Scaling Log-Structured KV-Stores

featuring

Monkey and Dostoevsky

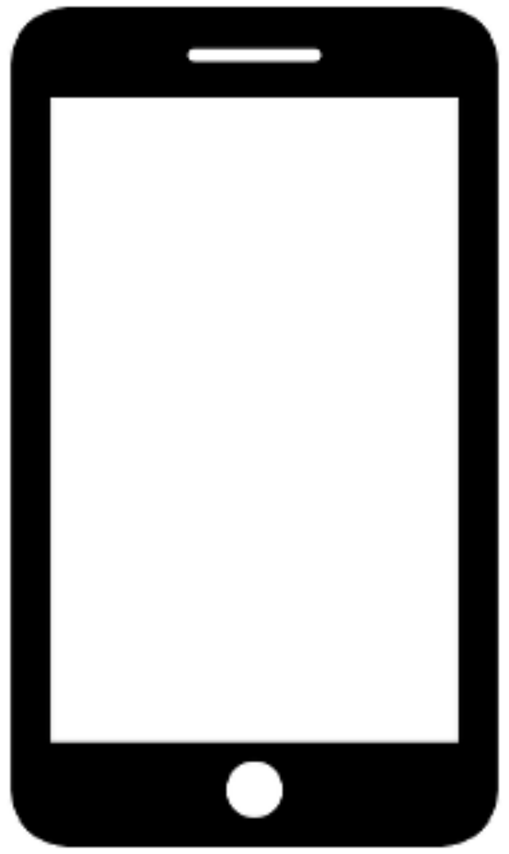
SIGMOD17 / SIGMOD18

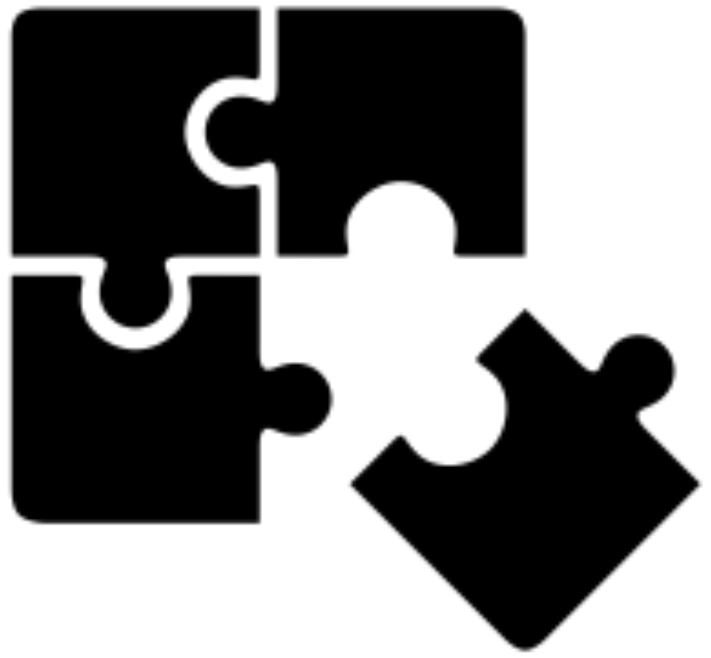
DASlab @ Harvard SEAS

Niv Dayan

Log-Structured KV-Stores

Log-Structured KV-Stores



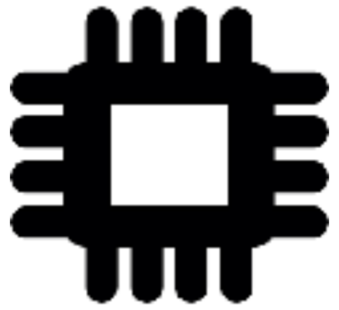


Why Log-Structured KV-Stores?

Why Log-Structured KV-Stores?

fast writes

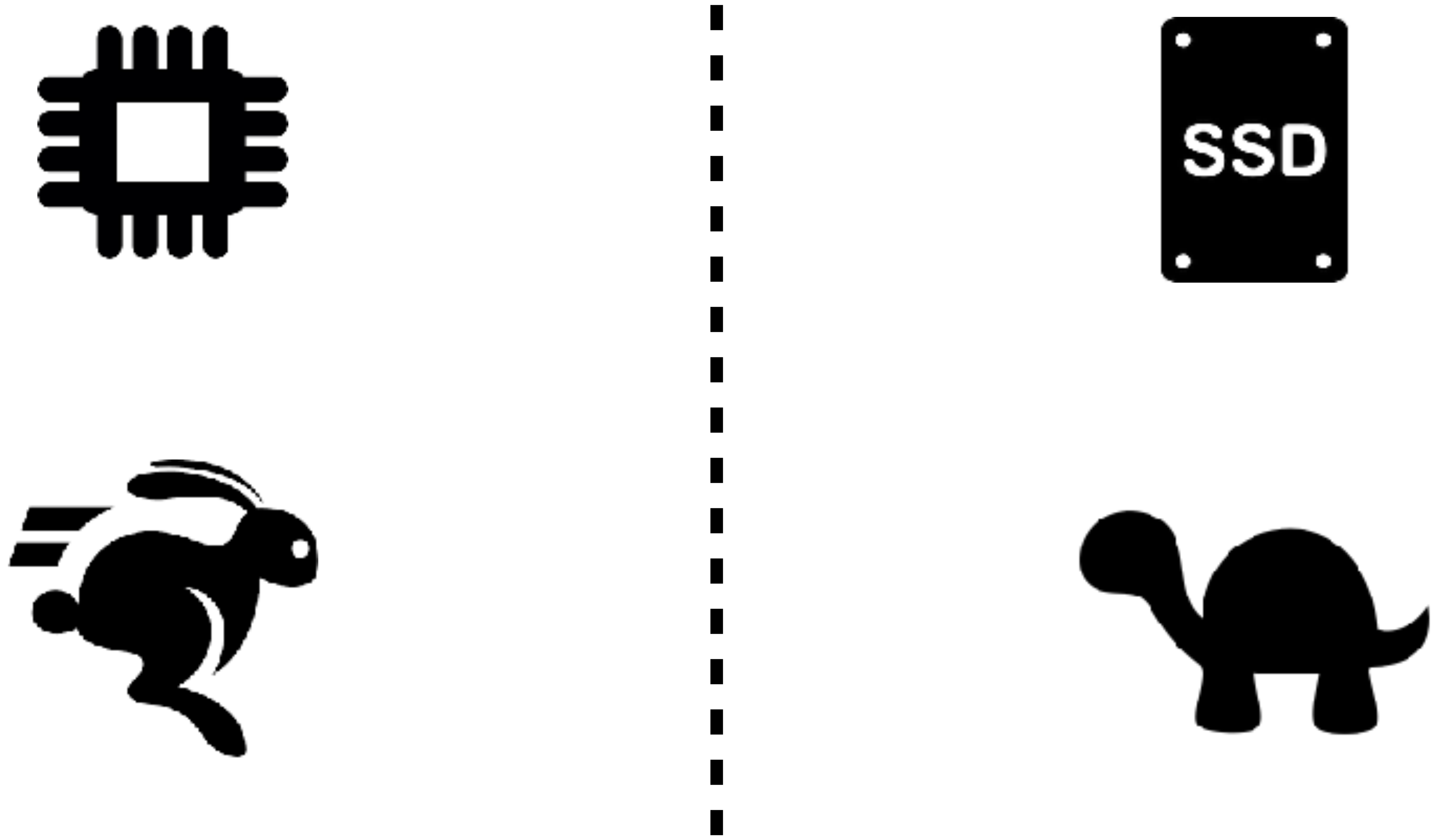
Why Log-Structured KV-Stores?



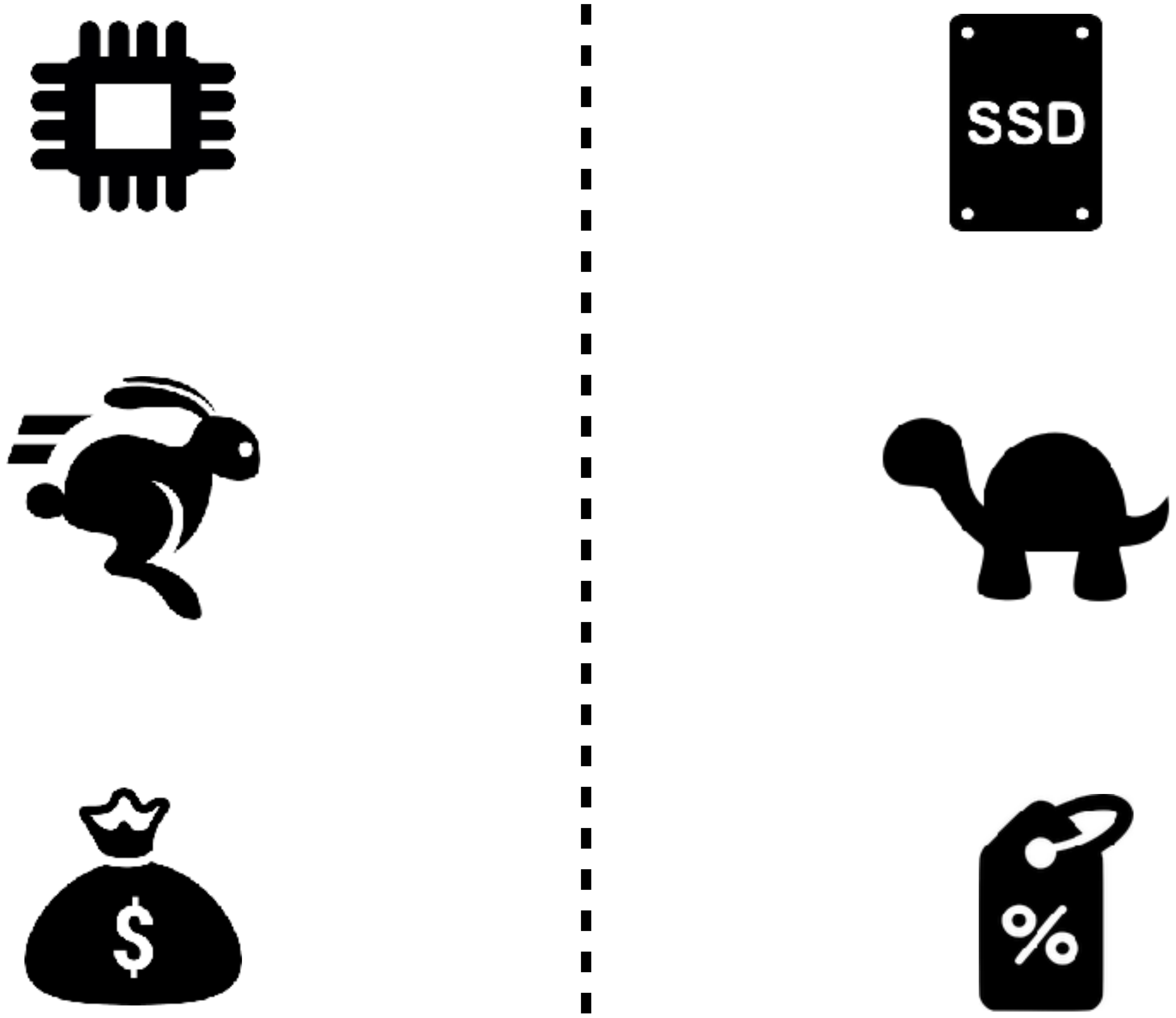
memory

storage

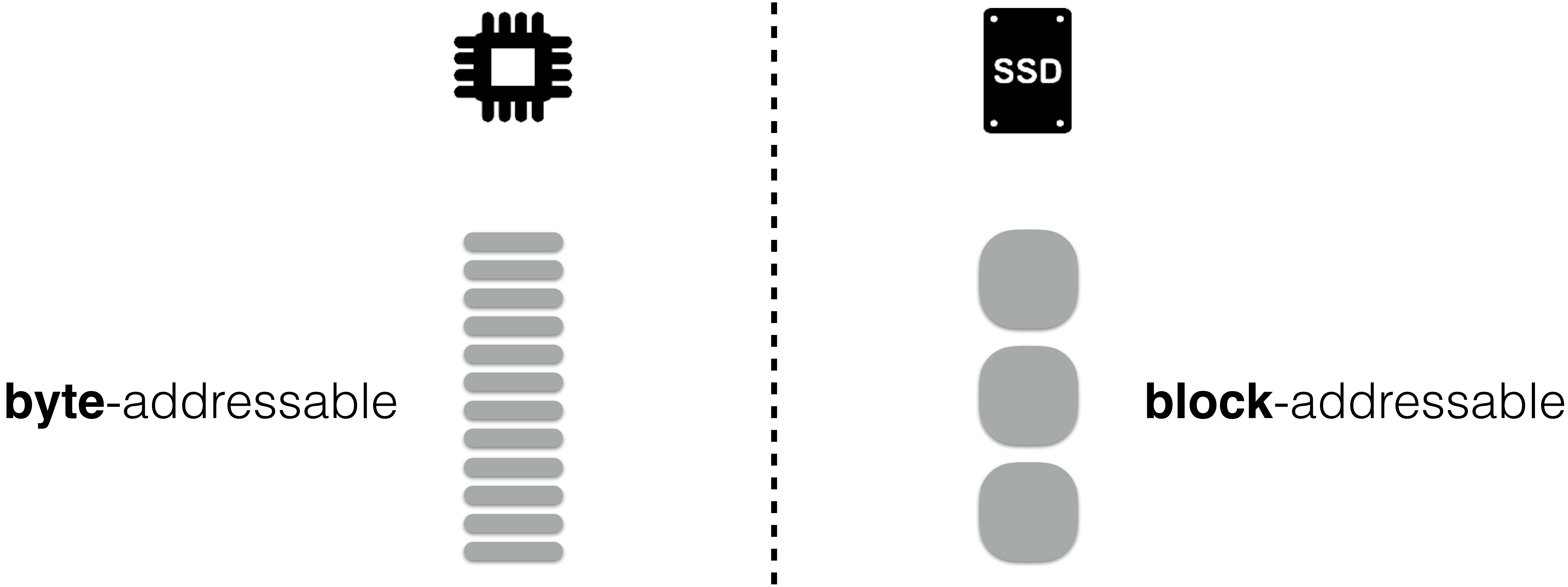
Why Log-Structured KV-Stores?



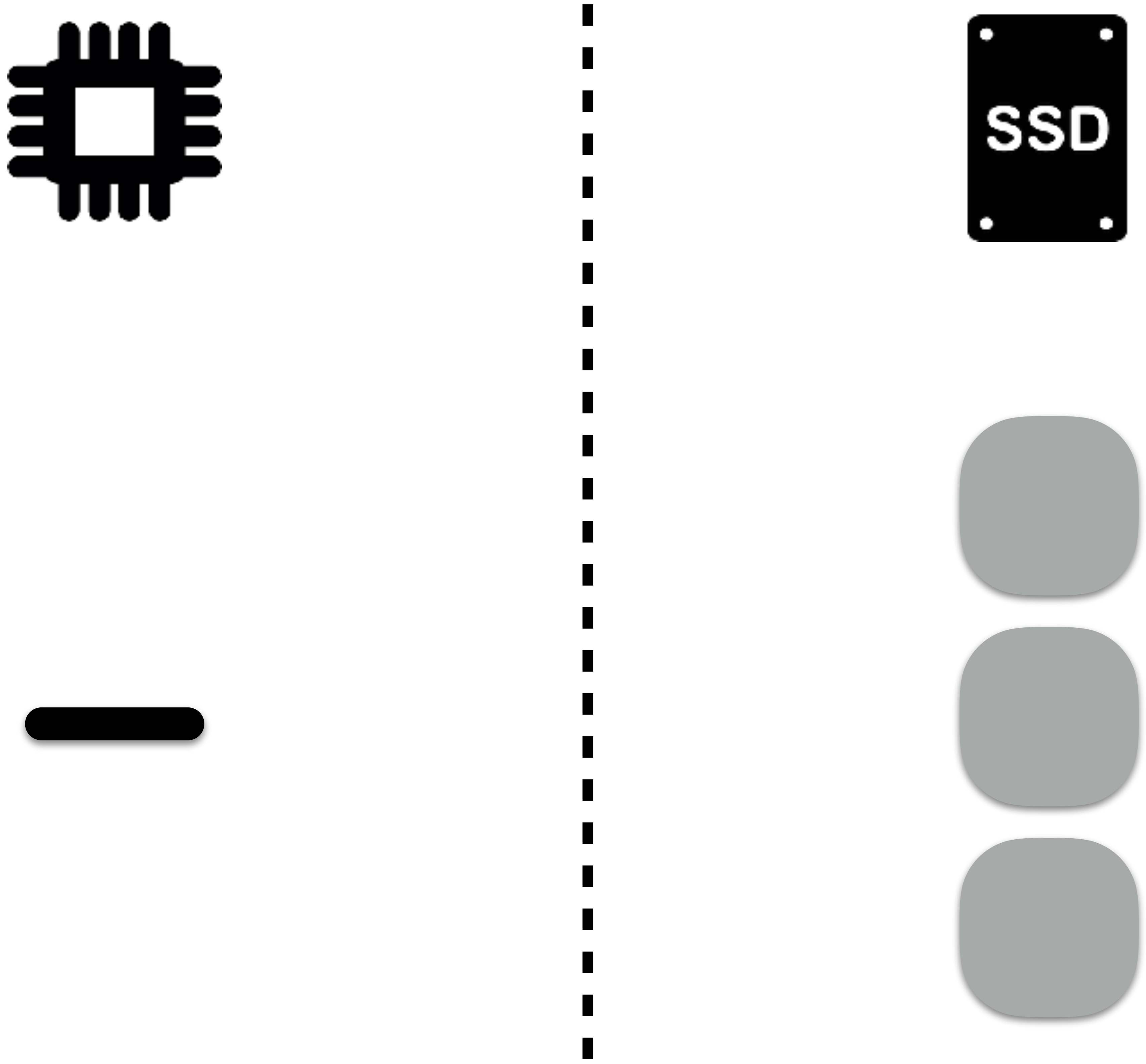
Why Log-Structured KV-Stores?



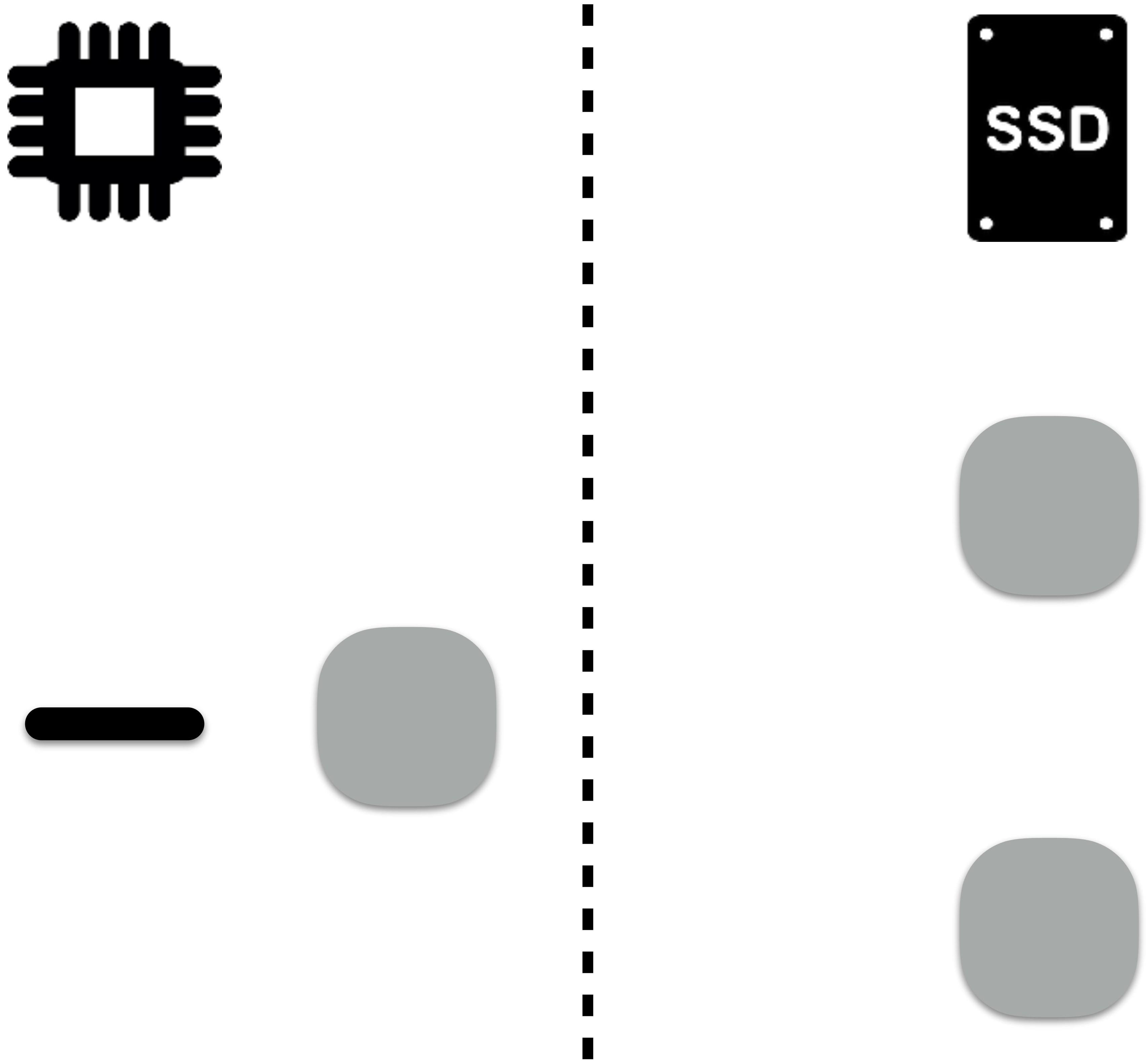
Why Log-Structured KV-Stores?

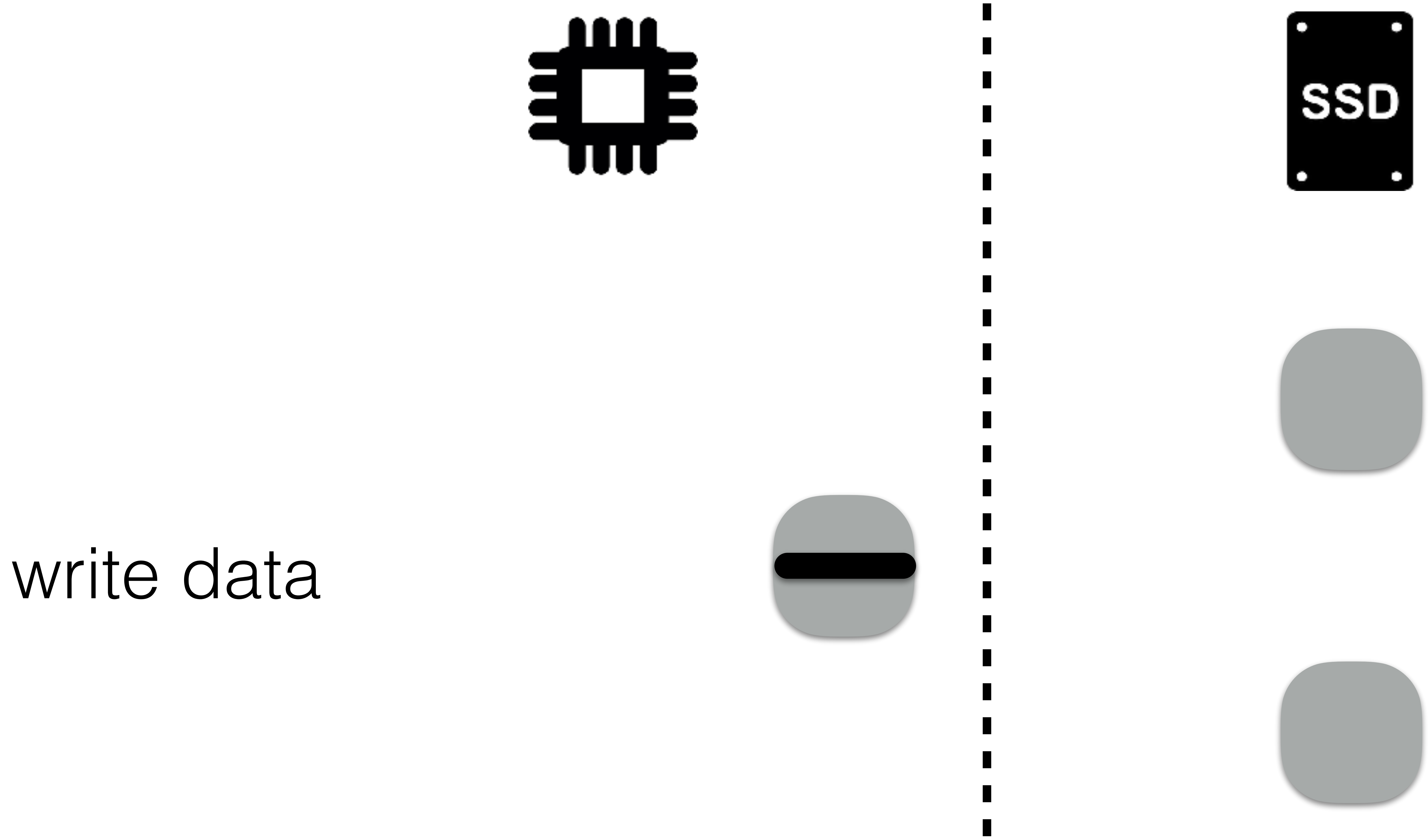


write data

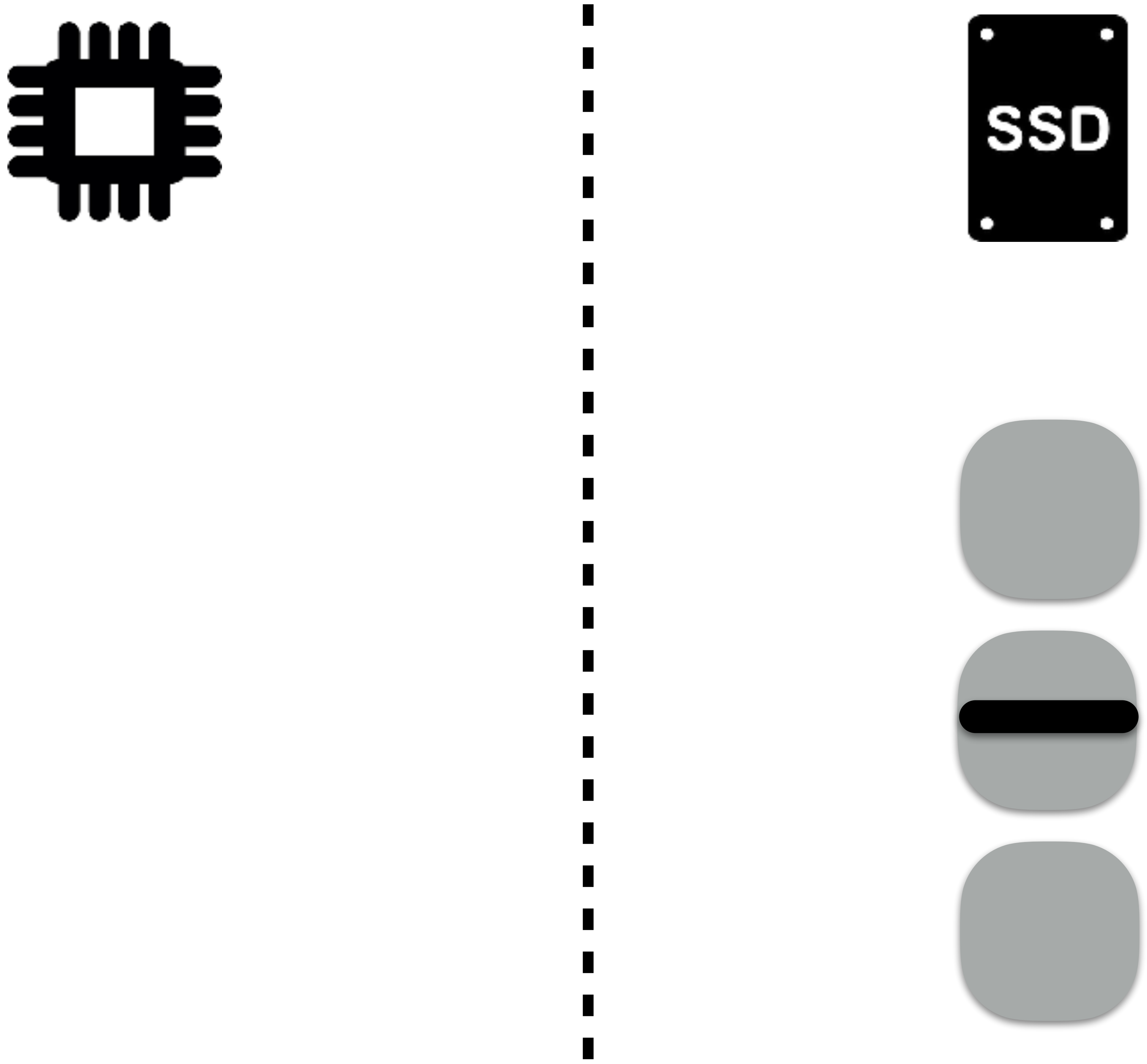


write data



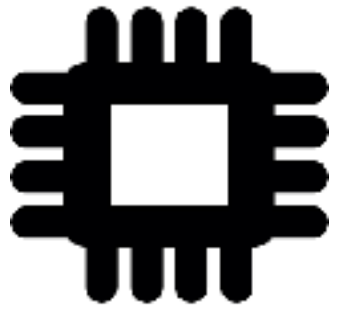

SSD
write data

In-Place Writes

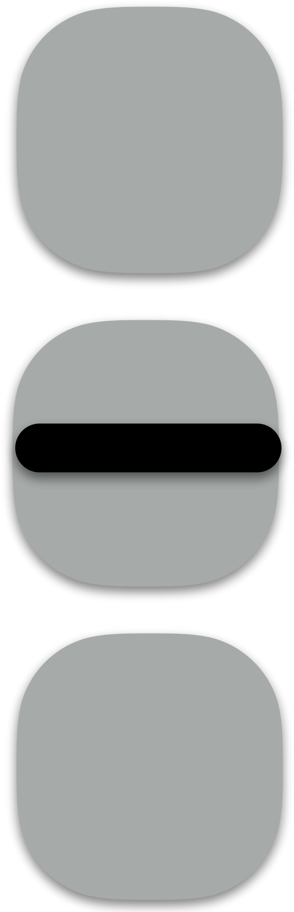


write data

In-Place Writes

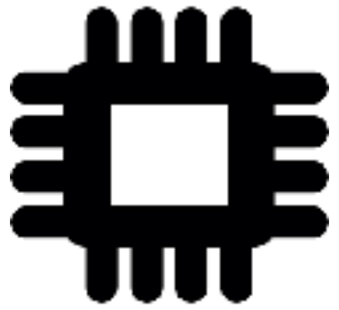


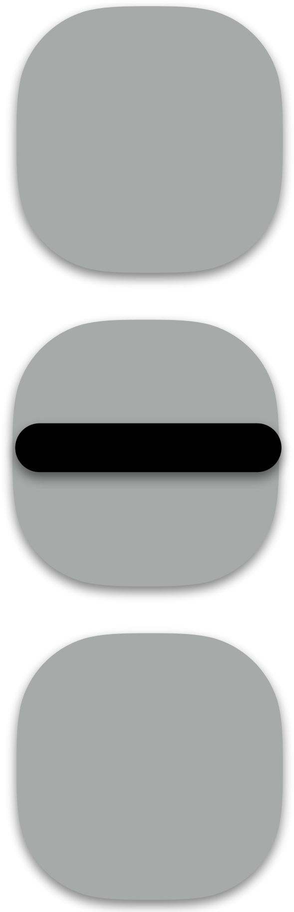
write data



B-trees

In-Place Writes

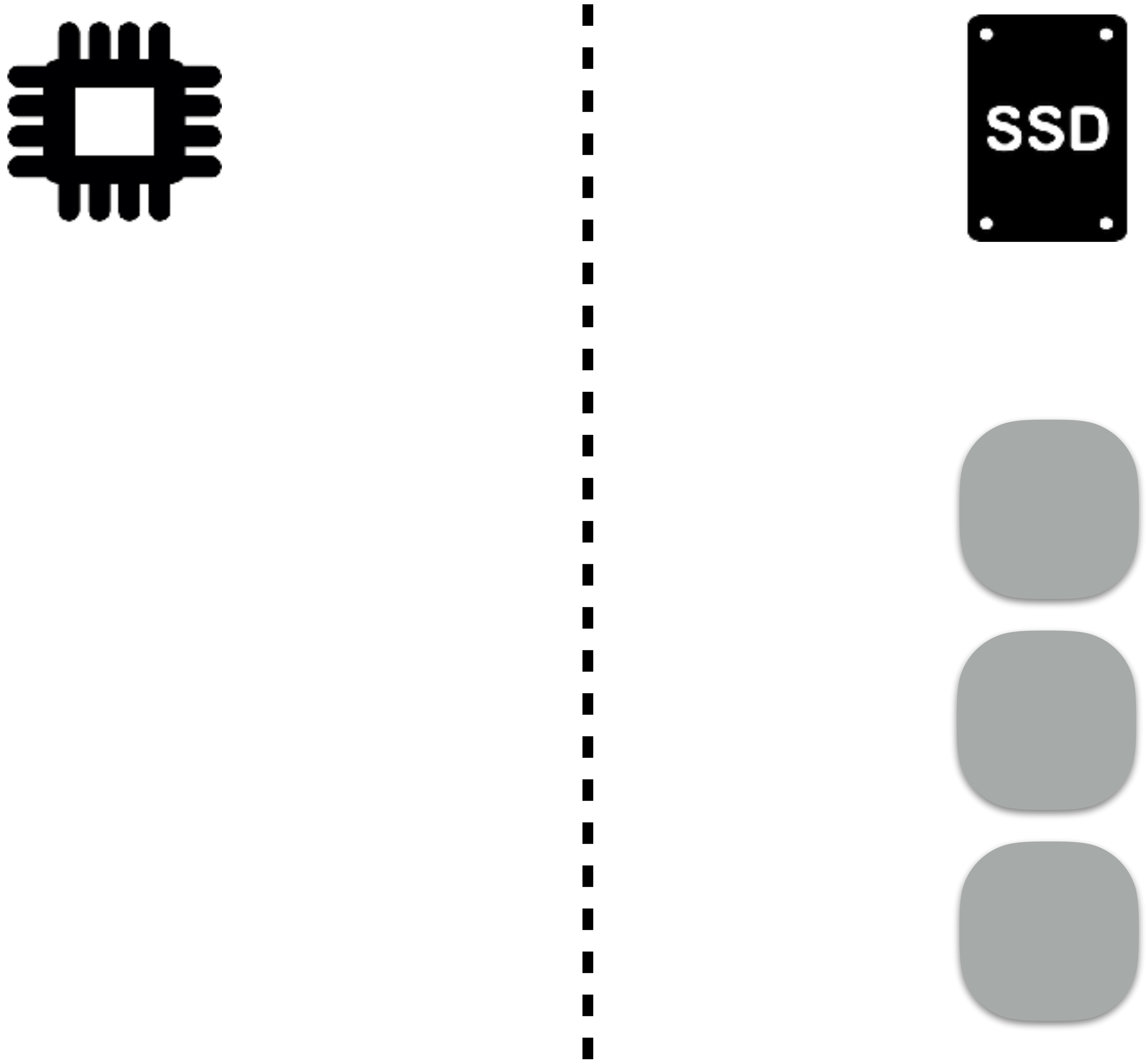




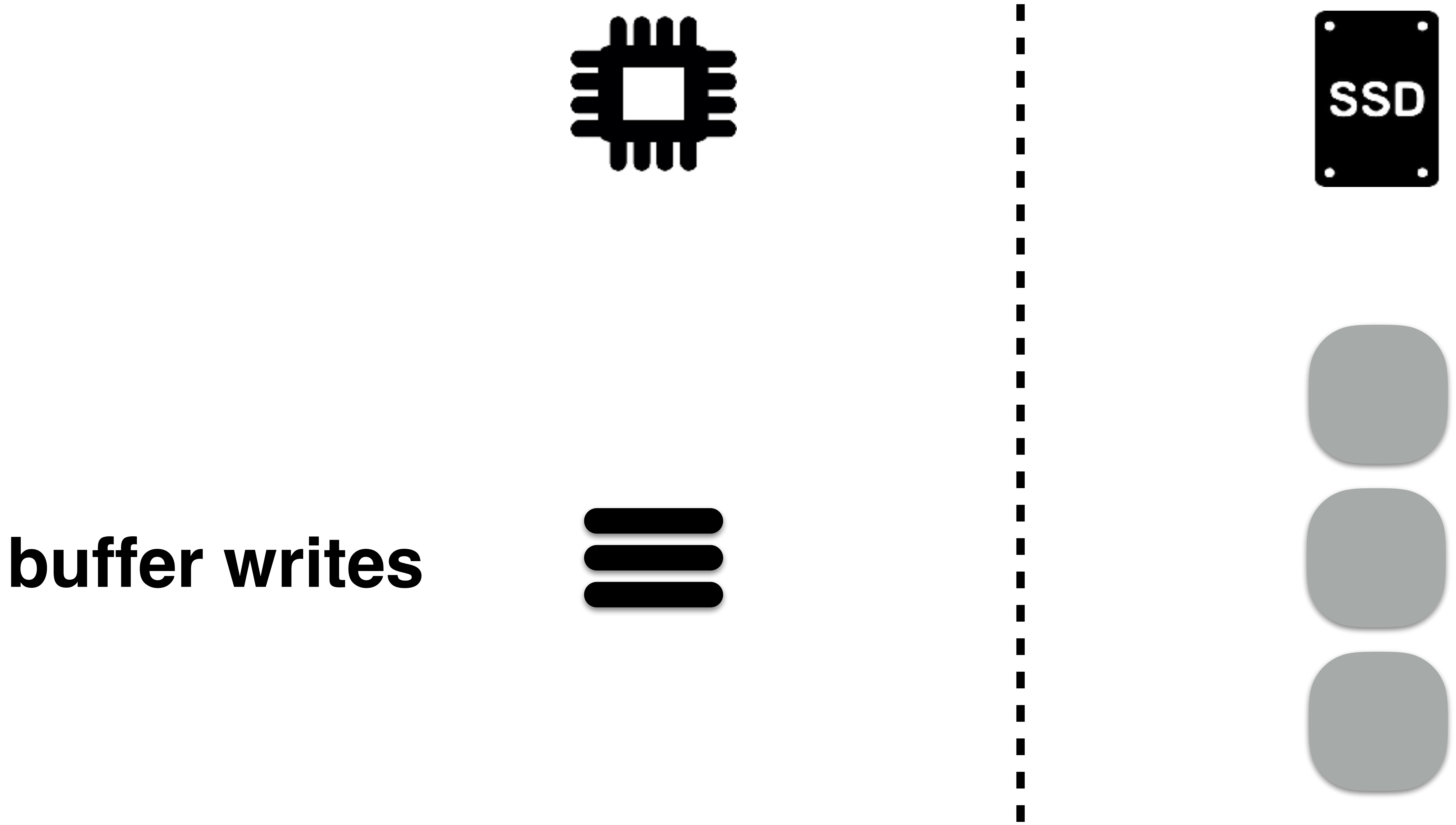
B-trees

write data

Log-Structured Writes

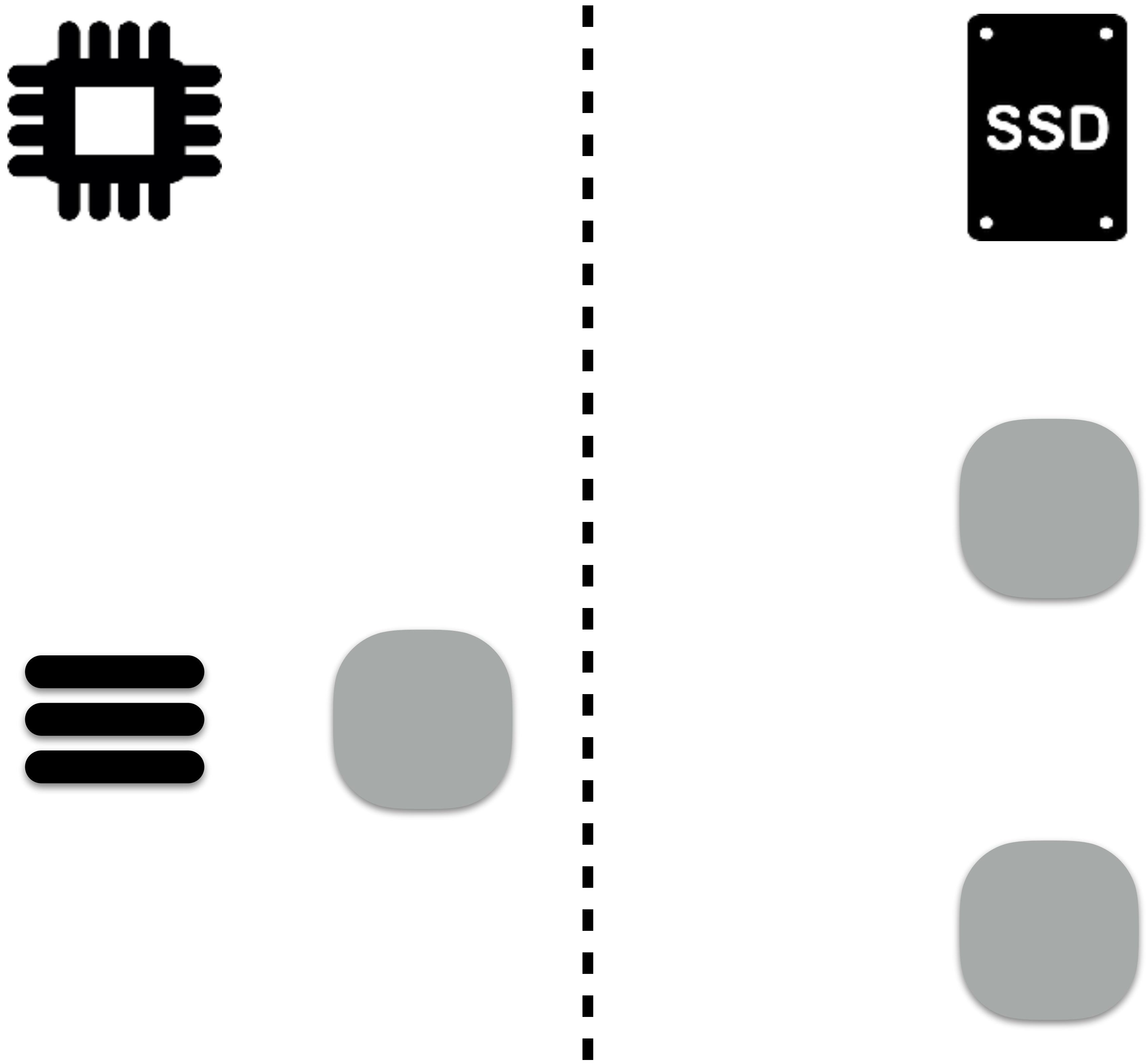


Log-Structured Writes

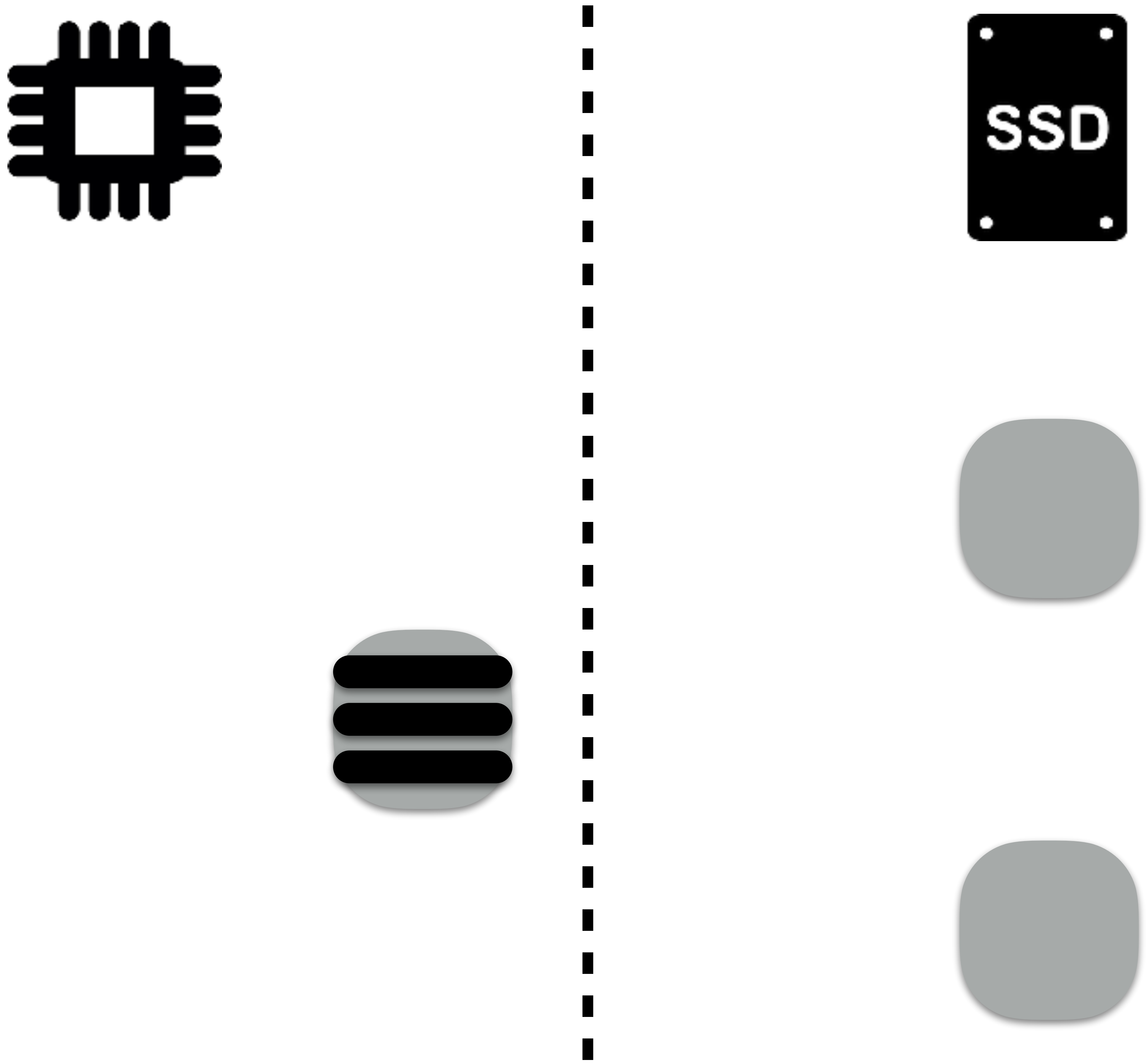


Log-Structured Writes

buffer writes

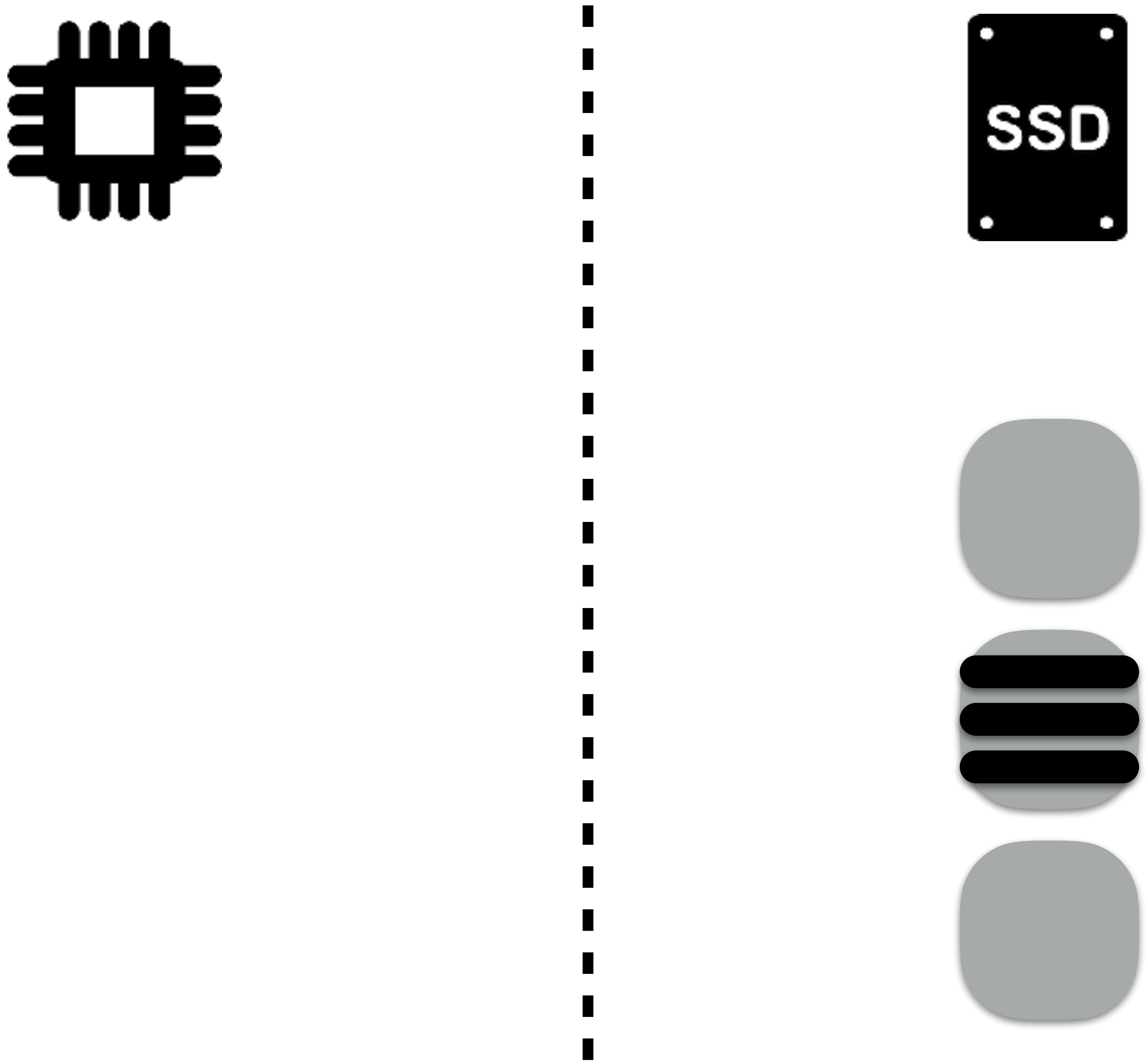


Log-Structured Writes



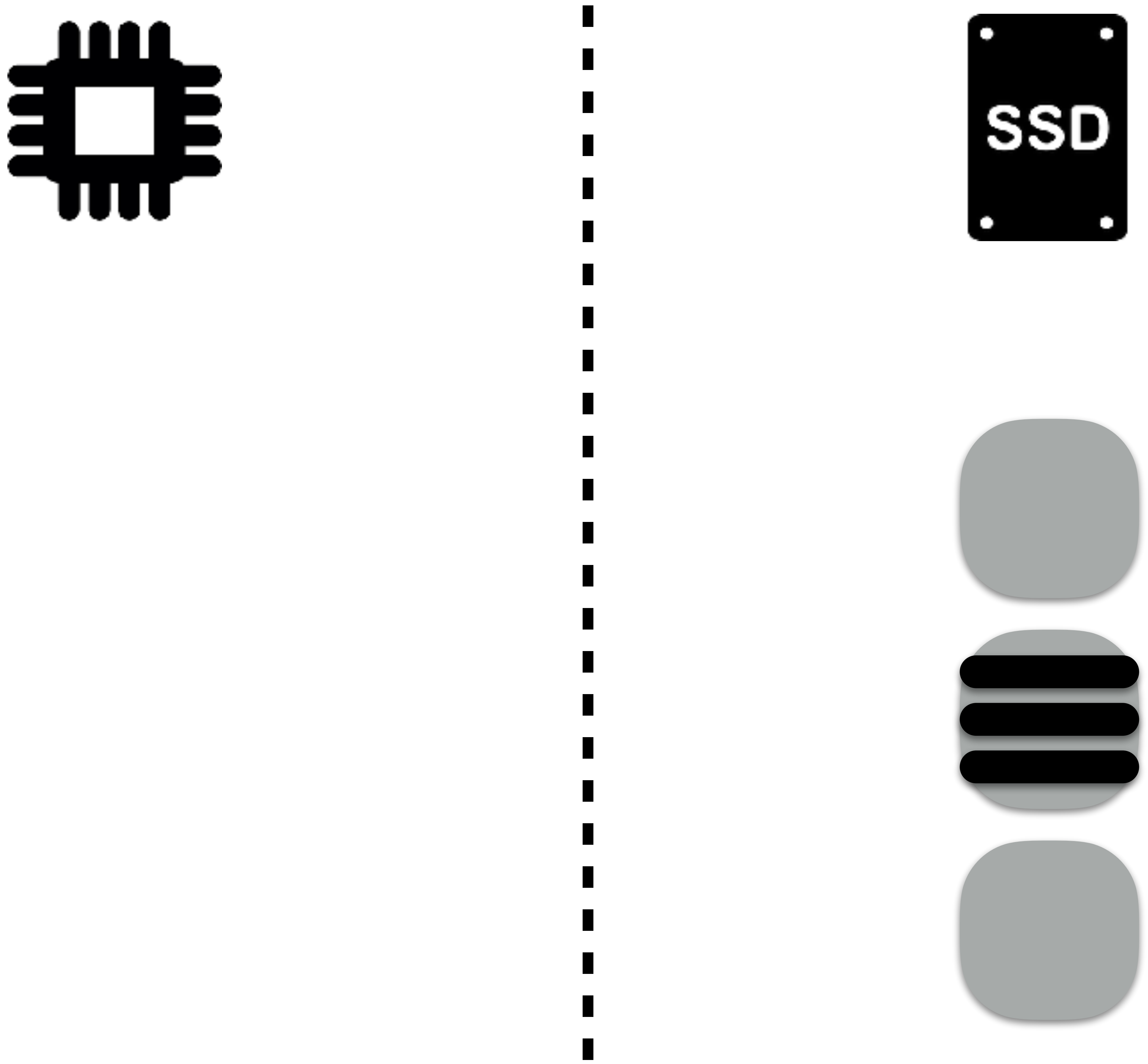
buffer writes

Log-Structured Writes



buffer writes

Log-Structured Writes



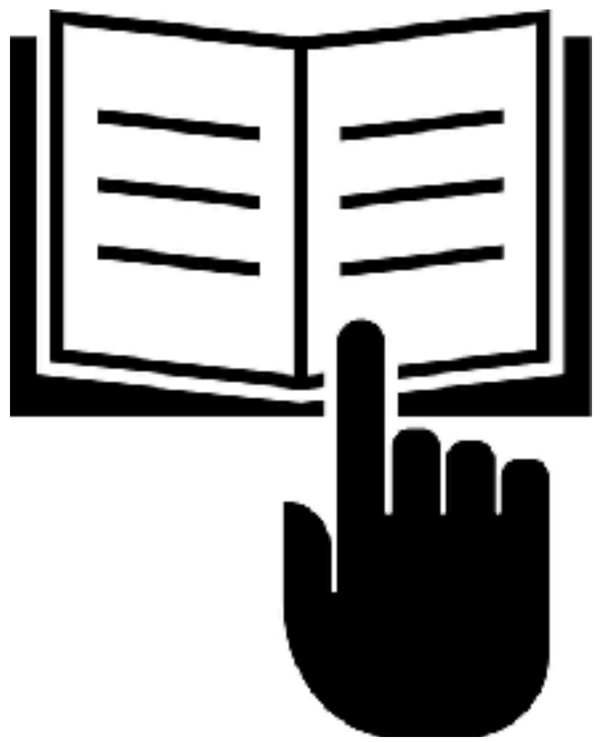
buffer writes

Log-Structured KV-Stores

fast writes

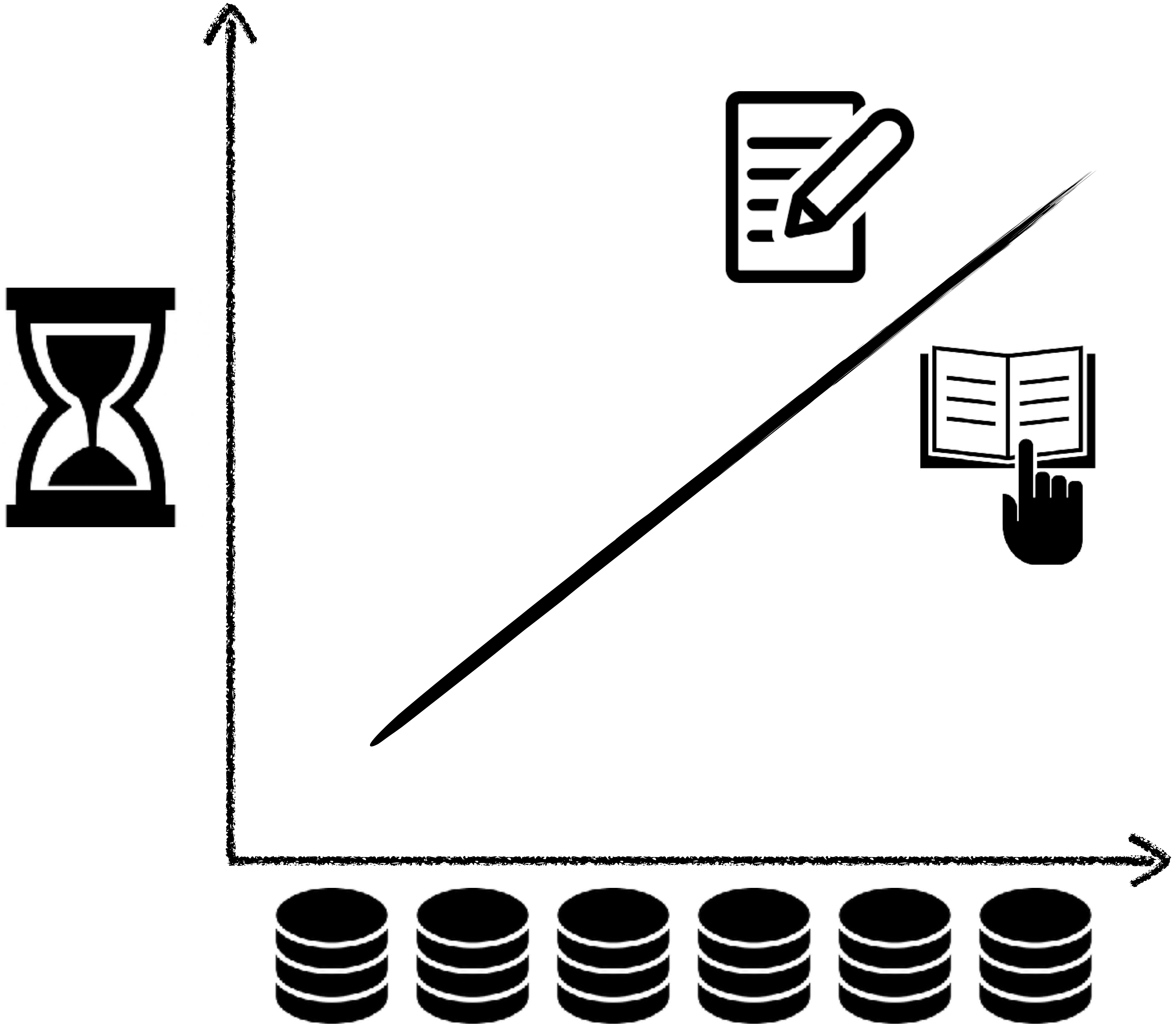
Log-Structured KV-Stores

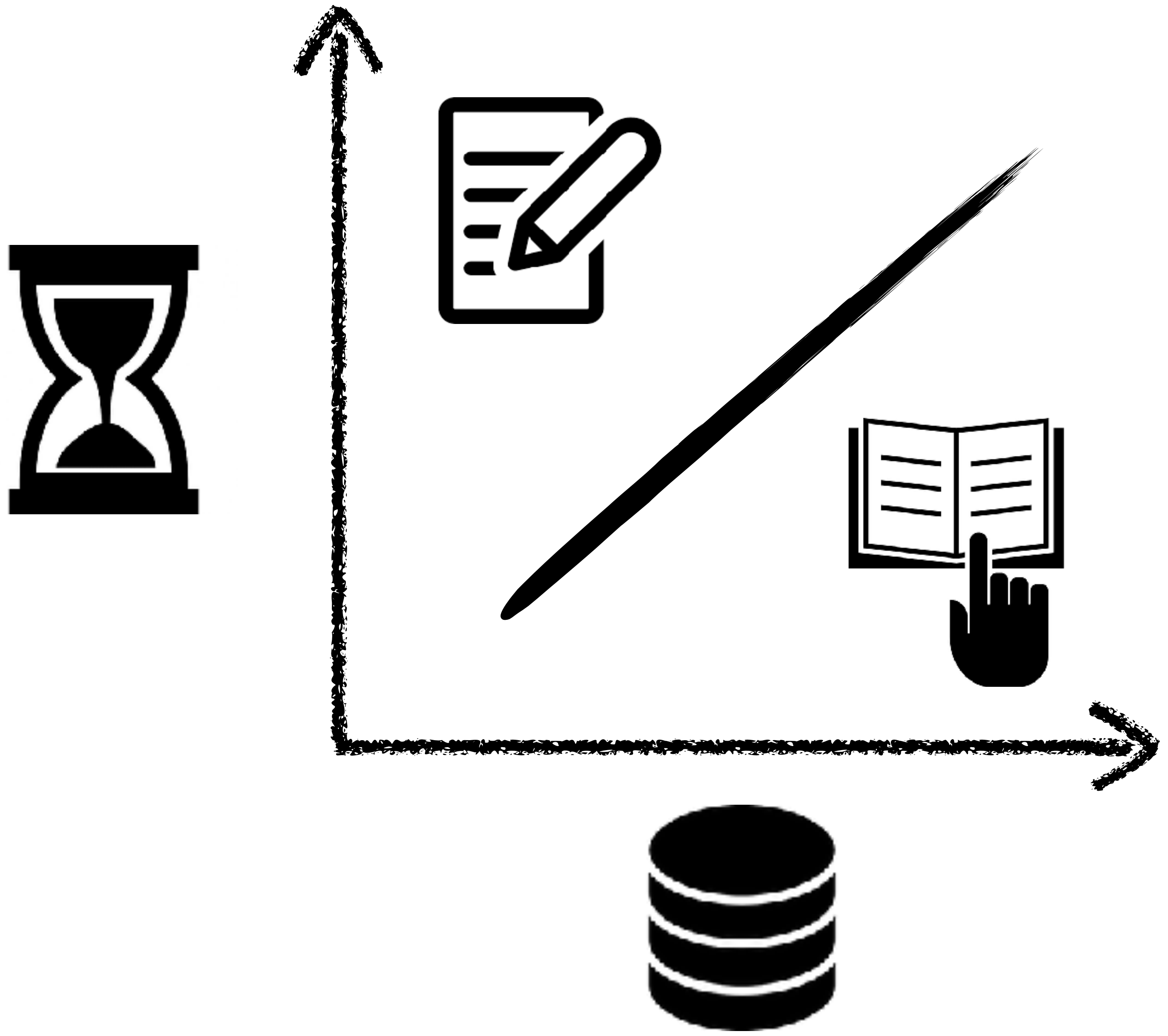
fast writes

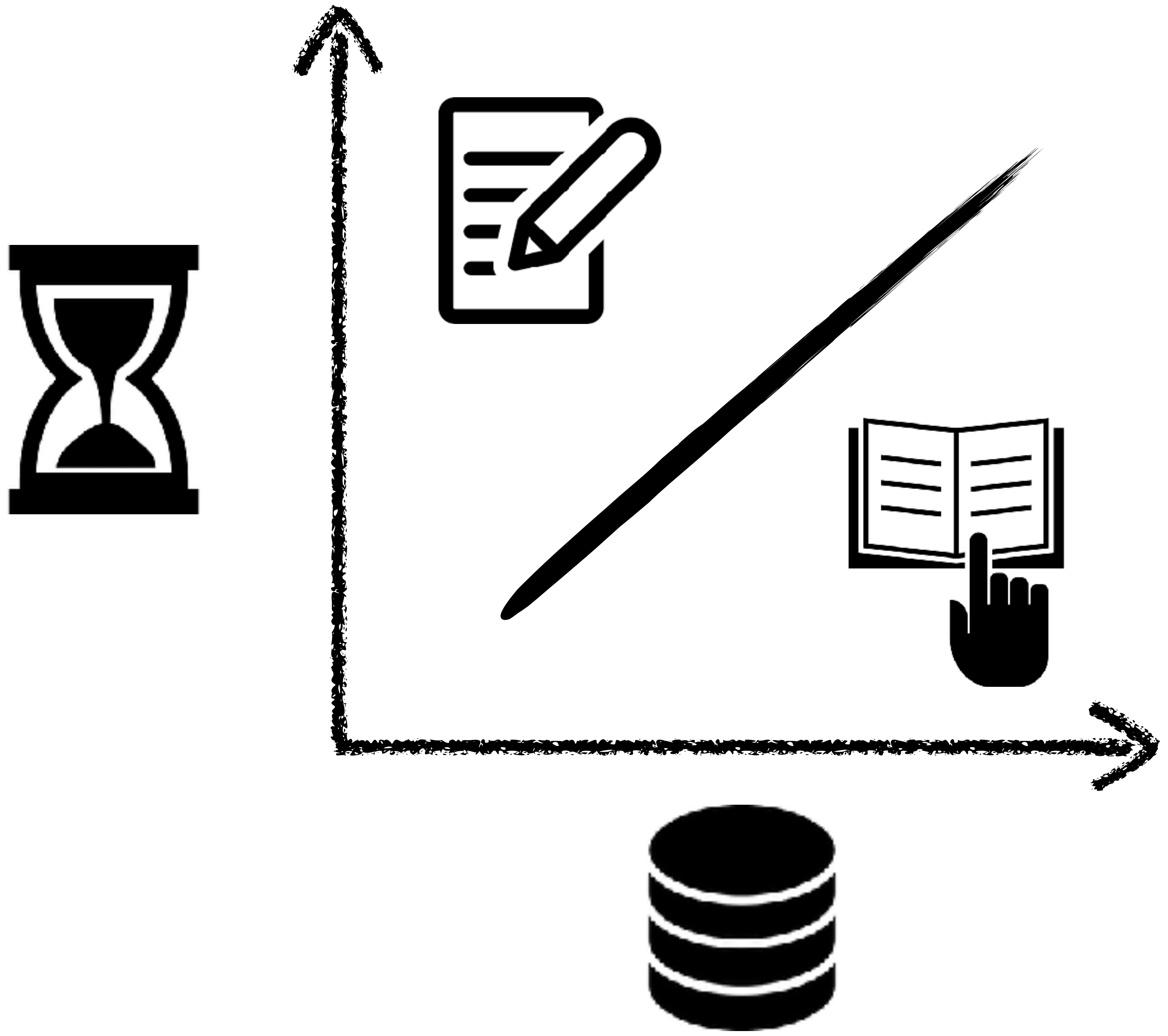


fast reads

massive data







Background

Background

The Log-Structured Merge-Tree

Background

LSM-tree

buffer

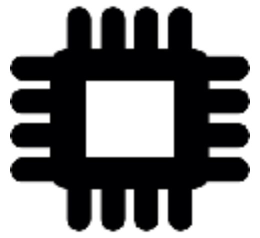
writes

buffer

key value pairs

↓

buffer

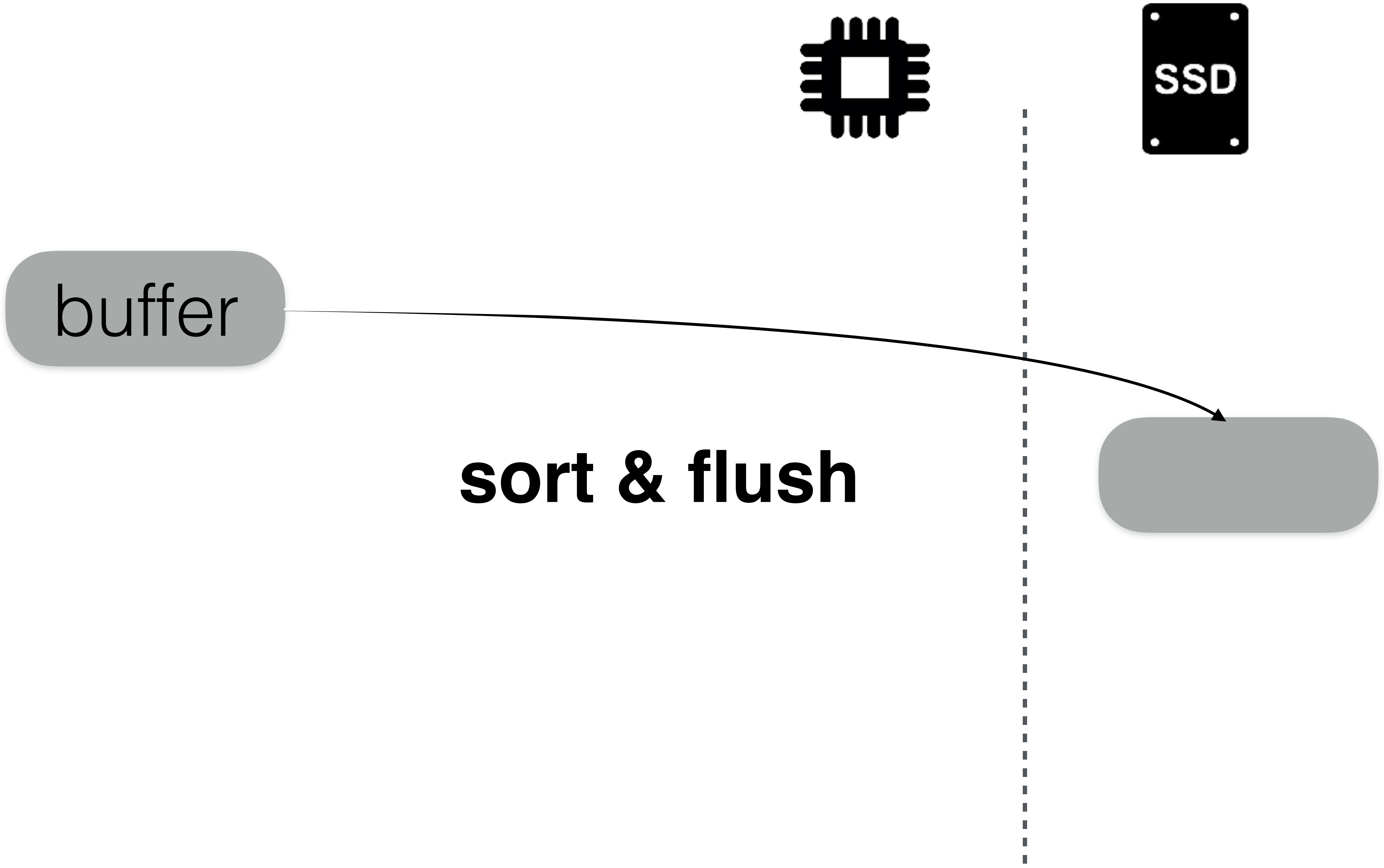


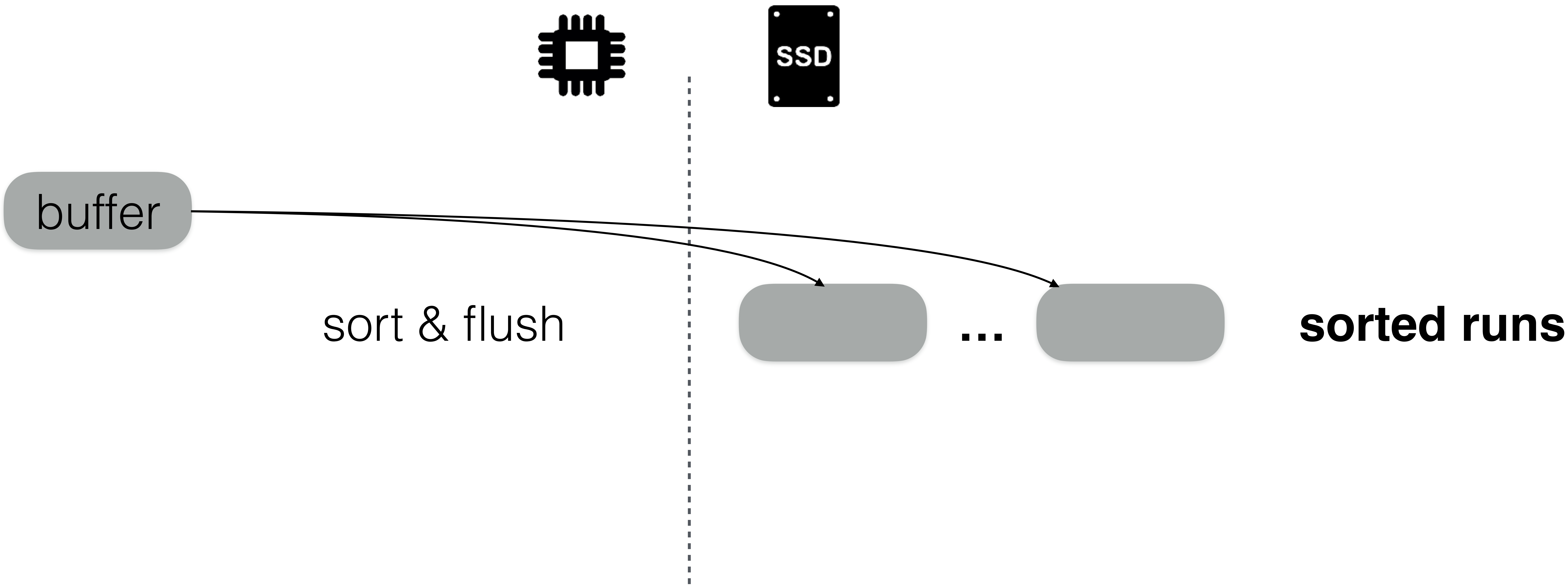
key	**value**
Sherlock:	a fictional detective
Waldo:	an inconspicuous traveler

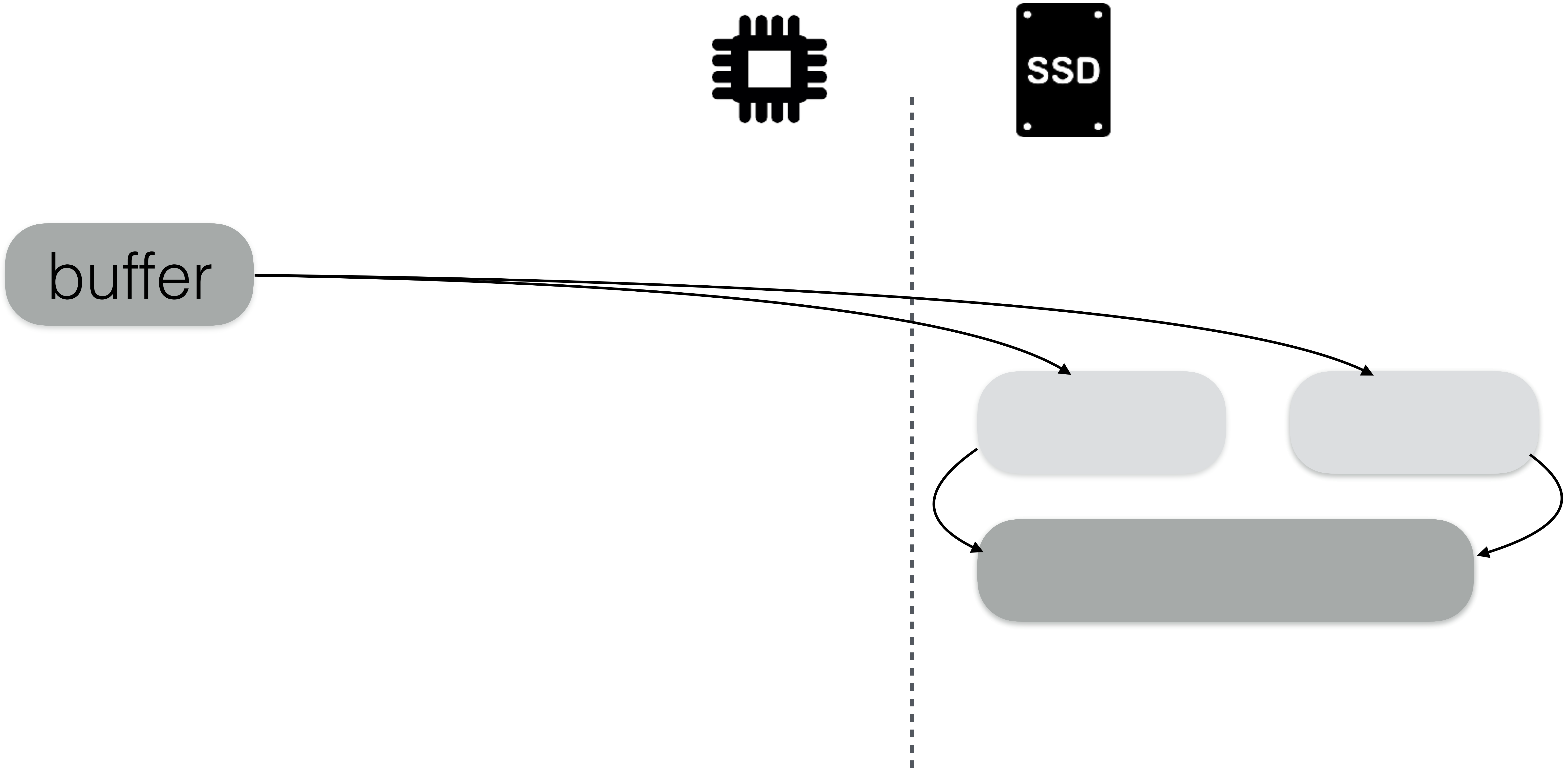
buffer

DASlab
@ Harvard SEAS

buffer gets full

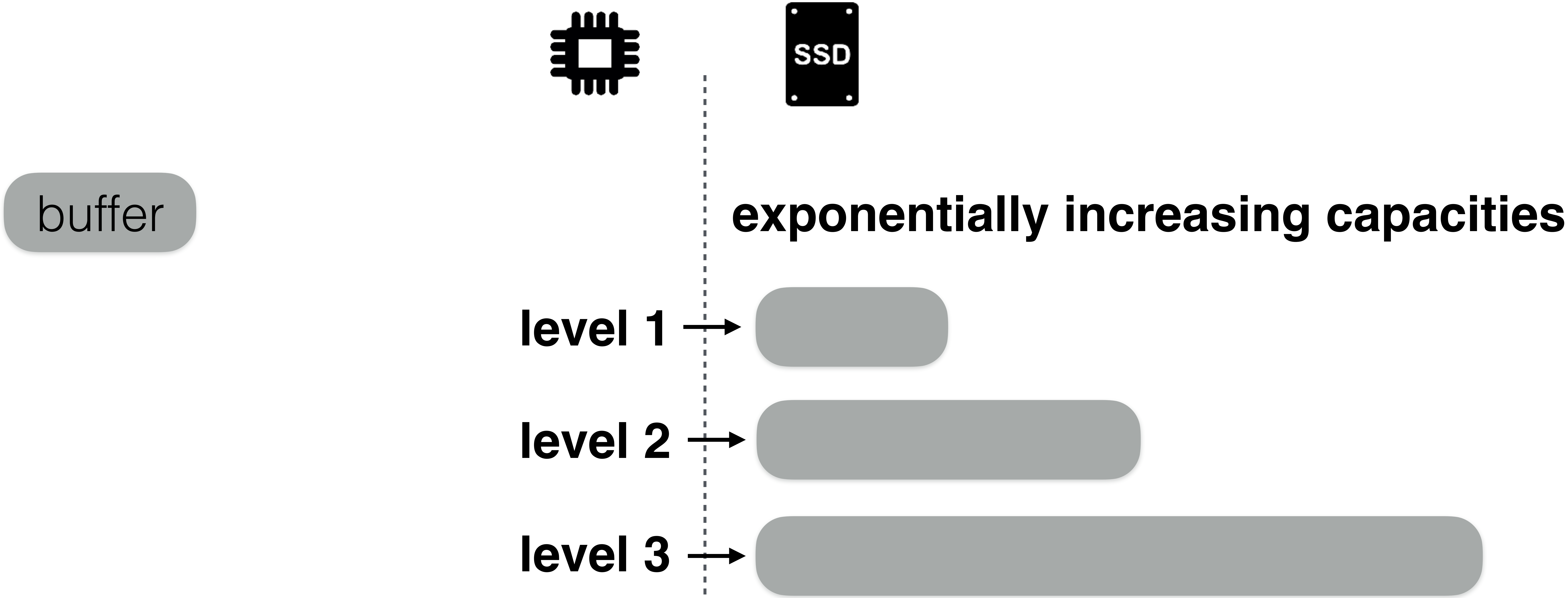

SSD
buffer
sort & flush

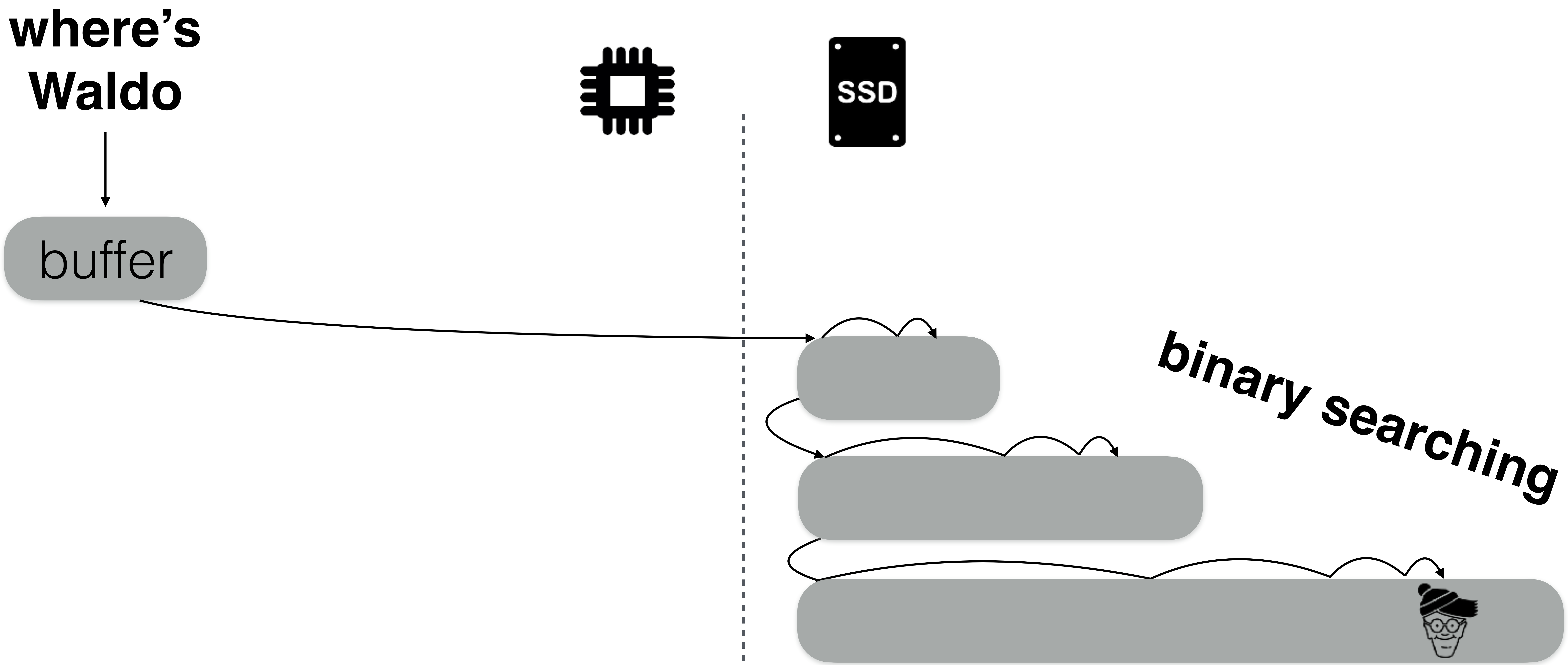

SSD
buffer
sort & flush
...
sorted runs

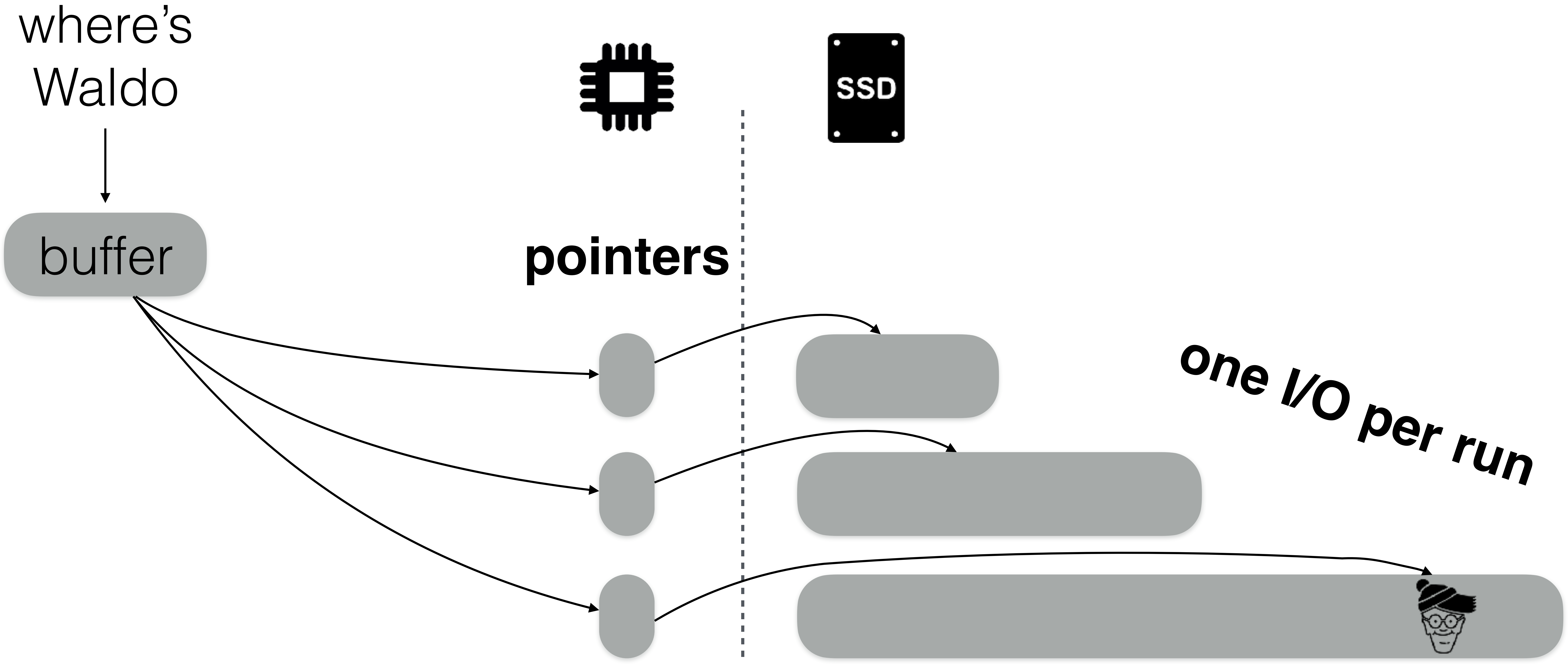


sort-merge

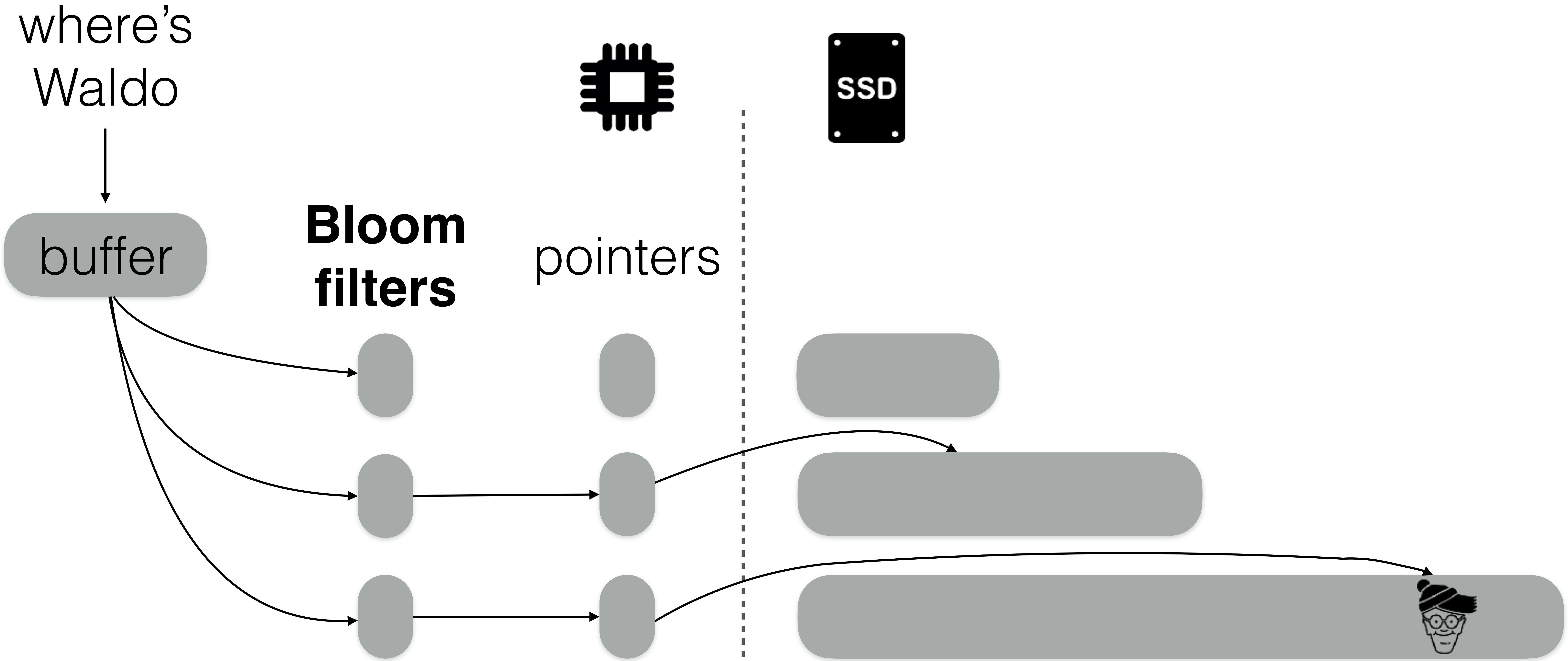
@ Harvard SEAS

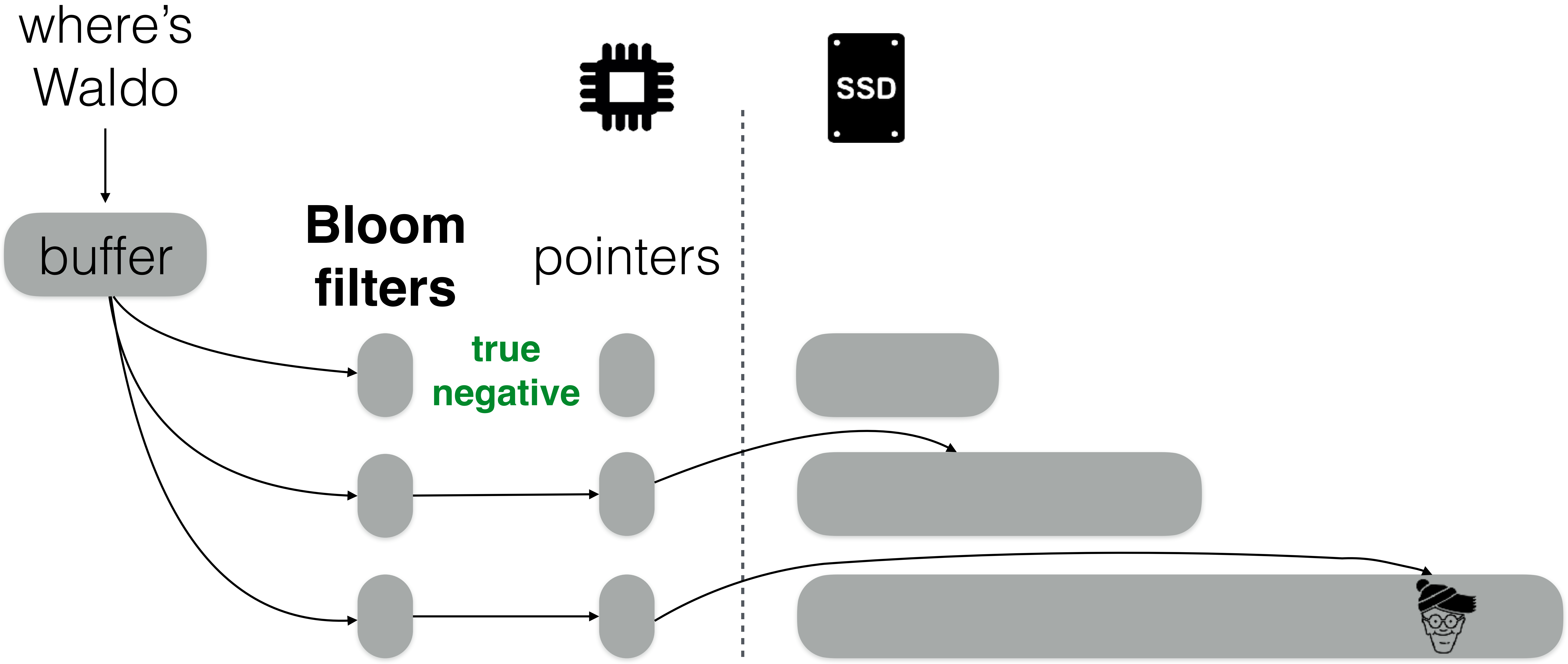

SSD
buffer
exponentially increasing capacities
level 1
level 2
level 3


where’s
Waldo
buffer
SSD
binary searching

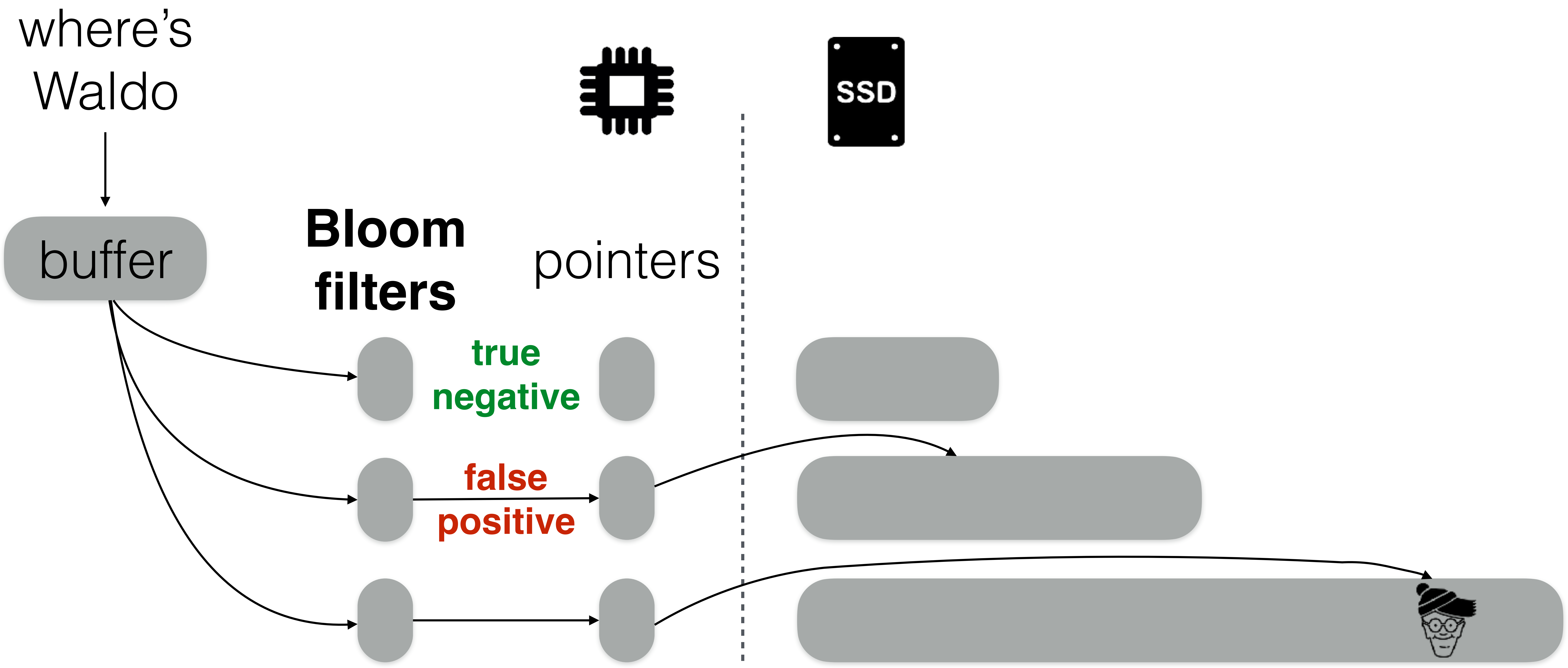


DASlab
@ Harvard SEAS

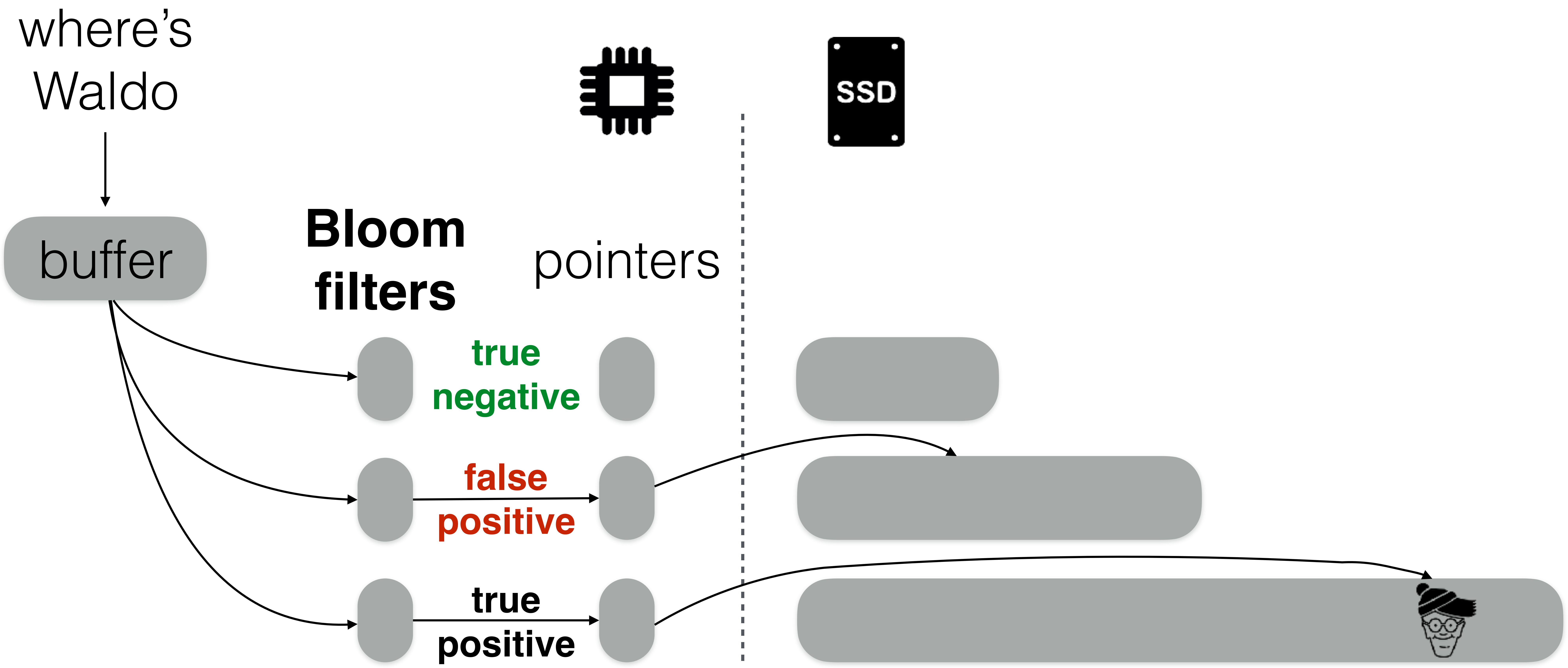

where's
Waldo
buffer
Bloom
filters
pointers
SSD



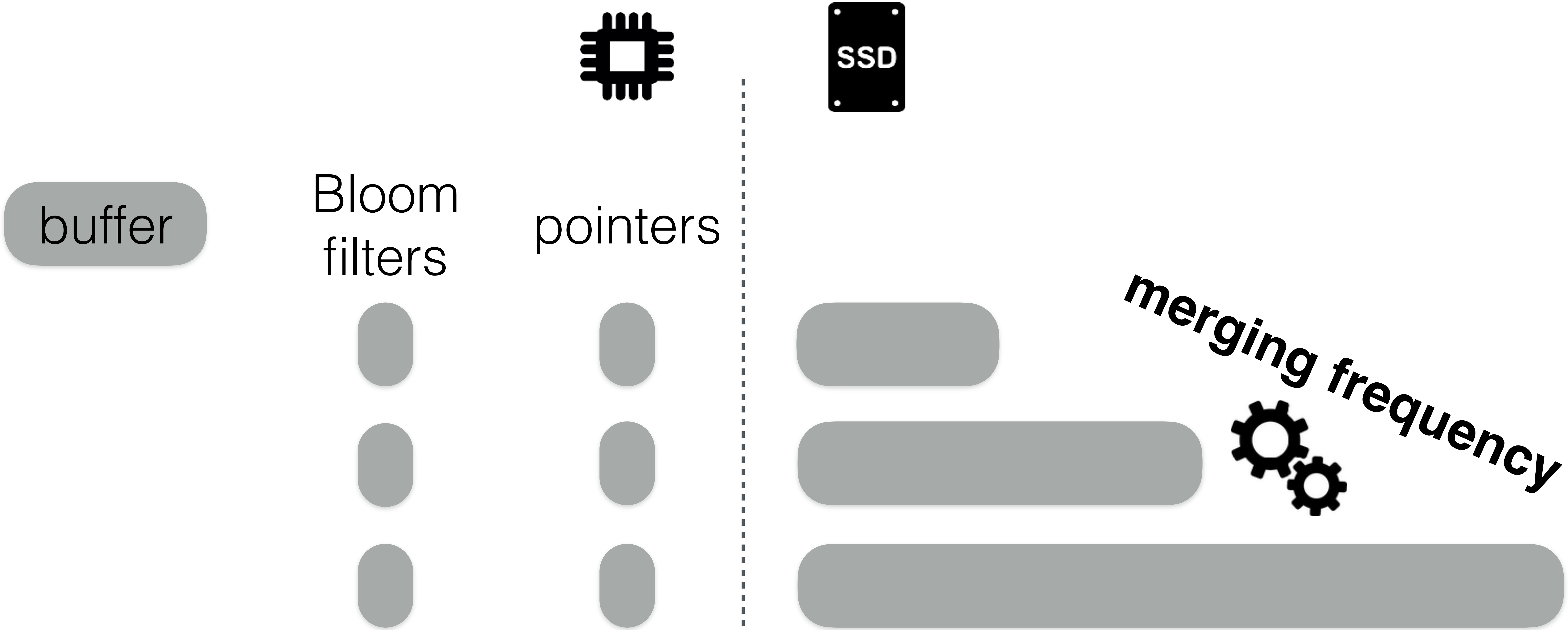
DASlab @ Harvard SEAS



DASlab
@ Harvard SEAS



DASlab
@ Harvard SEAS


SSD
buffer
Bloom filters
pointers
merging frequency

merging

writes

reads

merging ↑

writes ↑

↓ **reads**

merging

Tiering
write-optimized

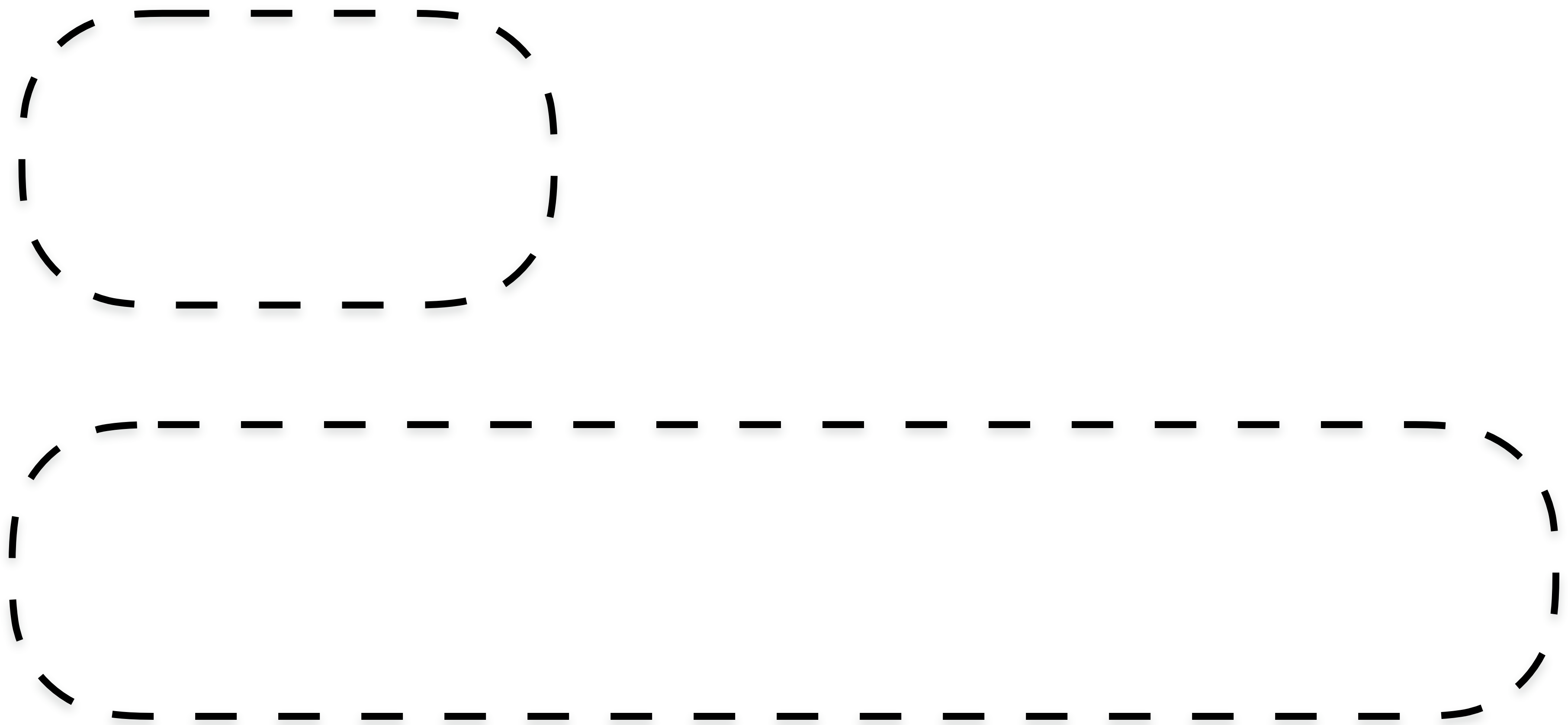
cassandra

Leveling
read-optimized

RocksDB

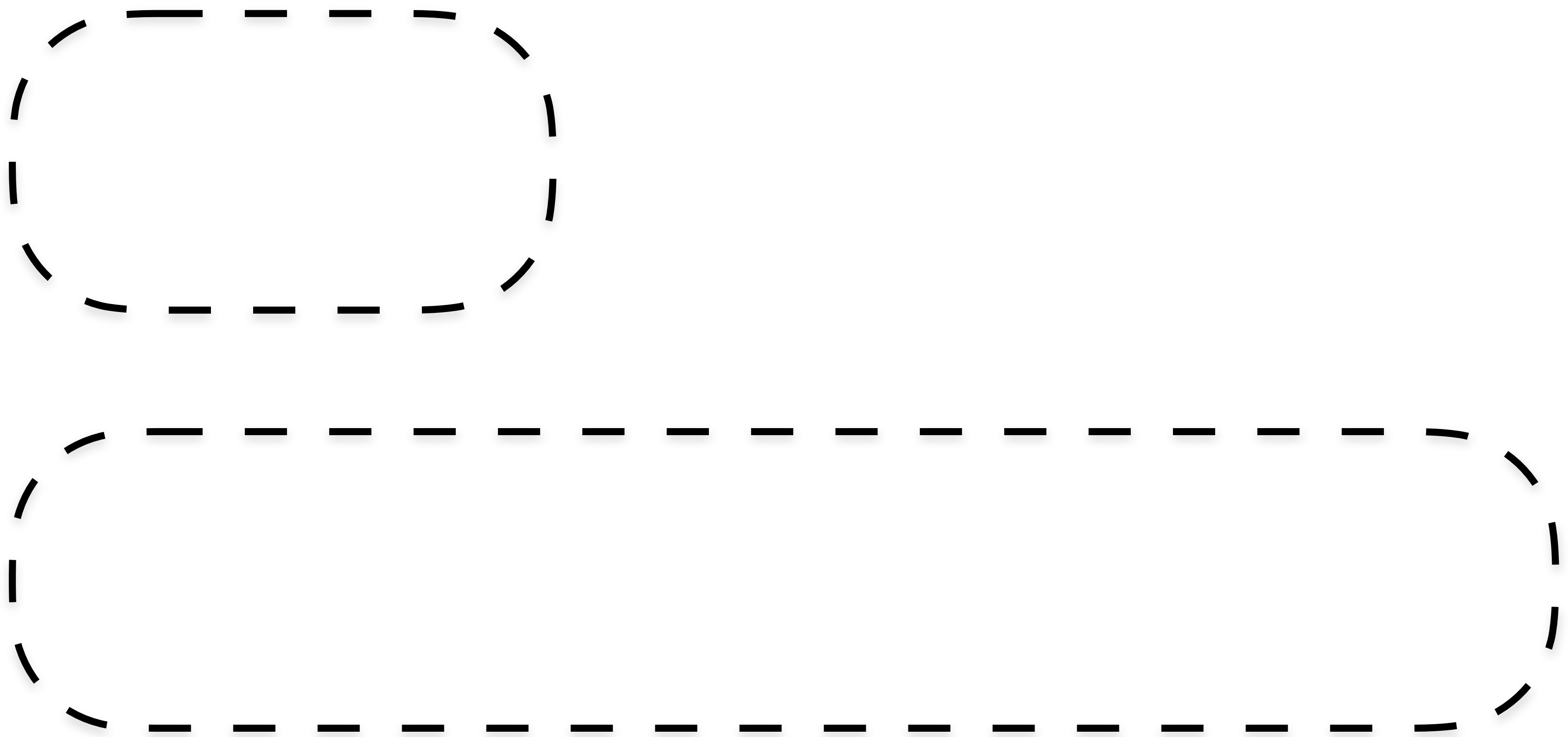
Tiering

write-optimized



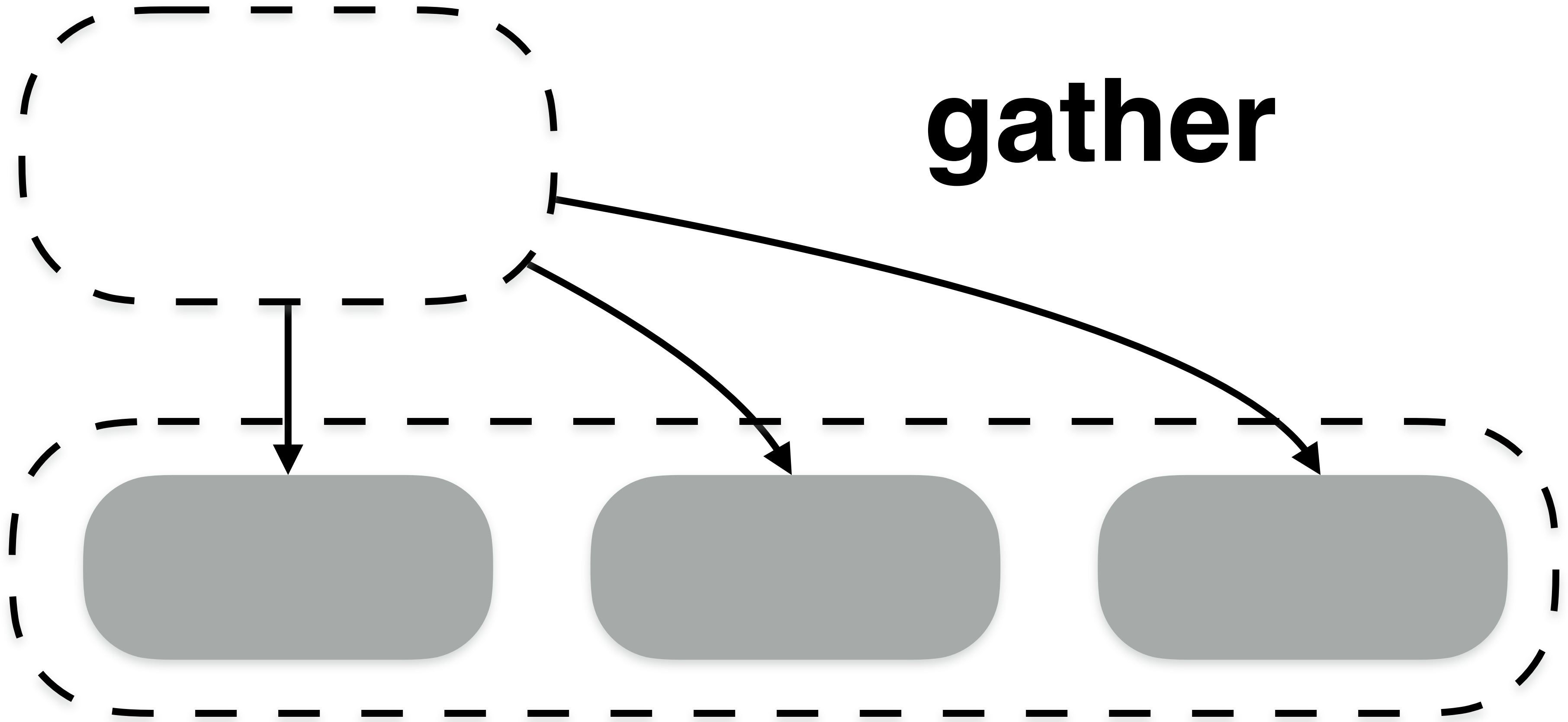
Leveling

read-optimized



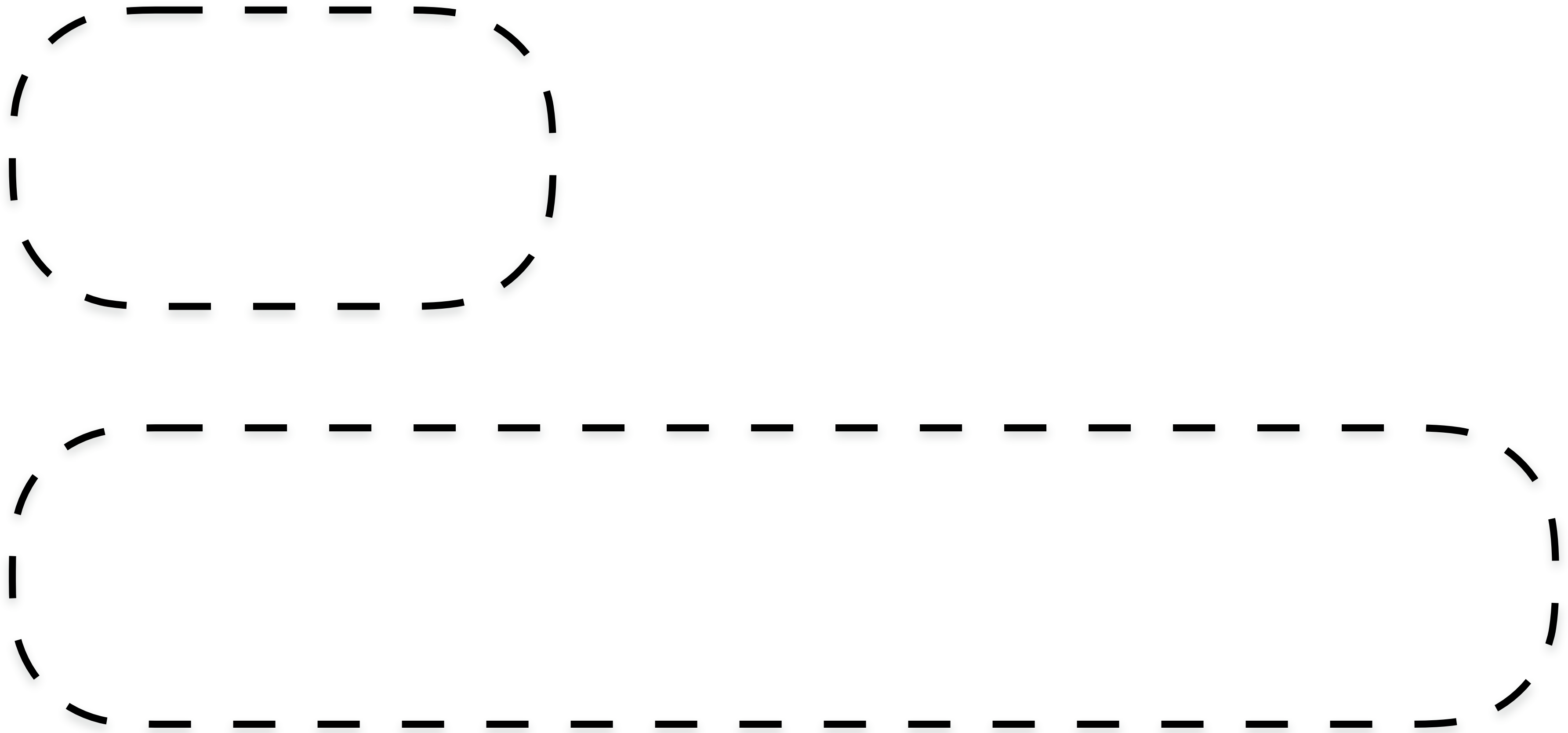
Tiering

write-optimized



Leveling

read-optimized



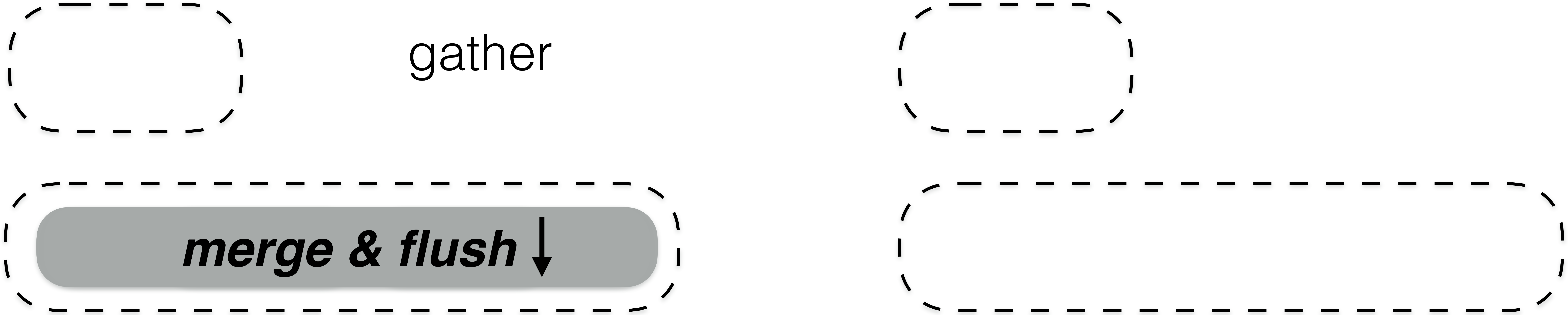
DASlab @ Harvard SEAS

Tiering

write-optimized

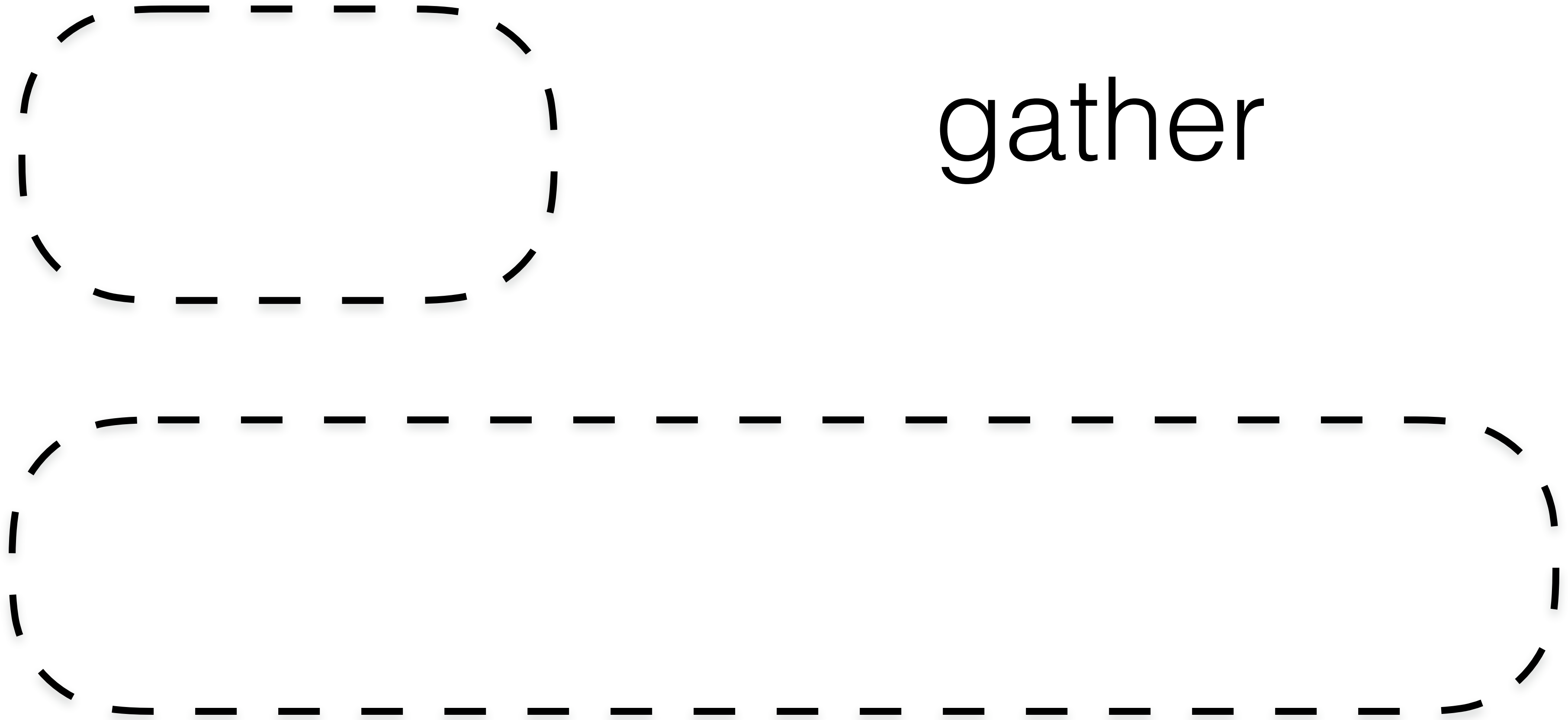
Leveling

read-optimized



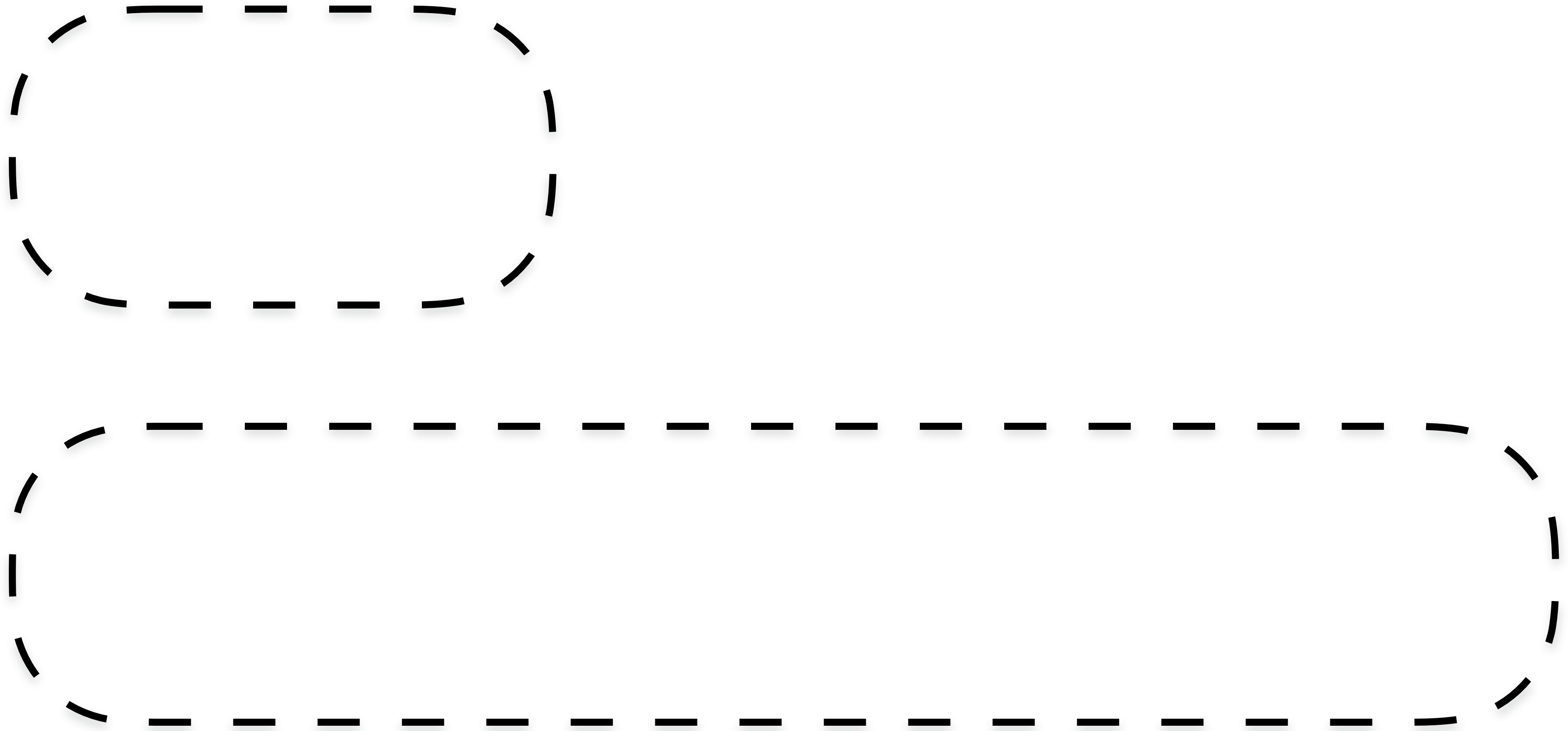
Tiering

write-optimized



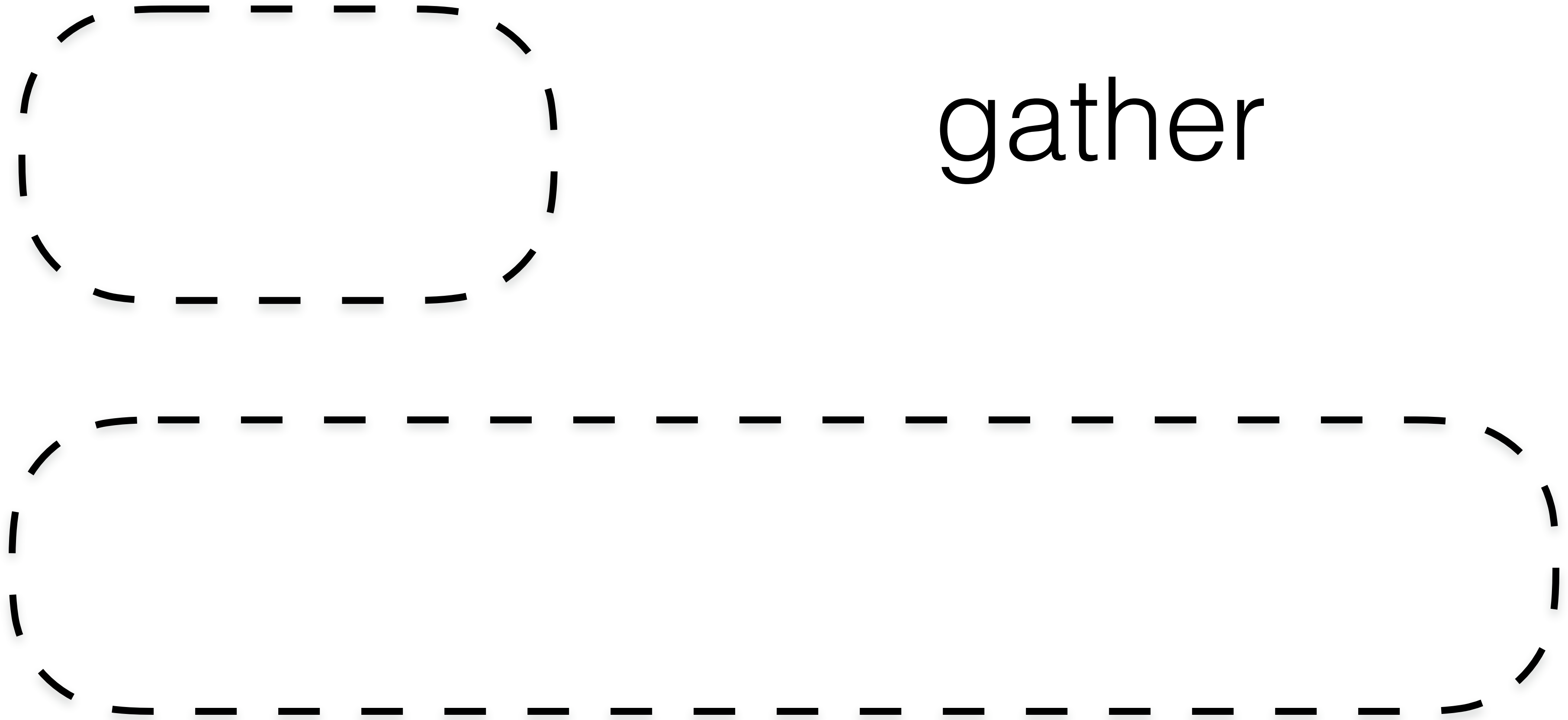
Leveling

read-optimized



Tiering

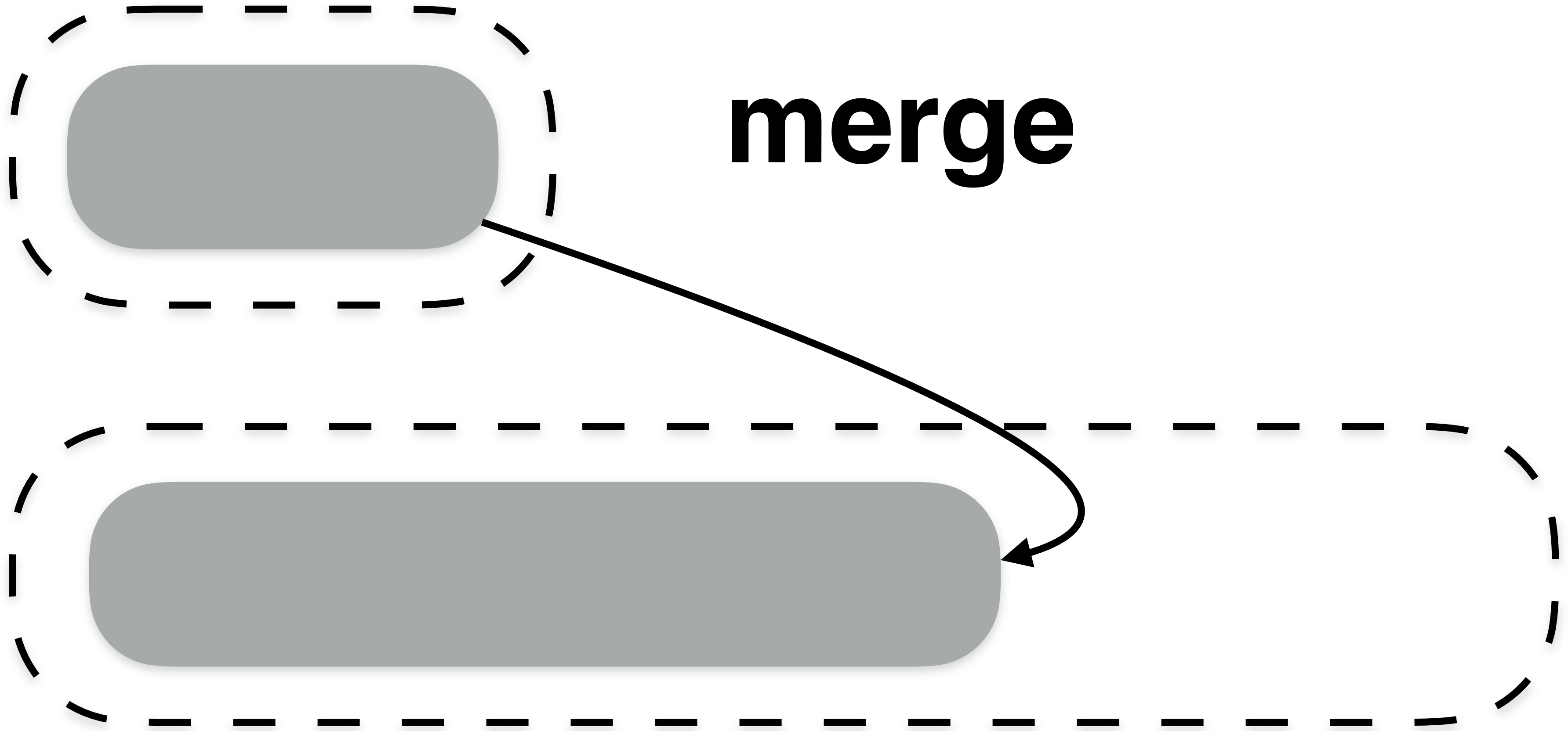
write-optimized



gather

Leveling

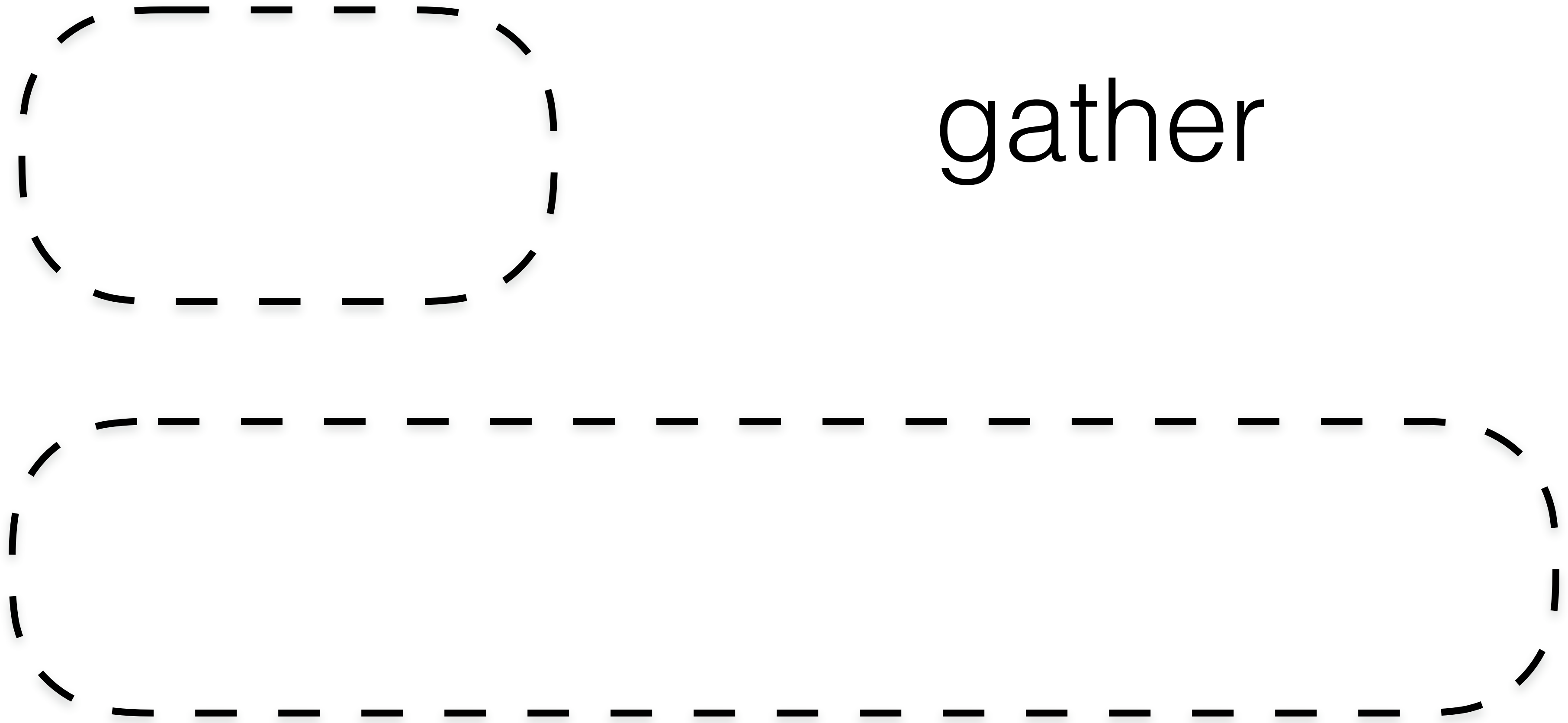
read-optimized



merge

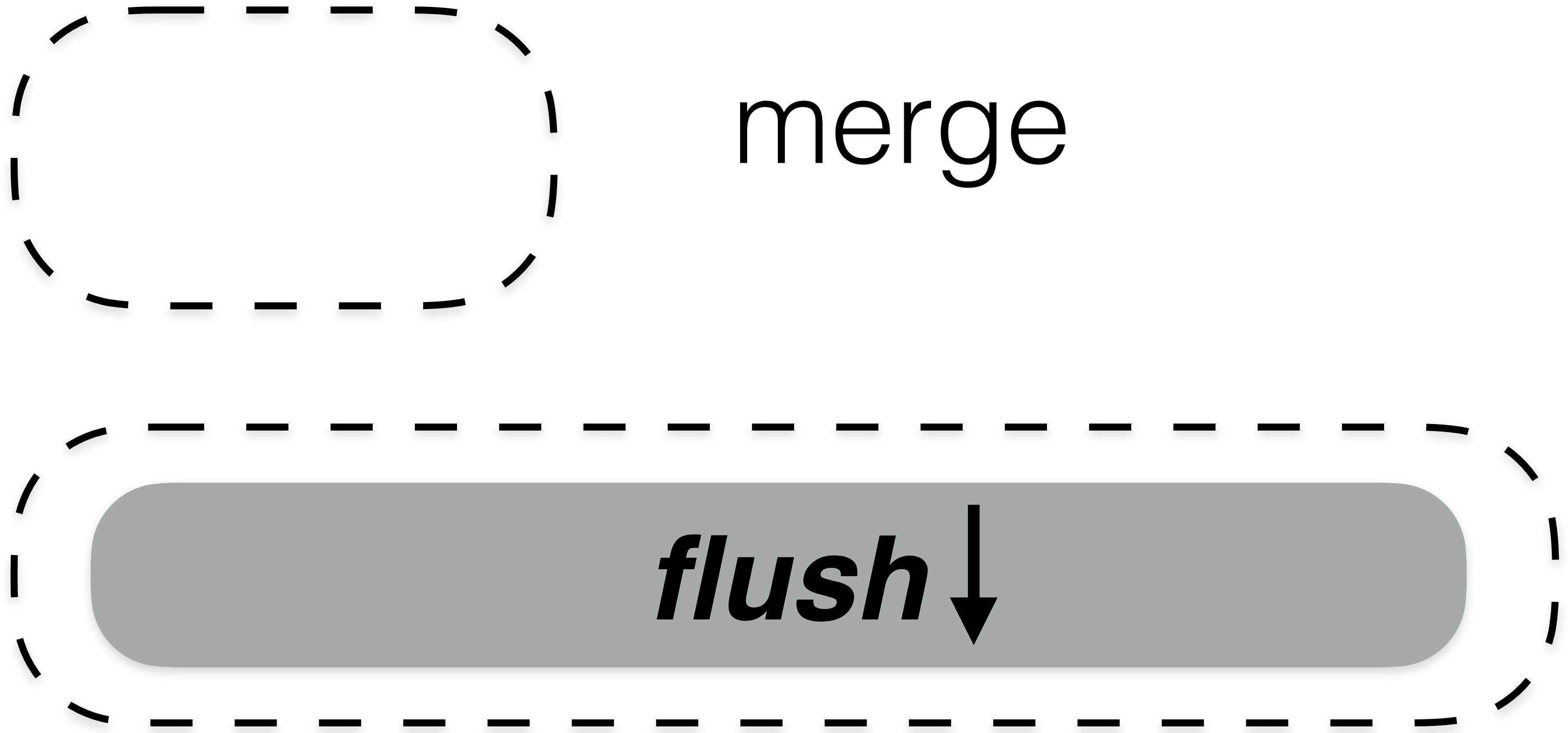
Tiering

write-optimized



Leveling

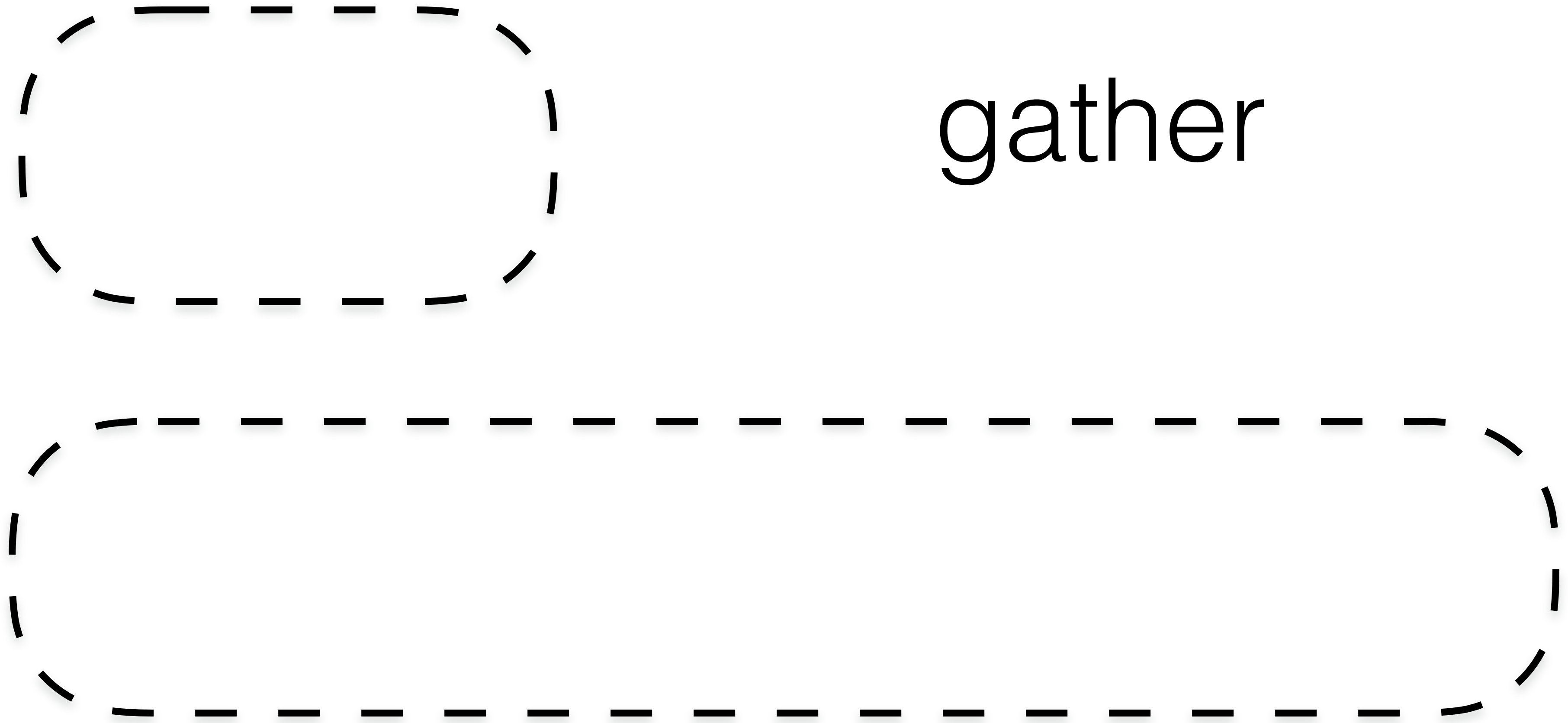
read-optimized



DASlab @ Harvard SEAS

Tiering

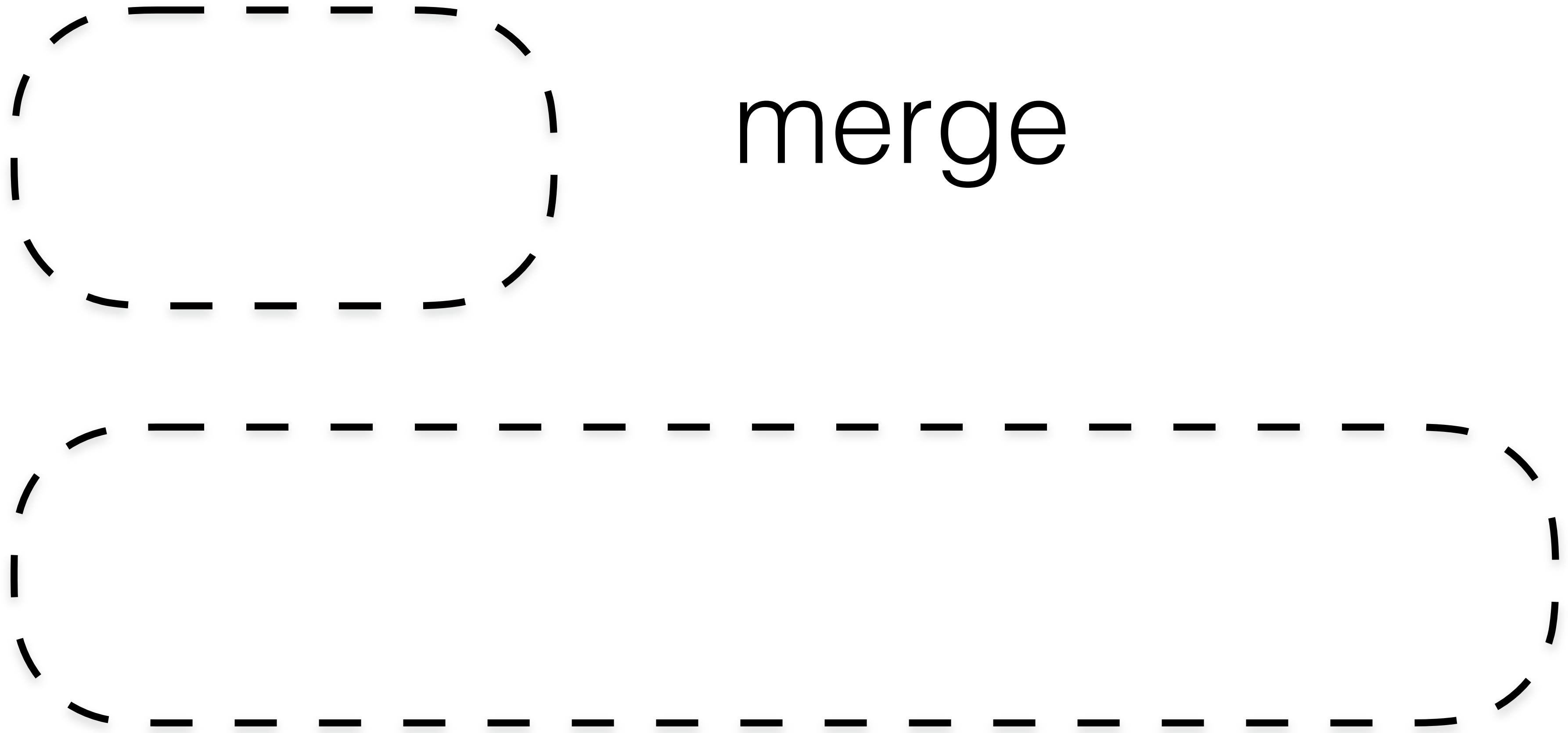
write-optimized



gather

Leveling

read-optimized



merge

Tiering

write-optimized

Leveling

read-optimized

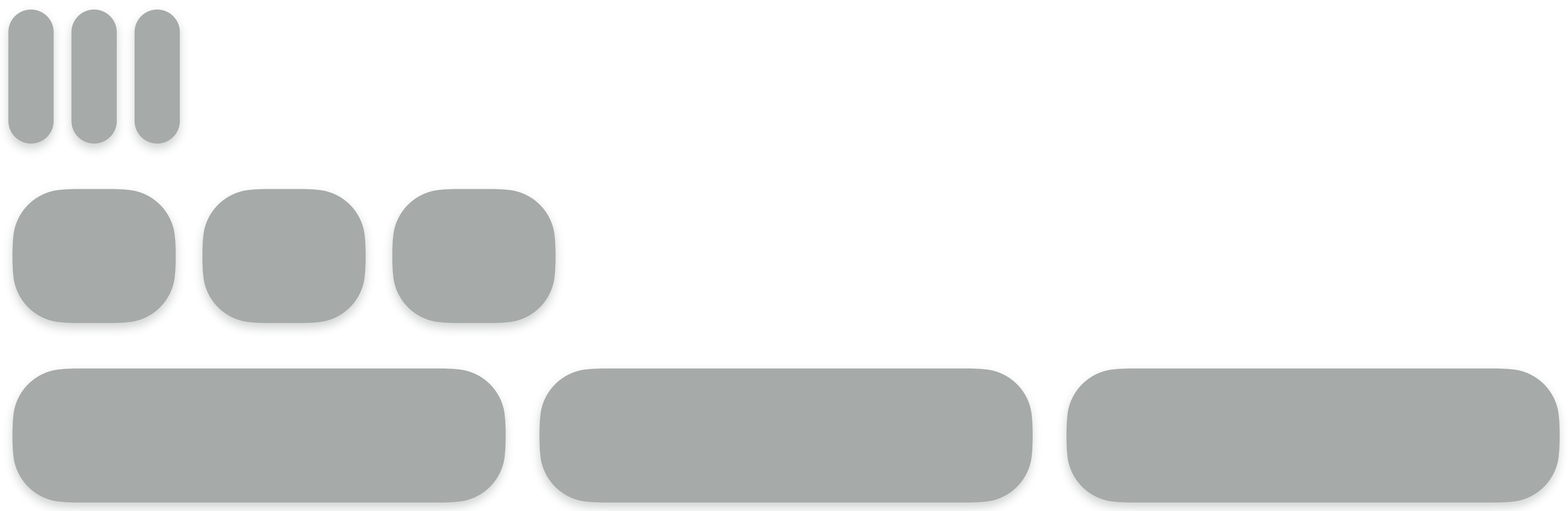
$\log_R(N)$

DASlab @ Harvard SEAS

Tiering

write-optimized

$\log_{R}(N)$



size ratio

Leveling

read-optimized

Tiering	Leveling
write-optimized	read-optimized

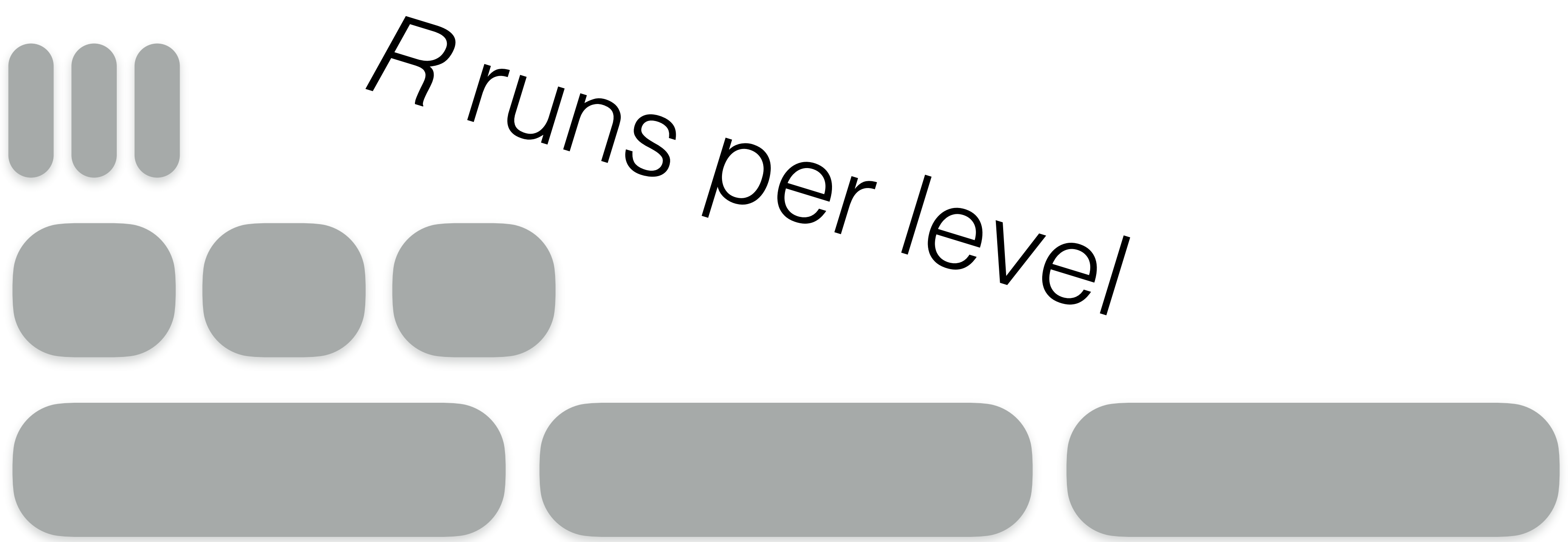
$\log_{R}(N)$

size ratio

DASlab
@ Harvard SEAS

Tiering

write-optimized



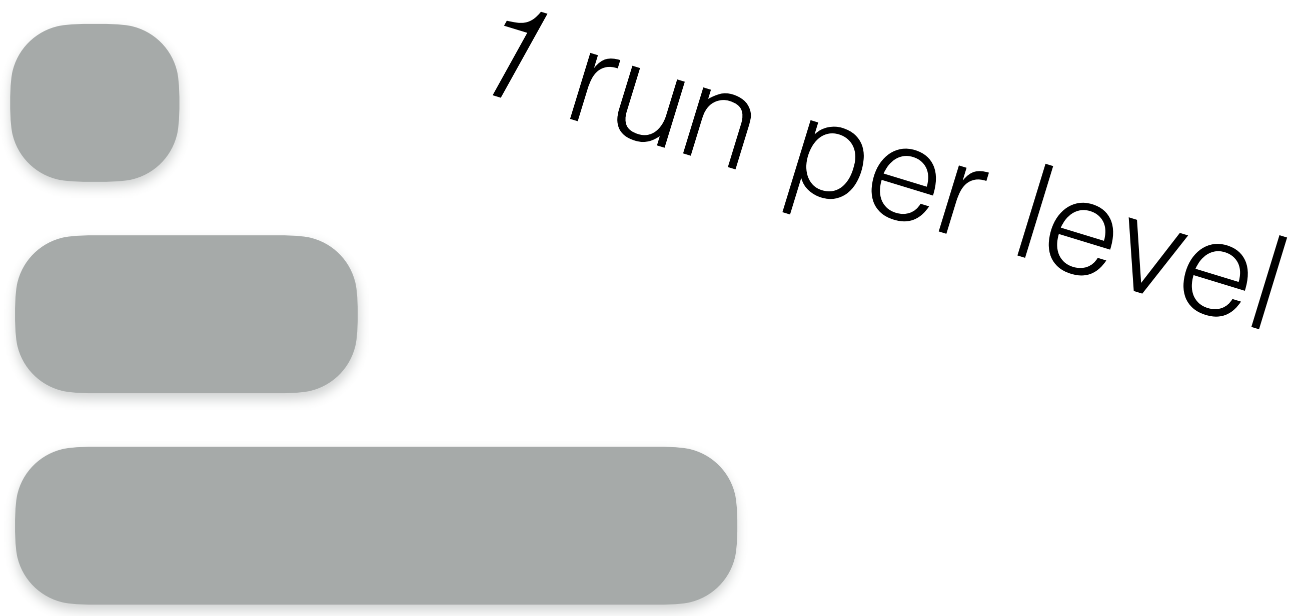
Leveling

read-optimized

size ratio *R*

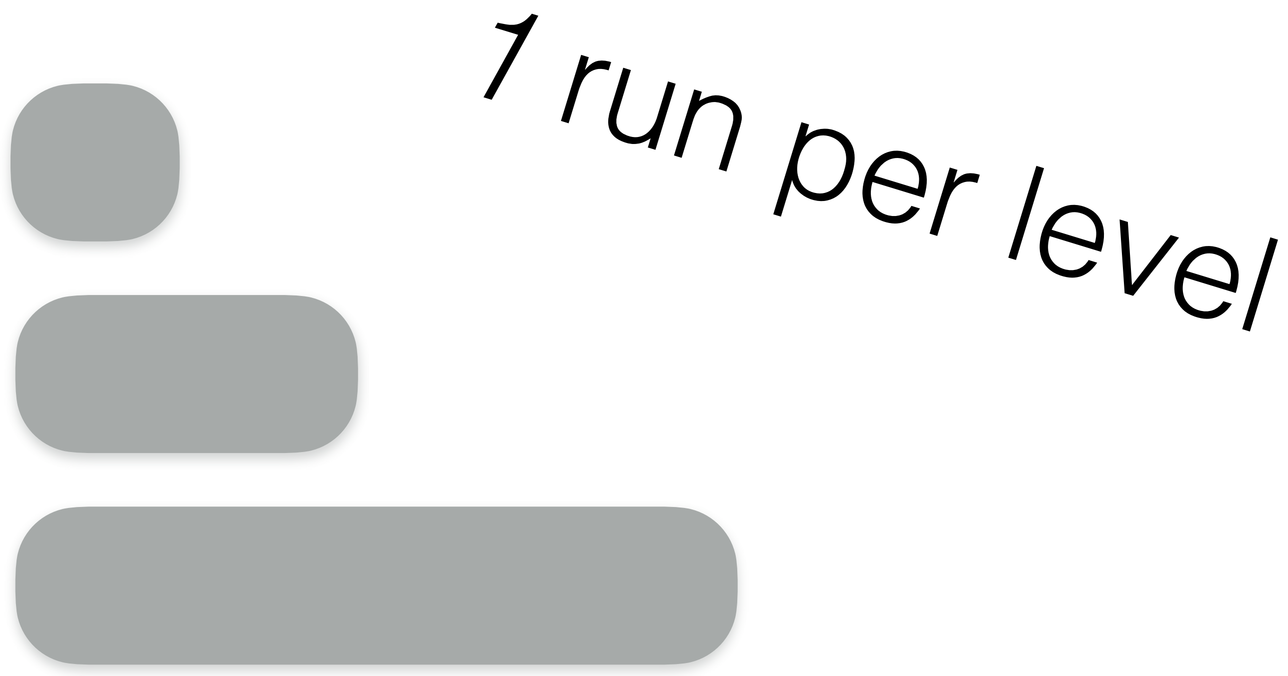
Tiering

write-optimized



Leveling

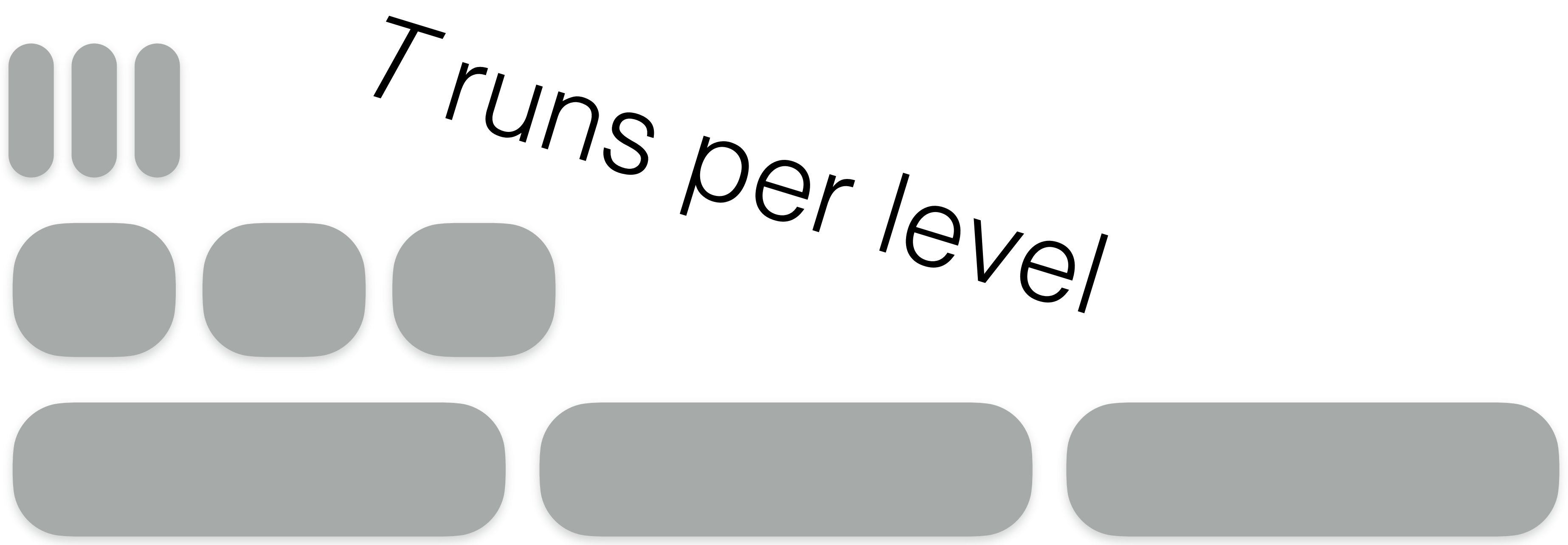
read-optimized



size ratio *R*

Tiering

write-optimized



Leveling

read-optimized

size ratio *R*

DASlab @ Harvard SEAS

Tiering

write-optimized

O(N) runs per level

log

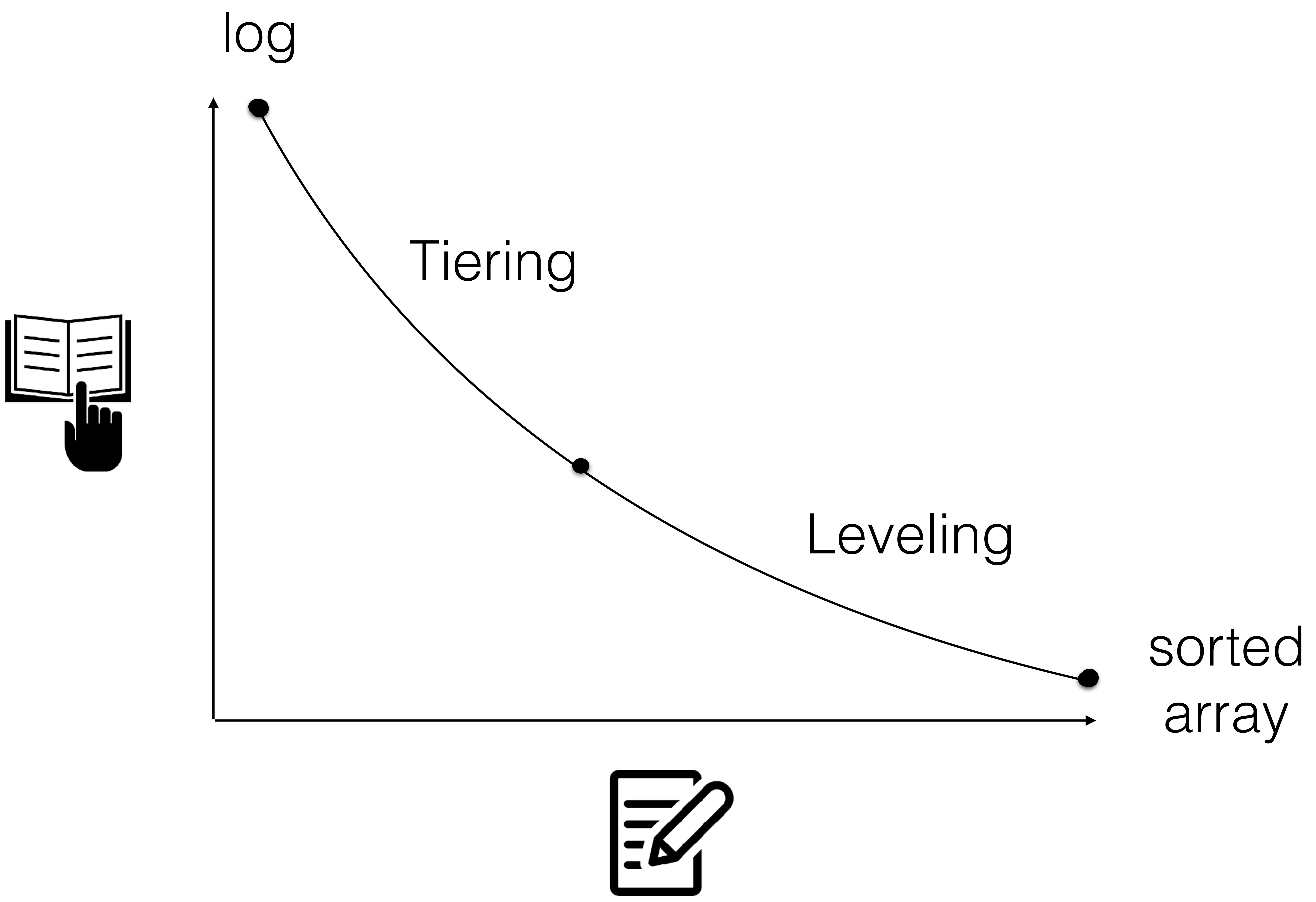
Leveling

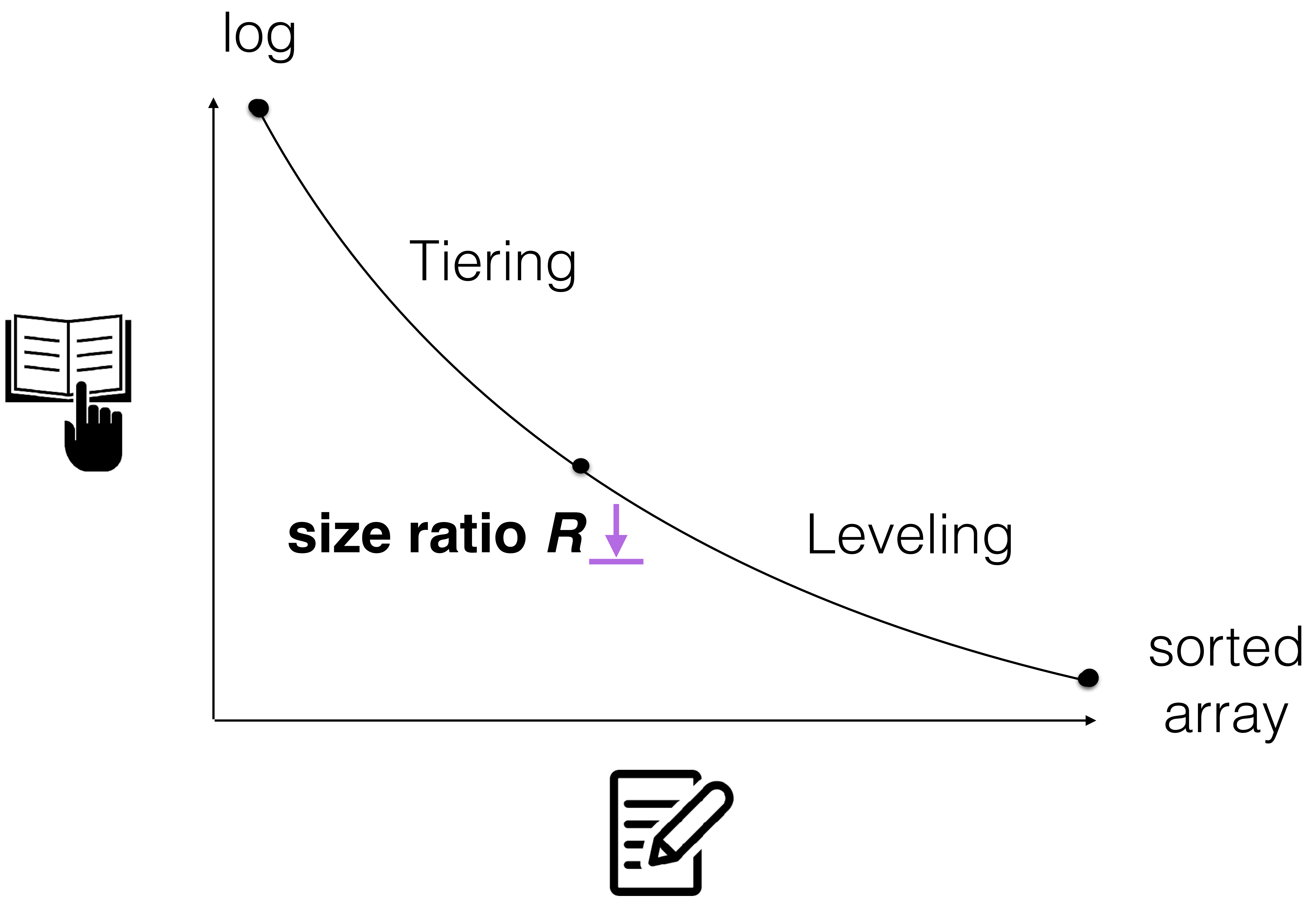
read-optimized

1 run per level

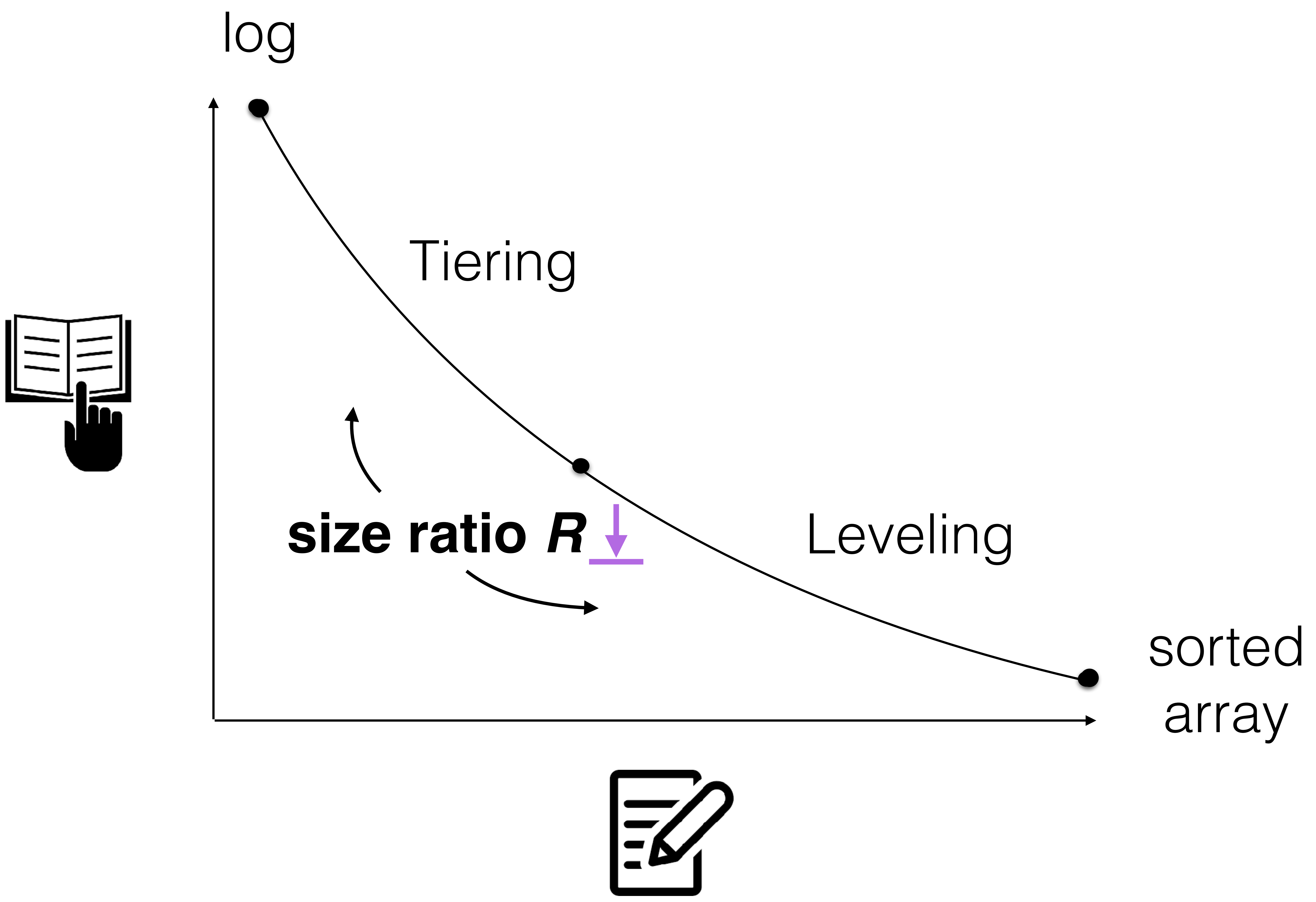
sorted array

size ratio *R*

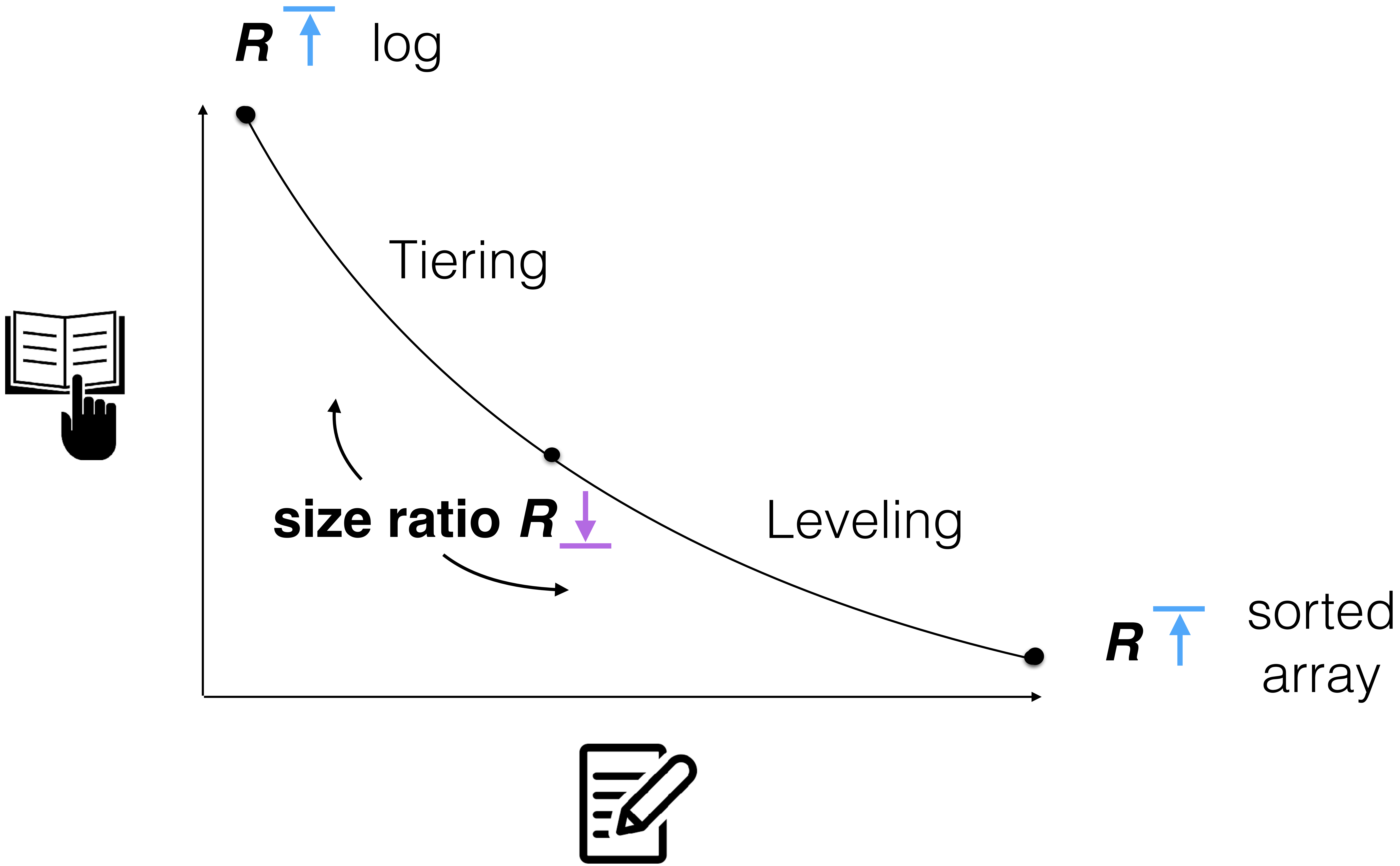

log
Tiering
Leveling
sorted
array



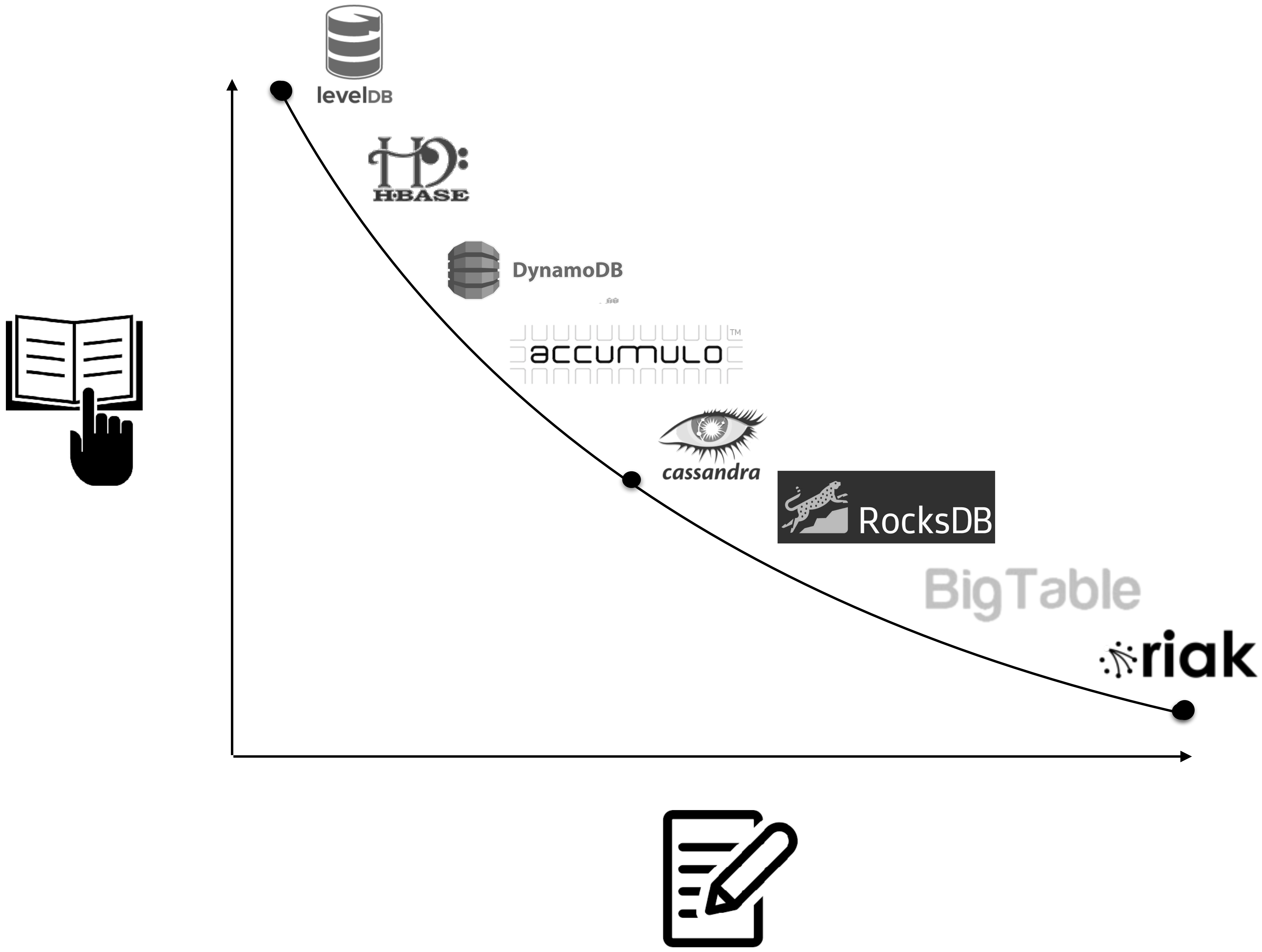
DASlab @ Harvard SEAS

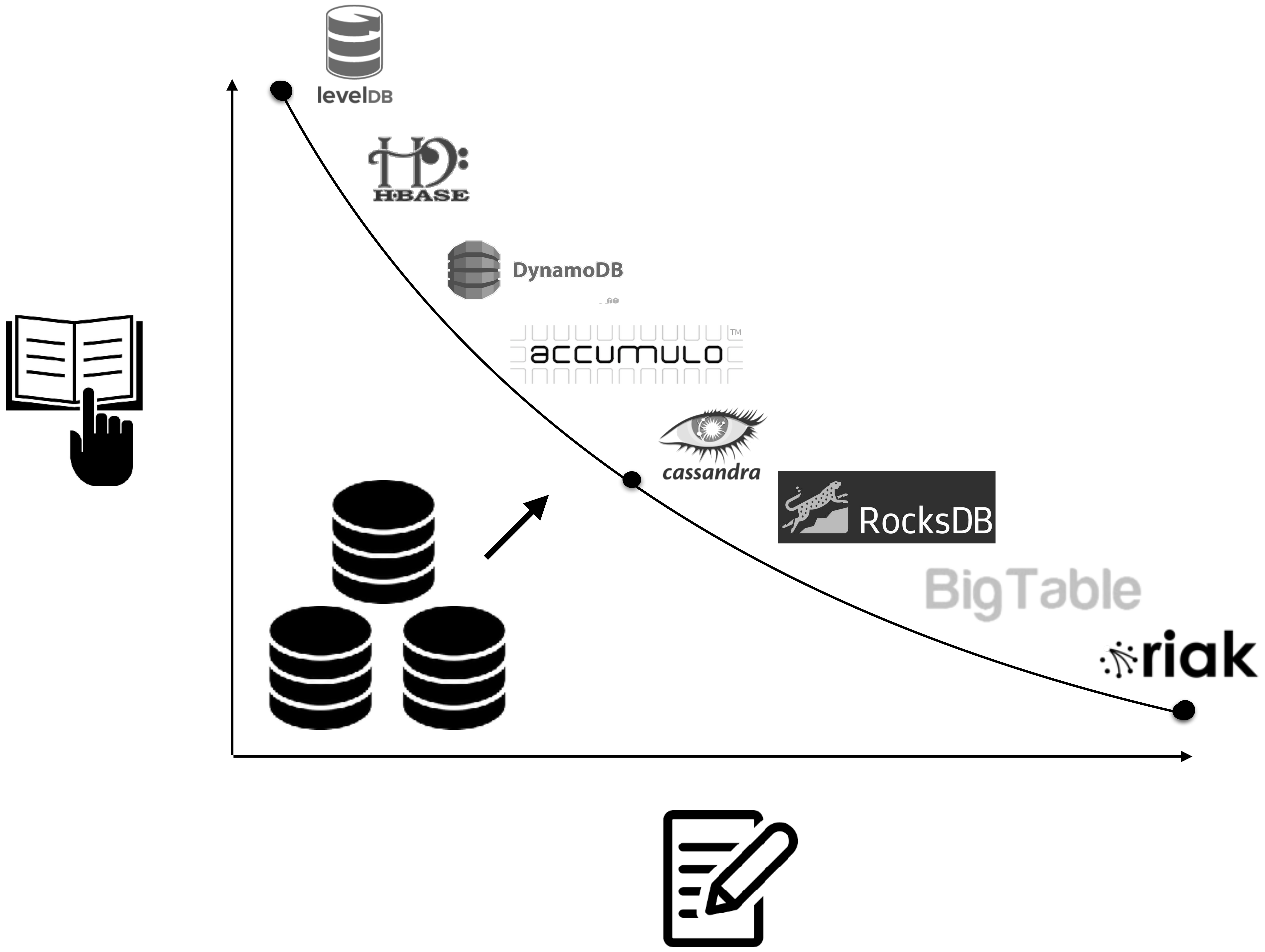


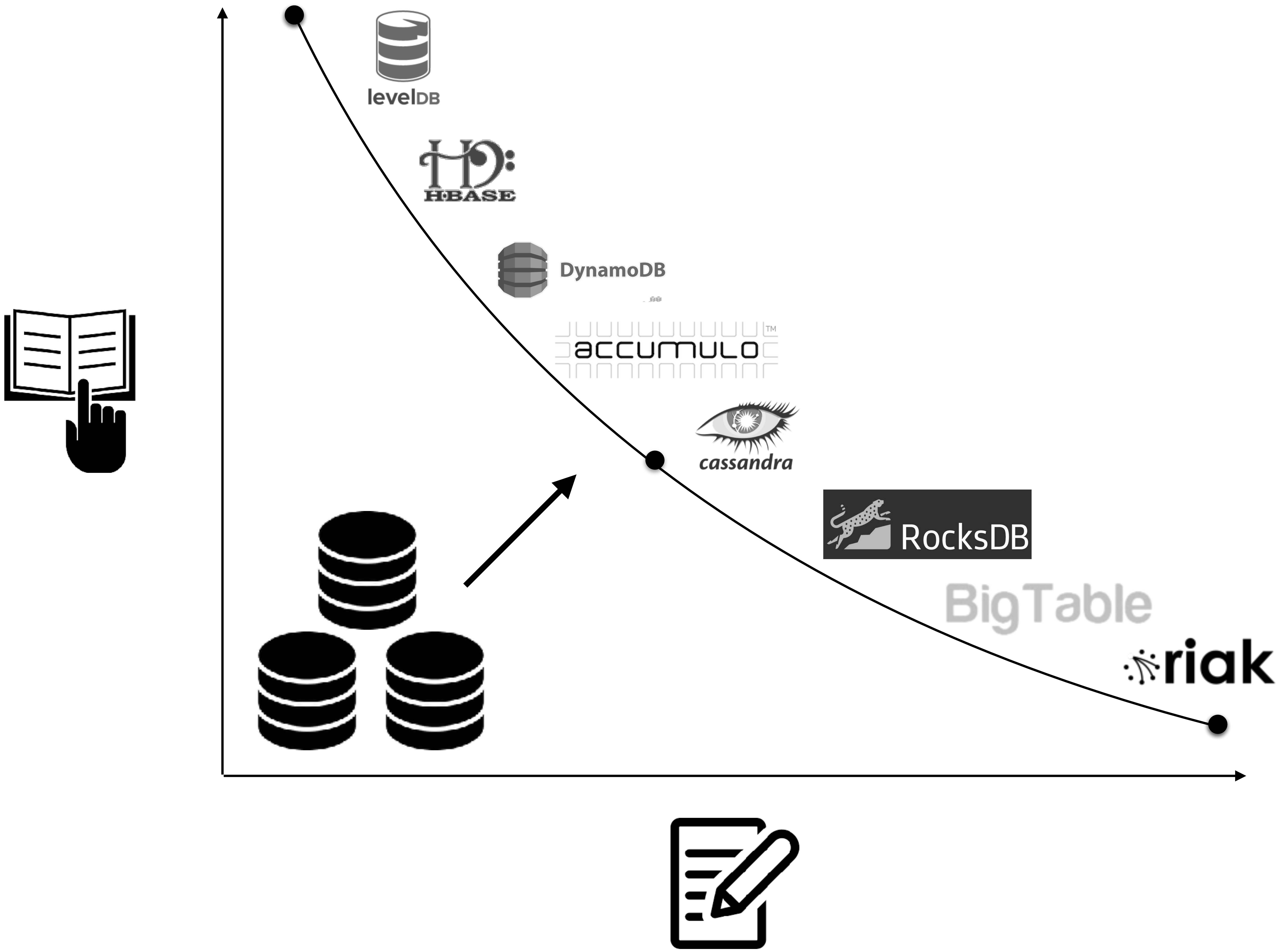
DASlab
@ Harvard SEAS

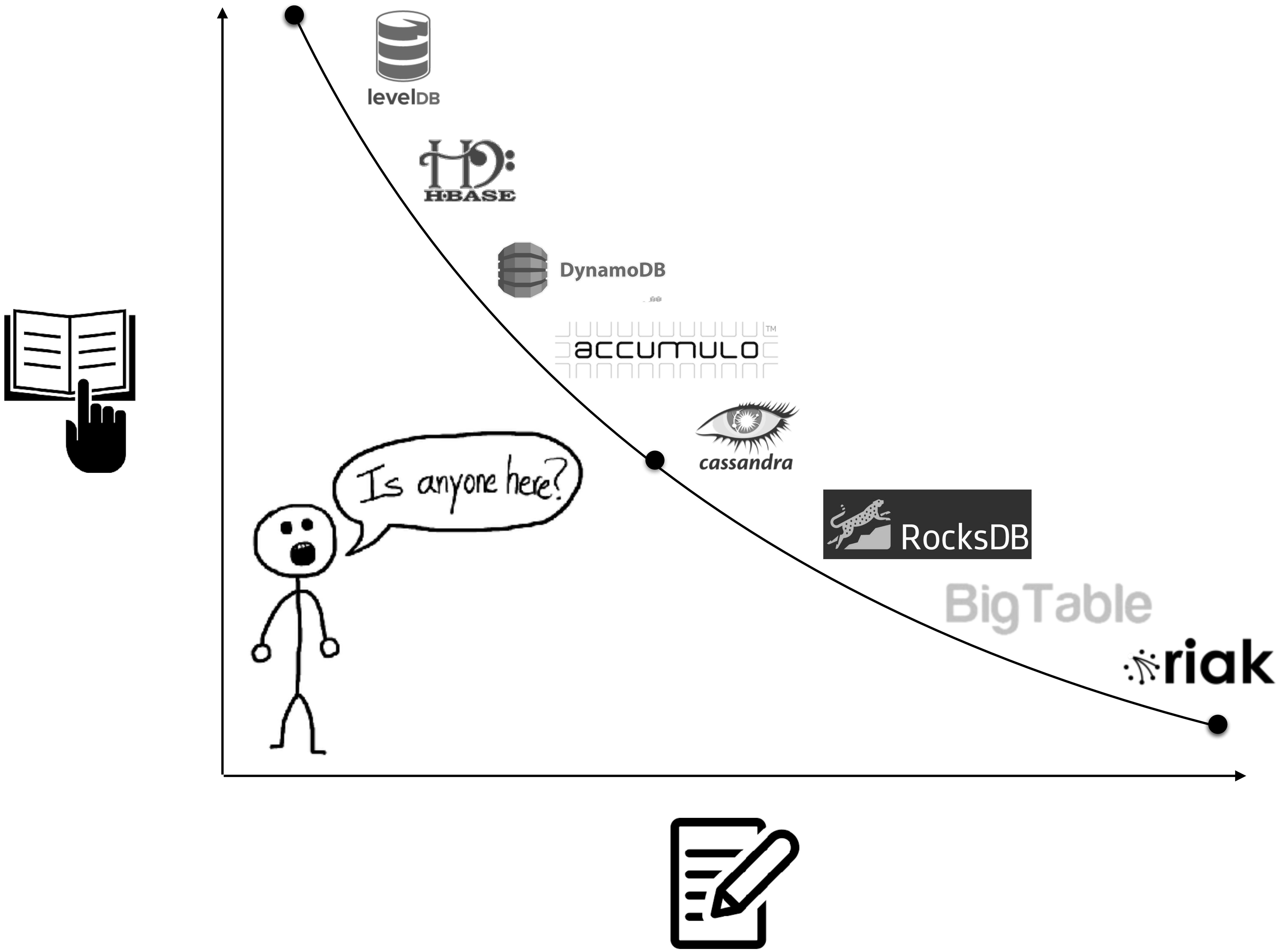


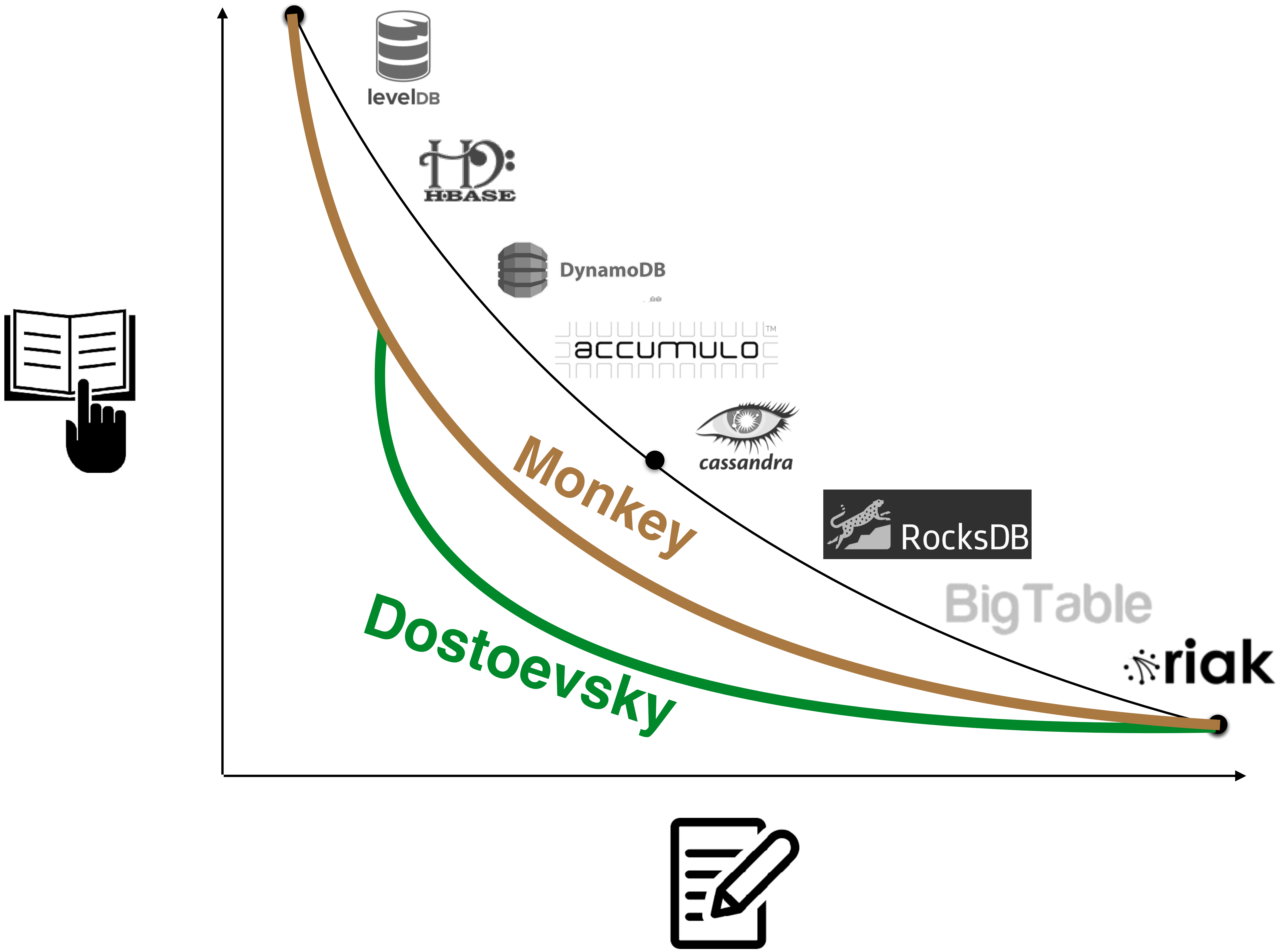
R log
Tiering
size ratio R
Leveling
R sorted array


levelDB
HBASE
DynamoDB
accumulo
cassandra
RocksDB
BigTable
riak


levelDB
HBASE
DynamoDB
accumulo
cassandra
RocksDB
BigTable
riak


levelDB
HBASE
DynamoDB
accumulo
cassandra
RocksDB
BigTable
riak


levelDB
HBASE
DynamoDB
accumulo
cassandra
RocksDB
BigTable
riak
Is anyone here?



DASlab
@ Harvard SEAS

Monkey: **O**ptimal **N**avigable **Key**-Value Store SIGMOD17

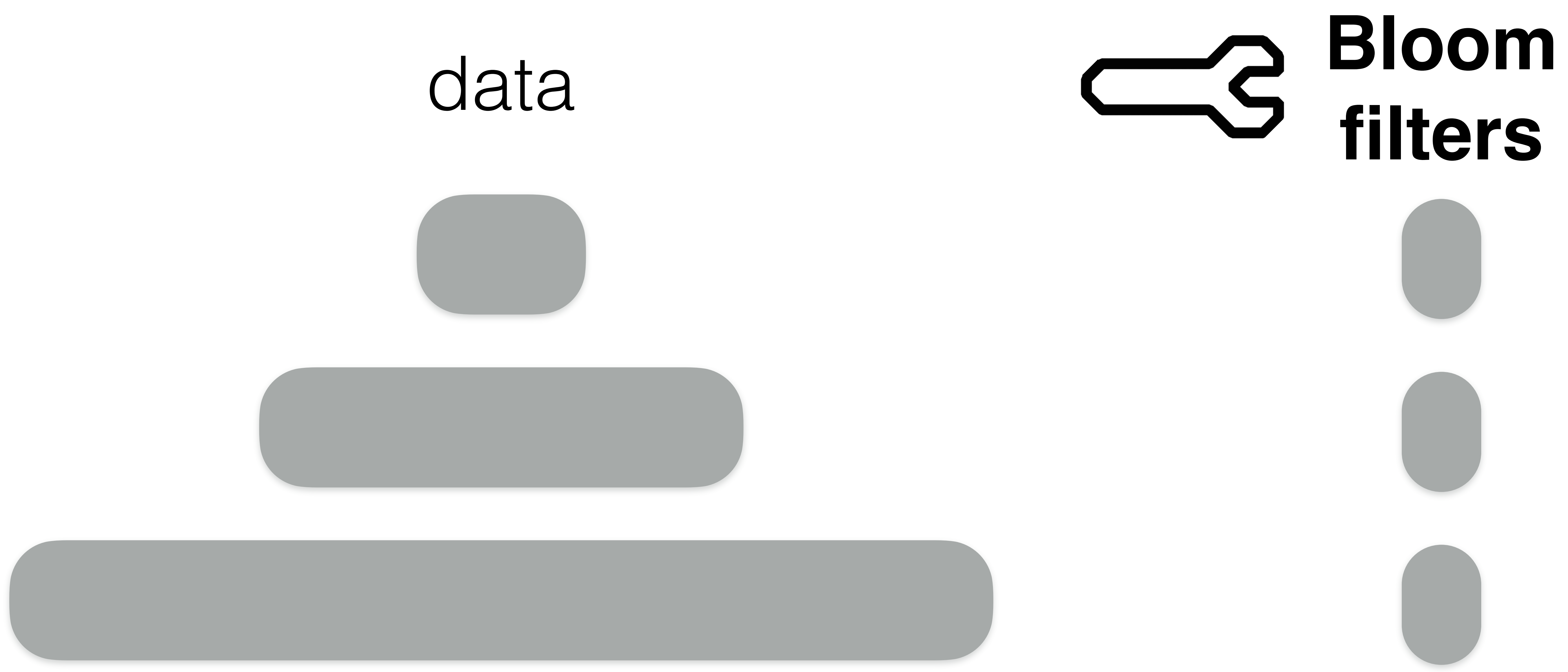
DASlab @ Harvard SEAS

Monkey: **O**ptimal **N**avigable **Key**-Value Store SIGMOD17

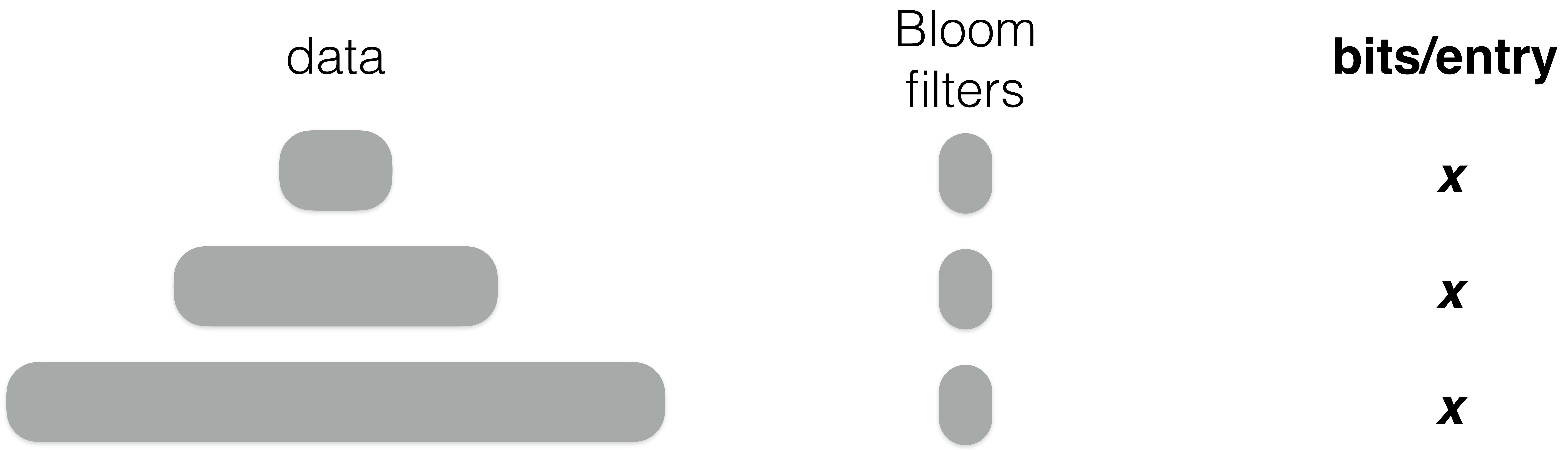
Niv Dayan
Manos Athanassoulis
Stratos Idreos

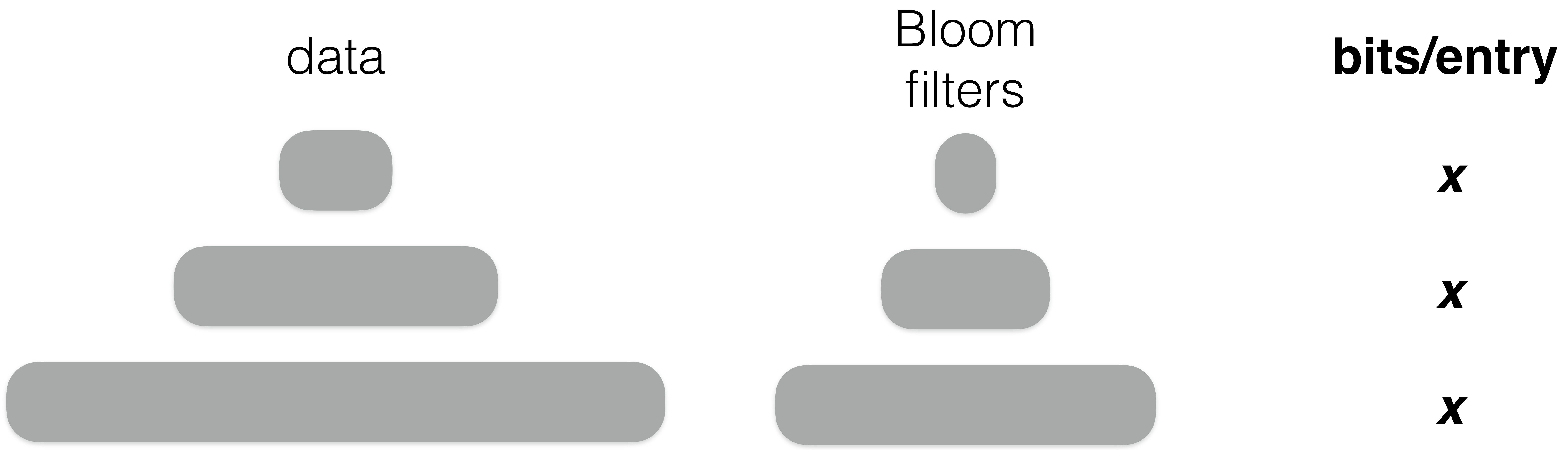
DASlab @ Harvard SEAS

Monkey: **O**ptimal **N**avigable **Key**-Value Store SIGMOD17



DASlab @ Harvard SEAS


data
Bloom filters
bits/entry
x
x
x


data
Bloom filters
bits/entry
x
x
x

data	Bloom filters	**false positive rate**
		$O(e^{-x})$
		$O(e^{-x})$
		$O(e^{-x})$

Bloom filters

false positive rate

$$\left.\begin{array}{c} O(e^{-x}) \\ O(e^{-x}) \\ O(e^{-x}) \end{array}\right\} = O(\boldsymbol{e^{-x}} \cdot \mathbf{log}_{\boldsymbol{R}}(\boldsymbol{N}))$$

Bloom filters

false positive rate

$$\left.\begin{array}{c} O(e^{-x}) \\ \\ O(e^{-x}) \\ \\ O(e^{-x}) \end{array}\right\} = O(\boldsymbol{e^{-x}} \cdot \mathbf{log}_{\boldsymbol{R}}(\boldsymbol{N}))$$

Bloom filters	false positive rate
	$O(e^{-x})$
	$O(e^{-x})$
most memory	$O(e^{-x})$

DASlab @ Harvard SEAS

Bloom
filters

most memory

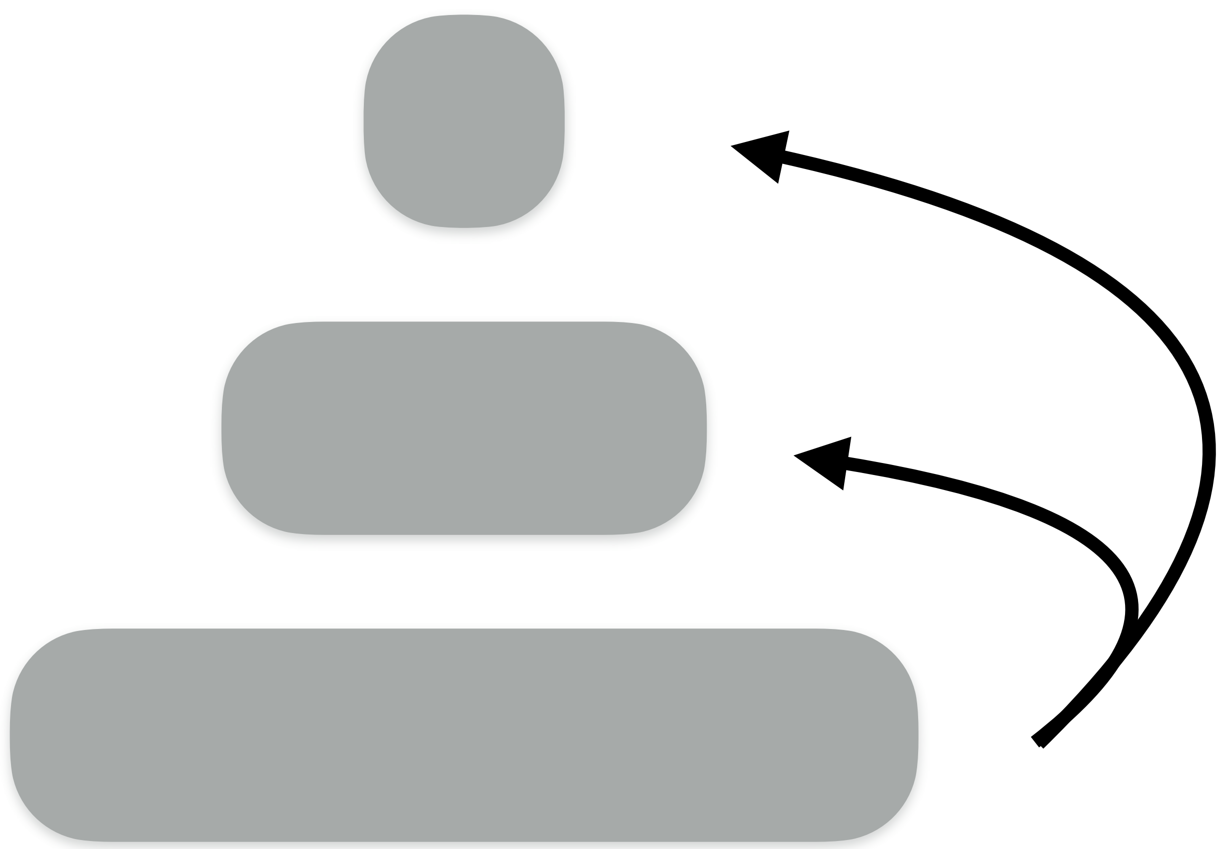
saves at most 1 I/O!

false
positive rate

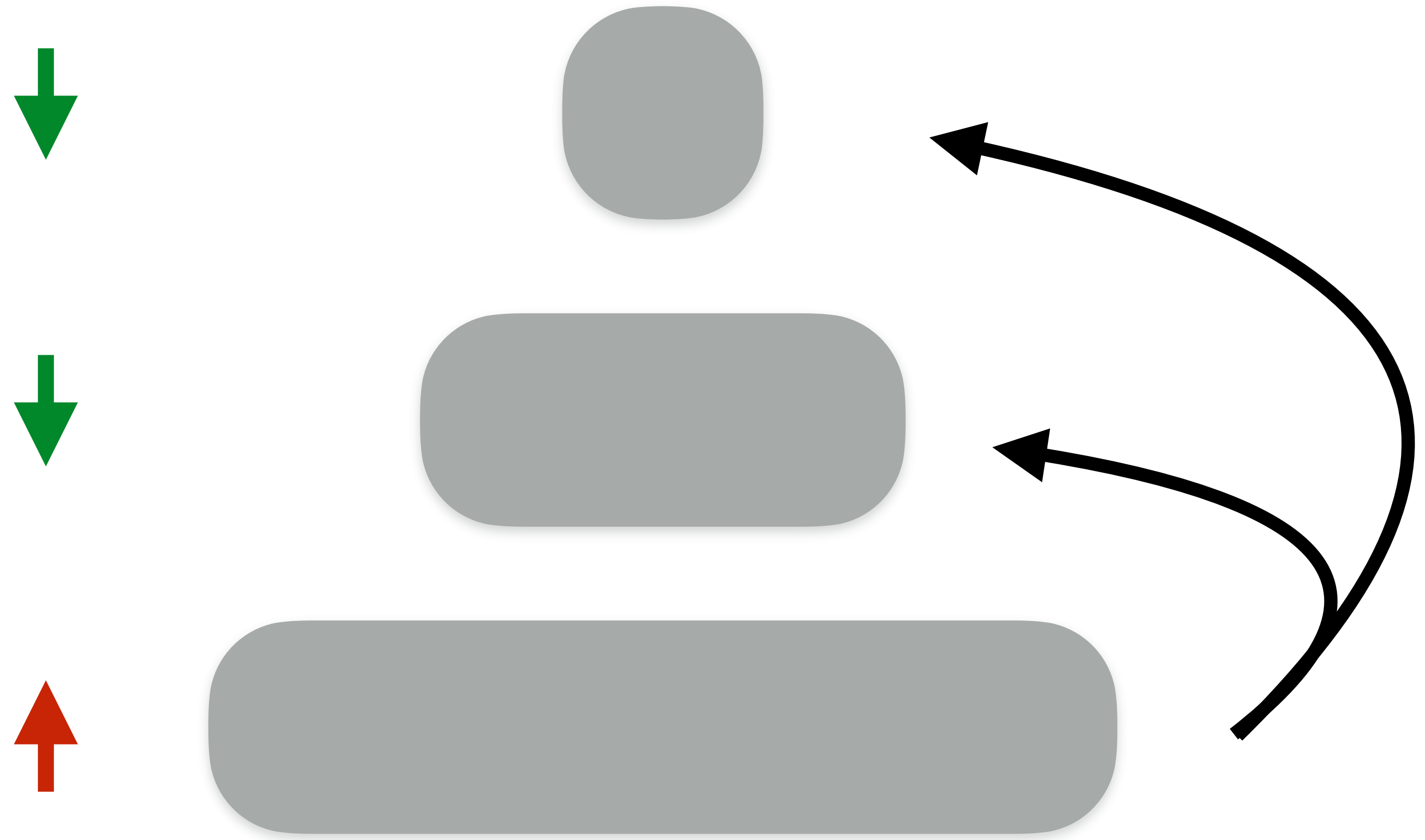
$O(e^{-x})$

$O(e^{-x})$

$O(e^{-x})$

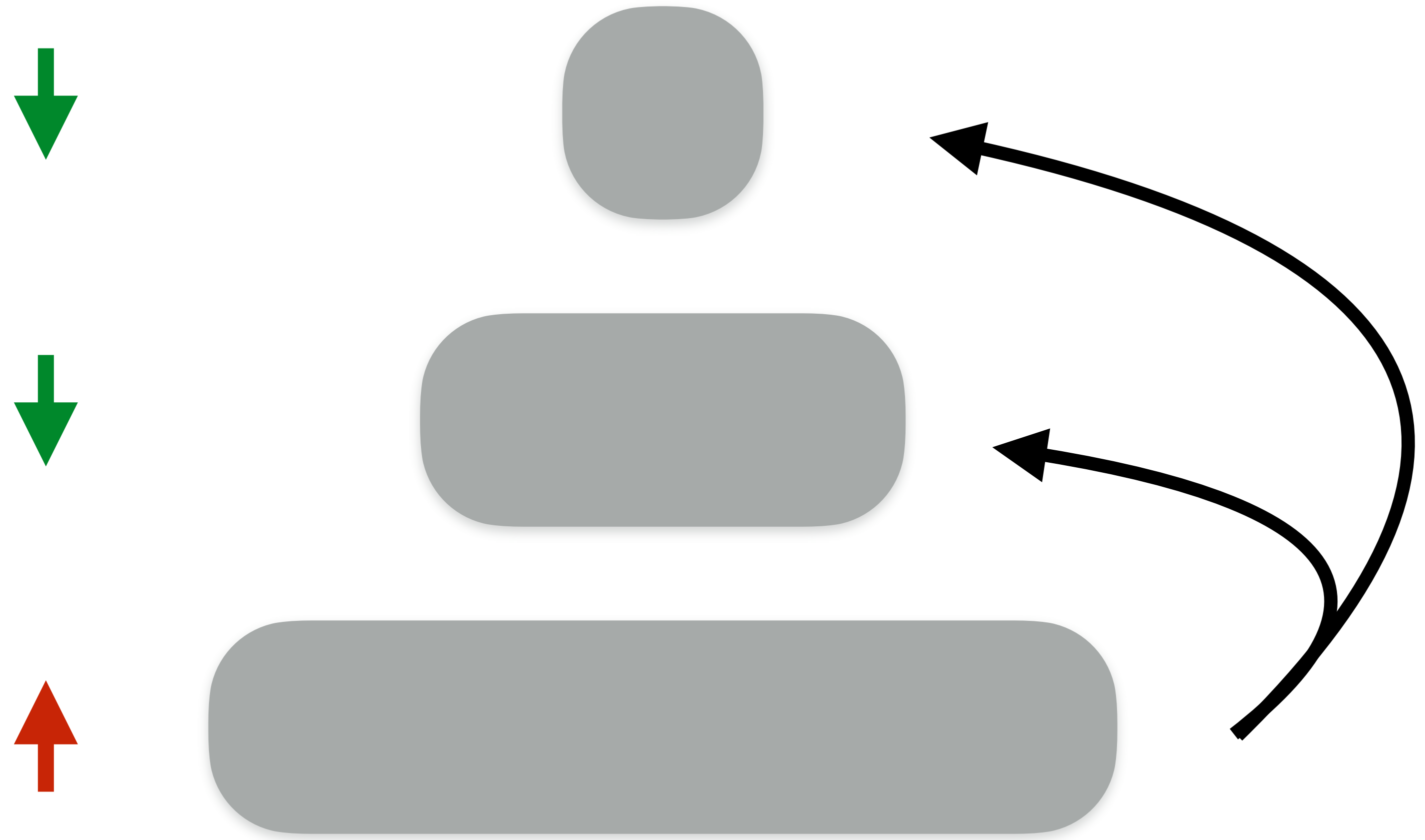


reallocate



reallocate

same memory - fewer false positives



reallocate

relax

false positive rates

$0 < p_0 < 1$

$0 < p_1 < 1$

$0 < p_2 < 1$

relax

model

false positive rates

read cost = f($\boldsymbol{p_0}$, $\boldsymbol{p_1}$ …)

$0 < p_0 < 1$

$0 < p_1 < 1$

$0 < p_2 < 1$

memory footprint = f($\boldsymbol{p_0}$, $\boldsymbol{p_1}$ …)

@ Harvard SEAS

relax

false positive rates

$0 < p_0 < 1$

$0 < p_1 < 1$

$0 < p_2 < 1$

model

read cost $= \sum_{1}^{L} p_i$

memory footprint $= -\sum_{i}^{L} \frac{N}{T^{L-i}} \cdot \frac{\ln(p_i)}{\ln(2)^2}$

DASlab @ Harvard SEAS

relax

false positive rates

$0 < p_0 < 1$

$0 < p_1 < 1$

$0 < p_2 < 1$

model

read cost $= \sum_{1}^{L} p_i$

memory footprint $= -\sum_{i}^{L} \frac{N}{T^{L-i}} \cdot \frac{\ln(p_i)}{\ln(2)^2}$

optimize

in terms of p_0, p_1 ...

DASlab @ Harvard SEAS

false
positive rate

$$p_0 \approx O(e^{-x}/\boldsymbol{R^2})$$

$$p_1 \approx O(e^{-x}/\boldsymbol{R^1})$$

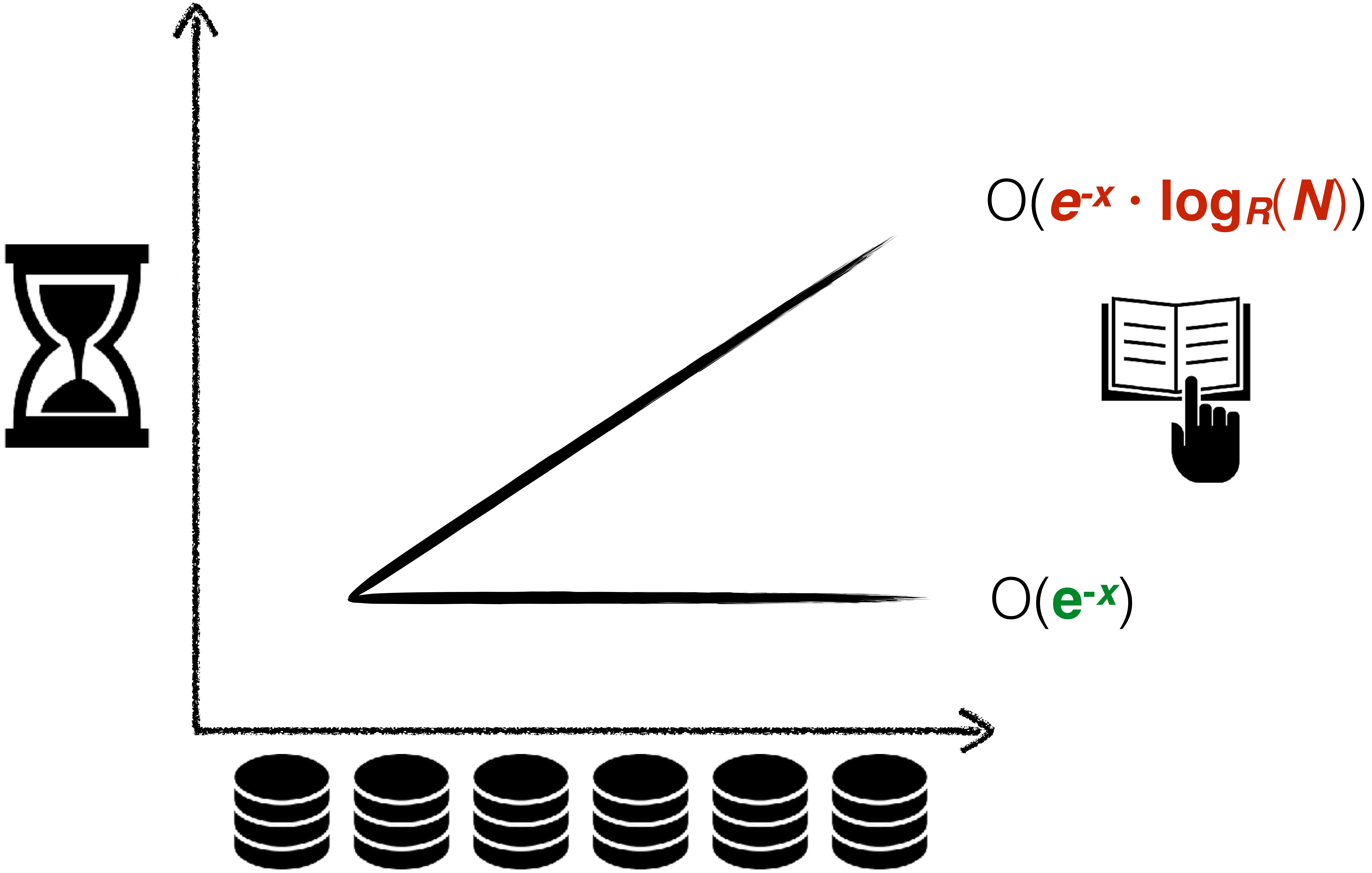
$$p_2 \approx O(e^{-x}/\boldsymbol{R^0})$$

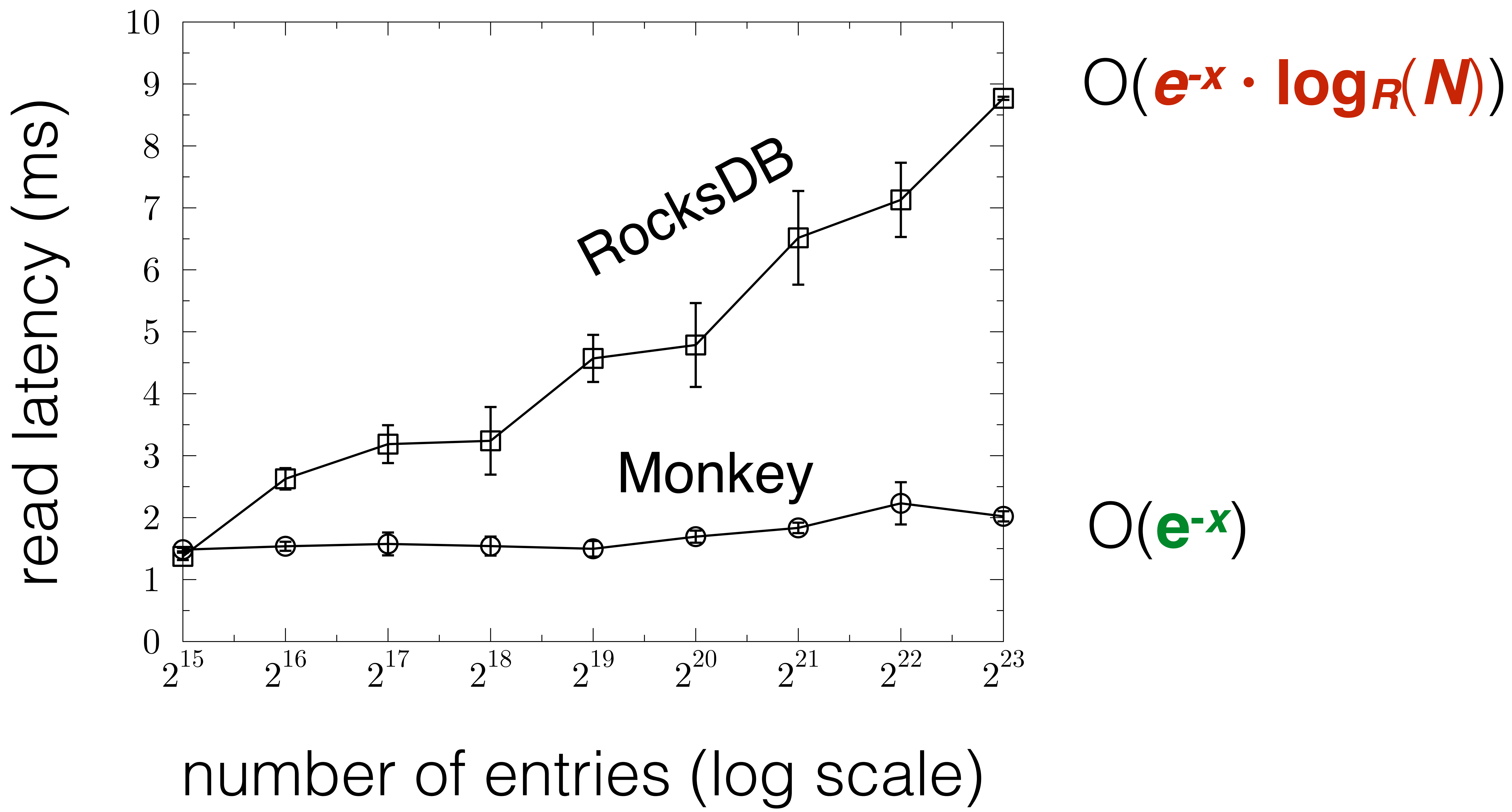
false
positive rate

geometric progression

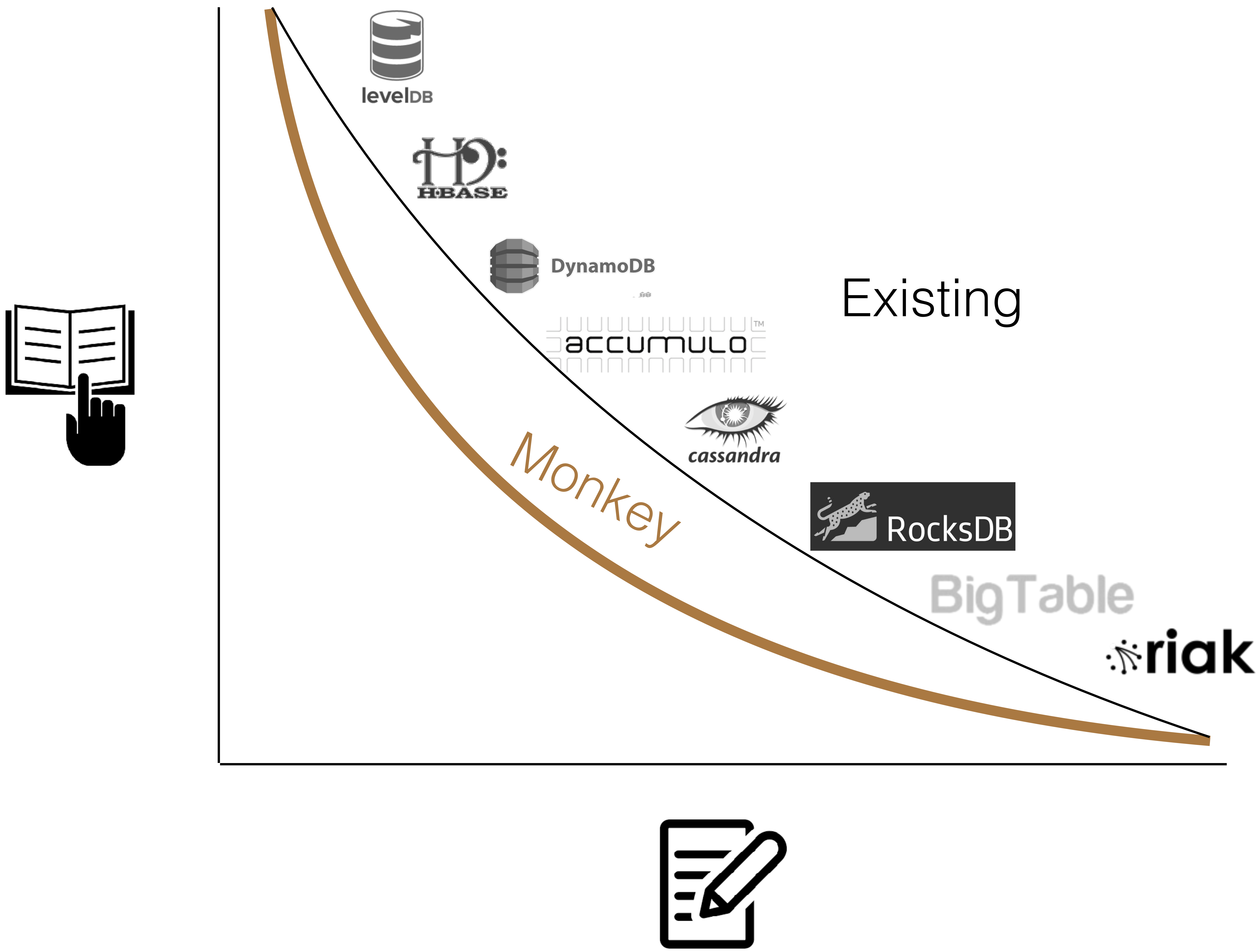
$$\left.\begin{array}{c} O(e^{-x}/R^{2}) \\[1em] O(e^{-x}/R^{1}) \\[1em] O(e^{-x}/R^{0}) \end{array}\right\} = \quad O(e^{-x})$$

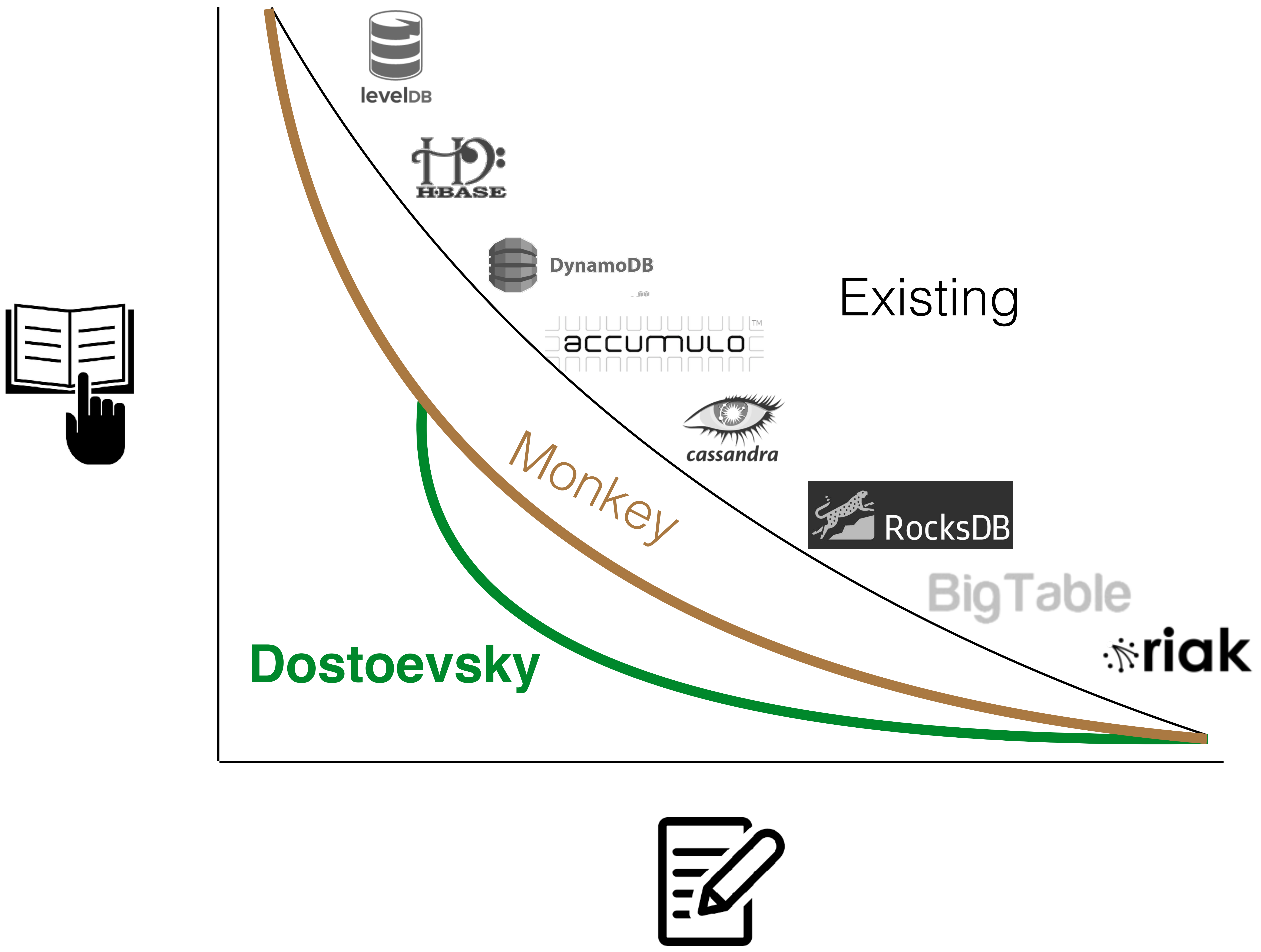
$$O(e^{-x} \cdot \log_R(N)) \quad > \quad O(e^{-x})$$


$O(e^{-x} \cdot \log_R(N))$
$O(e^{-x})$

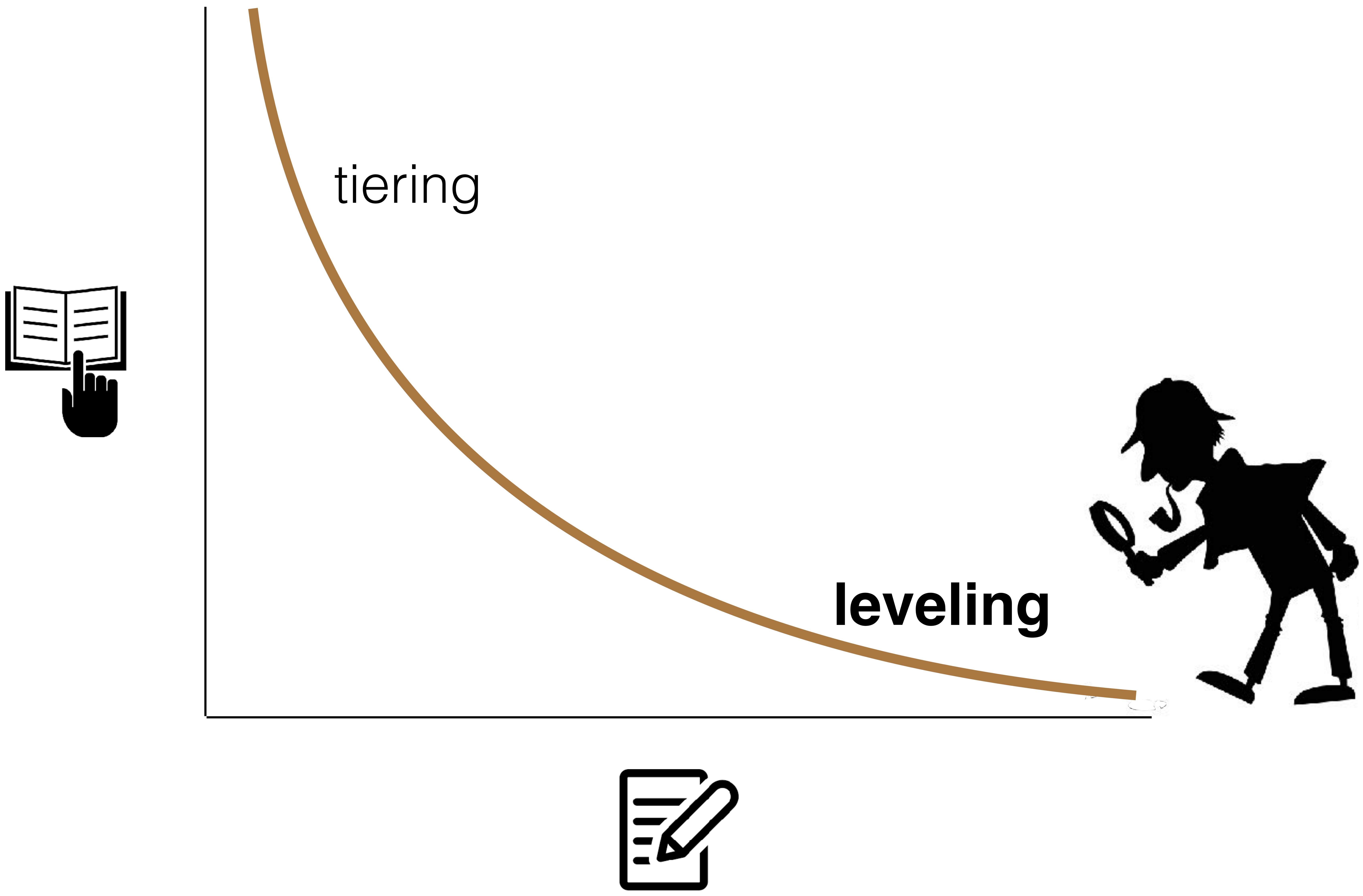


read latency (ms)
10
9
8
7
6
5
4
3
2
1
0
RocksDB
Monkey
2^{15} 2^{16} 2^{17} 2^{18} 2^{19} 2^{20} 2^{21} 2^{22} 2^{23}
number of entries (log scale)
$O(e^{-x} \cdot \log_R(N))$
$O(e^{-x})$


levelDB
HBASE
DynamoDB
accumulo
cassandra
RocksDB
BigTable
riak
Existing
Monkey



DASlab @ Harvard SEAS


tiering
leveling

I/O overheads with leveling

point | long range | short range | writes

point

false positive rates

exponentially decreasing

$O(e^{-x}/R^2)$

$O(e^{-x}/R)$

$O(e^{-x})$

false positive rates

$O(e^{-x}/R^2)$

$O(e^{-x}/R)$

point $\longrightarrow$ $O(e^{-x})$

point

long range

short range

writes

largest level

$O(e^{-x})$

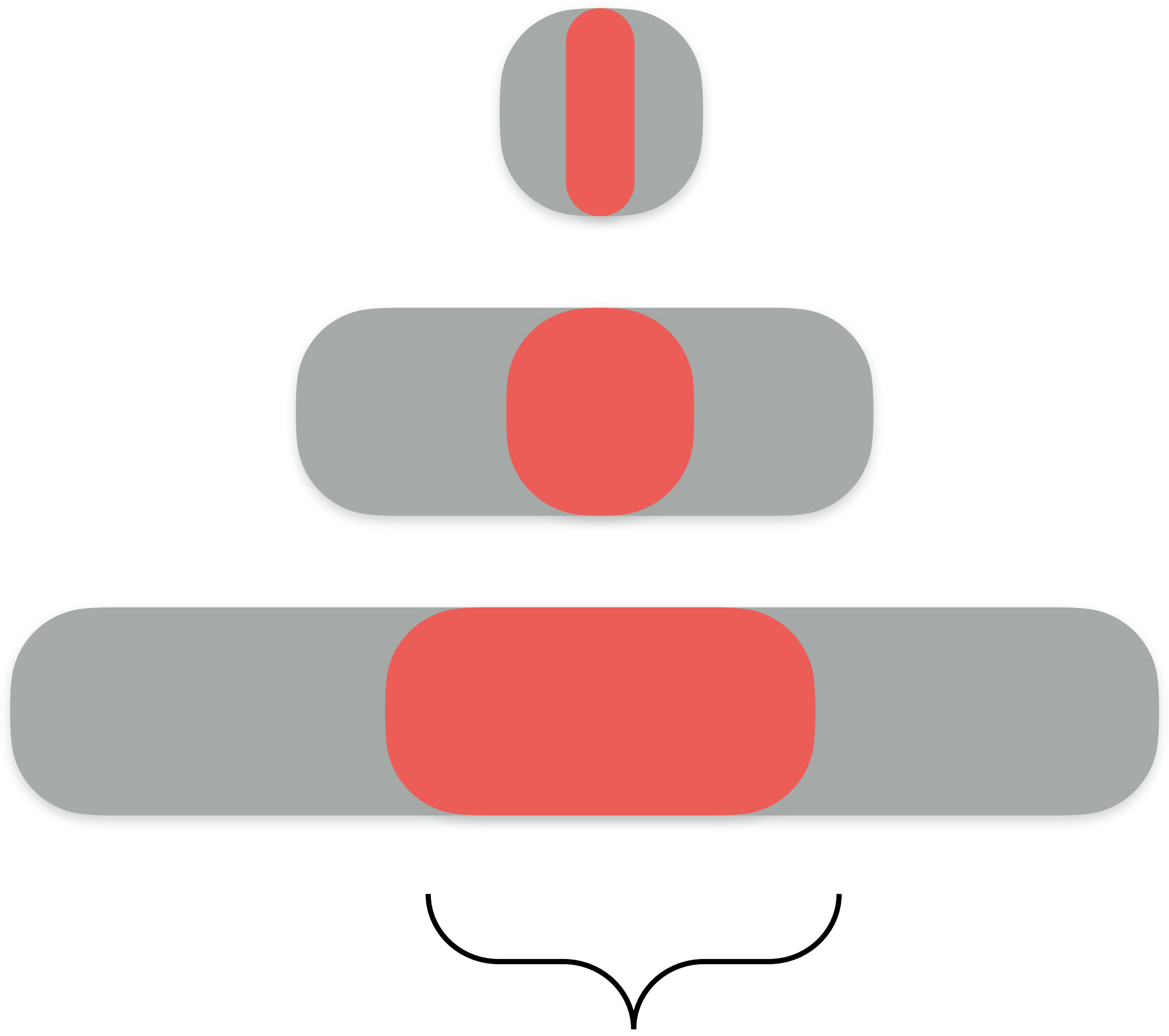
long range

target range

$O(s/R^2)$

$O(s/R)$

$O(s)$



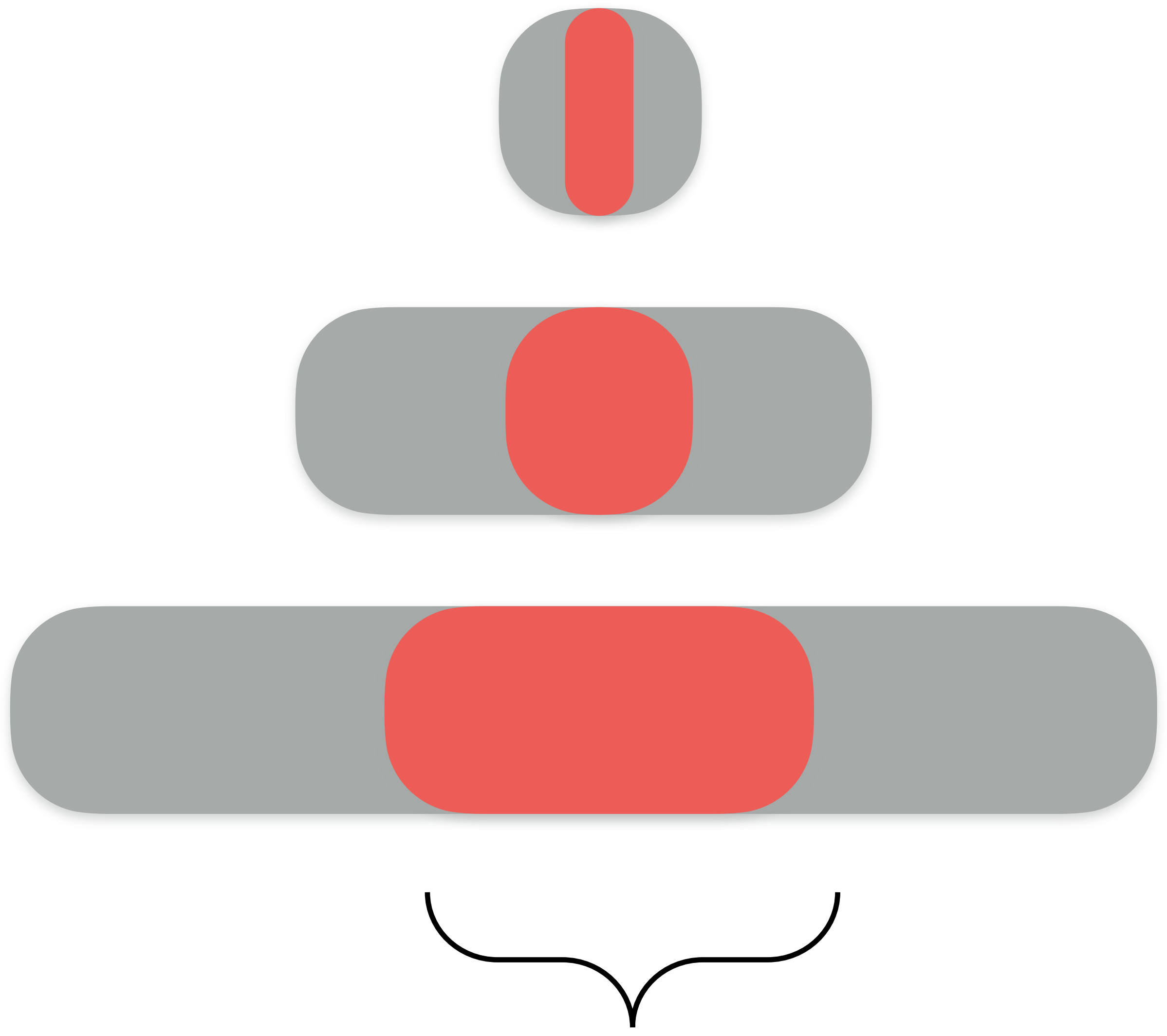
target key range

target range

$O(s/R^2)$

$O(s/R)$

long range $O(s)$ largest level



target key range

point

largest level

$O(e^{-x})$

long range

largest level

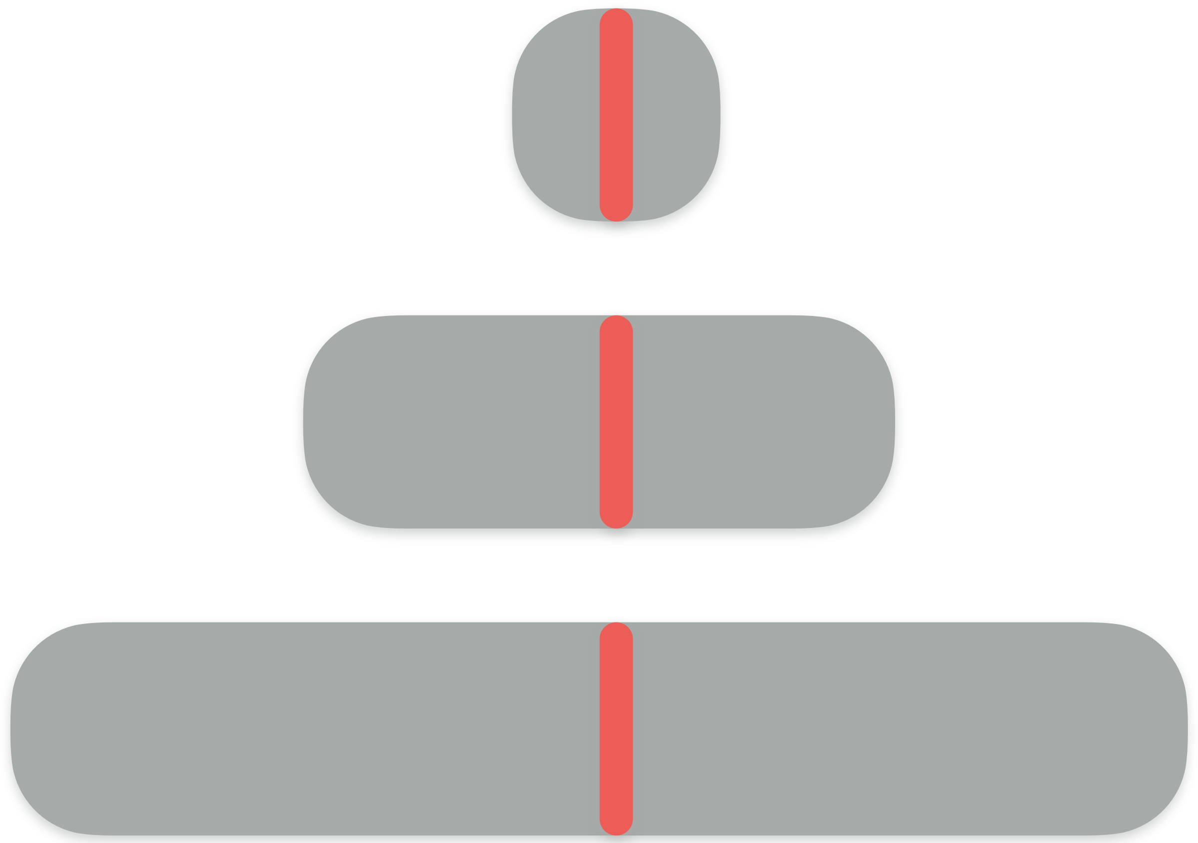
$O(s)$

short range

writes

short range

target range



1

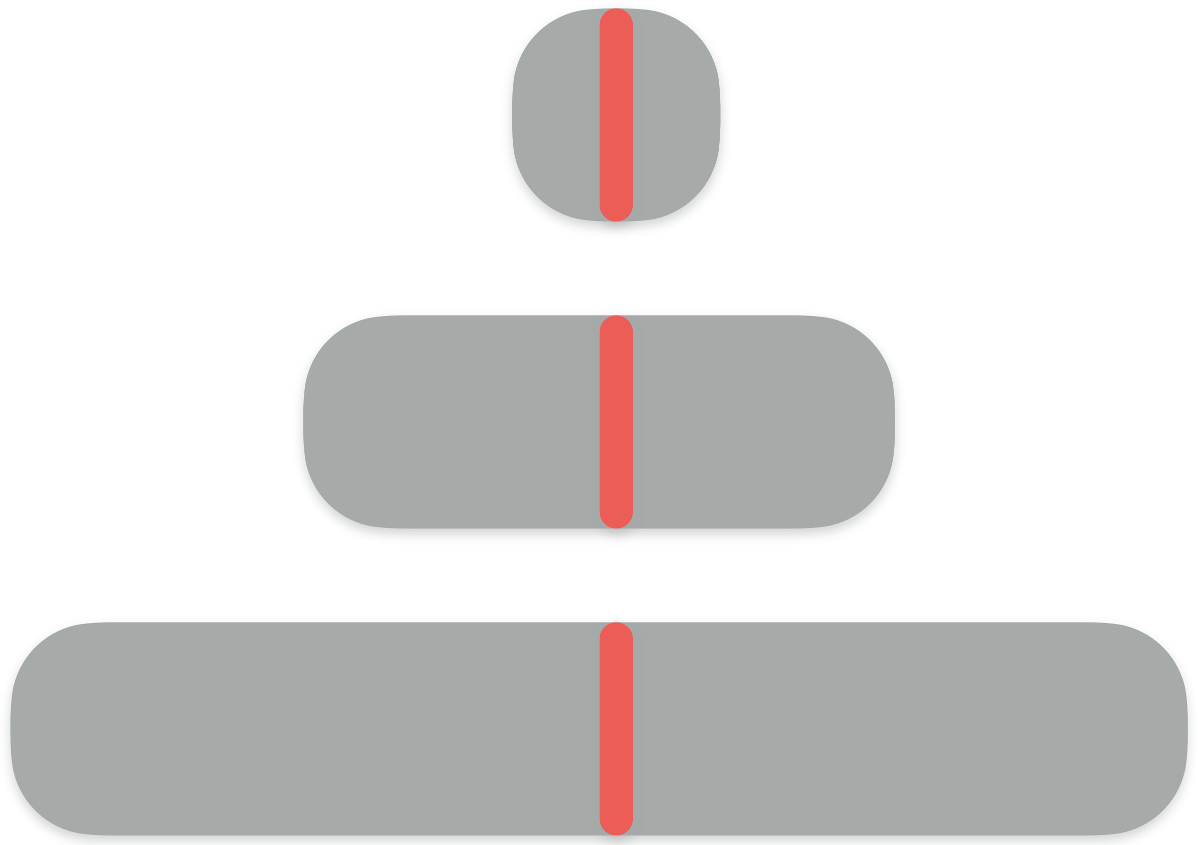
1

1

@ Harvard SEAS

short range

target range



1

1

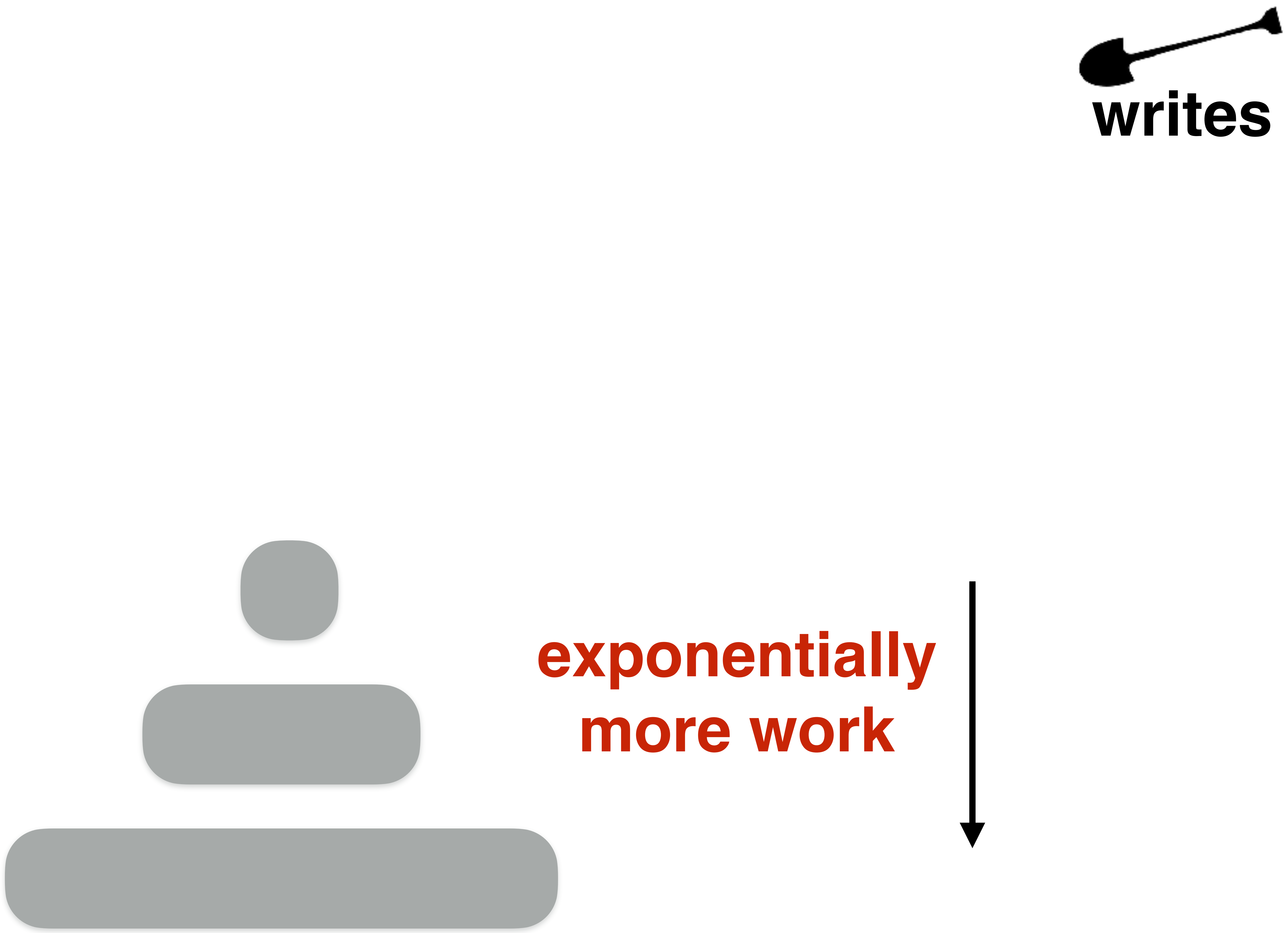
1

all levels

O(**$\log_R(N)$**)

point	long range	short range	**writes**
largest level	largest level	all levels	
$O(e^{-x})$	$O(s)$	$O(\log_R(N))$	

@ Harvard SEAS

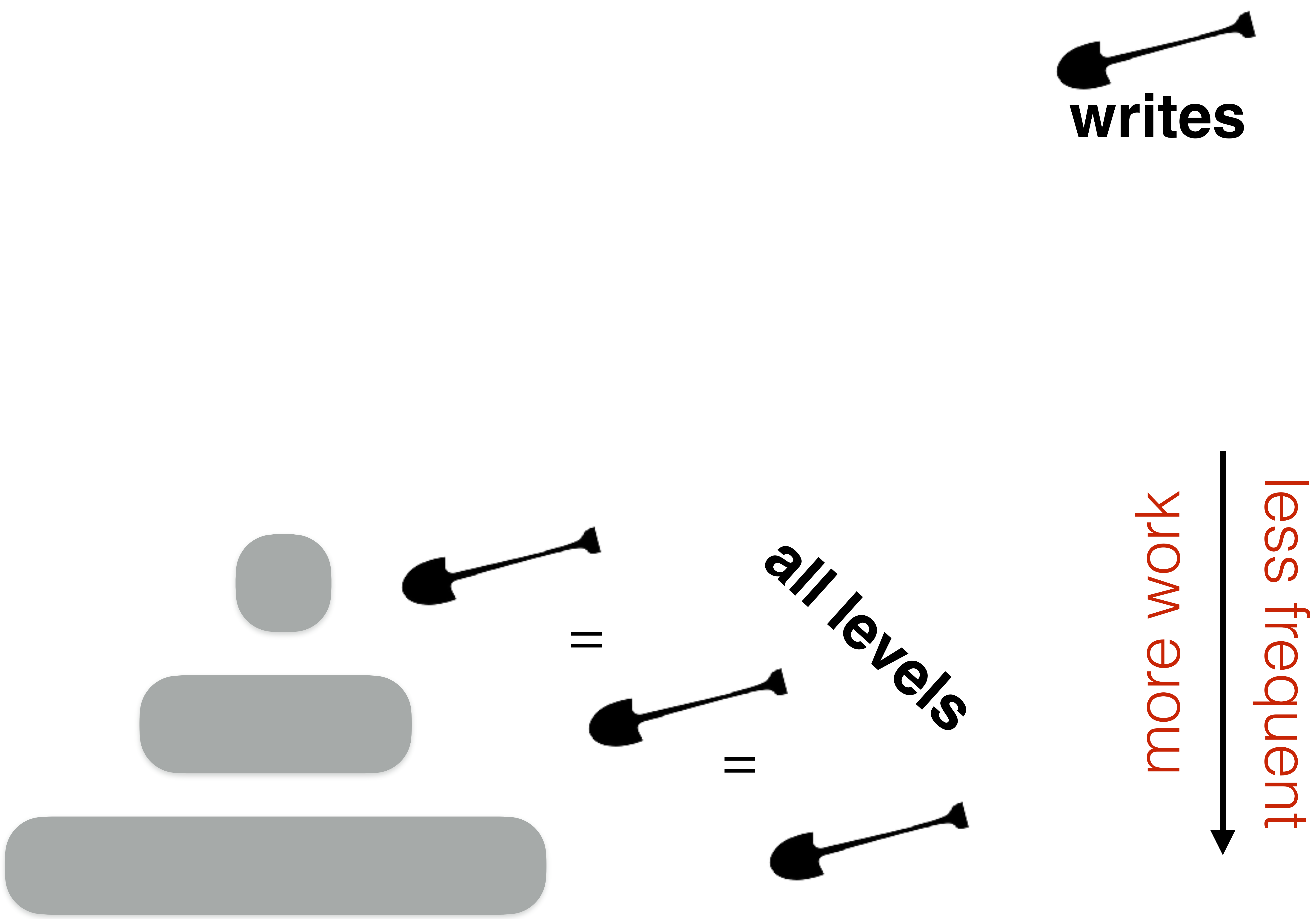

writes
exponentially more work

writes

exponentially more work

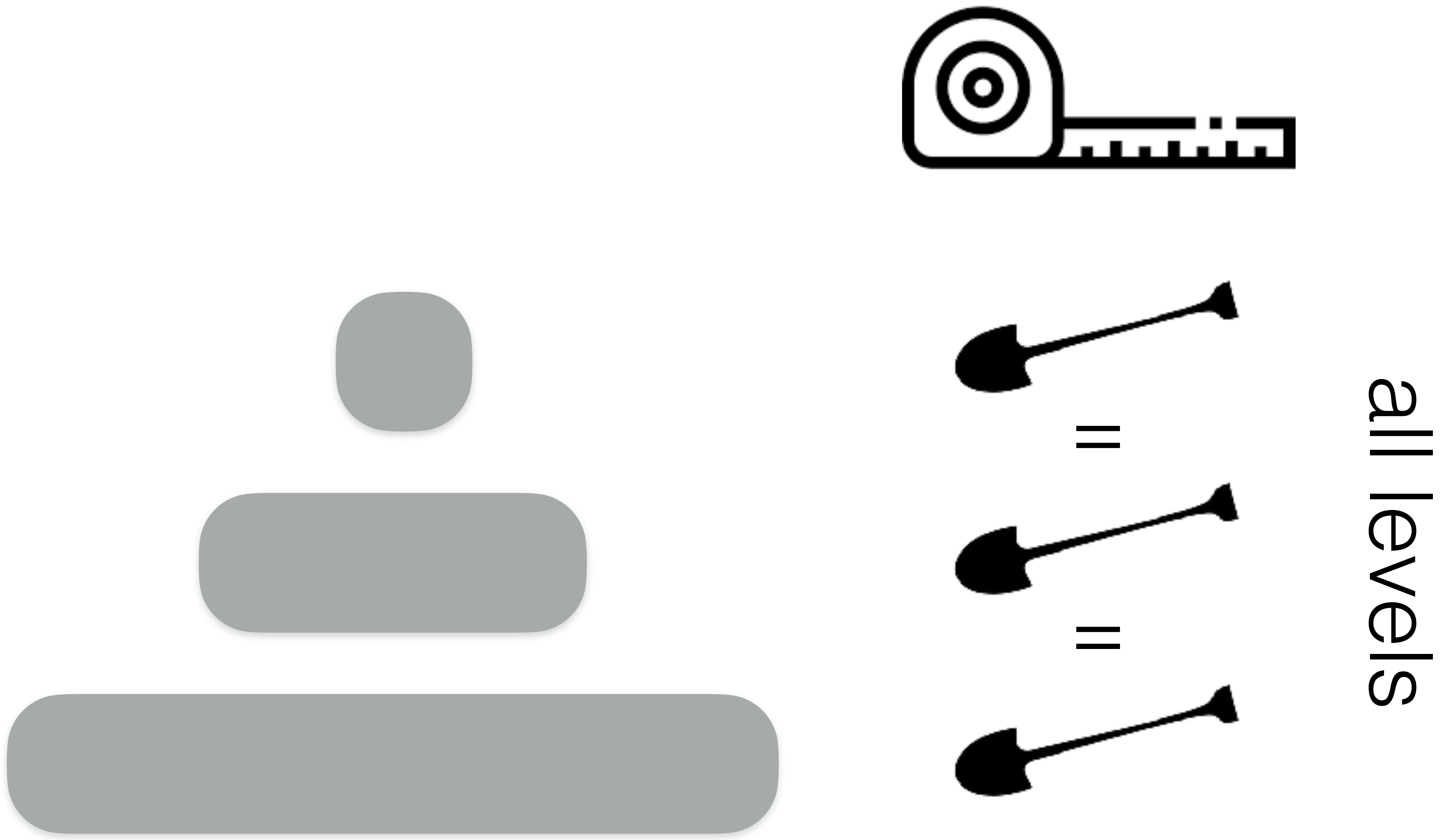
exponentially less frequent

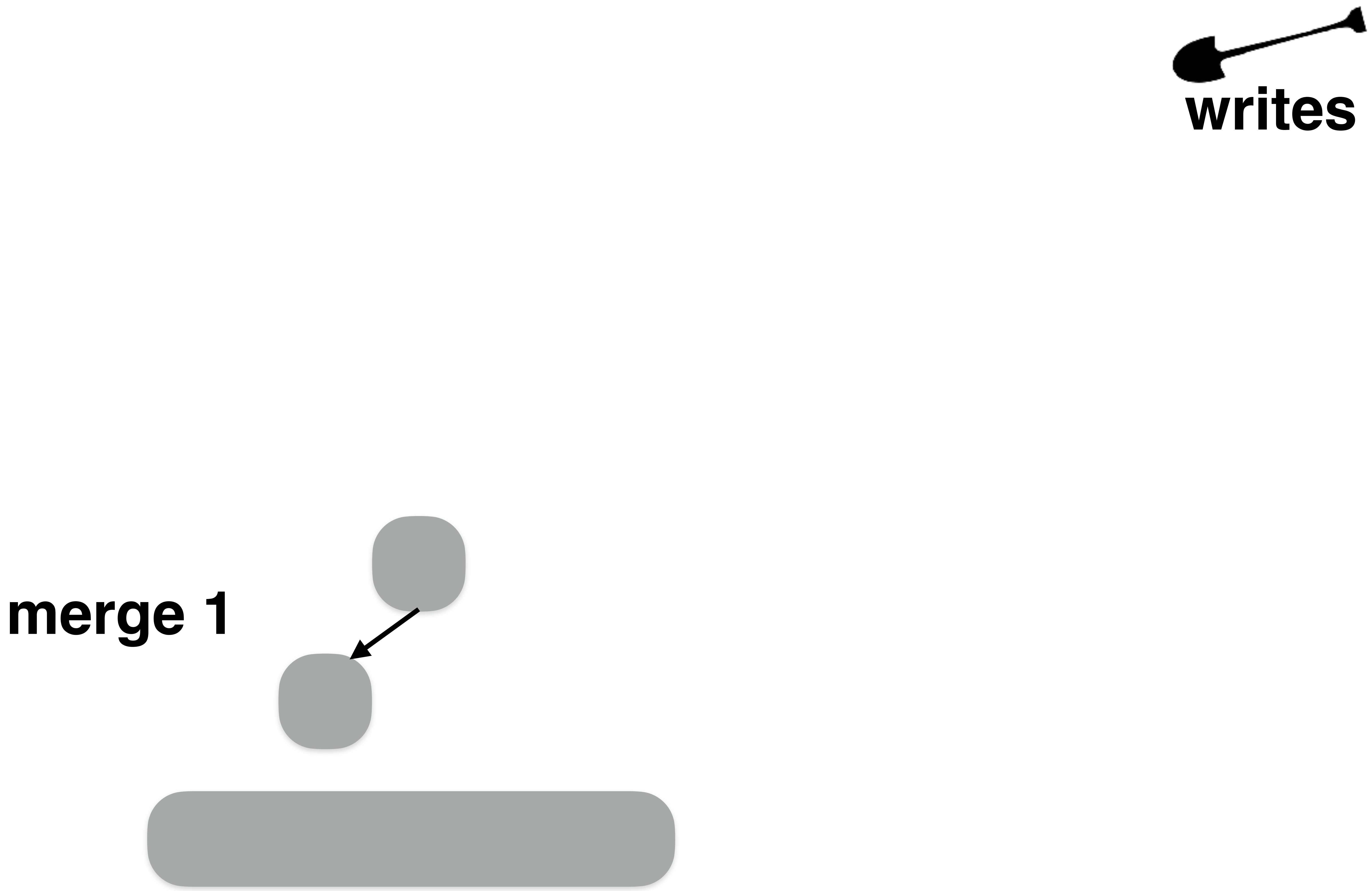
DASlab @ Harvard SEAS



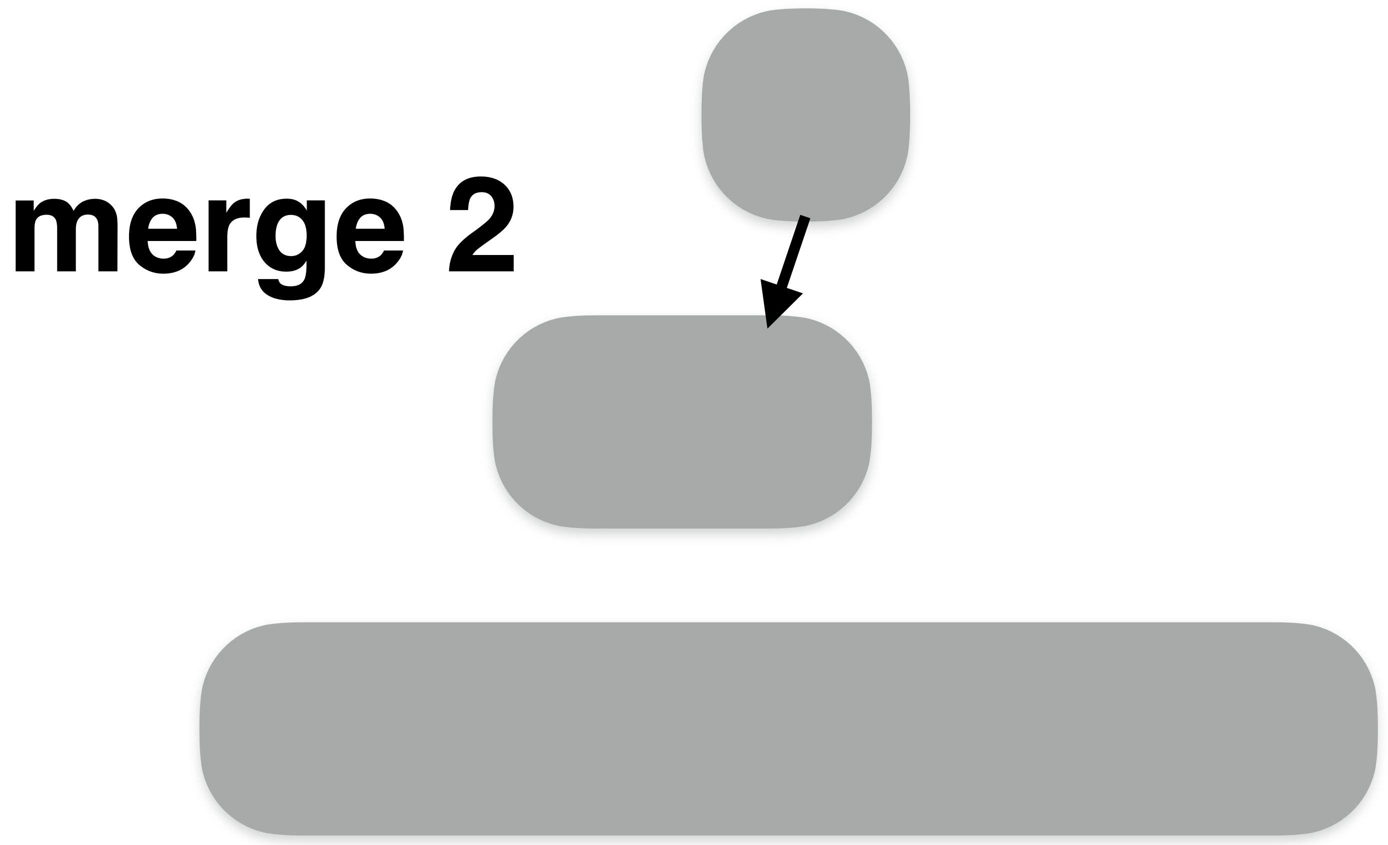
writes
=
=
all levels
more work
less frequent

writes

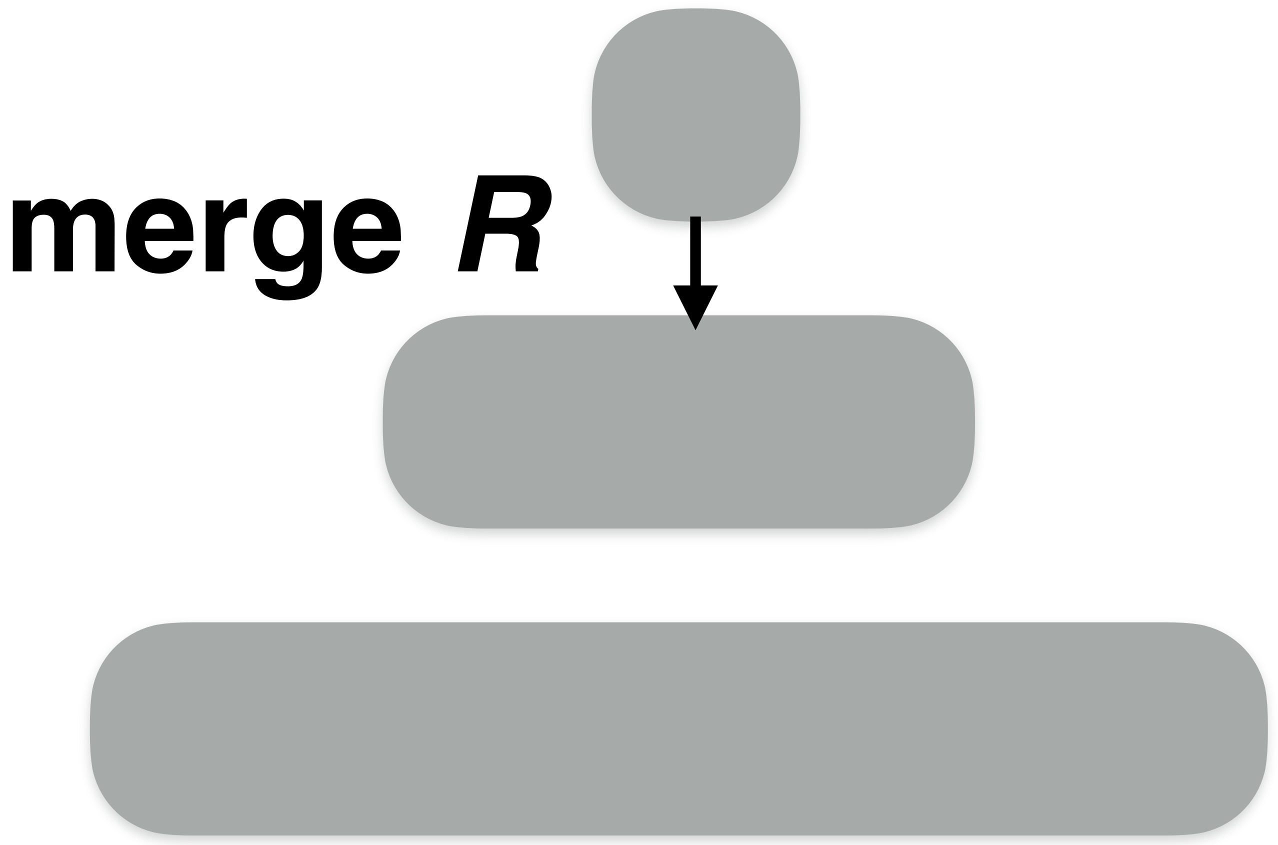



writes
merge 1

writes



writes



writes

write-amplification

$O(\boldsymbol{R})$

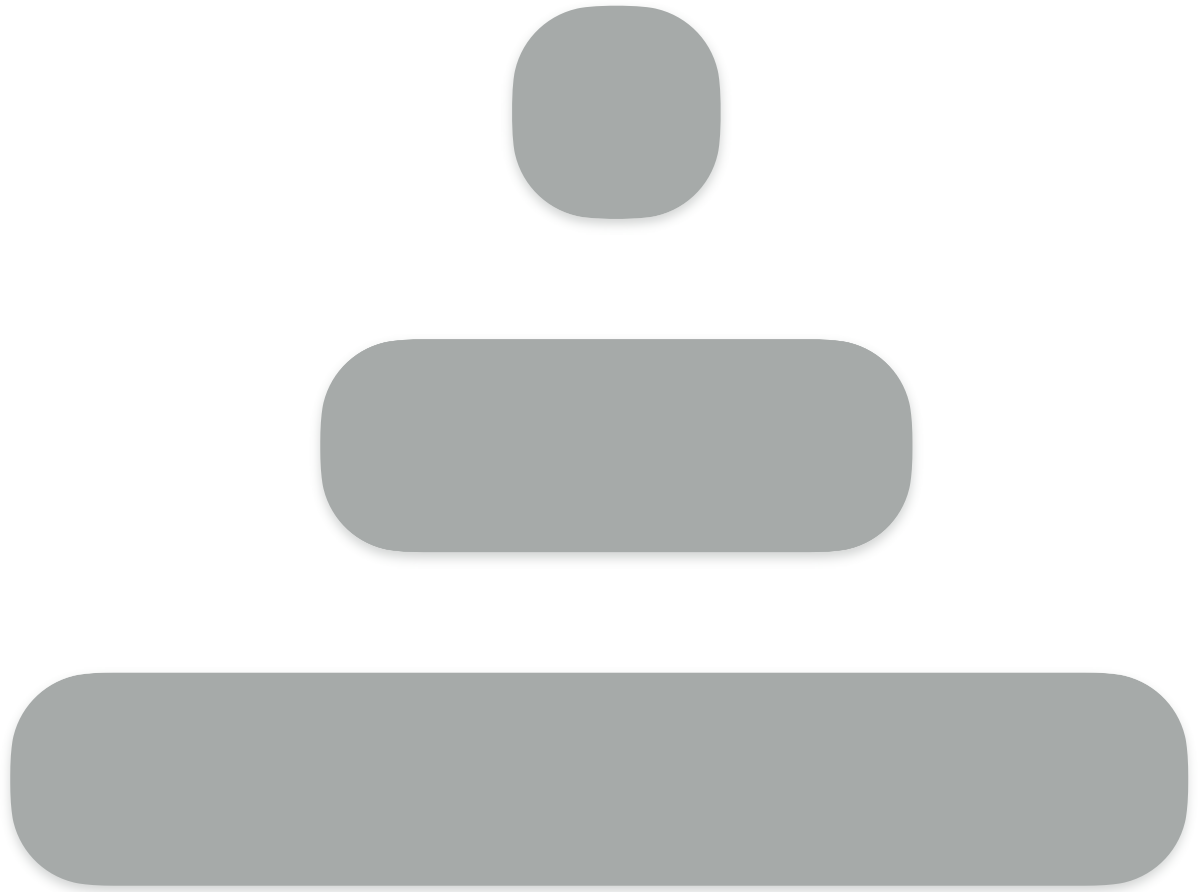
$O(\boldsymbol{R})$

$O(\boldsymbol{R})$

writes

$$\left.\begin{array}{c} O(R) \\ O(R) \\ O(R) \end{array}\right\} \quad \mathbf{O(R \cdot \log_R(N))}$$

point	long range	short range	writes
largest level	largest level	all levels	all levels
$O(e^{-x})$	$O(s)$	$O(\log_R(N))$	$O(R \cdot \log_R(N))$
=	=	=	=
$O(e^{-x}/R^2)$	$O(s/R^2)$	1	$O(R)$
+	+	+	+
$O(e^{-x}/R)$	$O(s/R)$	1	$O(R)$
+	+	+	+
$O(e^{-x})$	$O(s)$	1	$O(R)$



point

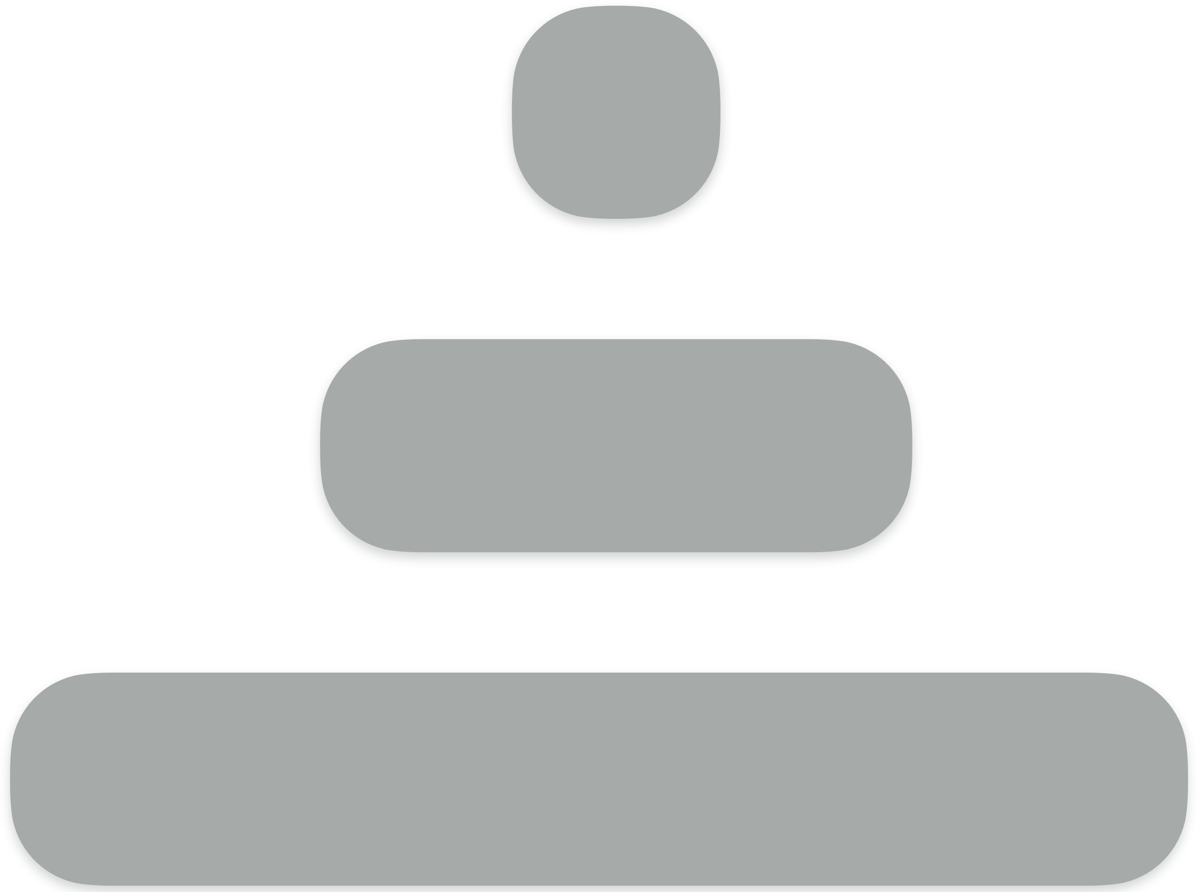
long range

writes

largest level

largest level

all levels



O(R)

O(R)

O(R)

O(e^{-x})

O(s)

point

largest level

$O(e^{-x})$

long range

largest level

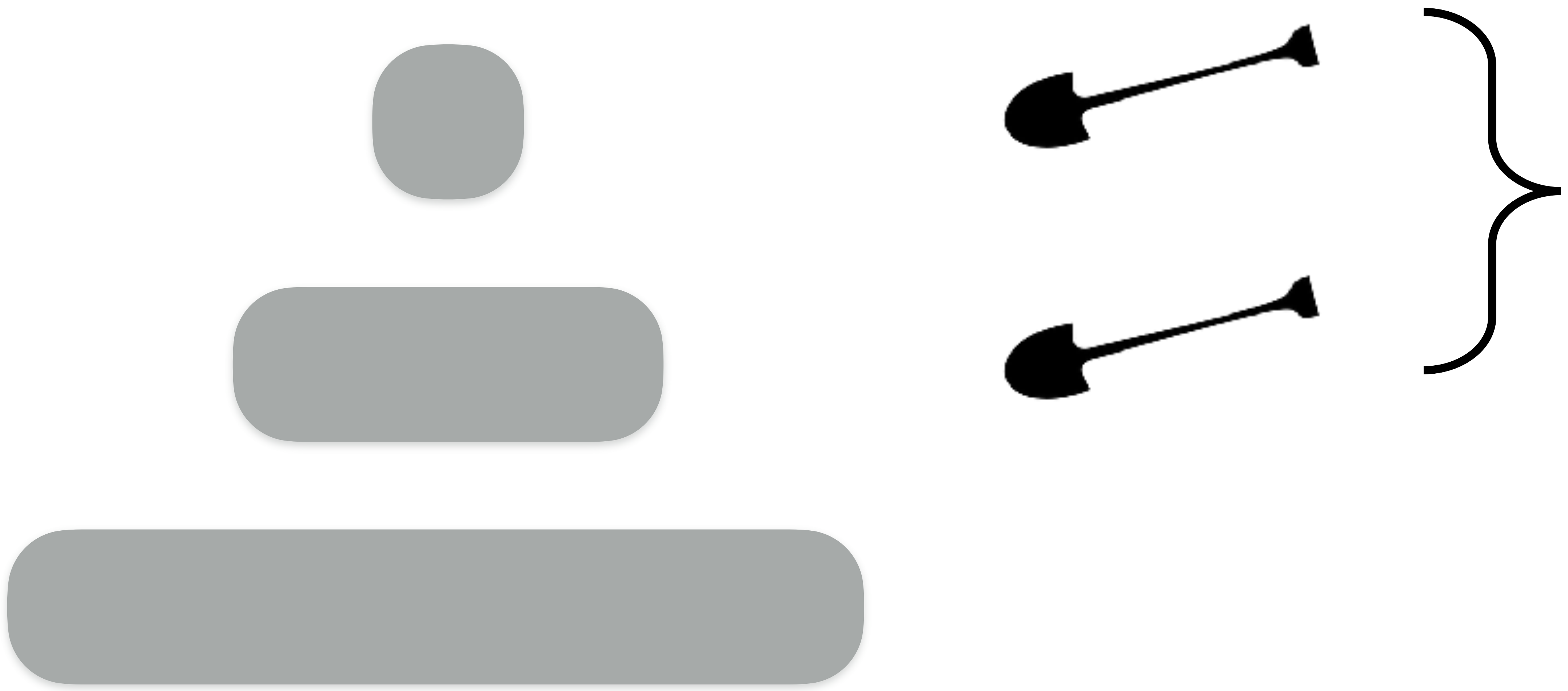
$O(s)$

writes

all levels

for point lookups and long range lookups

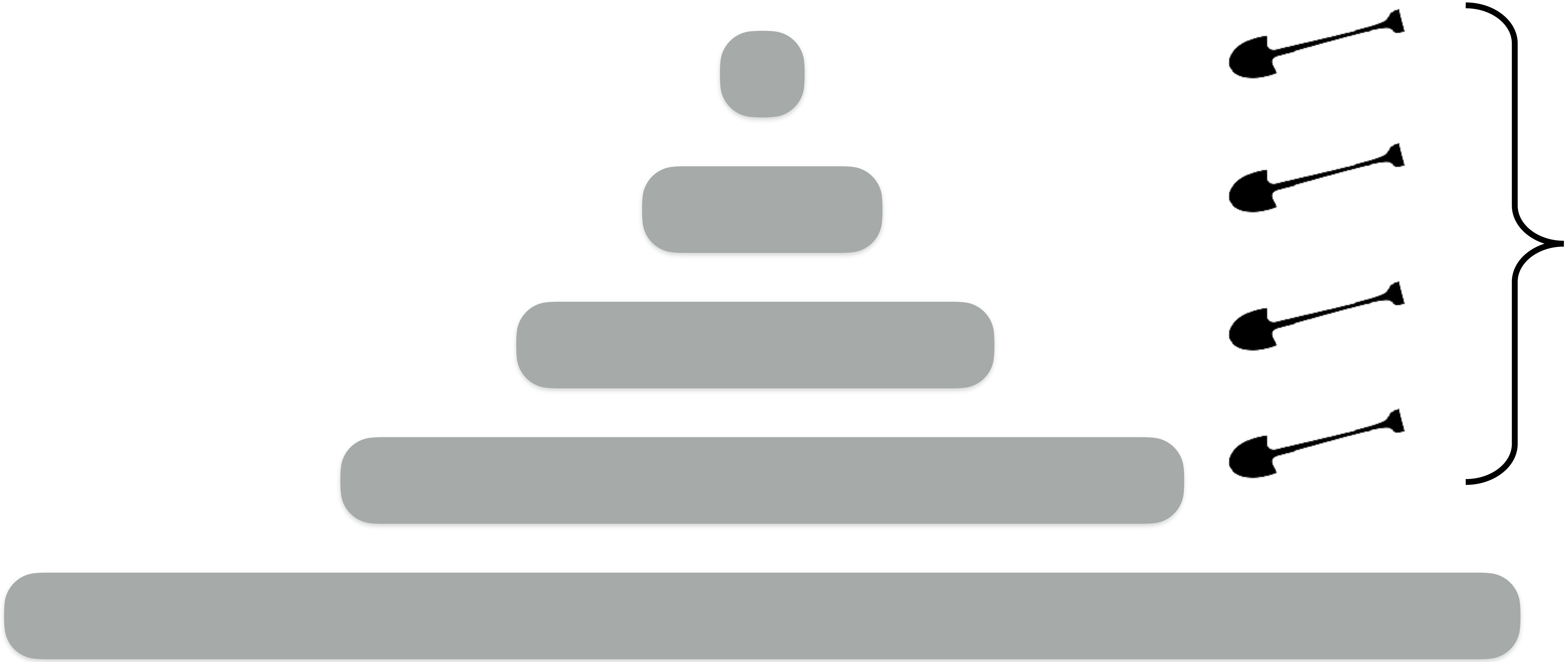
merging at **smaller levels** is **superfluous**



worse as data grows!

@ Harvard SEAS

poor performance



poor performance

lower device lifetime (on SSD)

DASlab @ Harvard SEAS

Dostoevsky

SIGMOD18

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

DASlab @ Harvard SEAS

very write-optimized

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

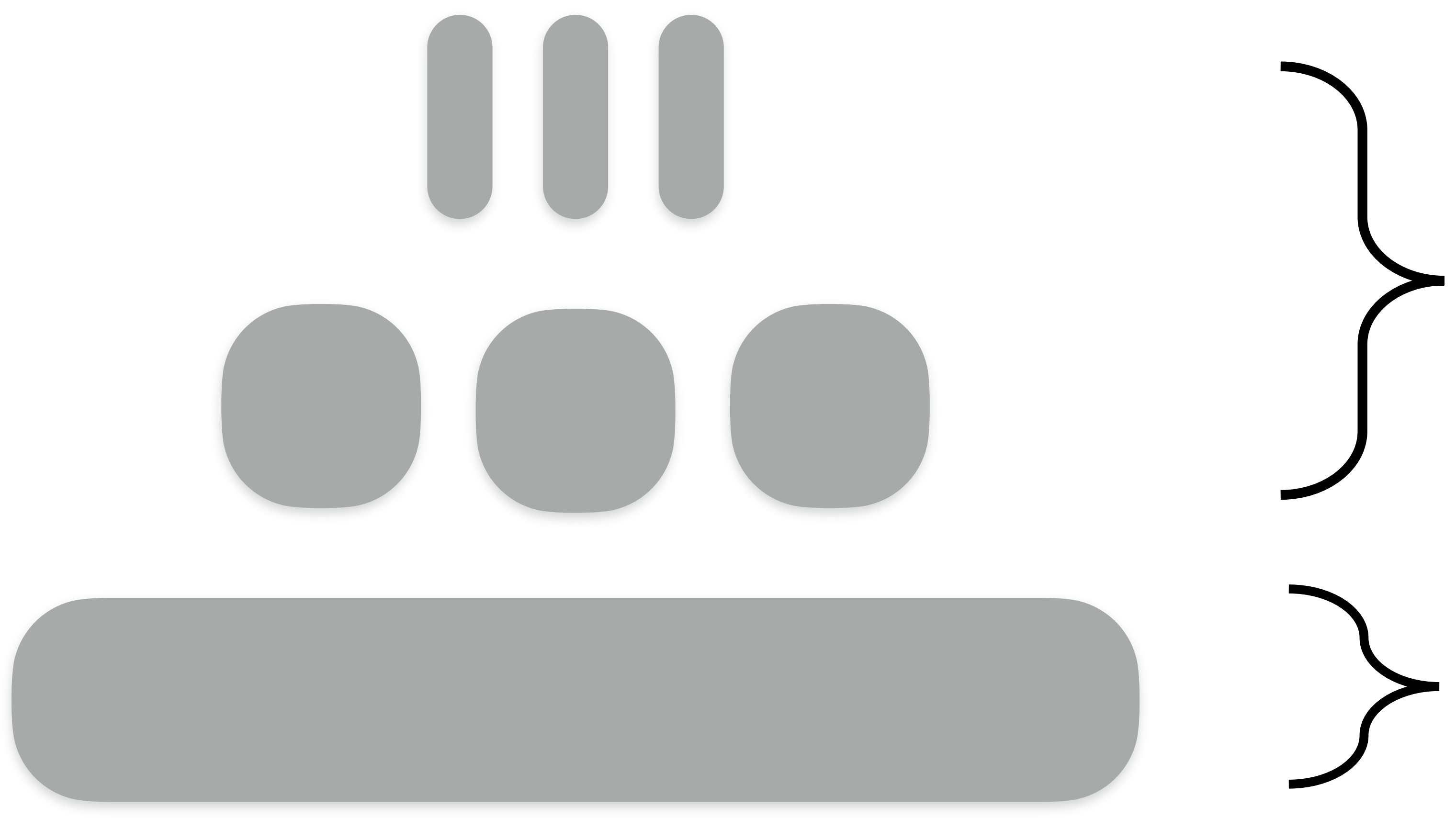
DASlab @ Harvard SEAS

Tiering
write-optimized

Leveling
read-optimized

Tiering	**Lazy Leveling**	Leveling
write-optimized	mixed-optimized	read-optimized

Lazy Leveling



Tiering

Leveling

Lazy Leveling

merge when level fills

merge to have at most 1 run

point

long range

short range

writes

point

false positive rates

$O(e^{-x}/R^3)$

$O(e^{-x}/R^2)$

$O(e^{-x})$

point

false positive rates

exponentially decreasing

$O(e^{-x}/R^3)$

$O(e^{-x}/R^2)$

$O(e^{-x})$

false positive rates

$O(e^{-x}/R^3)$

$O(e^{-x}/R^2)$

point $\longrightarrow$ $O(e^{-x})$

point

$O(e^{-x})$

point

$O(e^{-x})$

with uniform FPRs

$$O(\log_R(N) \cdot R \cdot e^{-x}) \left\{ \begin{array}{l} O(e^{-x}) \\ O(e^{-x}) \\ O(e^{-x}) \end{array} \right.$$

point | **long range** | short range | writes
$O(e^{-x})$

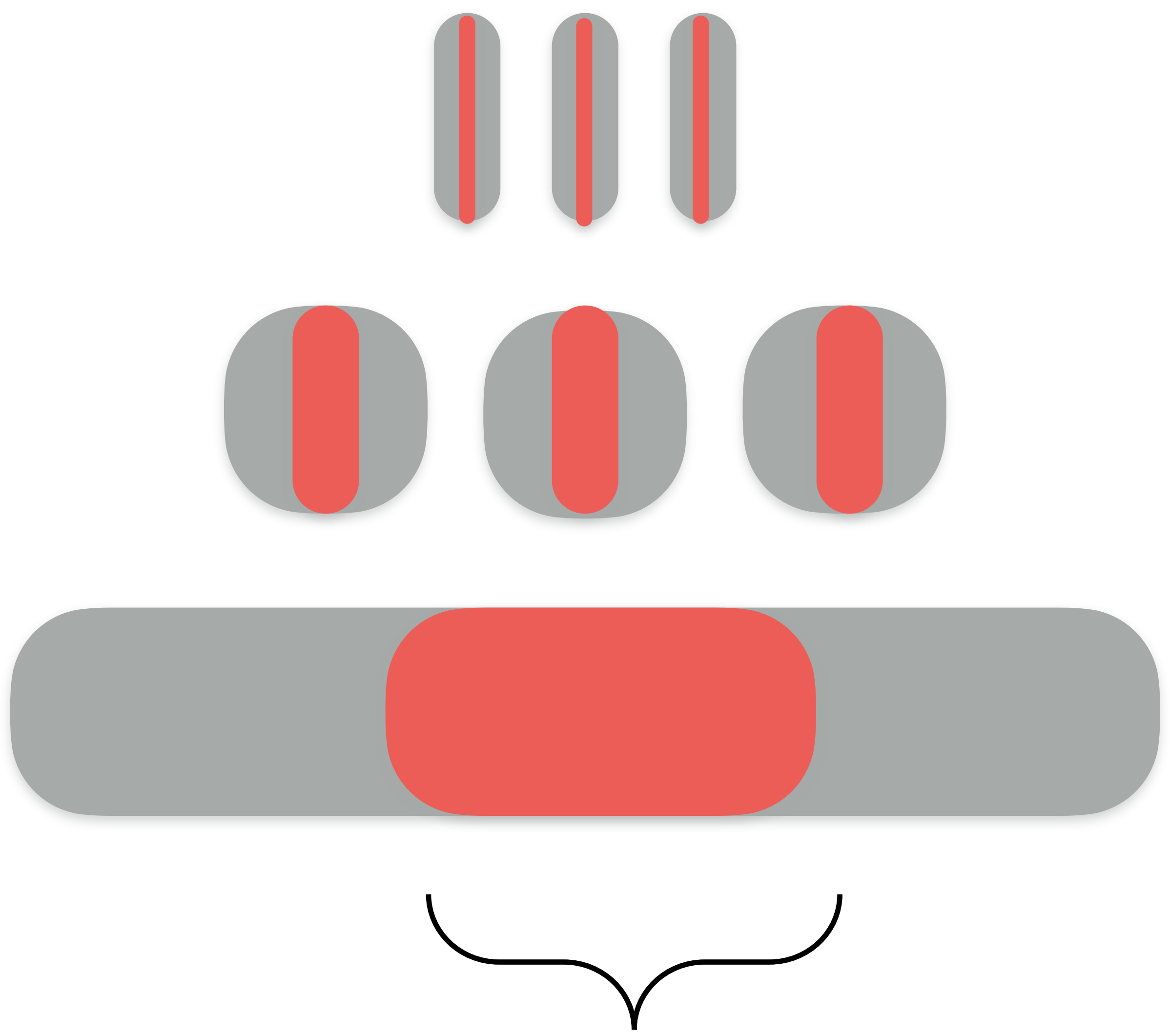
long range

target range

$O(s/R^2)$

$O(s/R)$

$O(s)$



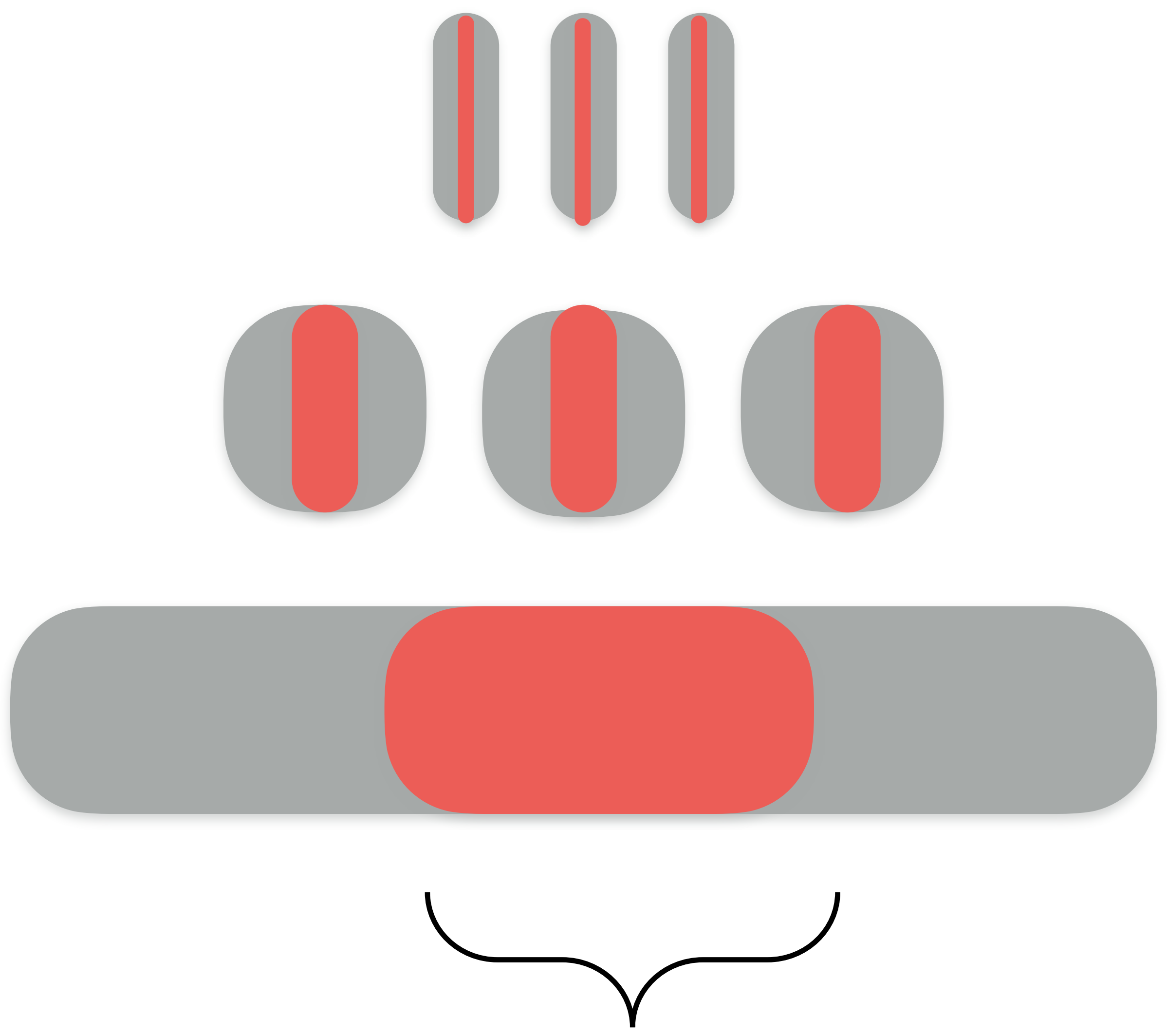
target key range

DASlab @ Harvard SEAS

target range

$O(s/R^2)$

$O(s/R)$



long range

$O(s)$

largest level

target key range

point

long range

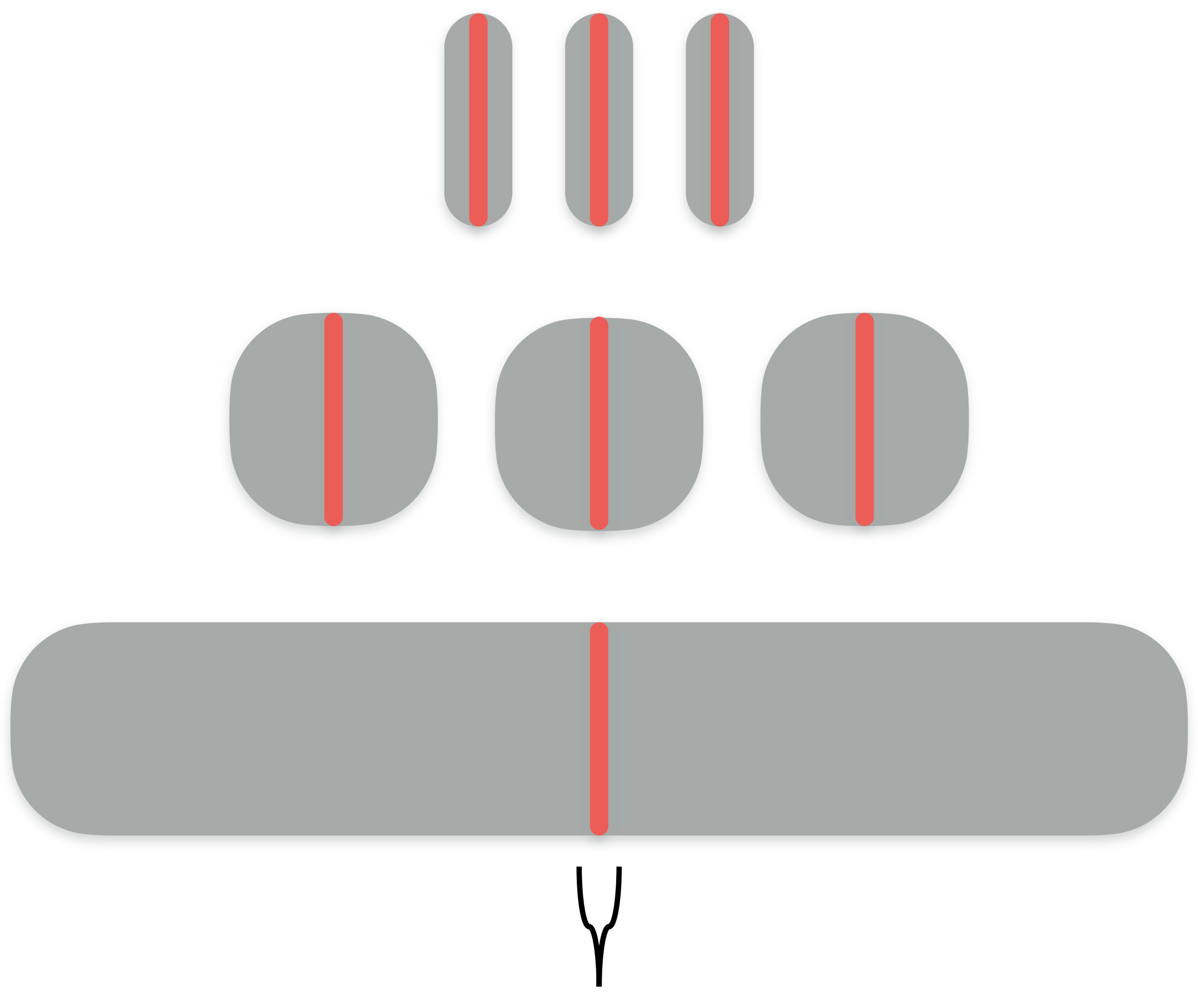
short range

writes

$O(e^{-x})$

$O(s)$

short range



$O(R)$

$O(R)$

1

$$O(1+R\cdot(\log_R(N)-1))$$

target key range

point	long range	short range	writes
$O(e^{-x})$	$O(s)$	$O(1+R\cdot(\log_R(N)-1))$	

writes

write-amplification

O(**1**)

O(**1**)

O(***R***)

writes

write-amplification

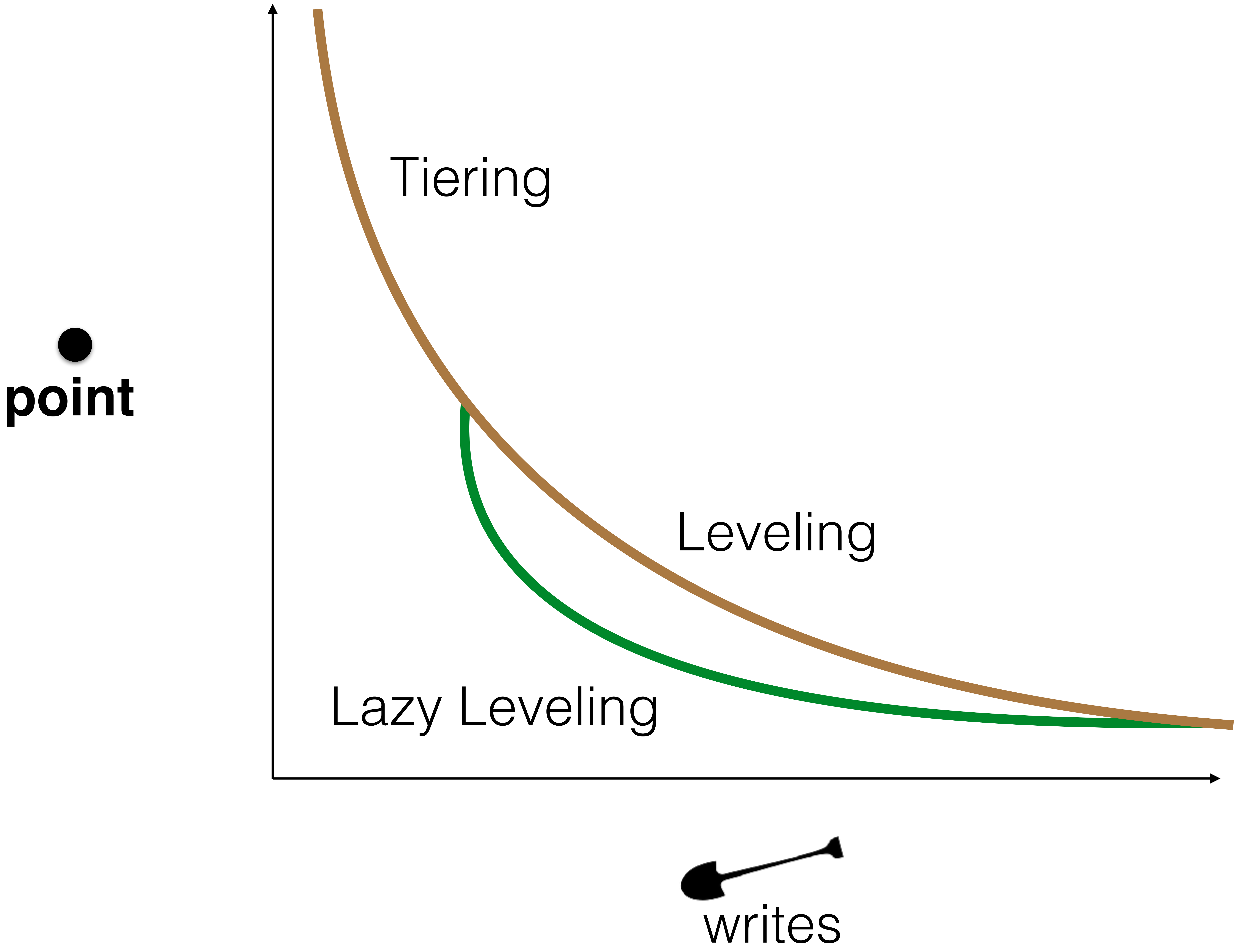
$$\left.\begin{array}{l} O(1) \\ O(1) \\ O(R) \end{array}\right\} O(\,R + \log_R(N)\,)$$

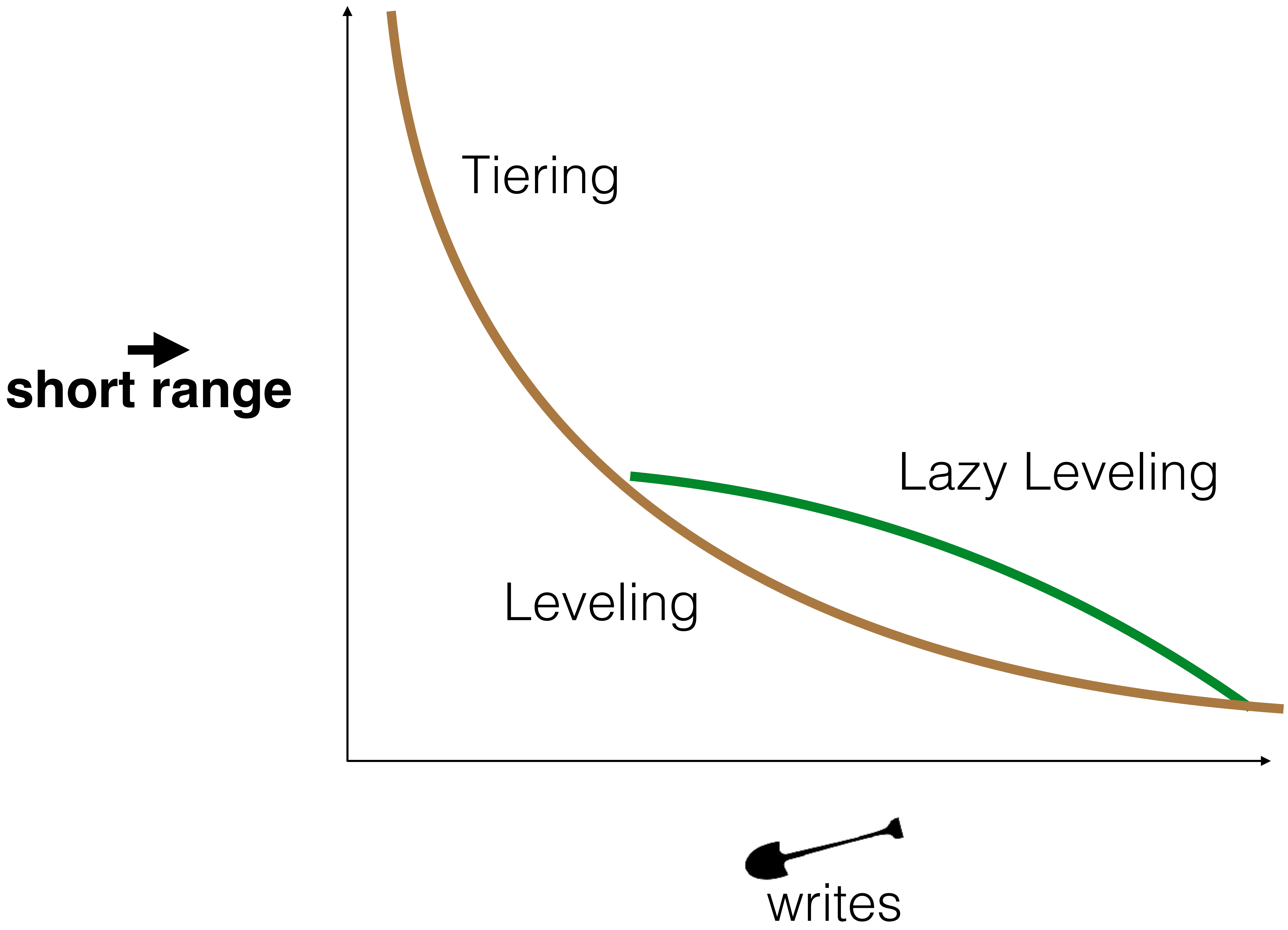
point	long range	short range	writes
$O(e^{-x})$	$O(s)$	$O(1+R \cdot (\log_R(N)-1))$	$O(R + \log_R(N))$

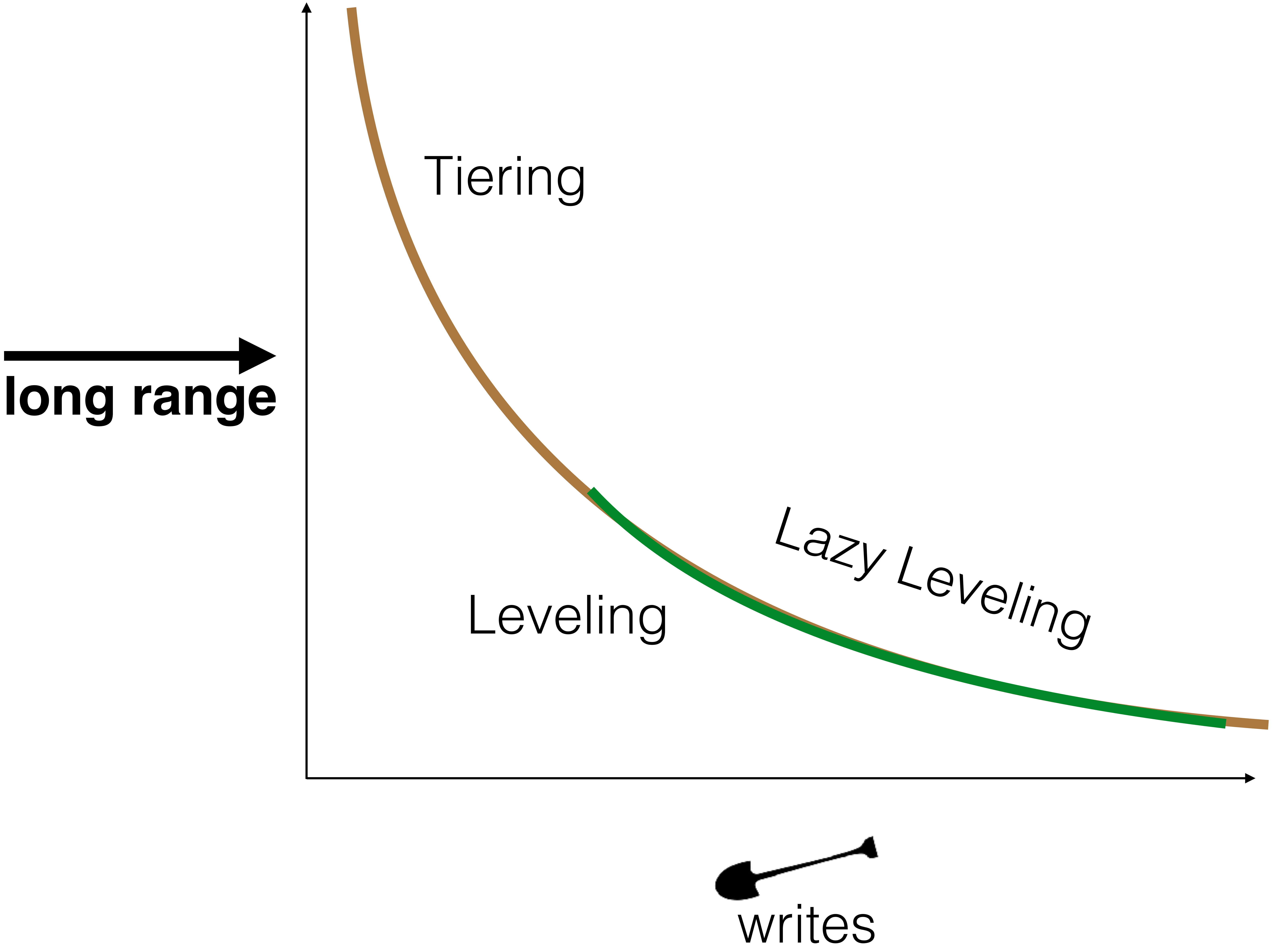
DASlab @ Harvard SEAS

	point	long range	short range	writes
Lazy Leveling	$O(e^{-x})$	$O(s)$	$O(1+R \cdot (\log_R(N)-1))$	$O(R + \log_R(N))$
	$=$	$=$	$\vee$	$\wedge$
Leveling	$O(e^{-x})$	$O(s)$	$O(\log_R(N))$	$O(R \cdot \log_R(N))$

	point	long range	short range	writes
Tiering	$O(R \cdot e^{-x})$	$O(R \cdot s)$	$O(R \cdot \log_R(N))$	$O(\log_R(N))$
	$\vee$	$\vee$	$\vee$	$\wedge$
Lazy Leveling	$O(e^{-x})$	$O(s)$	$O(1+R \cdot (\log_R(N)-1))$	$O(R + \log_R(N))$
Leveling	$O(e^{-x})$	$O(s)$	$O(\log_R(N))$	$O(R \cdot \log_R(N))$


Tiering
Leveling
Lazy Leveling
point
writes


short range
Tiering
Lazy Leveling
Leveling
writes


long range
Tiering
Leveling
Lazy Leveling
writes

Tiering

Lazy Leveling

Leveling

Tiering

writes

Lazy Leveling

Leveling

Tiering

writes 👍

Lazy Leveling

Leveling

short range 👍

Tiering

writes

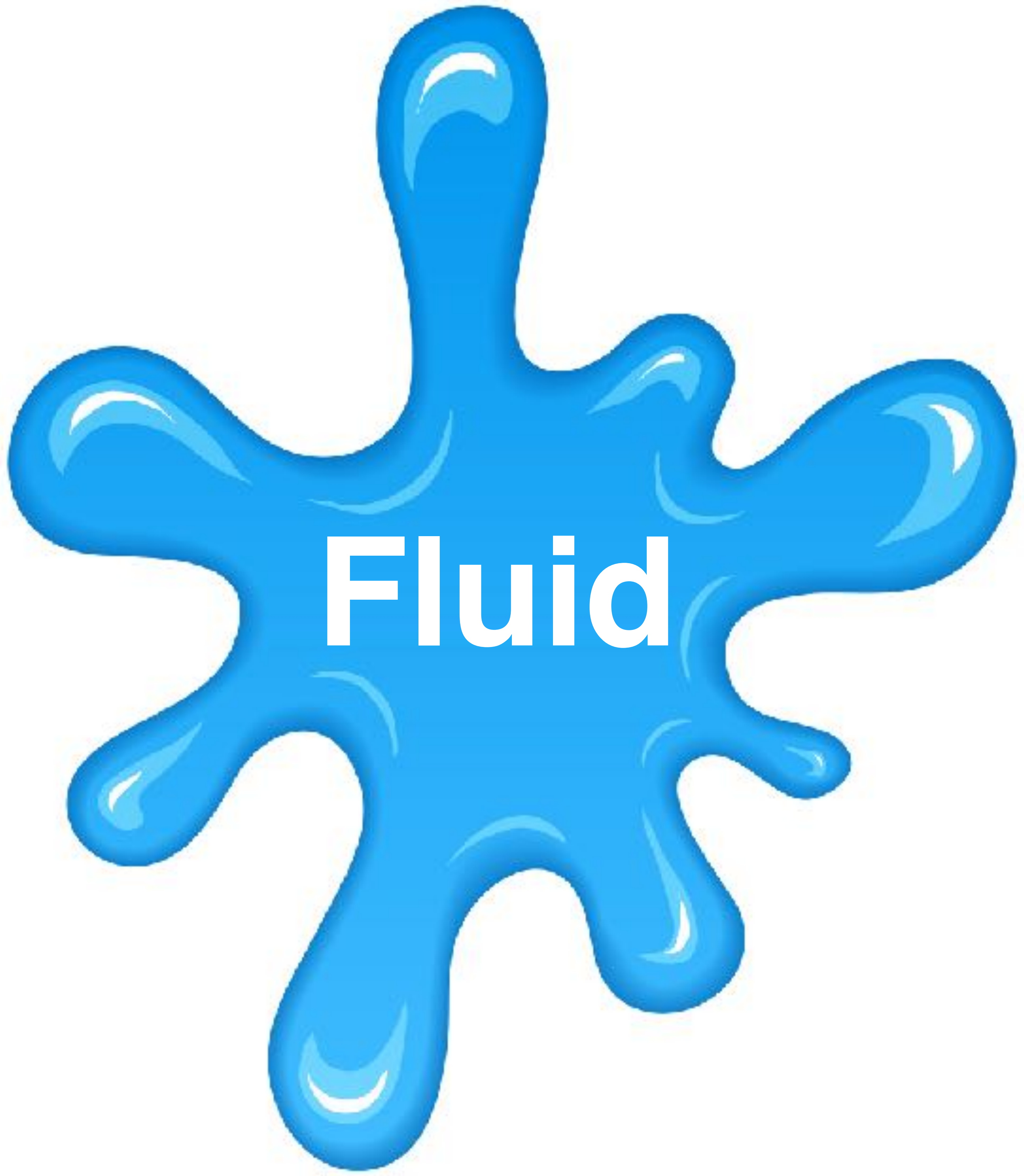
Lazy Leveling

writes & point

Leveling

short range

Lazy Leveling

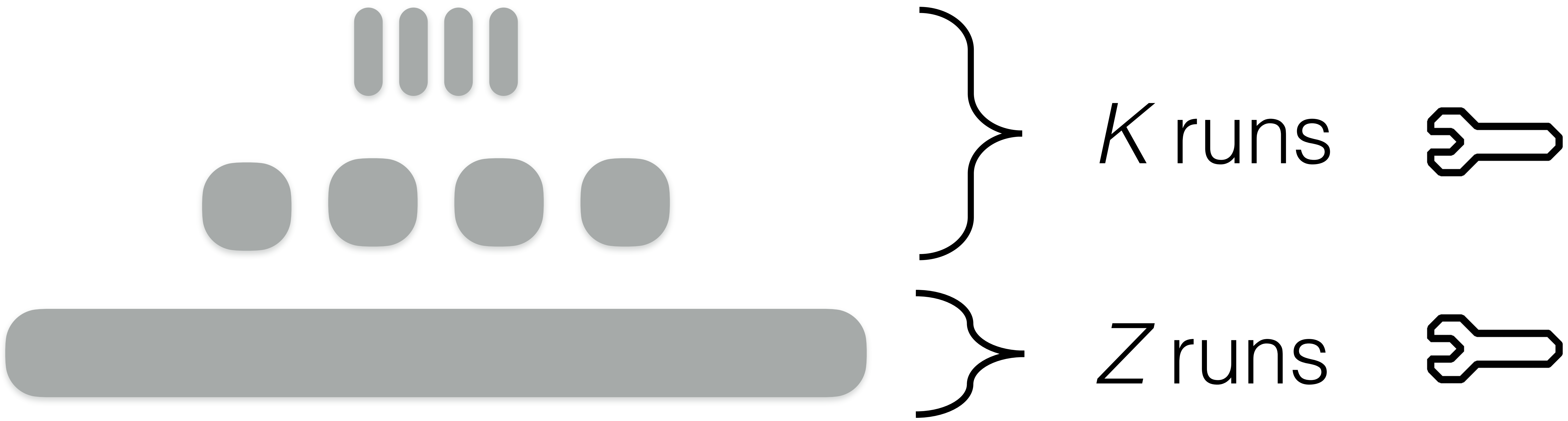


Tiering

Leveling

DASlab @ Harvard SEAS

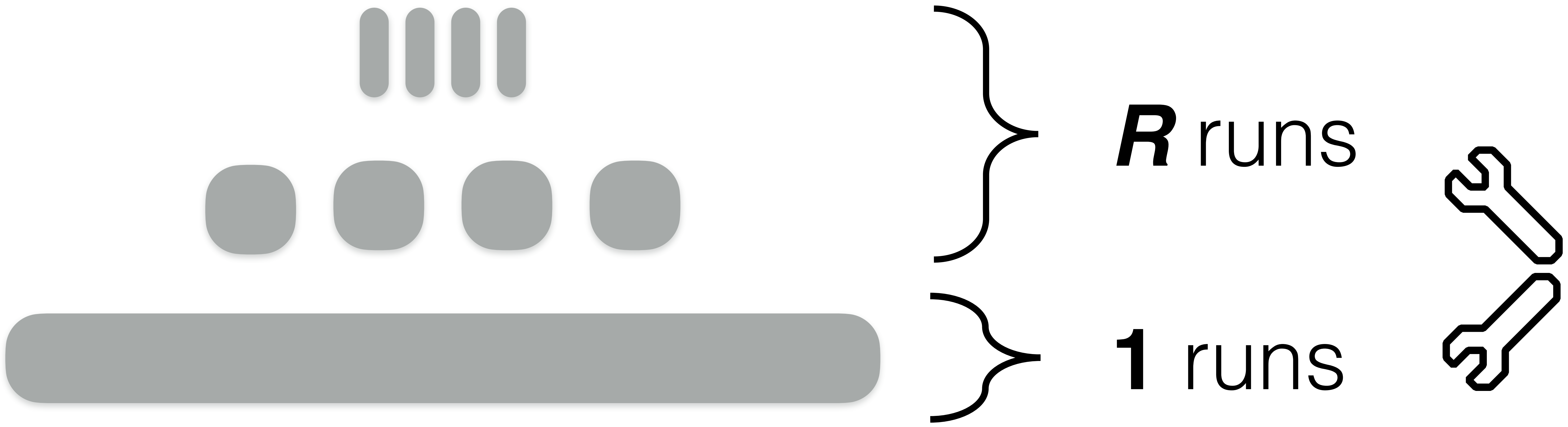
Fluid LSM-Tree



@ Harvard SEAS

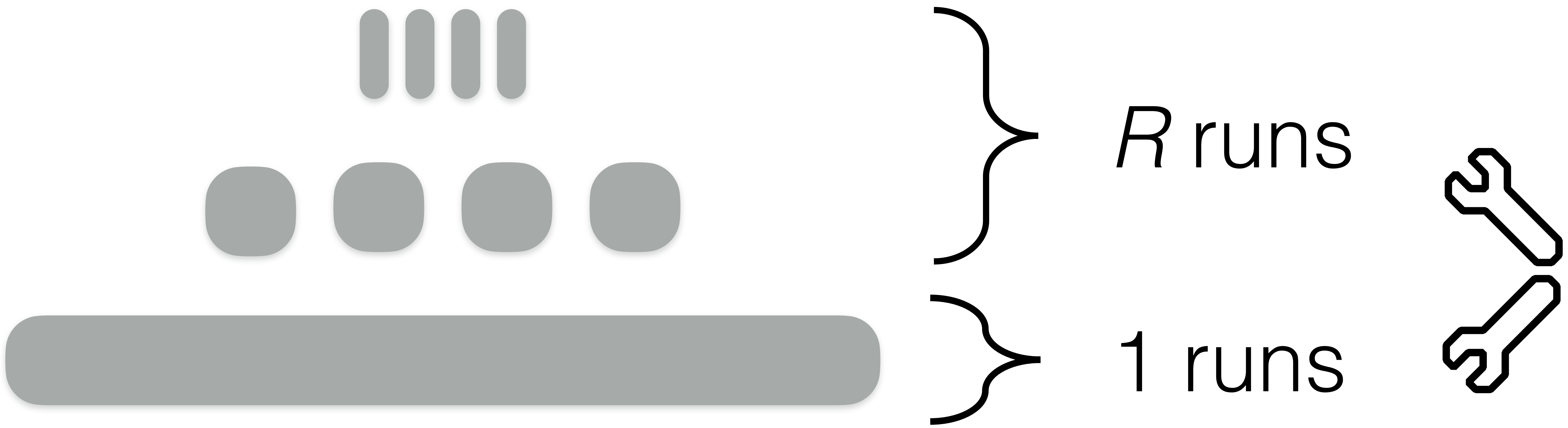
Fluid LSM-Tree

Lazy Leveling



point **long range** **short range** **writes**

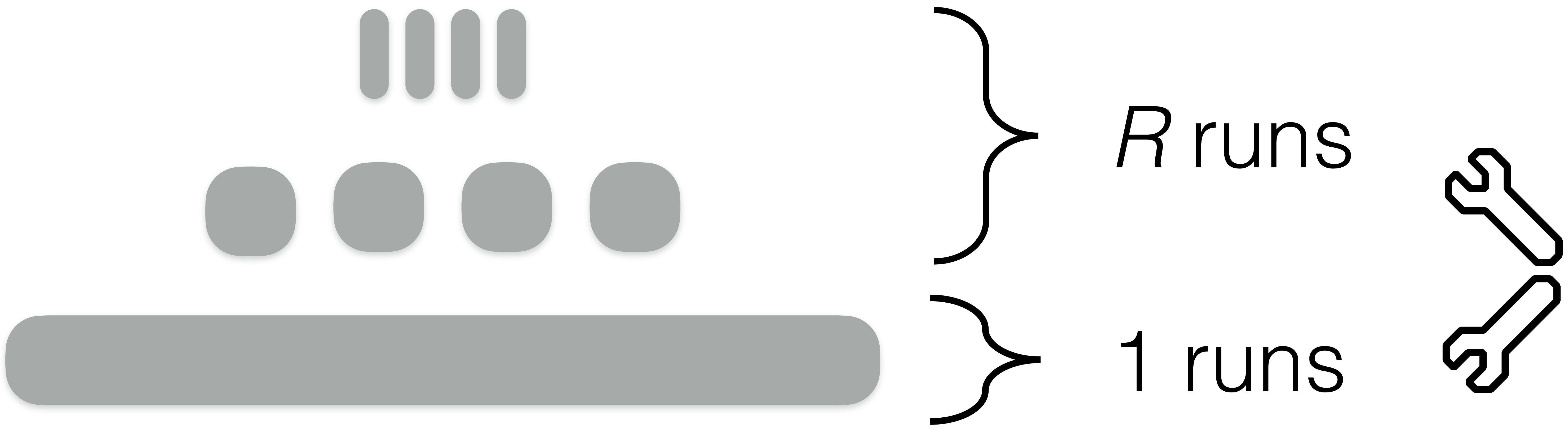
Lazy Leveling



optimize

point | long range | **short range** | writes

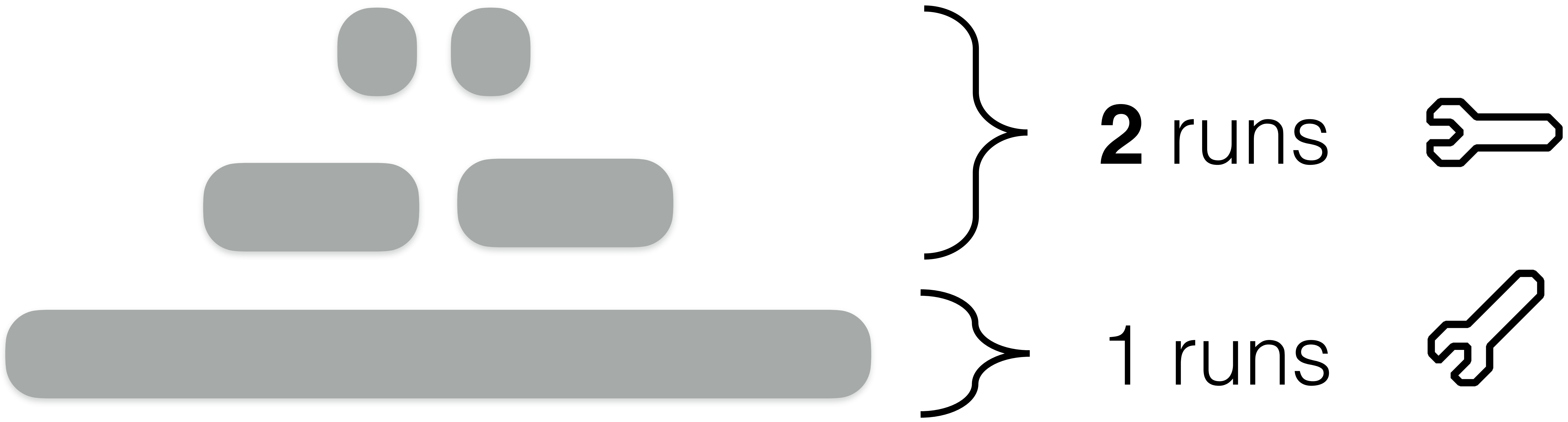
Lazy Leveling



optimize

point | long range | **short range** | **writes**

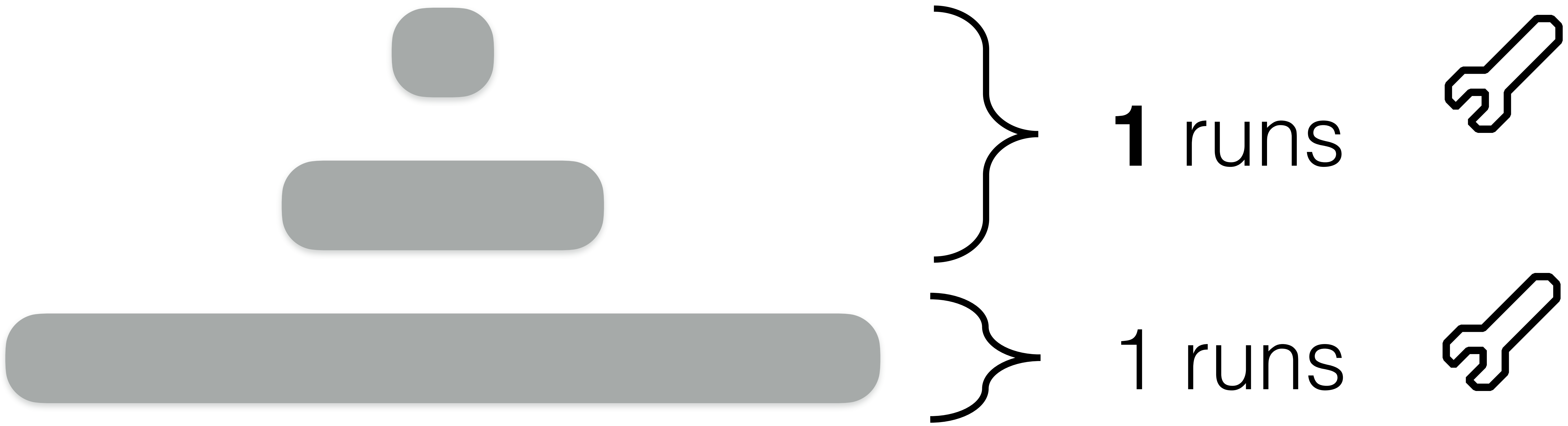
Lazy Leveling



optimize

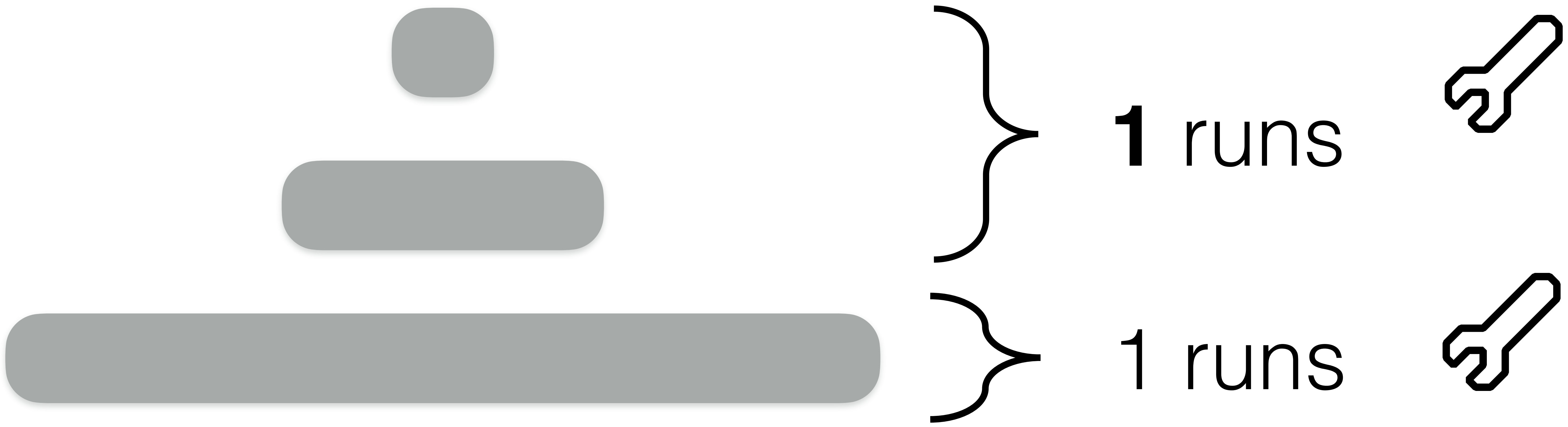
point | long range | **short range** | **writes**

Leveling



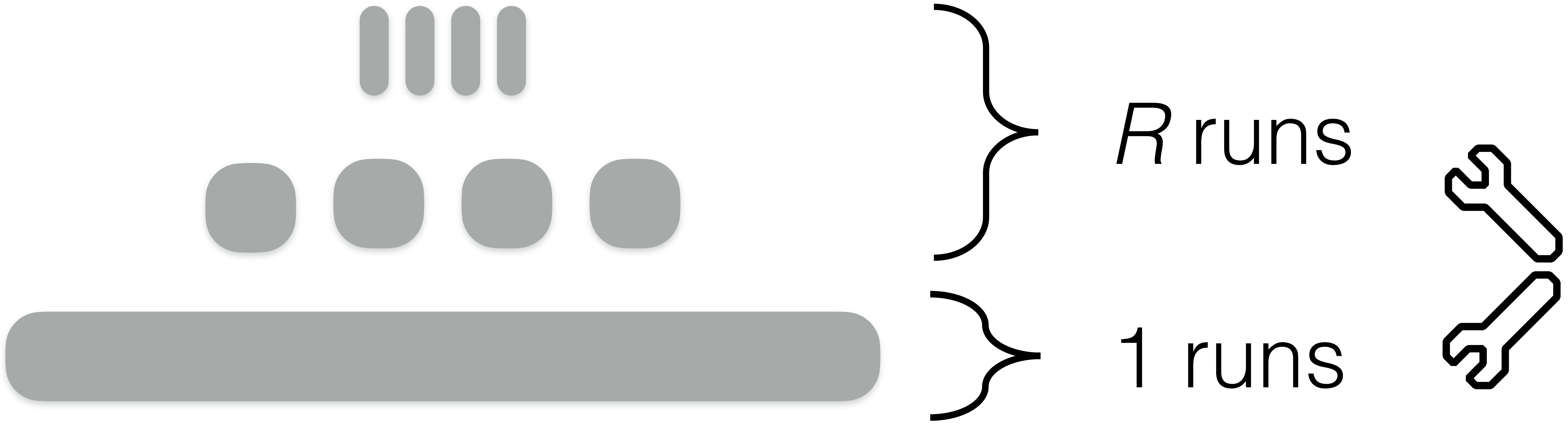
point | long range | short range | optimize **writes**

Leveling



point | long range | **short range** | optimize **writes**

Lazy Leveling



DASlab @ Harvard SEAS

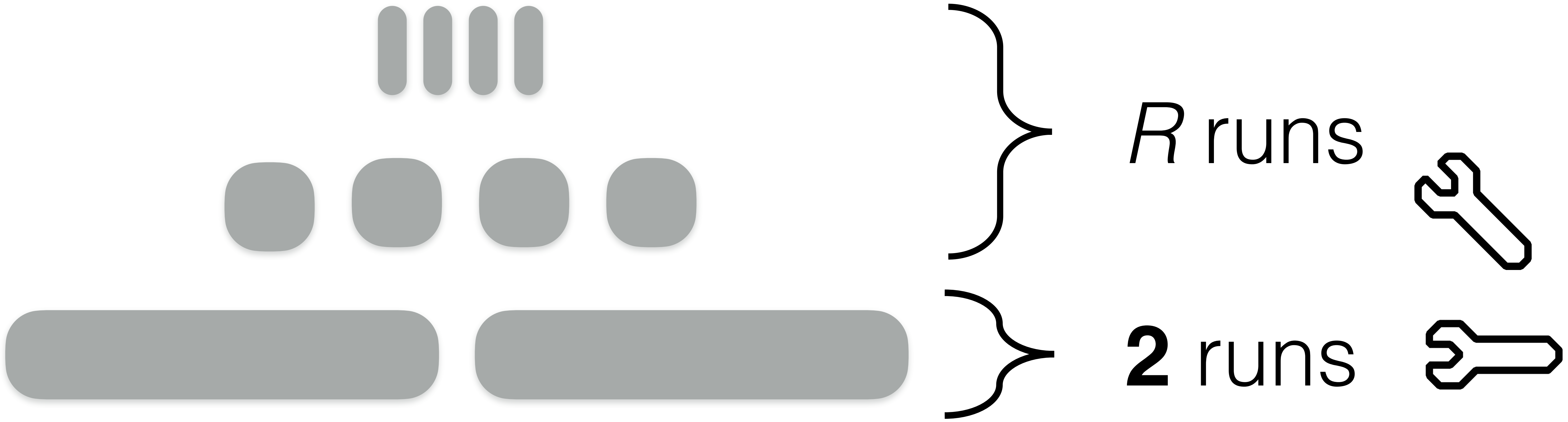
point

long range

short range

optimize
writes

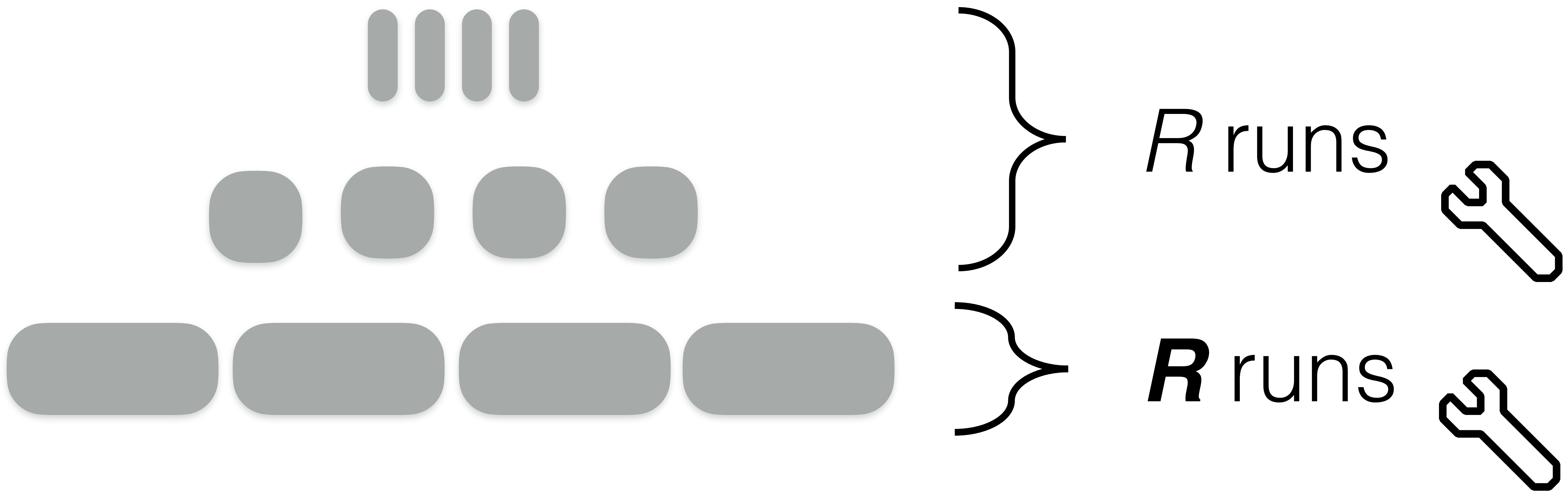
Lazy Leveling



optimize **writes**

point | long range | short range

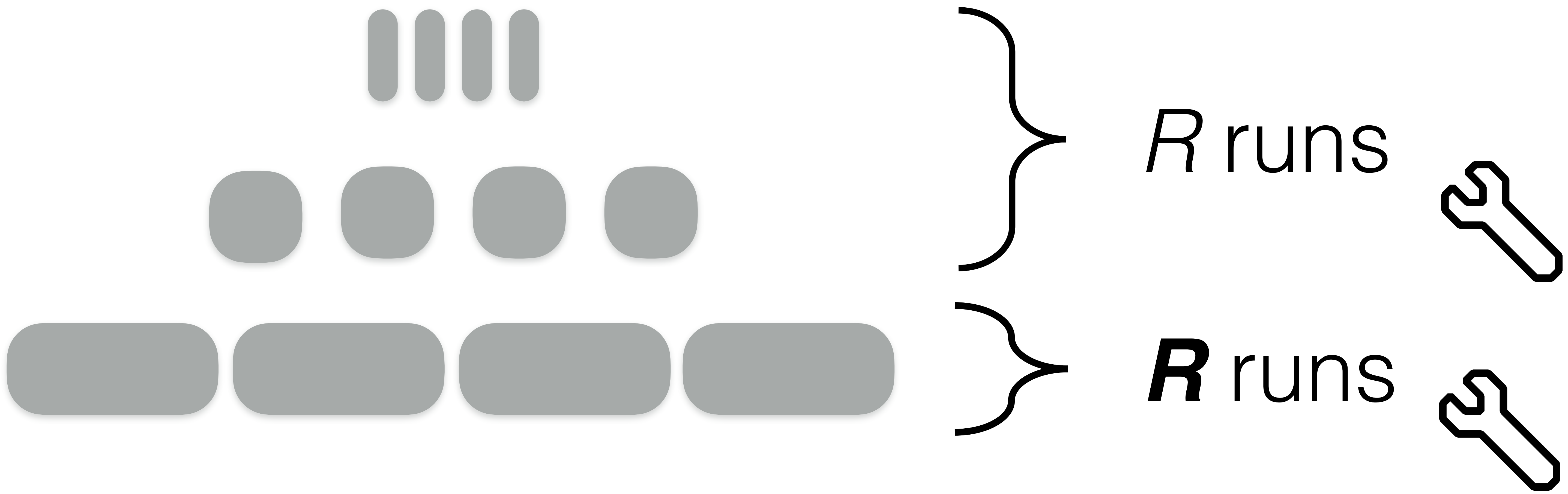
Tiering



optimize

point   long range   short range   writes

Tiering

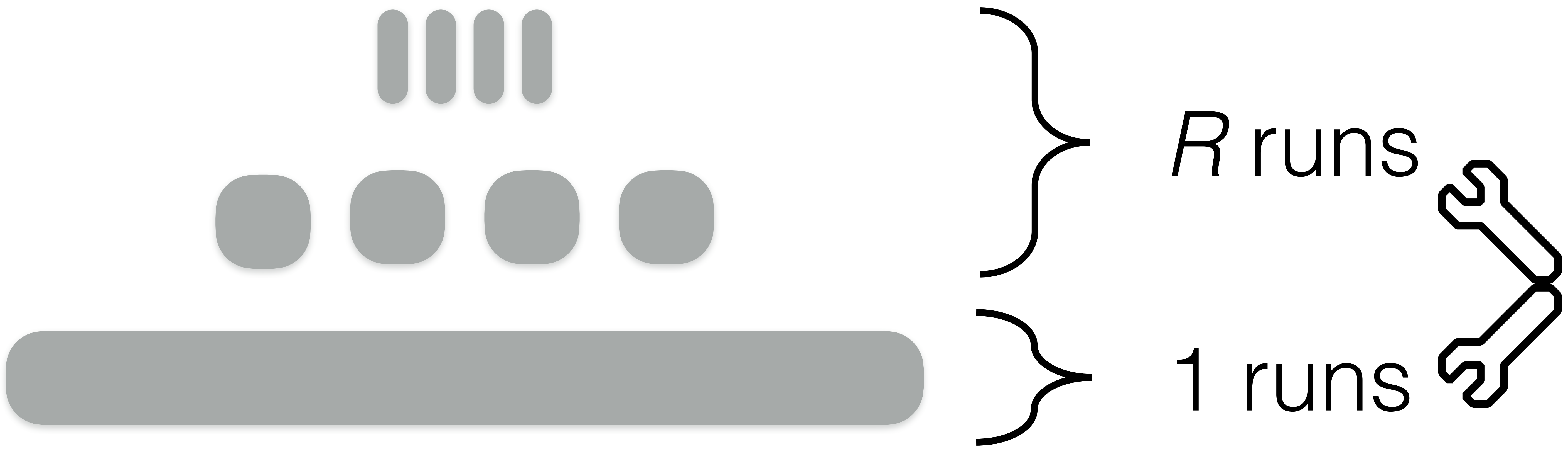


DASlab @ Harvard SEAS

optimize

point long range short range writes

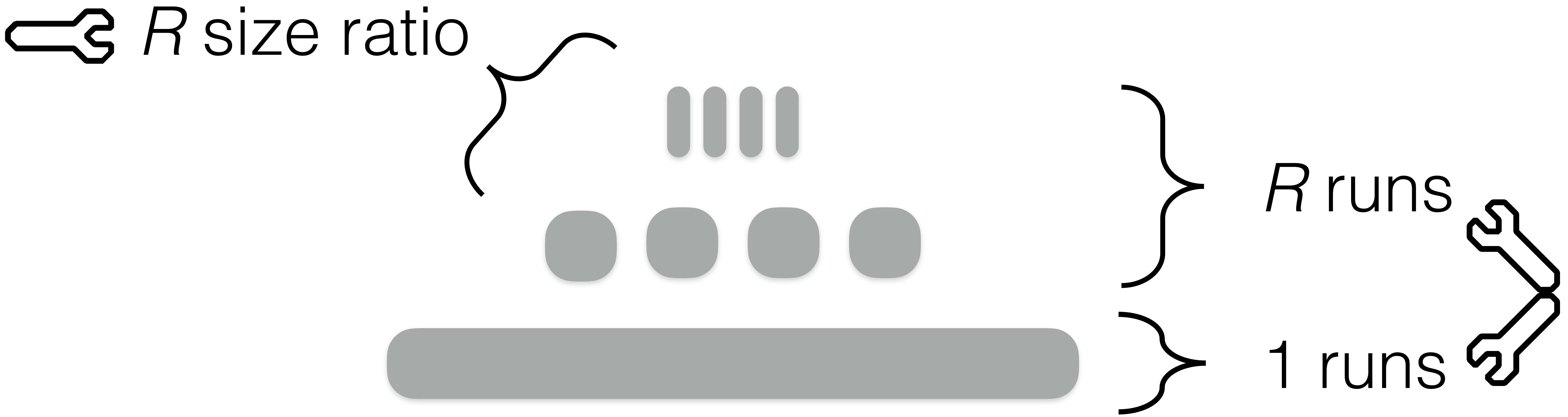
Lazy Leveling



optimize

point long range short range writes

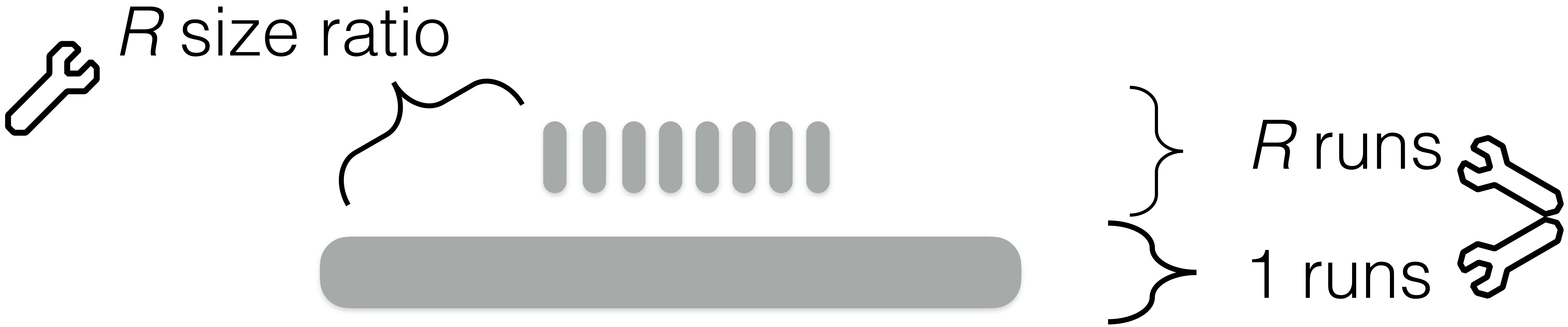
Lazy Leveling



optimize

point long range short range writes

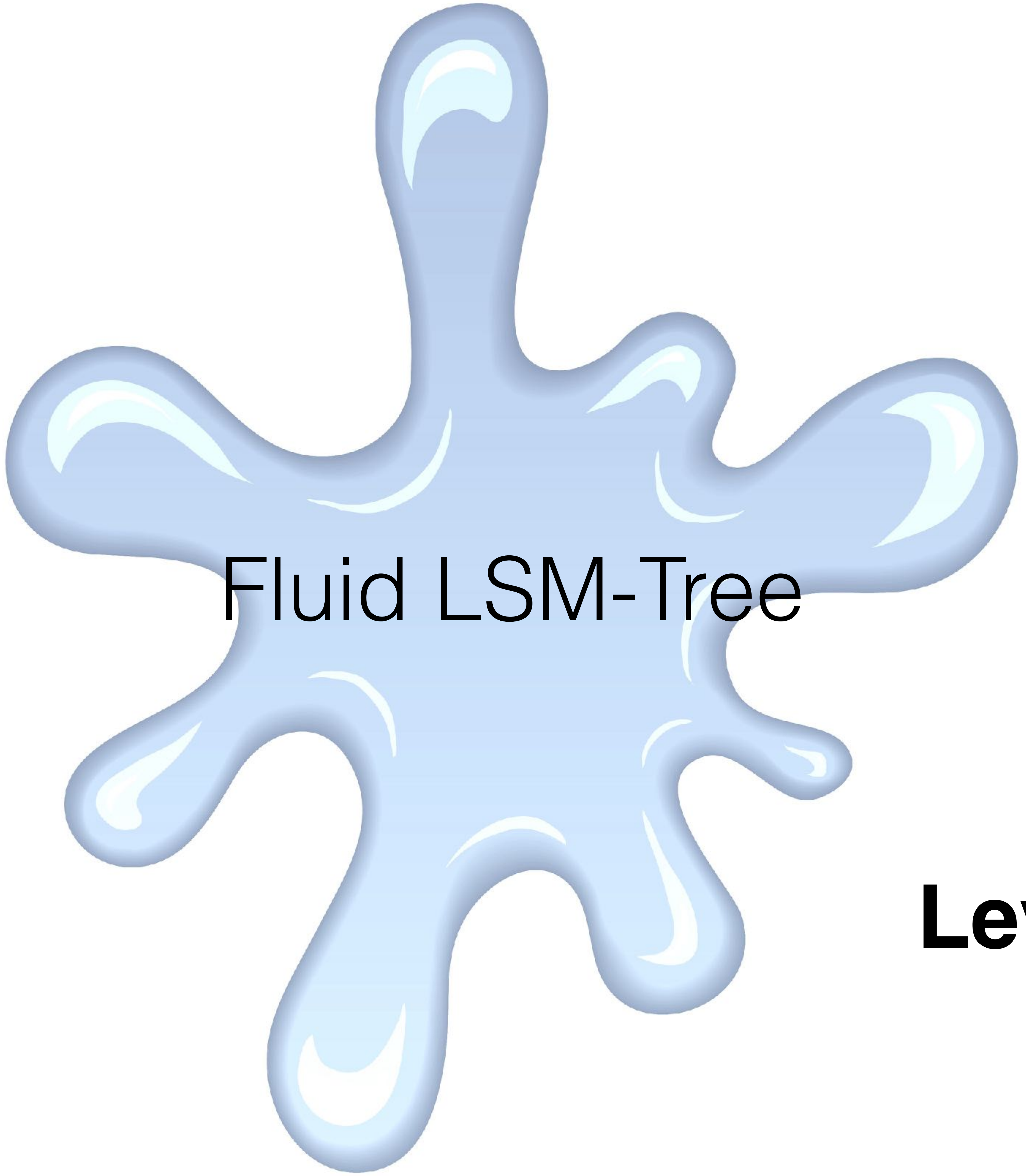
Lazy Leveling



Fluid LSM-Tree

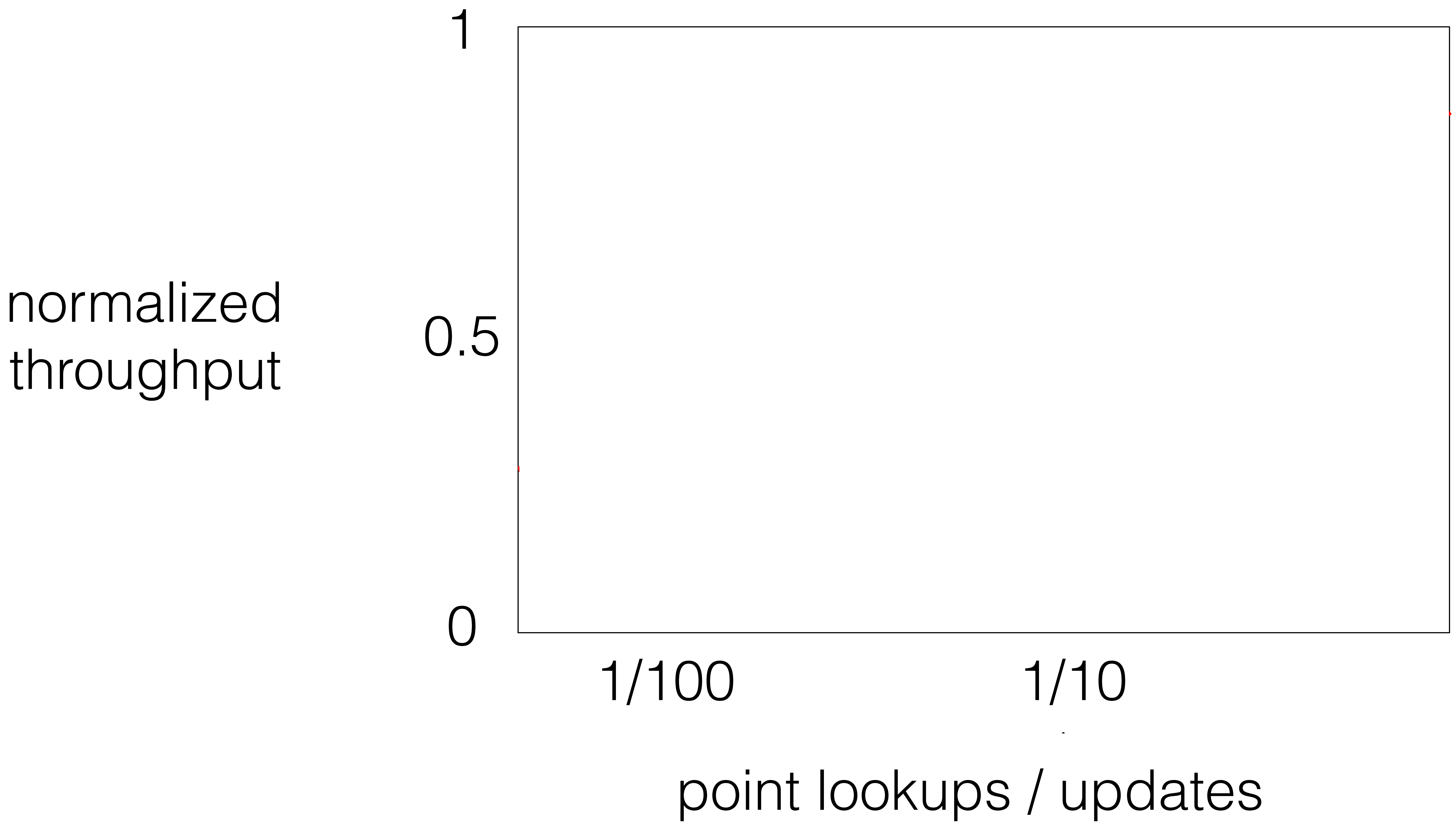
- *R* size ratio
- *K* runs at smaller levels
- *Z* runs at largest level

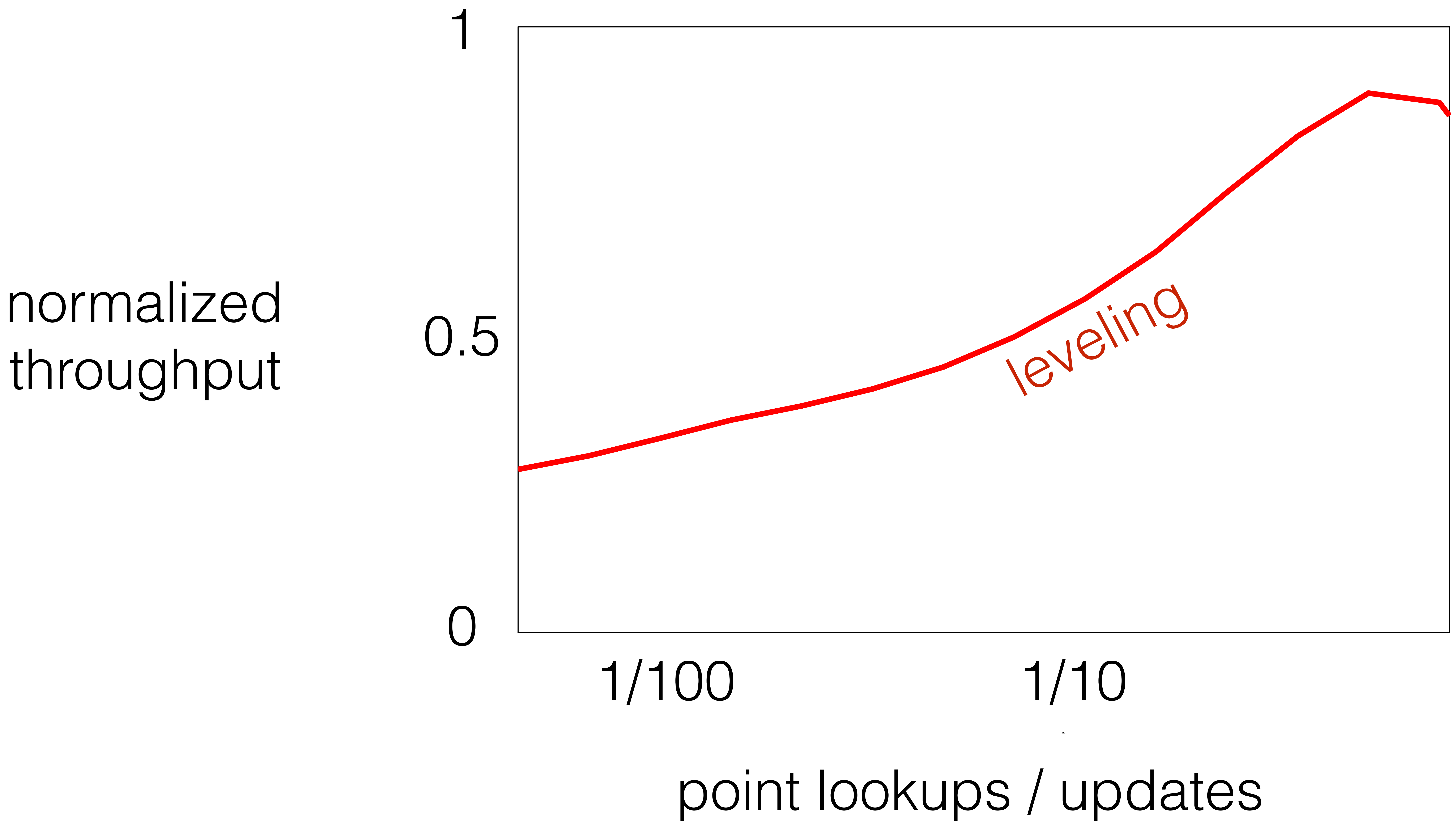
Lazy Leveling

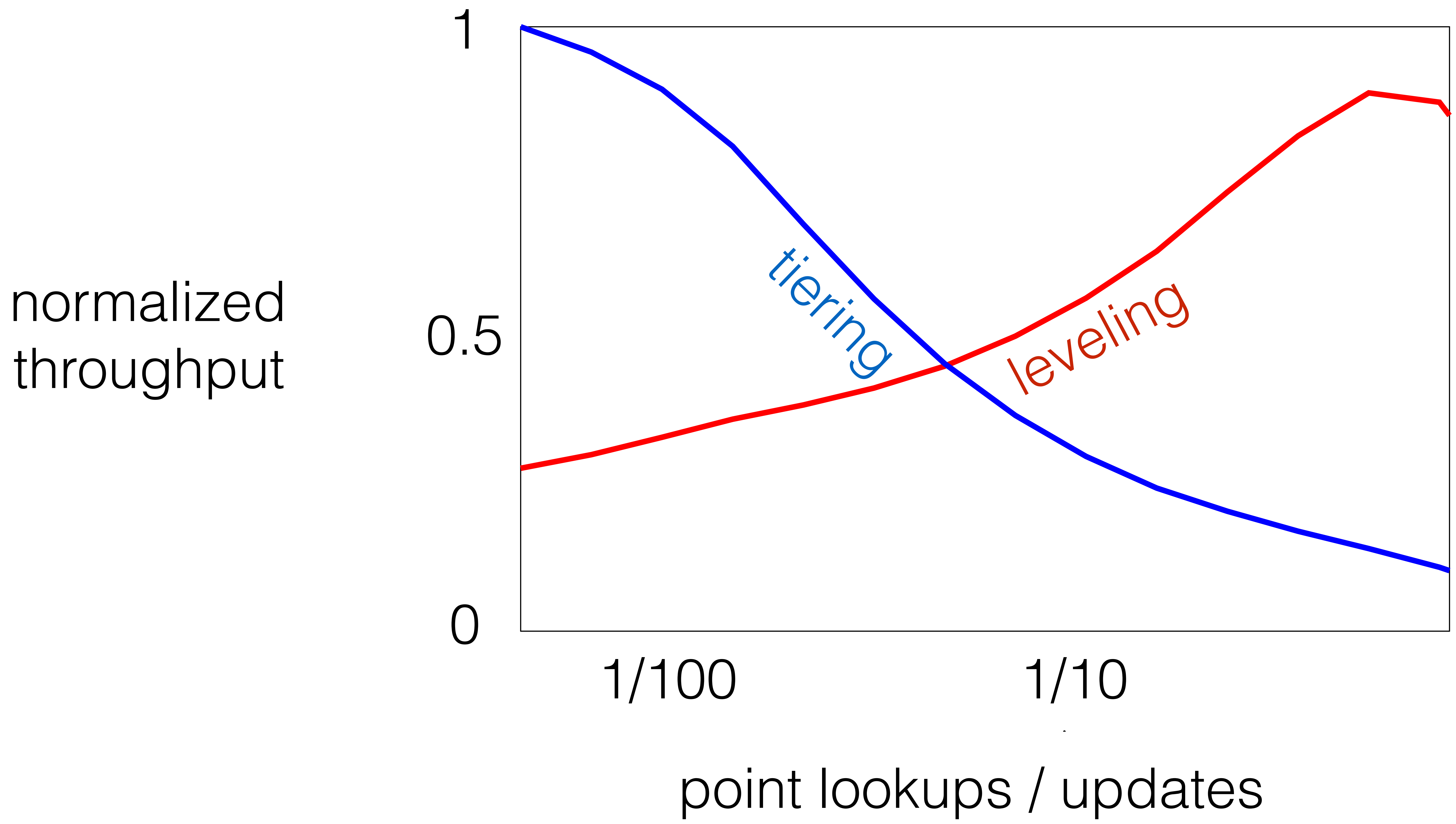


Tiering

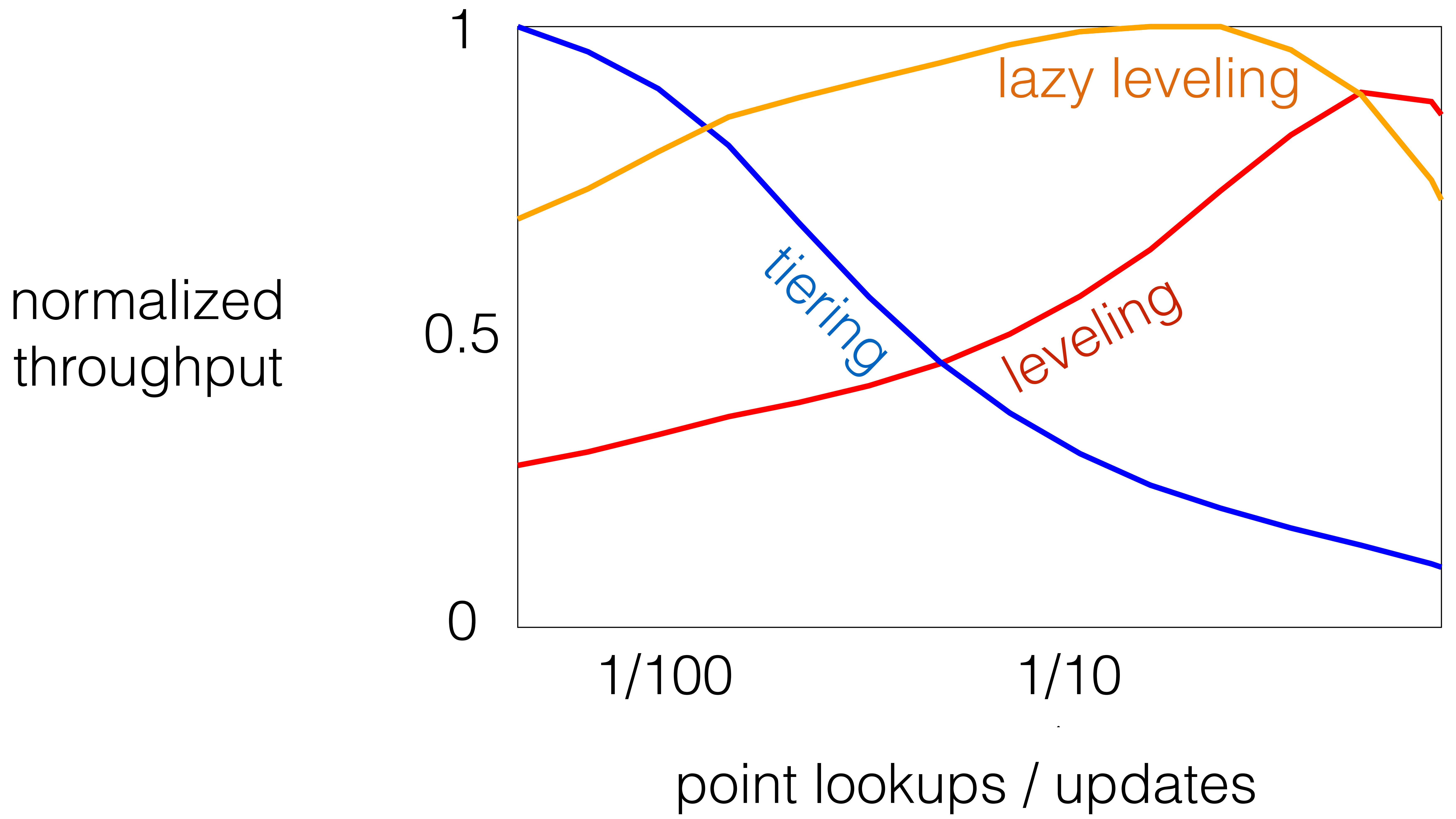
Leveling


normalized
throughput
1
0.5
0
1/100
1/10
point lookups / updates

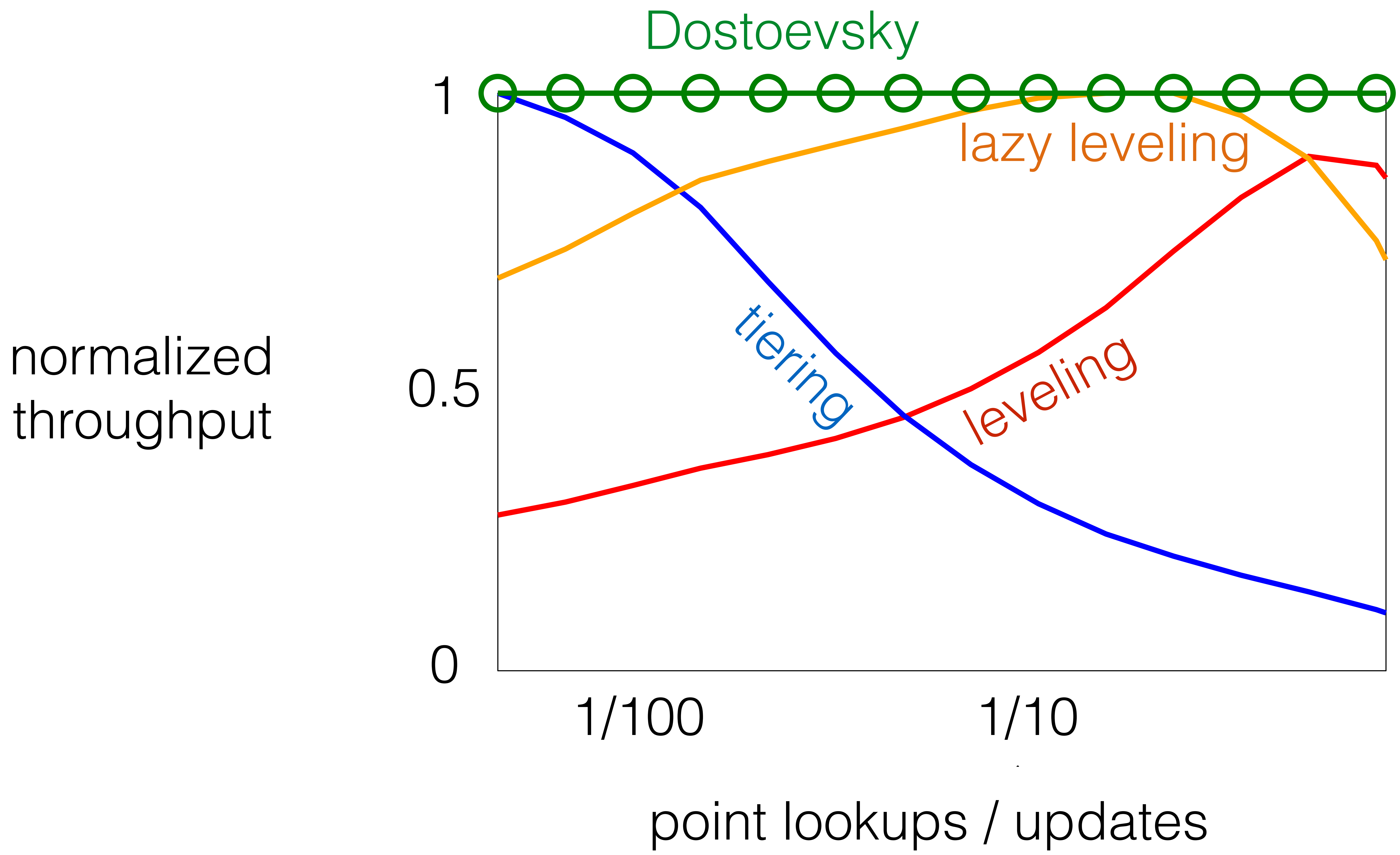

1
0.5
0
normalized throughput
leveling
1/100
1/10
point lookups / updates

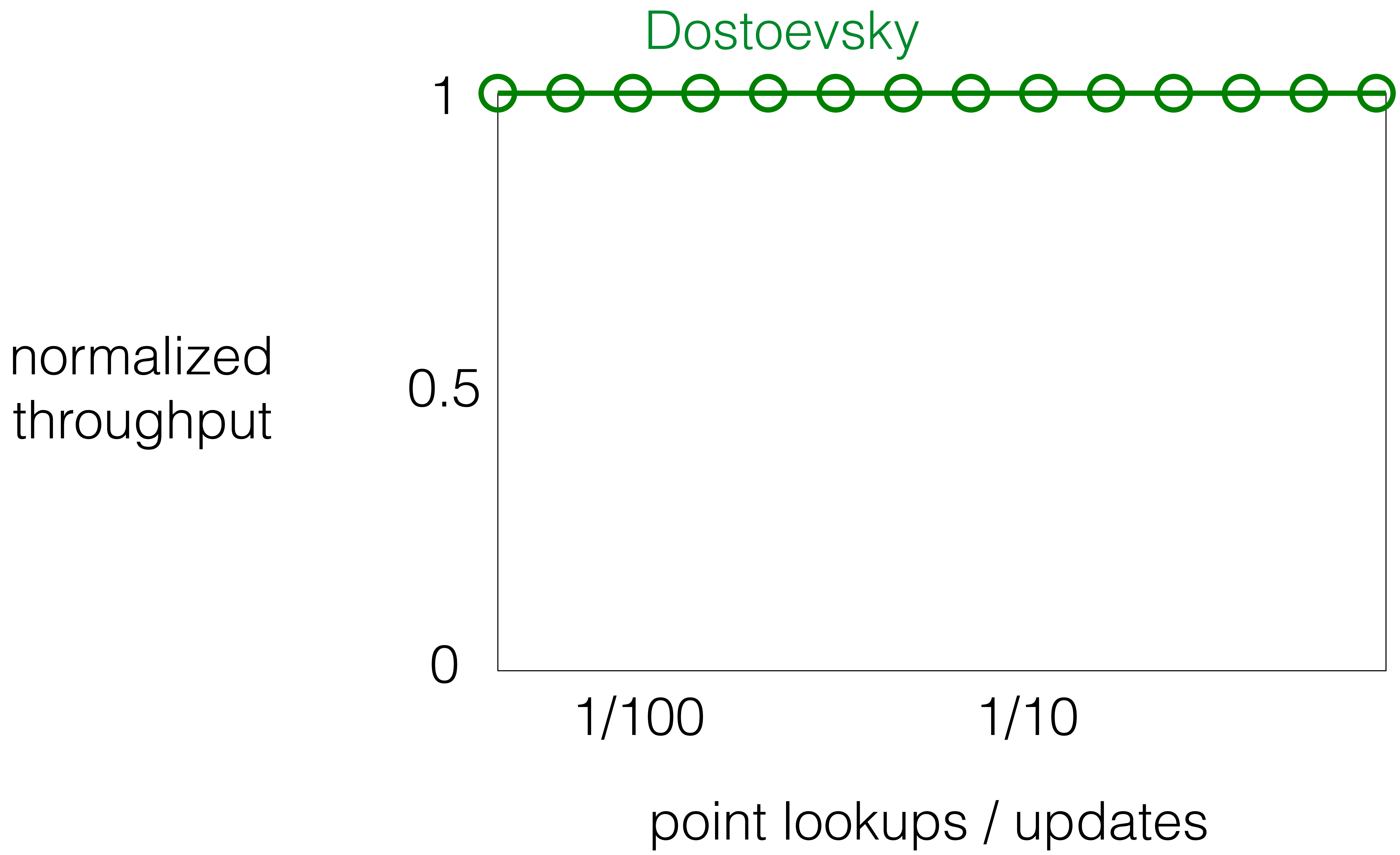


1
0.5
0
normalized
throughput
tiering
leveling
1/100
1/10
point lookups / updates

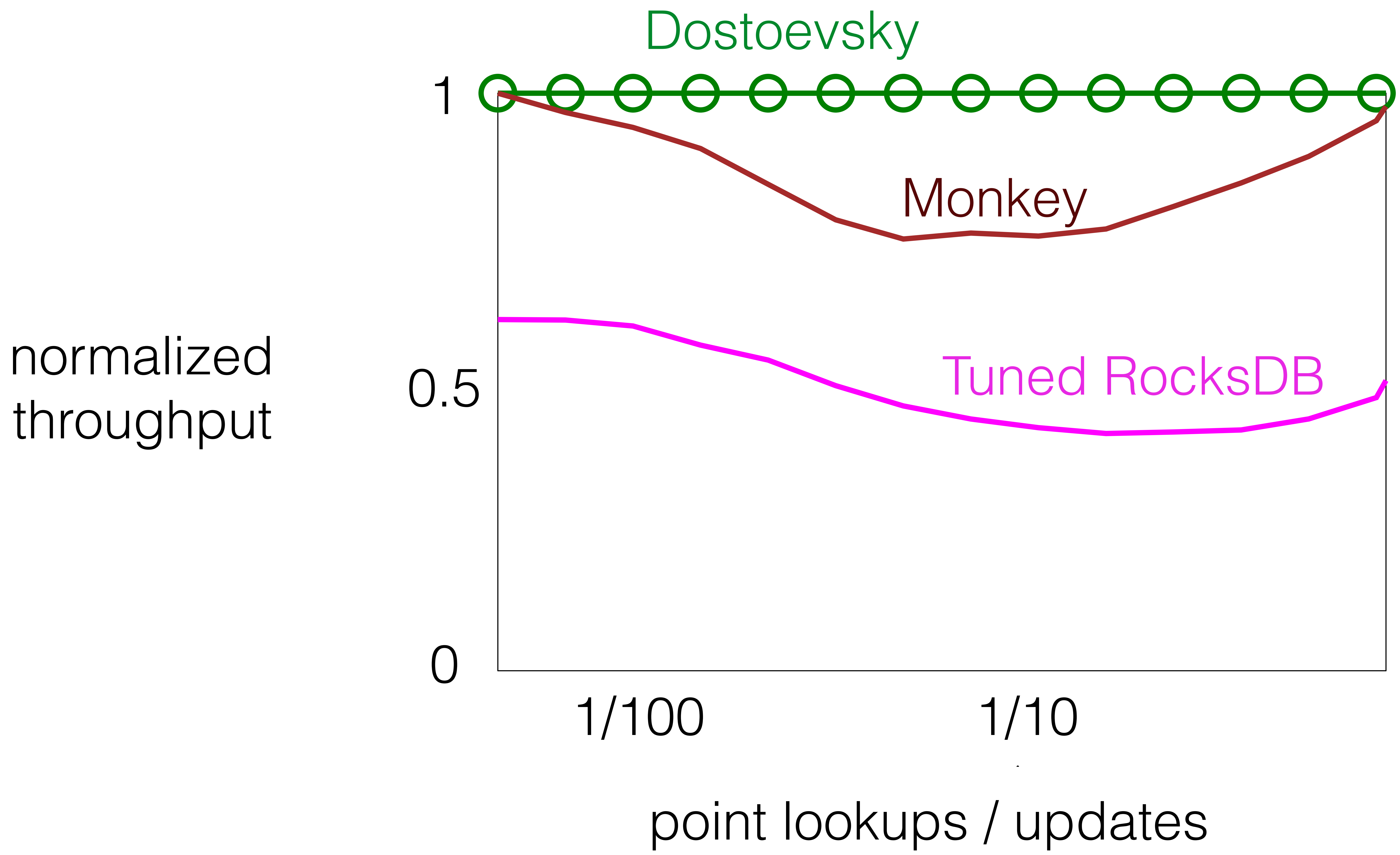


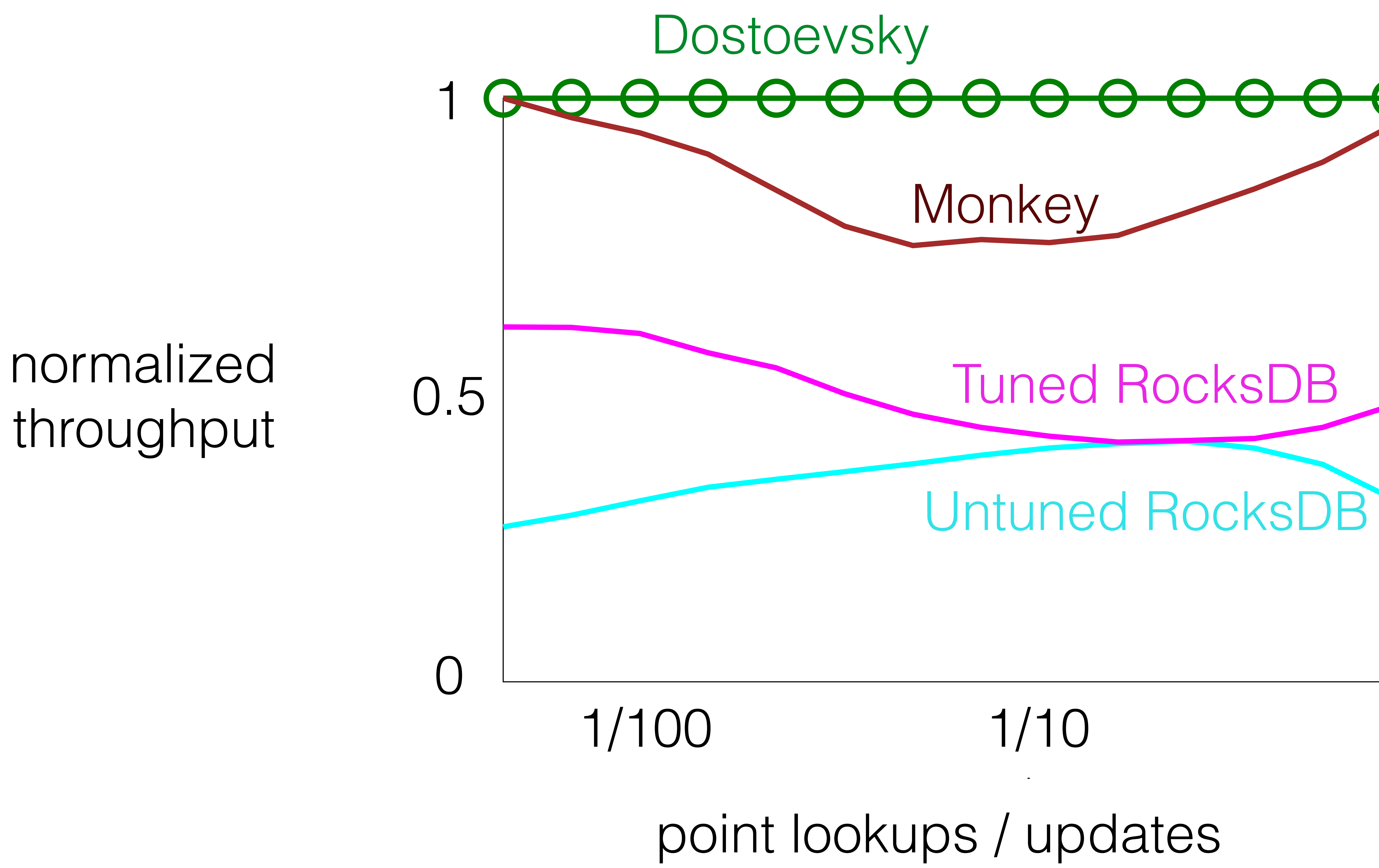
normalized
throughput
1
0.5
0
lazy leveling
tiering
leveling
1/100
1/10
point lookups / updates


Dostoevsky
1
0.5
0
normalized
throughput
lazy leveling
tiering
leveling
1/100
1/10
point lookups / updates



Dostoevsky
1
0.5
0
normalized
throughput
1/100
1/10
point lookups / updates


Dostoevsky
Monkey
Tuned RocksDB
normalized
throughput
1
0.5
0
1/100
1/10
point lookups / updates

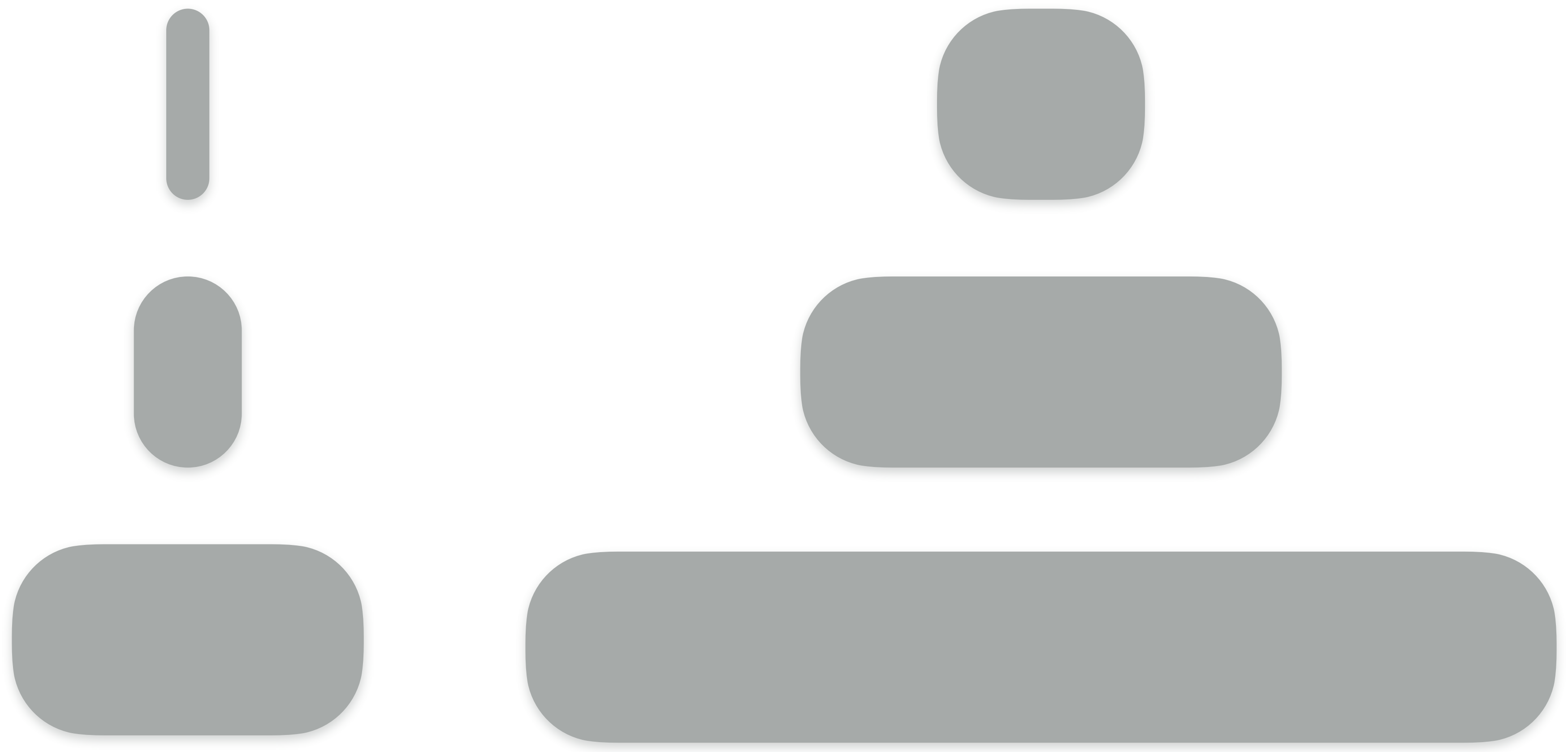

Dostoevsky
1
Monkey
normalized
throughput
0.5
Tuned RocksDB
Untuned RocksDB
0
1/100
1/10
point lookups / updates

Conclusion

Conclusion

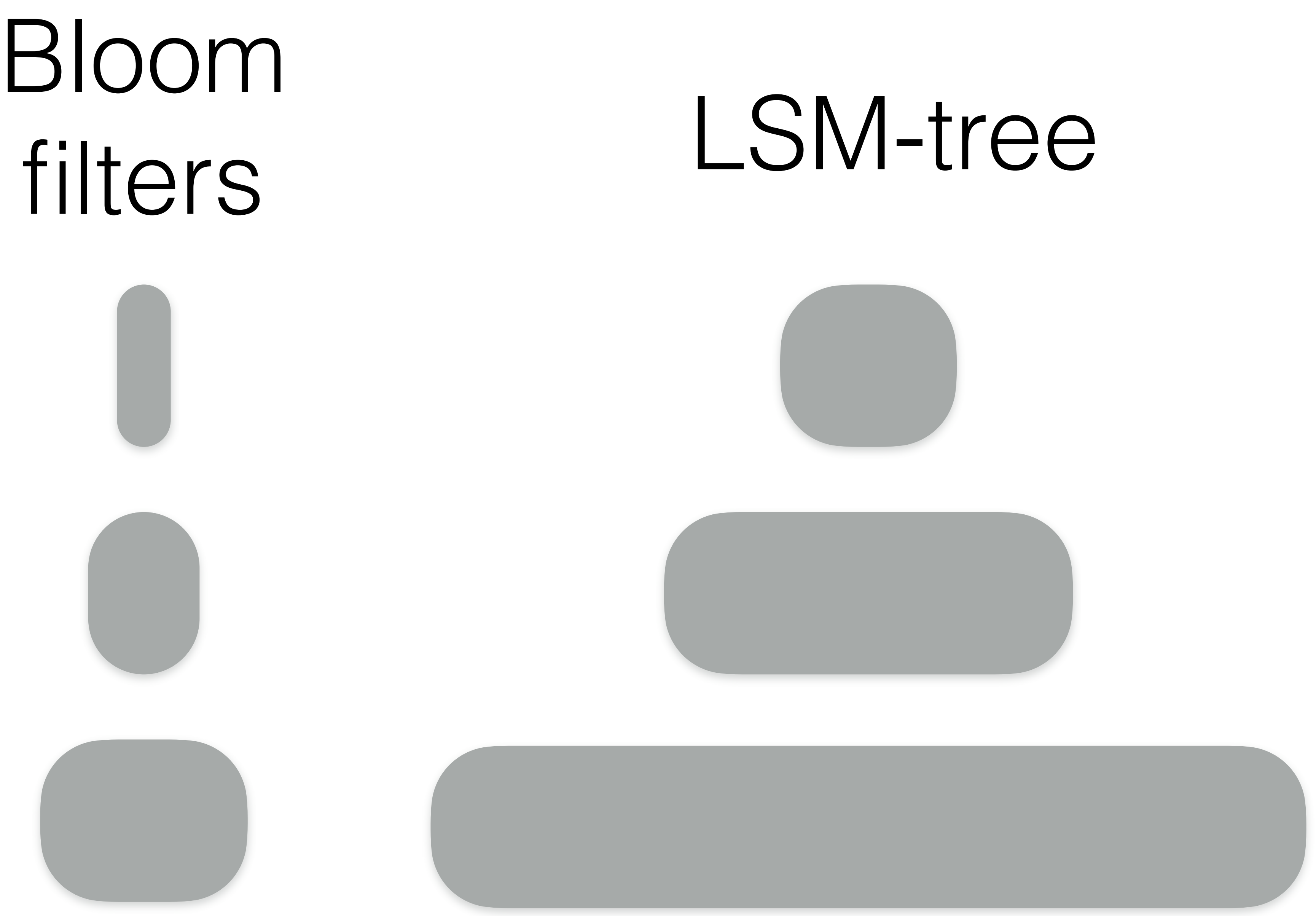
Bloom filters

LSM-tree



Conclusion

optimizes memory allocation



Conclusion

optimizes memory allocation

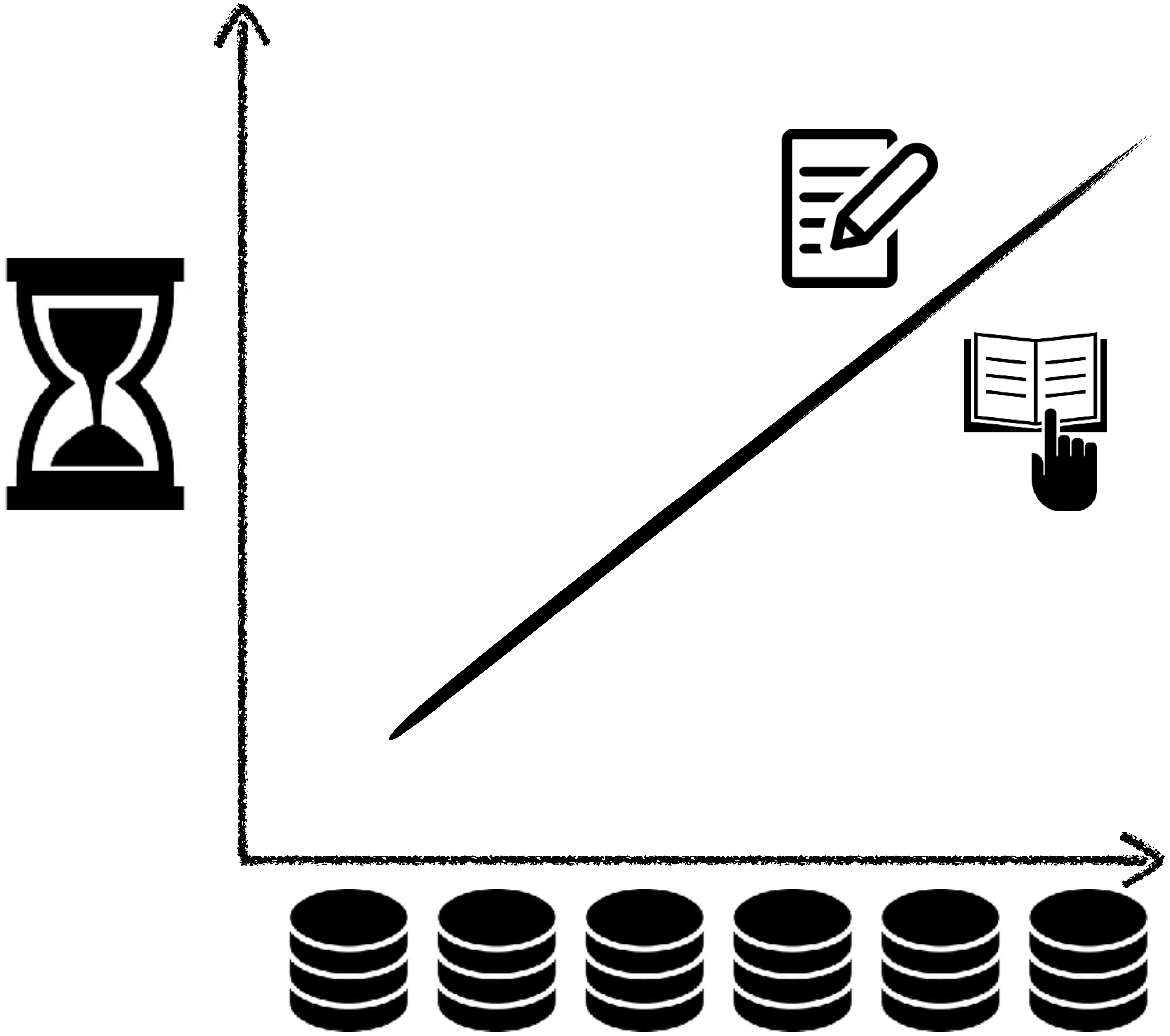
Bloom filters

LSM-tree

removes superfluous merging

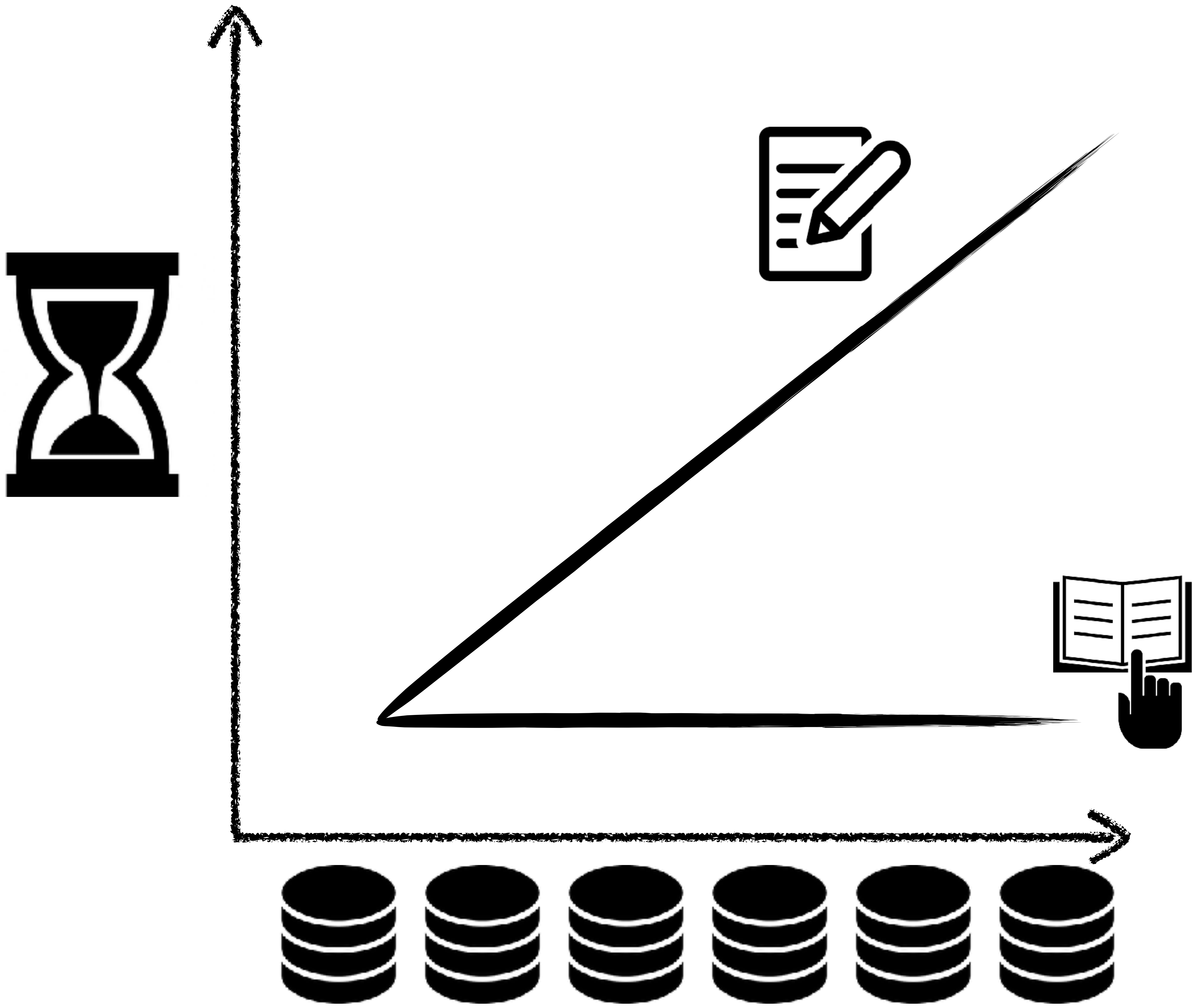
DASlab @ Harvard SEAS

Conclusion

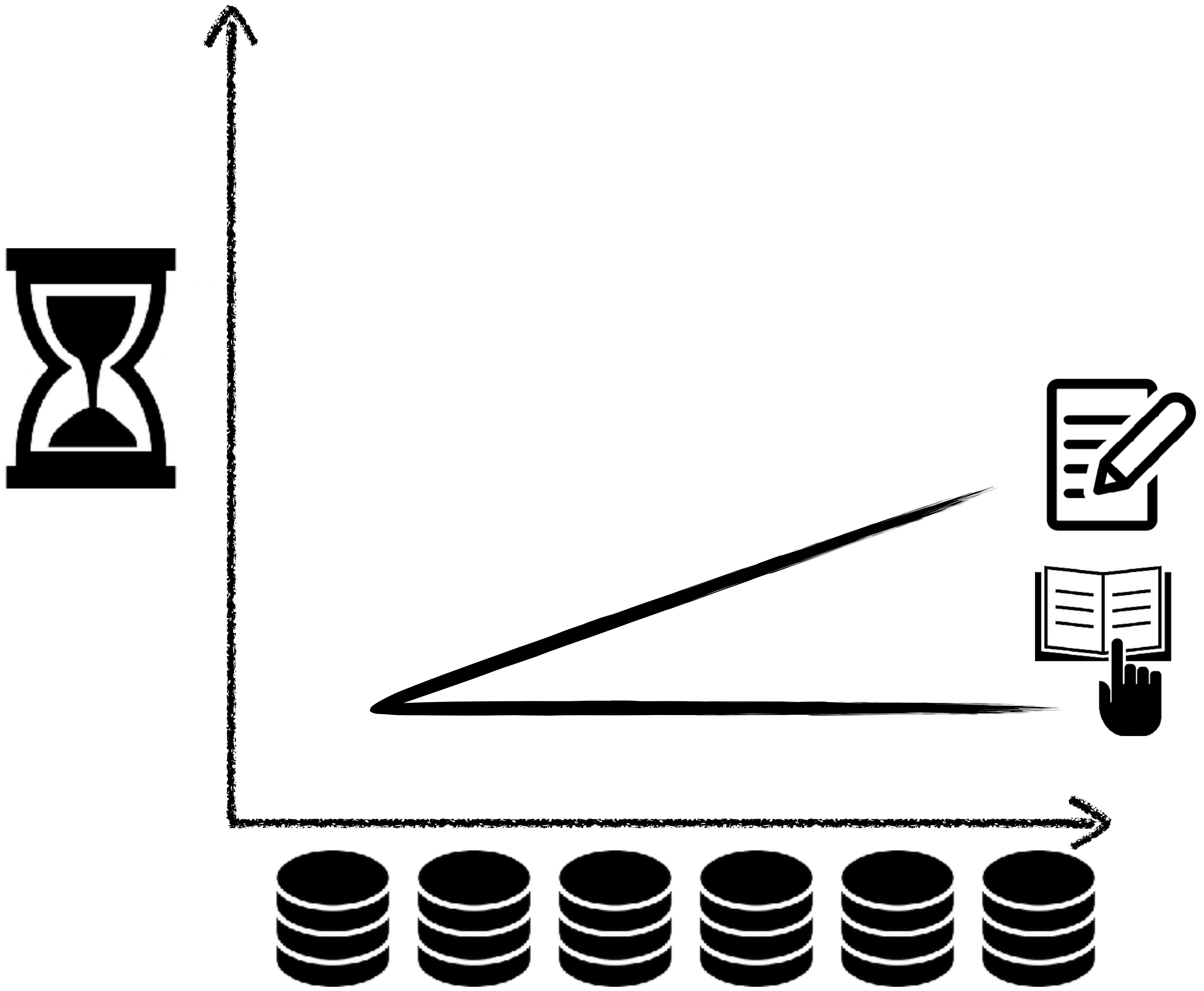


DASlab
@ Harvard SEAS

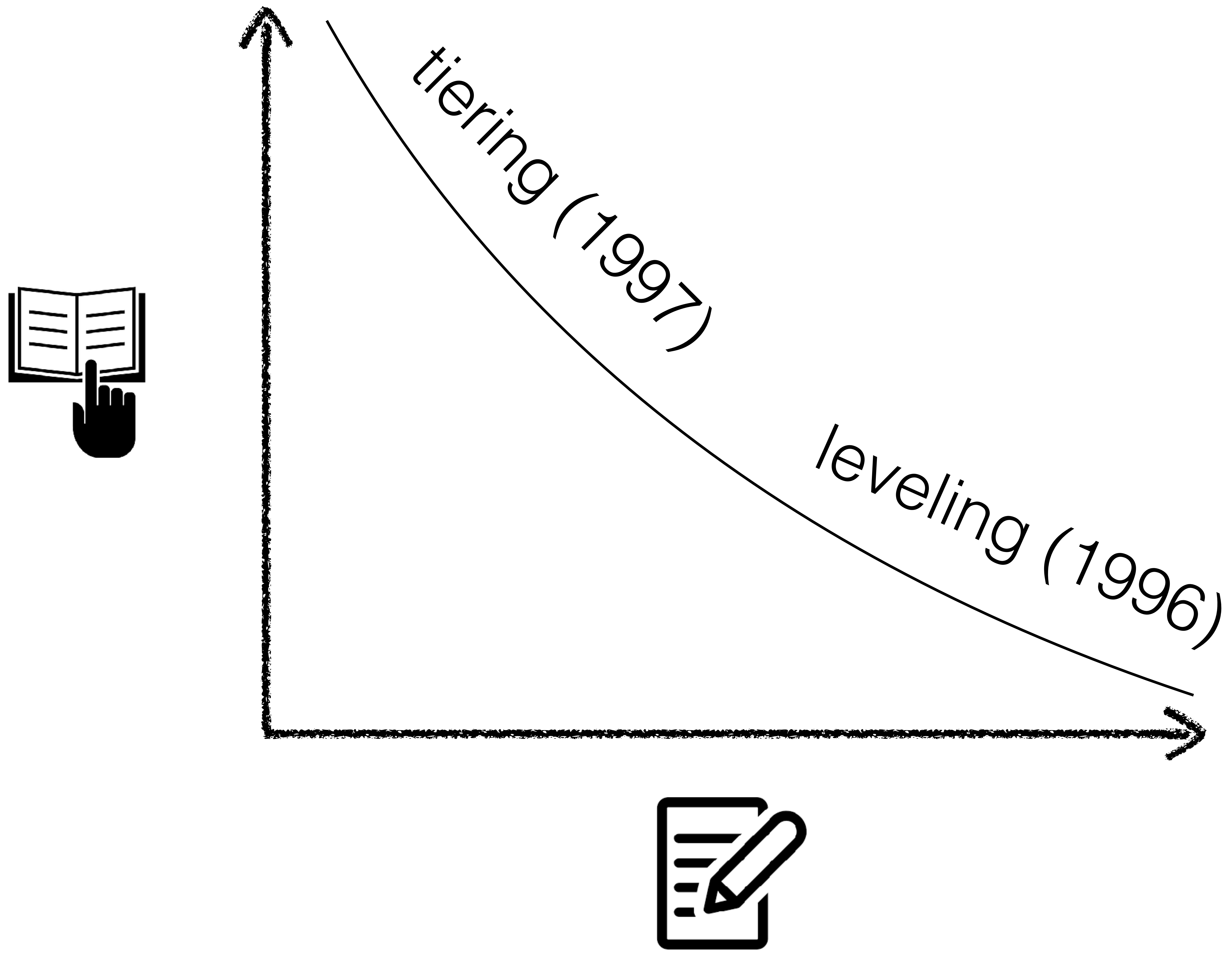
Conclusion



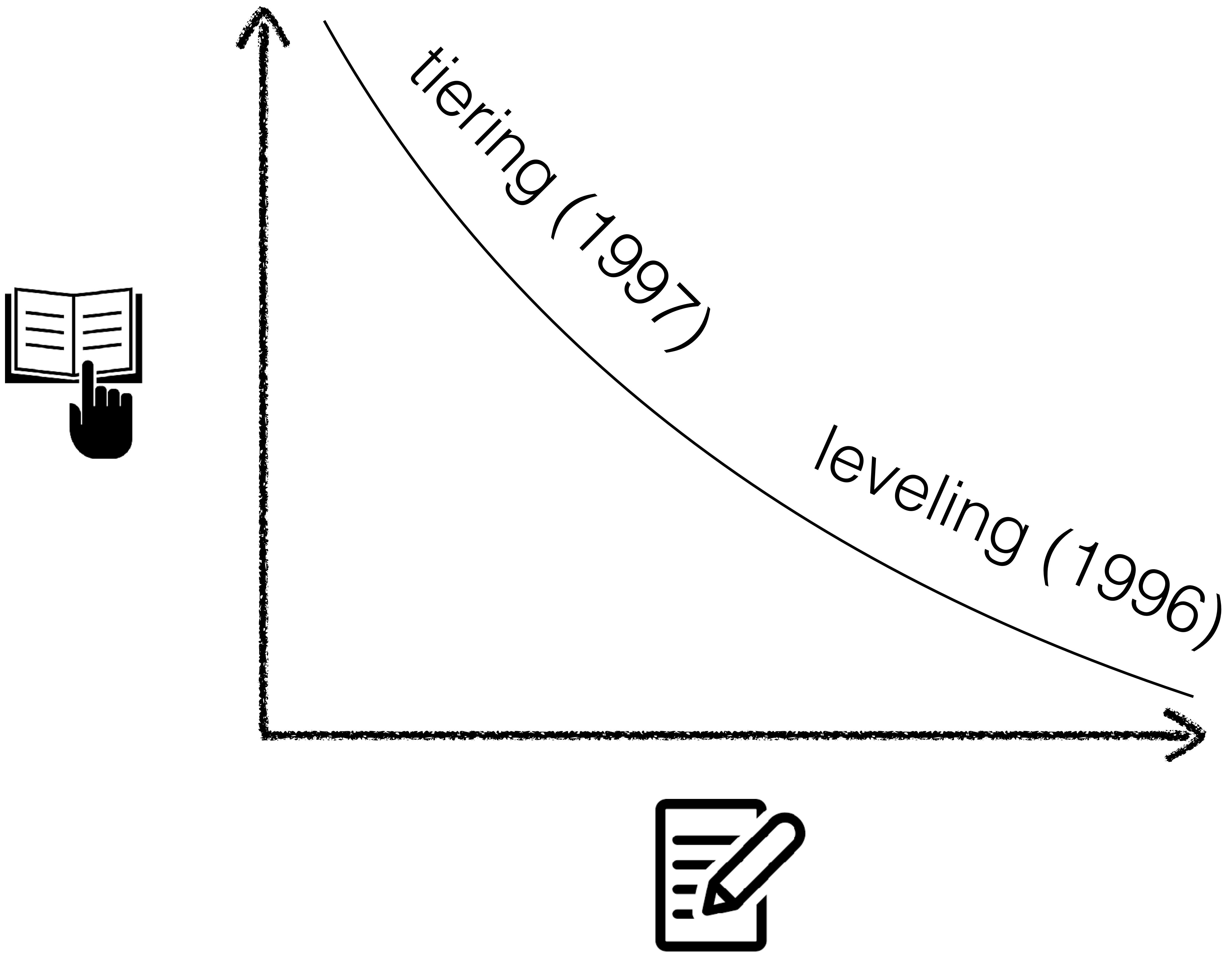
Conclusion



Conclusion

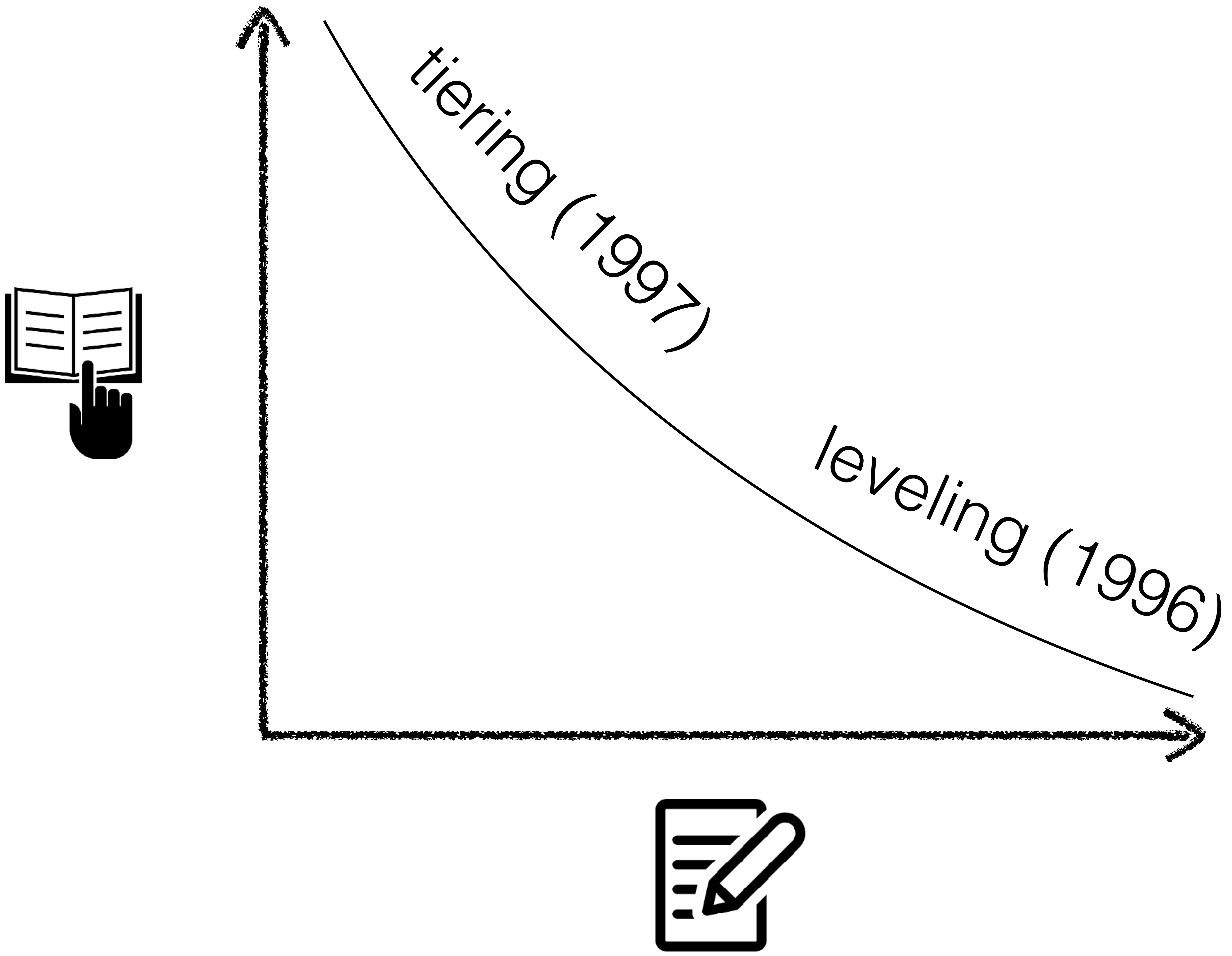


Conclusion



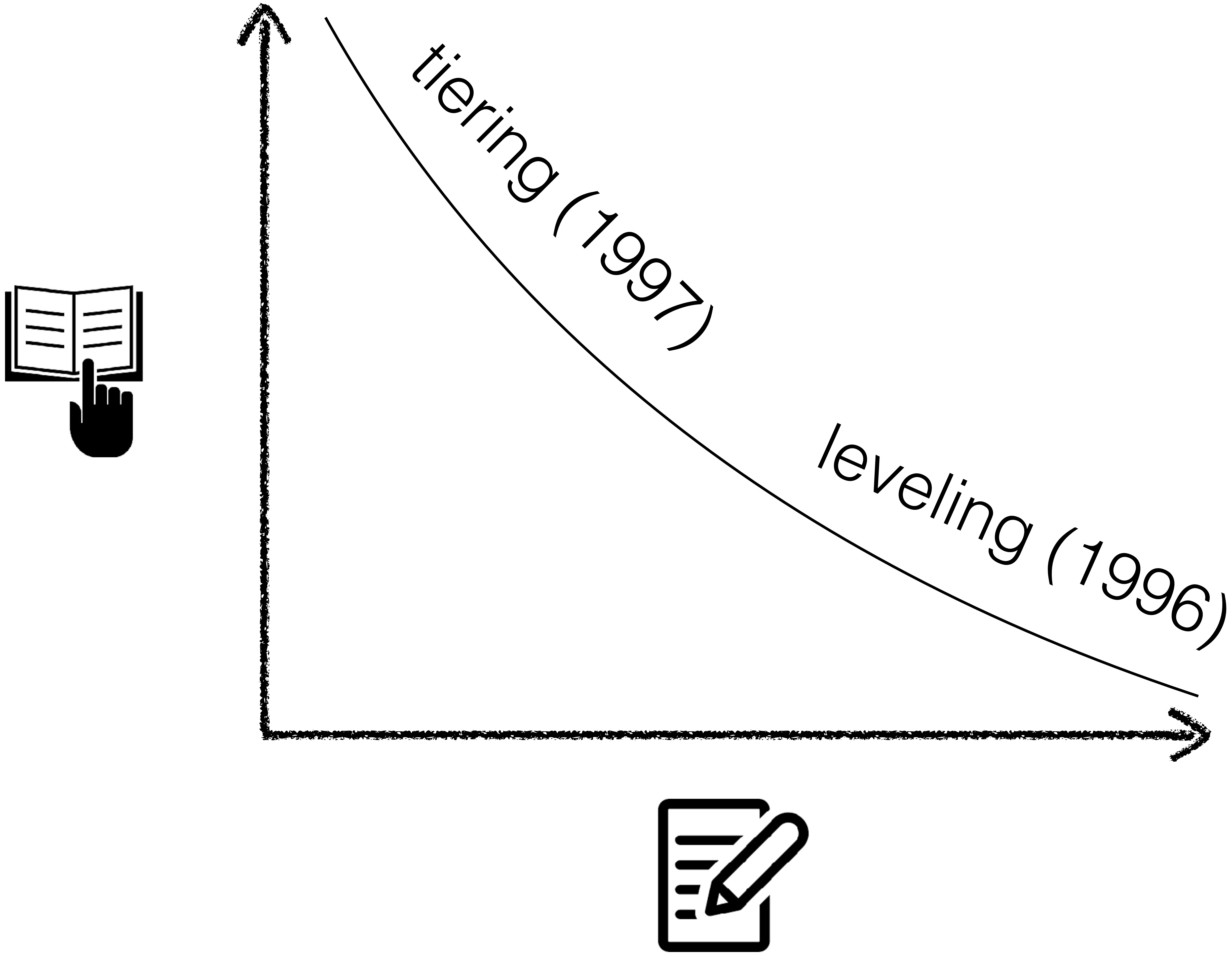
little memory

Conclusion



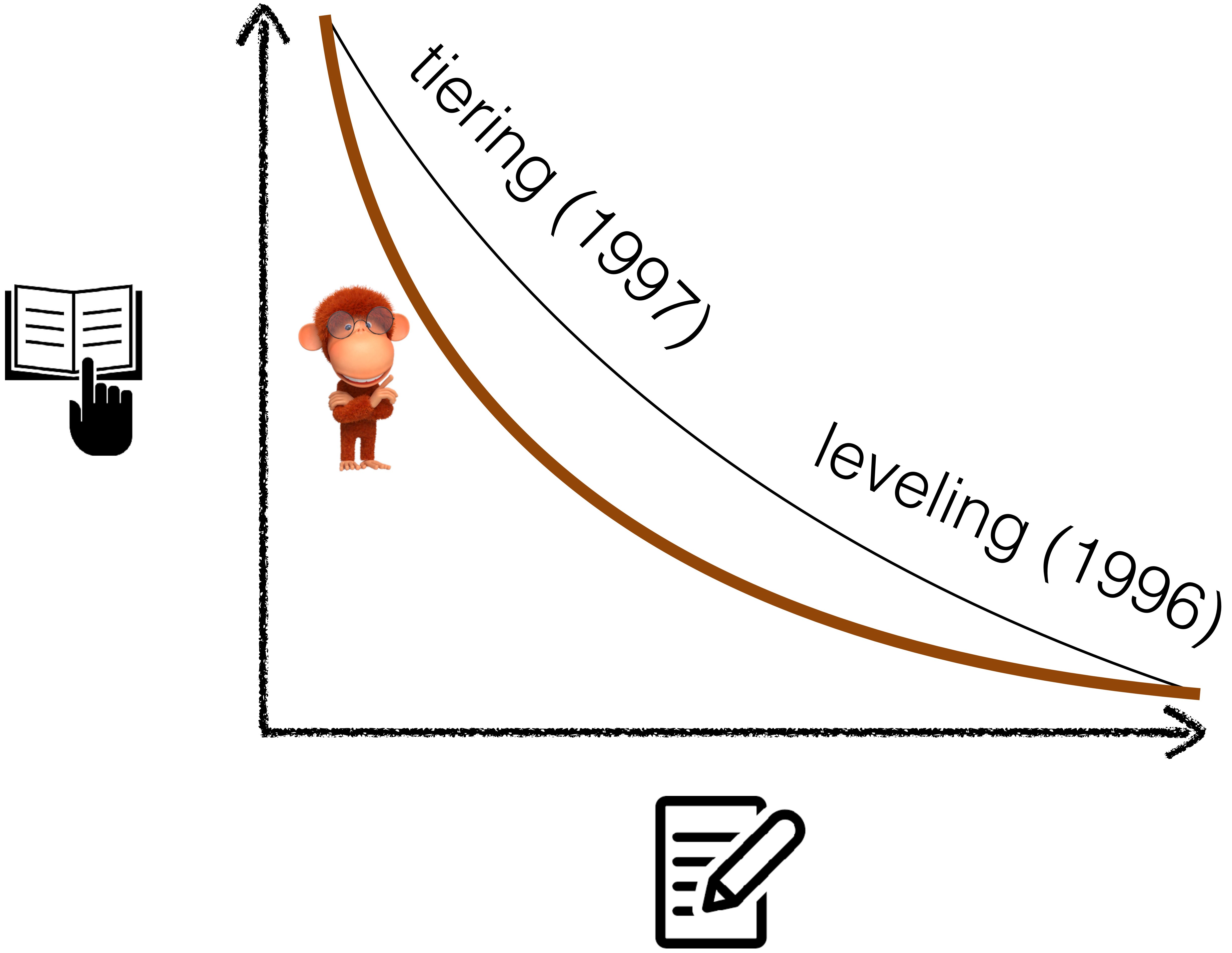
ample memory

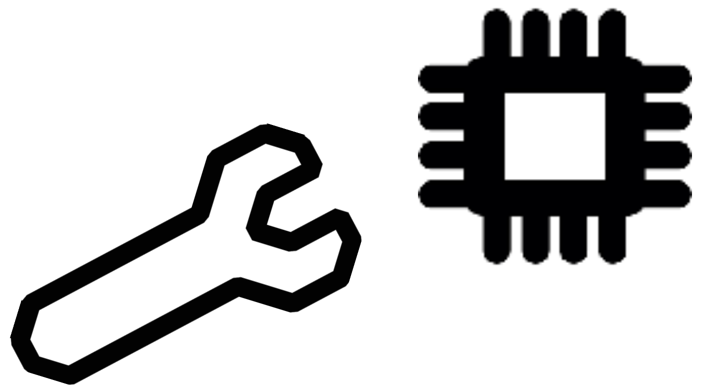
Conclusion



ample memory

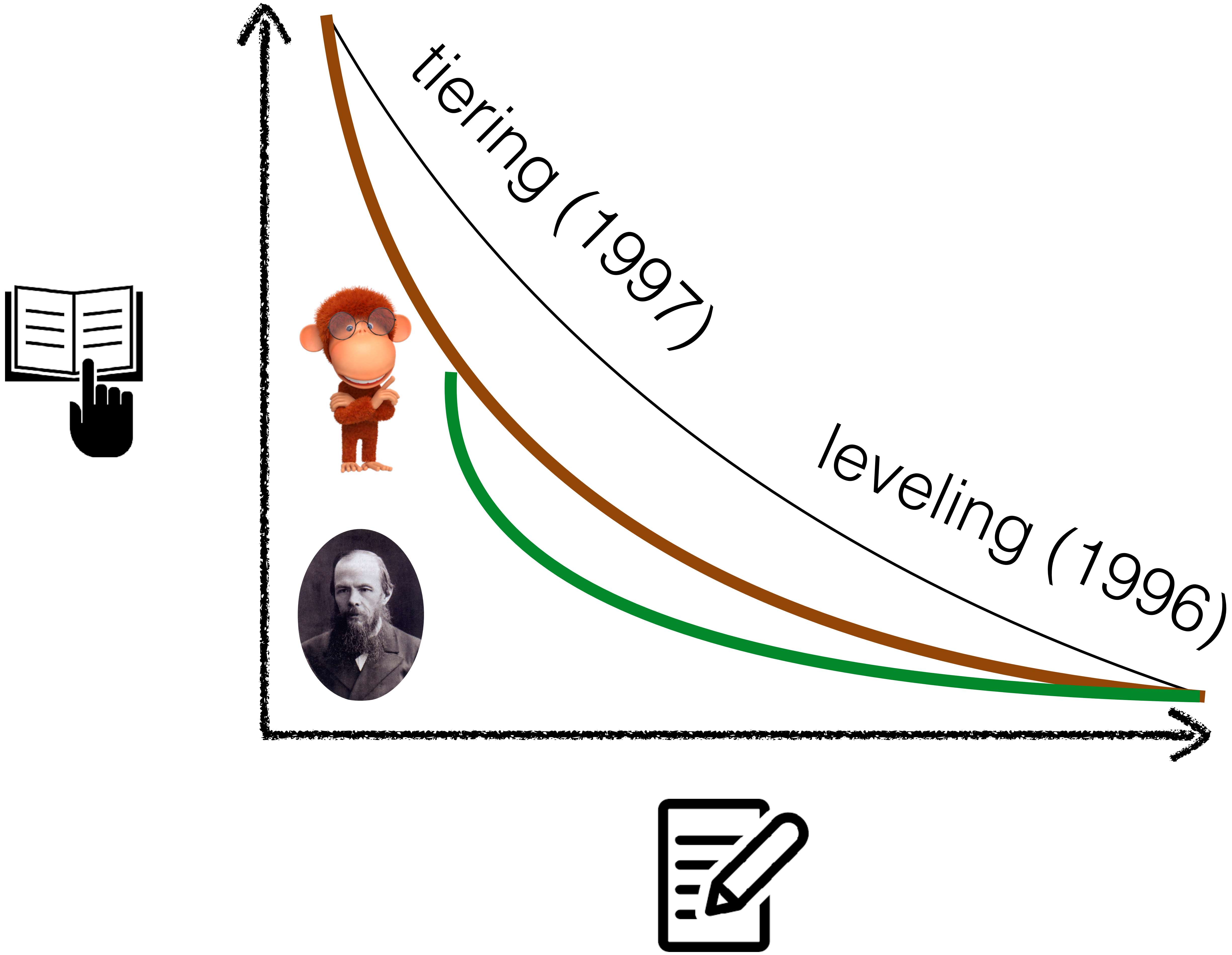
Conclusion

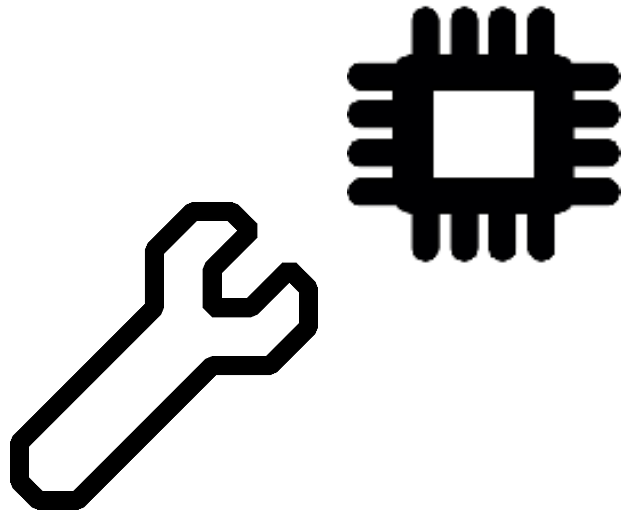




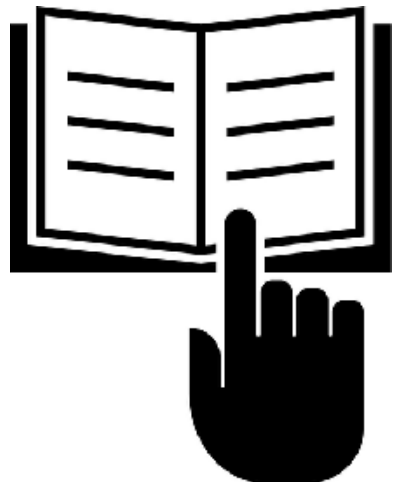
ample memory

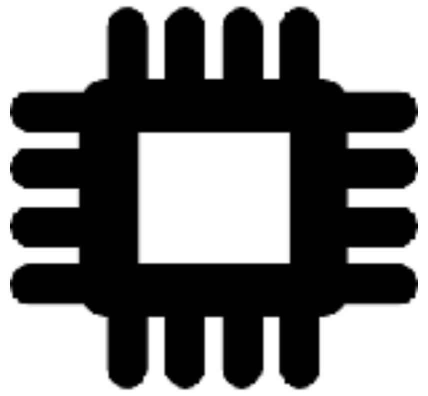
Conclusion



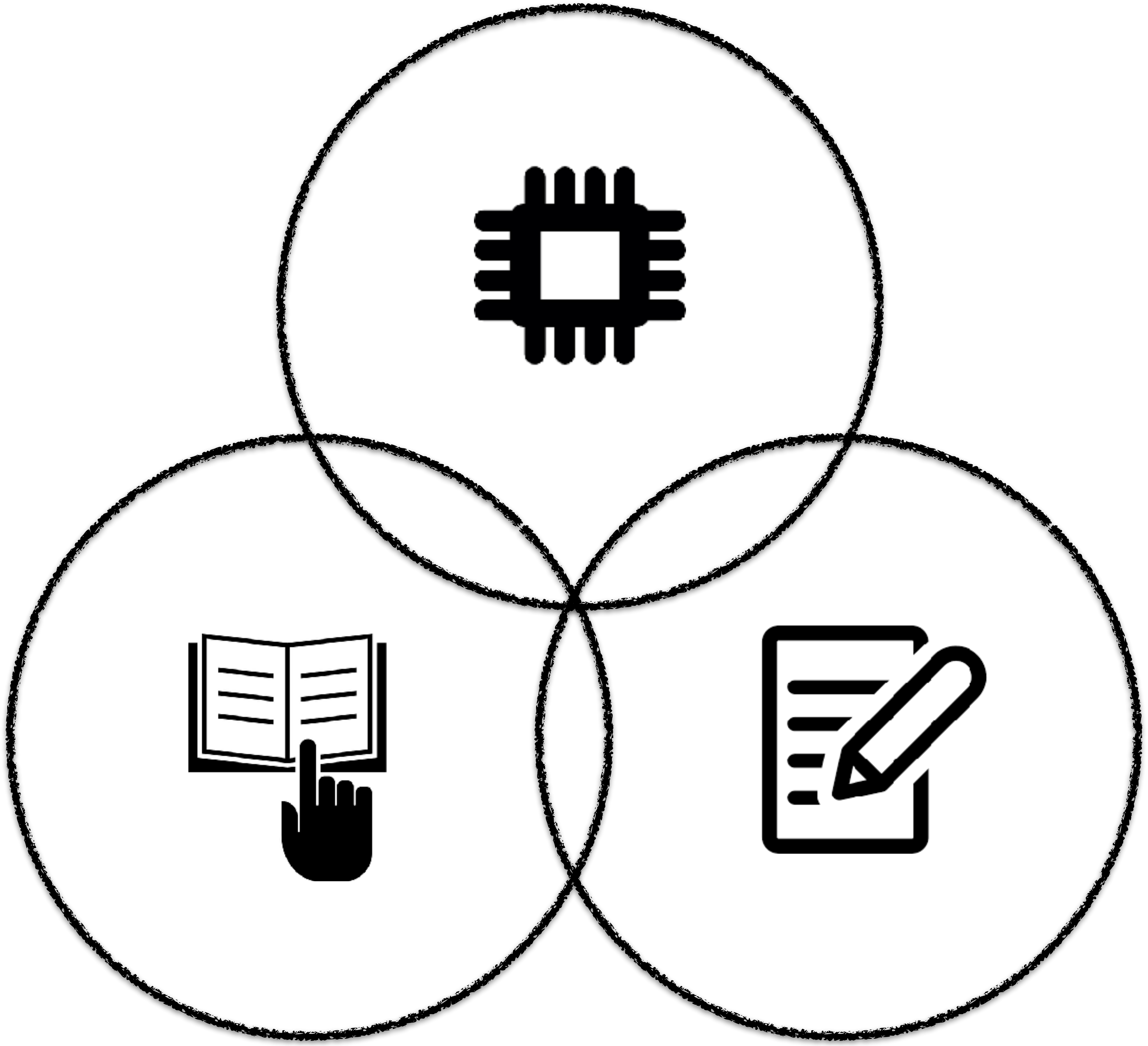


Conclusion

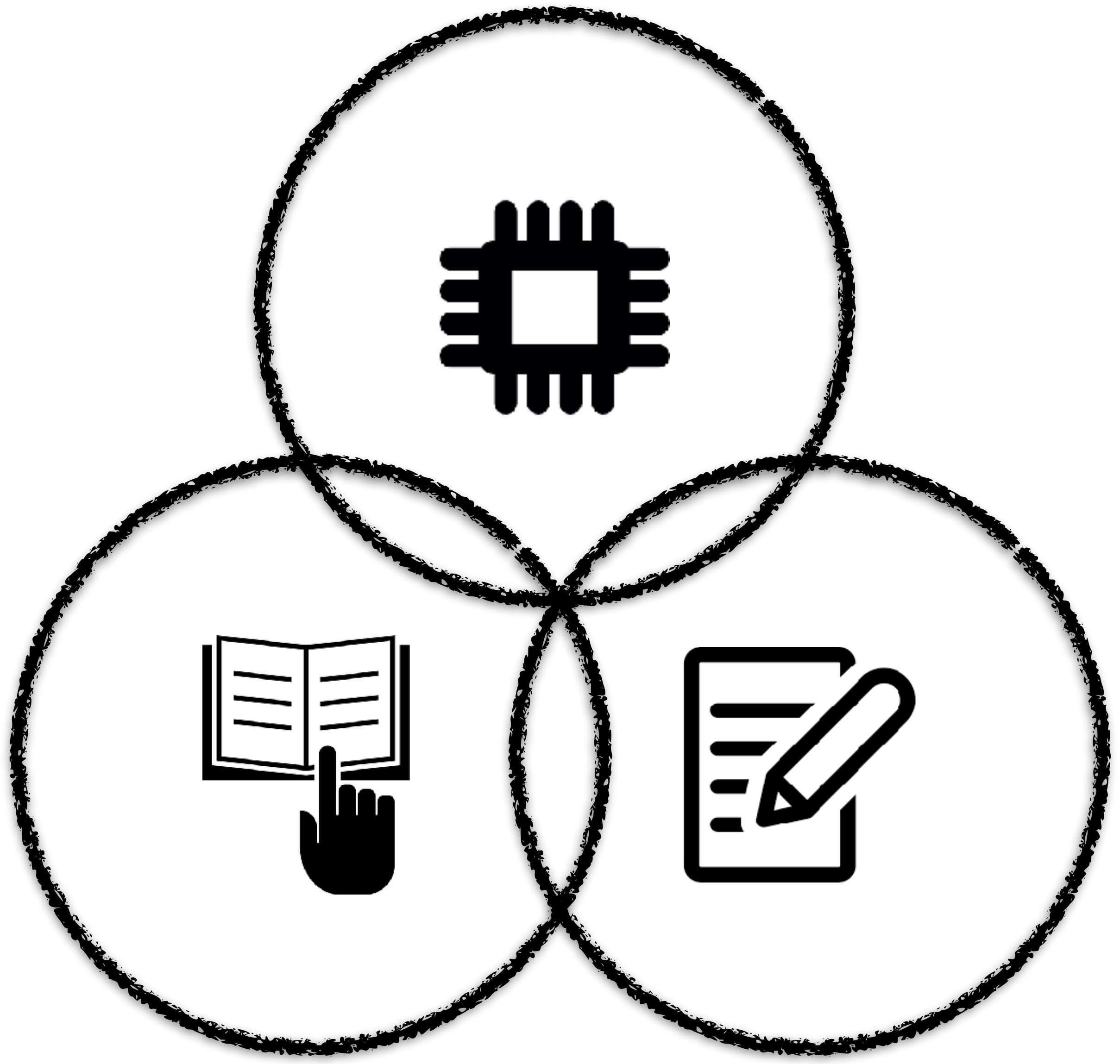




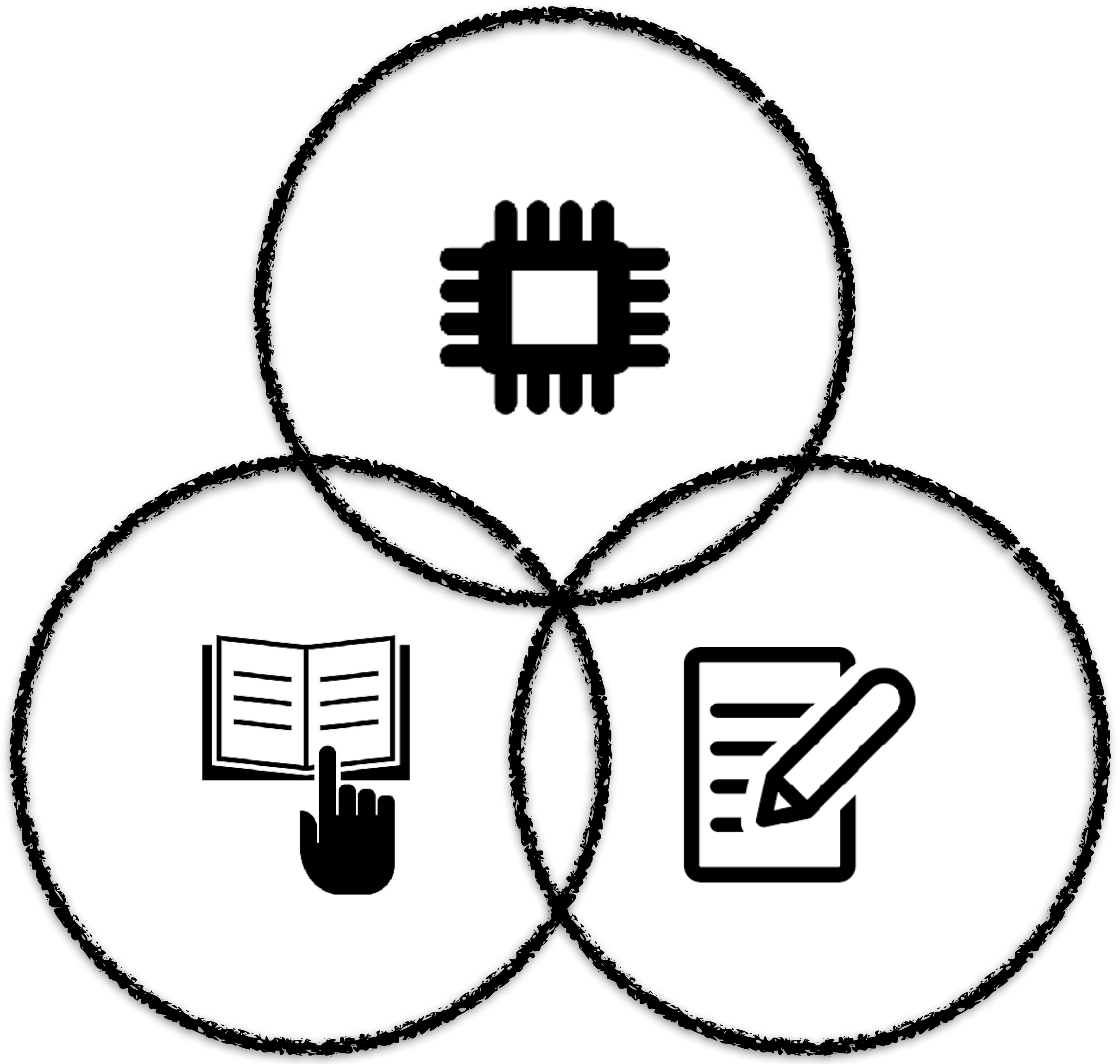
Conclusion



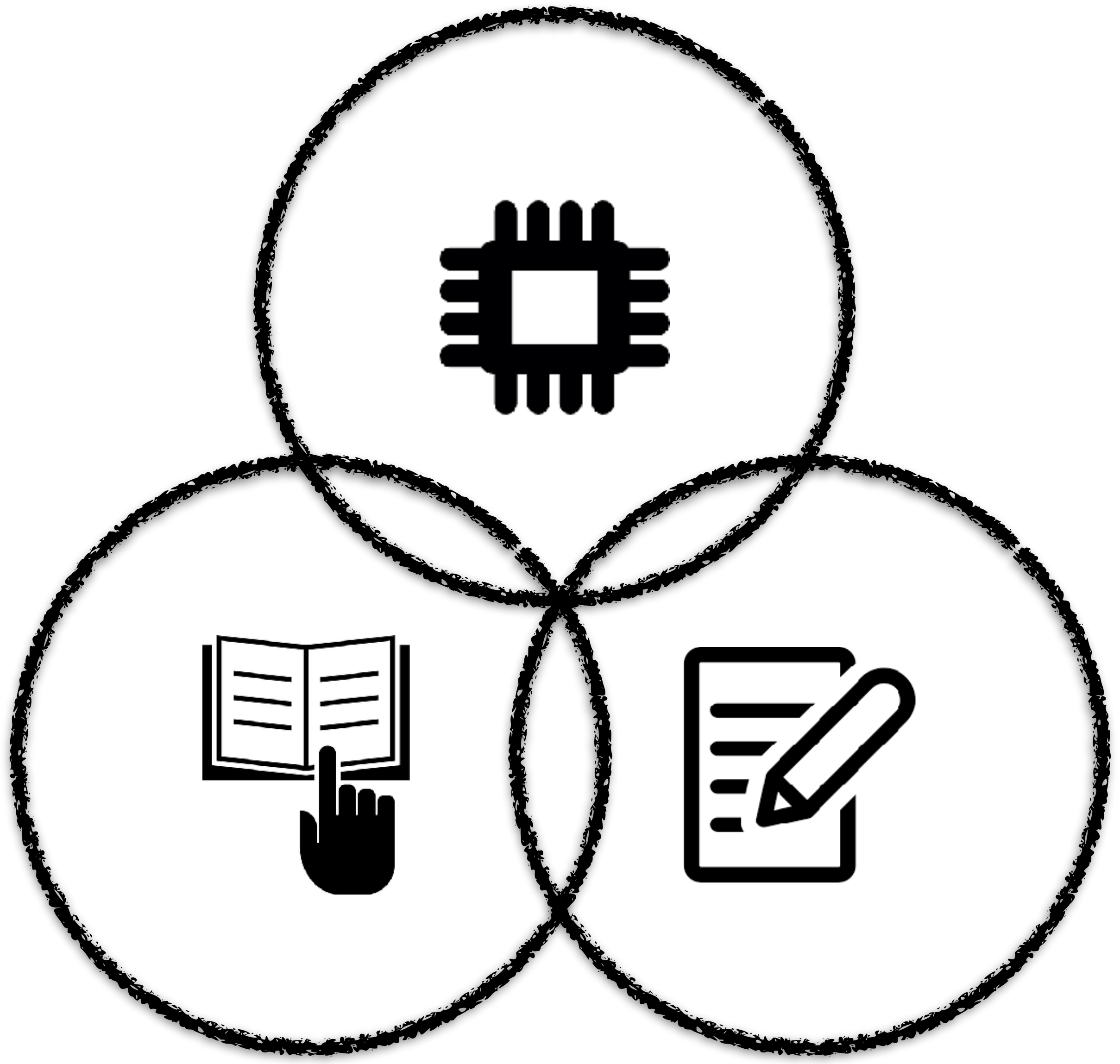
Conclusion



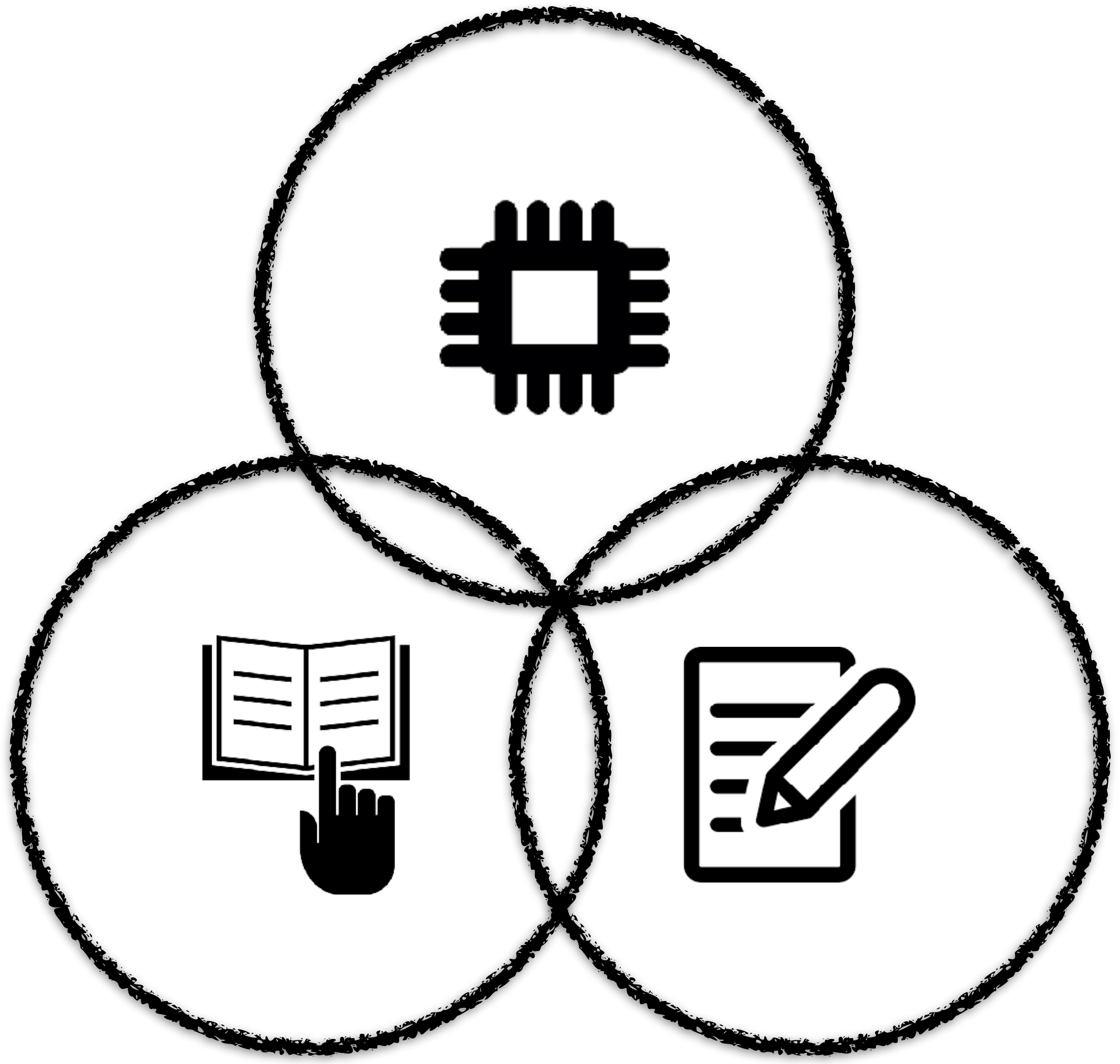
Conclusion



CrimsonDB



CrimsonDB



Thanks!