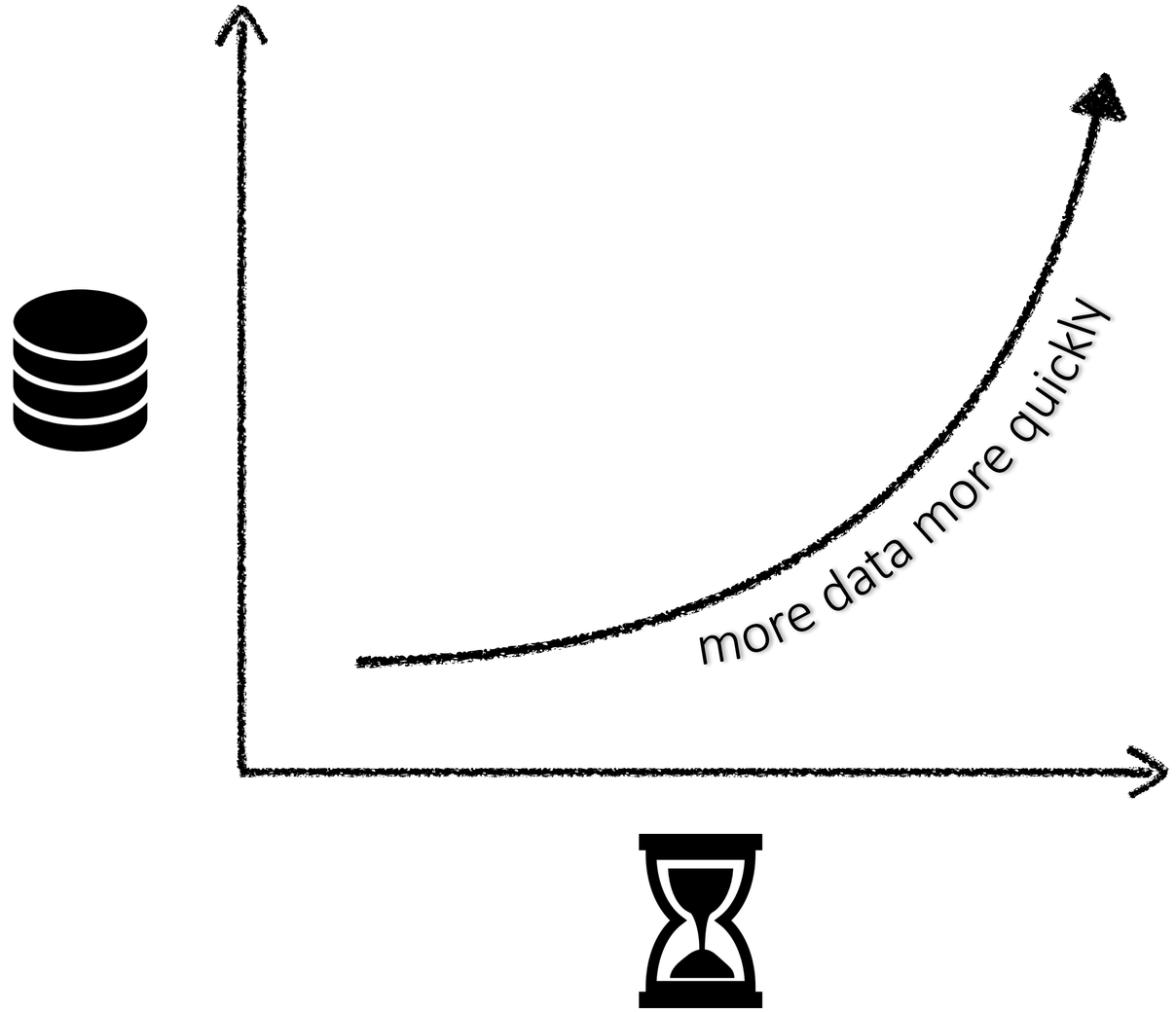
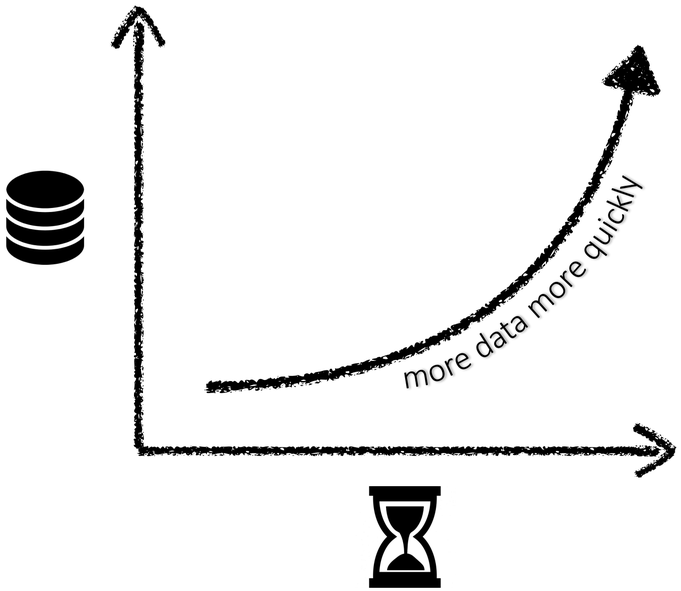
The Log-Structured Merge-Bush & the Wacky Continuum

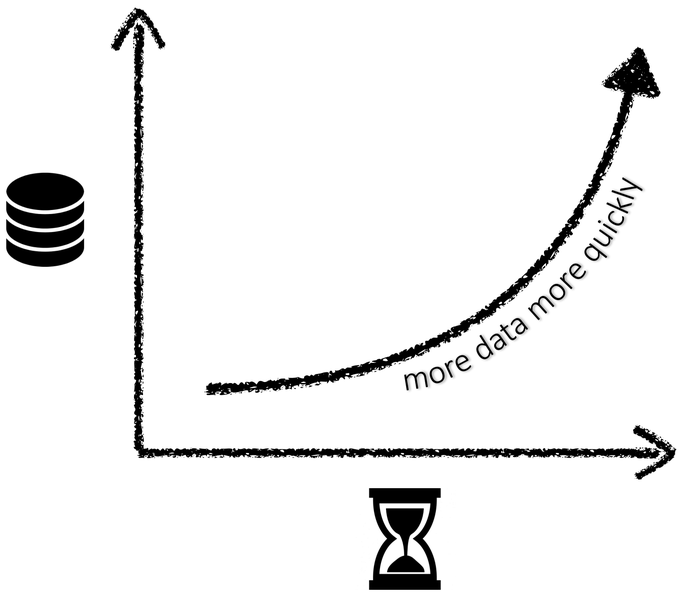
DASlab
@ Harvard SEAS

Niv Dayan &
Stratos Idreos


more data more quickly


more data more quickly


LSM-tree

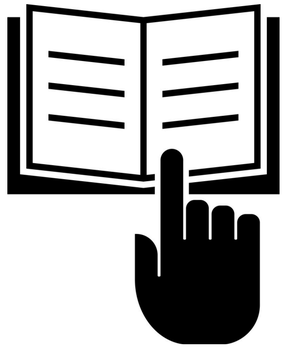

more data more quickly


LSM-tree

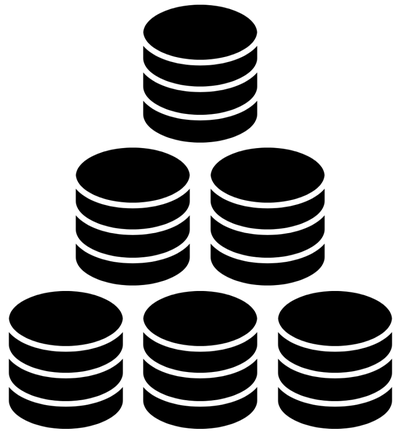

RocksDB
Cassandra
HBase
…
Couchbase

fast writes

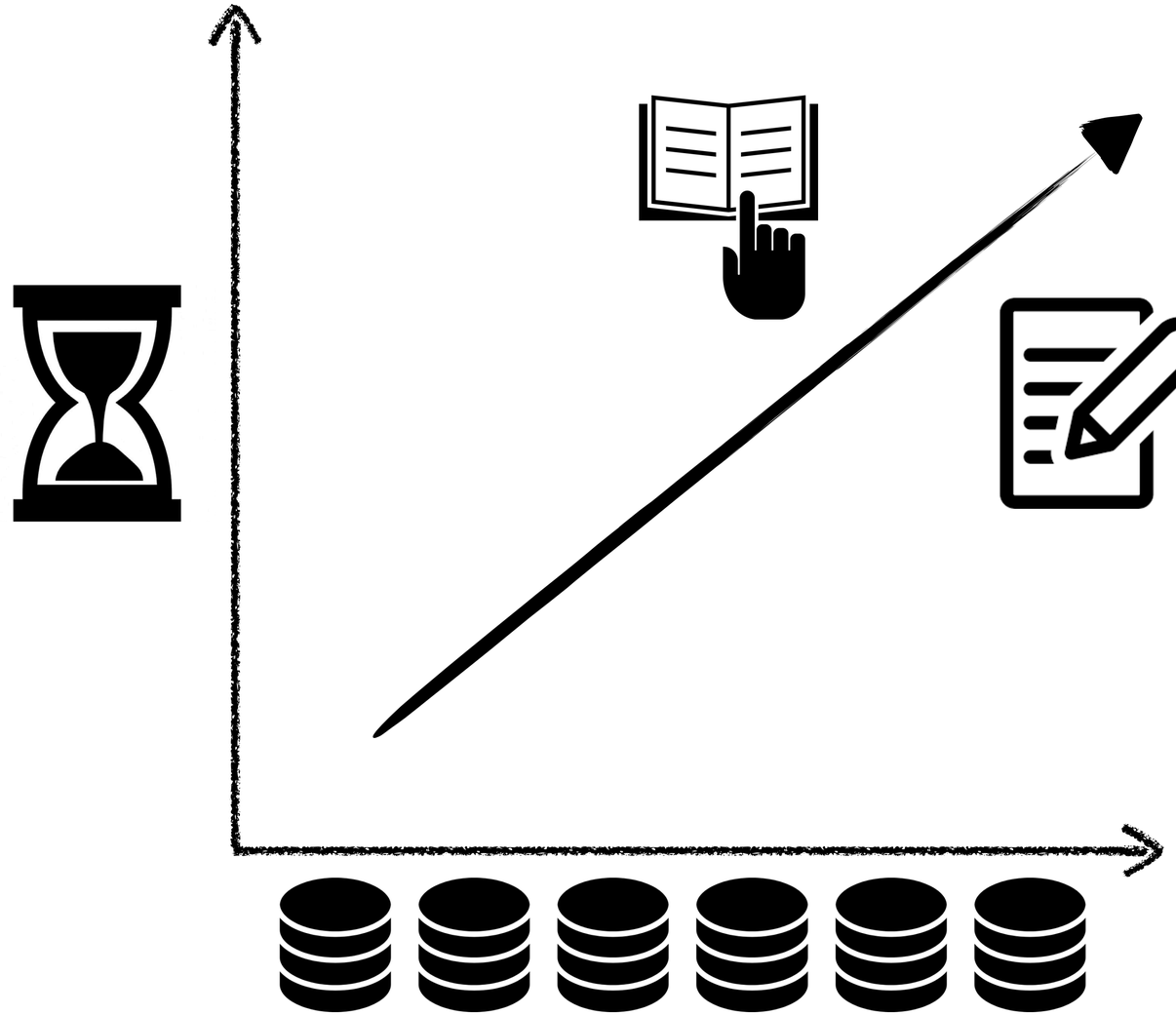
fast writes

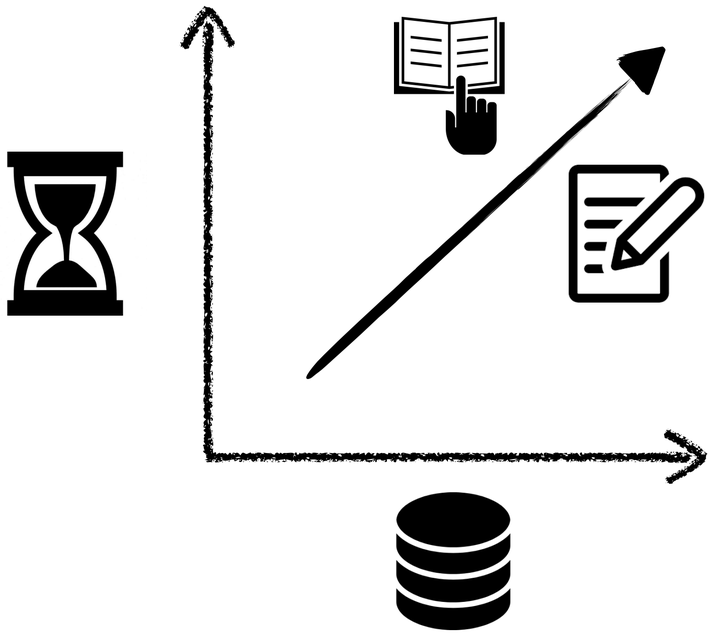


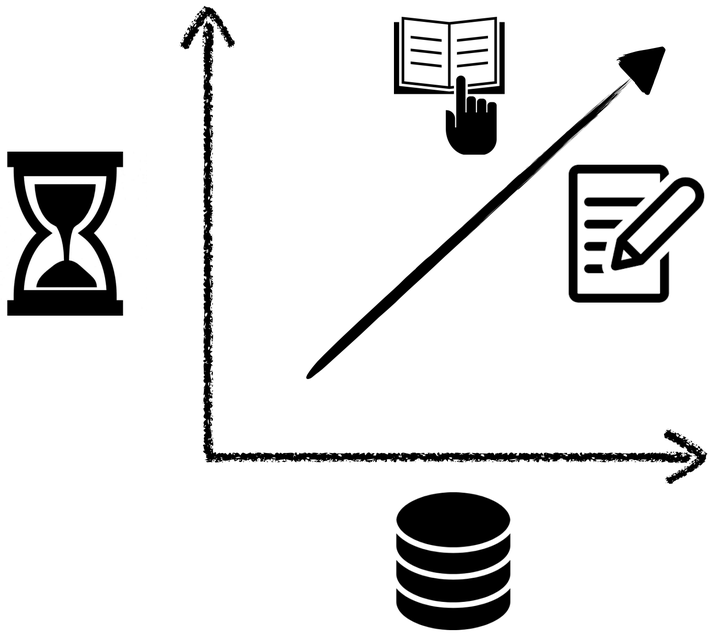
fast reads



massive data





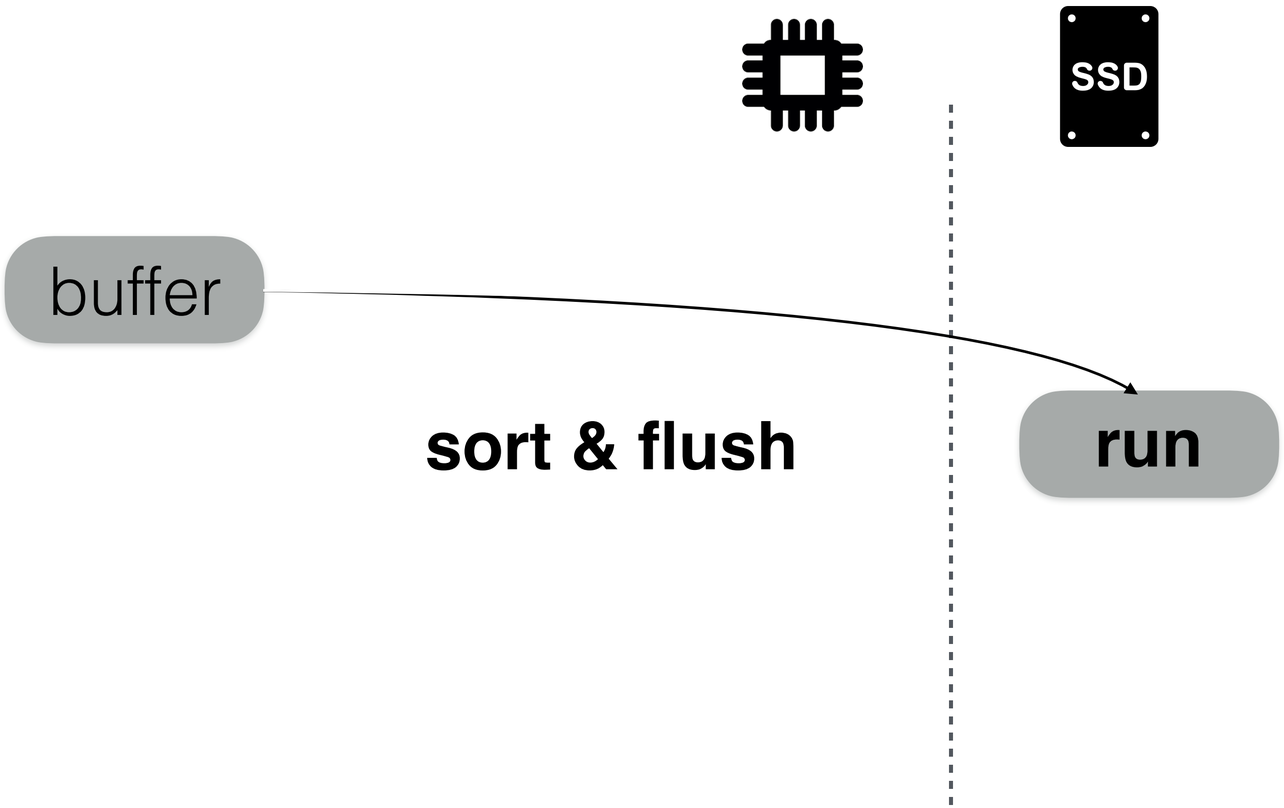


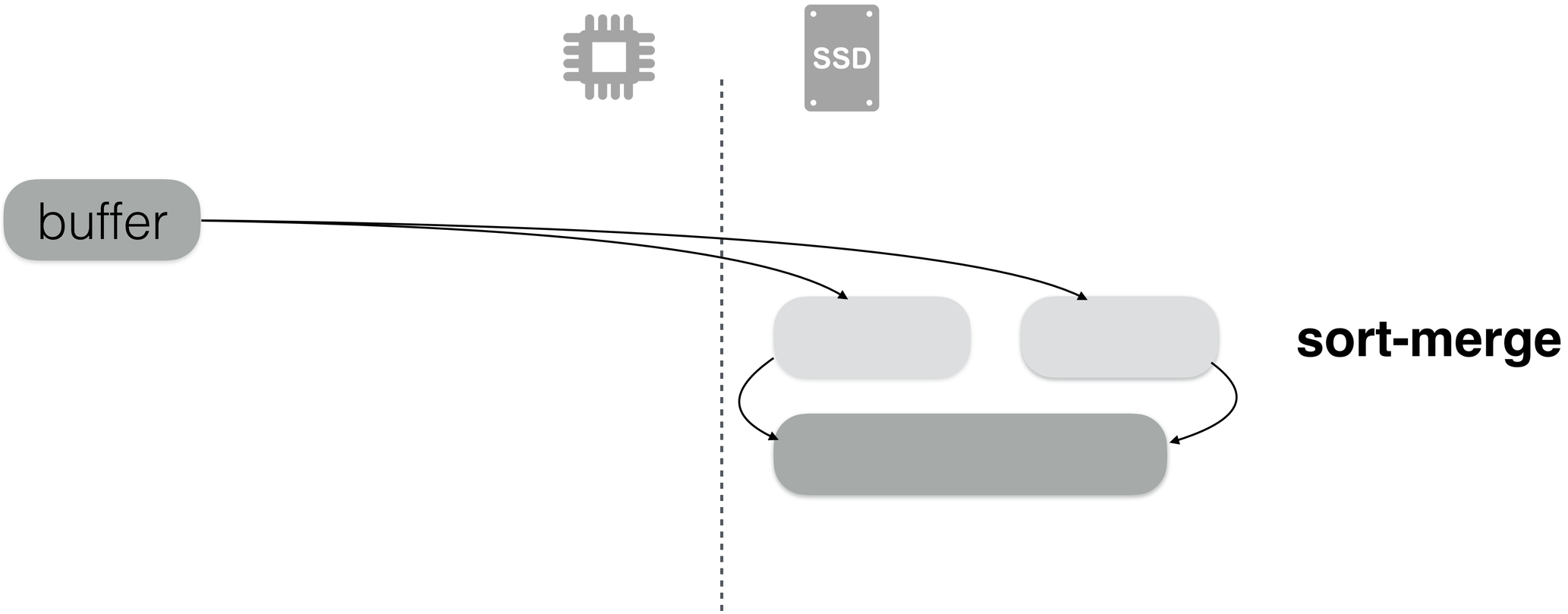
Background

writes

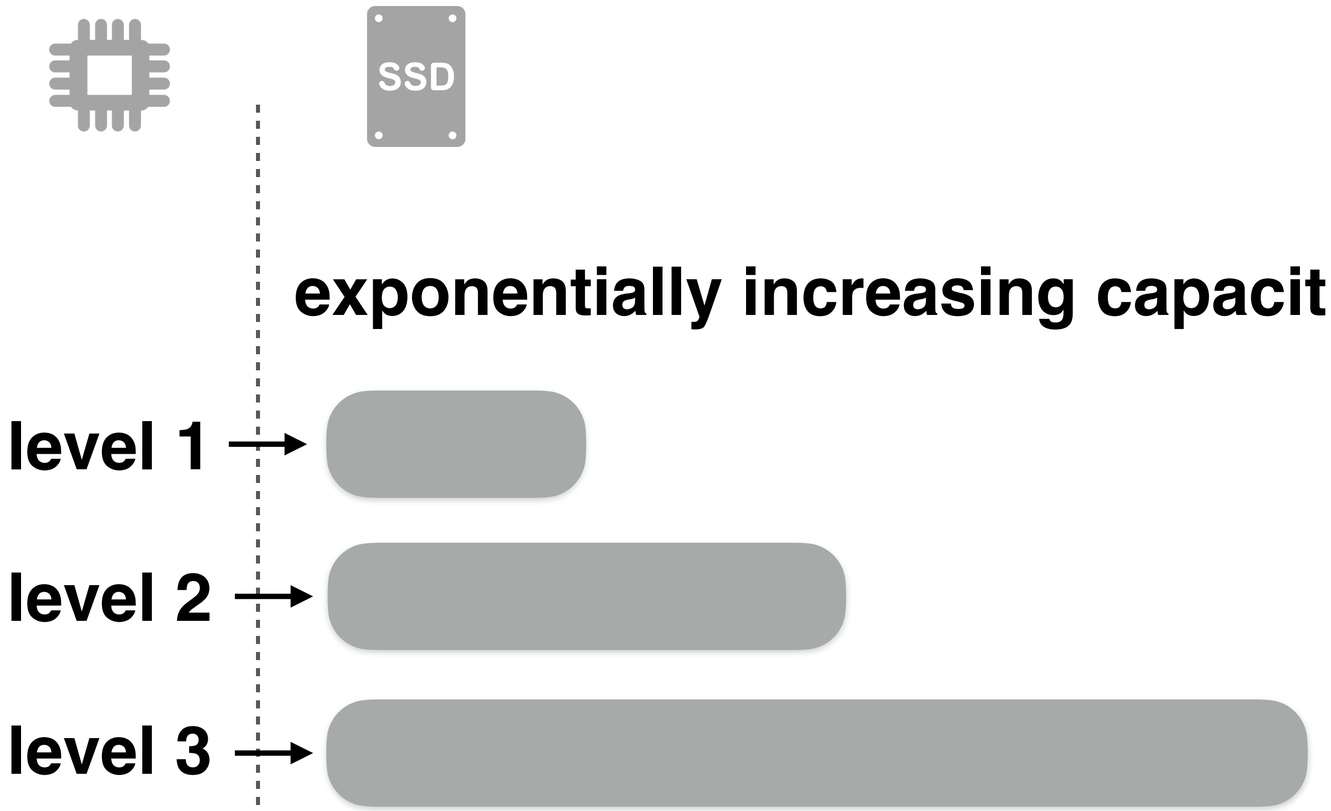
buffer

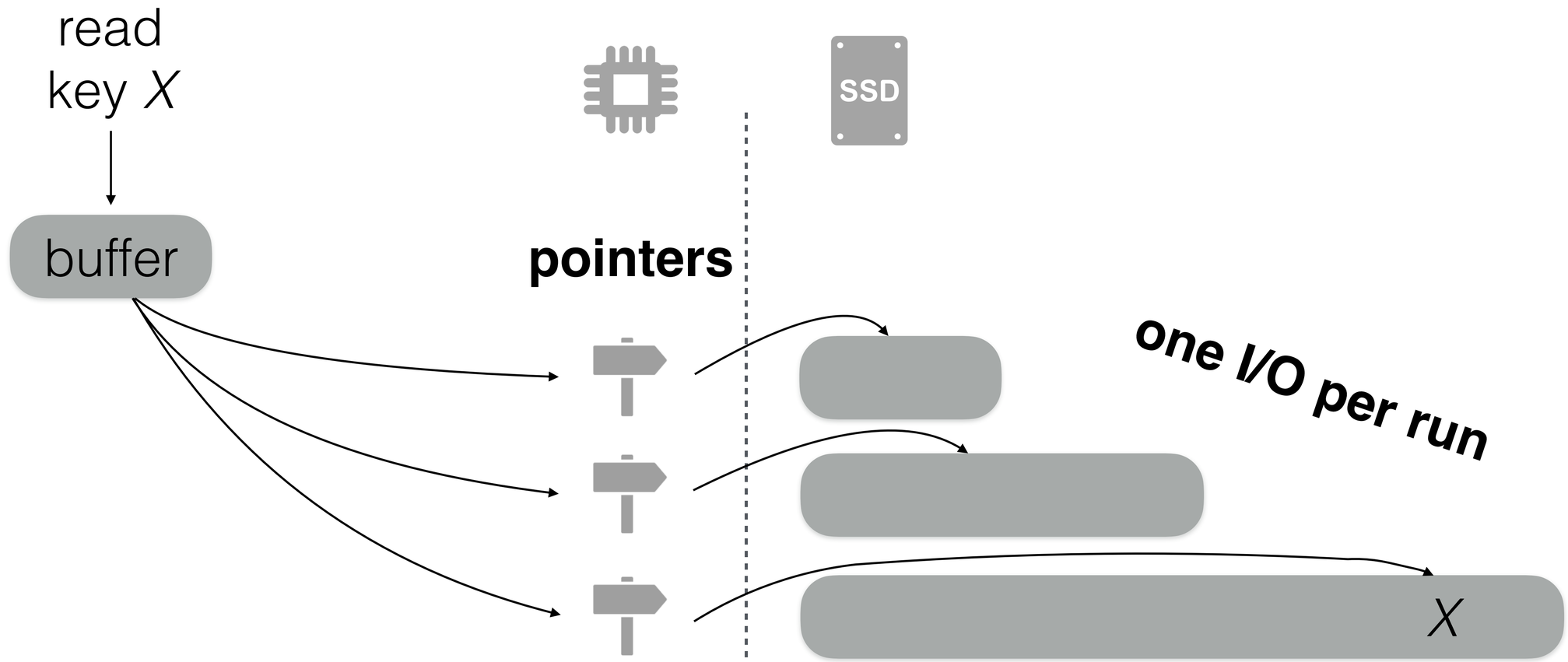
Background

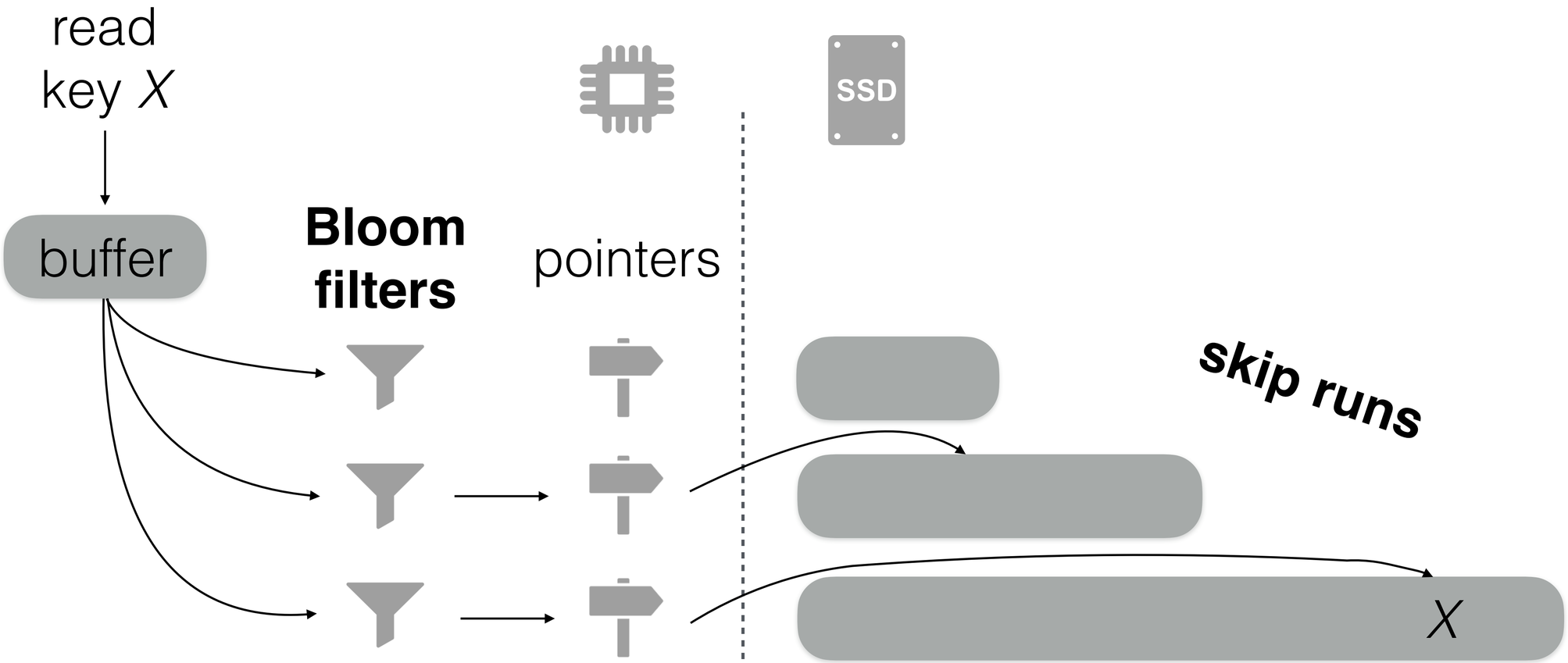


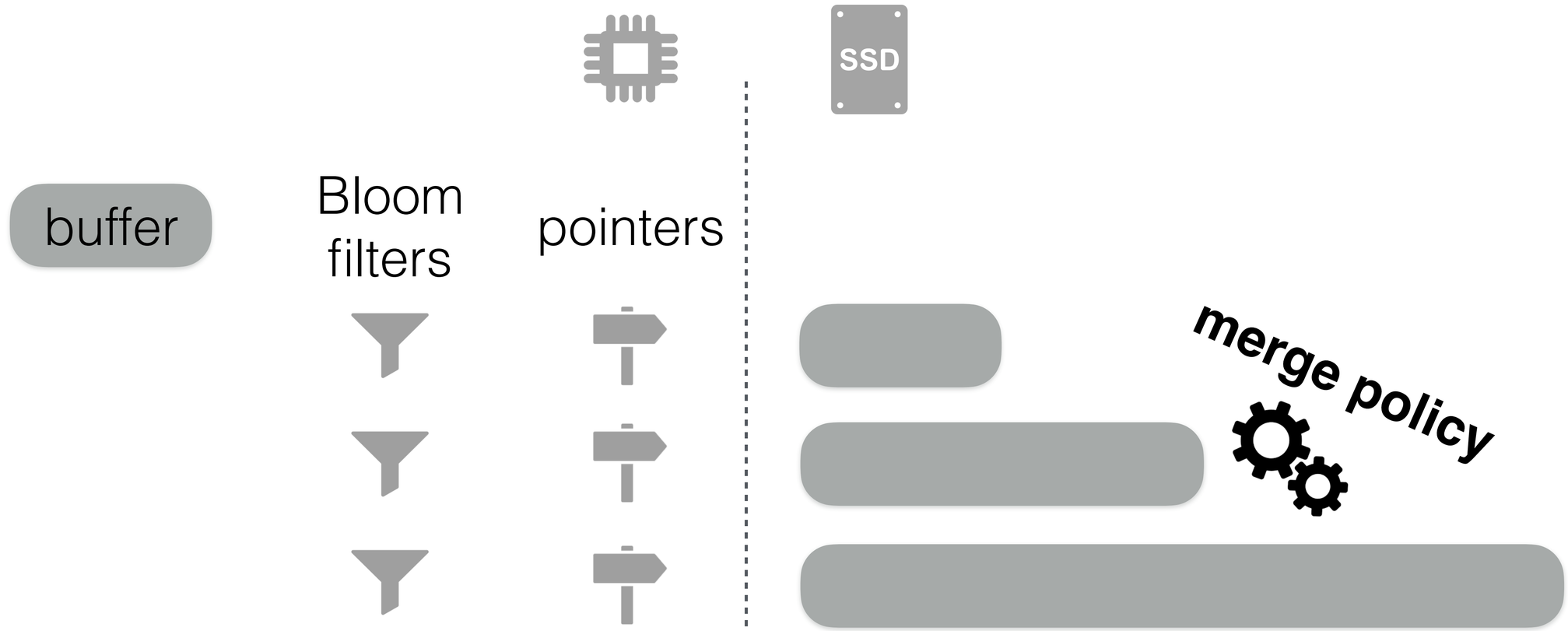

SSD
buffer
sort-merge

buffer


SSD
exponentially increasing capacities
level 1
level 2
level 3


read
key X
buffer
SSD
pointers
one I/O per run
X


read
key X
buffer
Bloom
filters
pointers
SSD
skip runs
X


SSD
buffer
Bloom filters
pointers
merge policy

merge policy

DASlab
@ Harvard SEAS


merge policy



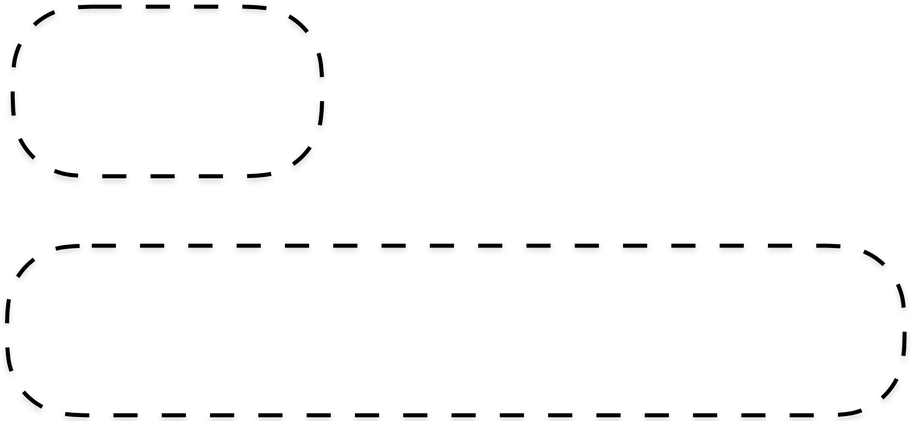
two merge policies

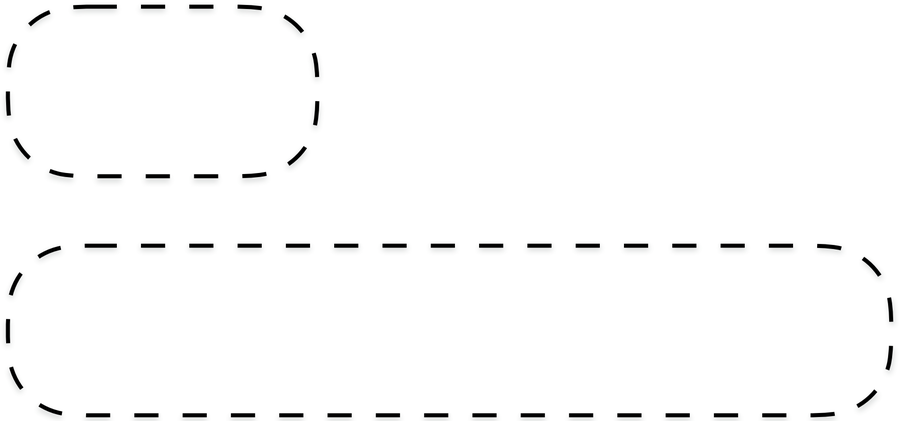
Tiering

Leveling

Tiering

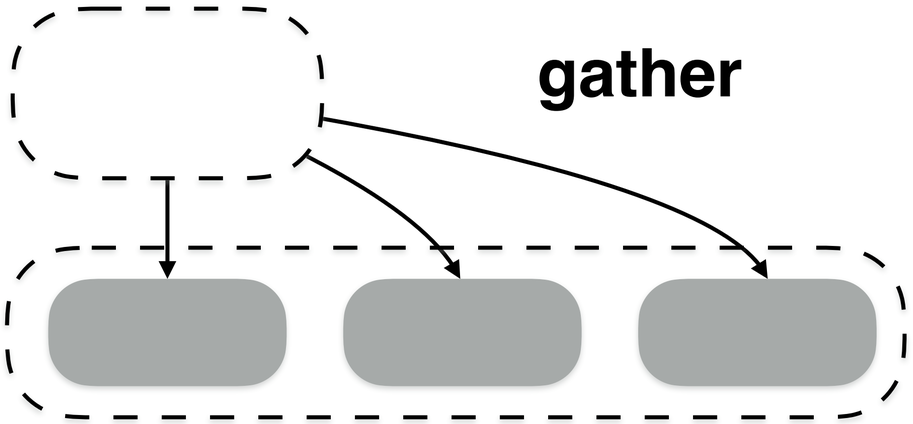
Leveling

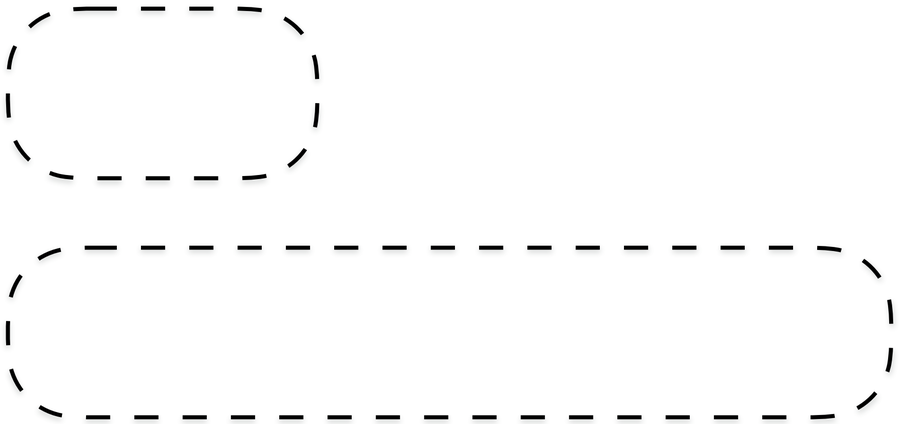




Tiering

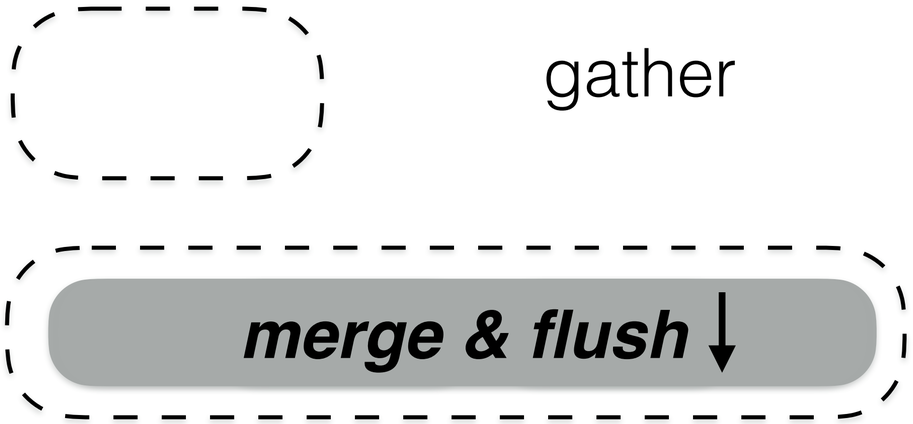
Leveling

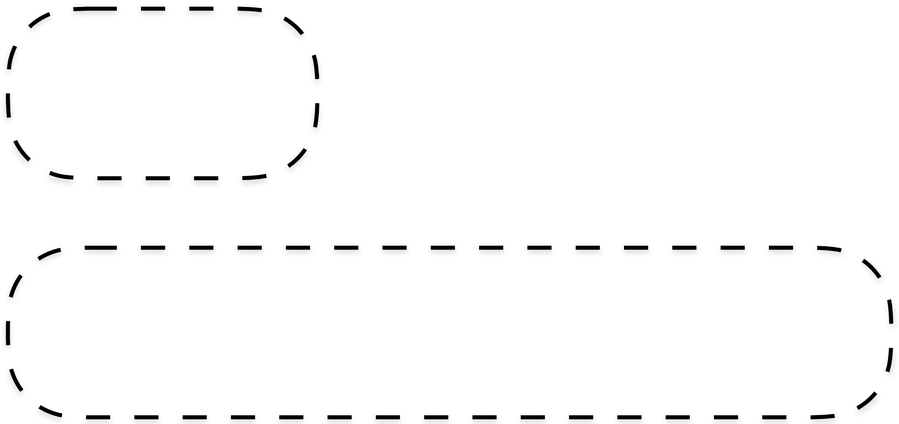




Tiering

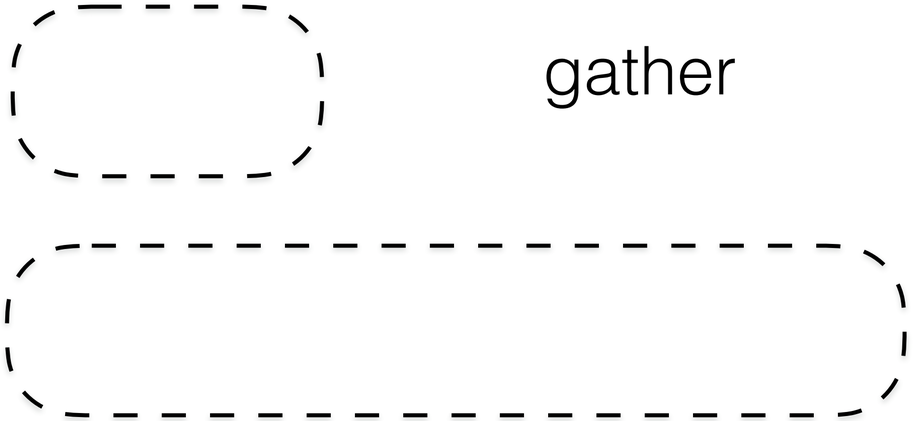
Leveling

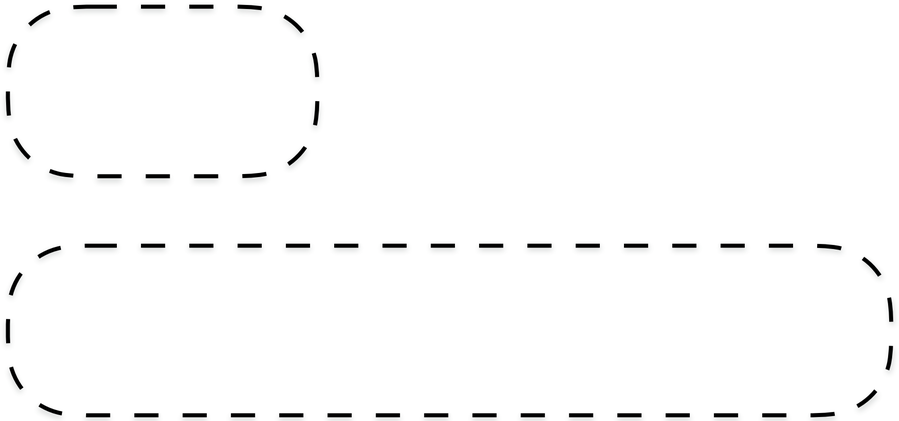




Tiering

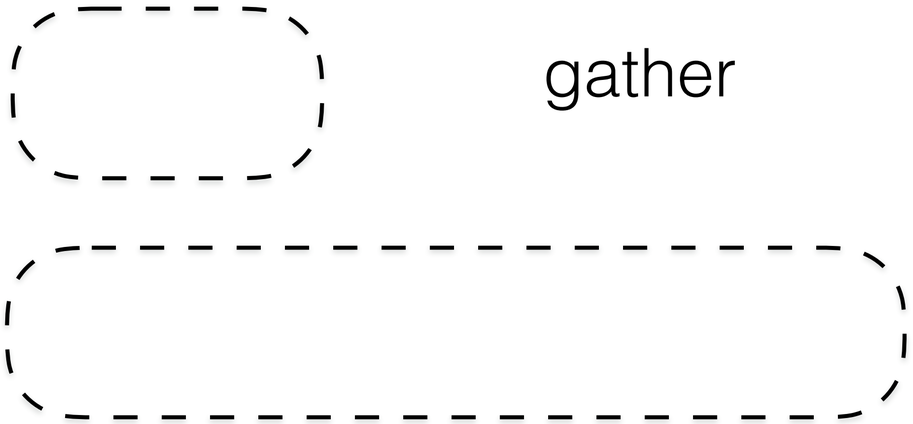
Leveling

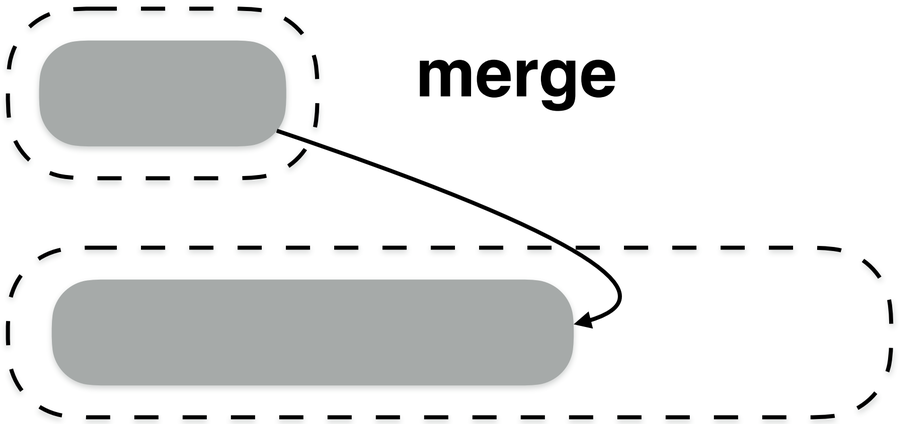




Tiering

Leveling





Tiering

Leveling

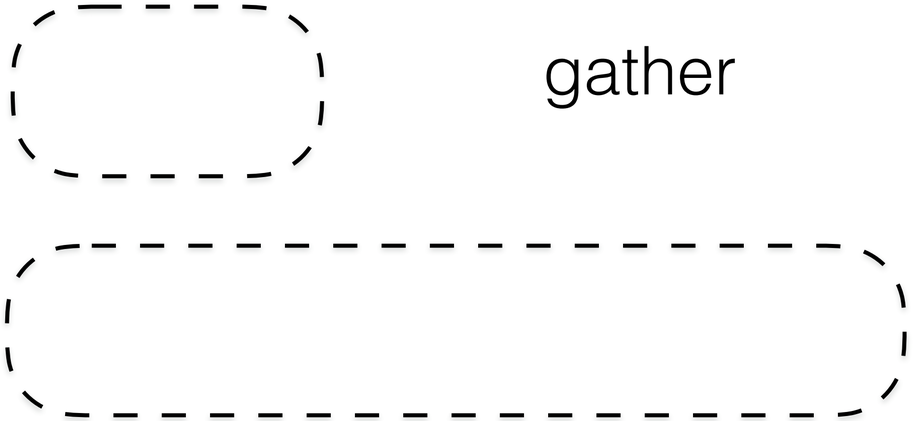
gather

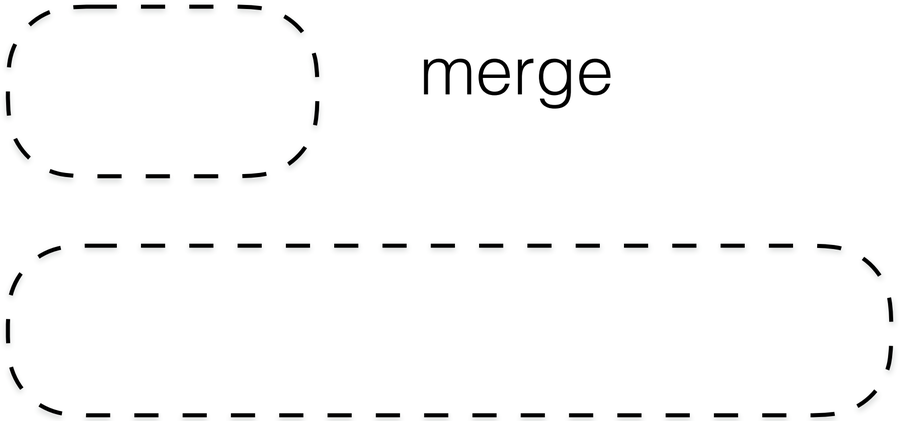
merge

flush

Tiering

Leveling

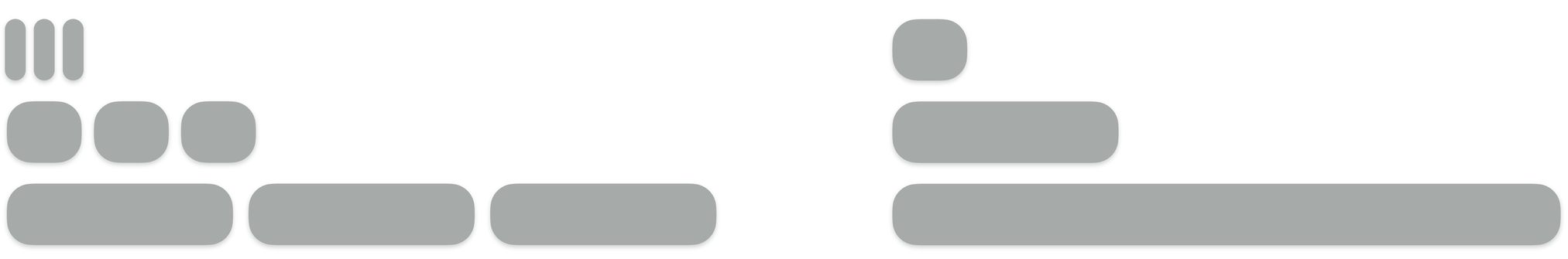




Tiering

Leveling

$\log_R(N)$

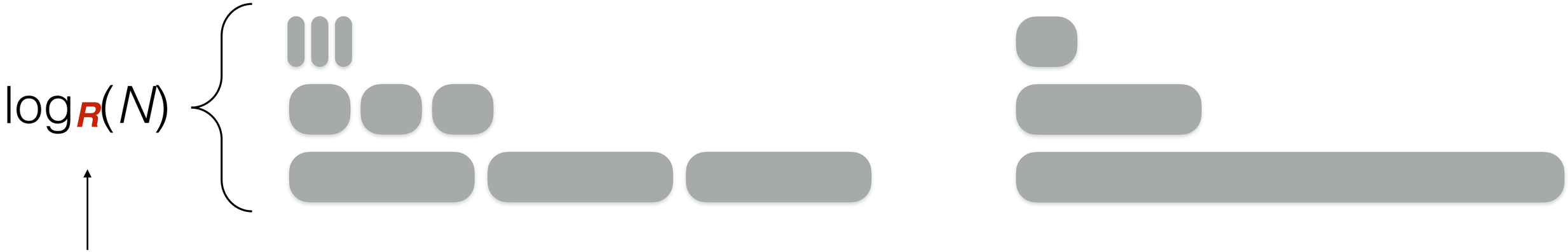


Tiering

Leveling

Tiering

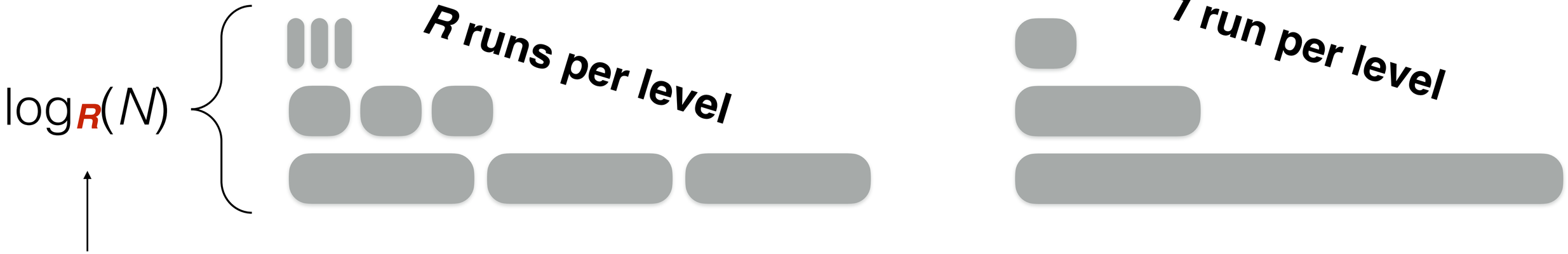
Leveling



base ratio

Tiering

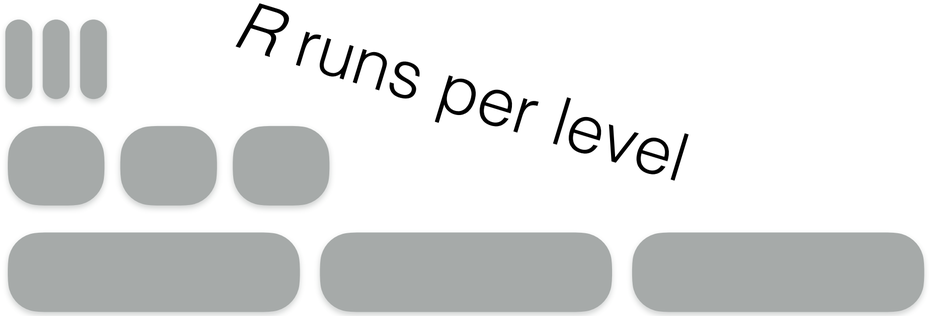
Leveling



base ratio

Tiering

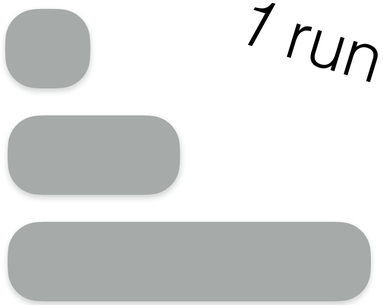
Leveling



base ratio *R*

Tiering

Leveling



1 run per level

1 run per level

base ratio *R*

Tiering

Leveling

base ratio *R*

Tiering

Leveling

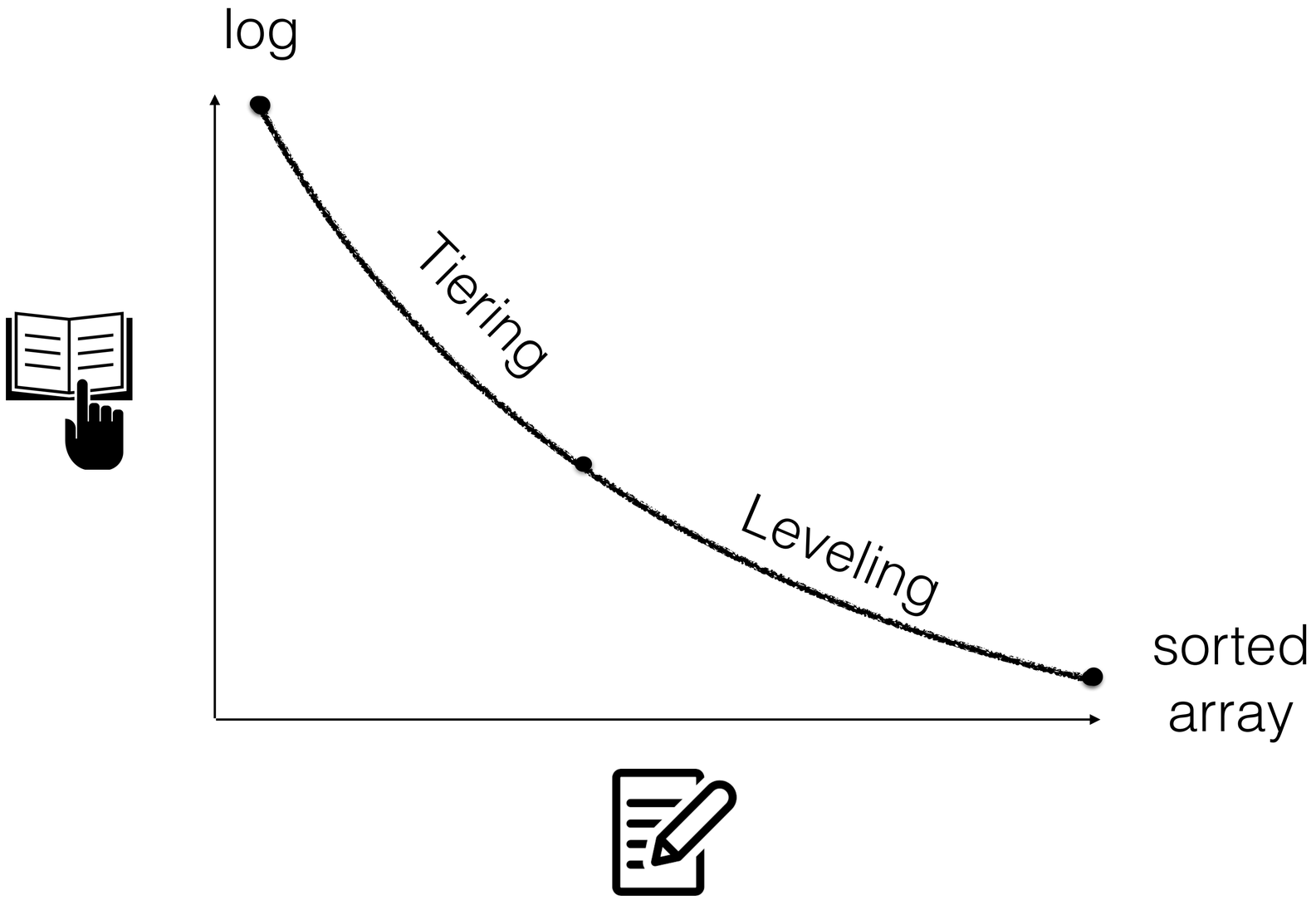
O(*N*) runs per level

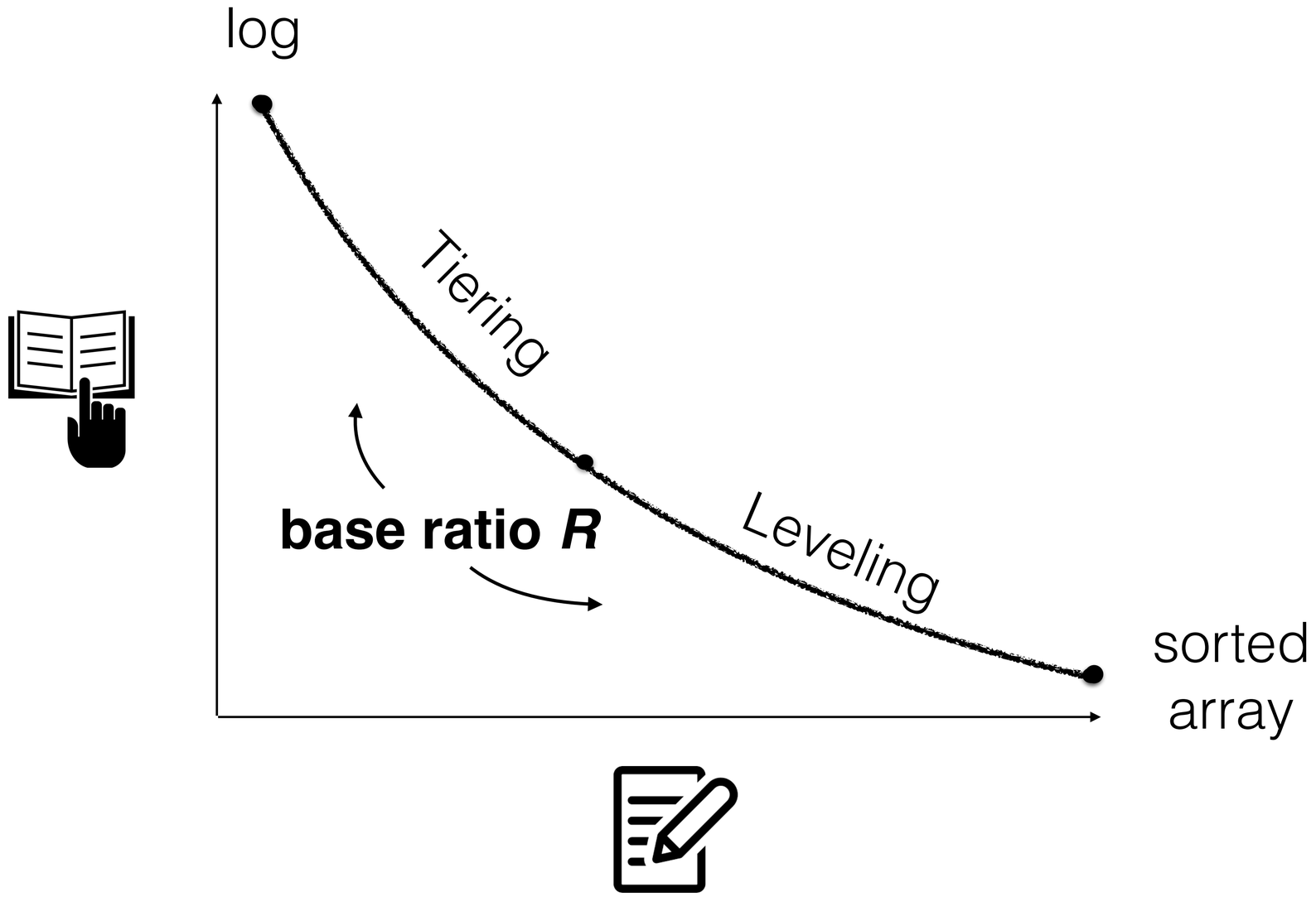
log

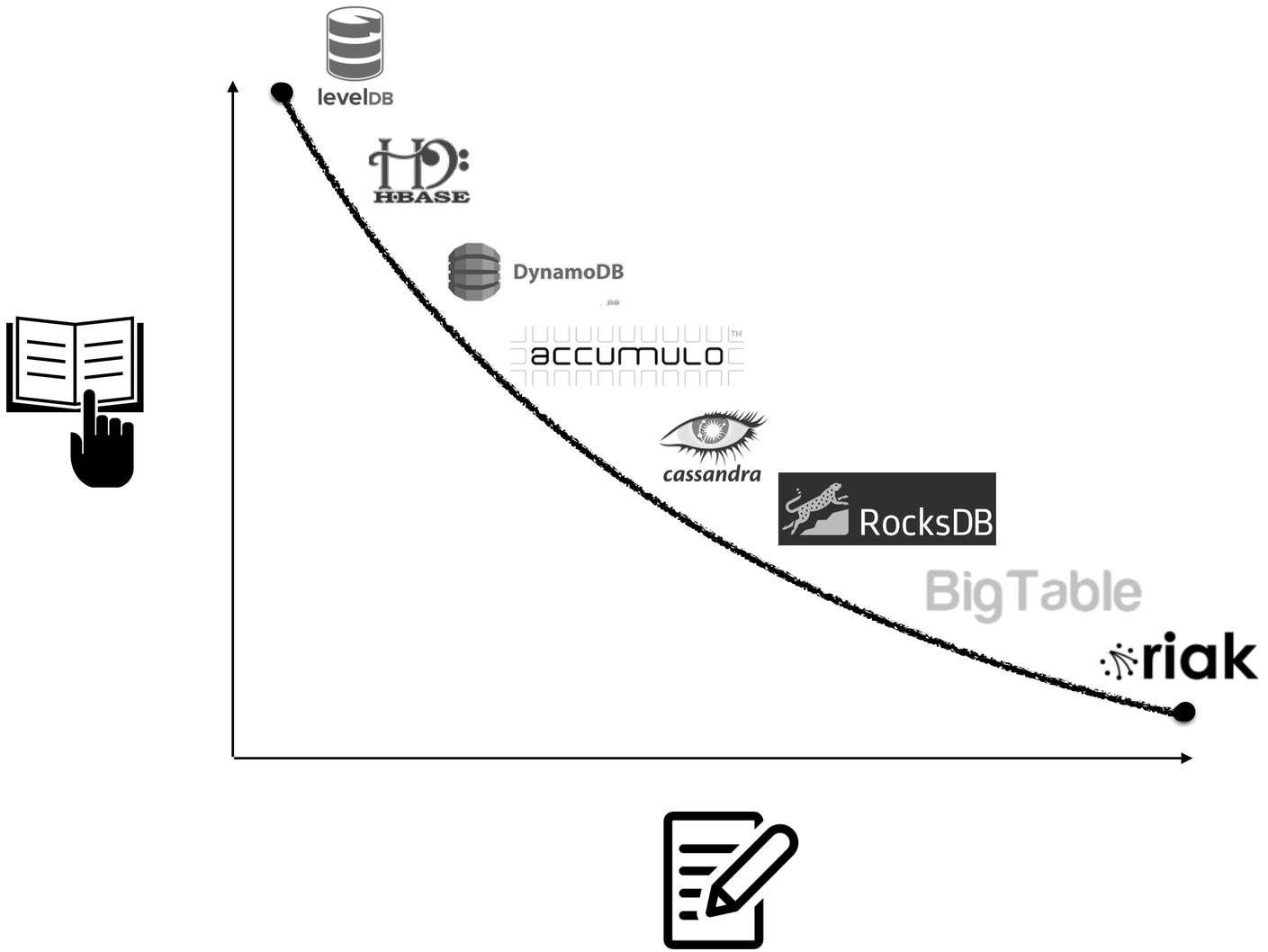
1 run per level

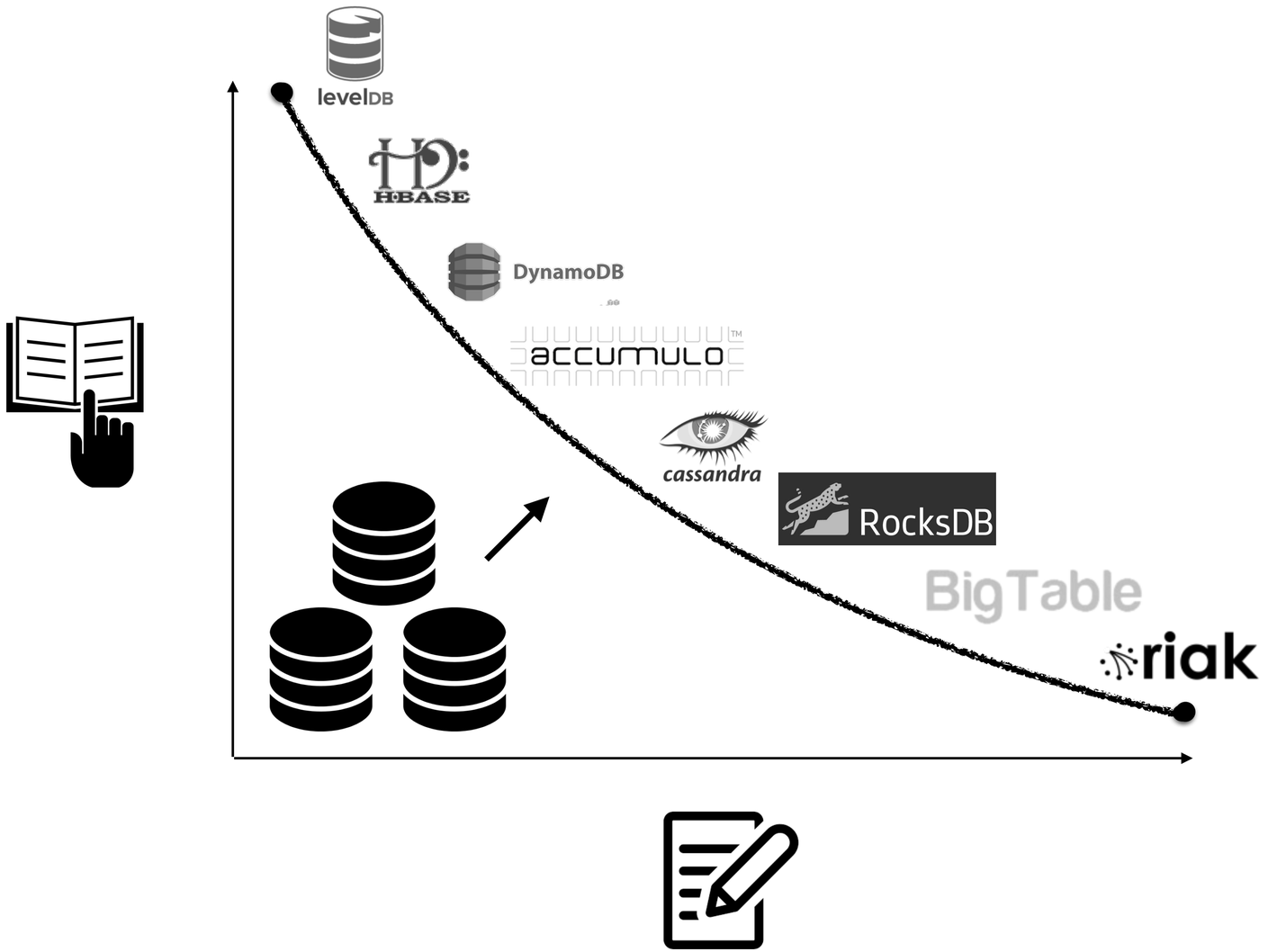
sorted array

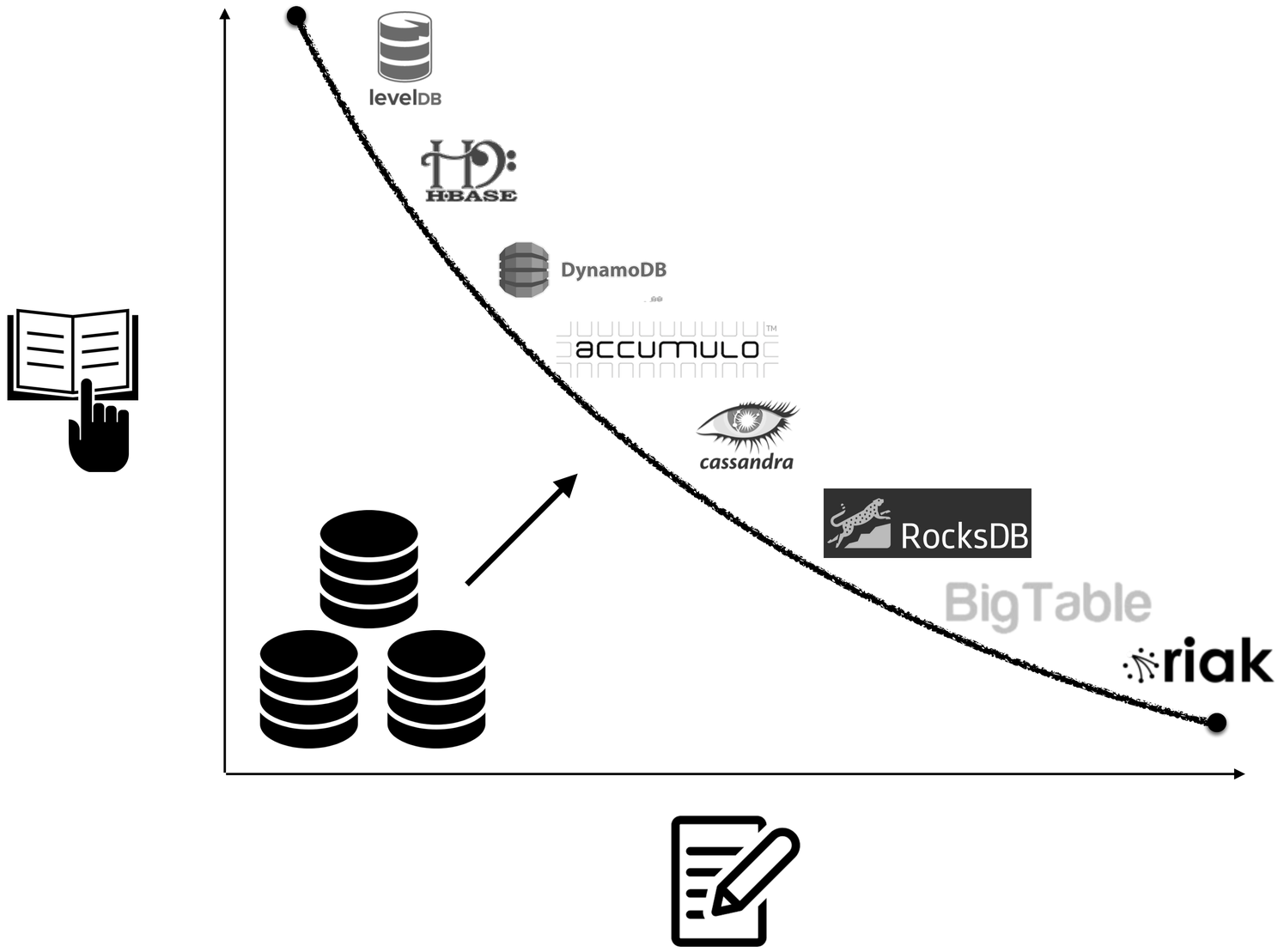
base ratio *R* ≫

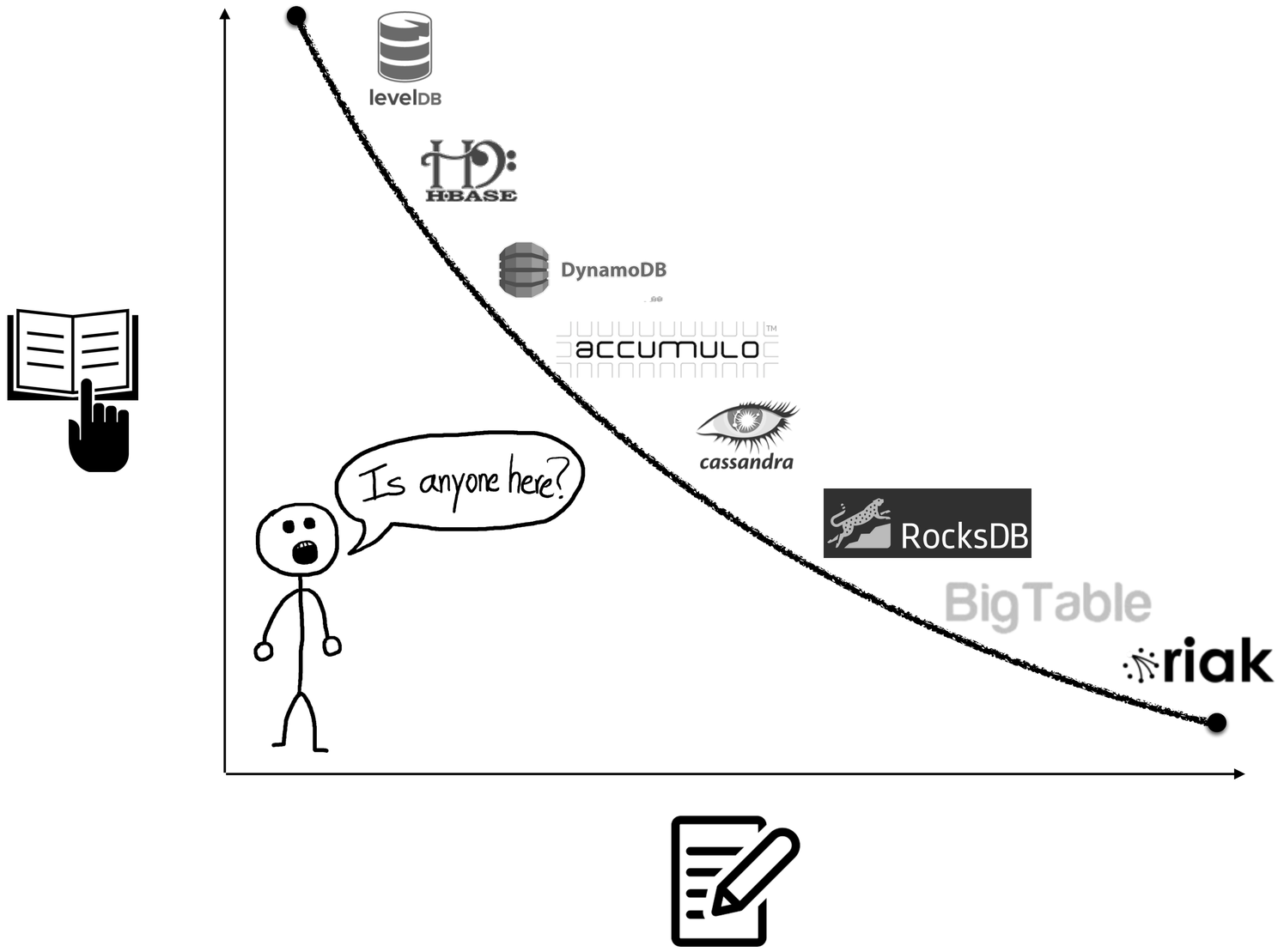

log
Tiering
Leveling
sorted
array

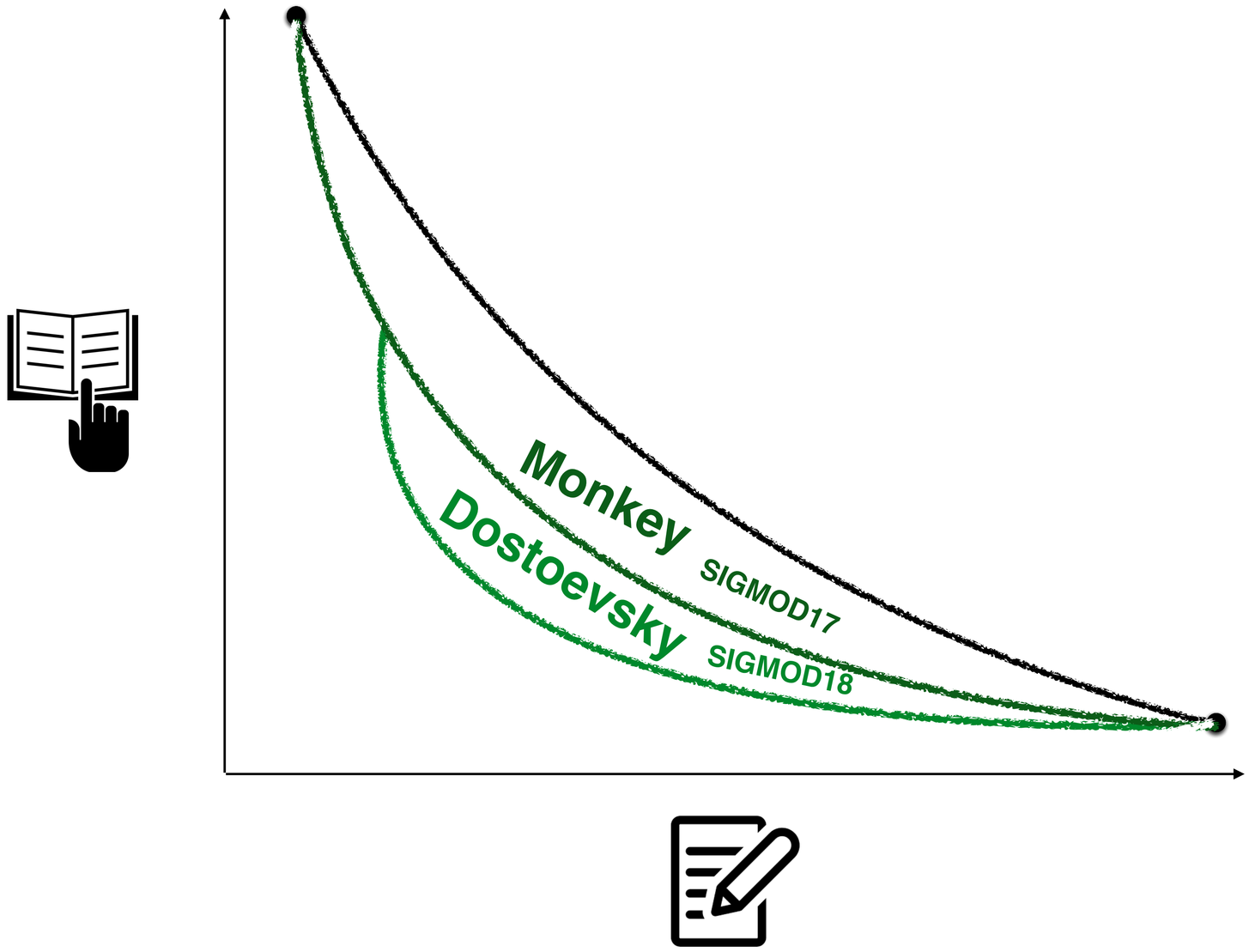

log
Tiering
Leveling
base ratio R
sorted
array

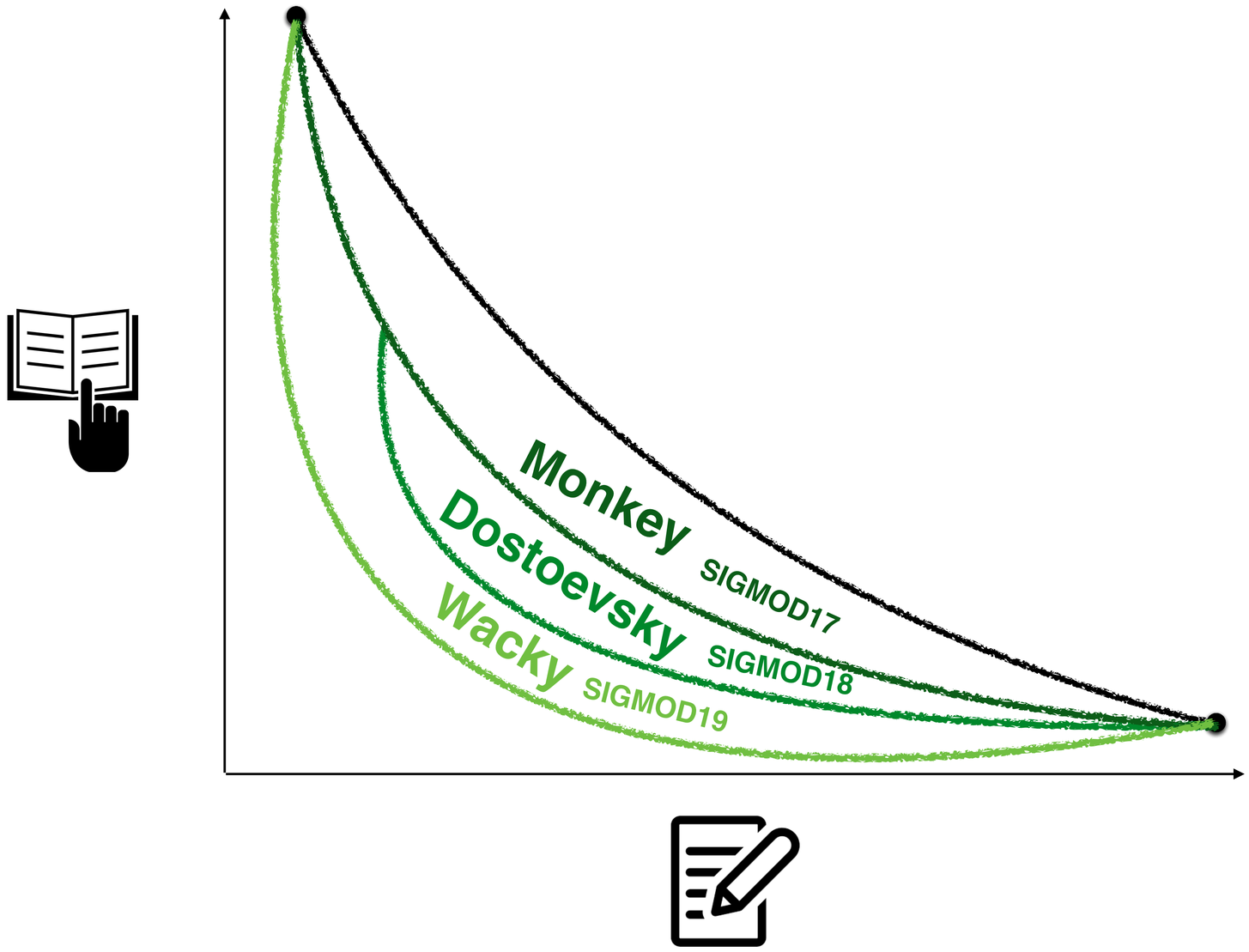

levelDB
HBASE
DynamoDB
accumulo
cassandra
RocksDB
BigTable
riak


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?


Monkey SIGMOD17
Dostoevsky SIGMOD18

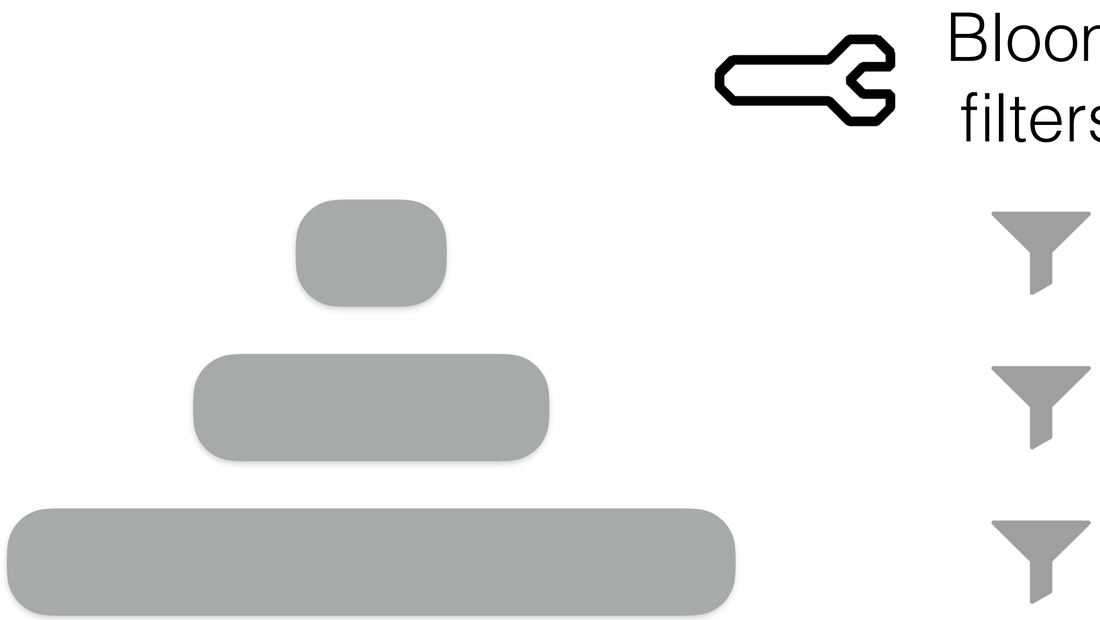

Monkey SIGMOD17
Dostoevsky SIGMOD18
Wacky SIGMOD19

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

SIGMOD17

Monkey: Optimal Navigable Key-Value Store

SIGMOD17



Monkey: Optimal Navigable Key-Value Store

SIGMOD17

Bloom filters

false positive rate

$O(e^{-M})$

$O(e^{-M})$

$O(e^{-M})$

Monkey: Optimal Navigable Key-Value Store

SIGMOD17

Bloom filters

false positive rate

$O(e^{-M})$

$O(e^{-M})$

$O(e^{-M})$

$= O(\mathbf{e^{-M} \cdot \log_R(N)})$

DASlab @ Harvard SEAS

Monkey: Optimal Navigable Key-Value Store

SIGMOD17

Bloom filters

false positive rate

$O(e^{-M}/\mathbf{R^2})$

$O(e^{-M}/\mathbf{R})$

$O(e^{-M})$

DASlab @ Harvard SEAS

Monkey: Optimal Navigable Key-Value Store

SIGMOD17

Bloom filters

false positive rate

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

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

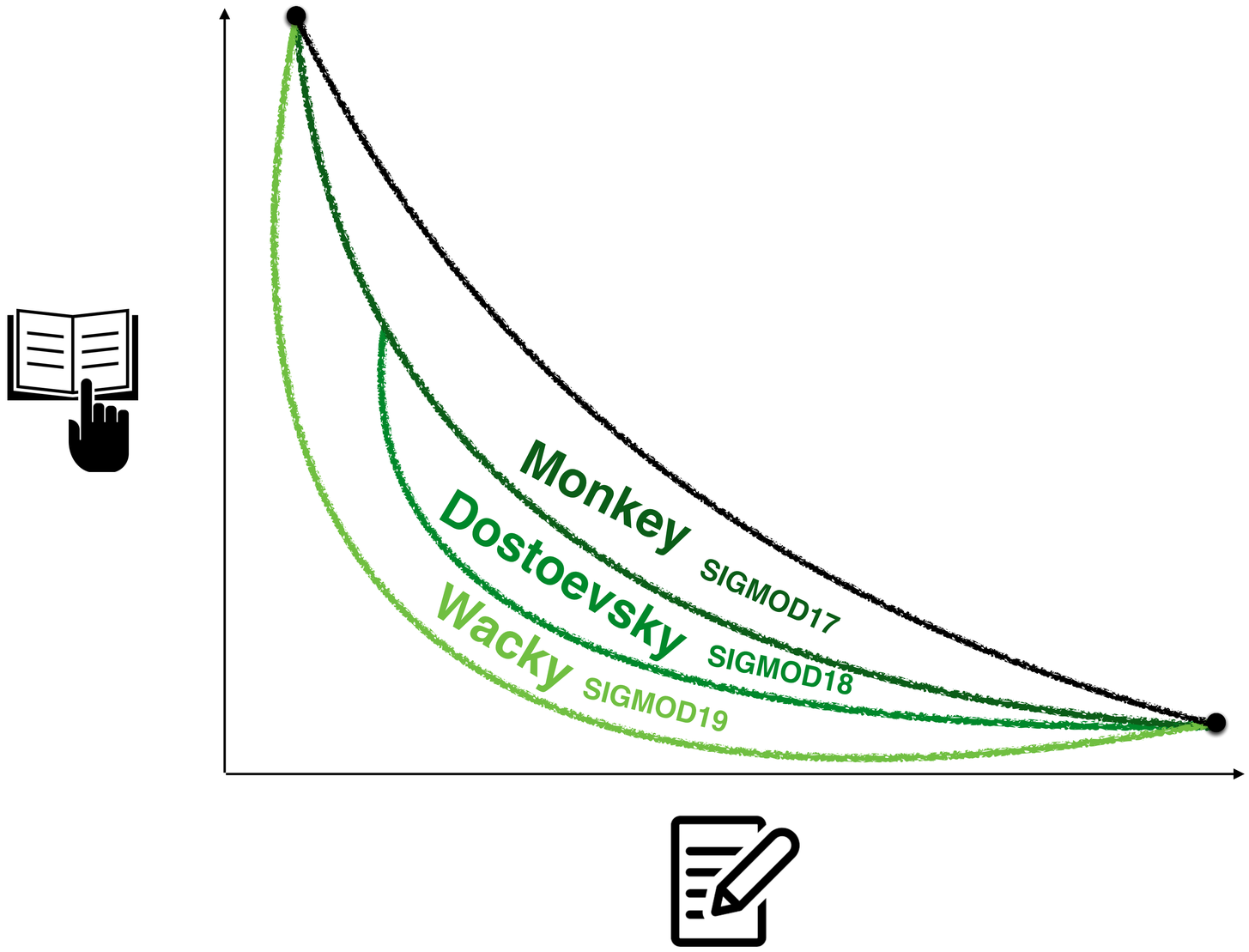
$O(e^{-M})$

$= \mathbf{O(e^{-M})}$

Monkey: Optimal Navigable Key-Value Store SIGMOD17

Bloom filters

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


Monkey
SIGMOD17
Dostoevsky
SIGMOD18
Wacky
SIGMOD19

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

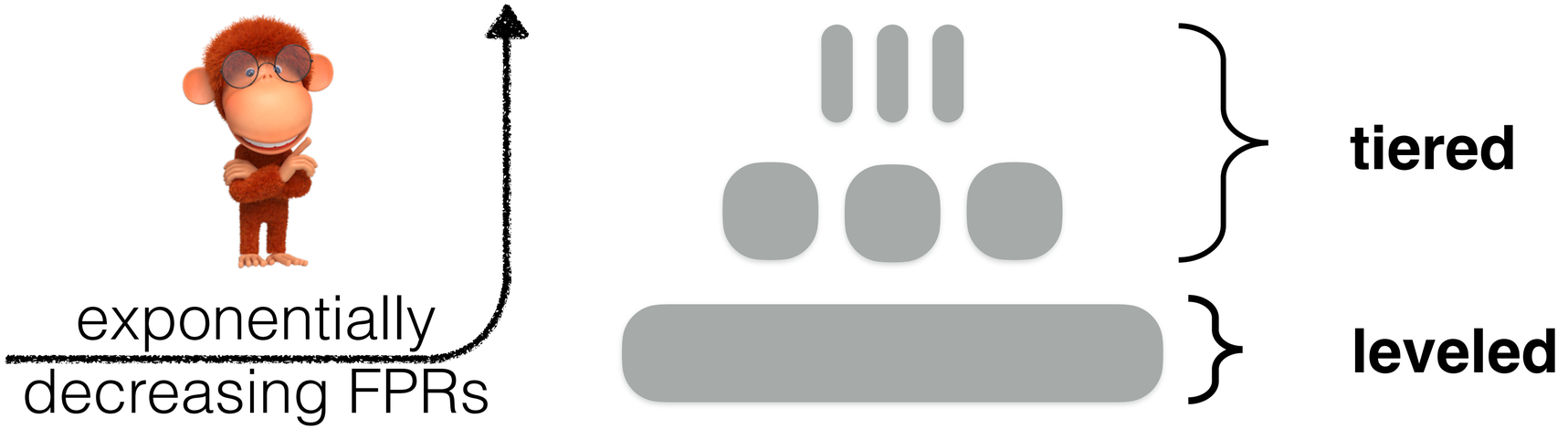
SIGMOD18

Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store

SIGMOD18

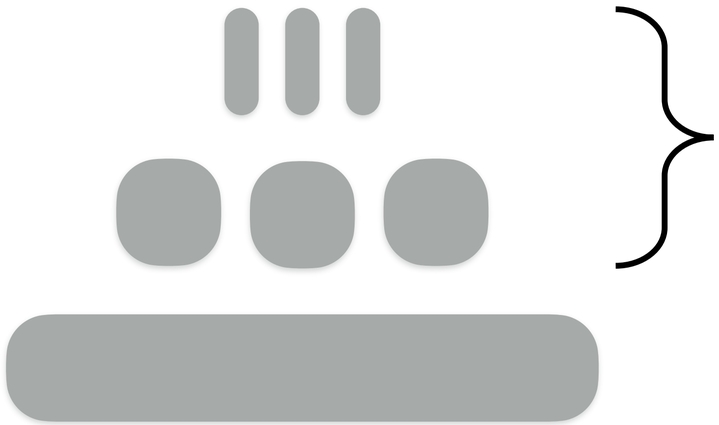
Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store

SIGMOD18



Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store
SIGMOD18

exponentially
decreasing FPRs



more runs

Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store
SIGMOD18

lower access probability

more runs

Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store

SIGMOD18

$O(e^{-M})$

$O(R + \log_R(N))$

Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store

SIGMOD18

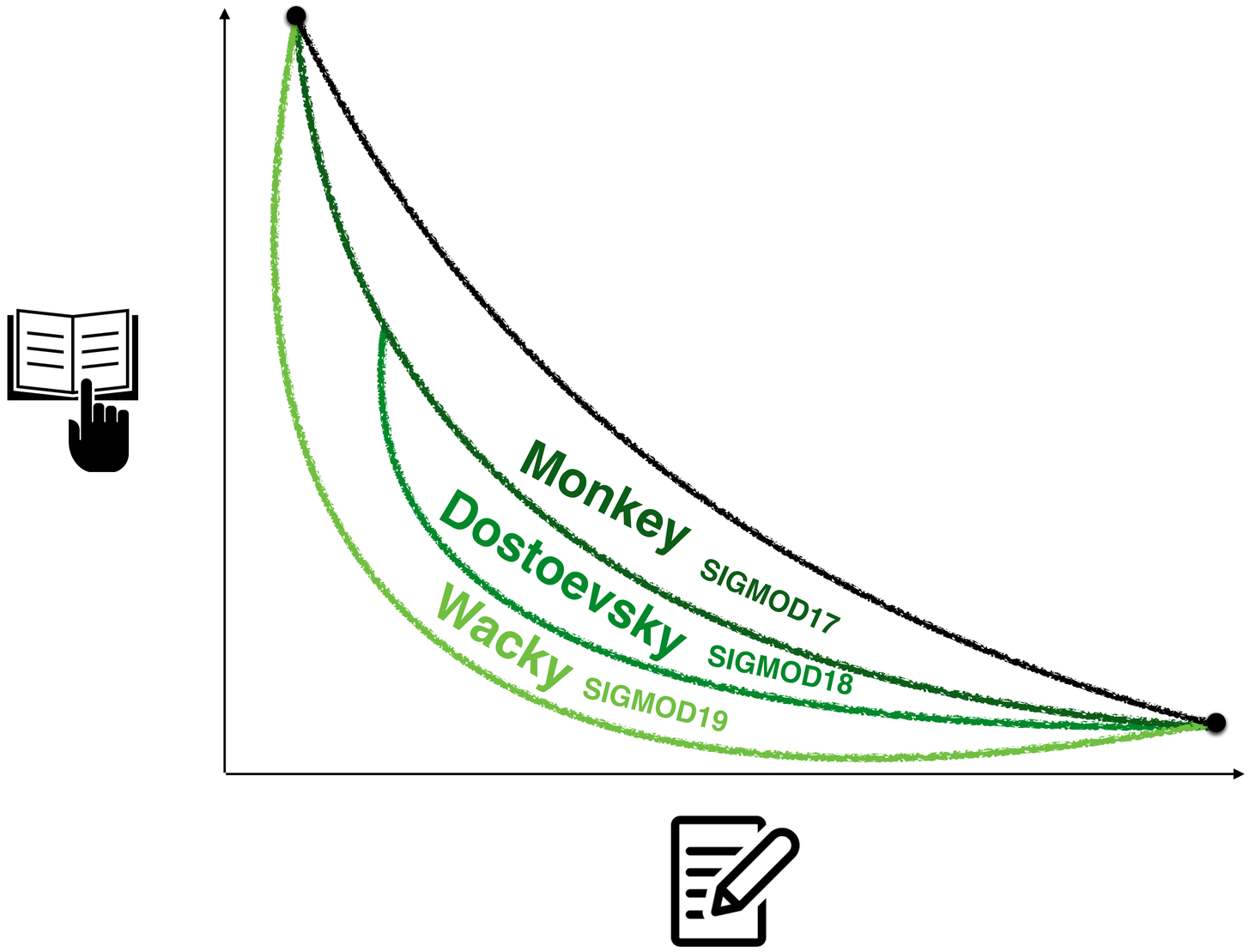
Leveling

Lazy Leveling

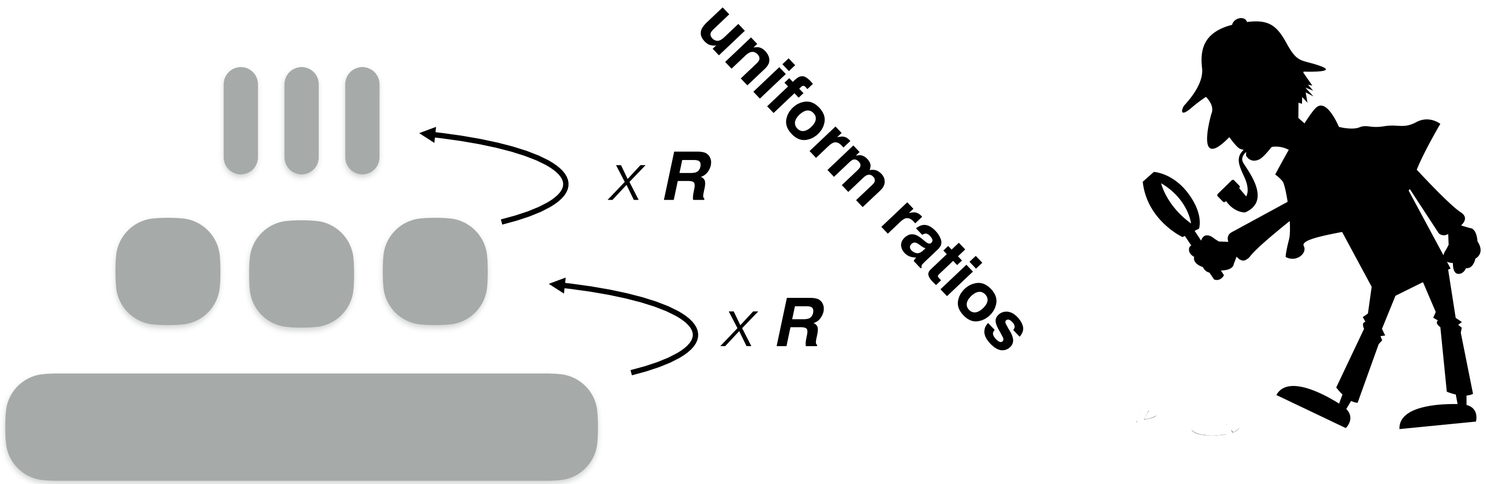
$O(e^{-M})$ = $O(e^{-M})$

$O(R \cdot \log_R(N))$ > $O(R + \log_R(N))$

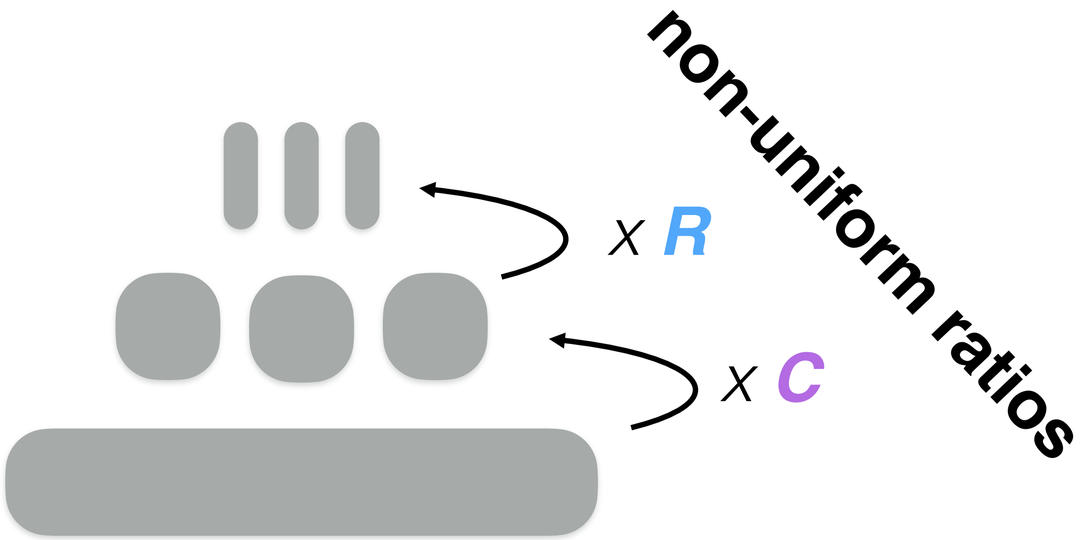
DASlab @ Harvard SEAS


Monkey
SIGMOD17
Dostoevsky
SIGMOD18
Wacky
SIGMOD19

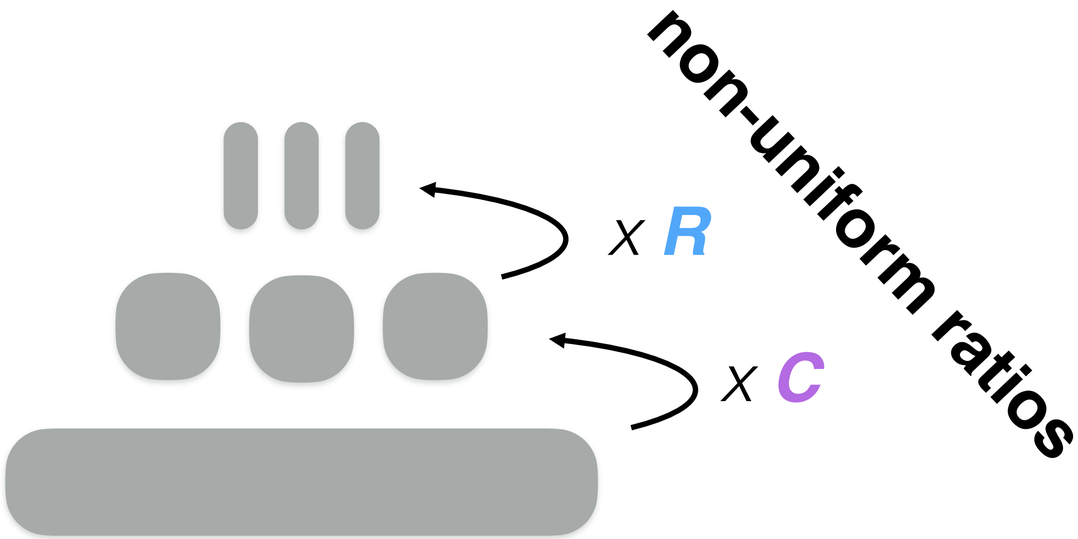
Wacky SIGMOD19



Wacky SIGMOD19

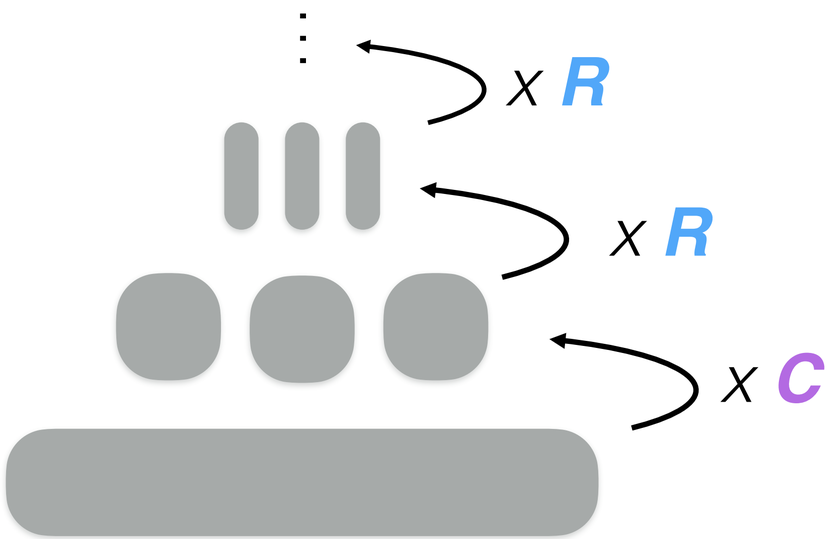


Wacky SIGMOD19



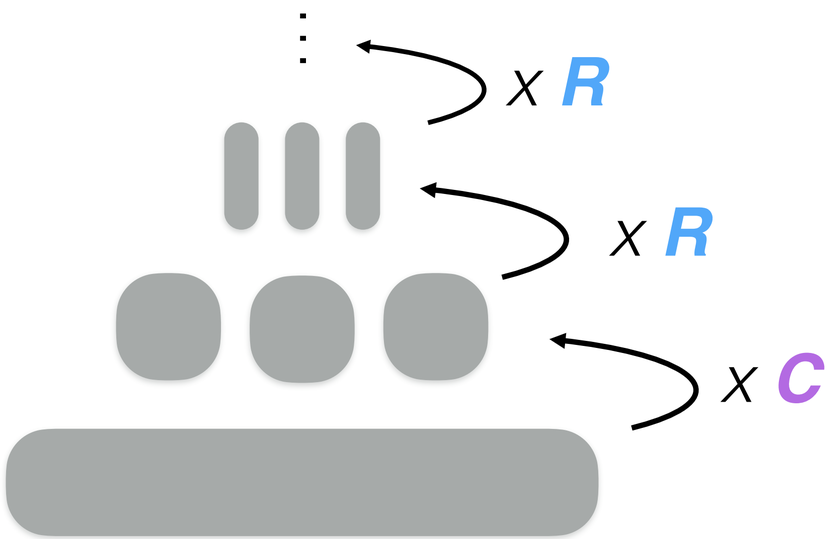
Wacky SIGMOD19

1. Squared Capped Lazy Leveling

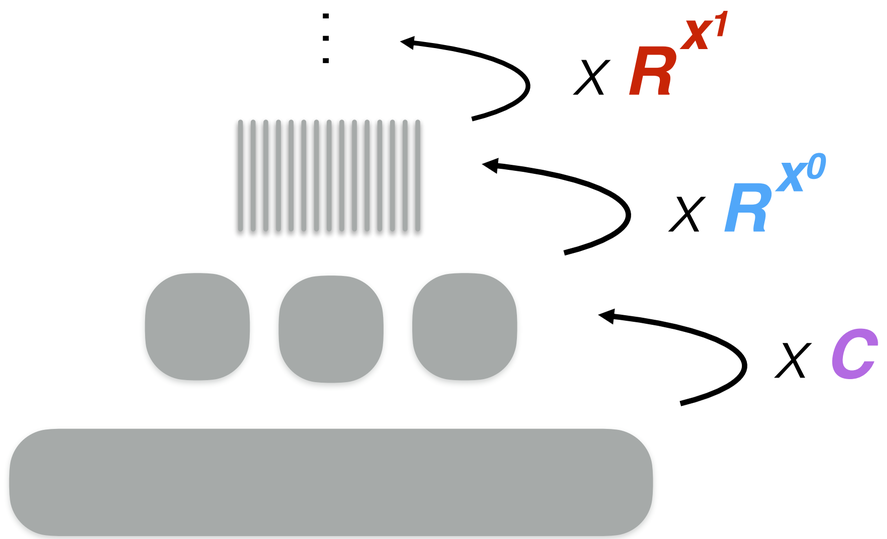


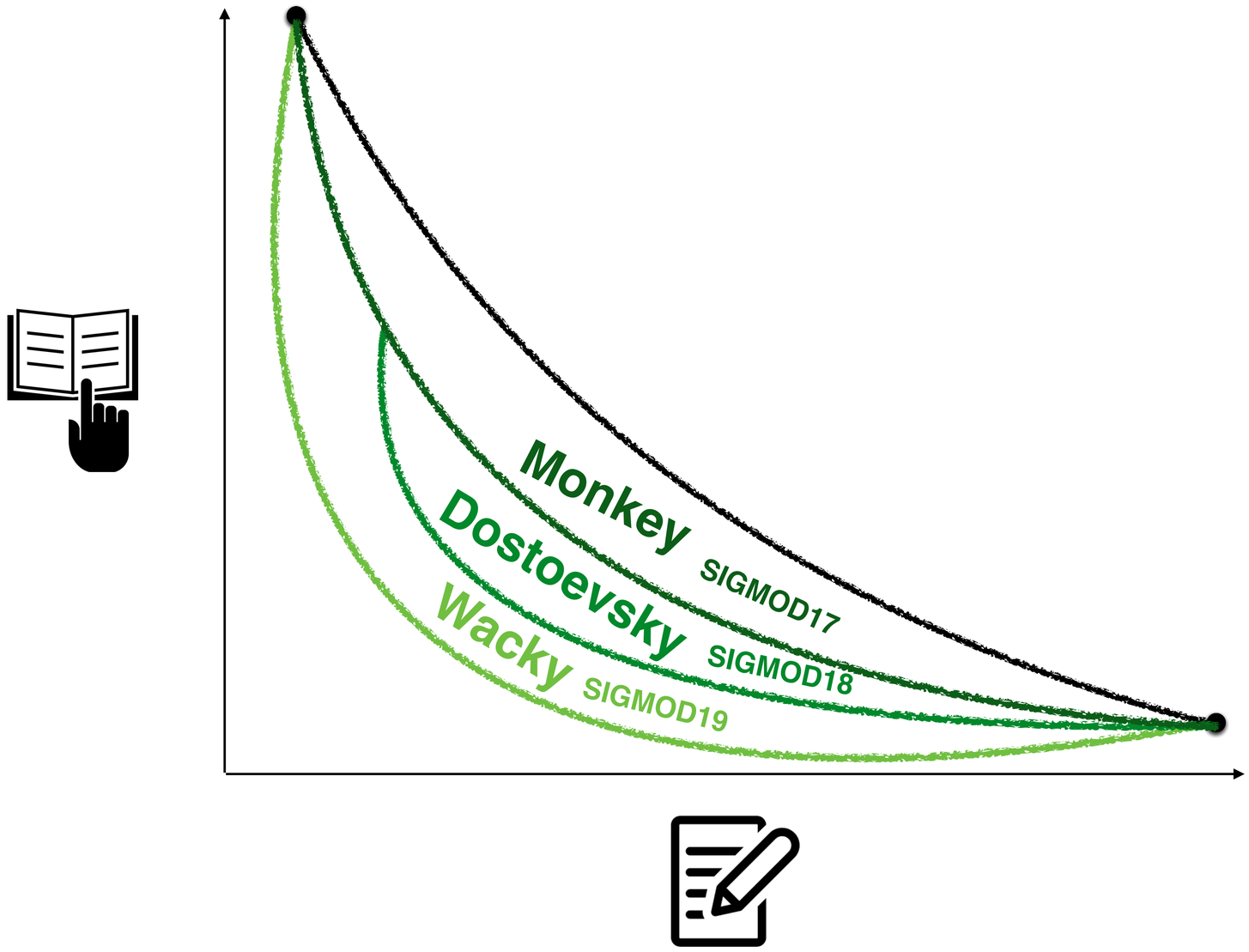
Wacky SIGMOD19

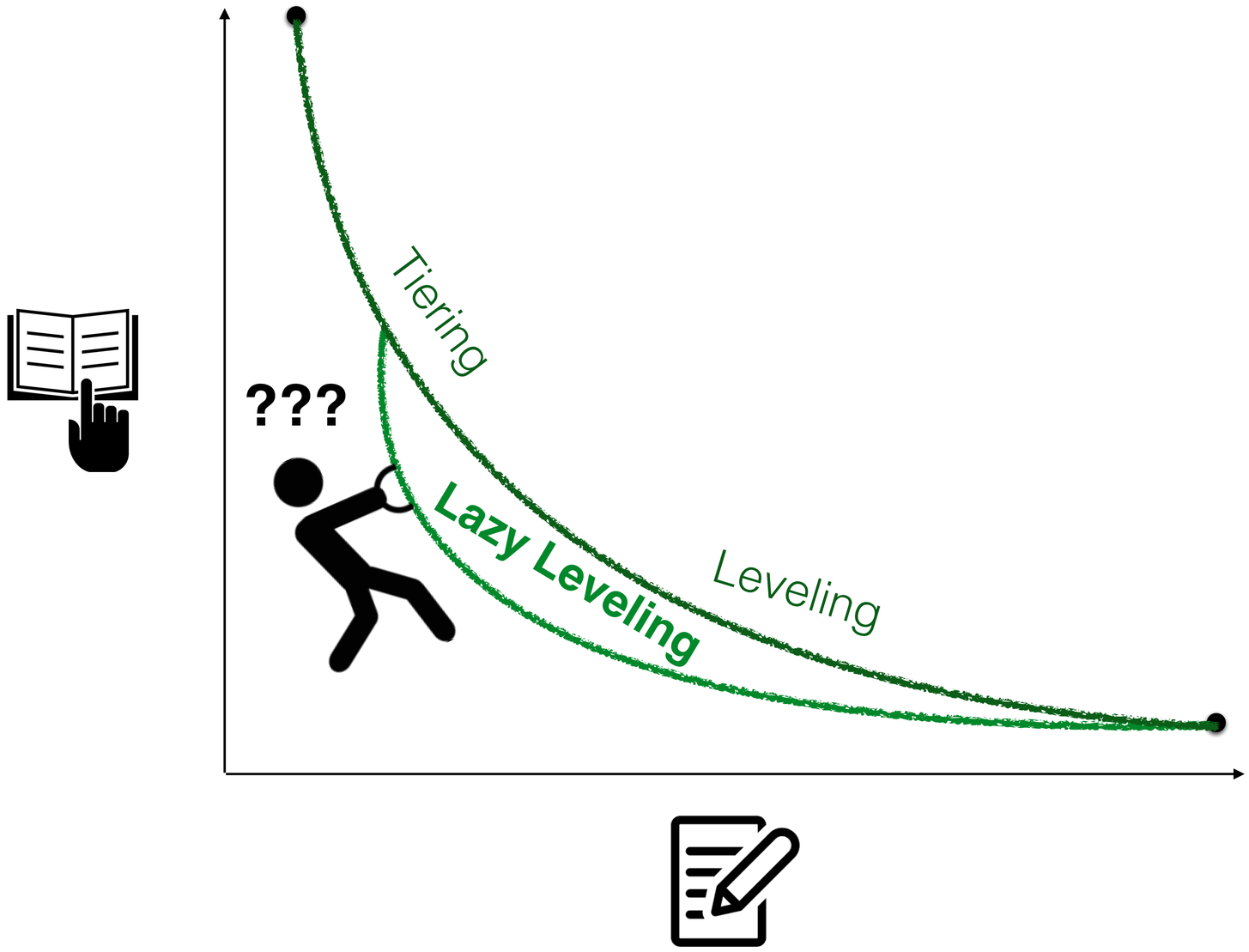
1. Squared Capped Lazy Leveling



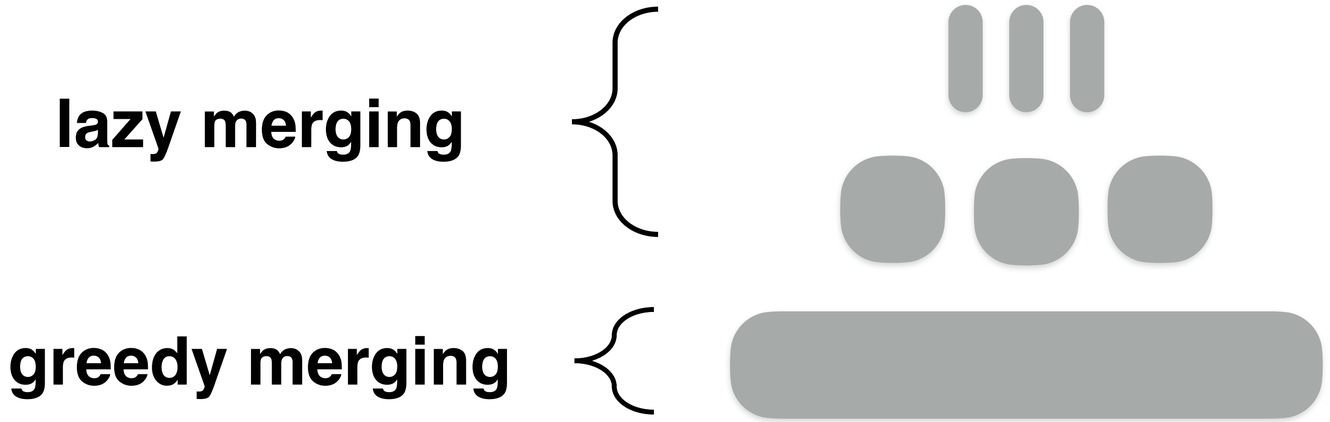
2. LSM-bush



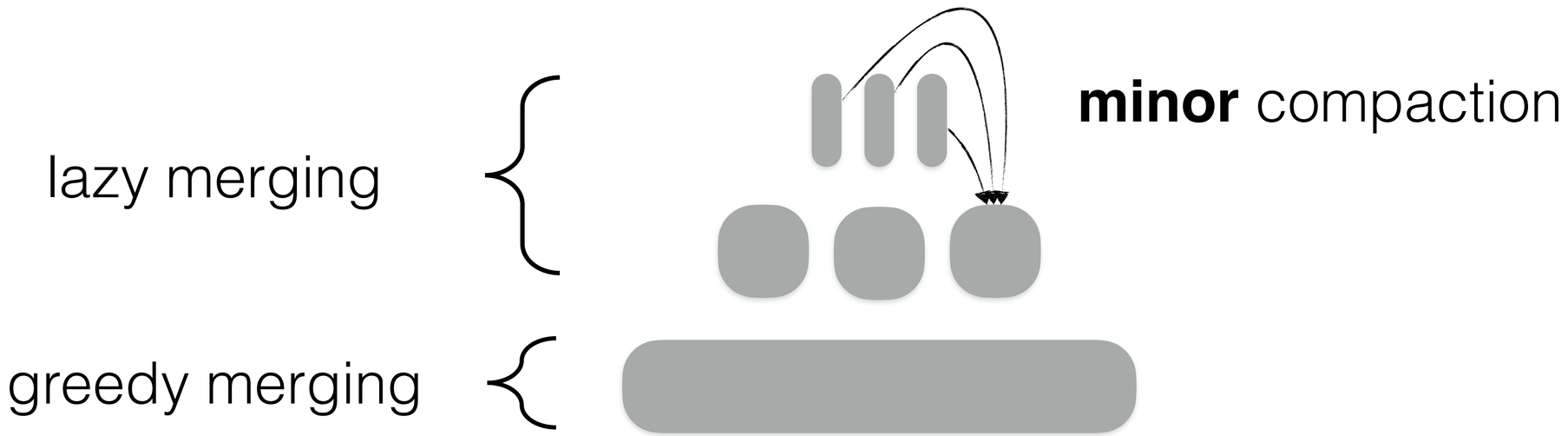

Monkey SIGMOD17
Dostoevsky SIGMOD18
Wacky SIGMOD19


???
Tiering
Lazy Leveling
Leveling

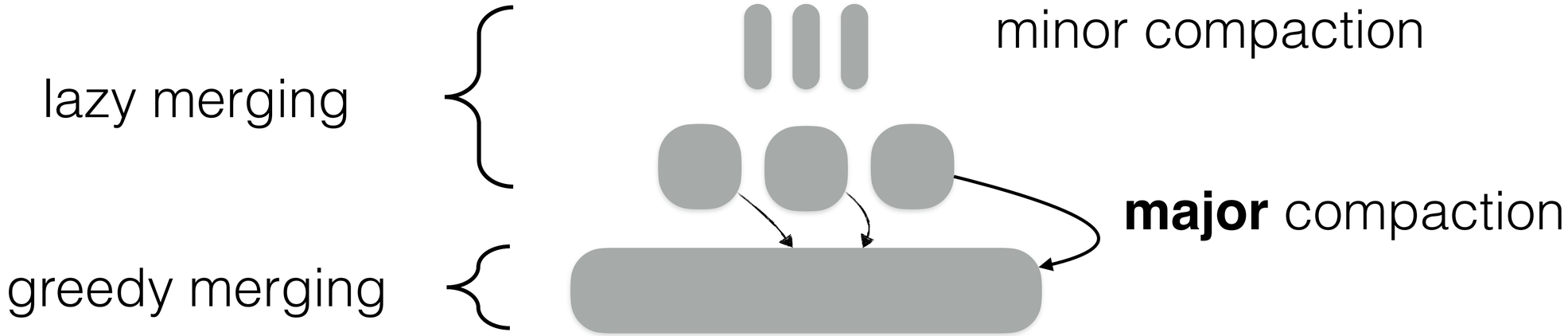
Problem Analysis: Lazy Leveling



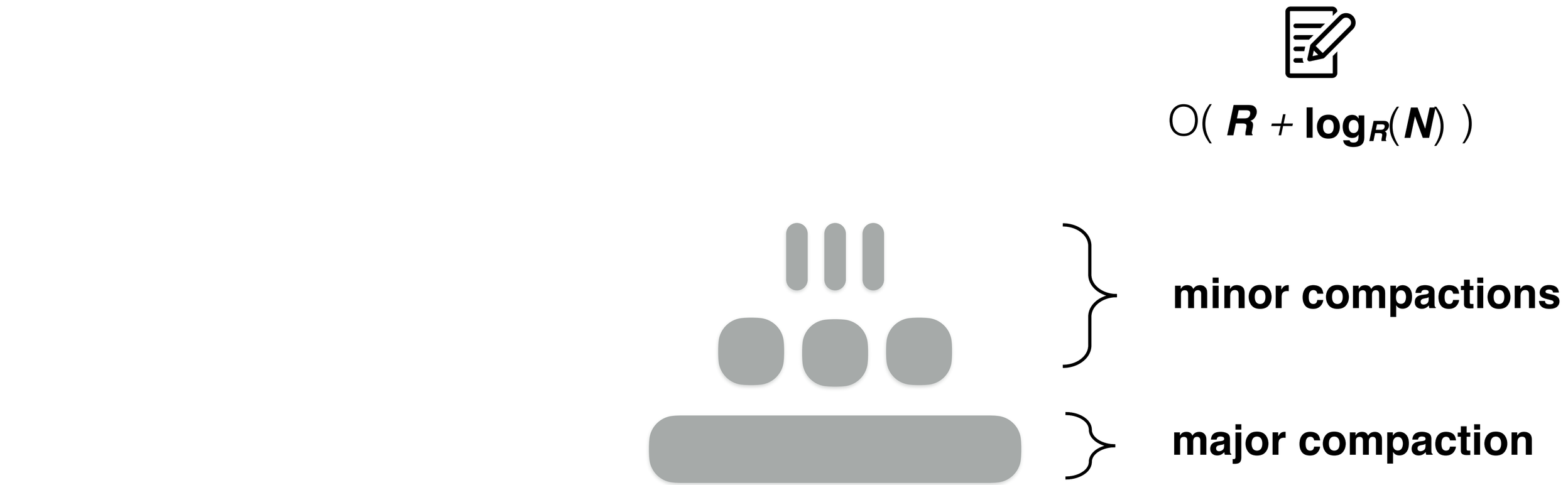
Problem Analysis: Lazy Leveling



Problem Analysis: Lazy Leveling



Problem Analysis: Lazy Leveling



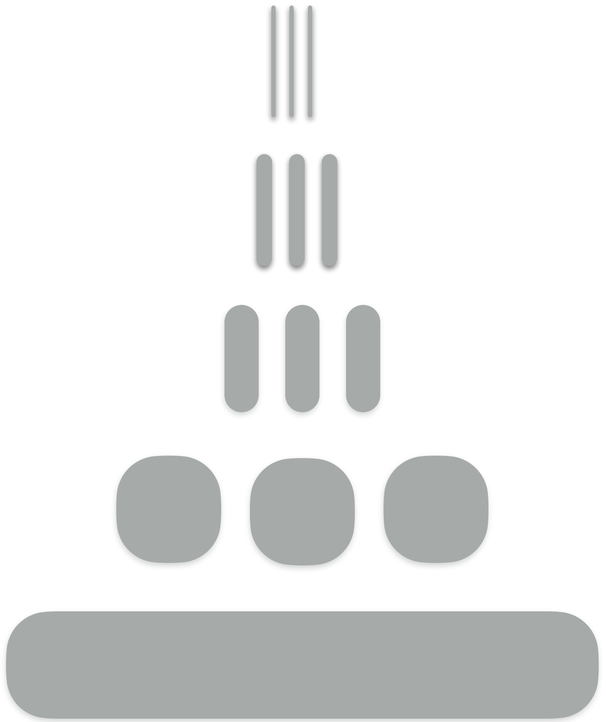
Problem Analysis: Lazy Leveling

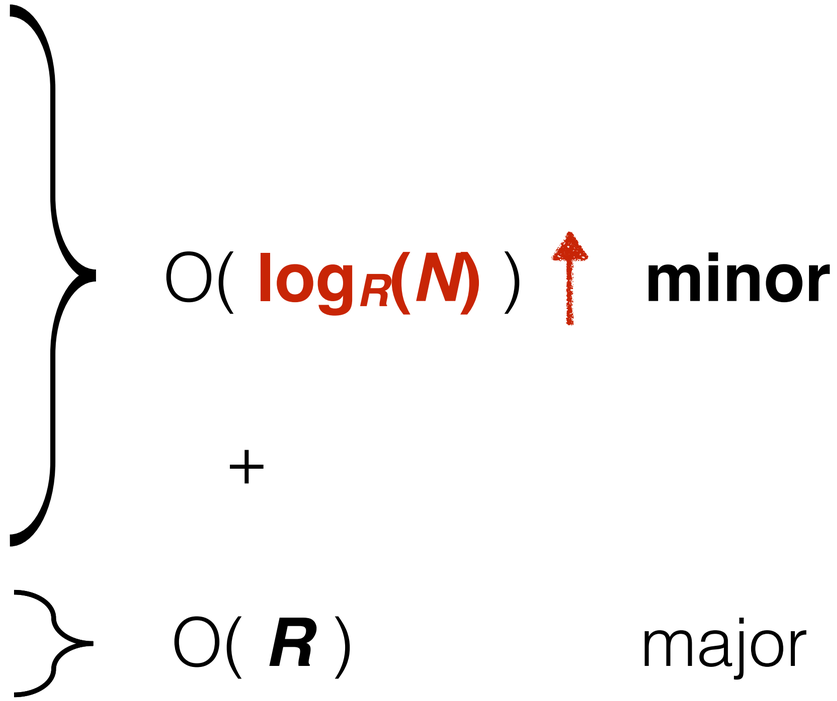
$O(\log_R(N))$ **minor**

+

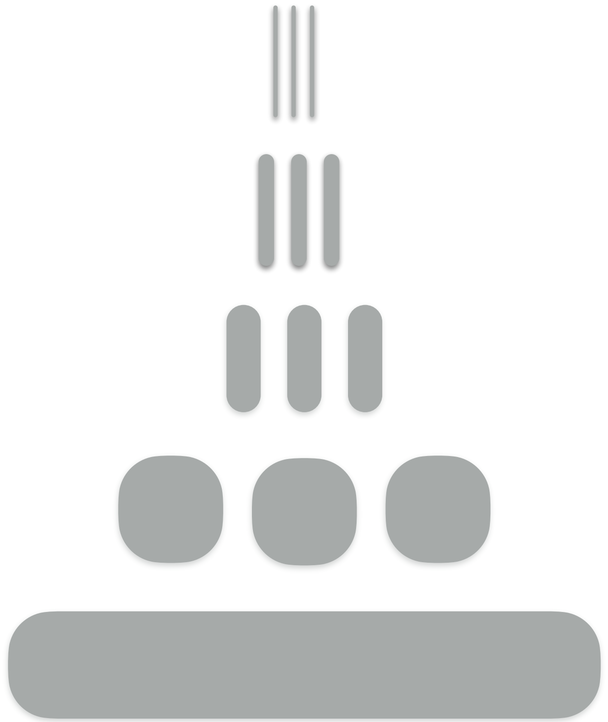
$O(R)$ **major**

Problem Analysis: Lazy Leveling





Problem Analysis: Lazy Leveling



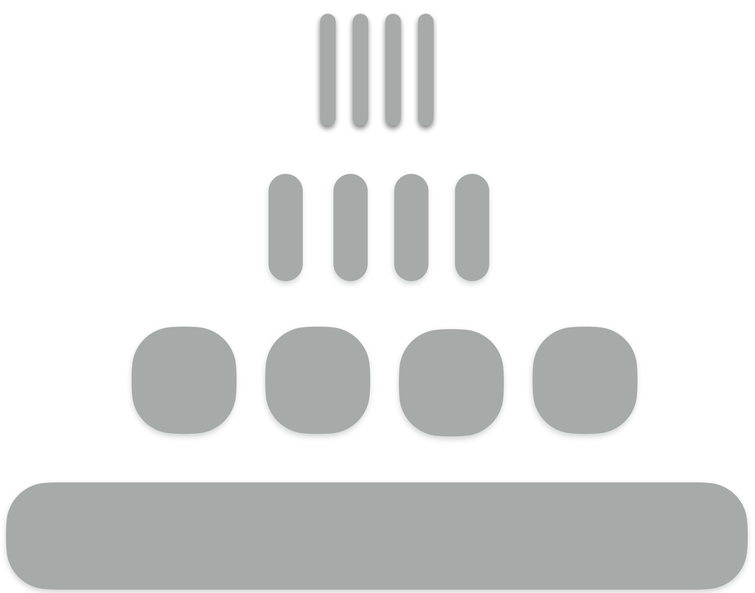
ratio ***R***

O($\log_{R}(N)$) **minor**

+

O(***R***) major

Problem Analysis: Lazy Leveling



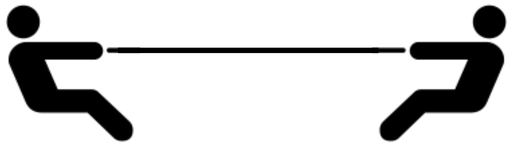
ratio ***R***

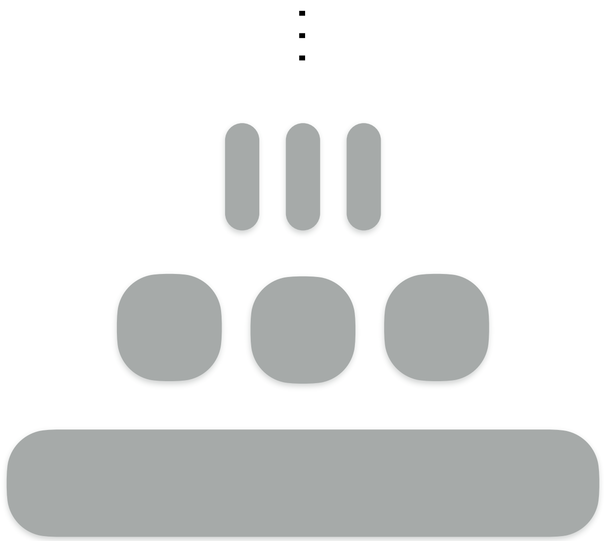
$O(\log_R(N))$ minor

+

$O(R)$ **major**

$$O(R + \log_R(N))$$

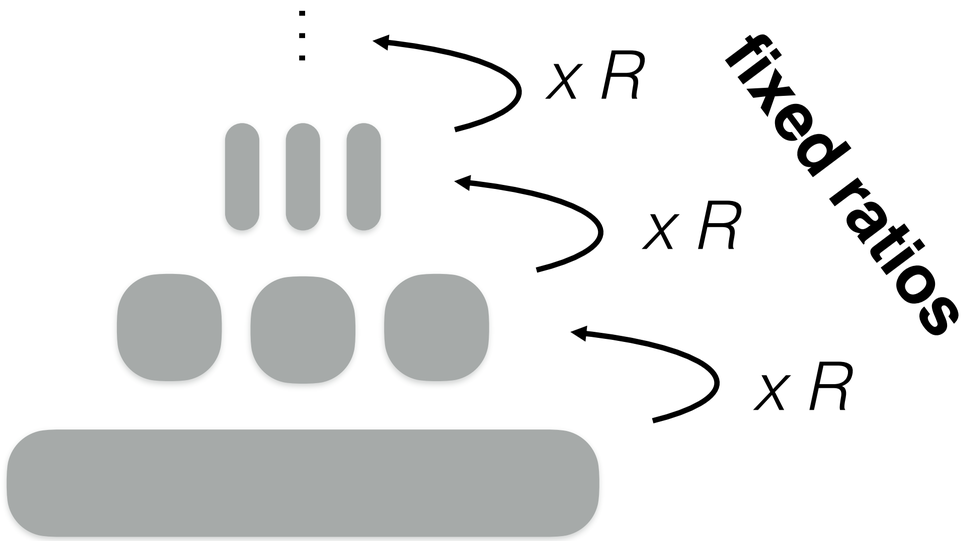
major  **minor**



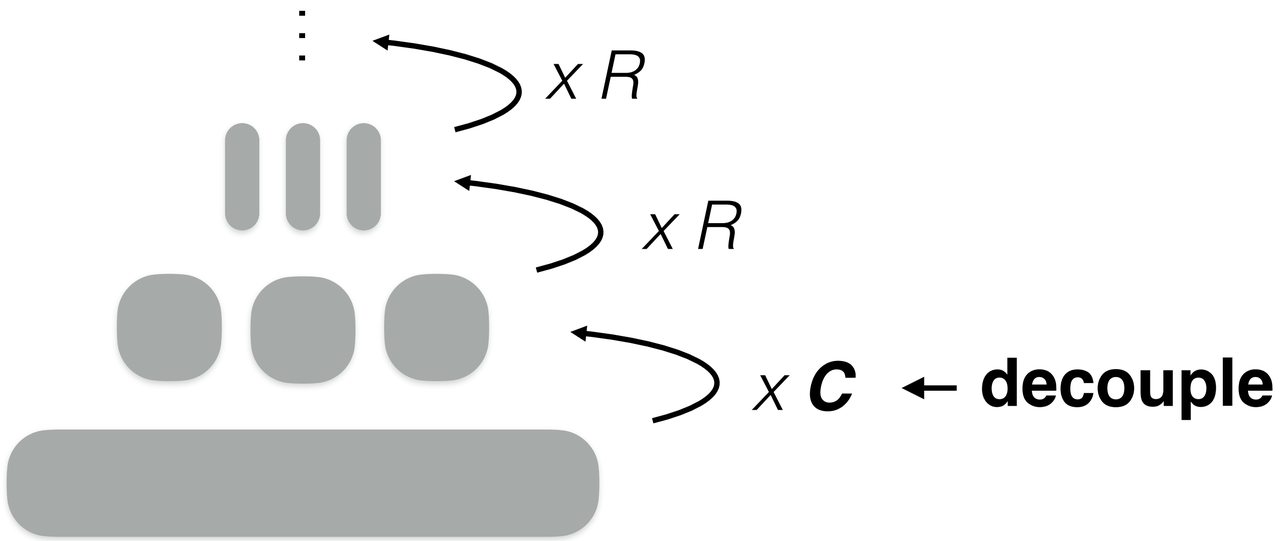
$$O(\ R + \log_R(N)\ )$$

major

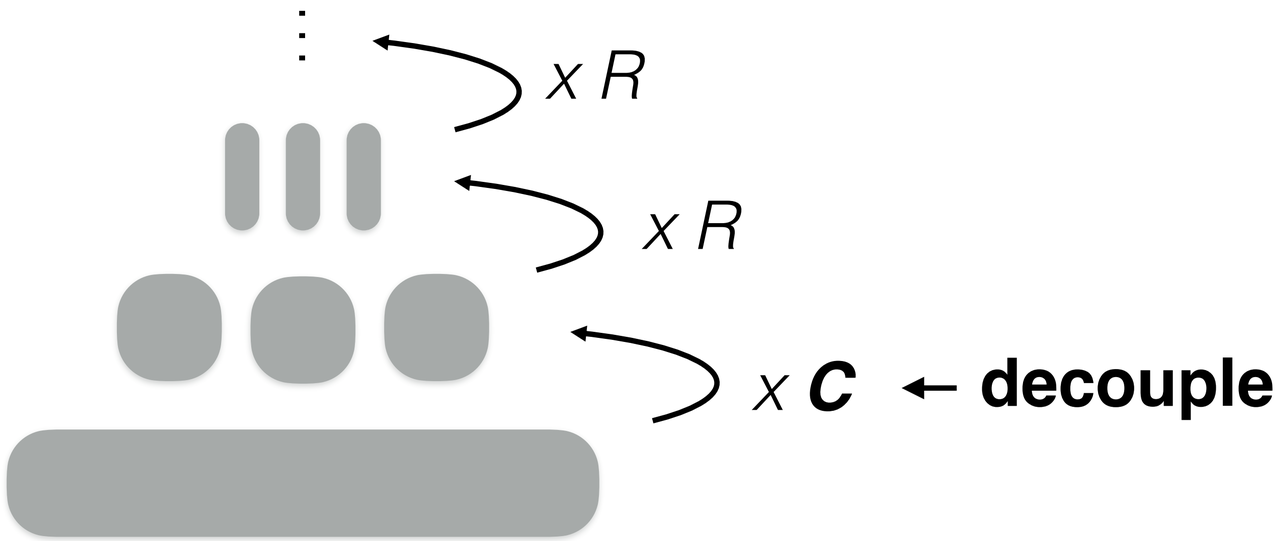
minor



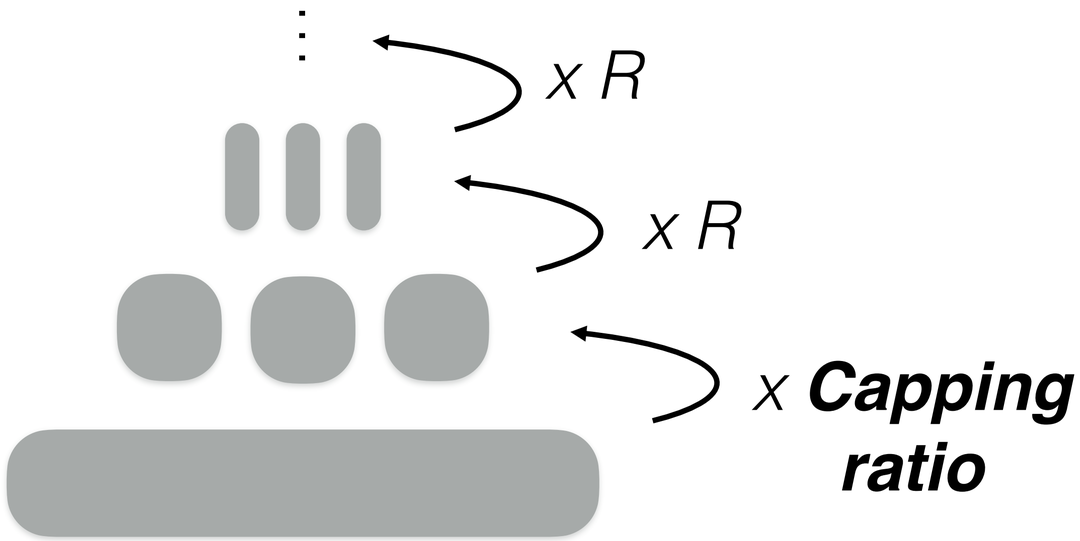
Squared Capped Lazy Leveling



SCLL

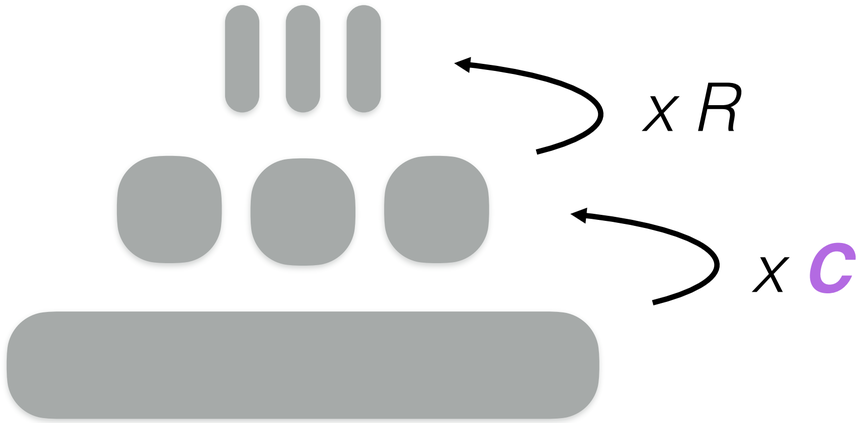


SCLL



SCLL

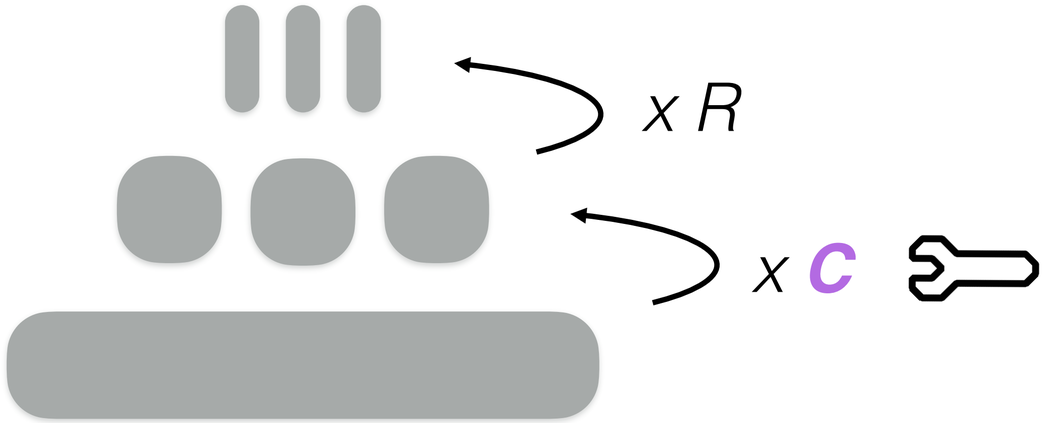
model:



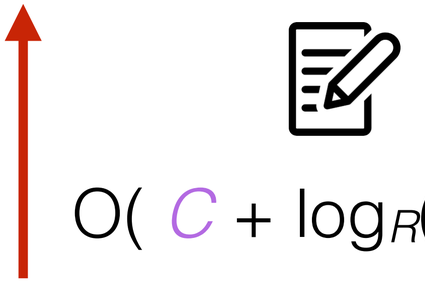
SCLL

$O(C + \log_R(N/C))$

$O(e^{-x} \cdot R^{1/C})$

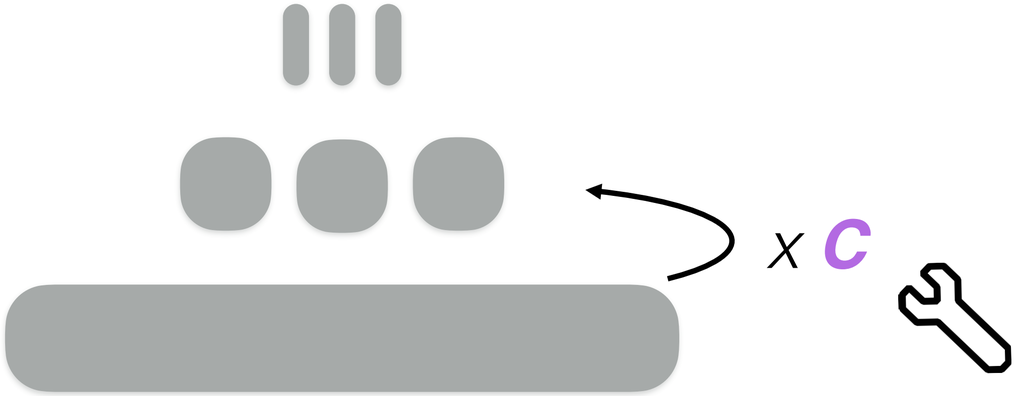


SCLL



$O(C + \log_R(N/C))$

$O(e^{-x} \cdot R^{1/C})$

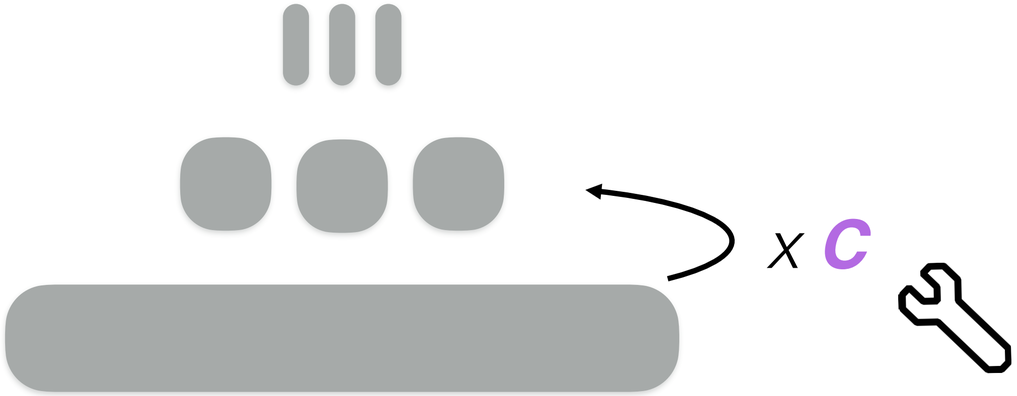


SCLL

$O(\ C + \log_R(N/C)\ )$

$O(e^{-x} \cdot R^{1/C})$

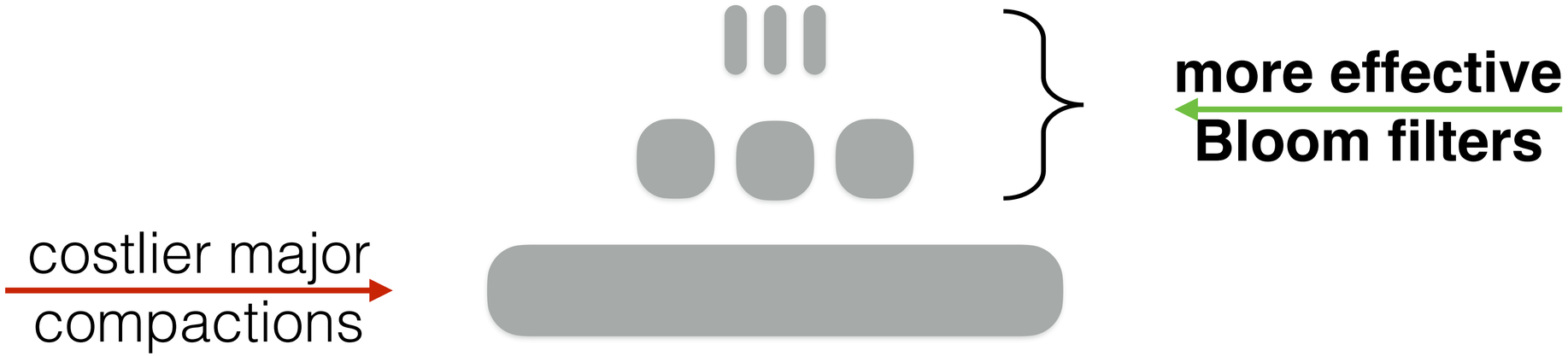
costlier major compactions



SCLL

$O(\ C + \log_R(N/C)\ )$

$O(e^{-x} \cdot R^{1/C})$



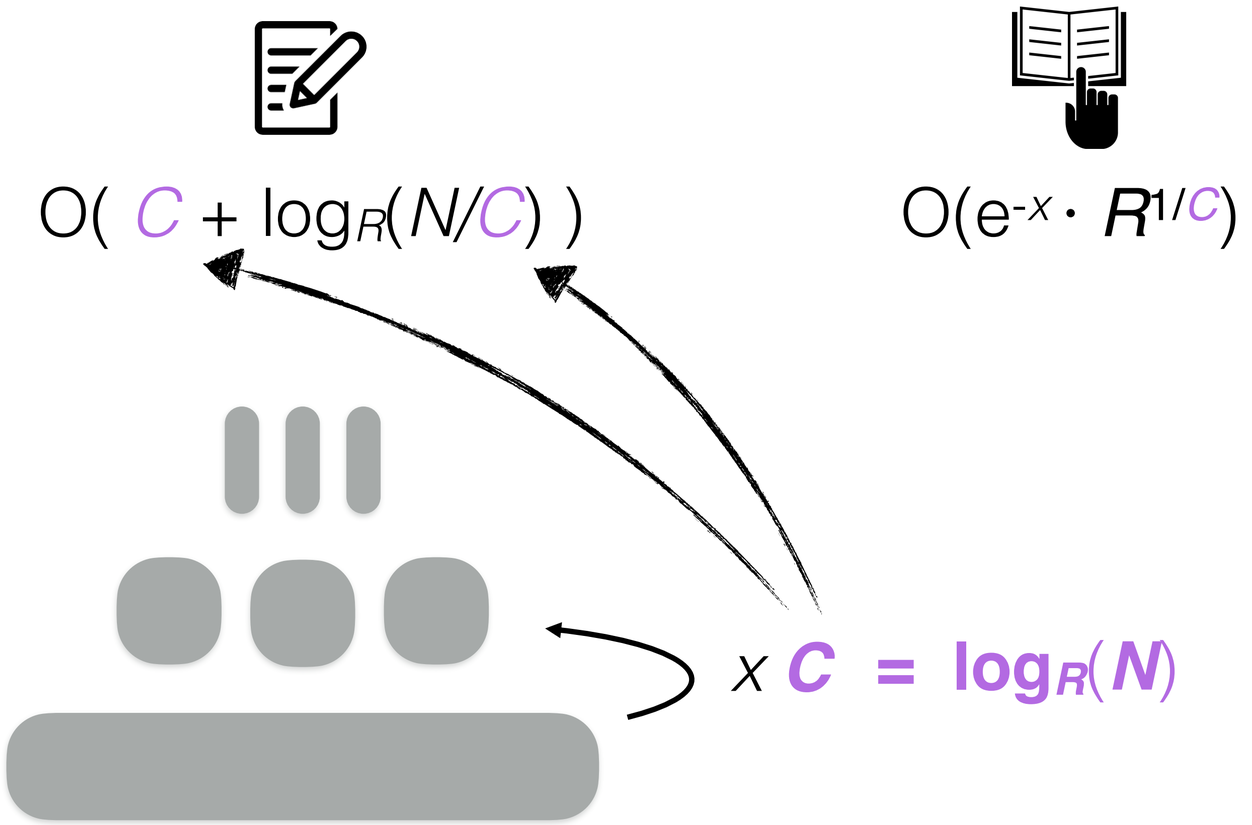
SCLL

$O(C + \log_R(N/C))$

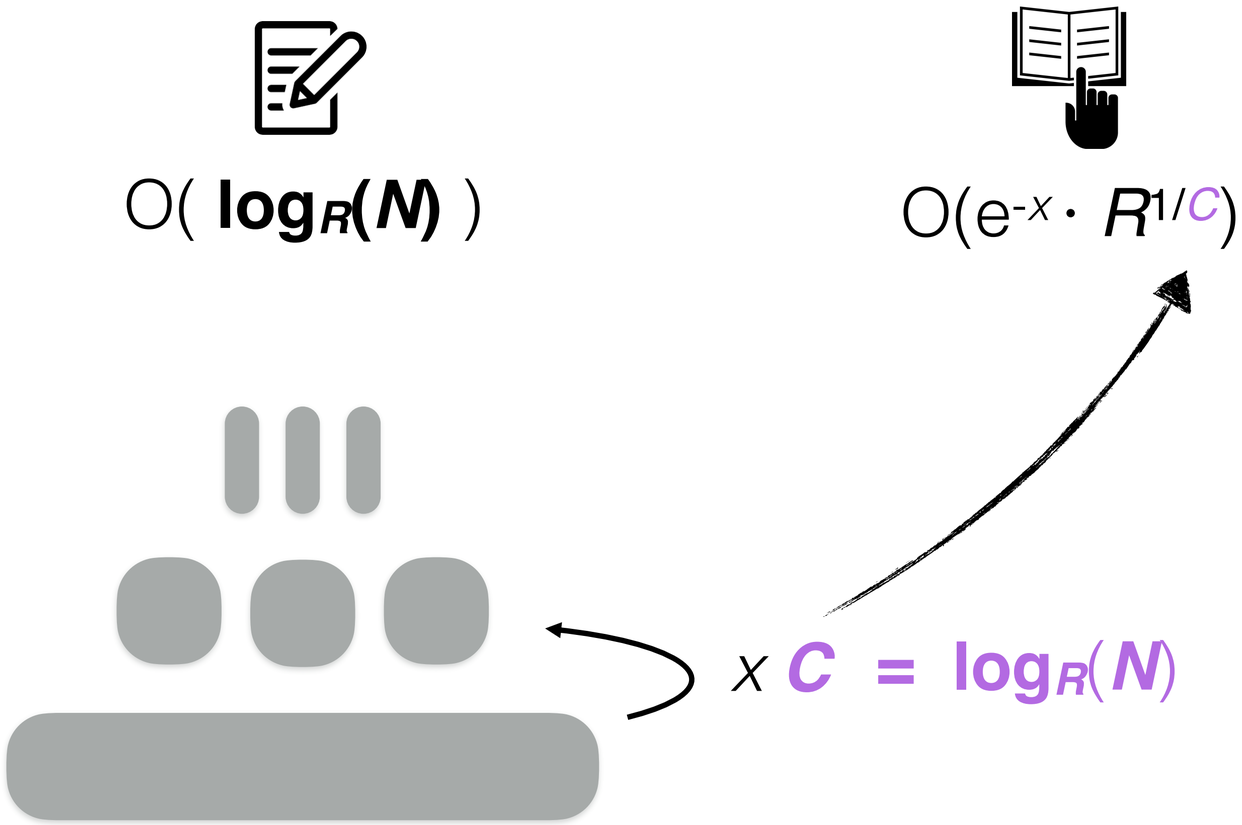
$O(e^{-x} \cdot R^{1/C})$

$x\ \boldsymbol{C} = \boldsymbol{\log_R(N)}$

SCLL



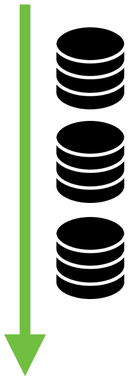
SCLL



SCLL

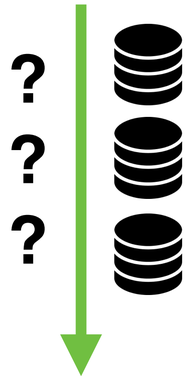
$O(\log_R(N))$

$O(e^{-x} \cdot R^{1/\log R(N)})$



$x\,C = \log_R(N)$

SCLL

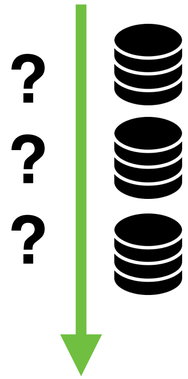


$O(\log_R(N))$

$O(e^{-x} \cdot R^{1/\log R(N)})$

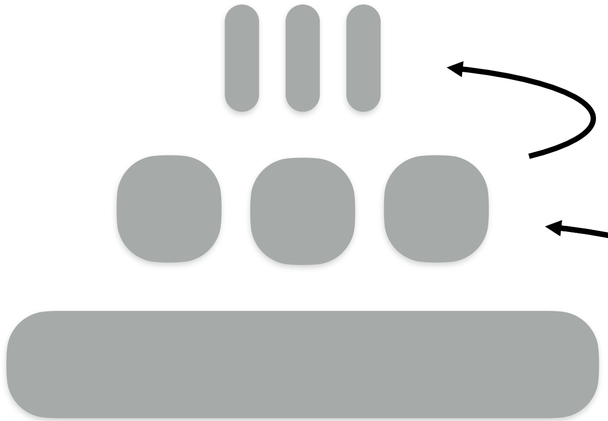
$x\,C = \log_R(N)$

SCLL



$O(\log_R(N))$

$O(e^{-x} \cdot R^{1/\log R(N)})$



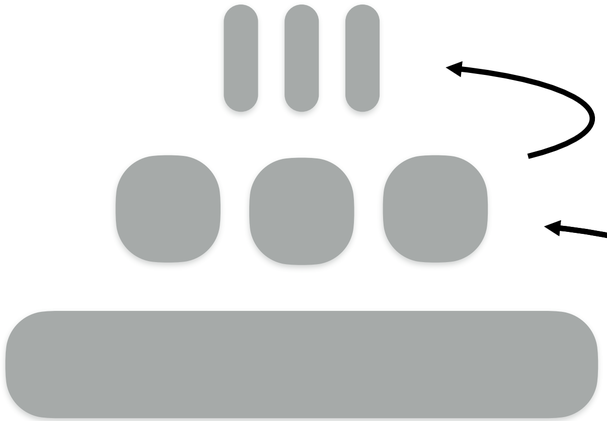
$x\ R$ **increase base ratio with** N

$x\ C = \log_R(N)$

SCLL

$O(\log_R(N))$

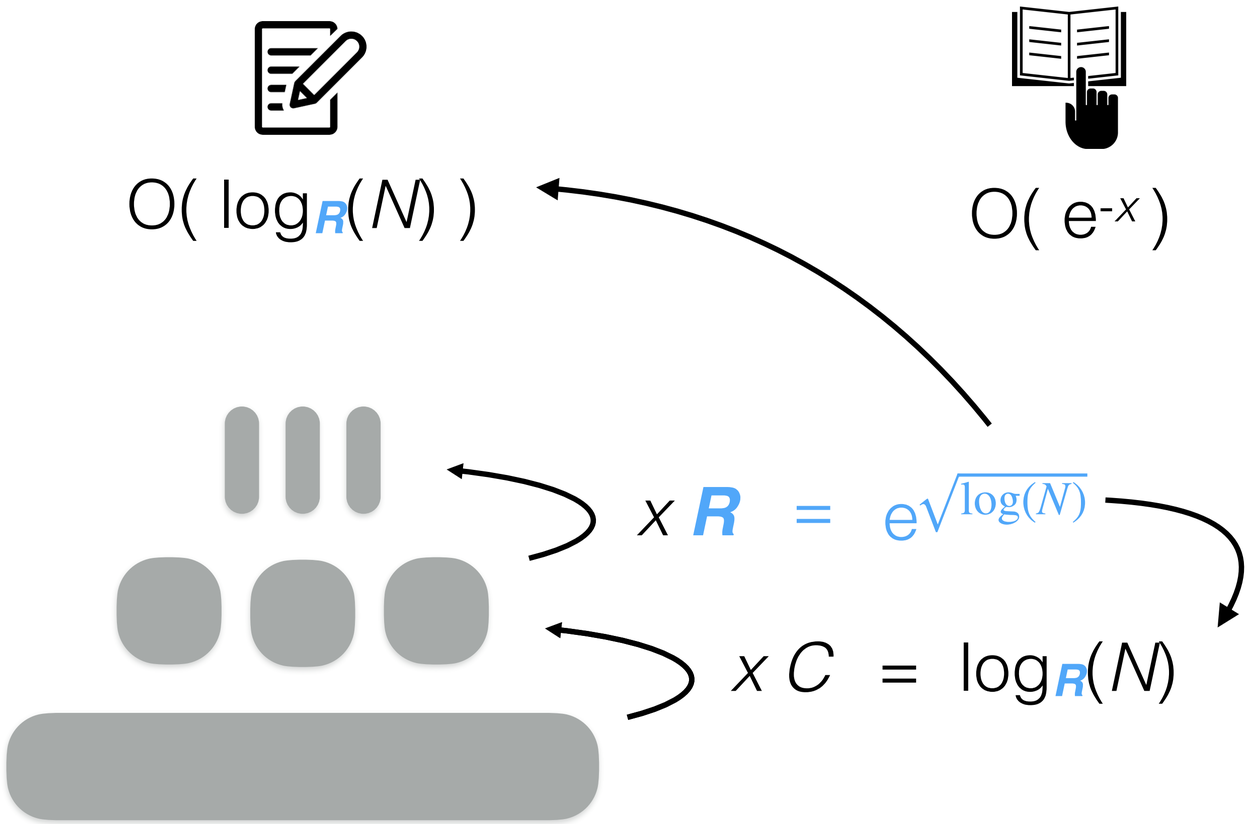
$O(e^{-x} \cdot R^{1/\log R(N)})$



$x\,R = e^{\sqrt{\log(N)}}$

$x\,C = \log_R(N)$

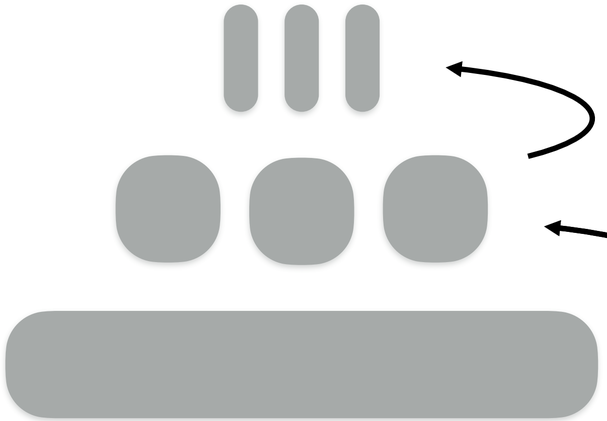
SCLL



SCLL

$O(\sqrt{\log(N)})$

$O(e^{-x})$



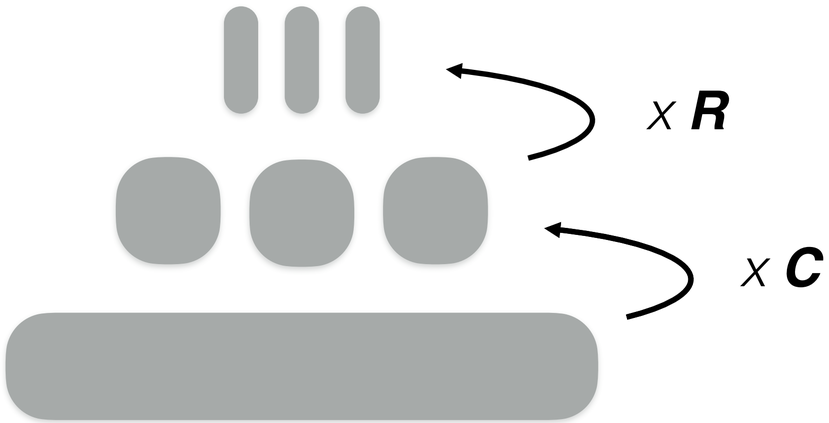
$x\,R = e^{\sqrt{\log(N)}}$

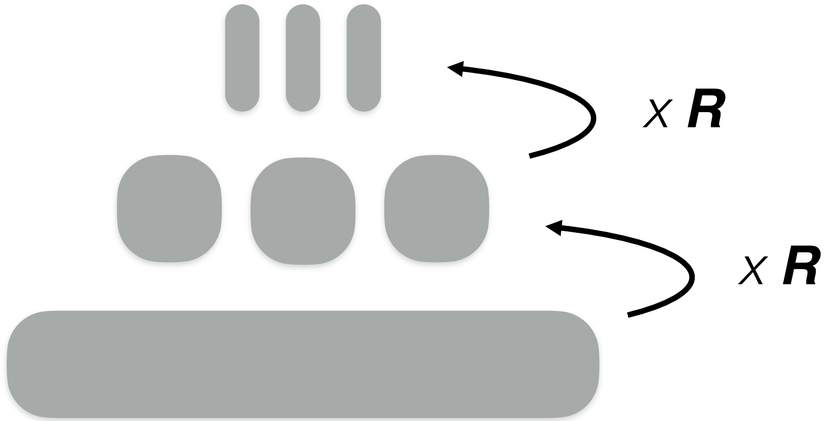
$x\,C = \sqrt{\log(N)}$

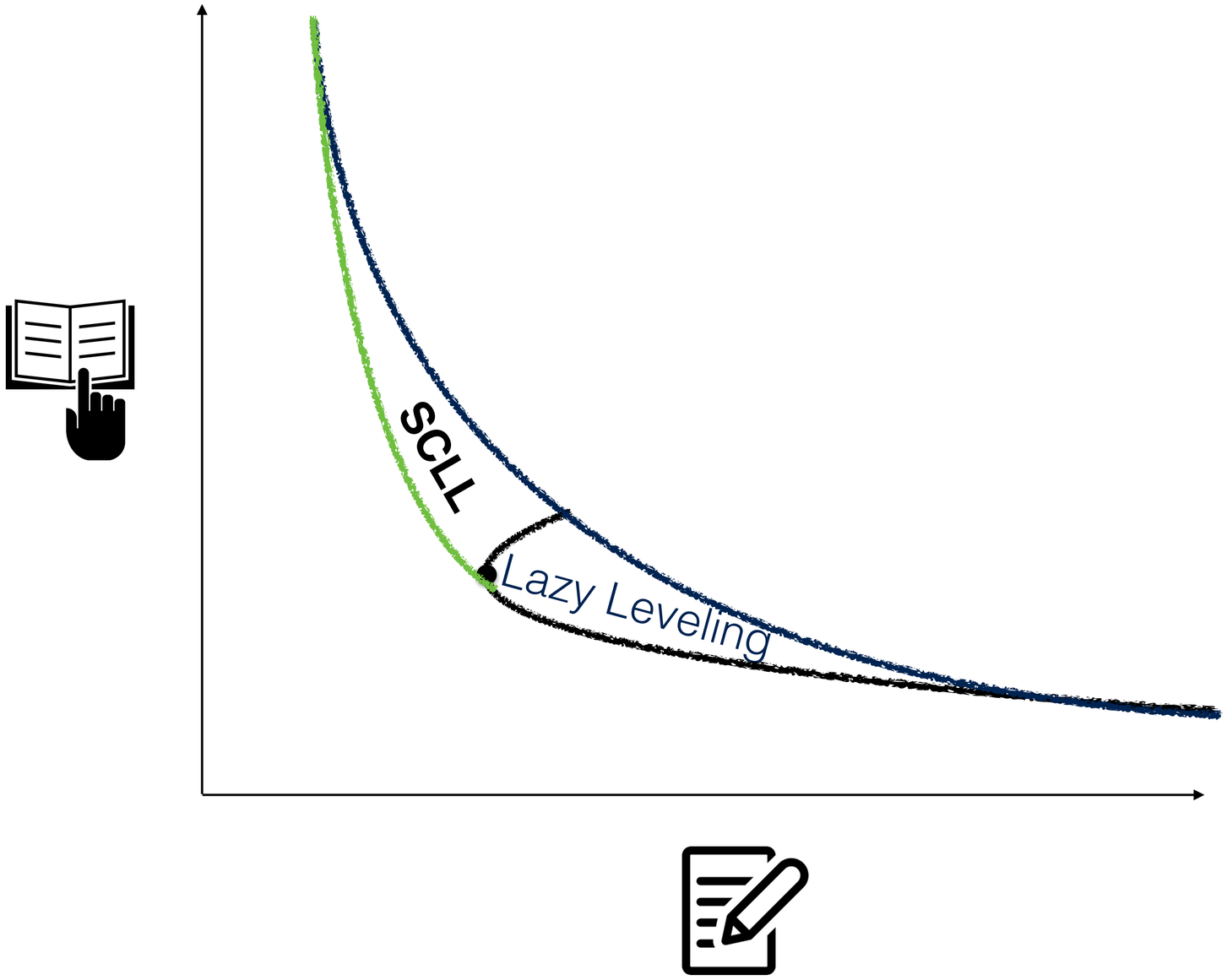
SCLL

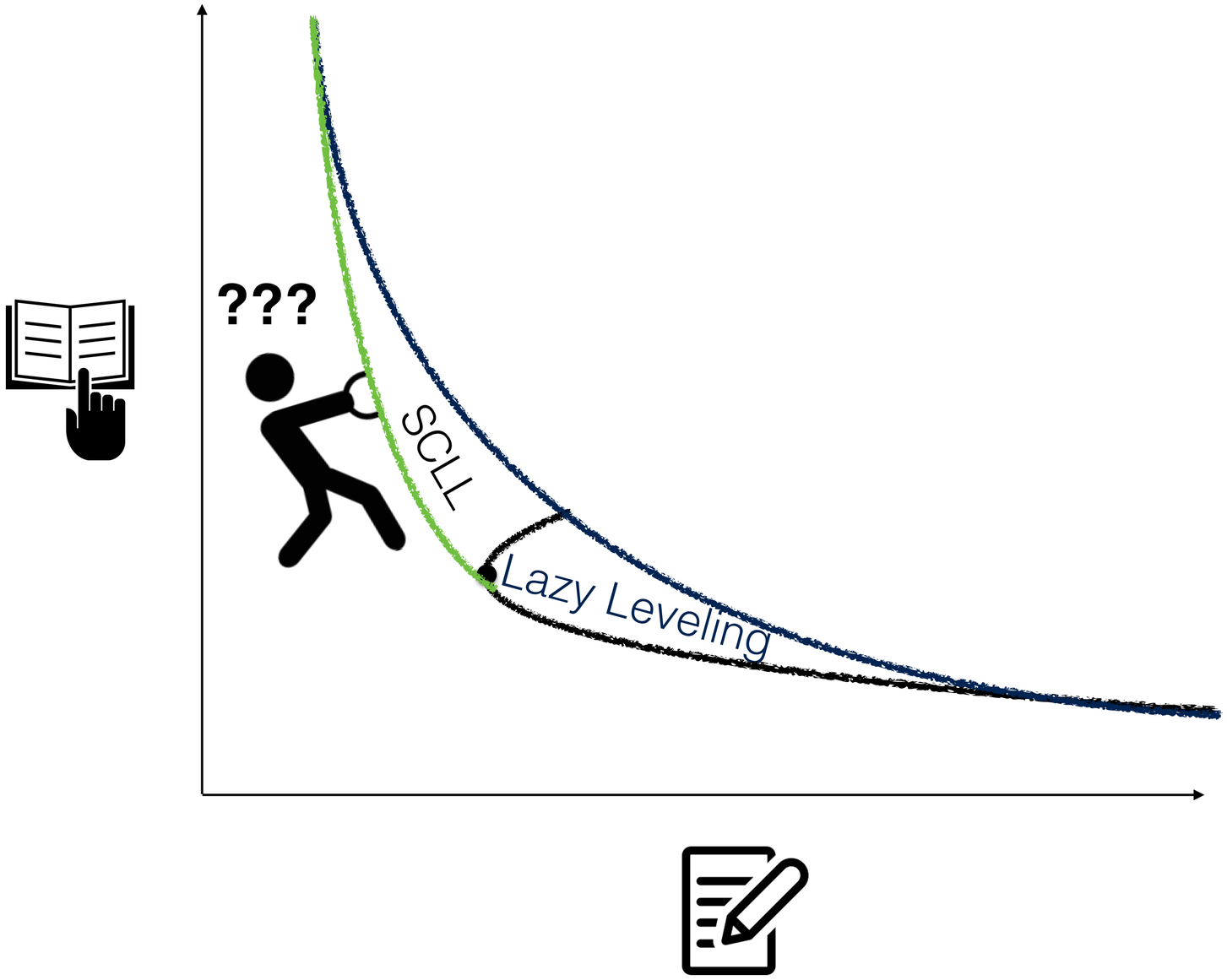
Traditional LSM

$O(\sqrt{\log(N)})$ < $O(\log(N))$

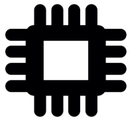


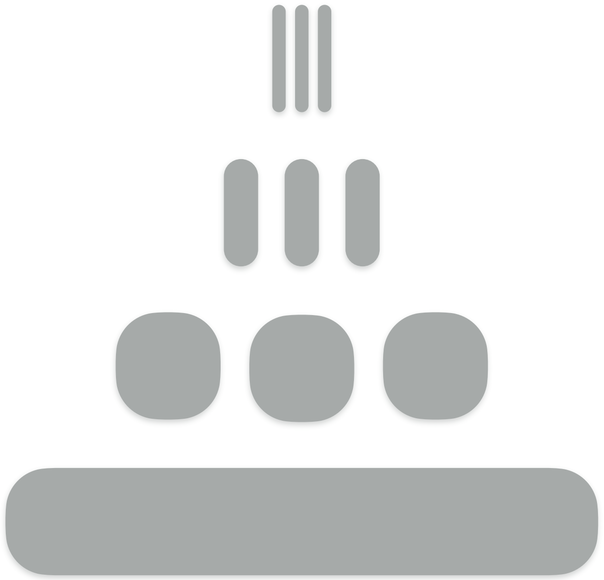



SCLL
Lazy Leveling

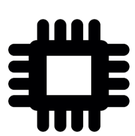

???
SCLL
Lazy Leveling

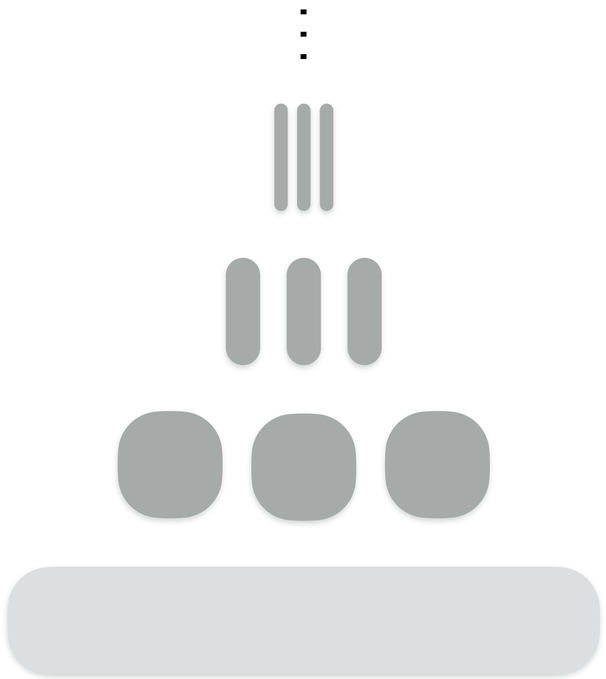
SCLL & Lazy Leveling **Cost Breakdown**



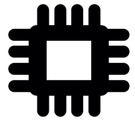


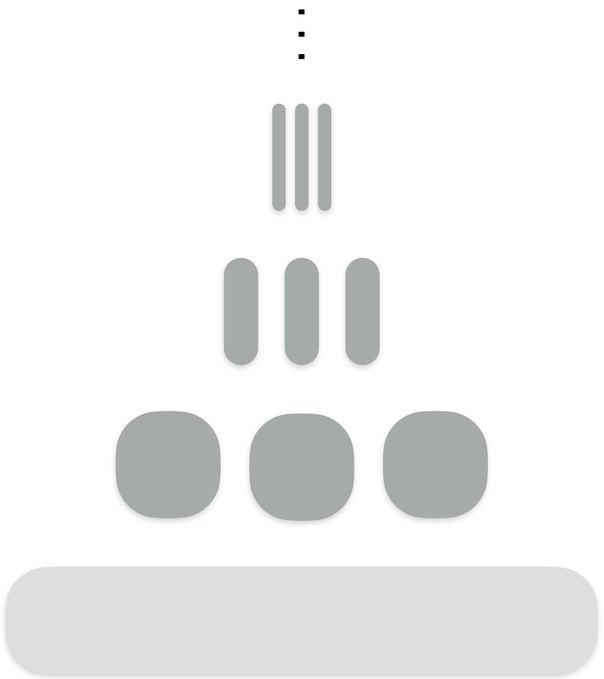
SCLL & Lazy Leveling **Cost Breakdown**





SCLL & Lazy Leveling **Cost Breakdown**





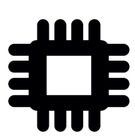
O(1)

O(1)

O(1)

constant

SCLL & Lazy Leveling **Cost Breakdown**

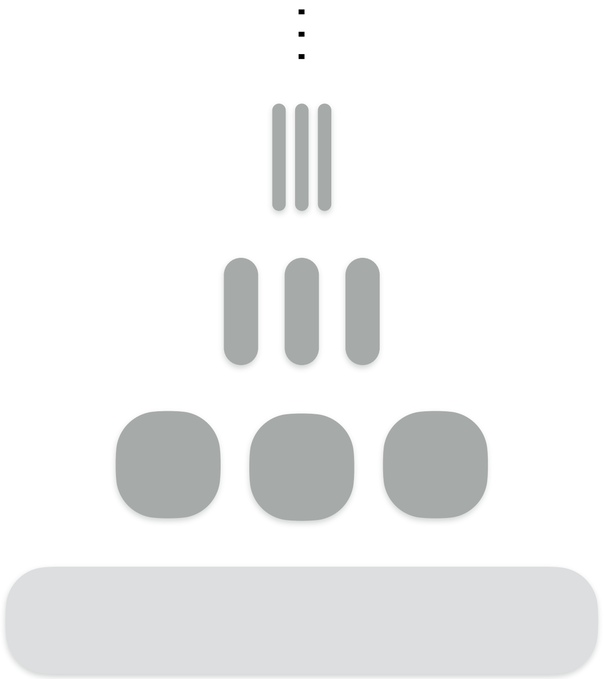


exponentially decreasing

$O(\frac{p}{R^2})$

$O(\frac{p}{R^1})$

$O(p)$



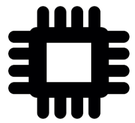
O(1)

O(1)

O(1)

constant

SCLL & Lazy Leveling **Cost Breakdown**



$O(\frac{3 \cdot M}{T^2})$

$O(\frac{2 \cdot M}{T^1})$

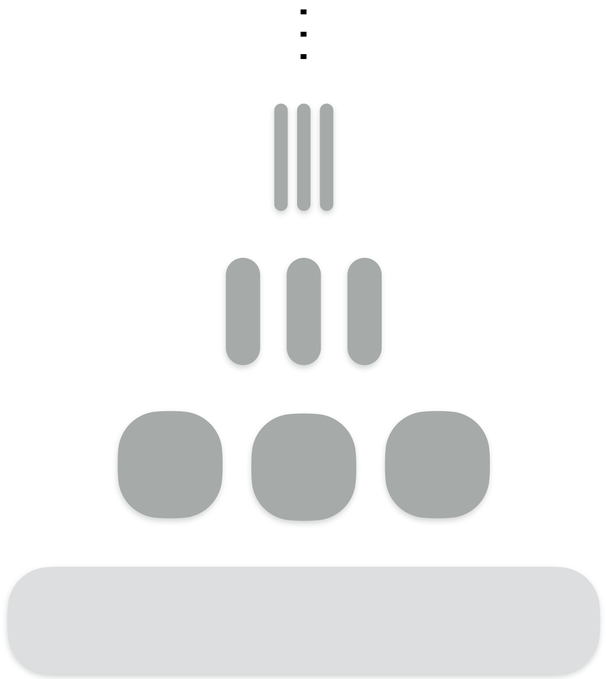
$O(M)$

exponentially decreasing

$O(\frac{p}{R^2})$

$O(\frac{p}{R^1})$

$O(p)$



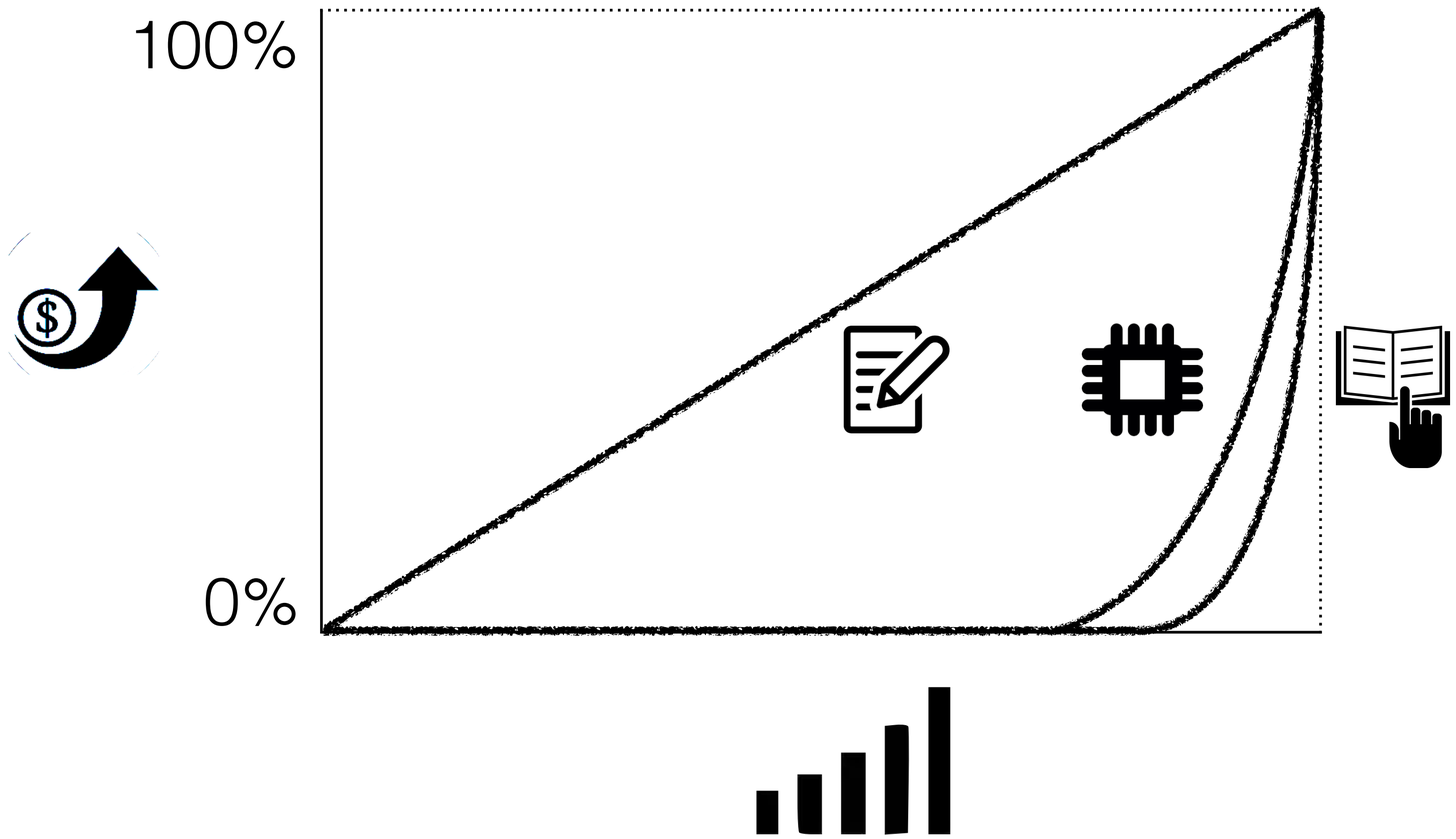
O(1)

O(1)

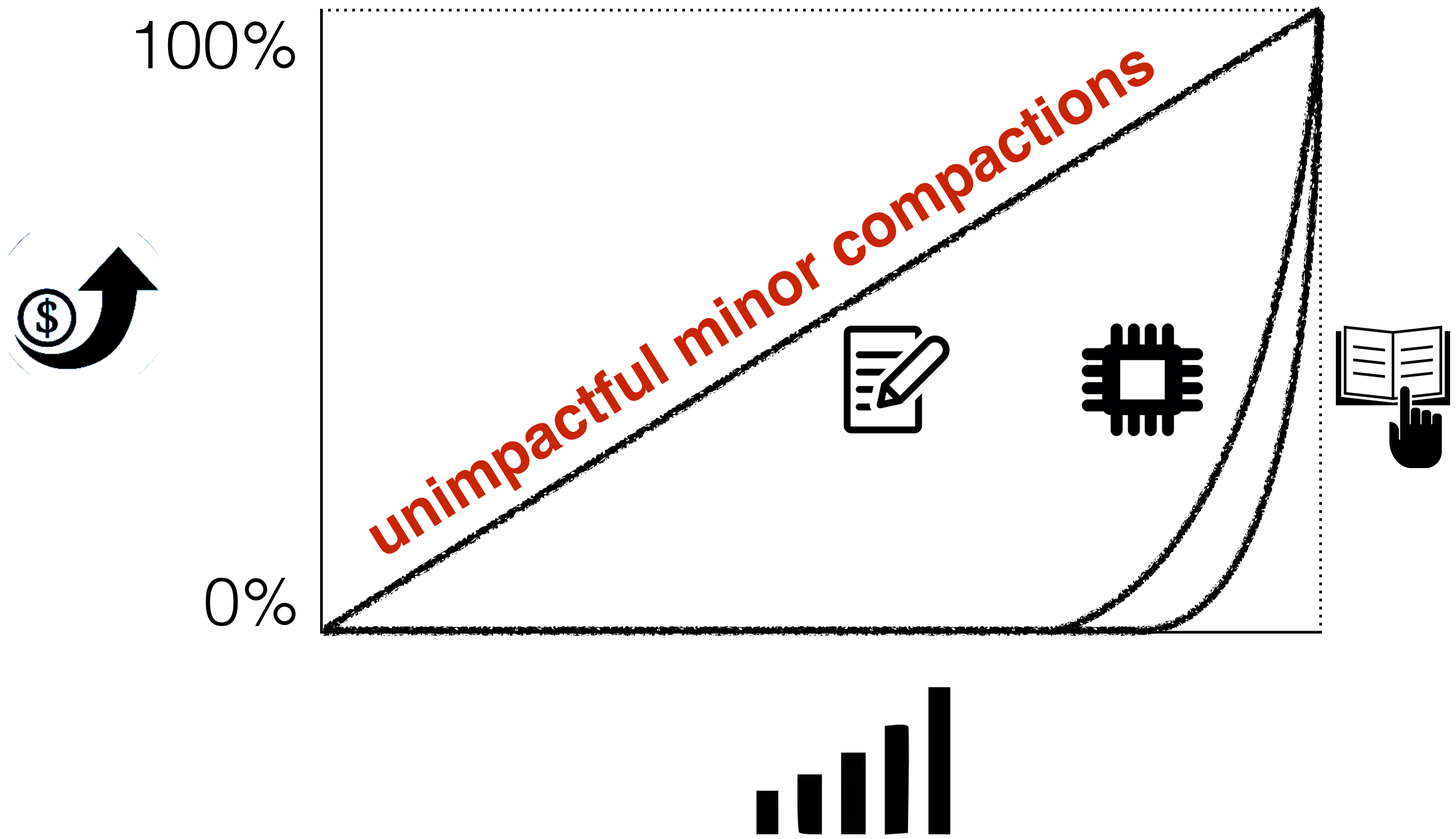
O(1)

constant

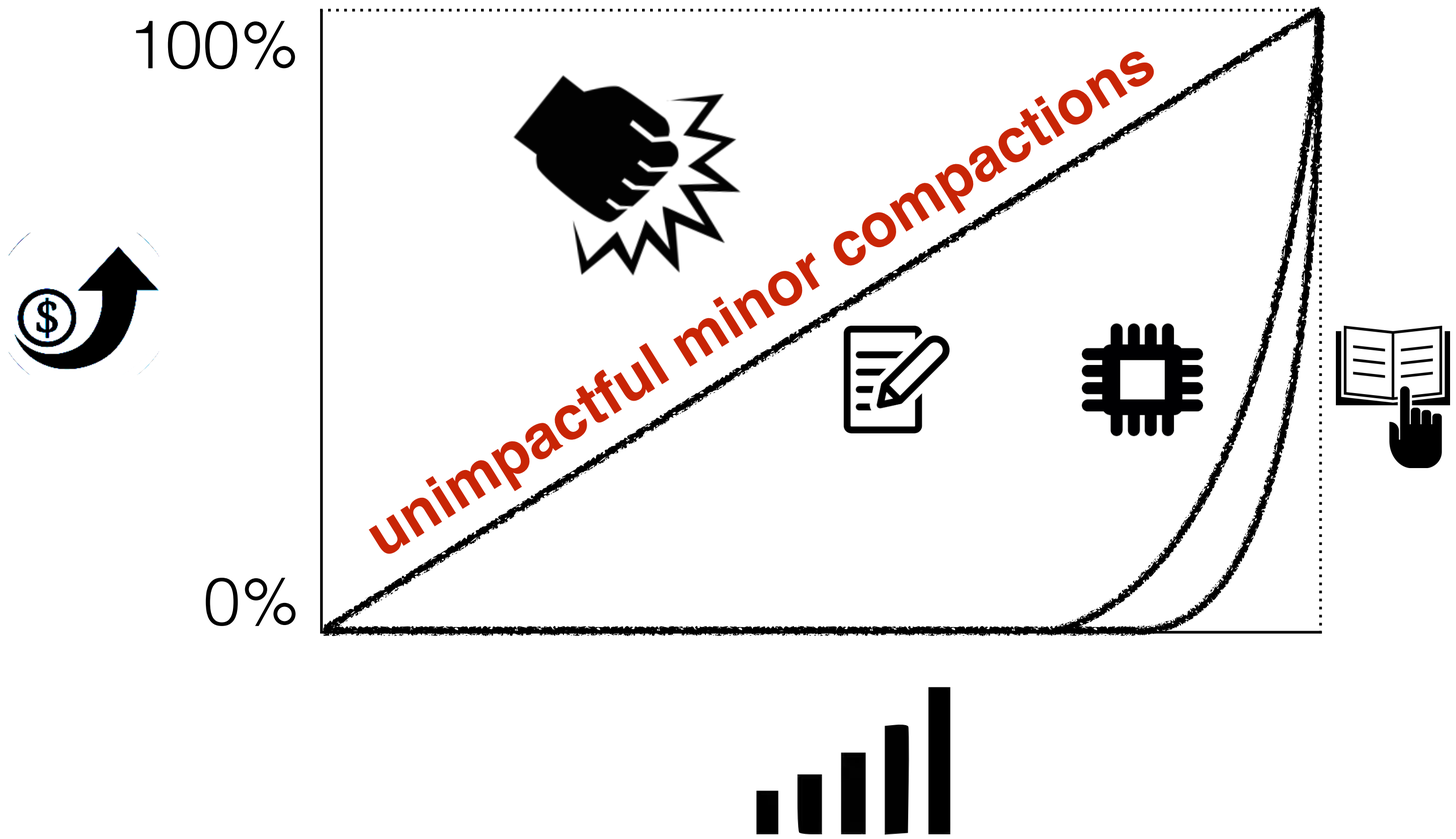
SCLL & Lazy Leveling **Cost Breakdown**



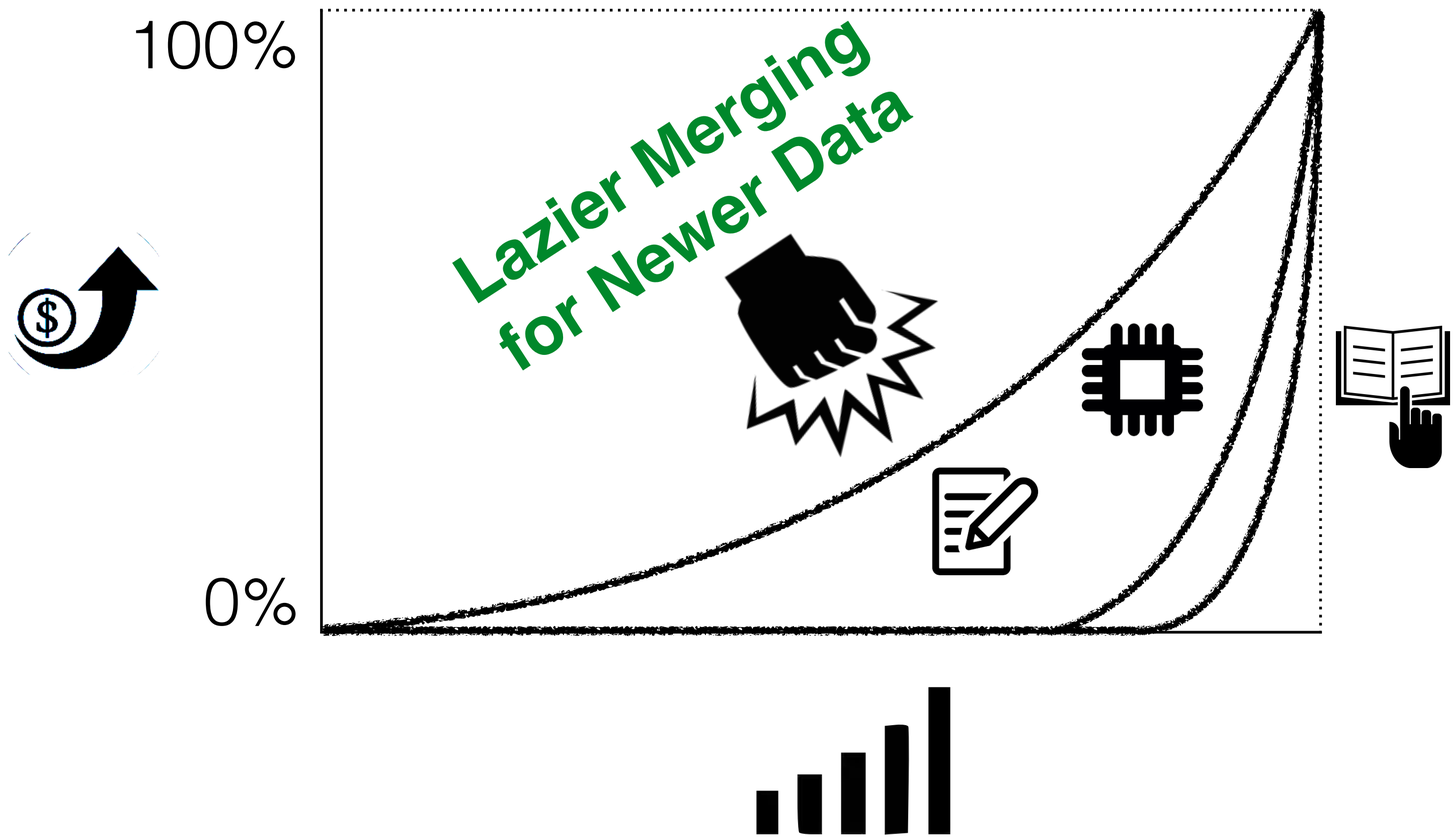
SCLL & Lazy Leveling **Cost Breakdown**



SCLL & Lazy Leveling **Cost Breakdown**

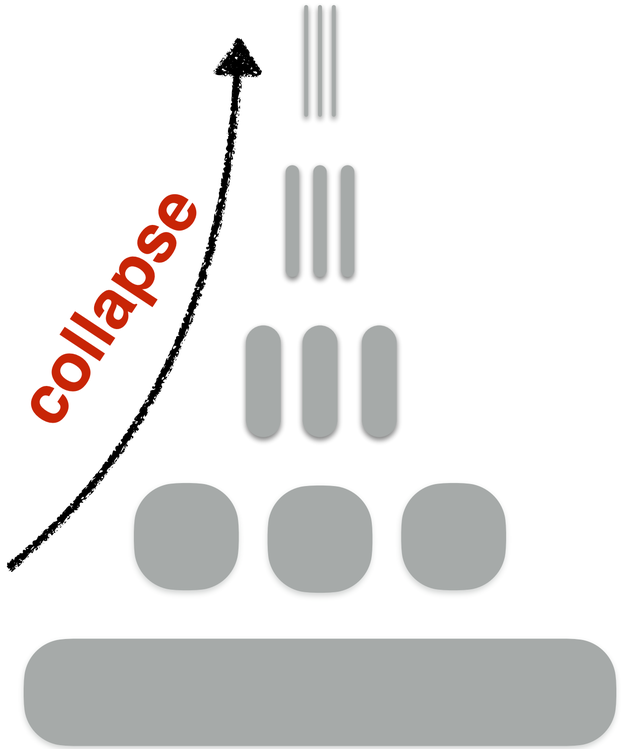


SCLL & Lazy Leveling **Cost Breakdown**

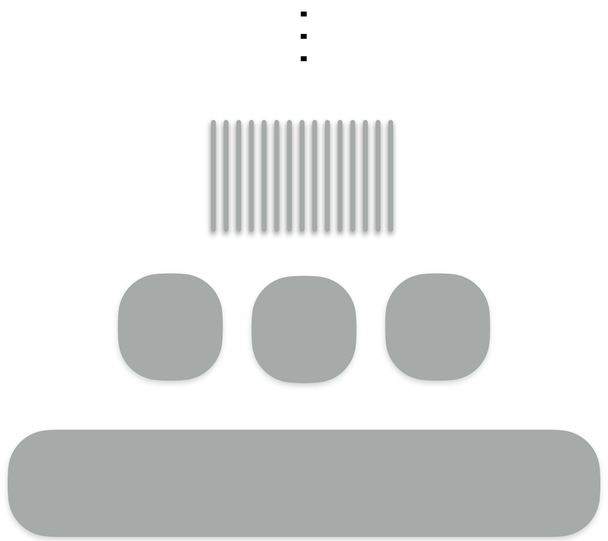


LSM-bush - Lazier Merging for Newer Data

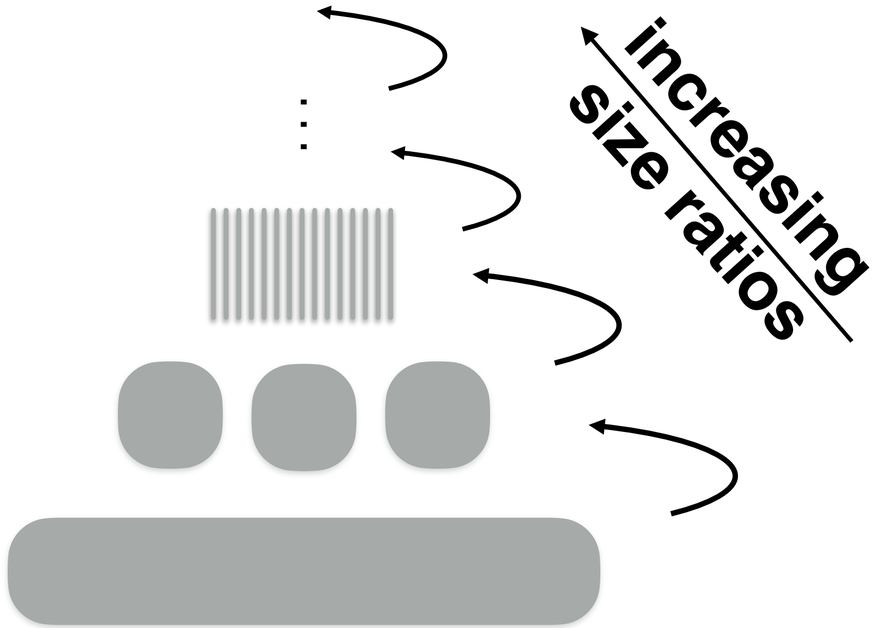
LSM-bush - Lazier Merging for Newer Data



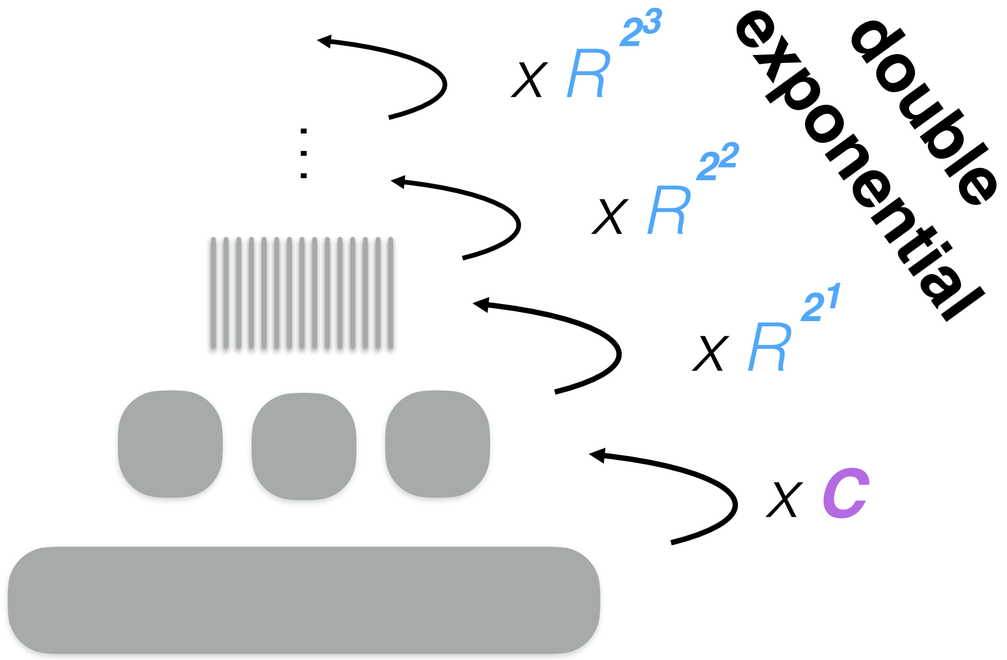
LSM-bush - Lazier Merging for Newer Data



LSM-bush - Lazier Merging for Newer Data

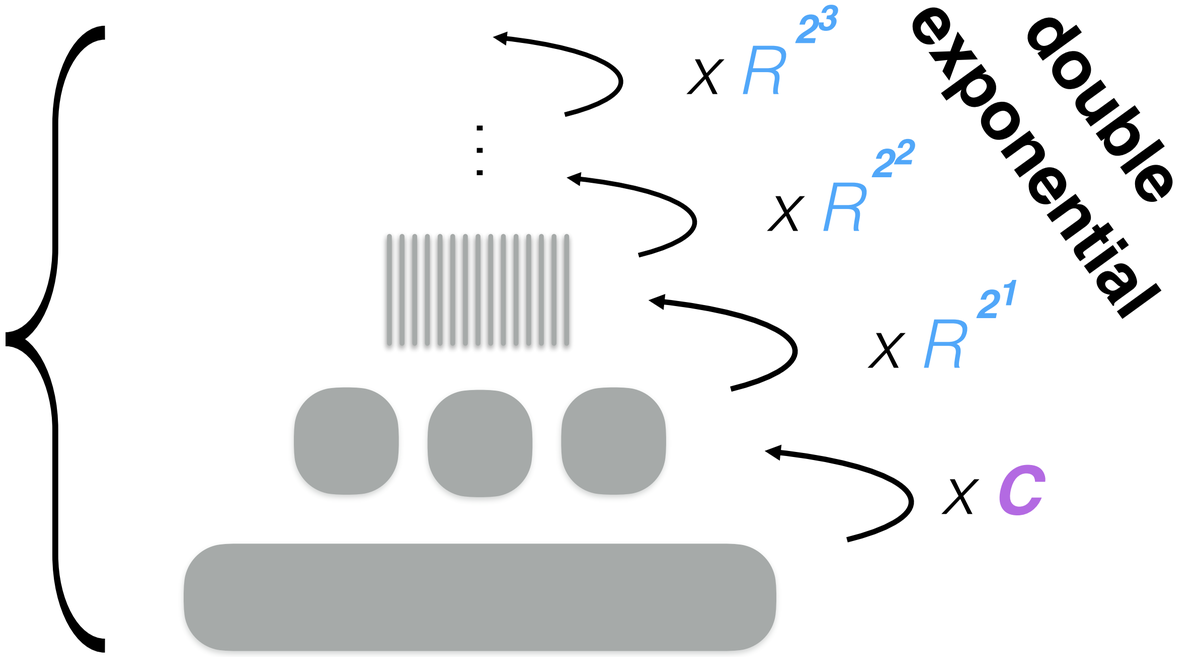


LSM-bush - Lazier Merging for Newer Data

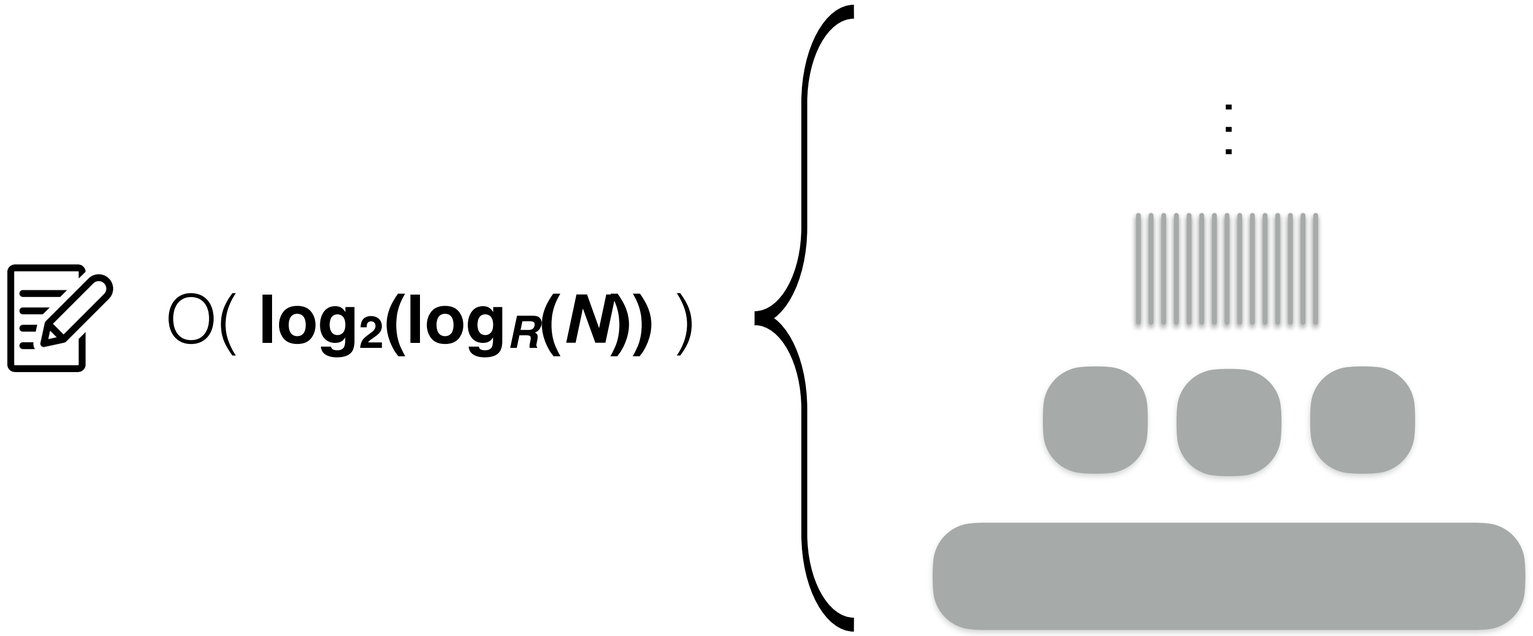


LSM-bush - Lazier Merging for Newer Data

O($\log_2(\log_R(N))$) levels



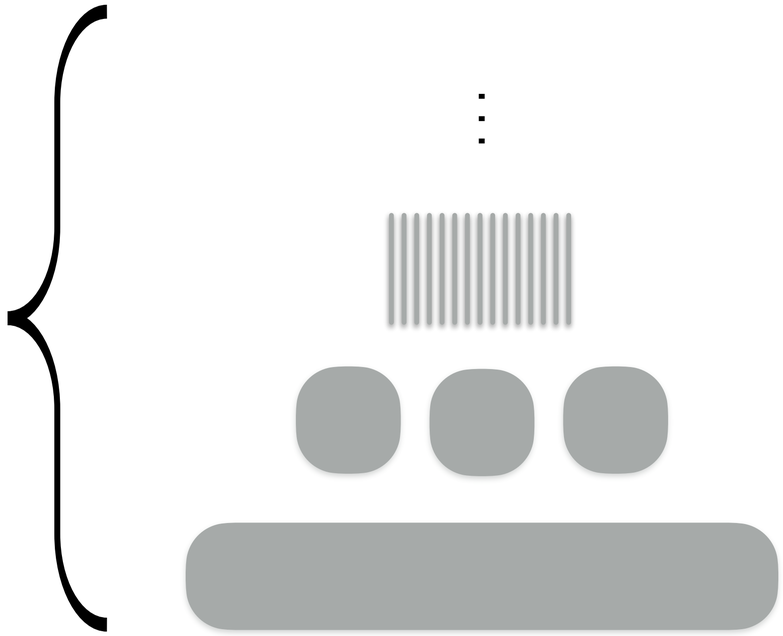
LSM-bush - Lazier Merging for Newer Data



O($\mathbf{\log_2(\log_R(N))}$)

LSM-bush - Lazier Merging for Newer Data

$O(\log_2(\log_R(N)))$



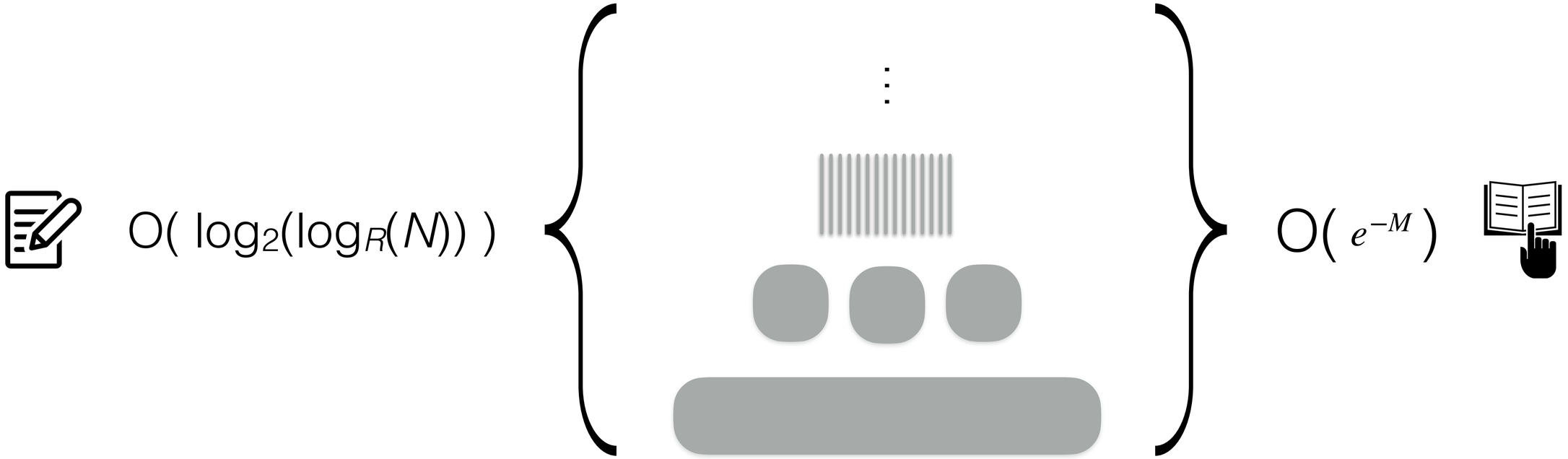
$O(\frac{e^{-M}}{R^{2^2}})$

$O(\frac{e^{-M}}{R^{2^1}})$

$O(\frac{e^{-M}}{R^{2^0}})$

$O(e^{-M})$

LSM-bush - Lazier Merging for Newer Data



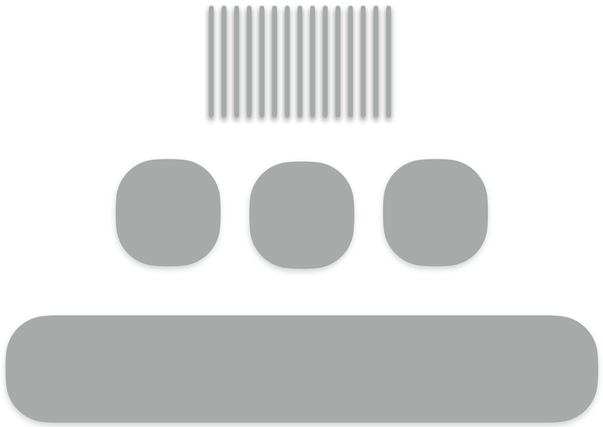
LSM-bush

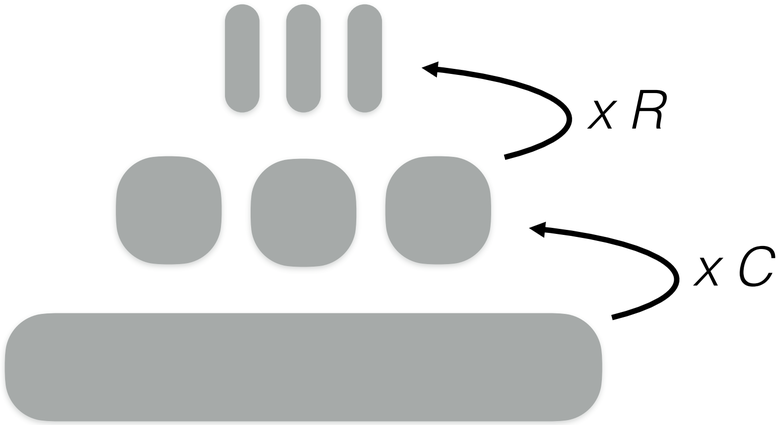
SCLL

O(**log(log(*M*))**)

<

O($\sqrt{\textbf{log(}\textbf{\textit{M}}\textbf{)}}$)



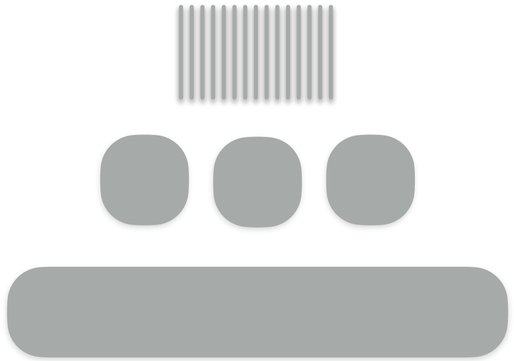


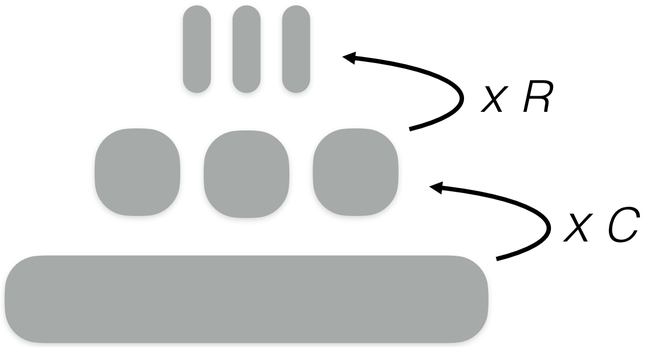
LSM-bush

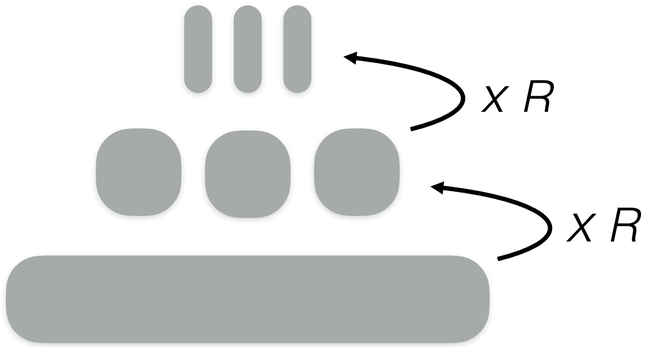
SCLL

LSM-tree

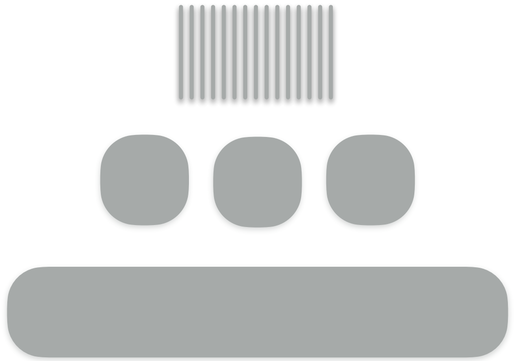
$O(\log(\log(N)))$ < $O(\sqrt{\log(N)})$ < $O(\log(N))$

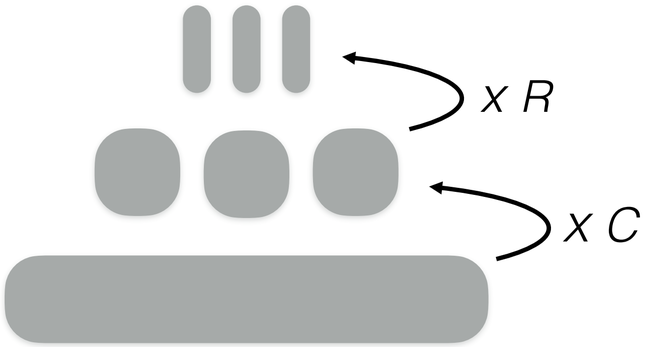


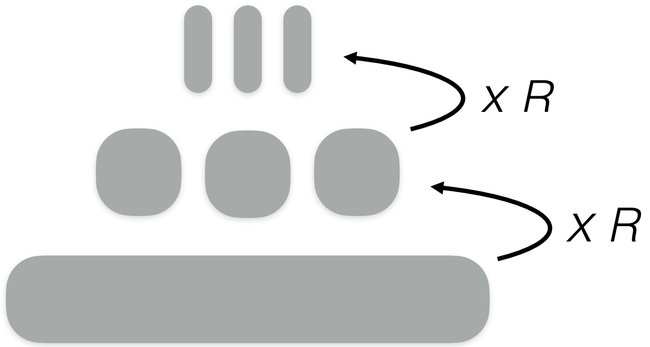




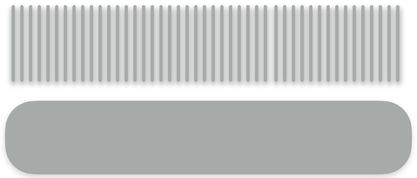
LSM-bush		SCLL		LSM-tree
$O(e^{-M})$	=	$O(e^{-M})$	=	$O(e^{-M})$
$O(\log(\log(N)))$	<	$O(\sqrt{\log(N)})$	<	$O(\log(N))$

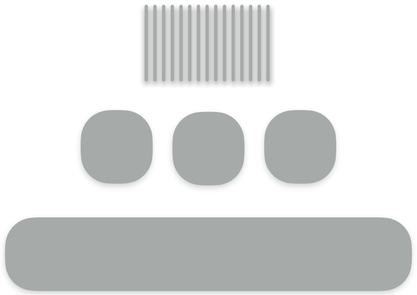


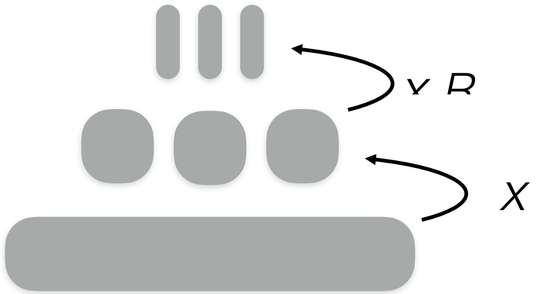


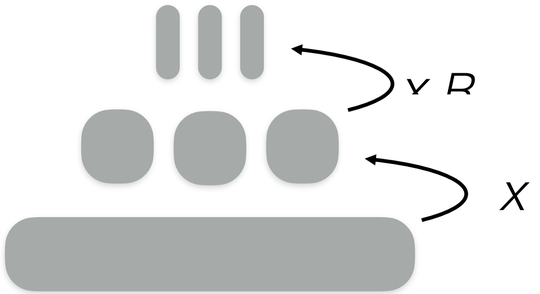


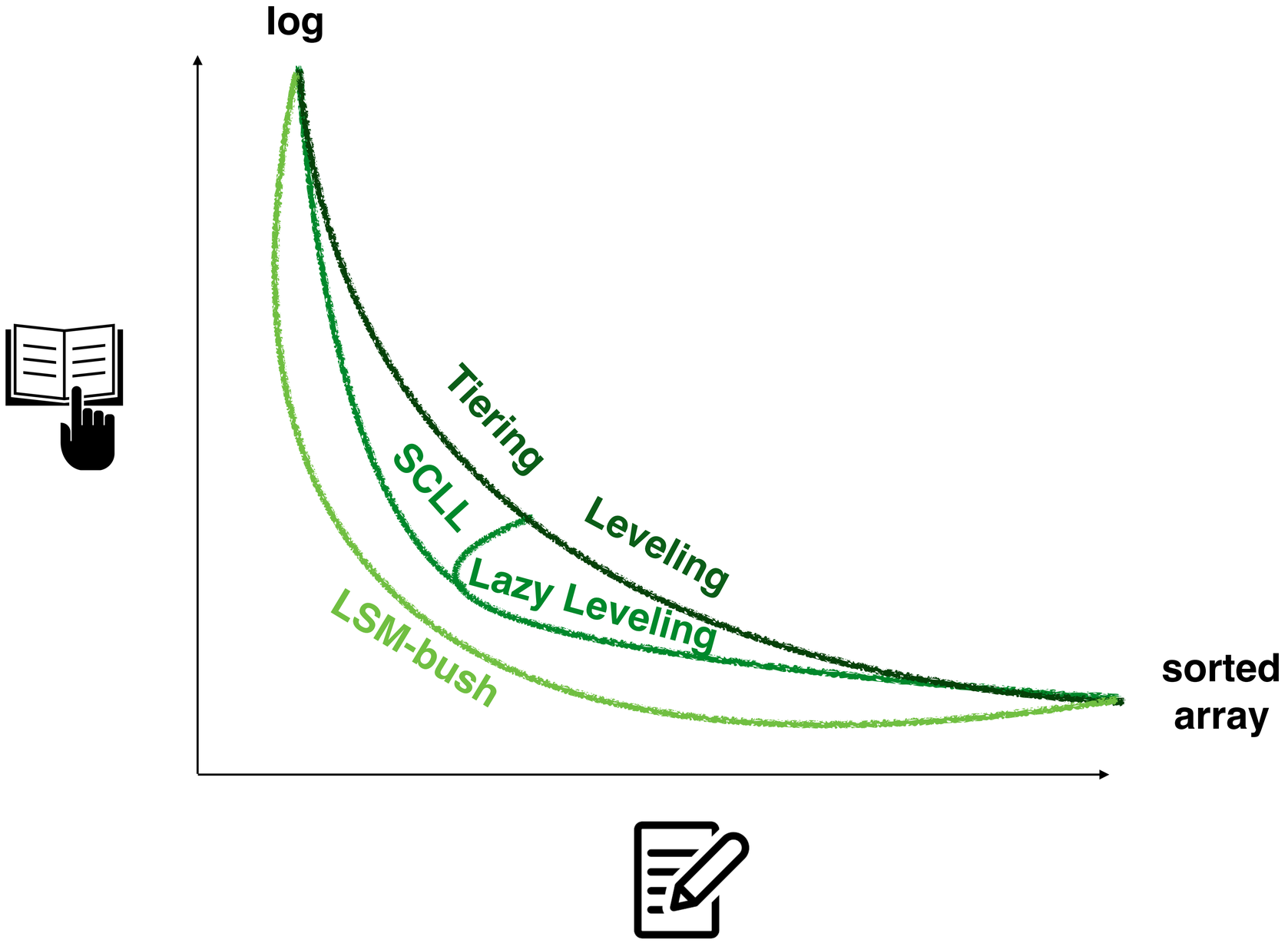
	log		LSM-bush		SCLL		LSM-tree
	$O(N \cdot e^{-M})$	>	$O(e^{-M})$	=	$O(e^{-M})$	=	$O(e^{-M})$
	$O(1)$	<	$O(\log(\log(N)))$	<	$O(\sqrt{\log(N)})$	<	$O(\log(N))$

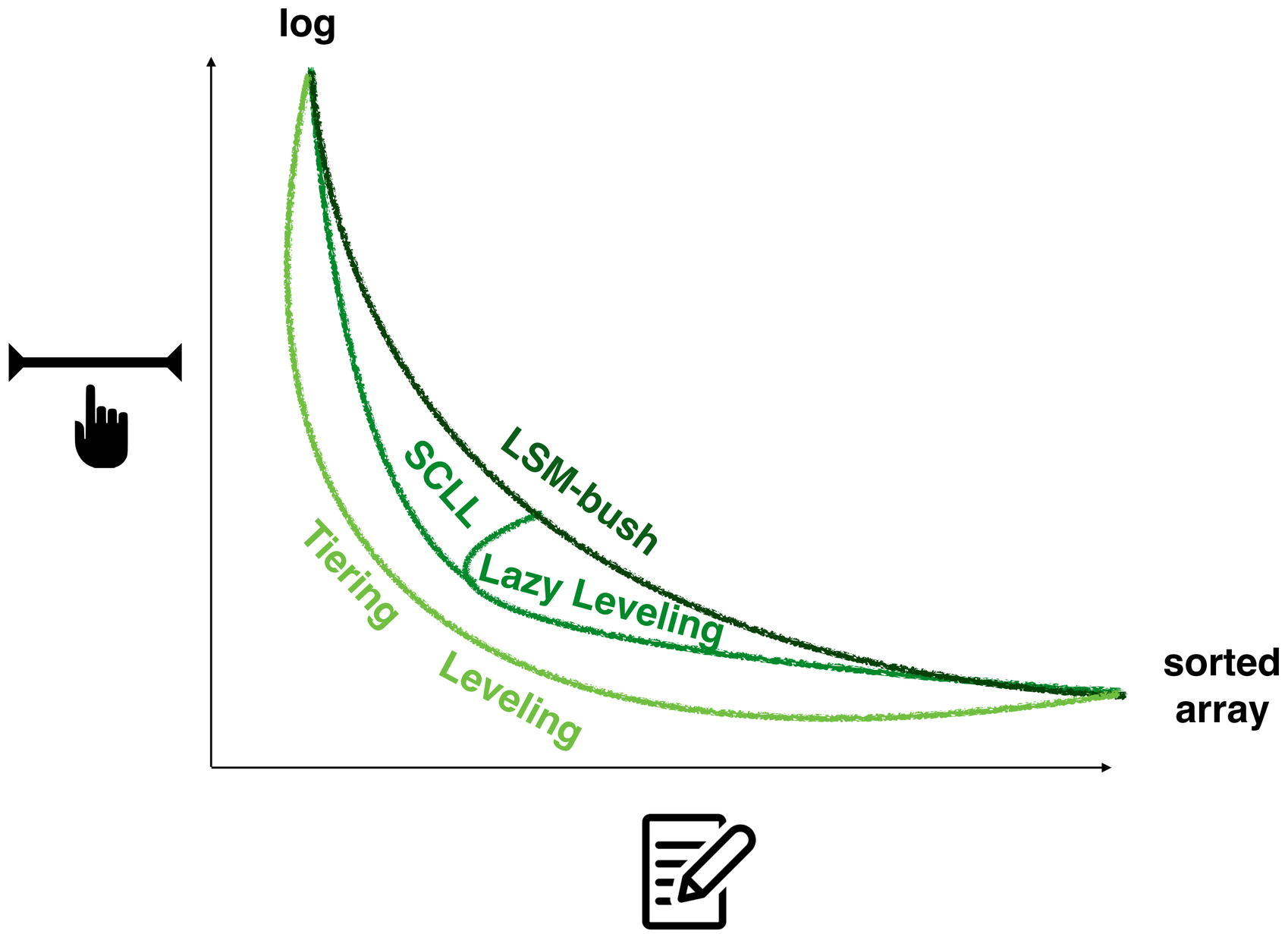


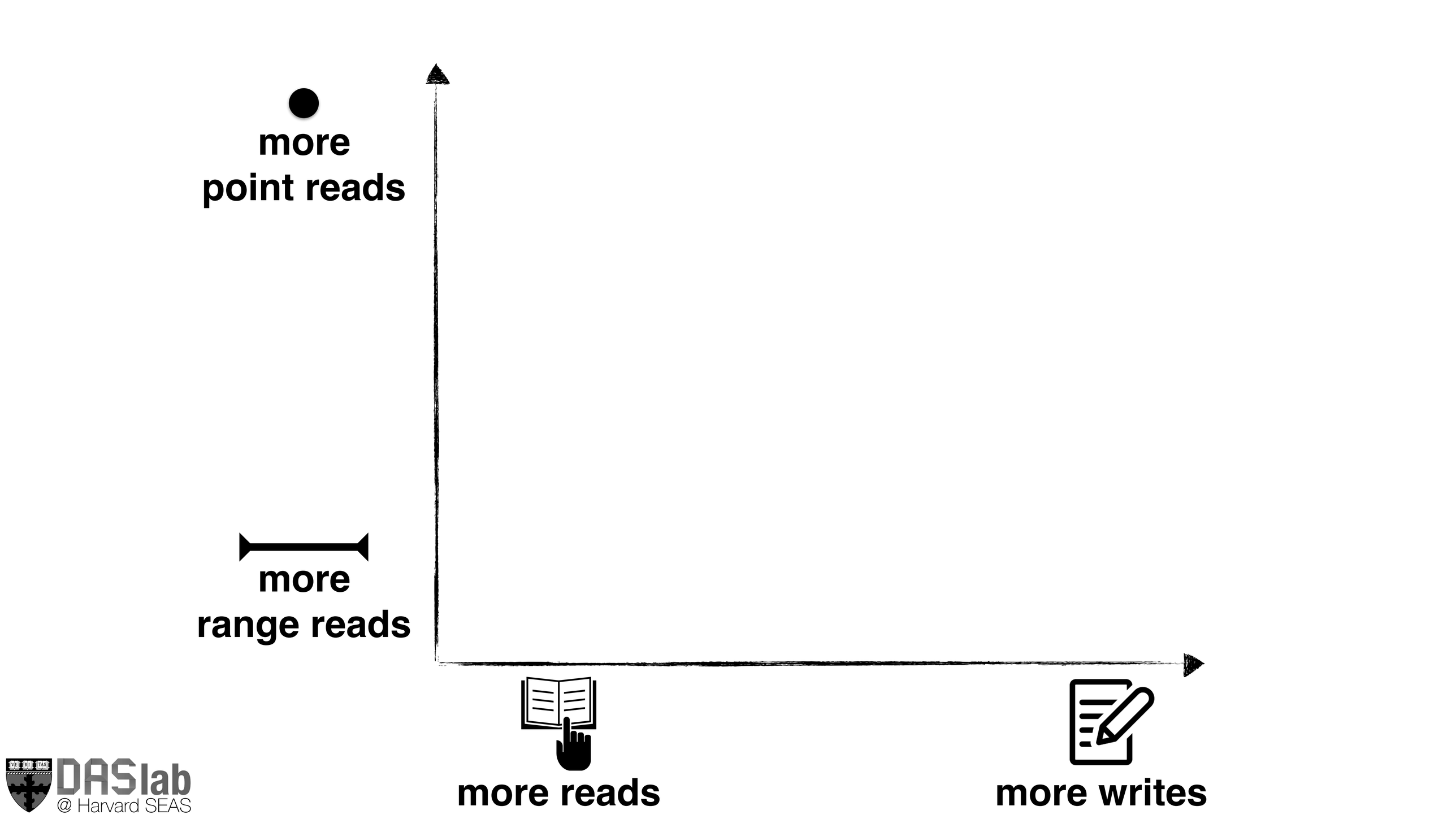


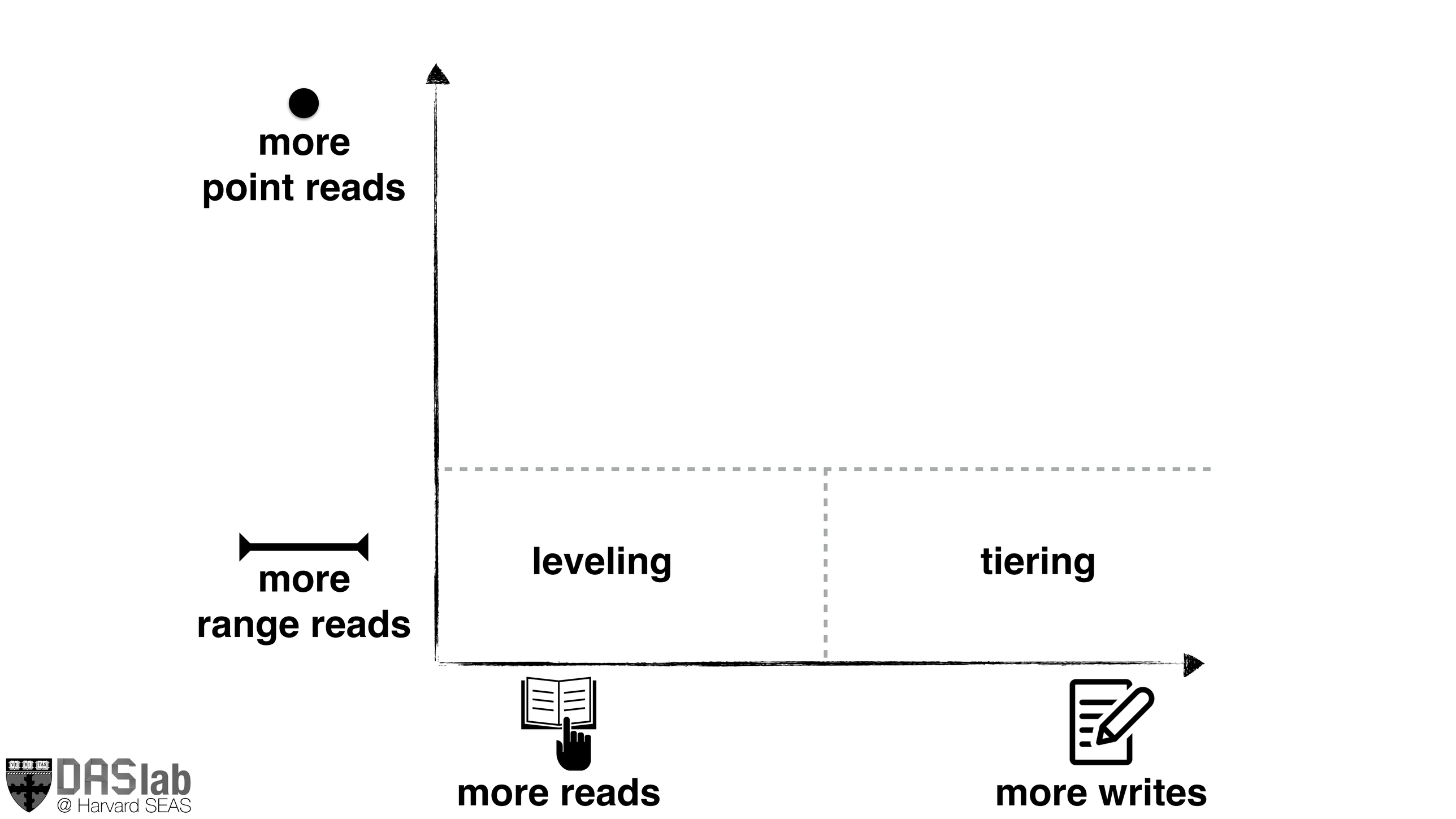


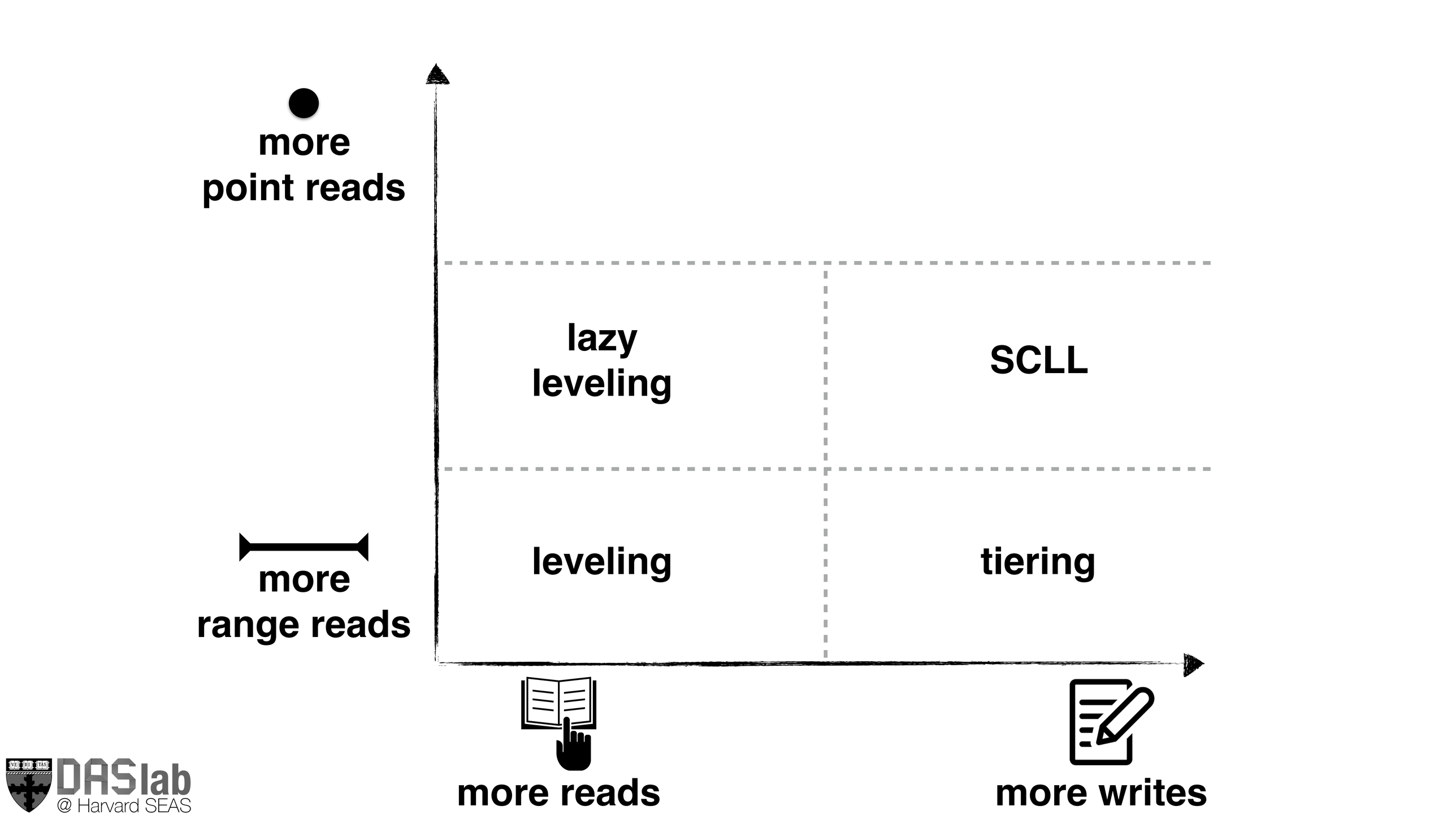


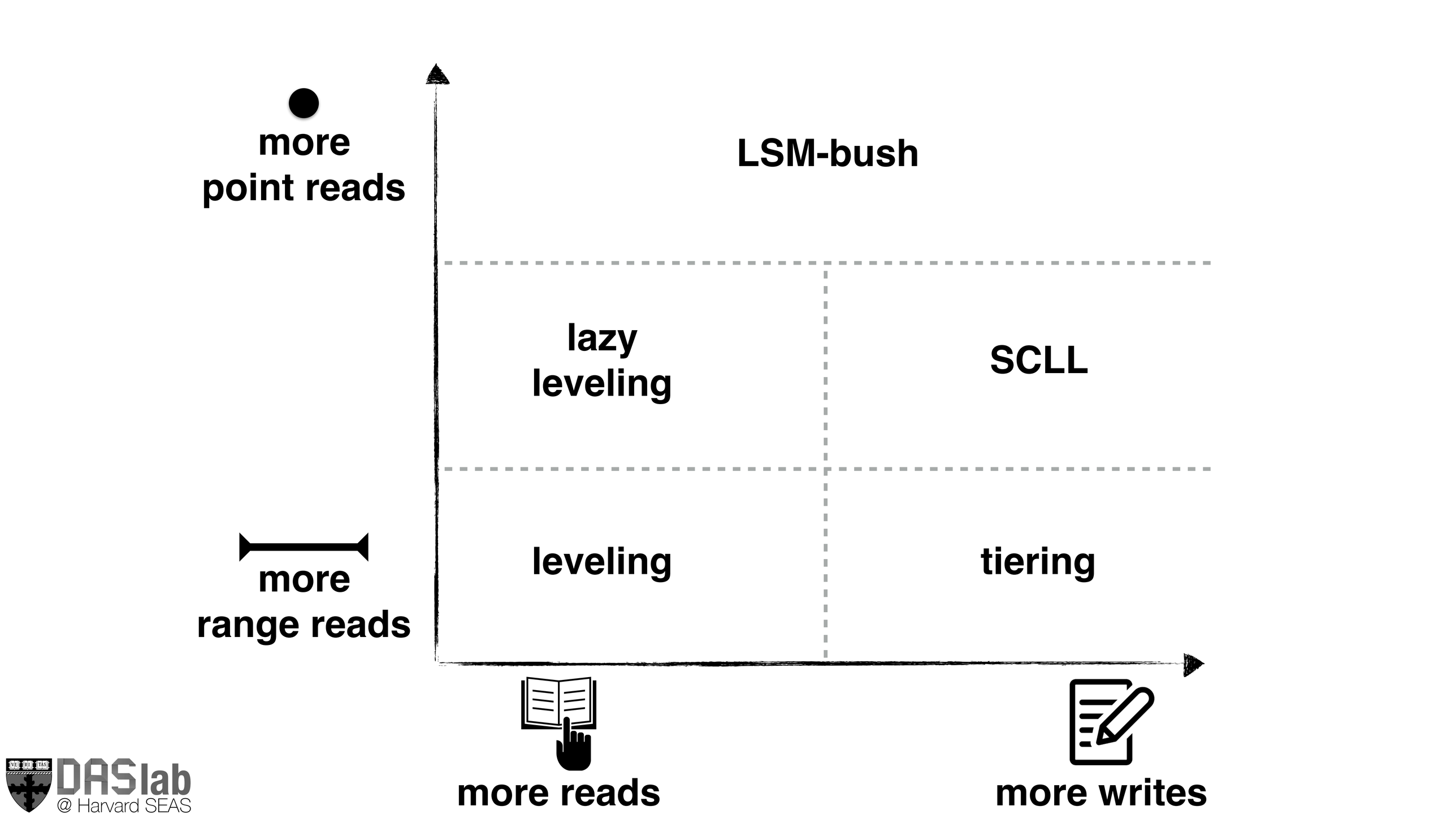

log
Tiering
Leveling
SCLL
Lazy Leveling
LSM-bush
sorted
array


log
LSM-bush
SCLL
Lazy Leveling
Tiering
Leveling
sorted
array


more
point reads
more
range reads
more reads
more writes


more
point reads
more
range reads
leveling
tiering
more reads
more writes


more
point reads
lazy
leveling
SCLL
leveling
tiering
more
range reads
more reads
more writes


more
point reads
LSM-bush
lazy
leveling
SCLL
leveling
tiering
more
range reads
more reads
more writes

Wacky: **A**morphous **C**alculable **K**e**y**-Value Store

LSM-bush

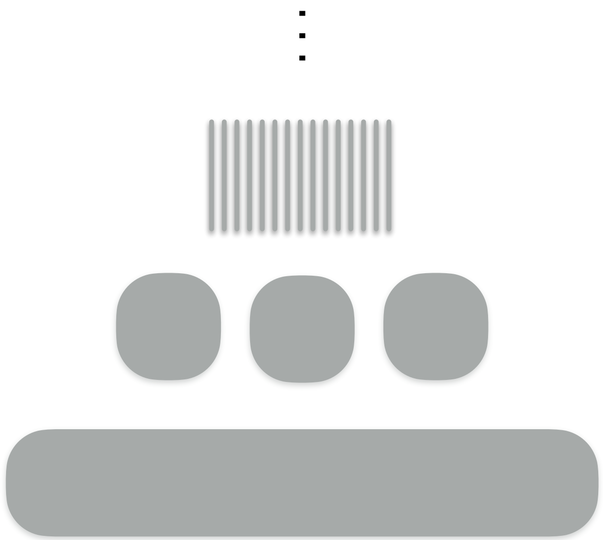
SCLL

lazy leveling

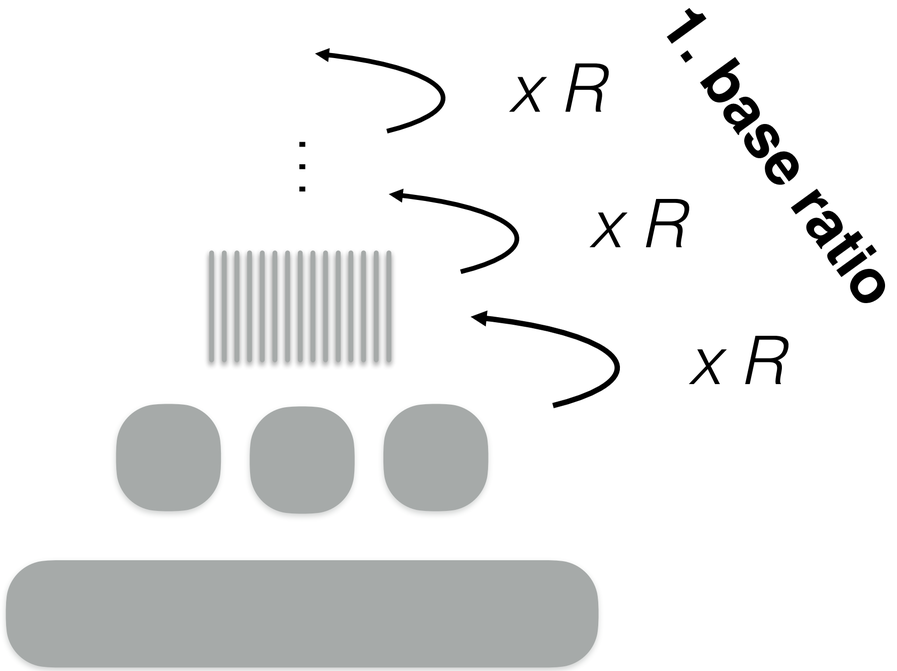
tiering

leveling

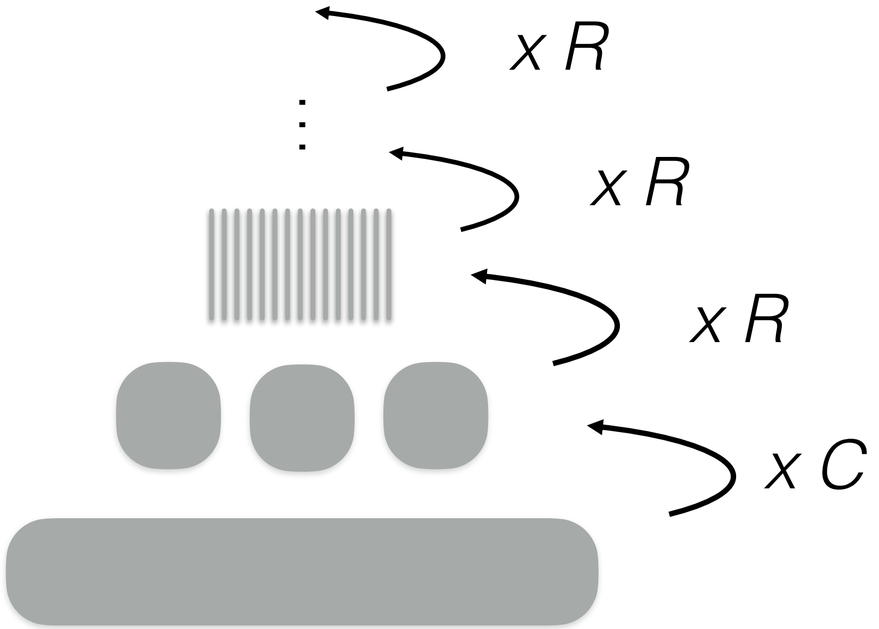
Wacky: **A**morphous **C**alculable **K**e**y**-Value Store



Wacky: **A**morphous **C**alculable **K**e**y**-Value Store



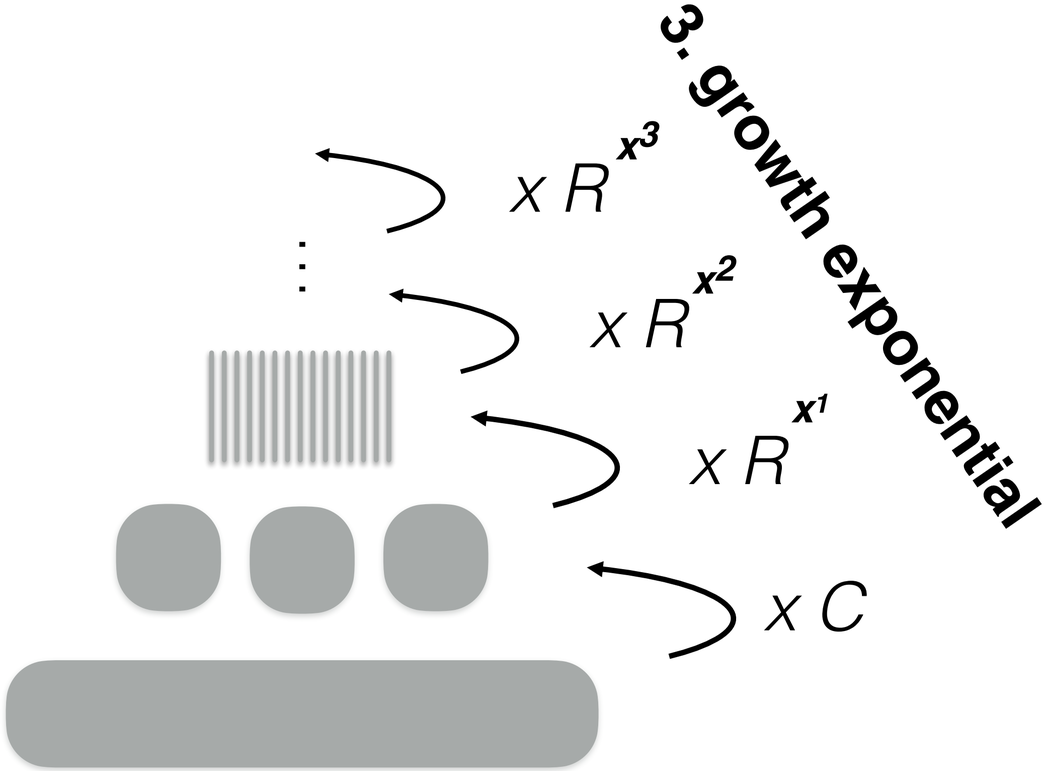
Wacky: **A**morphous **C**alculable **K**e**y**-Value Store



1. base ratio

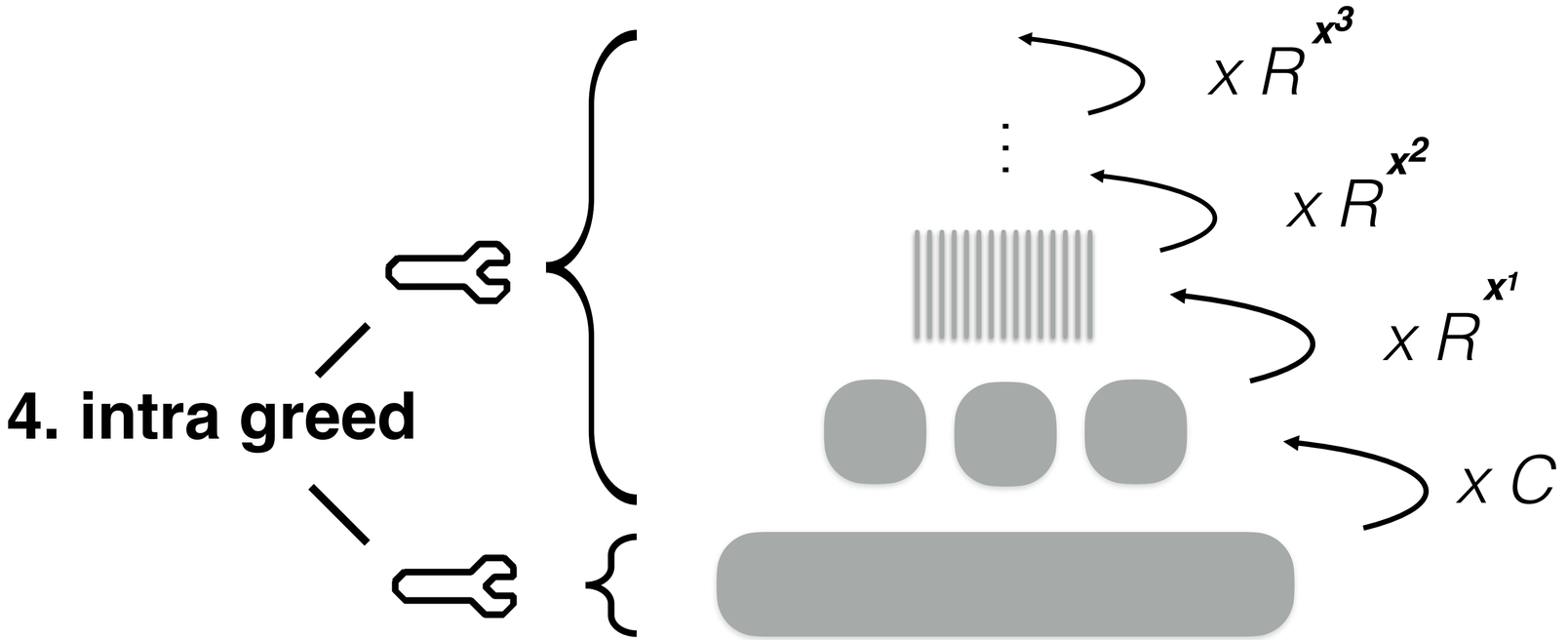
2. capping ratio

Wacky: **A**morphous **C**alculable **K**e**y**-Value Store

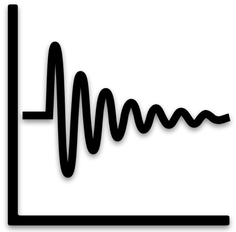


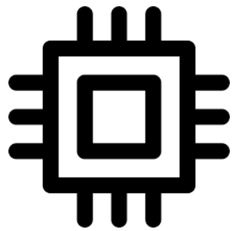
1. base ratio
2. capping ratio

Wacky: **A**morphous **C**alculable **K**e**y**-Value Store

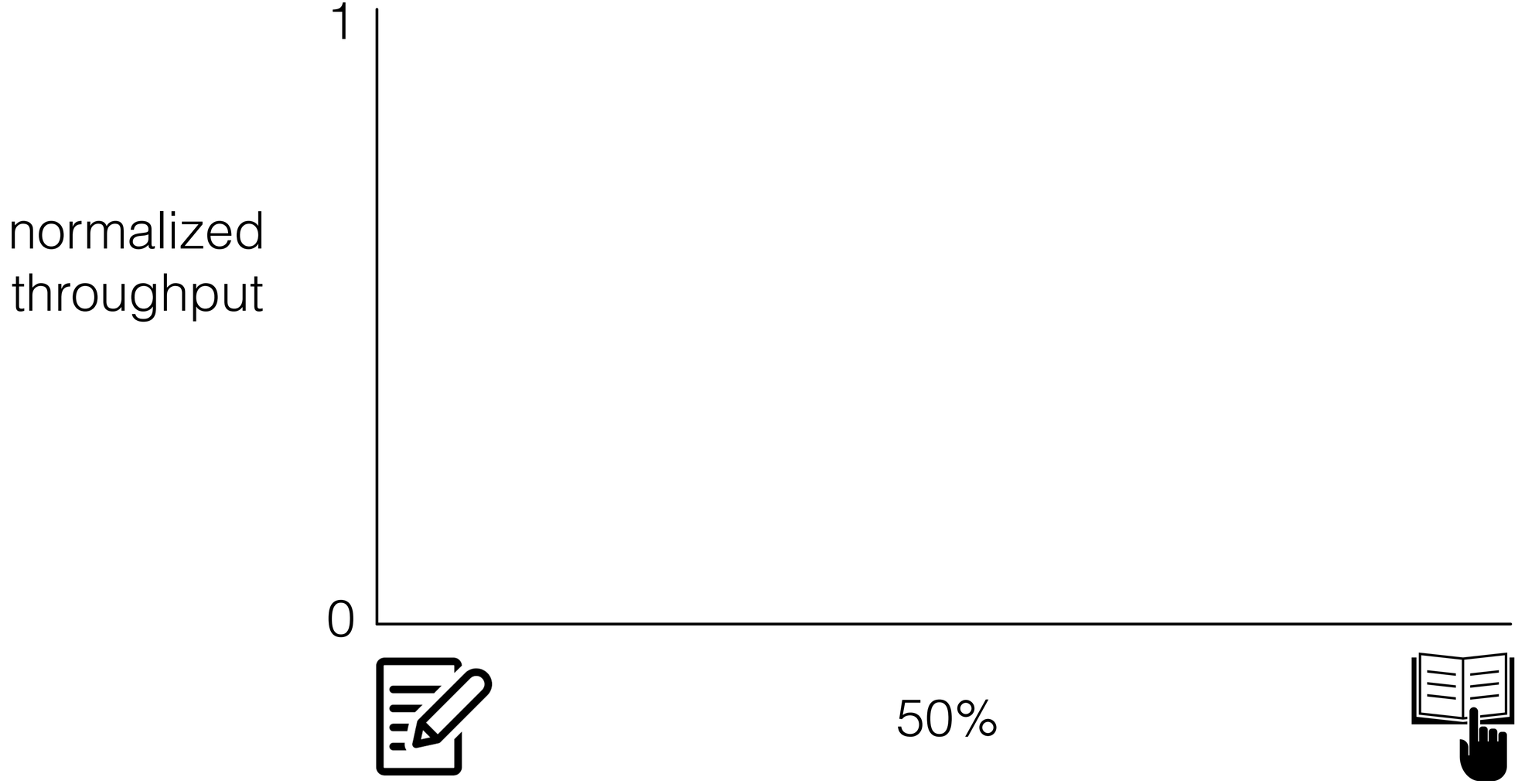


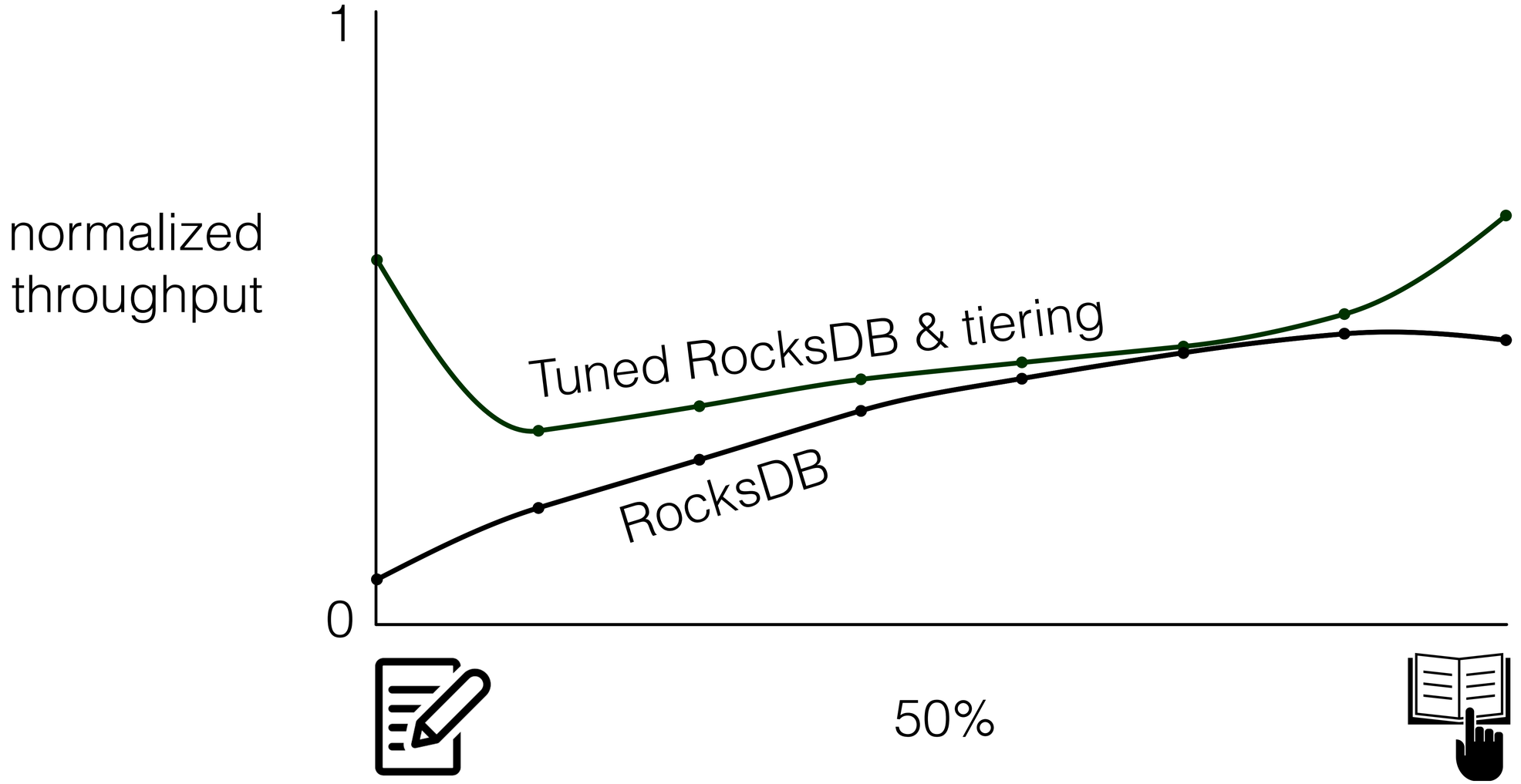
1. base ratio
2. capping ratio
3. growth exponential

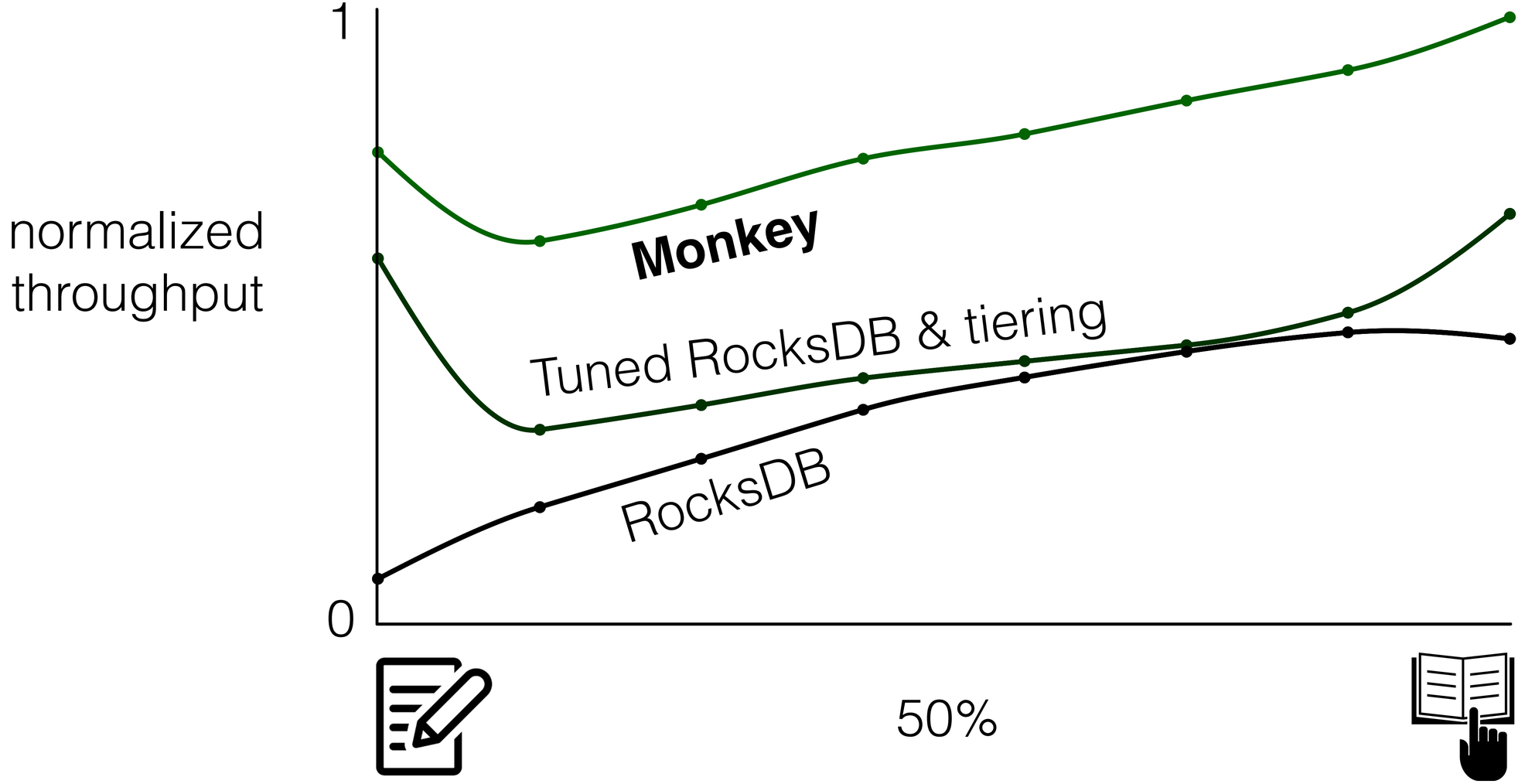


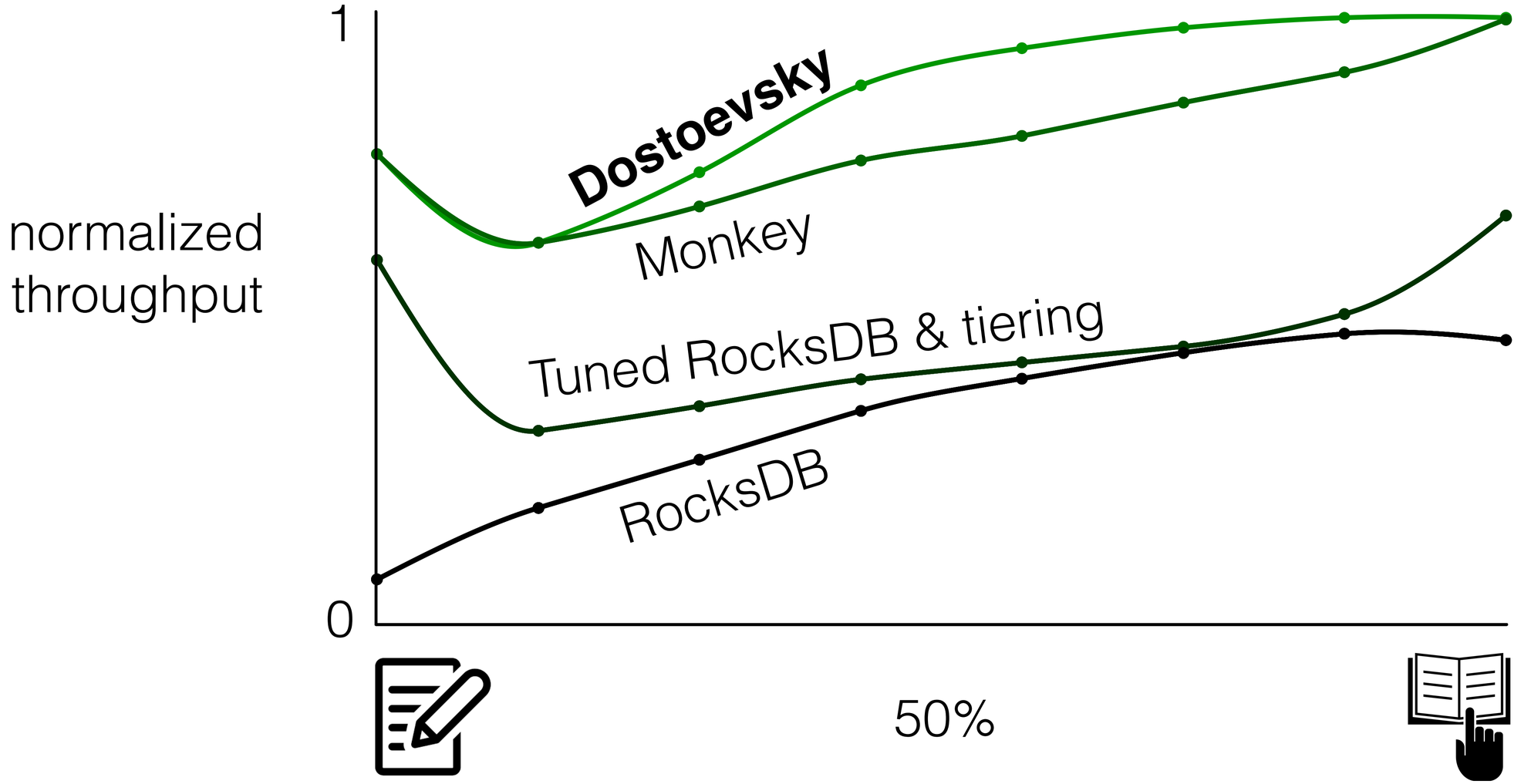


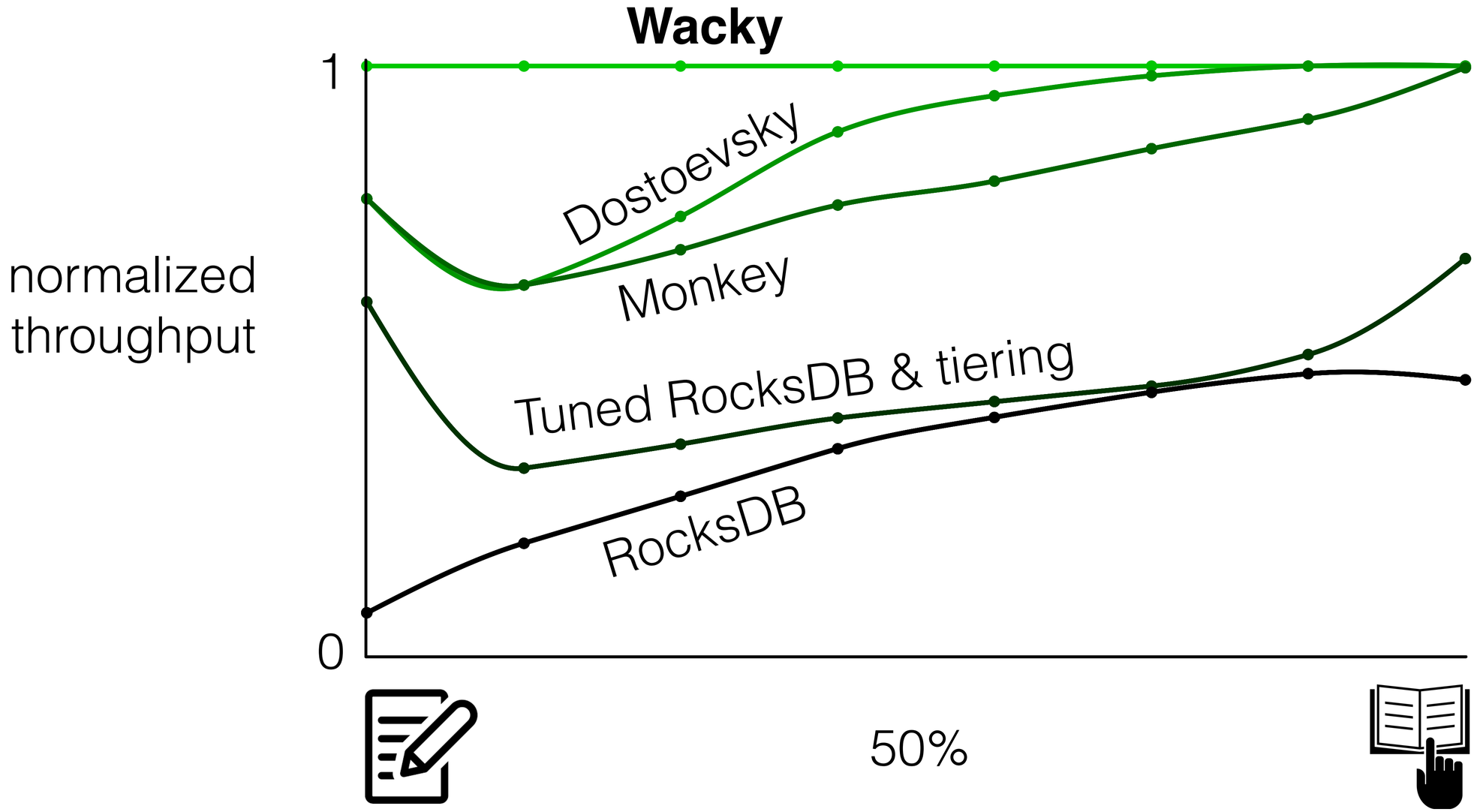
1. base ratio
2. capping ratio
3. growth exponential
4. intra greed

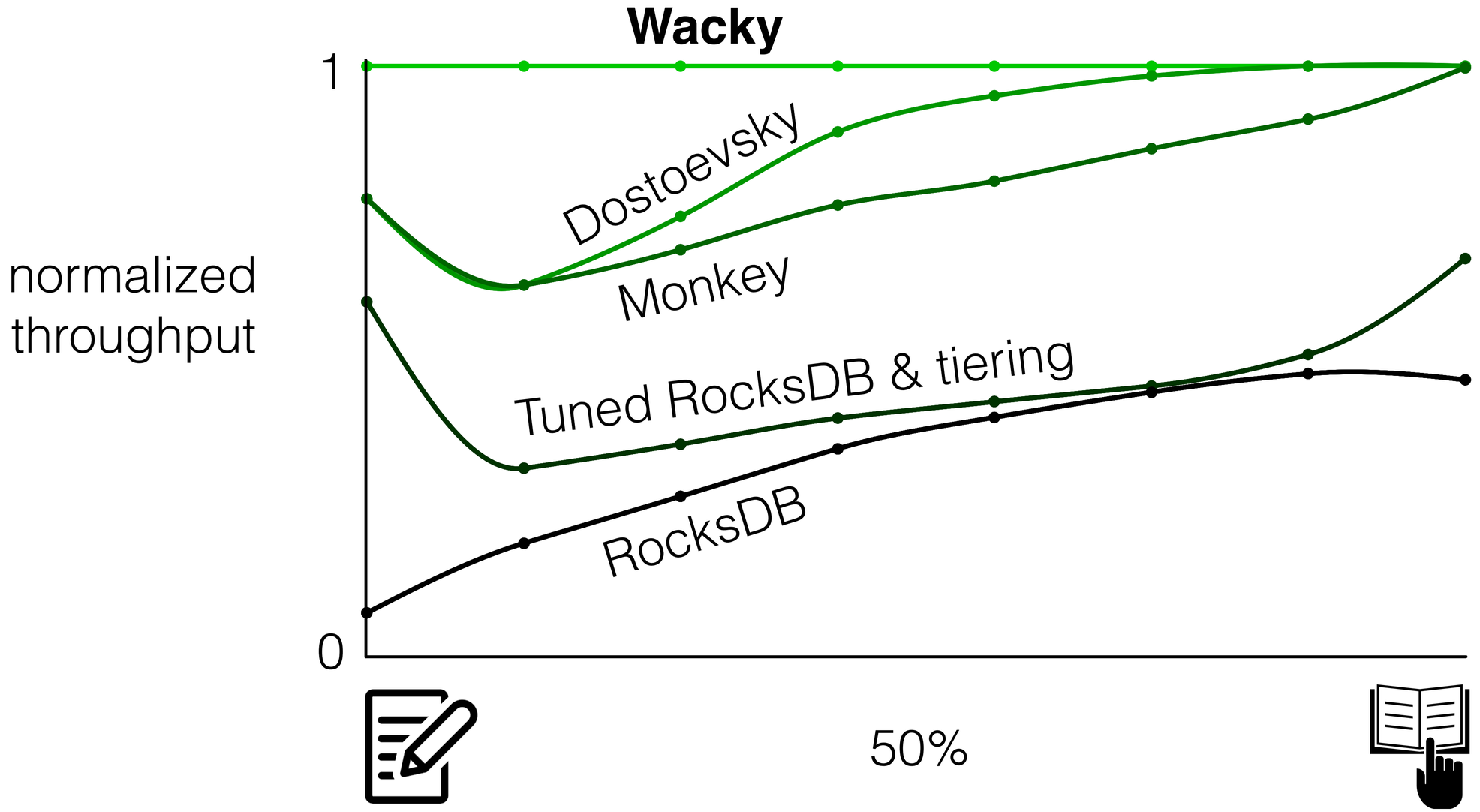

1
normalized
throughput
0
50%


normalized
throughput
1
0
Tuned RocksDB & tiering
RocksDB
50%


normalized
throughput
1
0
Monkey
Tuned RocksDB & tiering
RocksDB
50%


normalized
throughput
1
0
Dostoevsky
Monkey
Tuned RocksDB & tiering
RocksDB
50%


Wacky
1
0
normalized
throughput
Dostoevsky
Monkey
Tuned RocksDB & tiering
RocksDB
50%


Wacky
1
0
normalized
throughput
Dostoevsky
Monkey
Tuned RocksDB & tiering
RocksDB
50%

thanks!

Monkey

Dostoevsky

Wacky

DASlab
@ Harvard SEAS