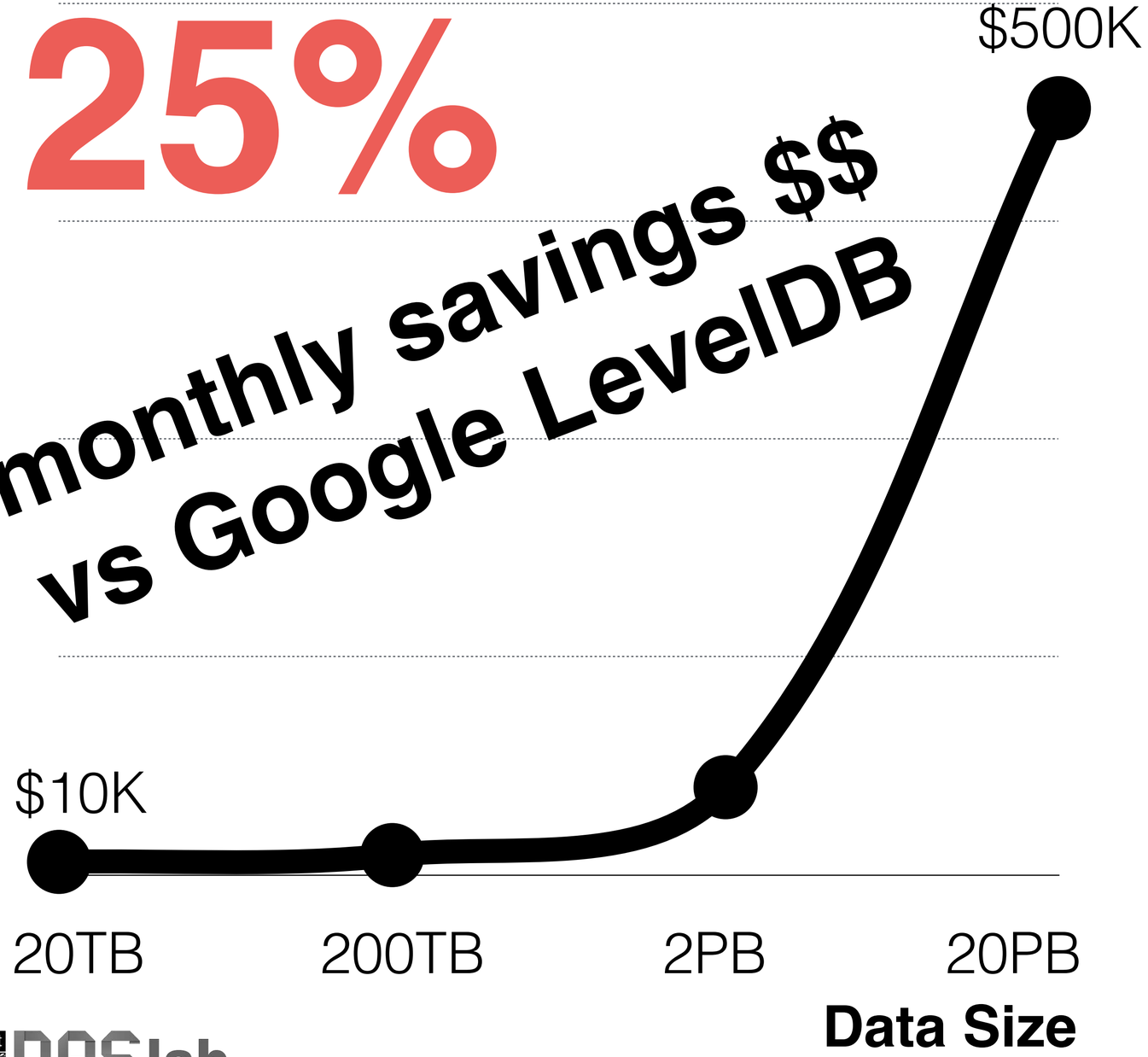


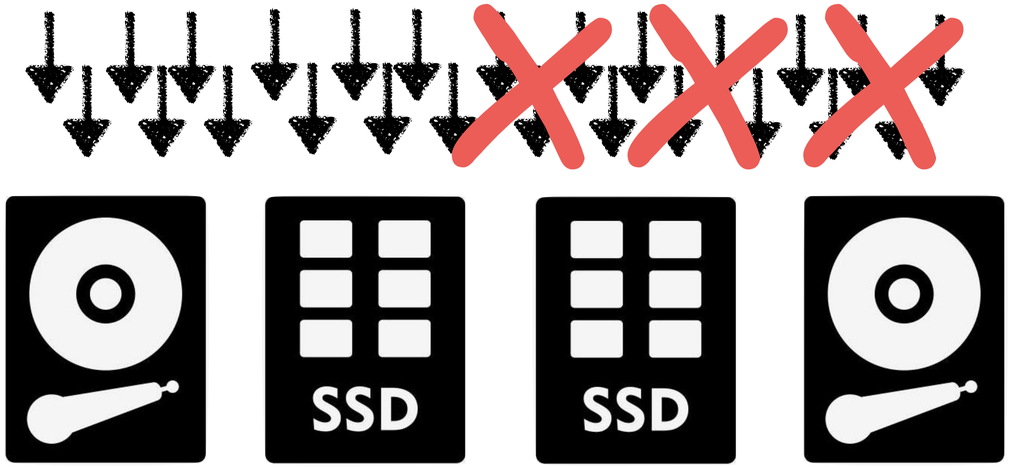
DASlab @ Harvard SEAS

Amazon Cloud (North America)

25%

monthly savings $$ vs Google LevelDB



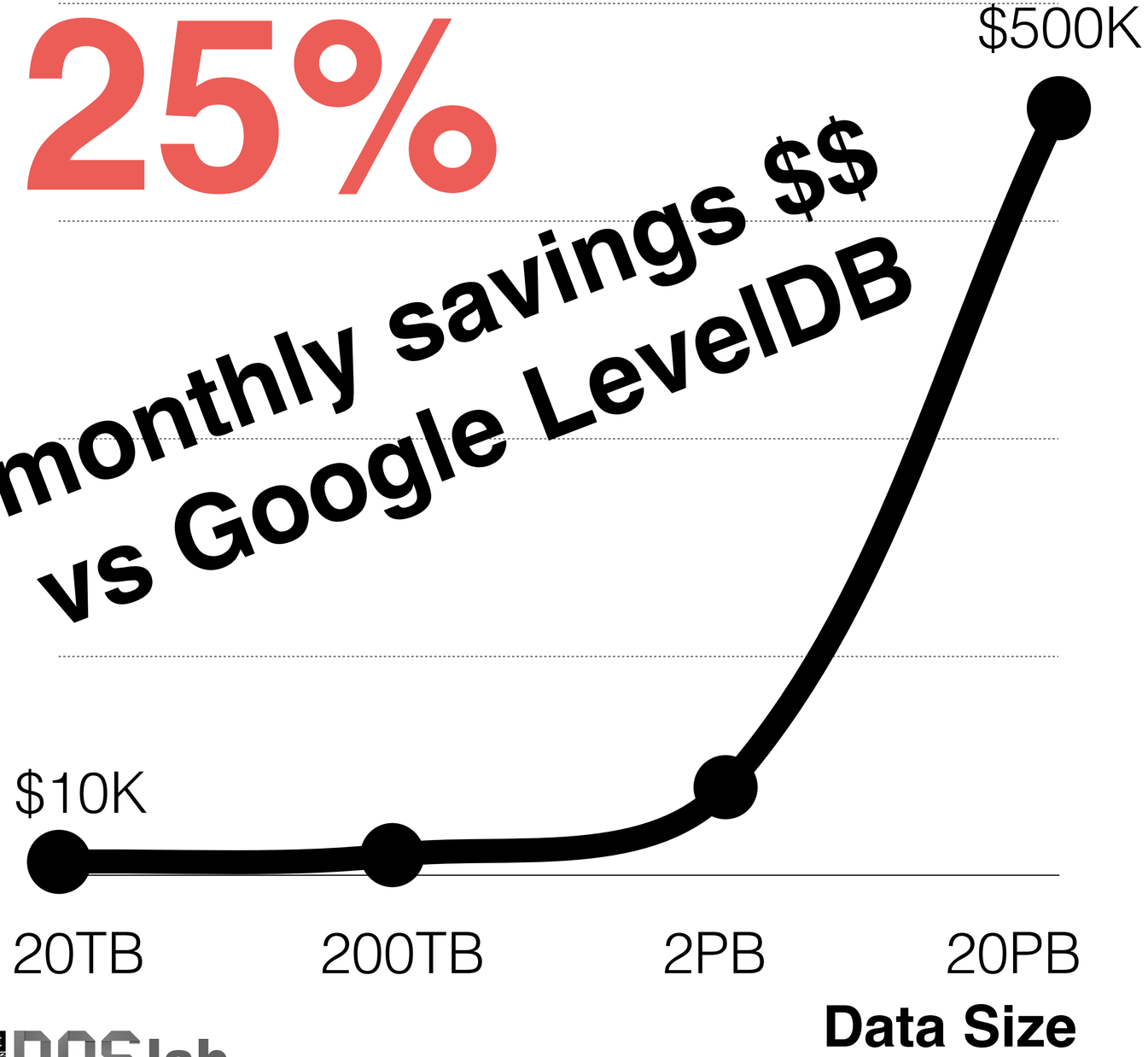


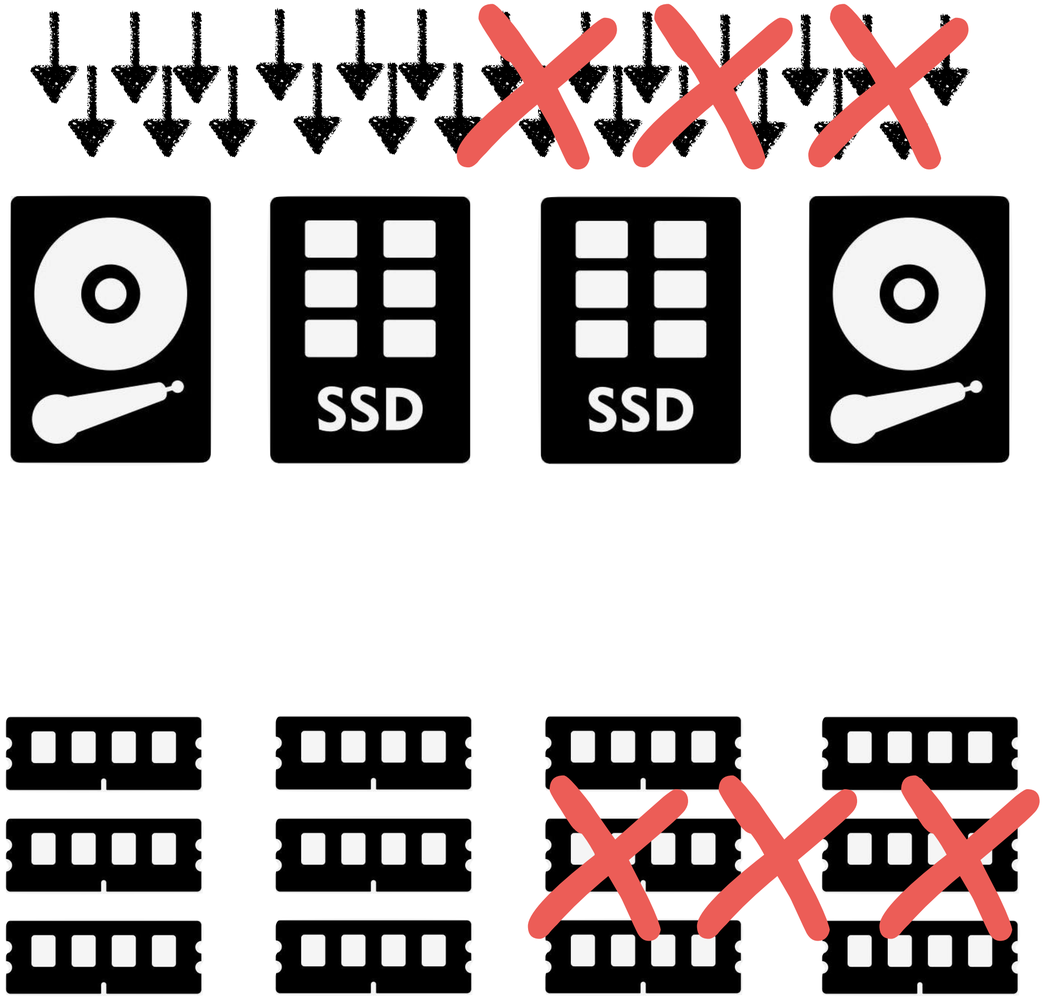
DASlab @ Harvard SEAS

Amazon Cloud (North America)

25%

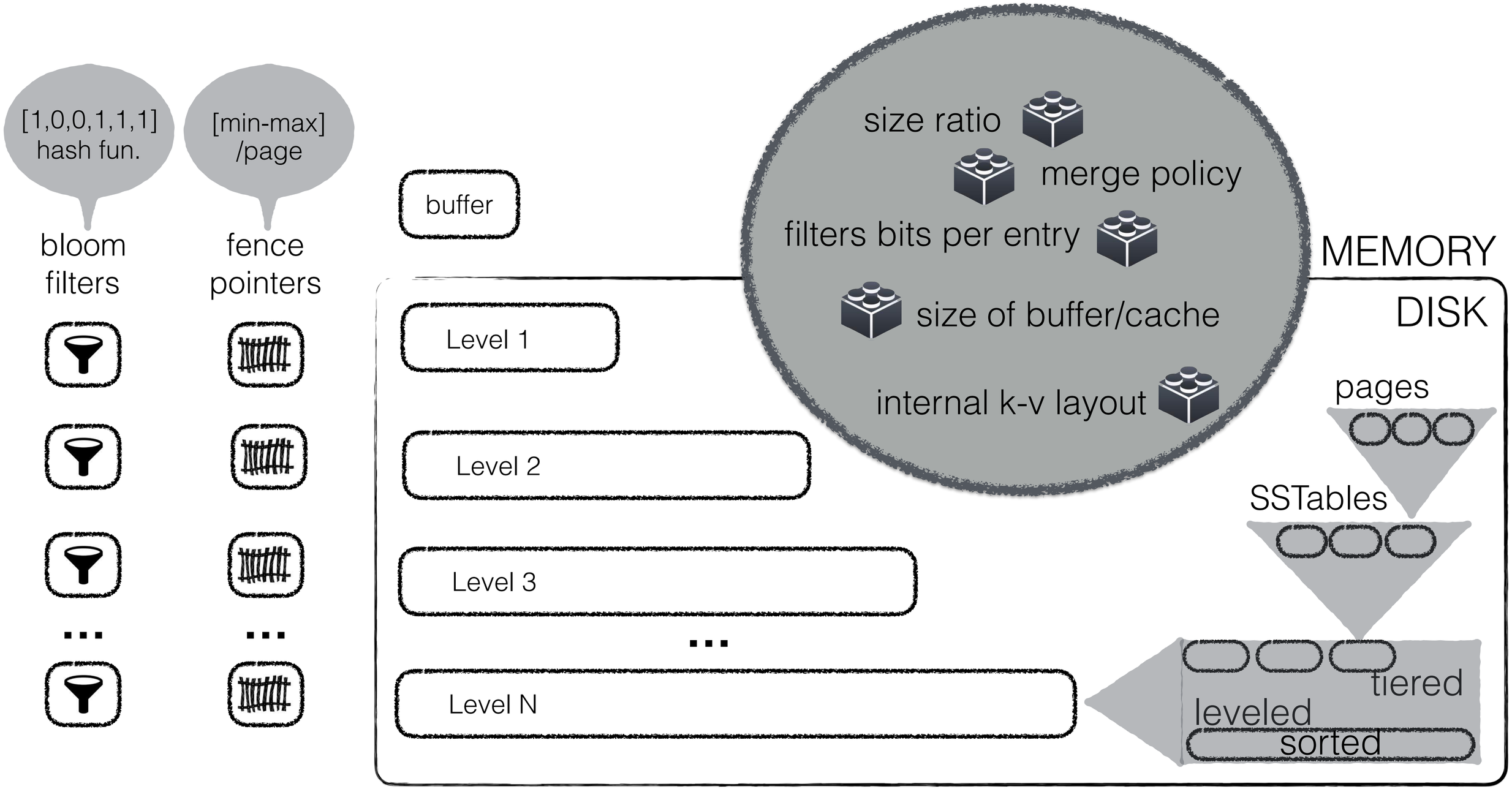
monthly savings $$ vs Google LevelDB



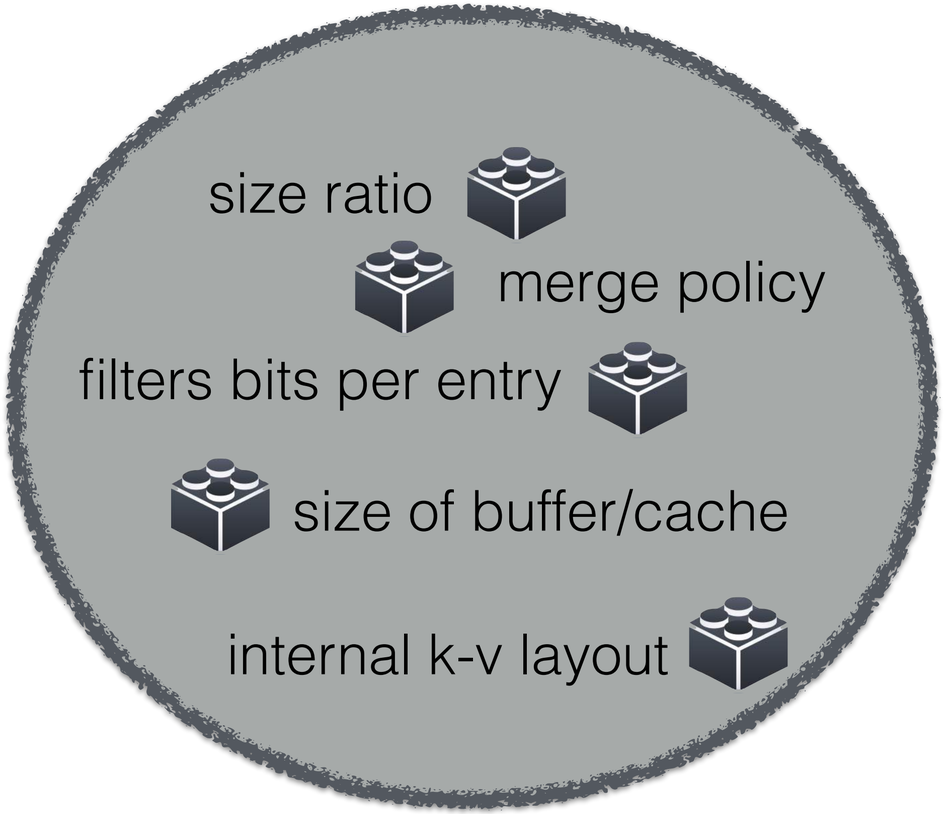


Amazon Cloud (North America)

DASlab @ Harvard SEAS



LSM-trees


size ratio
merge policy
filters bits per entry
size of buffer/cache
internal k-v layout

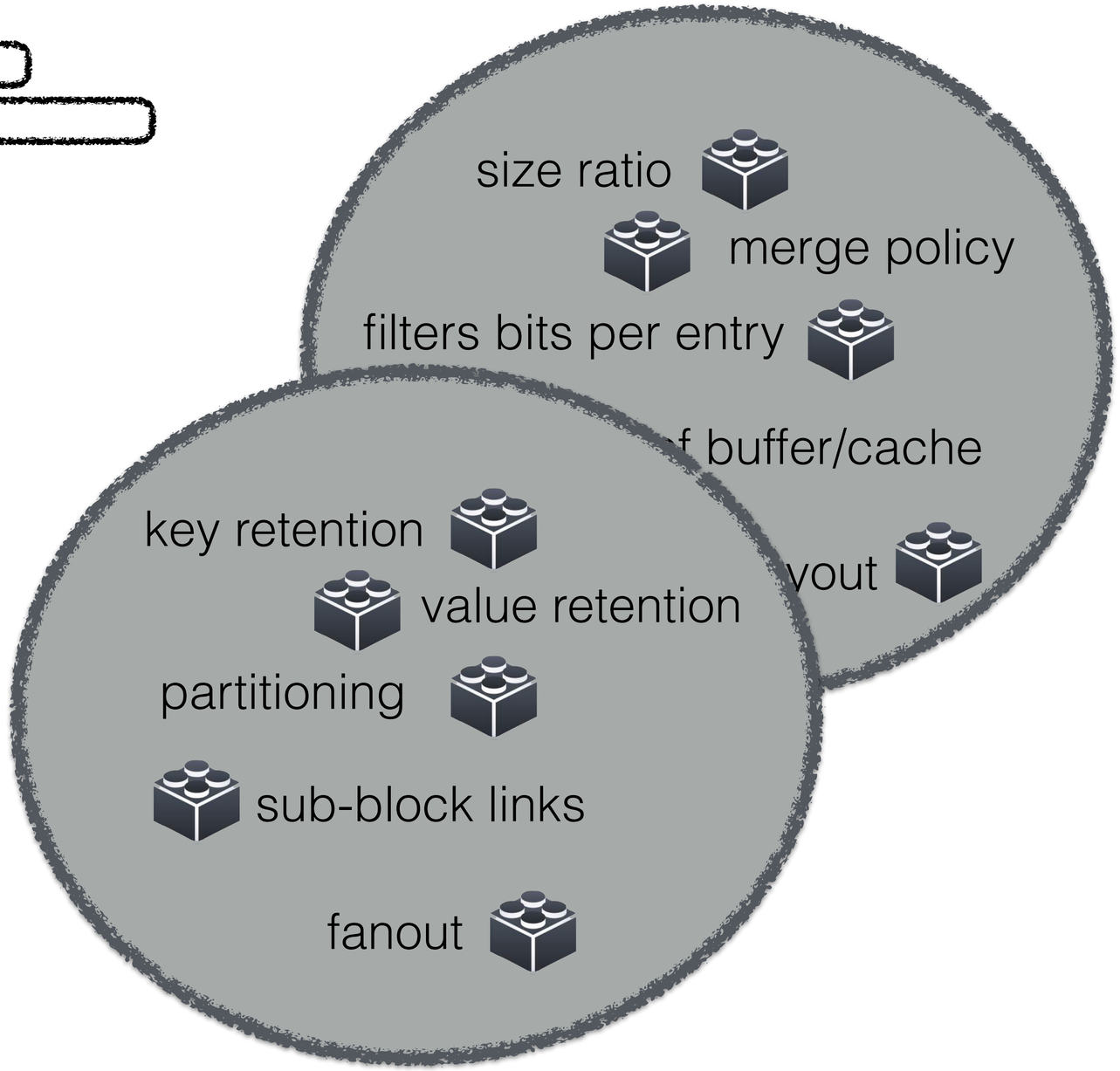
LSM-trees

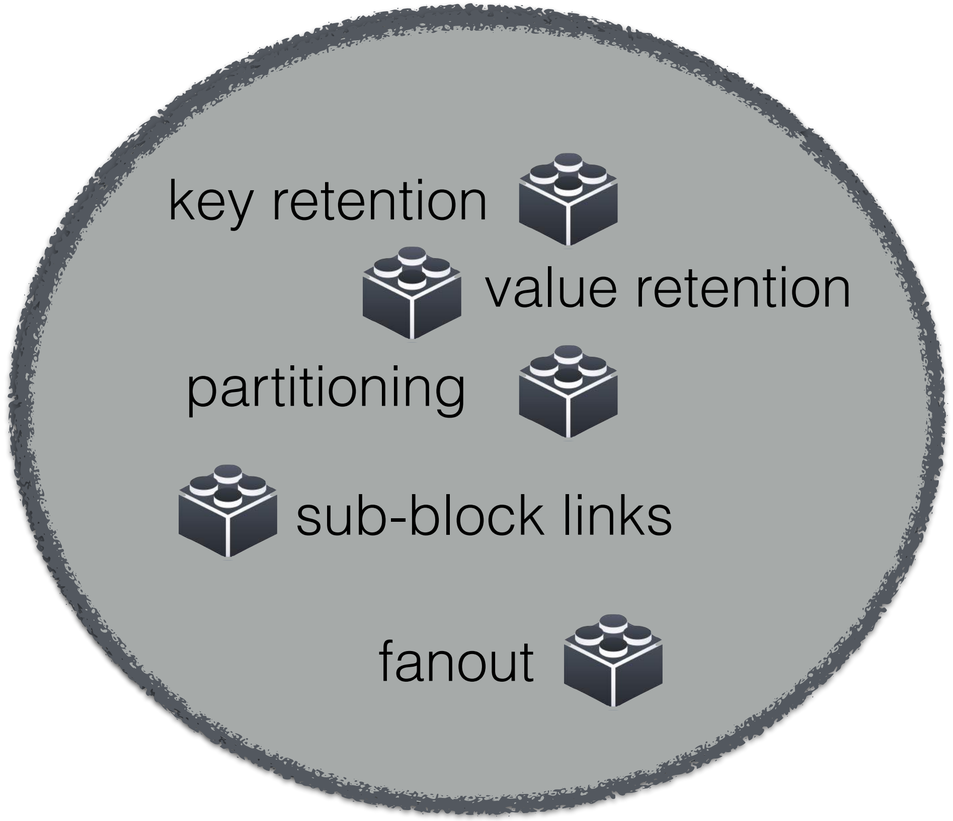
B-trees

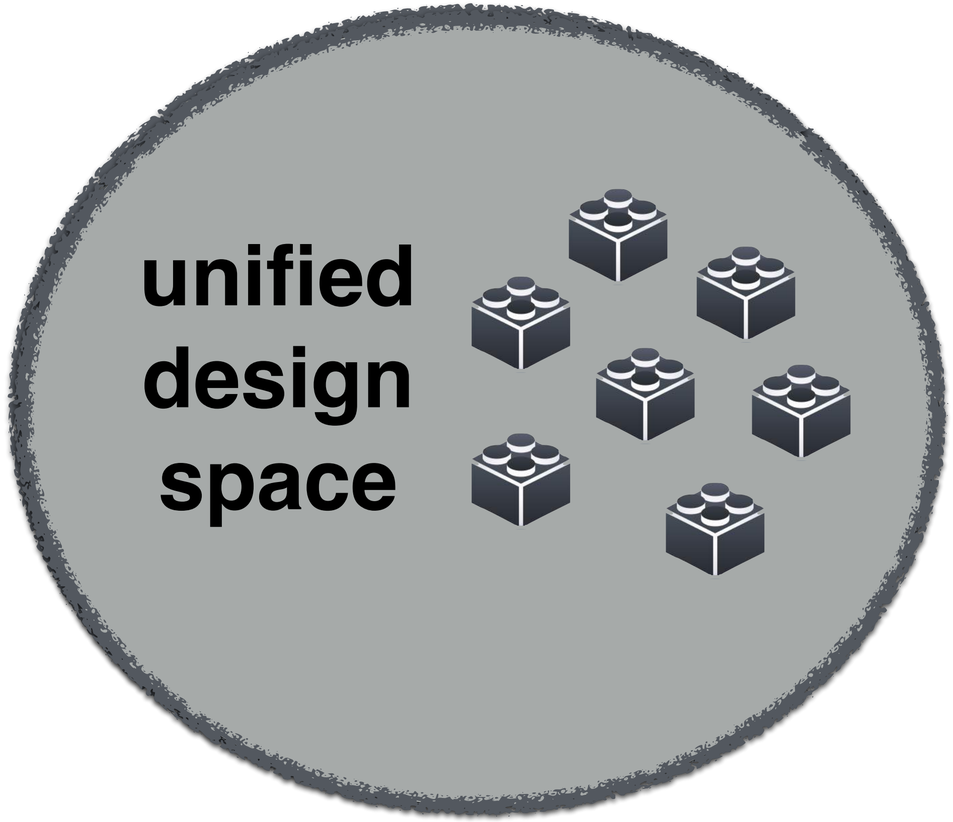
Logs

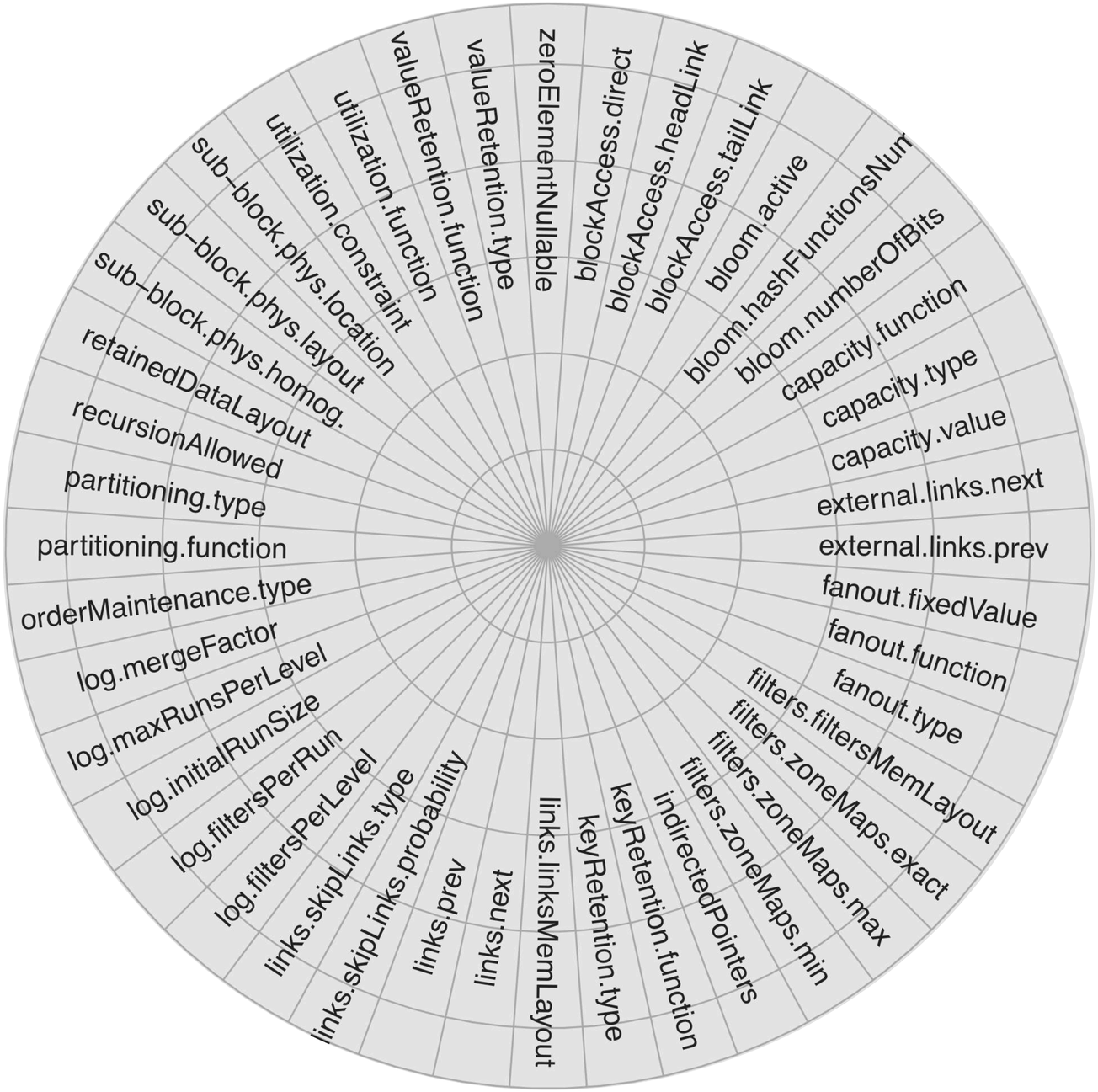
Arrays

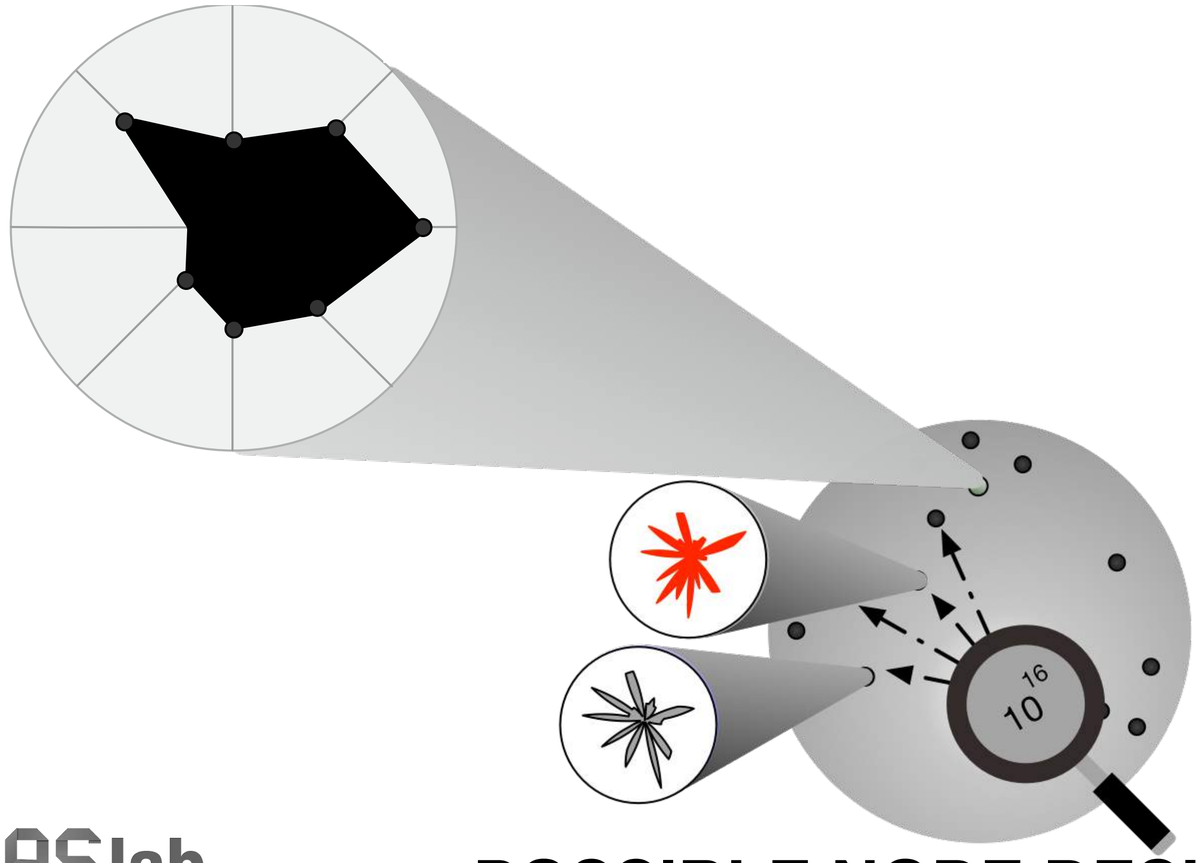
Bitmaps



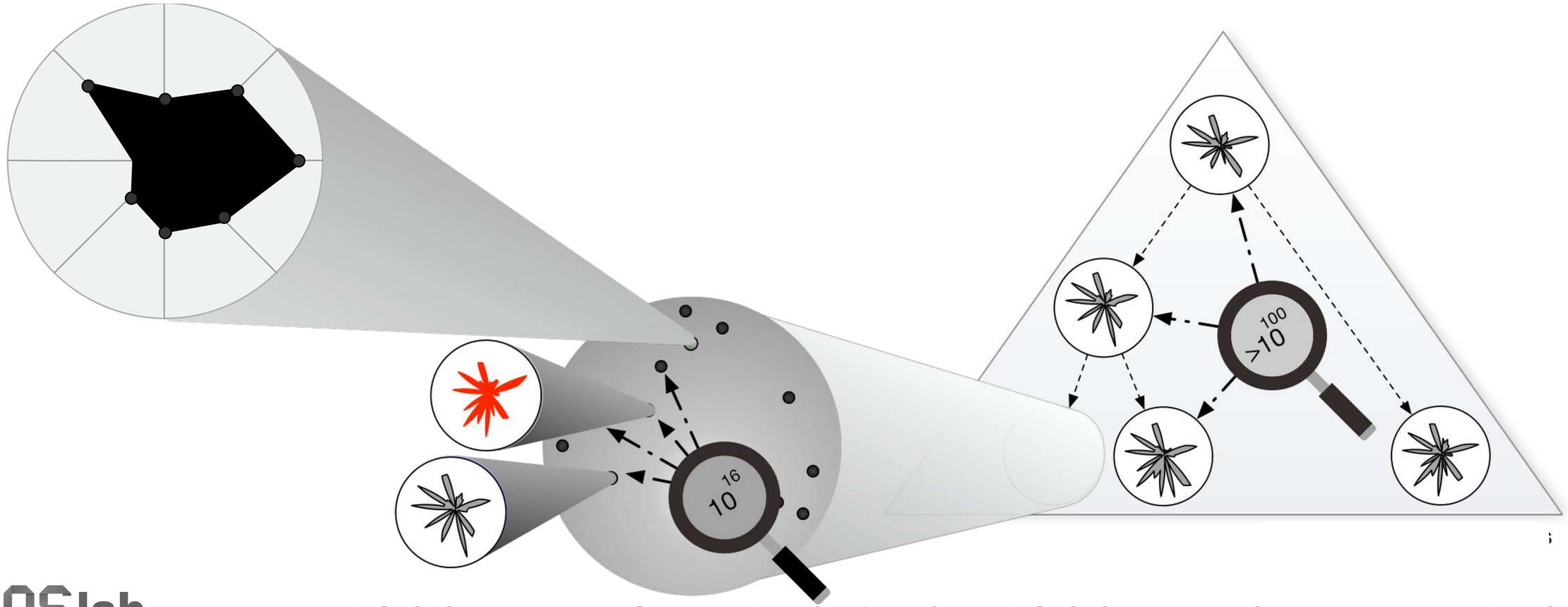

key retention
value retention
partitioning
sub-block links
fanout


unified
design
space

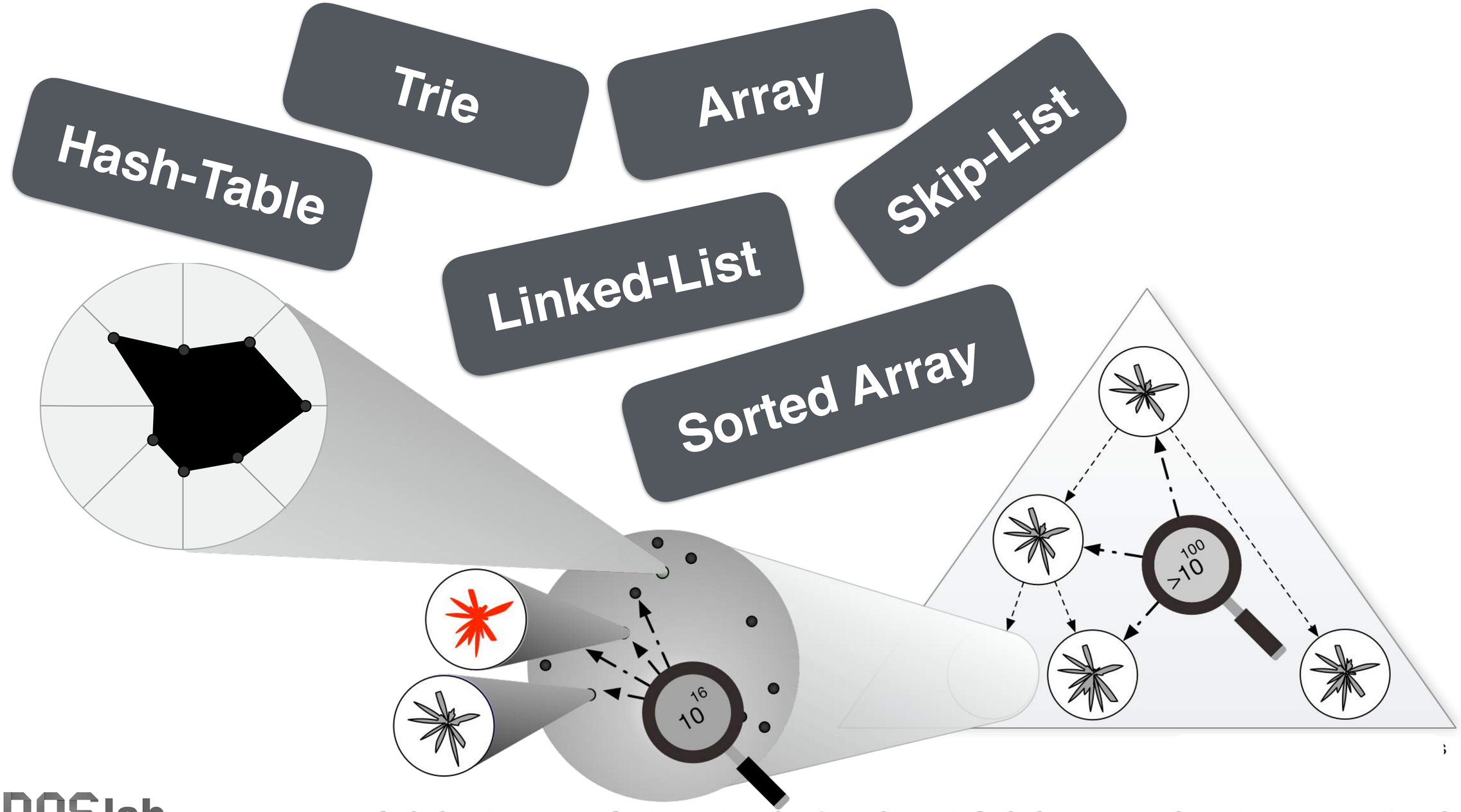

zeroElementNullable
blockAccess.direct
blockAccess.headLink
blockAccess.tailLink
bloom.active
bloom.hashFunctionsNum
bloom.numberOfBits
capacity.function
capacity.type
capacity.value
external.links.next
external.links.prev
fanout.fixedValue
fanout.function
fanout.type
filters.filtersMemLayout
filters.zoneMaps.exact
filters.zoneMaps.max
filters.zoneMaps.min
indirectedPointers
keyRetention.function
keyRetention.type
links.linksMemLayout
links.next
links.prev
links.skipLinks.probability
links.skipLinks.type
log.filtersPerLevel
log.filtersPerRun
log.initialRunSize
log.maxRunsPerLevel
log.mergeFactor
orderMaintenance.type
partitioning.function
partitioning.type
recursionAllowed
retainedDataLayout
sub–block.phys.homog.
sub–block.phys.layout
sub–block.phys.location
utilization.constraint
utilization.function
valueRetention.function
valueRetention.type



POSSIBLE NODE DESIGNS



POSSIBLE NODE DESIGNS **POSSIBLE STRUCTURES**



POSSIBLE NODE DESIGNS POSSIBLE STRUCTURES

CS 265

Stratos Idreos

BIG DATA SYSTEMS

NoSQL | Neural Networks | SQL | Graph | Data Science