Cosine: A Cloud-Cost Optimized Self-Designing Key-Value Storage Engine

Subarna Chatterjee, Meena Jagadeesan,
Wilson Qin, Stratos Idreos

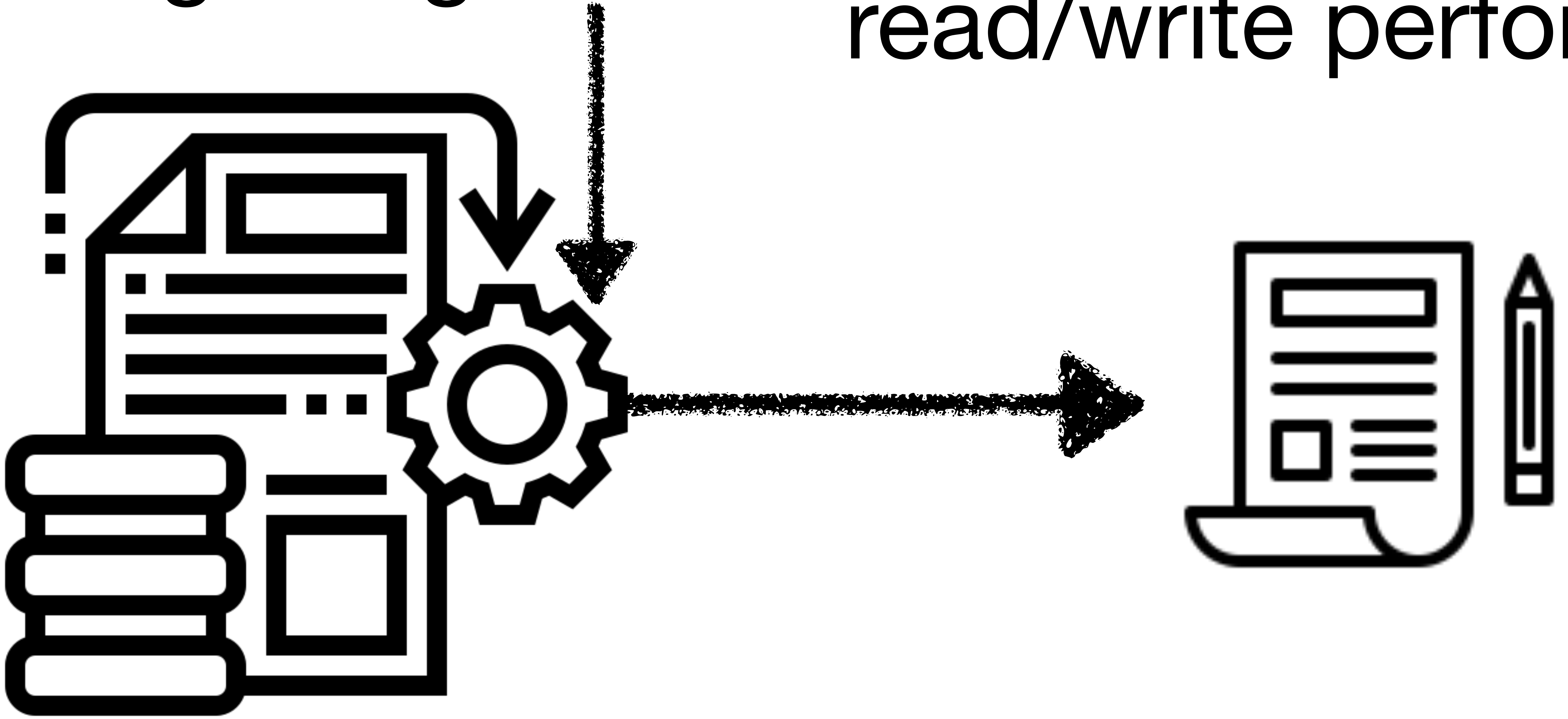
Data Systems Laboratory (DASLab)
Harvard University

DASlab @ Harvard SEAS

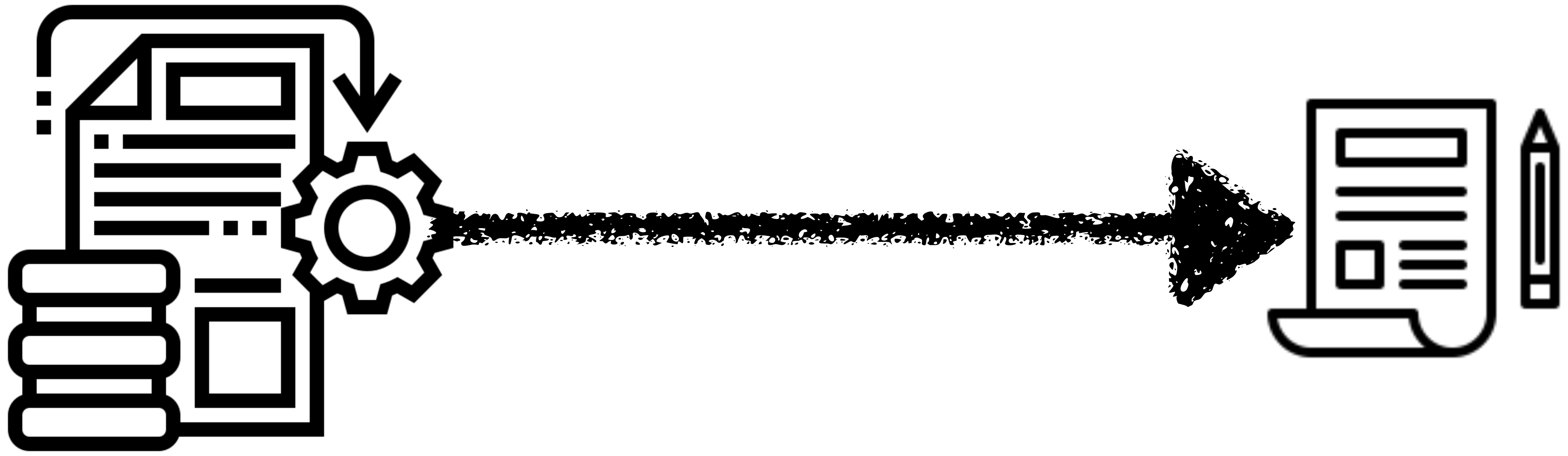
DATASTAX

storage-engines

read/write performance



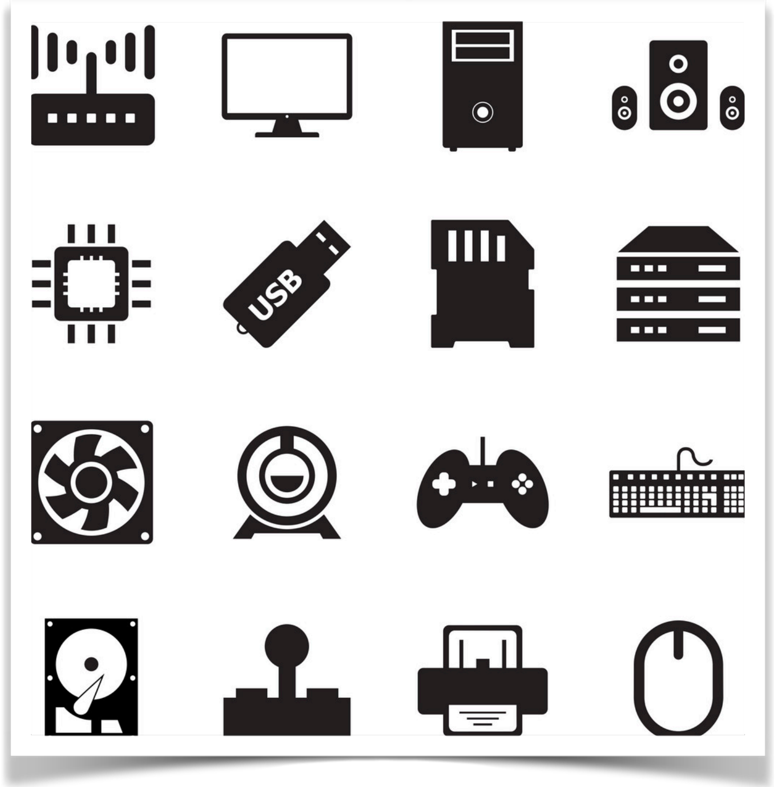
Data-systems



CONTEXT

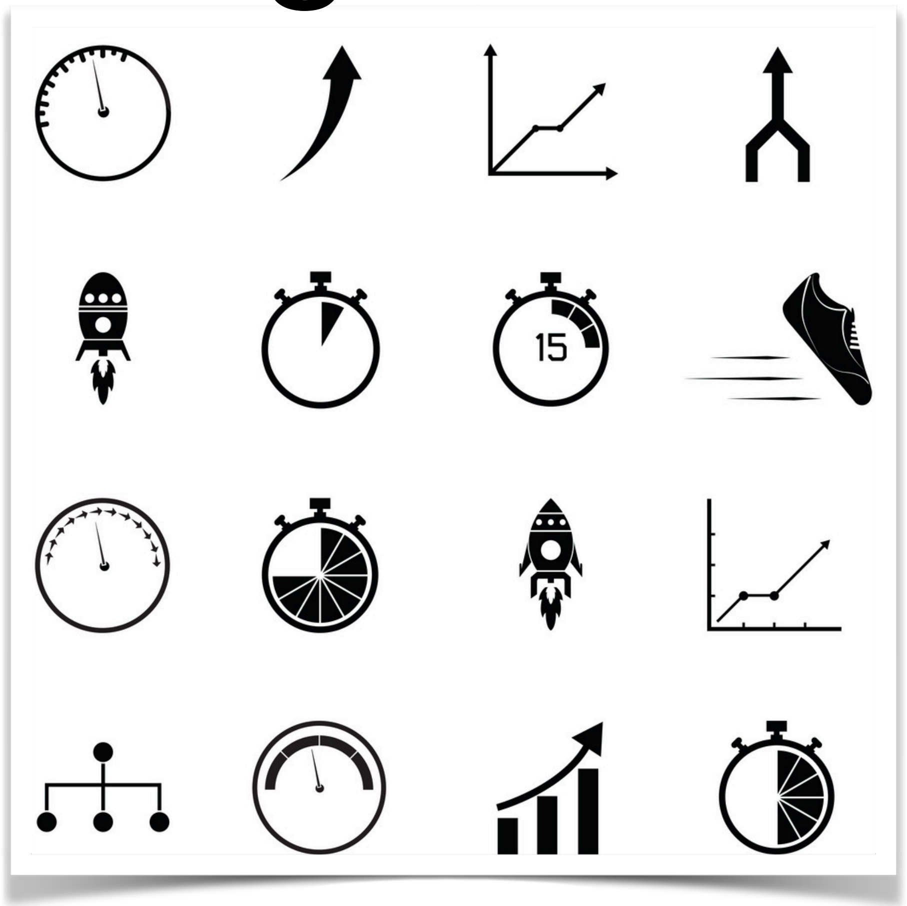
data

hardware



applications

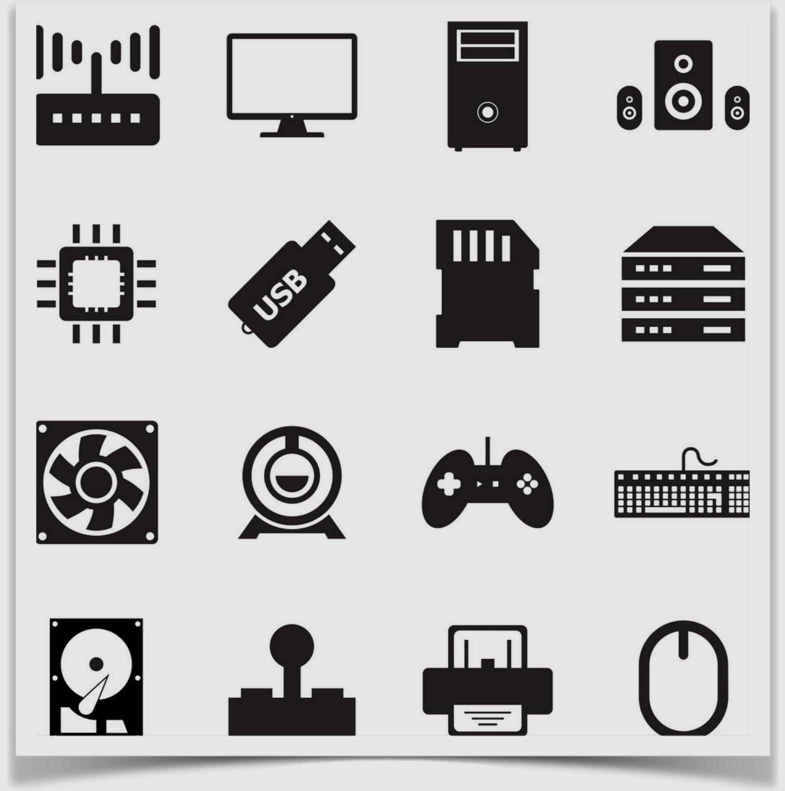
performance goals



The CONTEXT keeps changing …

data

hardware



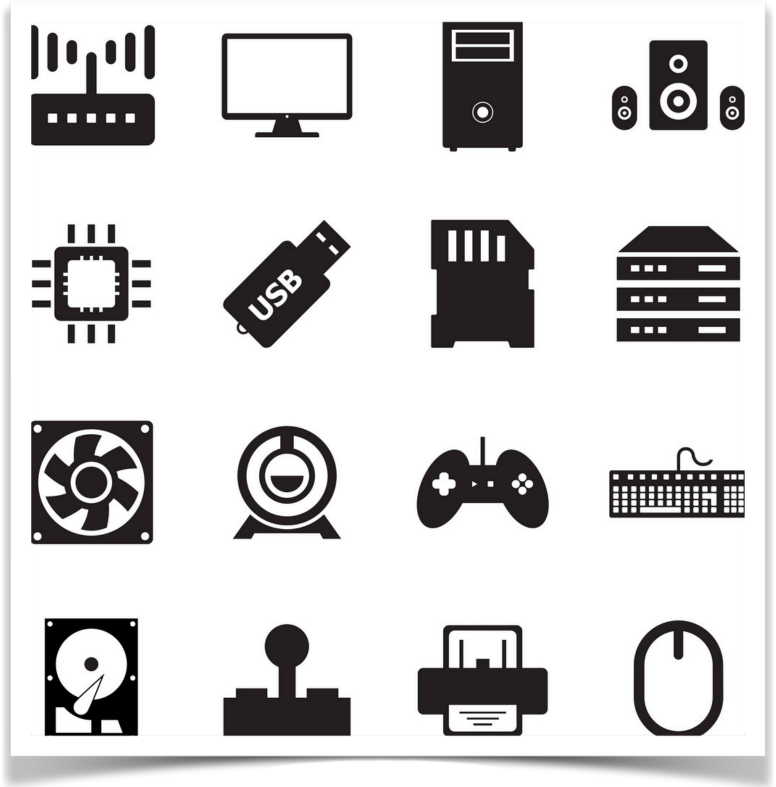
applications

performance goals

The **CONTEXT** keeps changing …

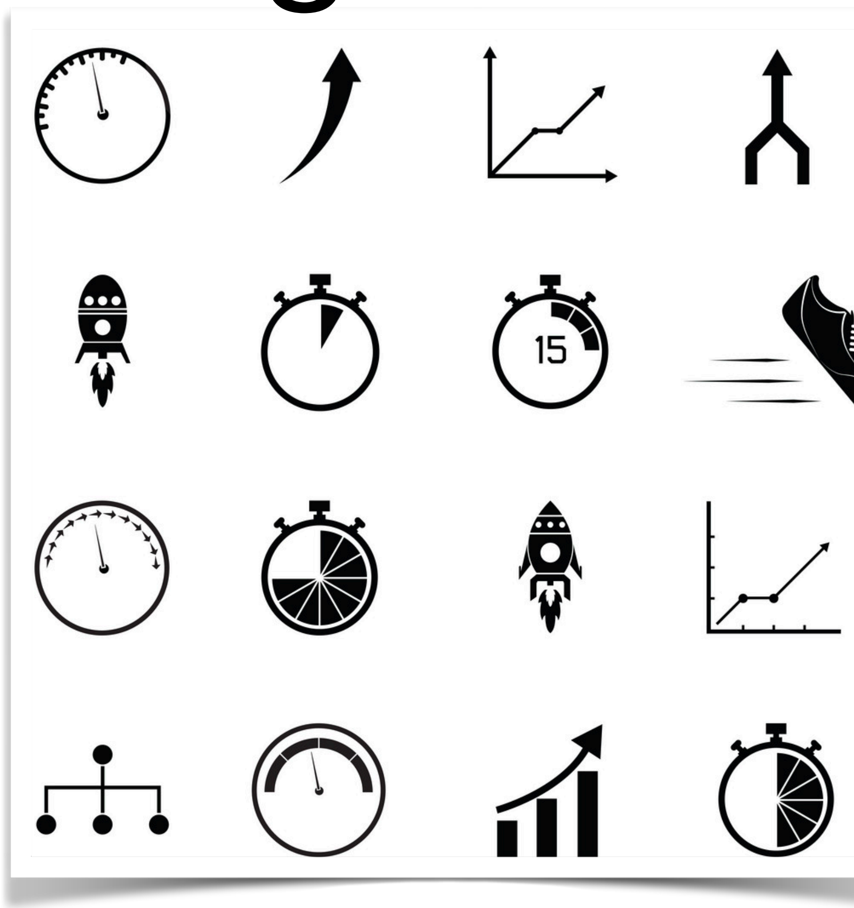
data

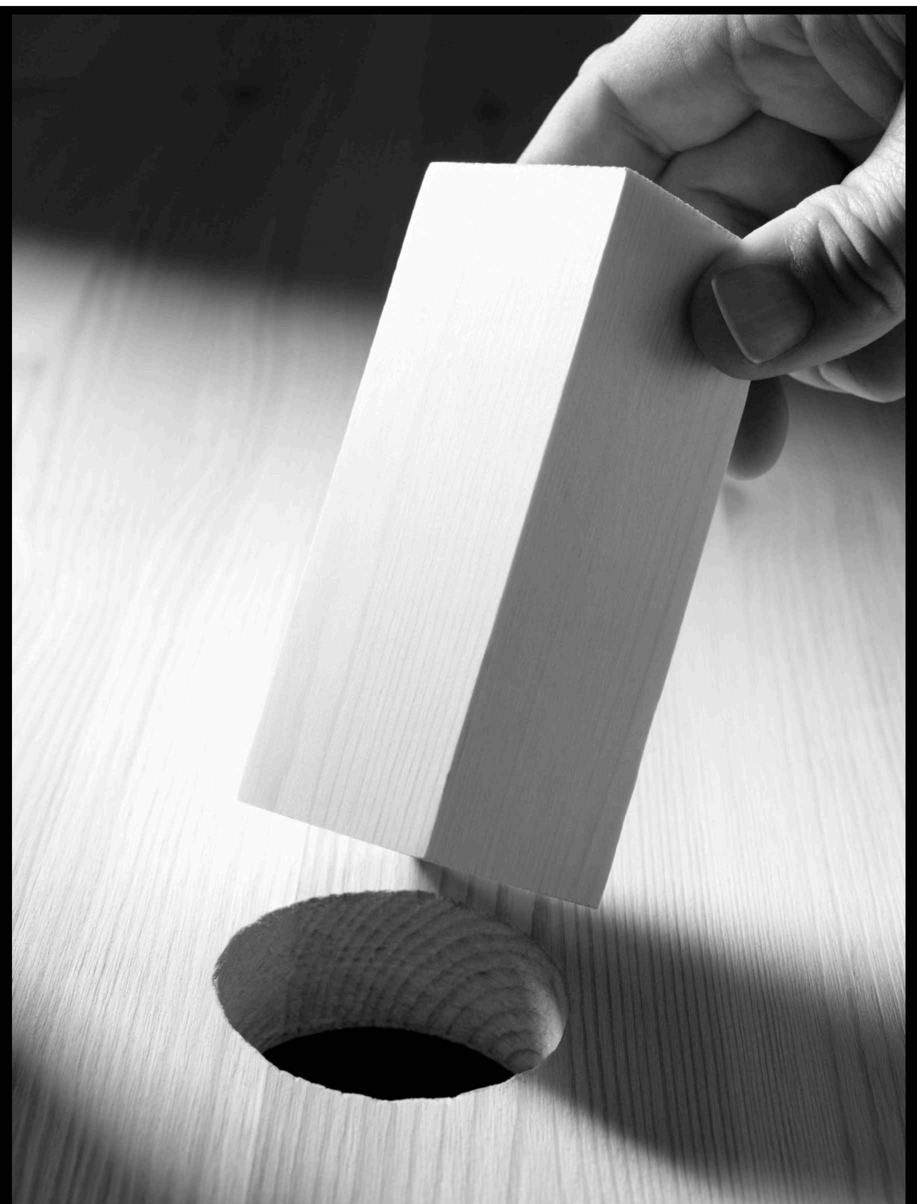
hardware



applications

performance goals

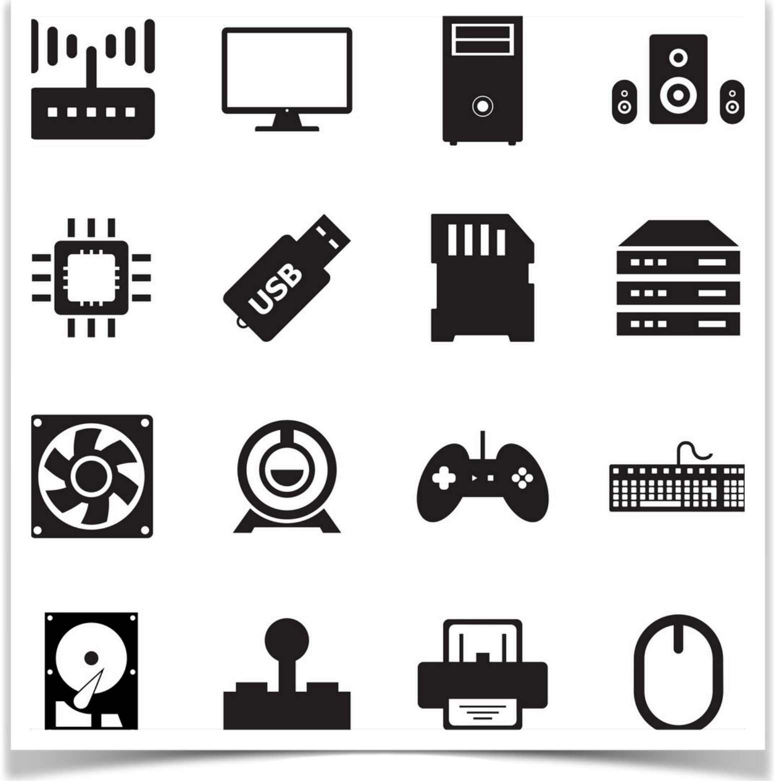




The **CONTEXT** keeps changing …

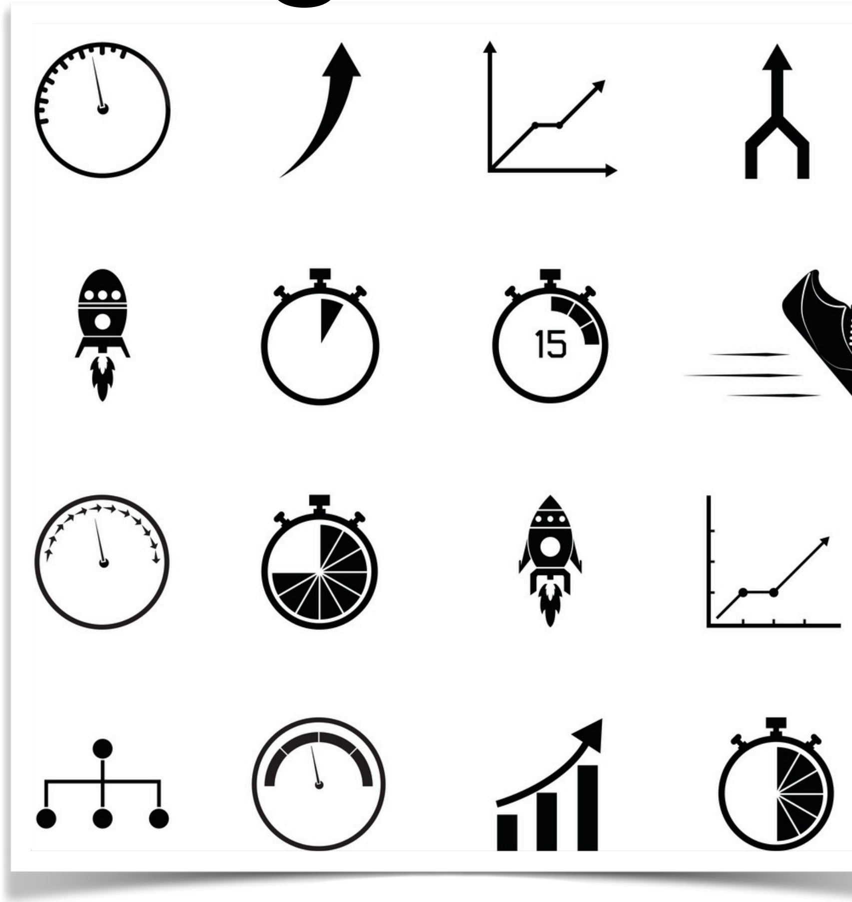
data

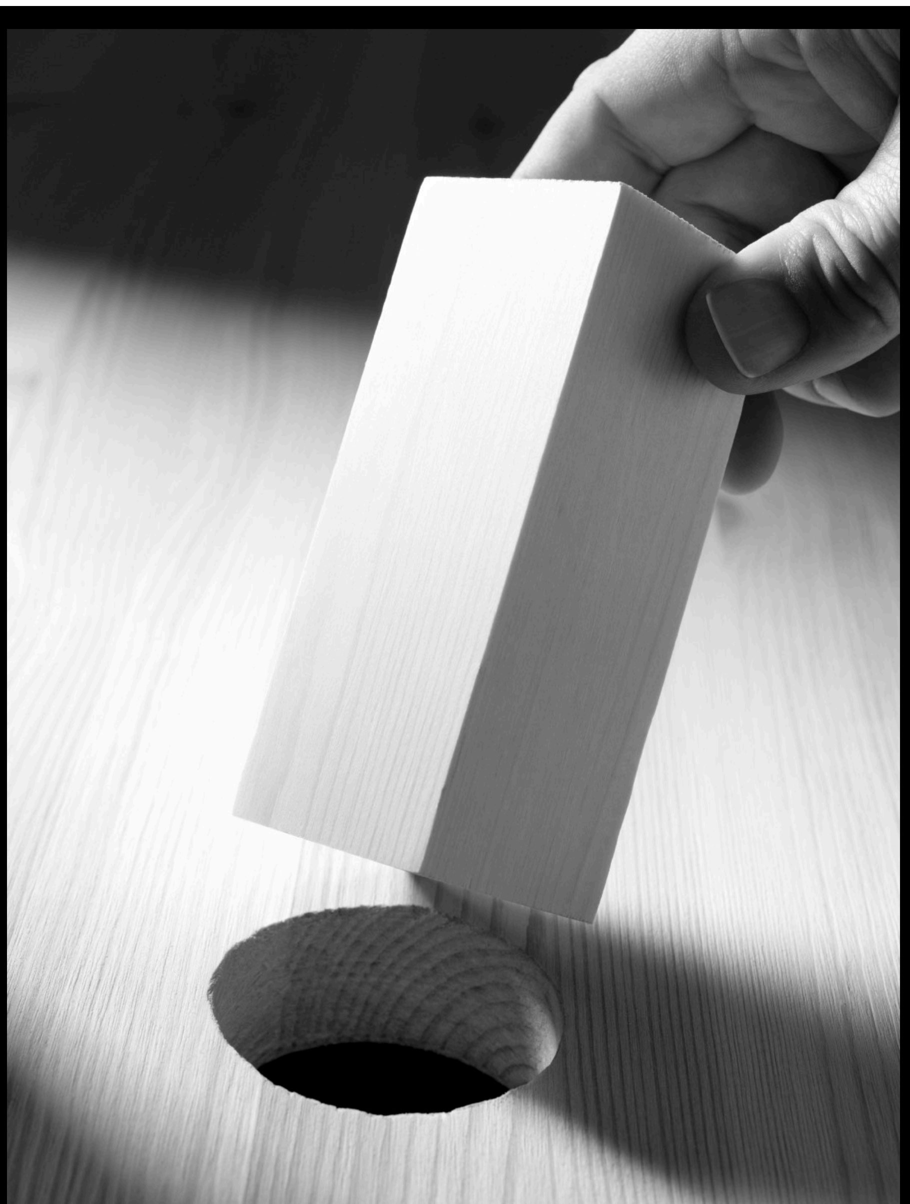
hardware



applications

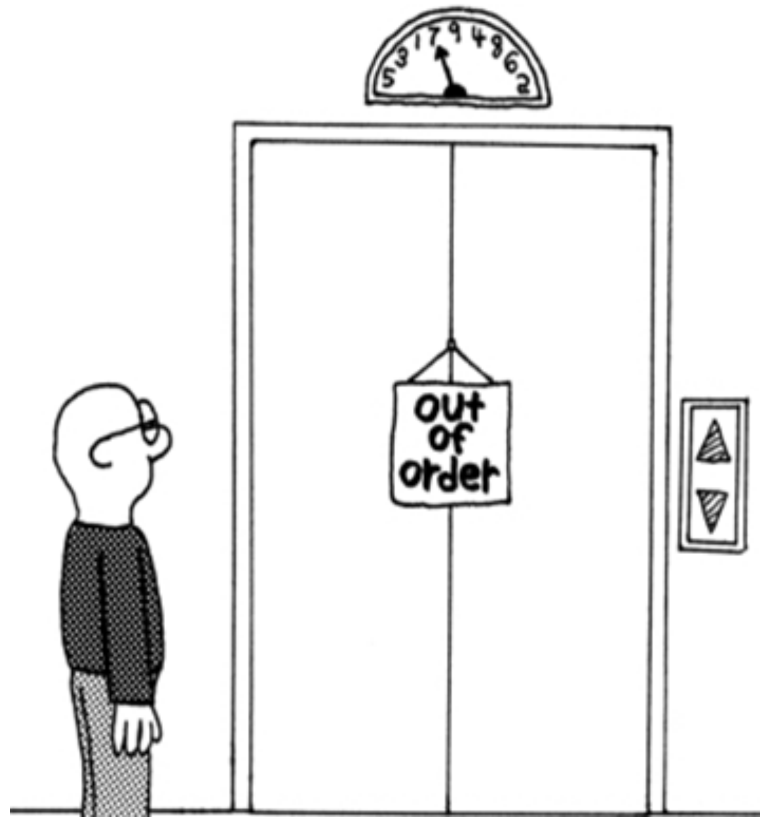
performance goals



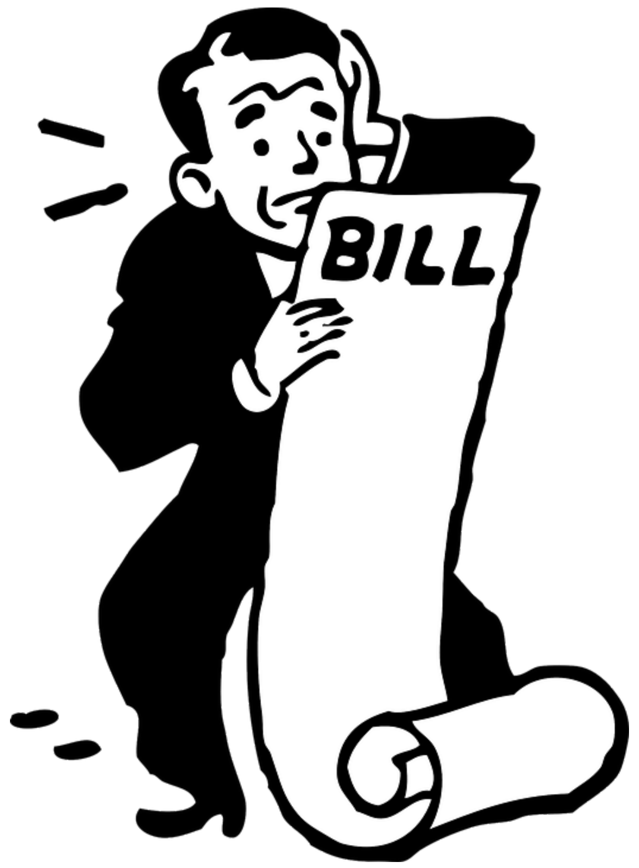


Bottleneck: Sub-Optimal Data Systems

Bottleneck: Sub-Optimal Data Systems



PERFORMANCE



COST

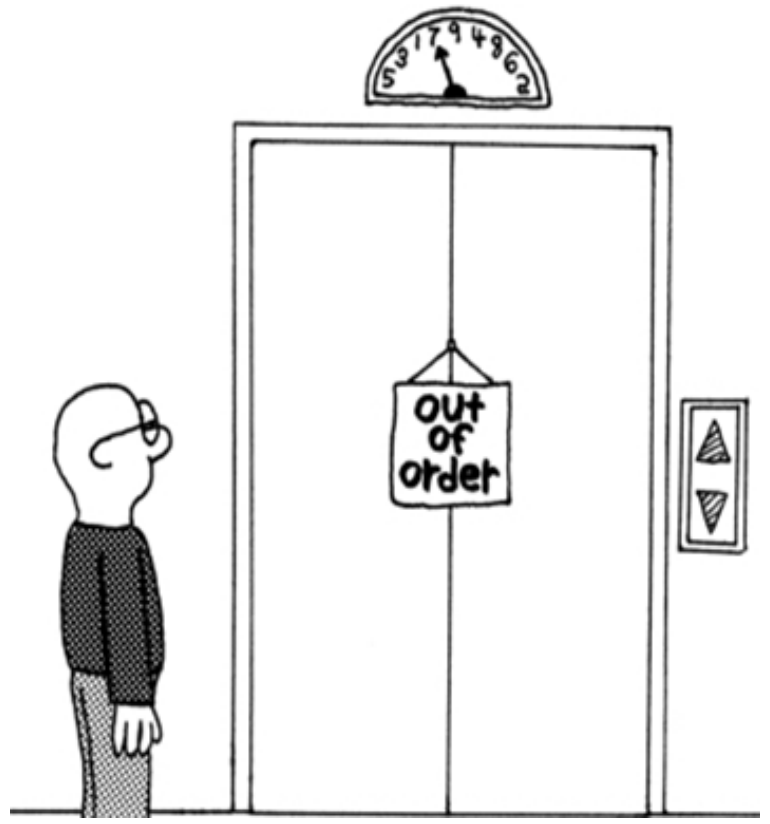
Ryan Booth @that1guy_15 · Mar 4
Oh fun! Another **high cloud bill**. This is exactly how I wanted to spend the rest of my week...

Matt Getty @aspen · Feb 16
The **Cloud** is as **high** in the sky as the **bill** from AWS*

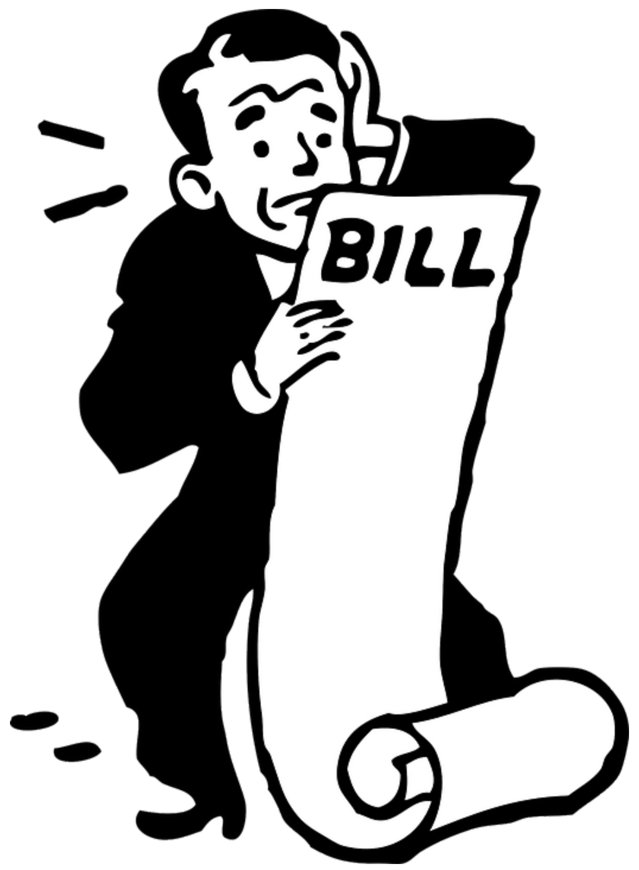
* for a tremendous amount of workloads

CiscoEvents @CiscoEvents · Jan 28
Is your **cloud bill** too **high**? Learn how you can control **cloud** costs with CloudCenter Suite.

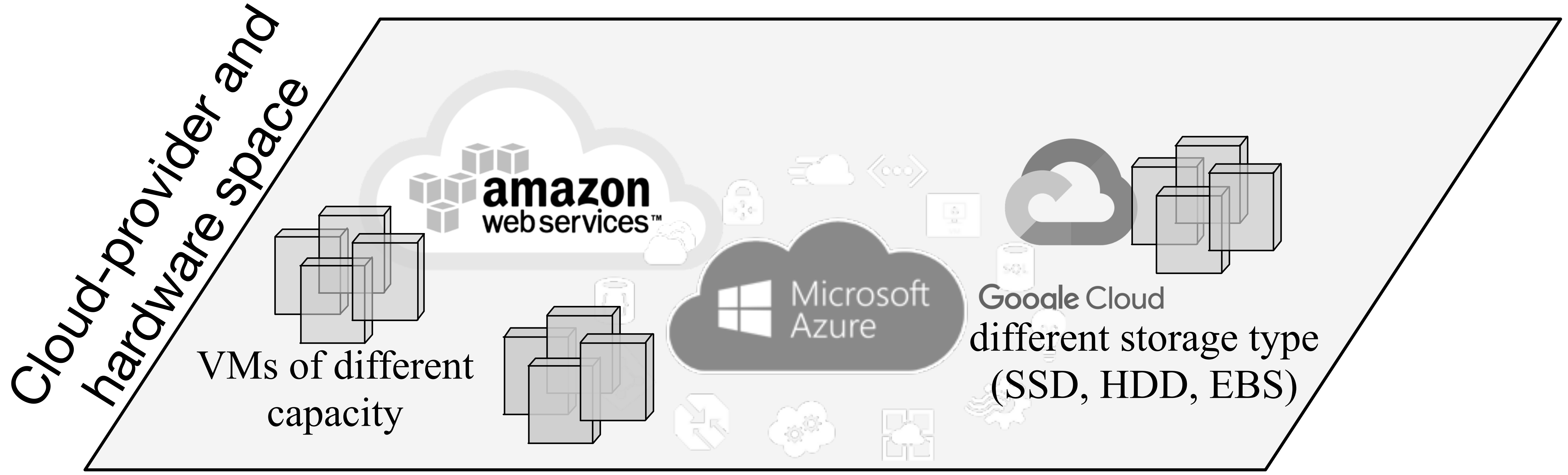
Translucent Computing @translucentcomp · Jan 20
#Kubernetes **cloud** costs are getting out of hand. Working with many clients and different **cloud** service providers, in more than 70% of cases, we see the VM cluster nodes are underutilized, which leads to a **high cloud bill**.



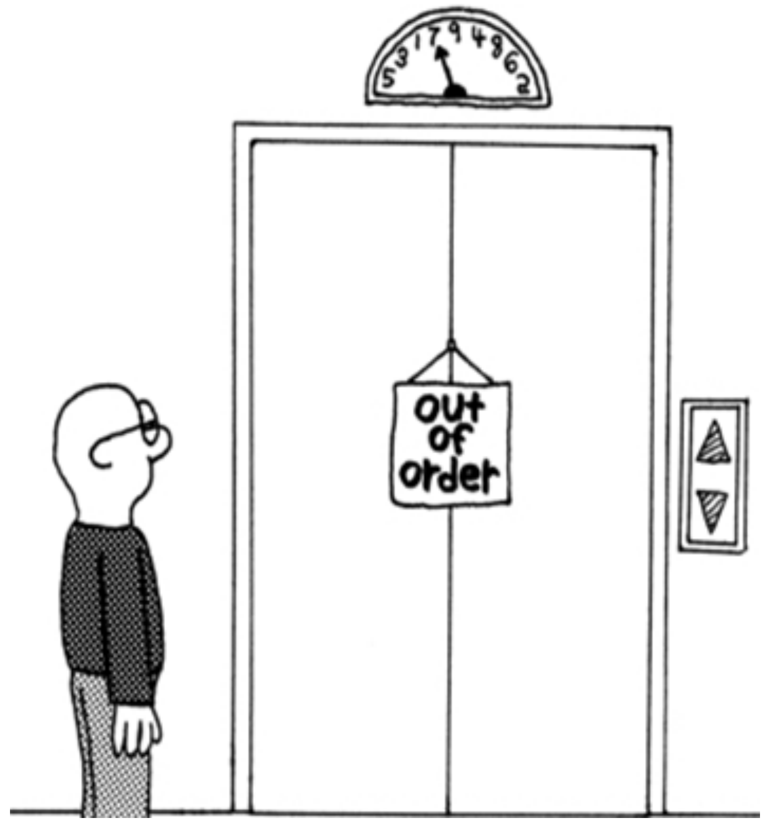
PERFORMANCE



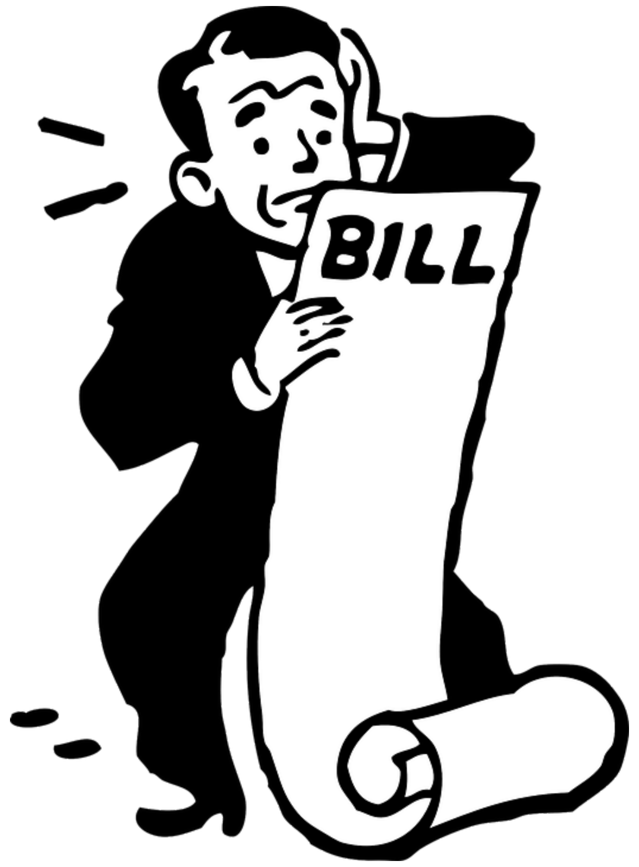
COST



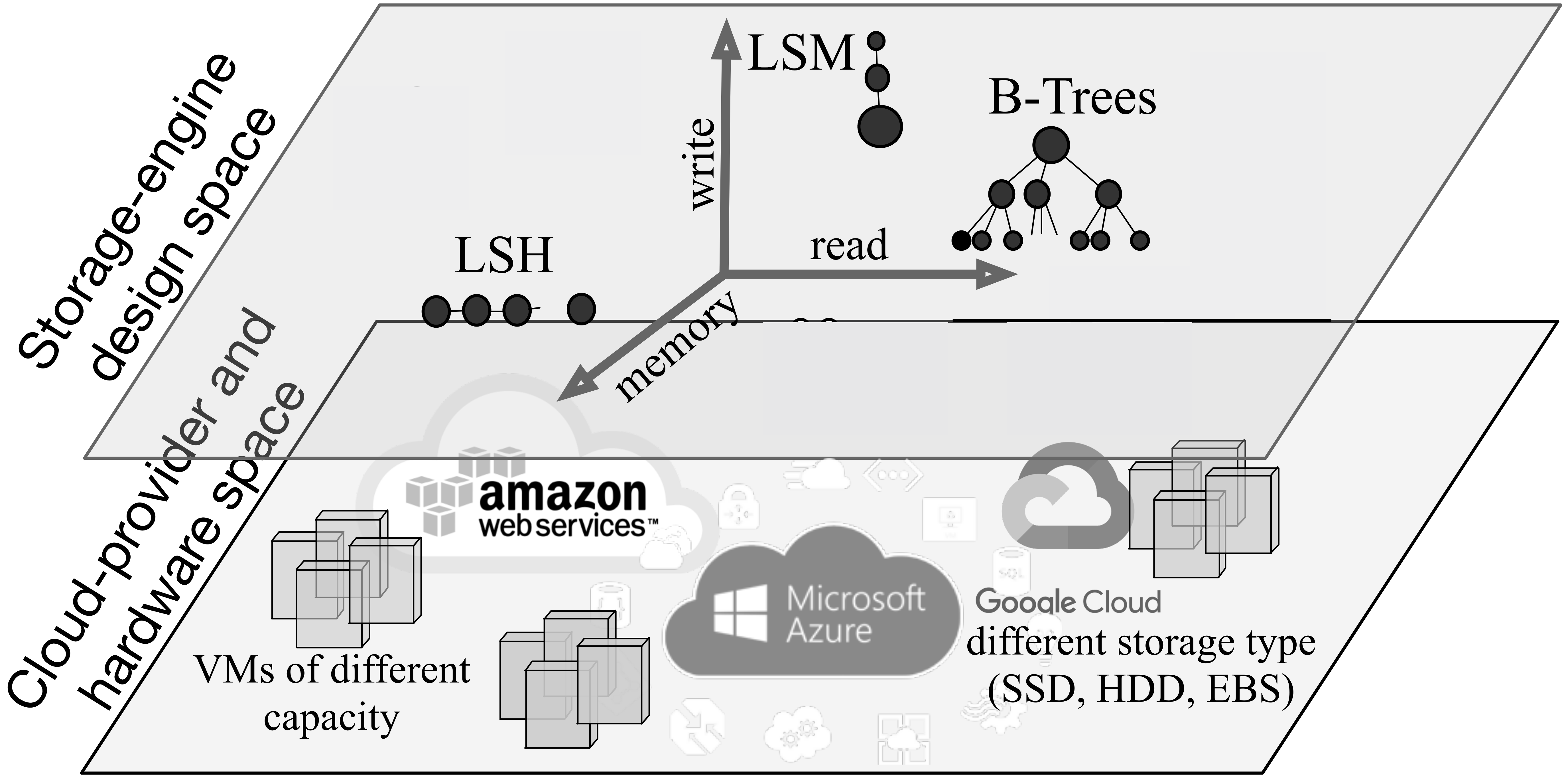
DASlab @ Harvard SEAS



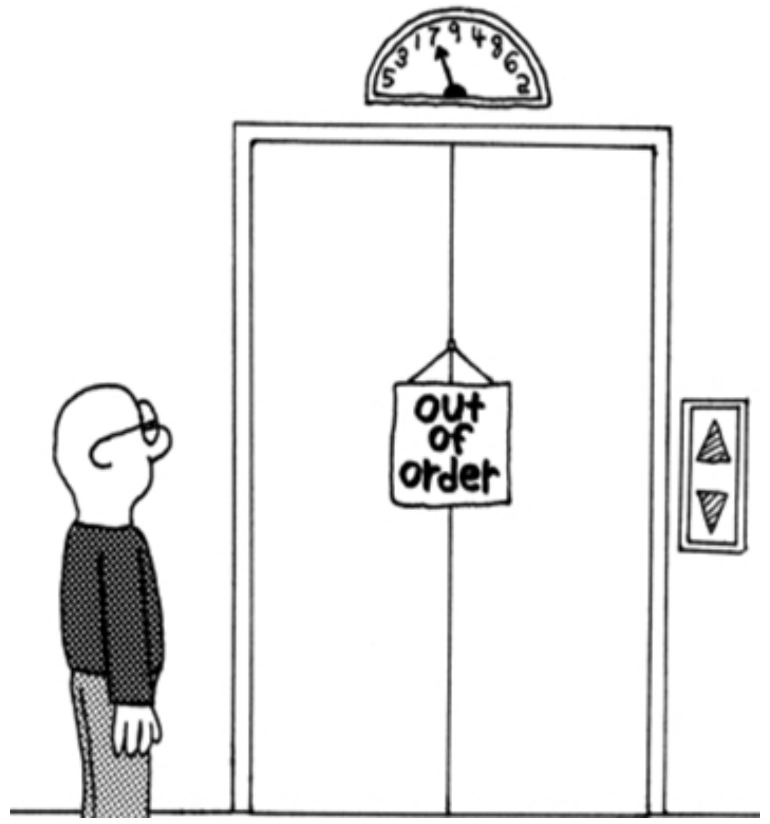
PERFORMANCE



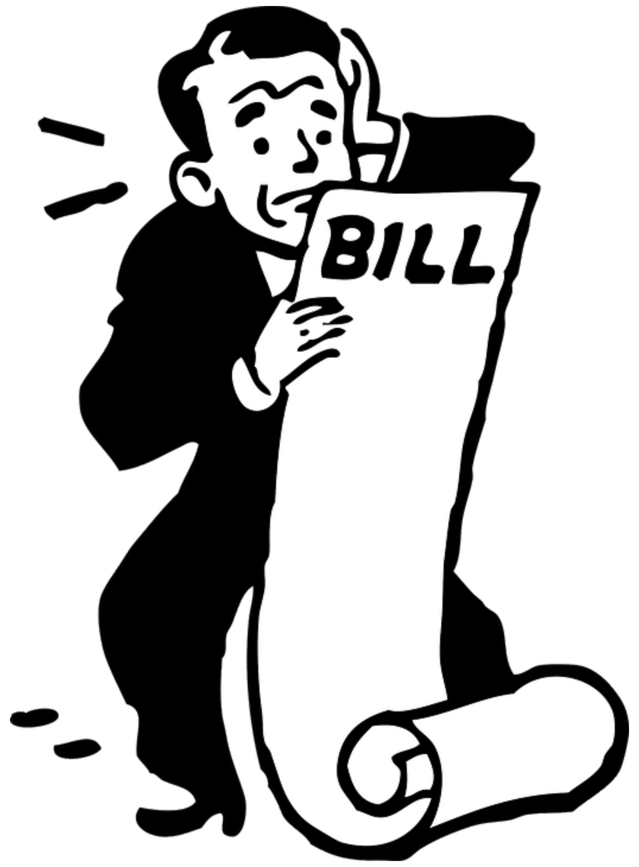
COST



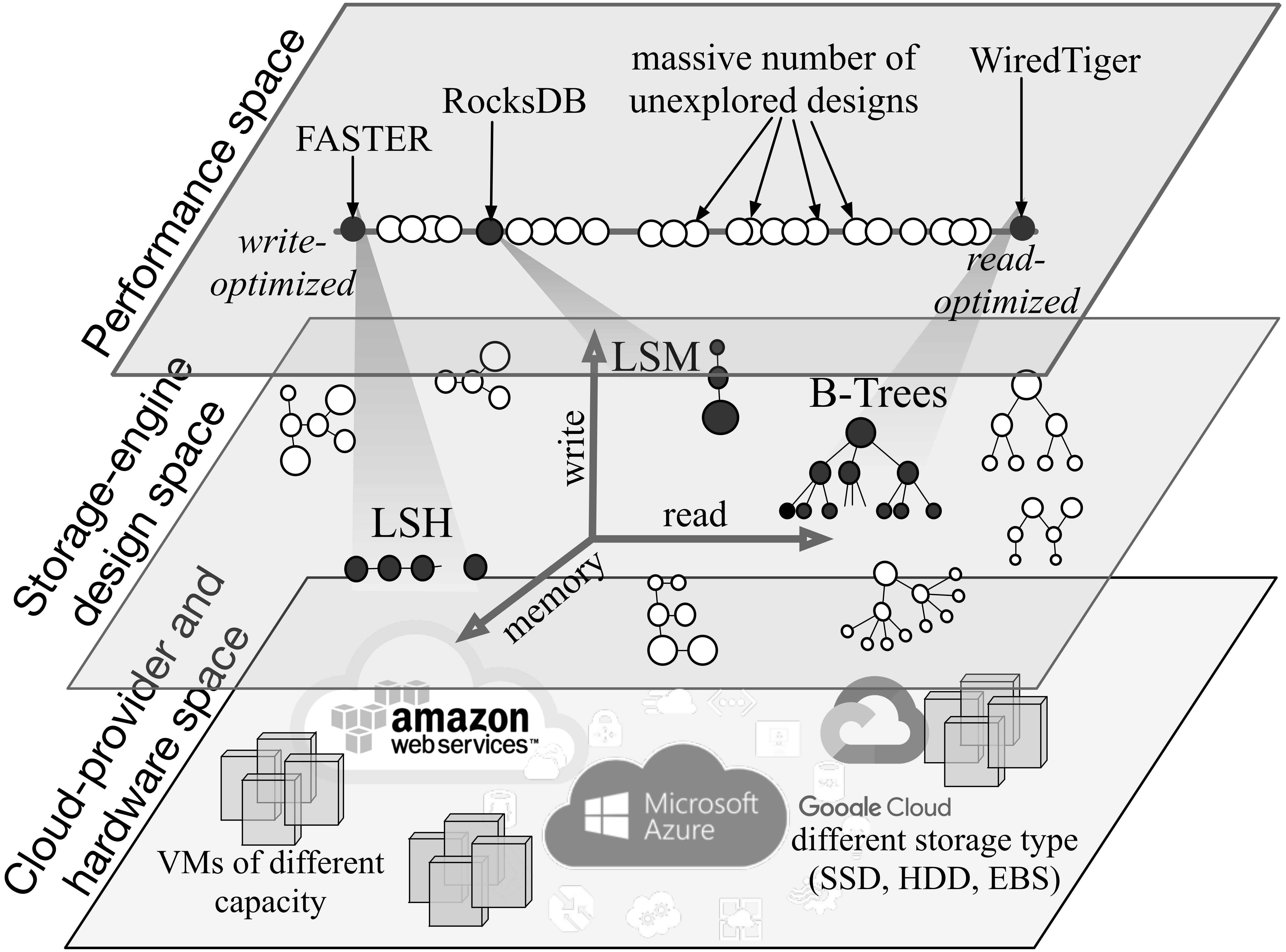
DASlab @ Harvard SEAS



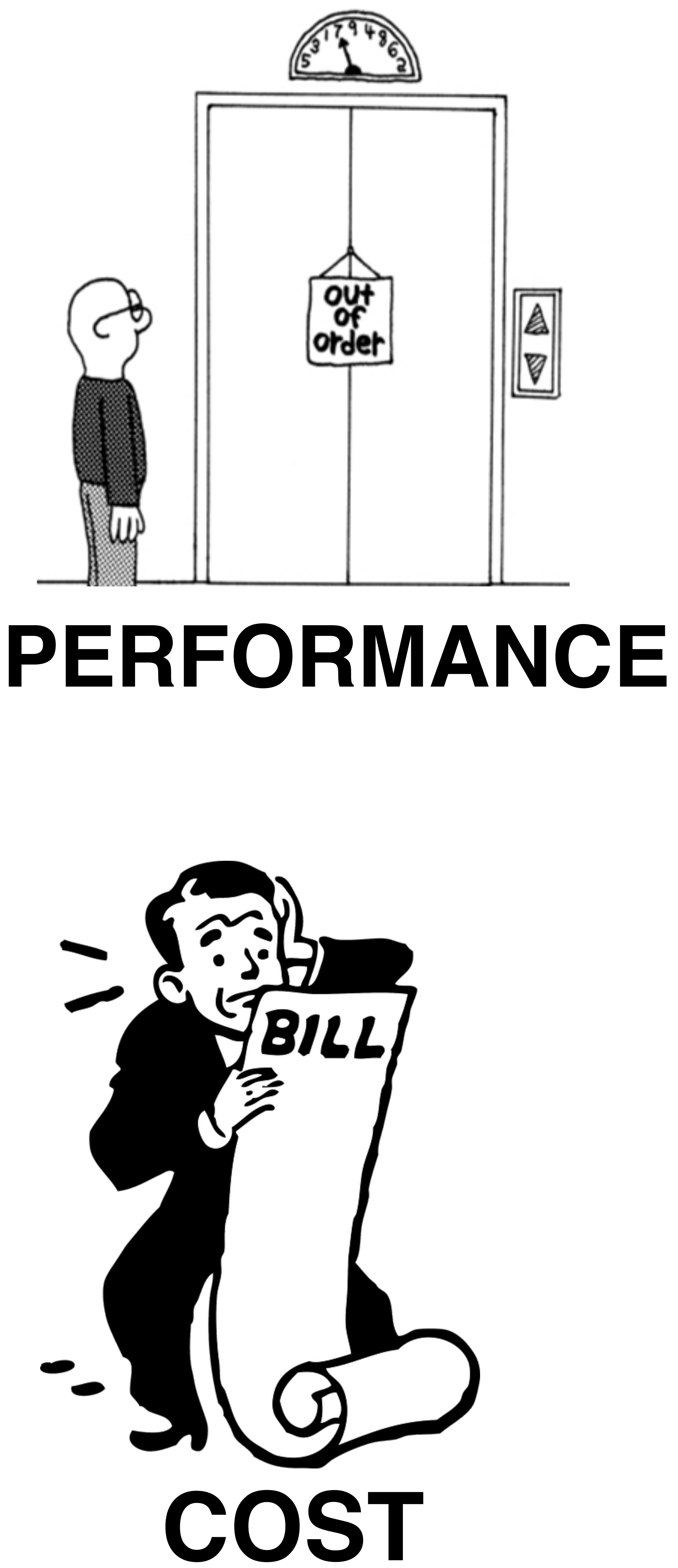
PERFORMANCE

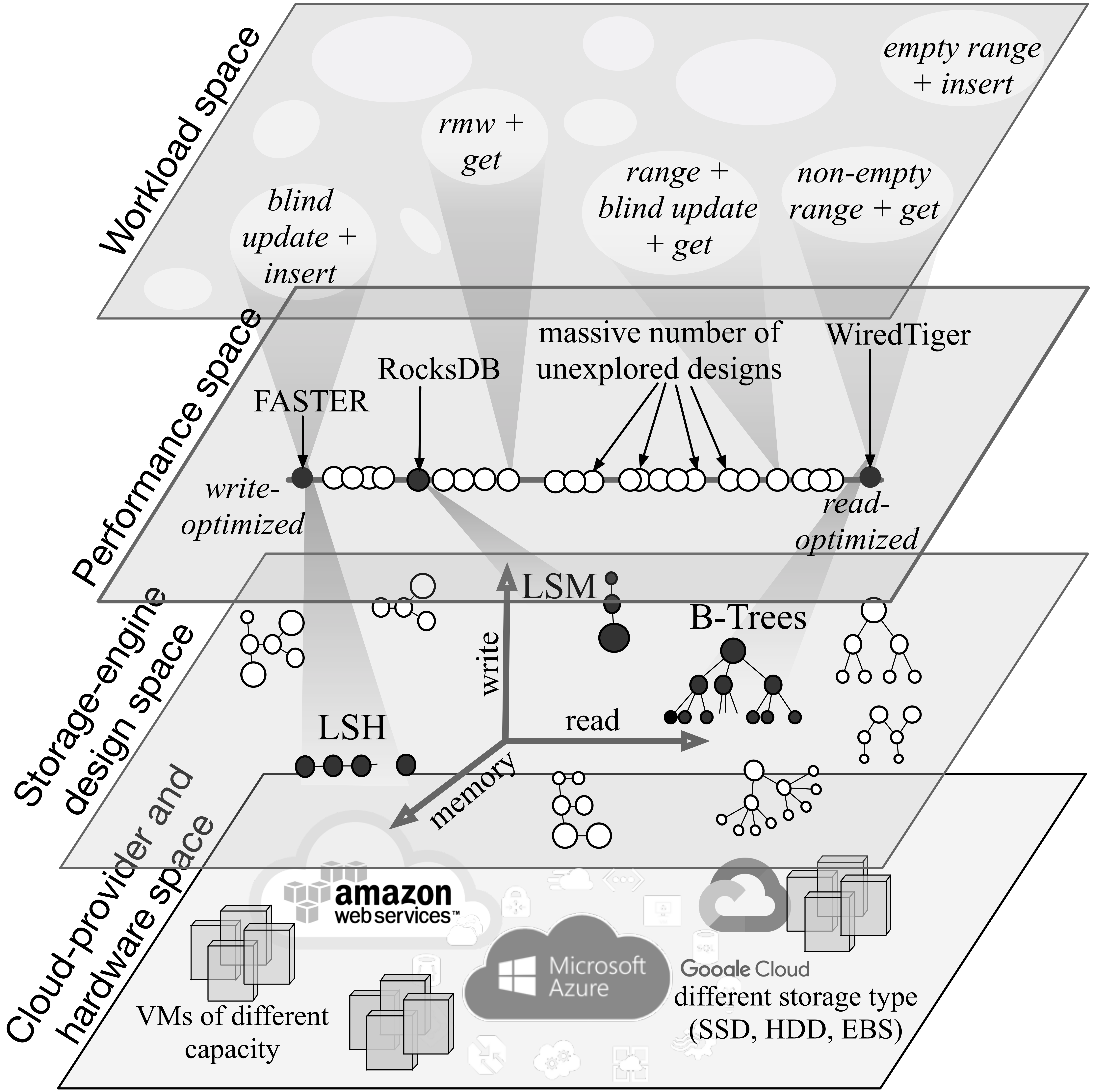


COST

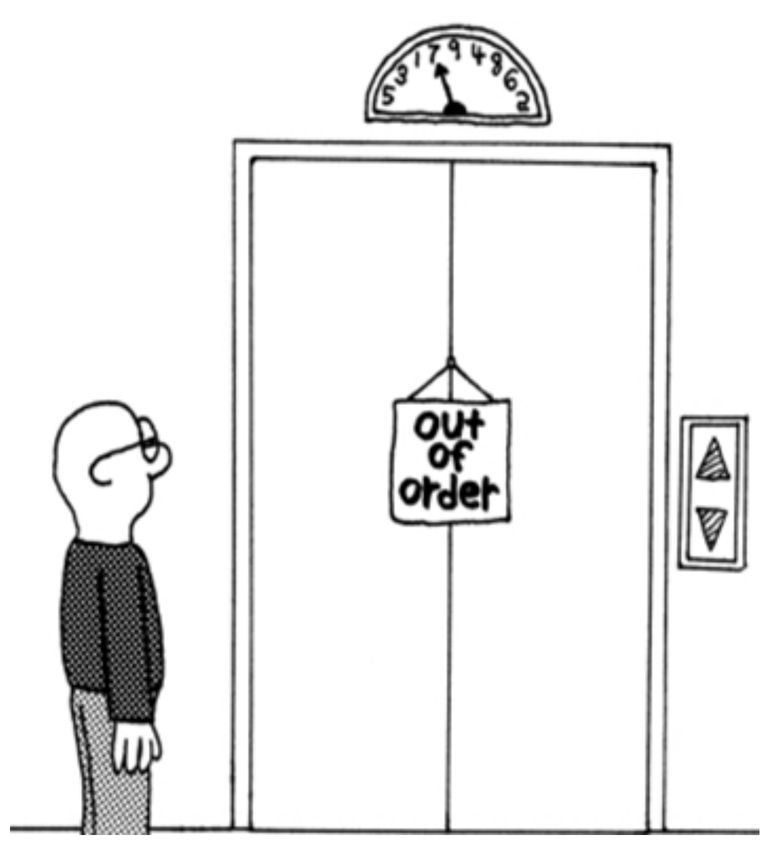


DASlab @ Harvard SEAS

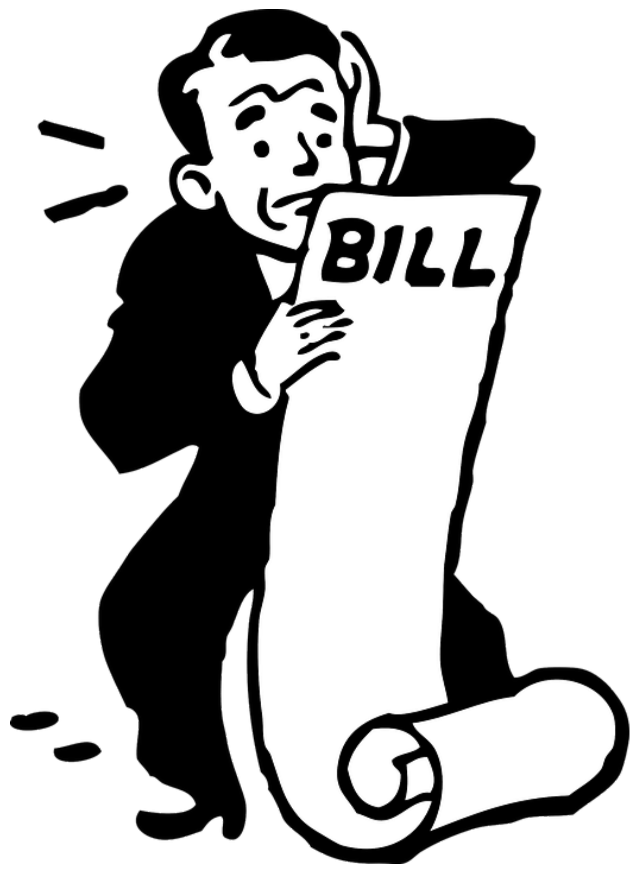




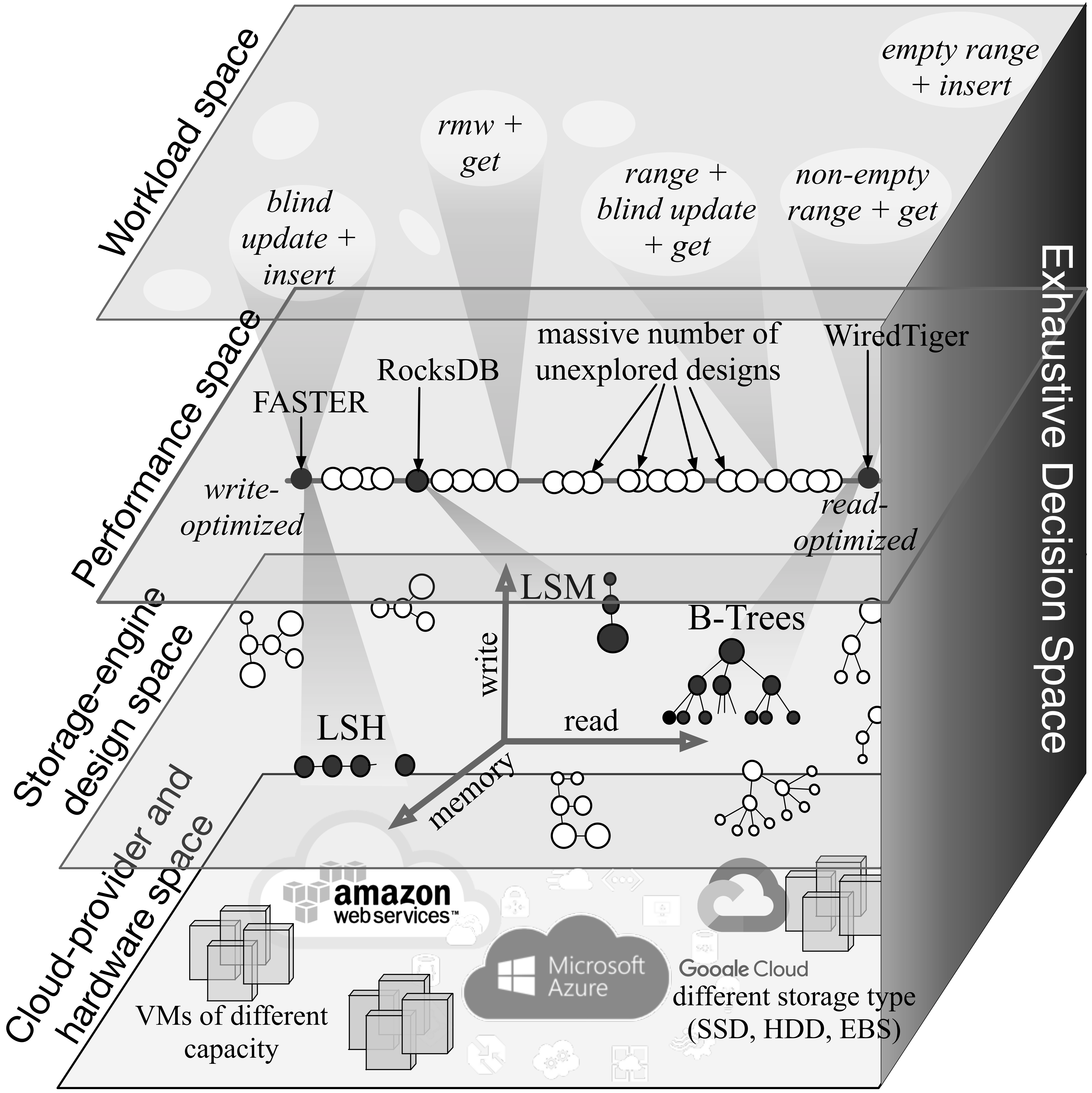
DASlab @ Harvard SEAS



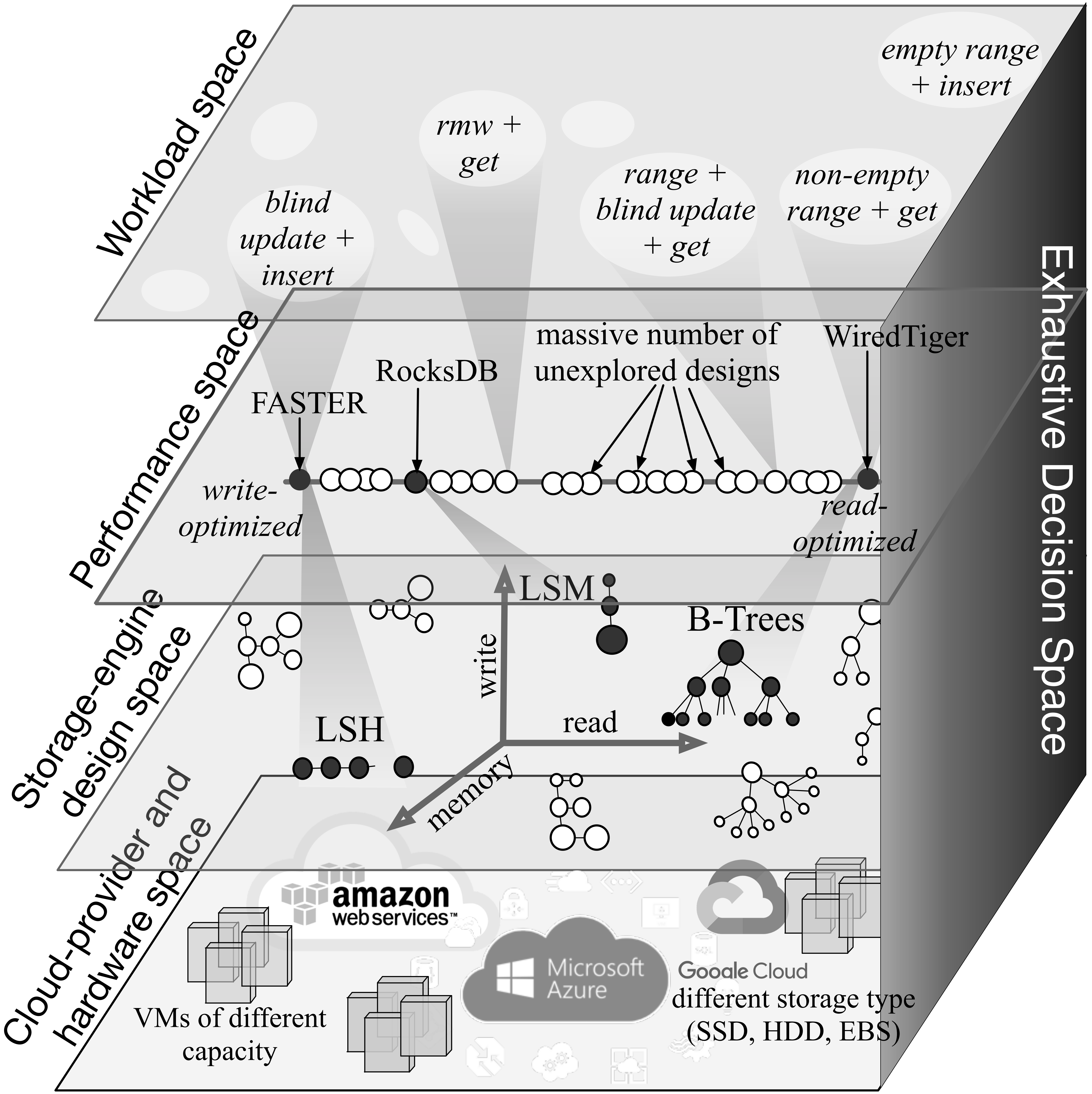
PERFORMANCE



COST



DASlab @ Harvard SEAS



Complex

Vast
(10^35 possibilities)

Manual
(Limited exploration of systems)

DASlab @ Harvard SEAS

Complex

Vast
(10^35 possibilities)

Manual

GOAL: To create the perfect data system tailored for each context

Gilad David Maayan
Posted on Sep 25, 2020 • Updated on Dec 10, 2020

AWS to Azure: Making the Move

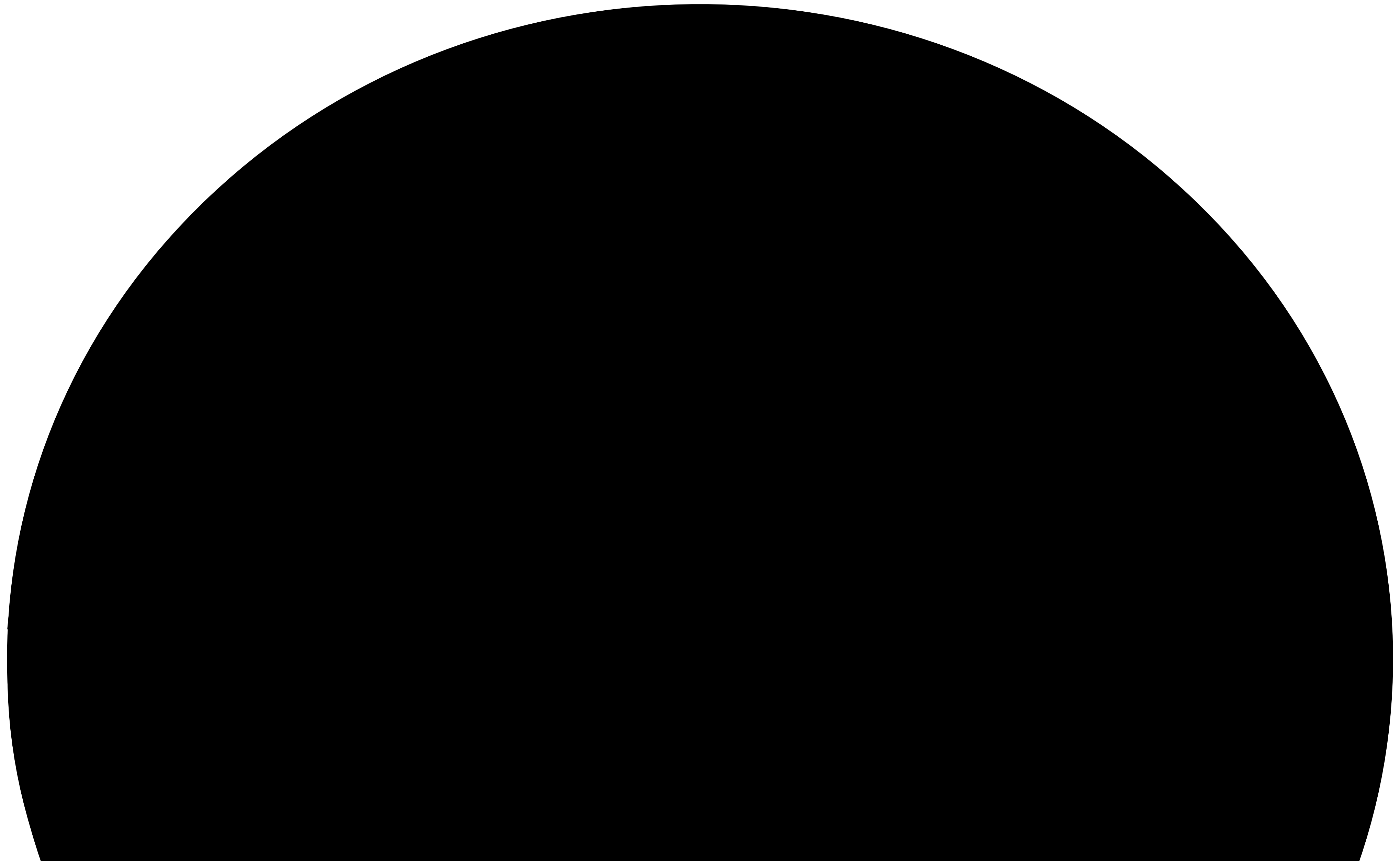
#azure #aws

Both Amazon Web Services and Microsoft Azure are considered top cloud

DASlab @ Harvard SEAS

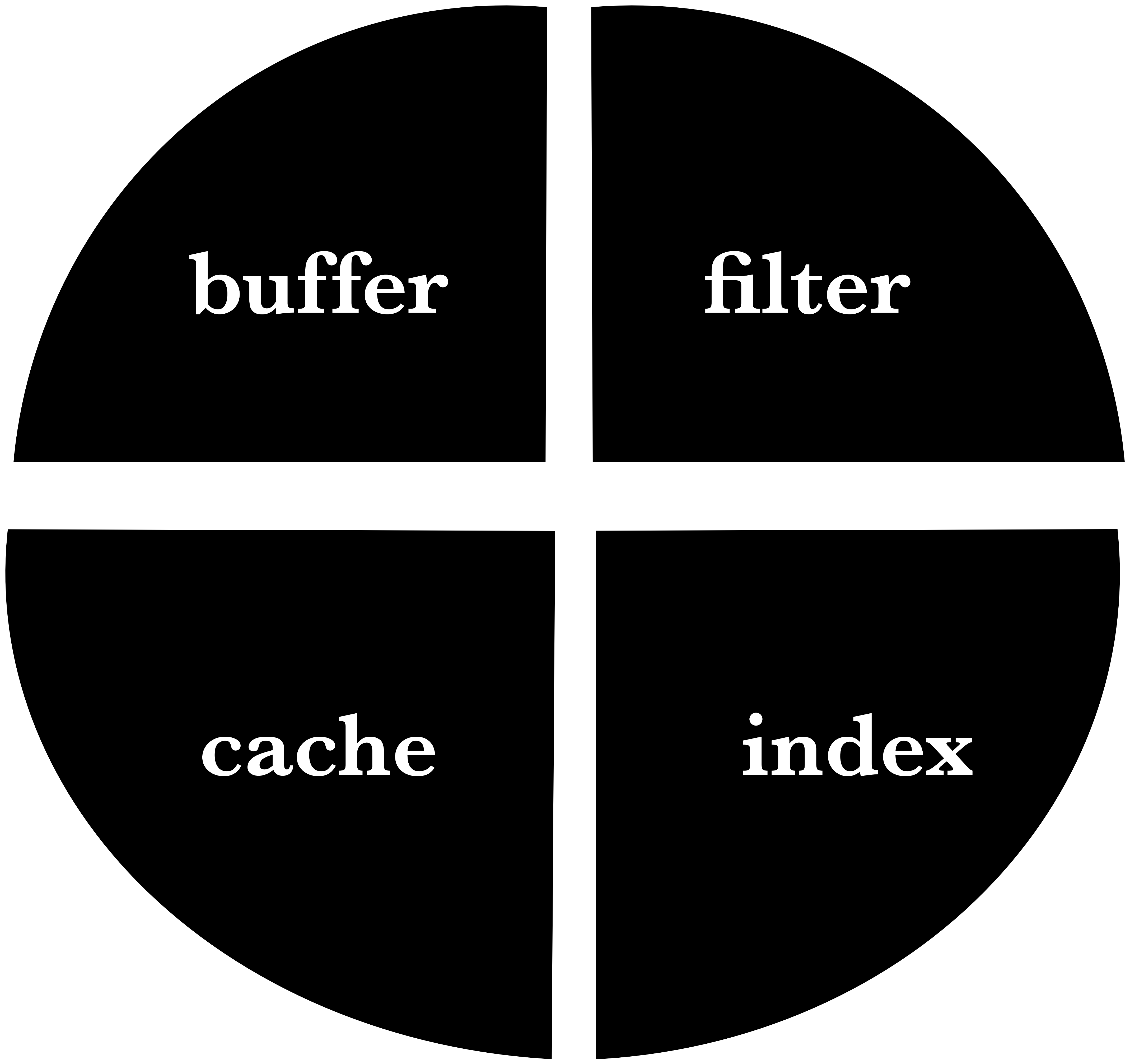
Key Intuition

Storage engine = amalgamation of data structures



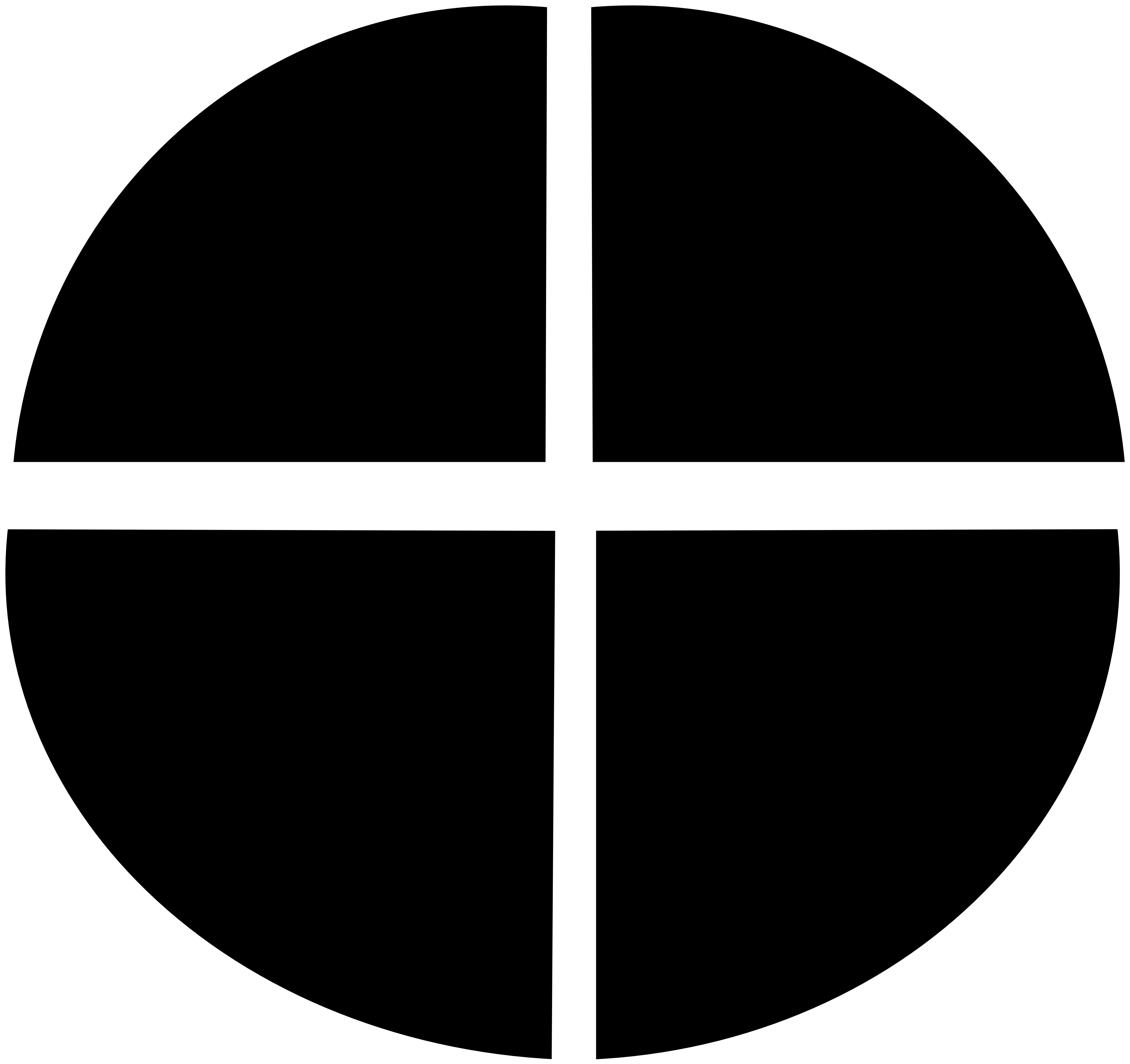
Key Intuition

Storage engine = amalgamation of data structures



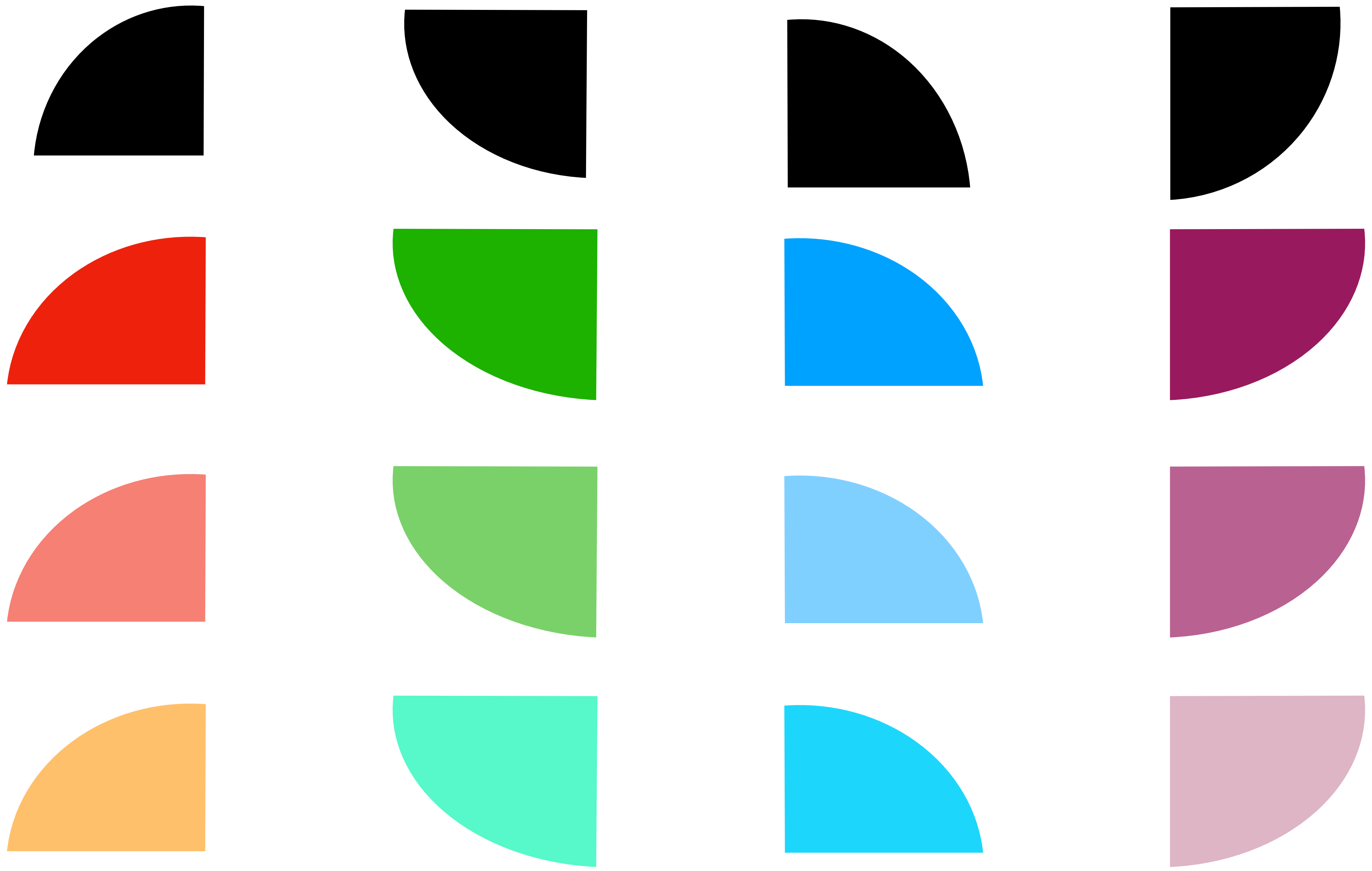
Key Intuition

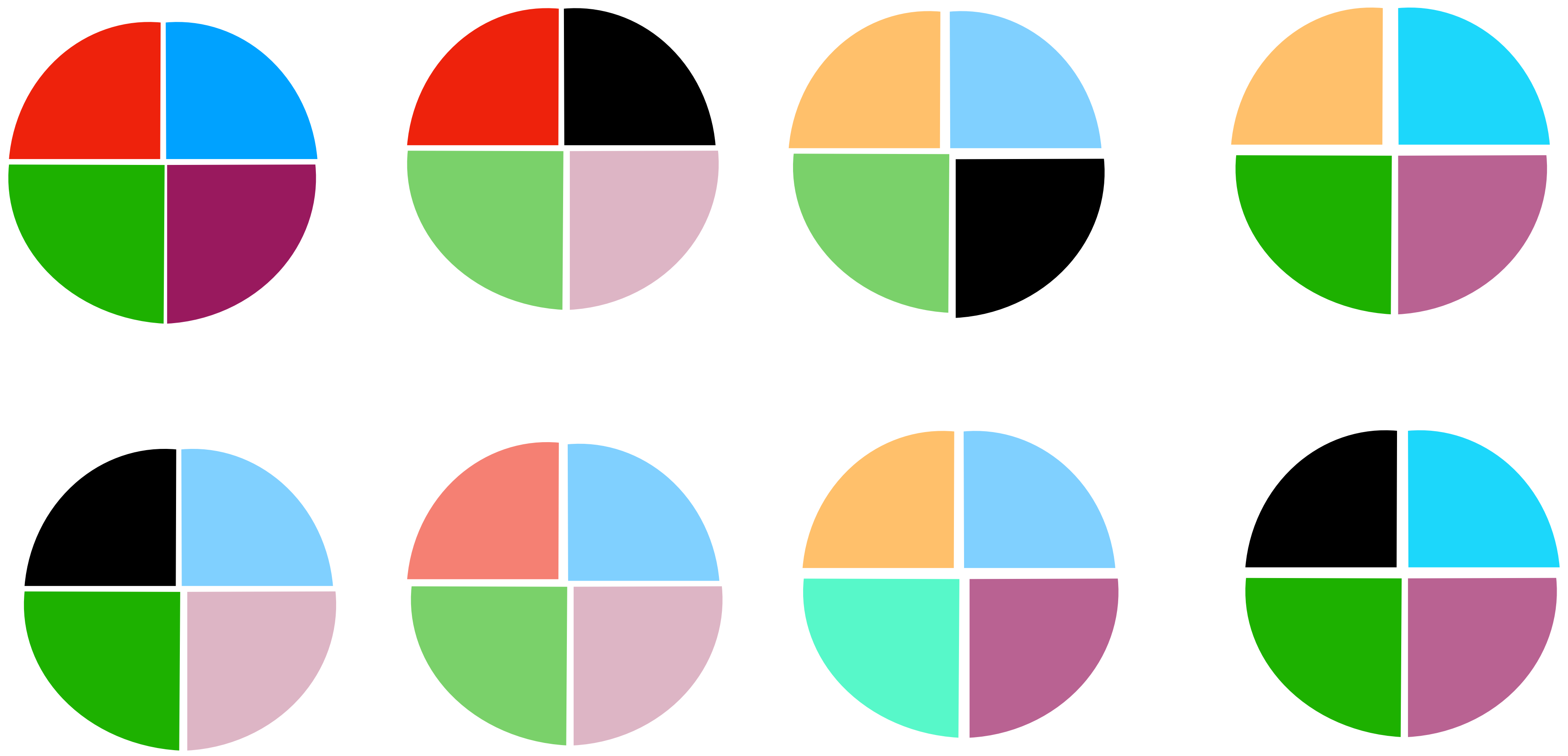
Storage engine = amalgamation of data structures



Key Intuition

Every-growing space of data structures → Massive design space of key-value stores



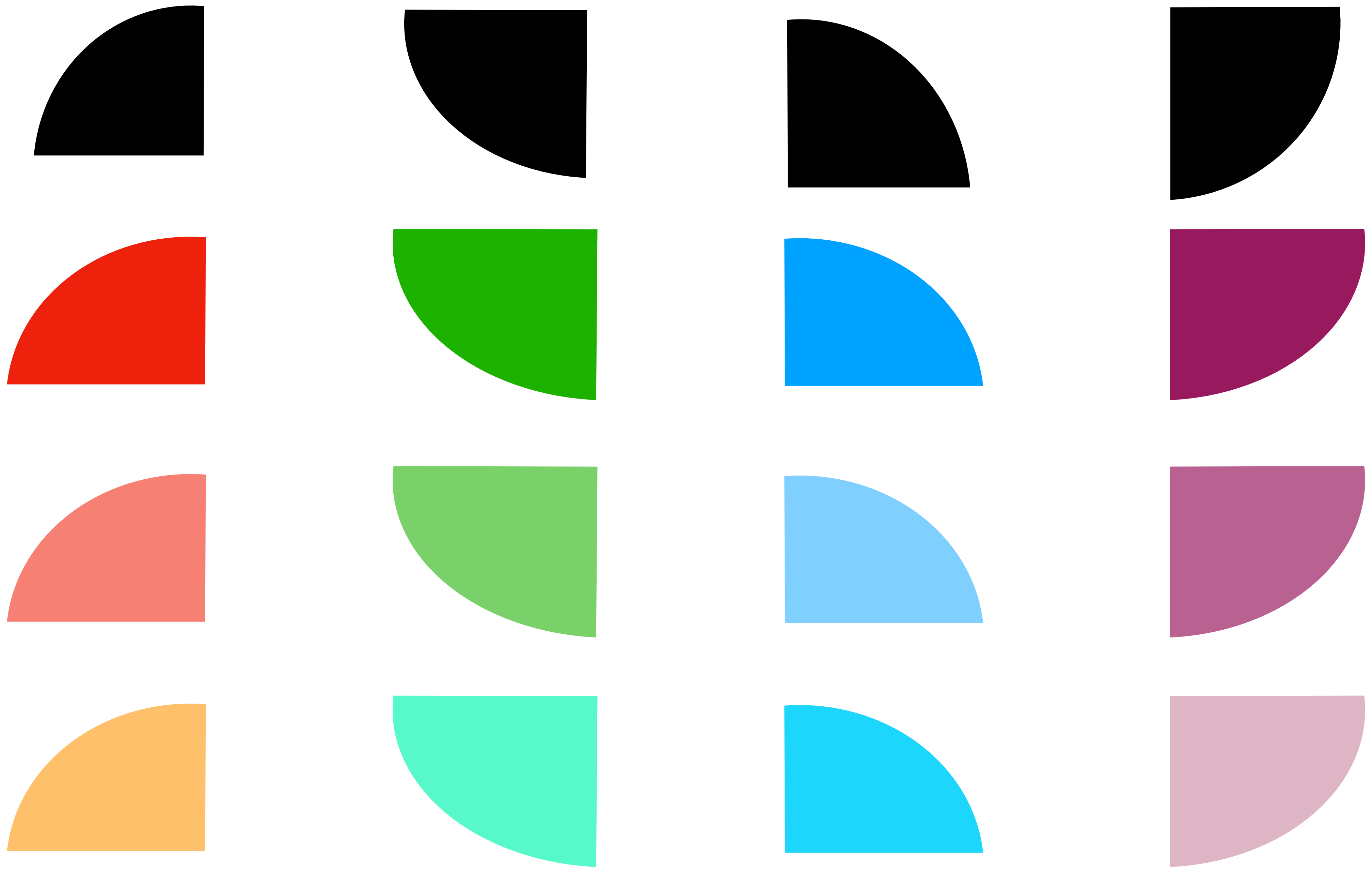


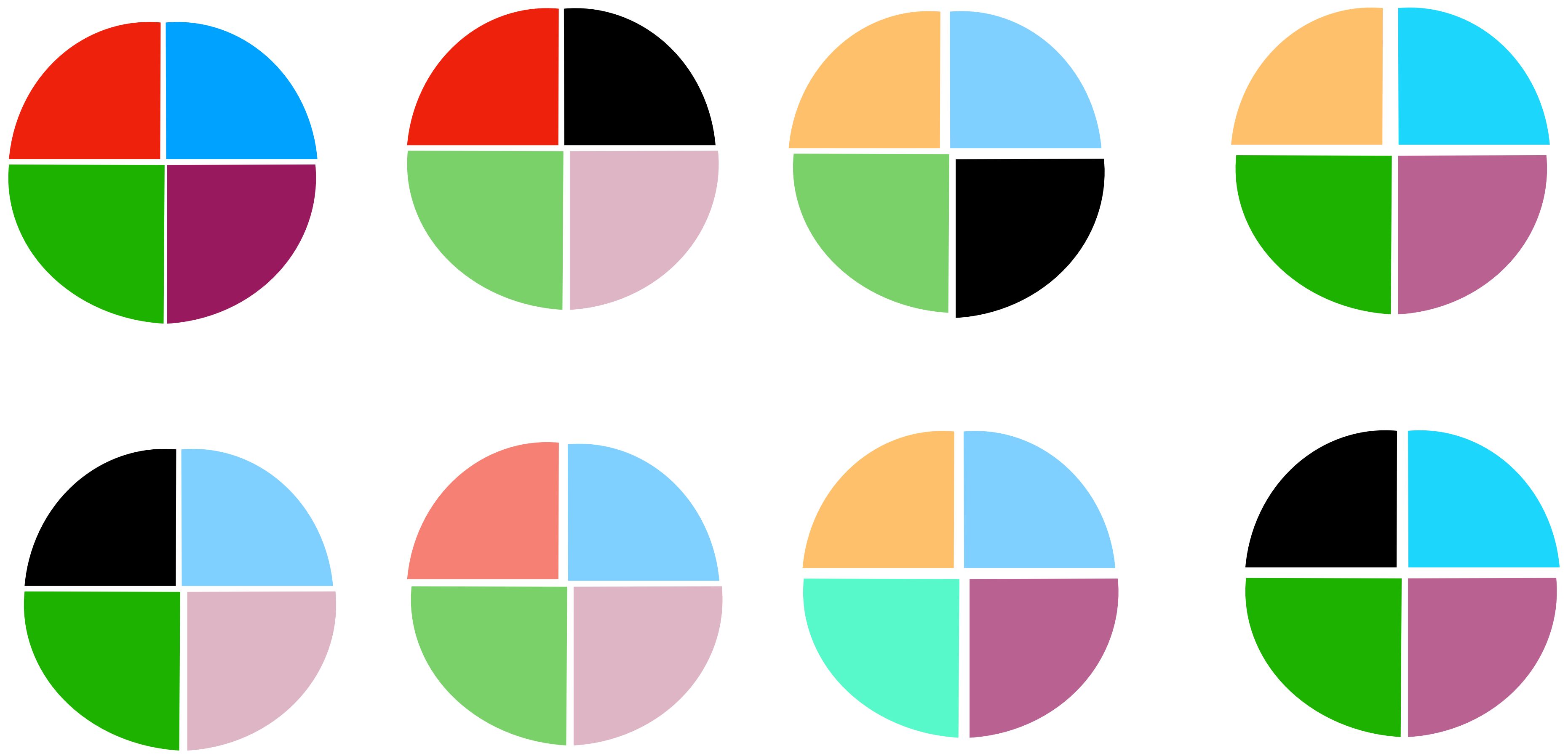
and more …

What if we could **reason** about the **massive design space** of key-value storage engines?

Key Intuition

Every-growing space of data structures → Massive design space of key-value stores

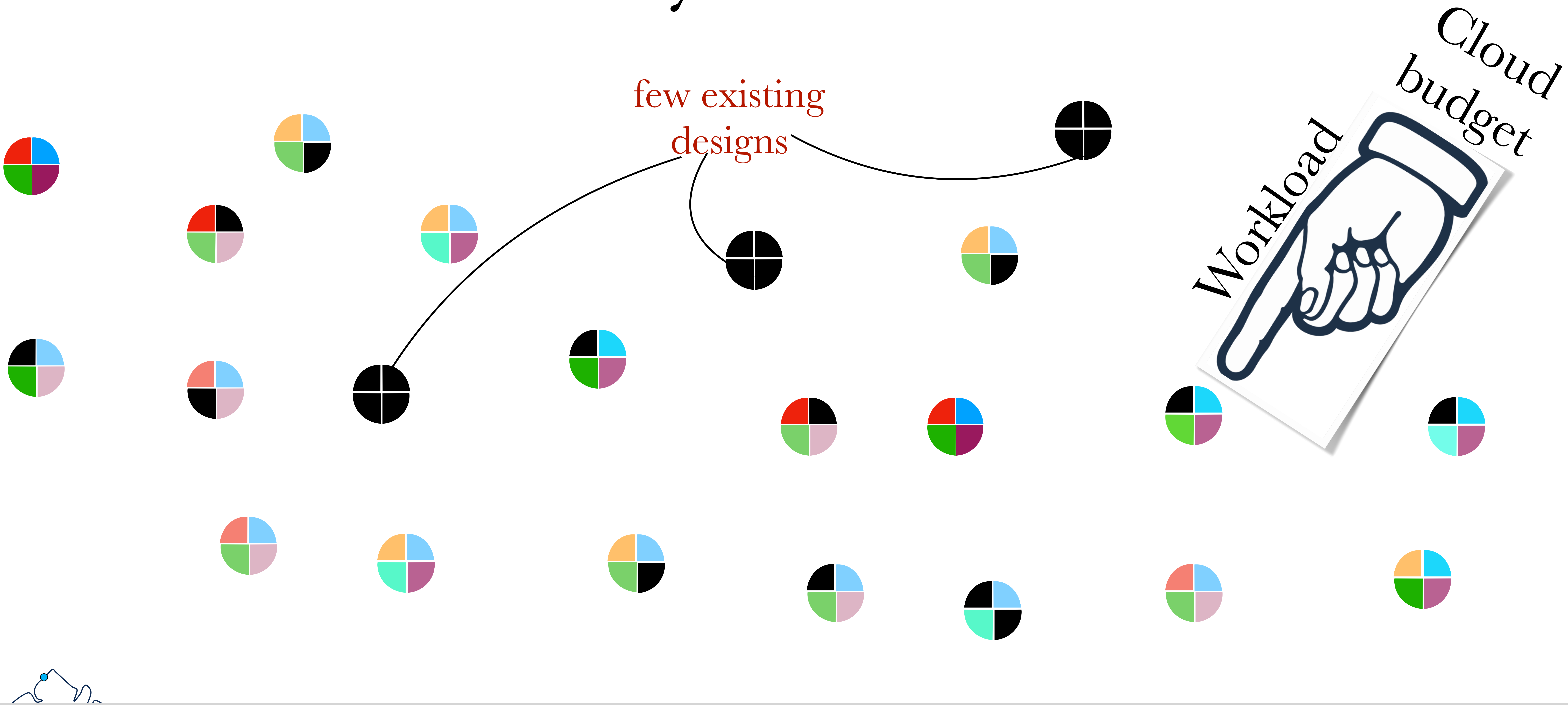




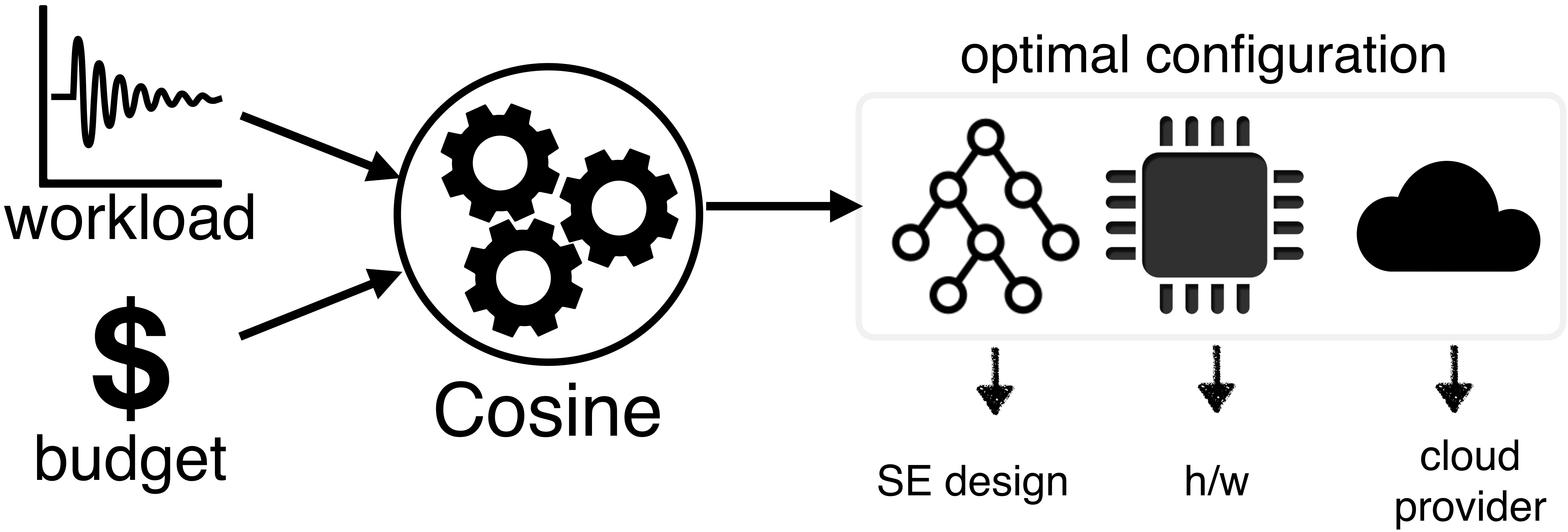
and more …

What if we could **reason** about the **massive design space** of key-value storage engines?

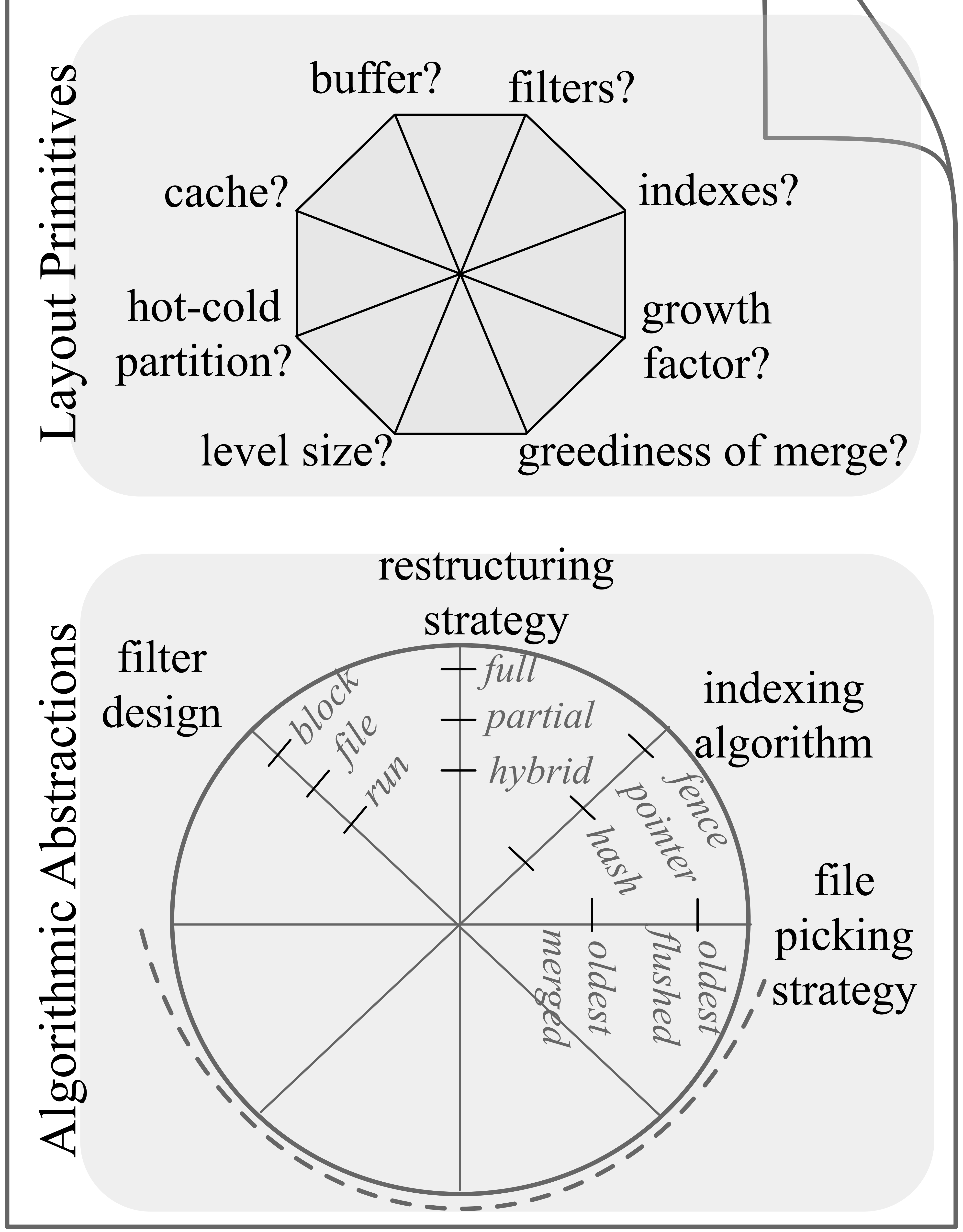
Key Intuition

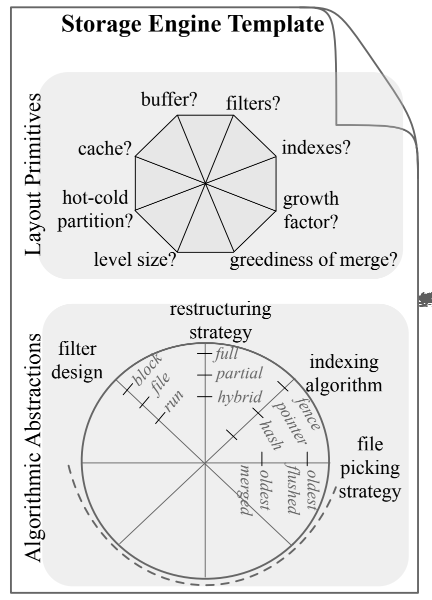


What if we could **reason** about the **massive design space** of key-value storage engines?


workload
$
budget
Cosine
optimal configuration
SE design
h/w
cloud
provider

Storage Engine Template




Storage Engine Template
Layout Primitives
buffer?
filters?
cache?
indexes?
hot-cold partition?
growth factor?
level size?
greediness of merge?
Algorithmic Abstractions
restructuring strategy
filter design
indexing algorithm
file picking strategy
full
partial
hybrid

MEMORY

DISK

MEMORY	DISK

storage pattern?
{flat logs, hierarchical}

growth factor?
[1, .., multiples of block size]

level size?
[1, .., L]

greediness of merge?
[1 (high), .., T (low)]

file size?
MB ... GB

hot-cold partition?

MEMORY	DISK

MEMORY

buffer? [1..M]

filters? Bloom? Cuckoo? … [1..M]

indexes? hash table? zone map? …

cache? [1..M]

DISK

storage pattern? {flat logs, hierarchical}

growth factor? [1, .., multiples of block size]

level size? [1, .., L]

greediness of merge? [1 (high), .., T (low)]

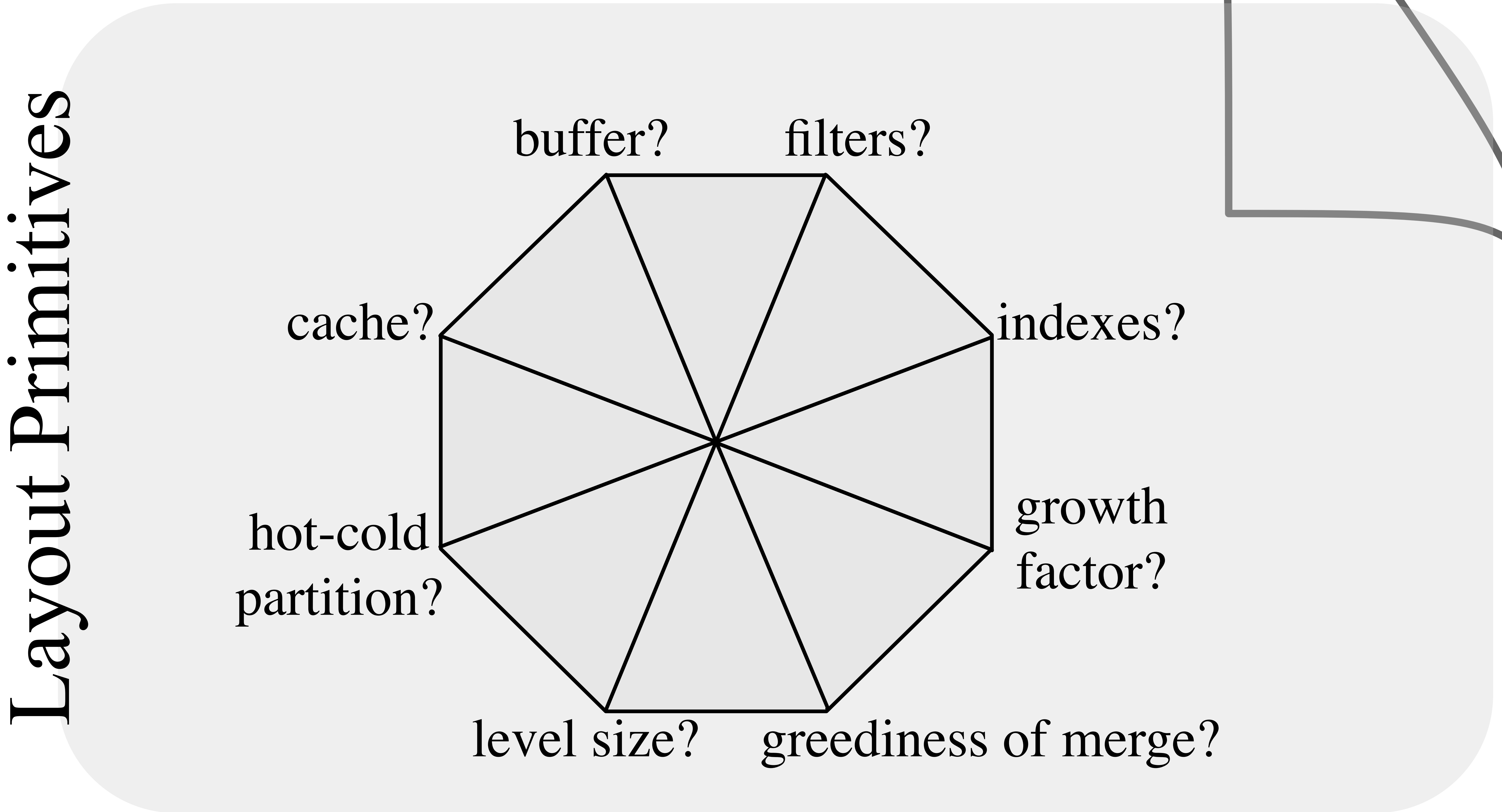
file size? MB … GB

hot-cold partition?

Superstructure

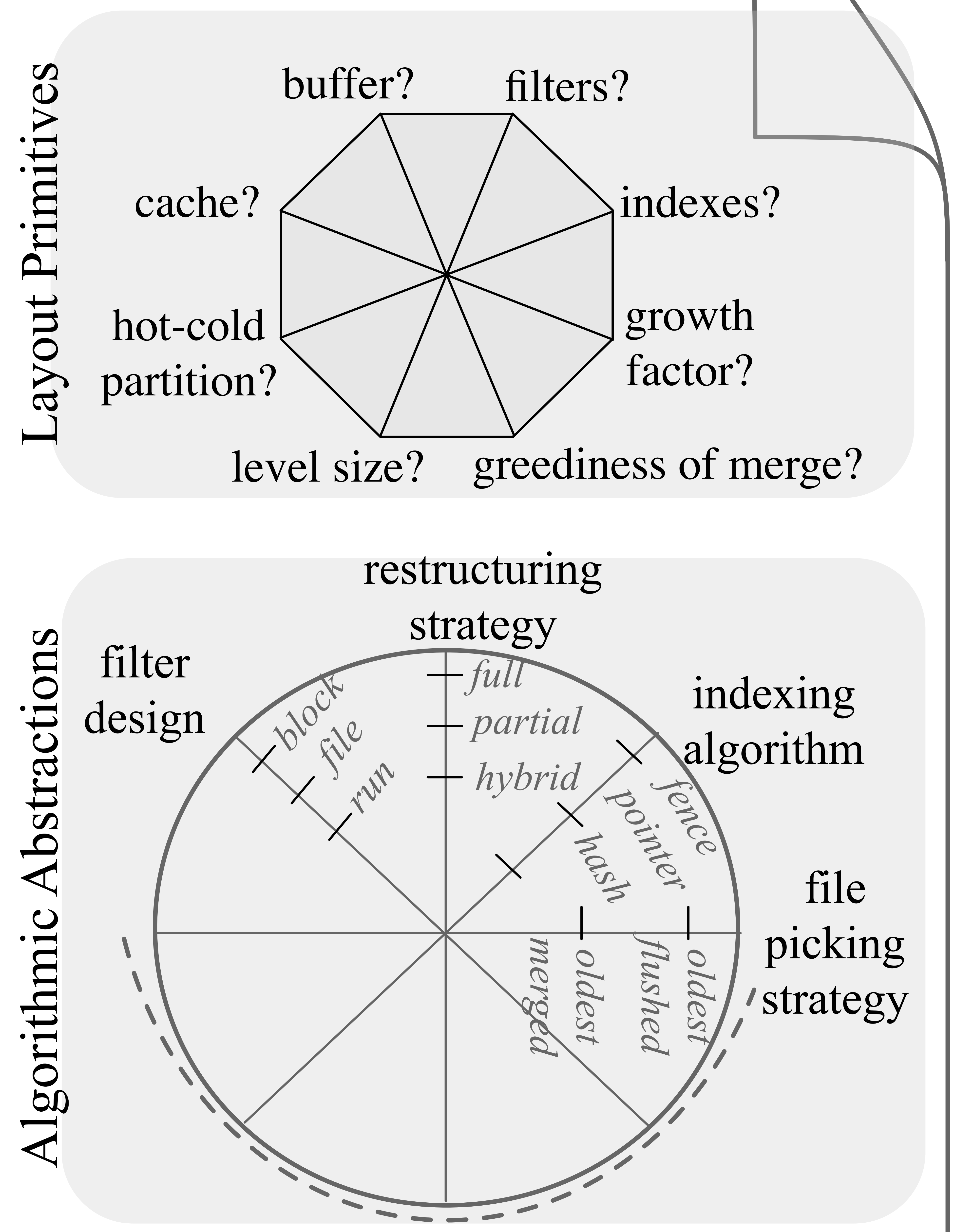

MEMORY
DISK
buffer?
filters?
indexes?
cache?
hot
cold
storage pattern?
growth factor?
level size?
greediness of merge?
hot-cold partition?
file size?
Leveled LSM
Tiered LSM
BTree
New layout

Storage Engine Template



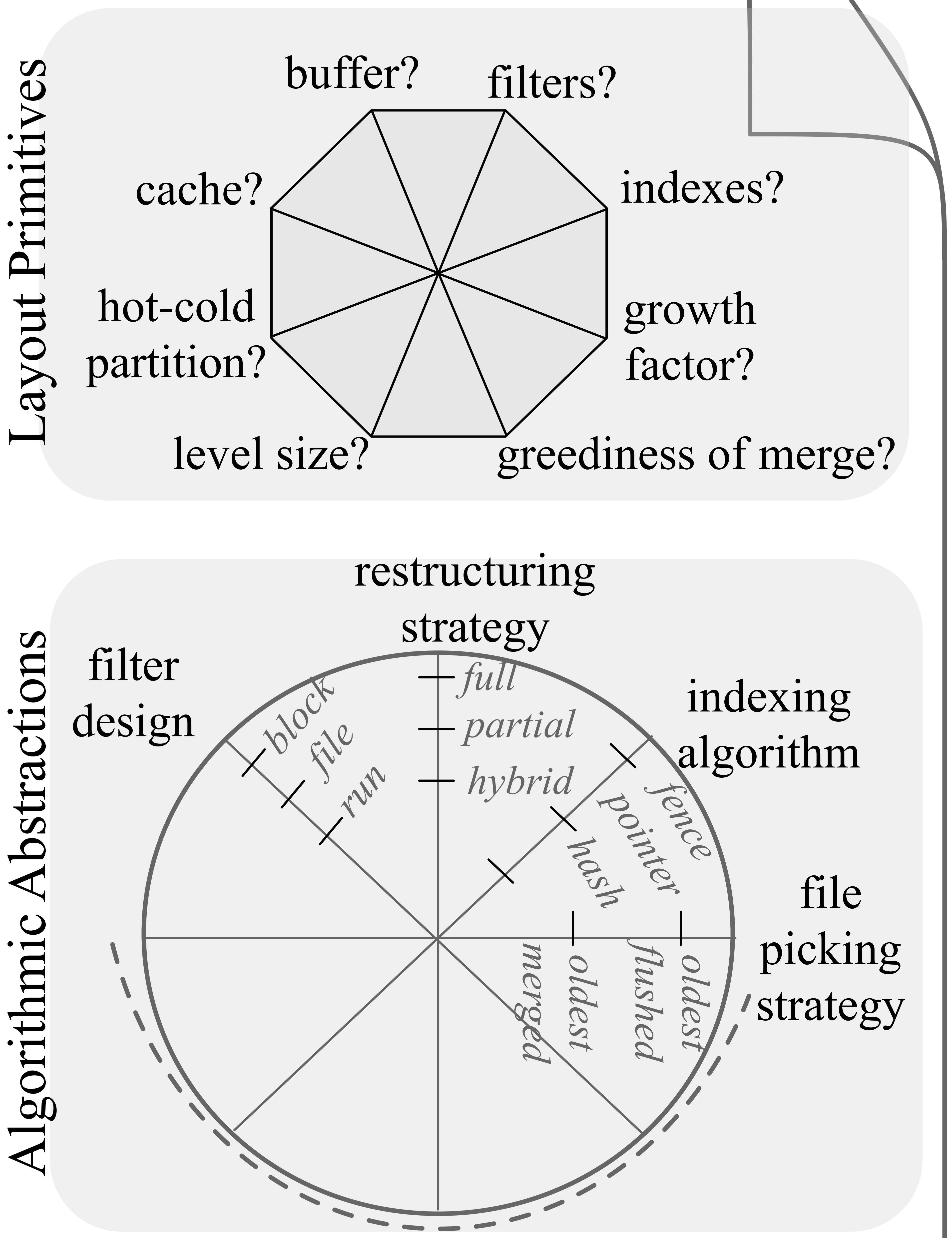
DASlab @ Harvard SEAS

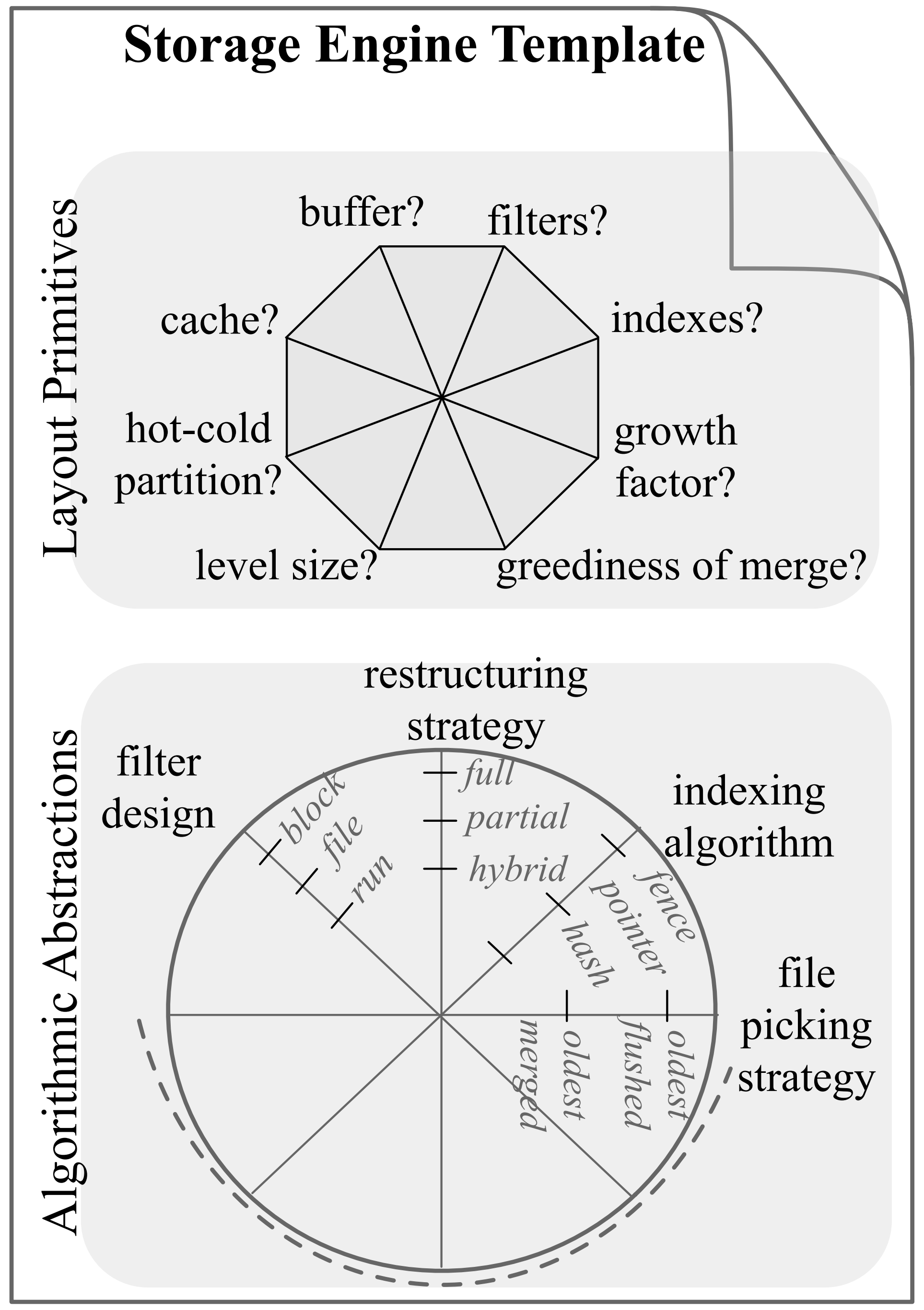
Storage Engine Template



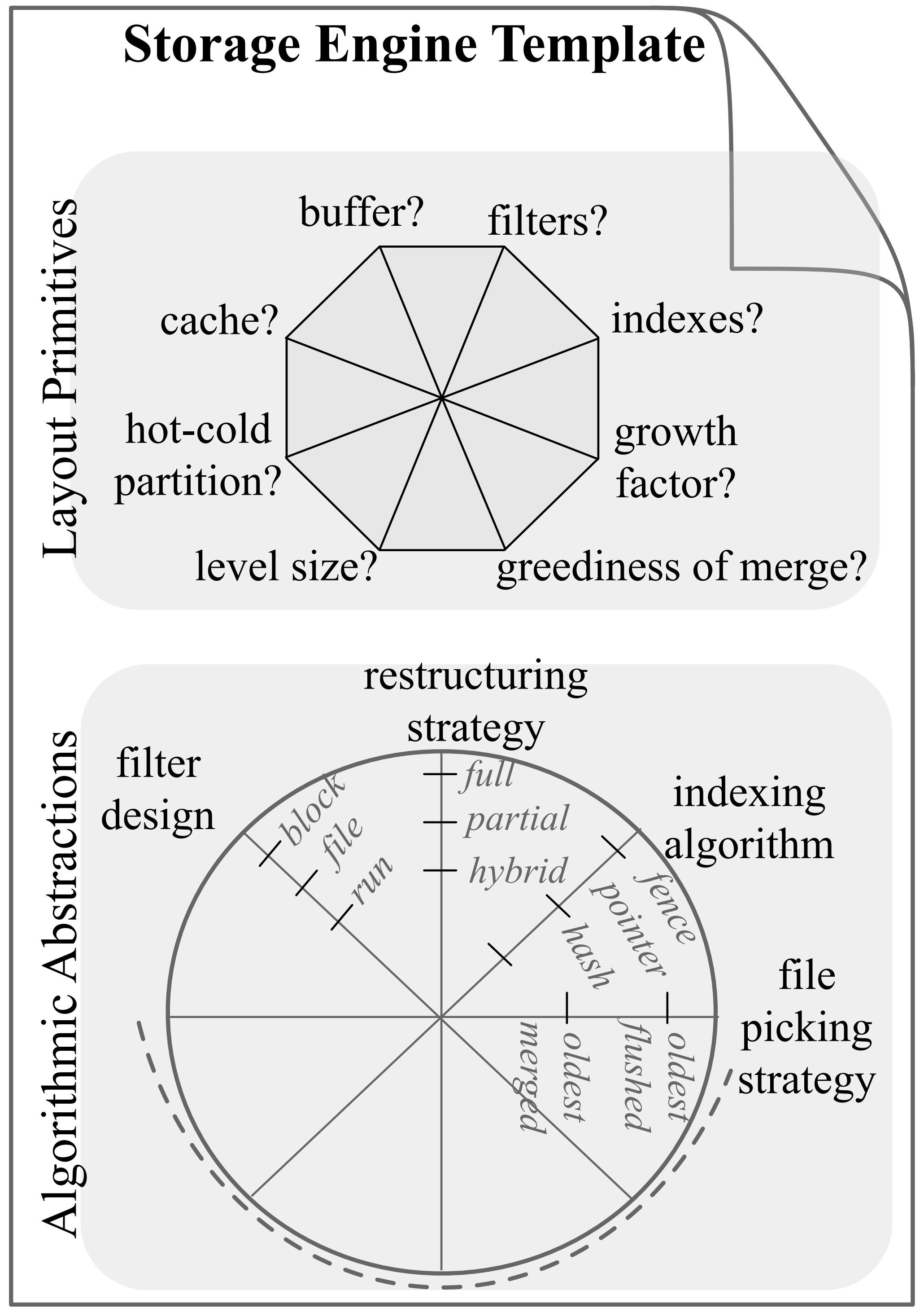
DASlab @ Harvard SEAS

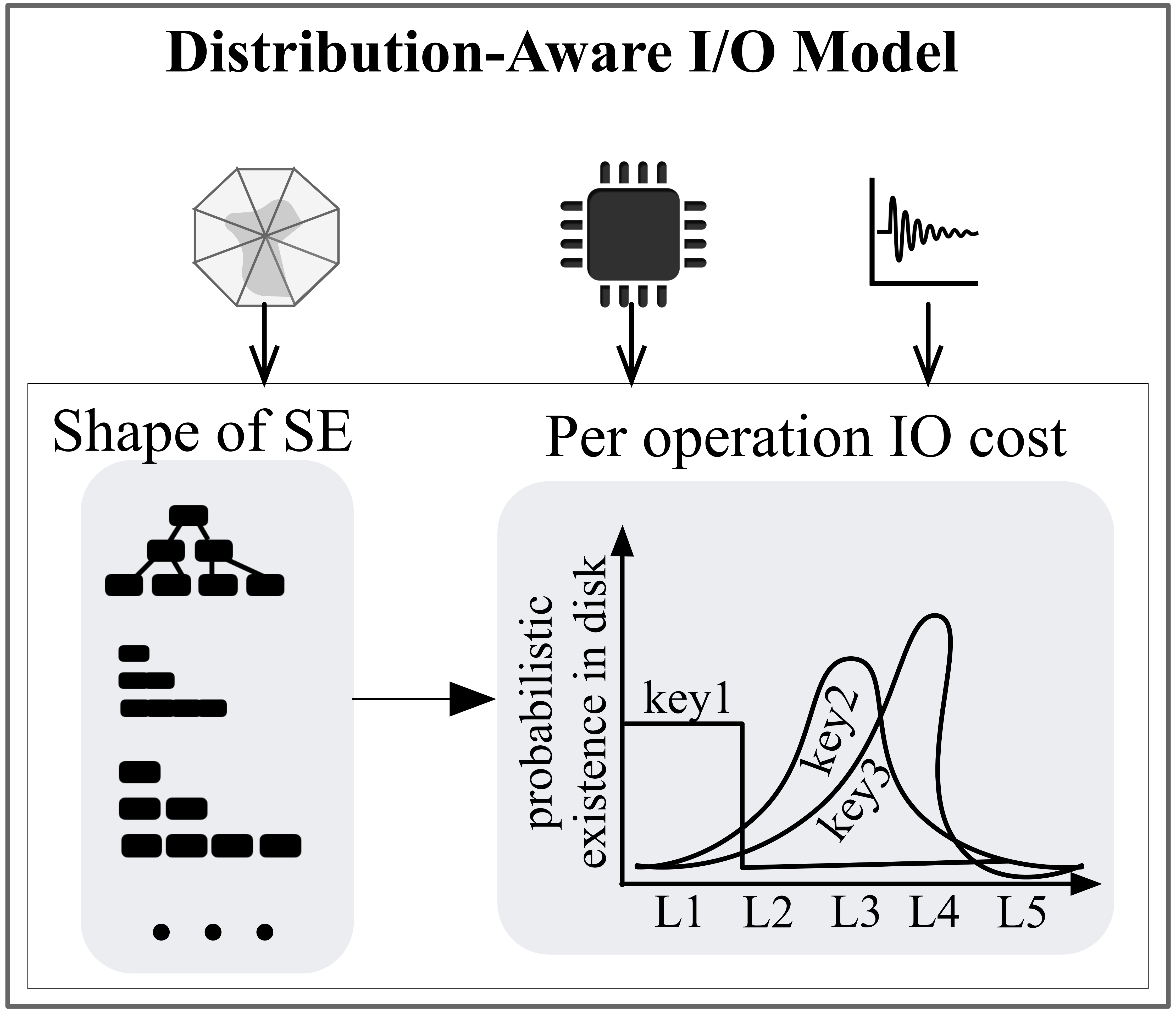
Storage Engine Template

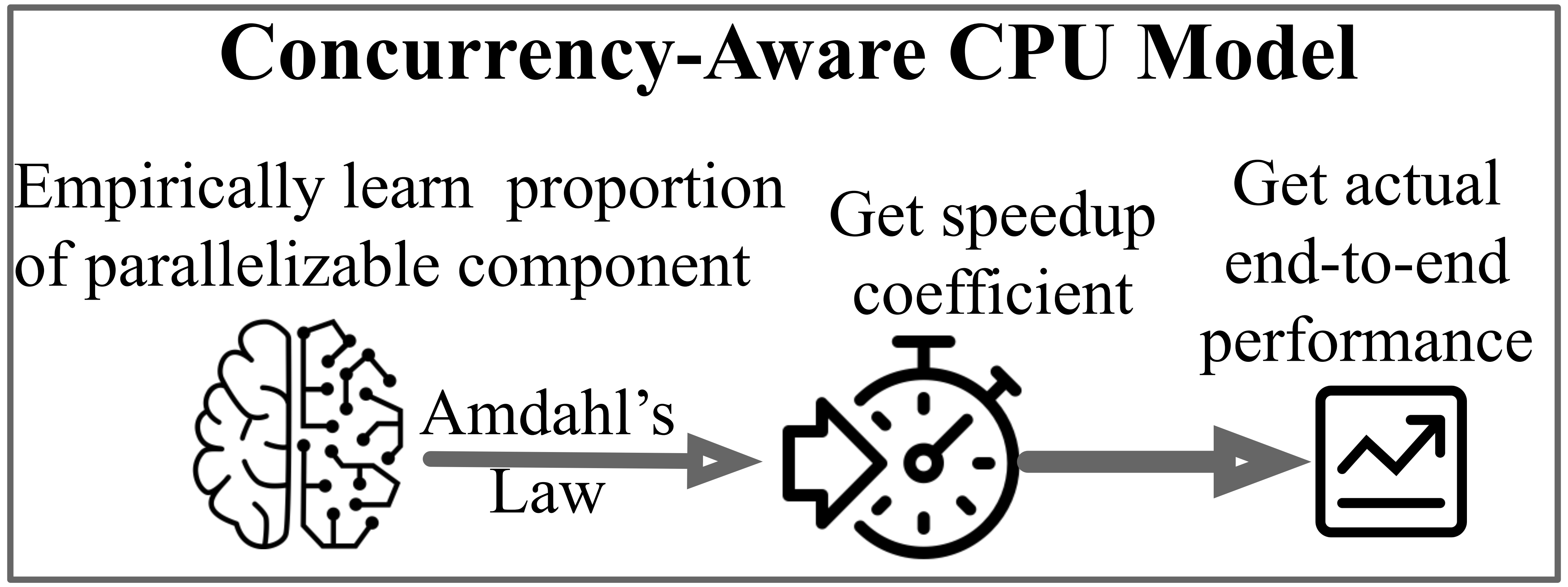




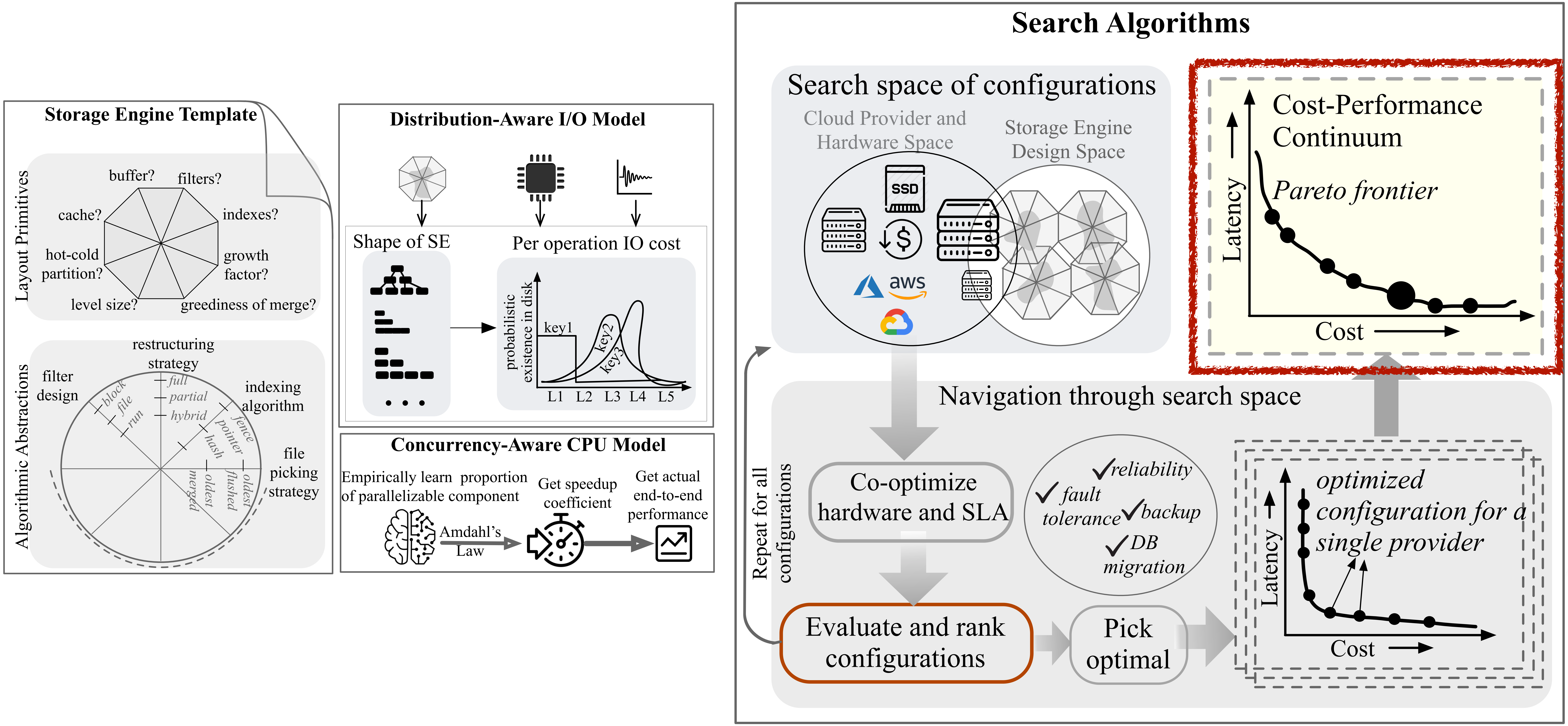
Category	#	Design Abstractions of Template	Type/Domain	Example templates for diverse data structures: RocksDB variants	WiredTiger variants	FASTER variants	A new design
LAYOUT PRIMITIVES; Design and hardware specification; *initialized by search through engine design space*	1.	**Key size:** Denotes the size of keys in the workload.	unsigned int	auto-configured from the sample workload			
	2.	**Value size:** Denotes the size of values in the workload. All values are accepted as variable-length strings.	string/slice *max size set to 1 GB*	auto-configured from the sample workload			
	3.	**Size ratio (T):** The maximum number of entries in a block (e.g. growth factor in LSM trees or fanout of B-trees.	unsigned integer | function (func)	[2,.. 32]	[32, 64, 128, 256, ..]	[1000, 1001, …] (T is large)	2
	4.	**Runs per hot level (K):** At what capacity hot levels are compacted. Rule: should be less than size ratio.	unsigned int	[1.. T]		[T-1]	7
	5.	**Runs per cold level (Z):** At what capacity cold levels are compacted. Rule: should be less than size ratio.	unsigned int	[1.. T]	[1]		32
	6.	**Logical block size (B):** Number of consecutive disk blocks.	unsigned int	[2048, 4096, …]			
	7.	**Buffer capacity** (M_B)**:** Denotes the amount of memory allocated to in-memory buffer/memtables. Configurable w.r.t file size.	64-bit floating point | function (func)	[64 MB, 128 MB, …]	[1 MB, 2 MB, …]	[64 MB, 128 MB, …]	h/w dependent
	8.	**Indexes** (M_{FP})**:** Amount of memory allocated to indexes (fence pointers/hashtables).	64-bit floating point | function (func)	memory to cover L	memory for first level	memory for hash table	h/w dependent
	9.	**Bloom filter memory** (M_{BF})**:** Denotes the bits/entry assigned to Bloom filters.	64-bit float | func(FPR)	10 bits/key			func(FPR)
ALGORITHMIC ABSTRACTIONS; Data access; *derived with empirically verified rules*	10.	**Bloom filter design:** Denotes the granularity of Bloom filters, e.g., one Bloom filter instance per block or per file or per run. The default is file.	block | file | run	file			file
	11.	**Compaction/Restructuring algorithm:** Full does level-to-level compaction; partial is file-to-file; and hybrid uses both full and partial at separate levels.	partial | full | hybrid	full, partial	partial	partial	hybrid
	12.	**Run strategy:** Denotes which run to be picked for compaction (only for partial/ hybrid compaction).	first | last_full | fullest	first, fullest, last_full		first	fullest
	13.	**File picking strategy:** Denotes which file to be picked for compaction (for partial/ hybrid compaction). For LSM-trees we set default to dense_fp as it empirically works the best. B-trees pick the first file found to be full. LSH-table restructures at the granularity of runs.	oldest_merged | oldest_flushed | dense_fp | sparse_fp | choose_first	dense_fp	choose_first		dense_fp (hot), choose_first (cold)
	14.	**Merge threshold:** If a level is more than x% full, a compaction is triggered.	64-bit floating point	[0.7..1]	0.5		0.75
	15.	**Full compaction levels:** Denotes how many levels will have full compaction (only for hybrid compaction). The default is set to 2.	unsigned integer | function (func)	[1..L]			L-Y (from optimal config)
Parallelism	16.	**No. of CPUs:** Number of available cores to use in a VM.	unsigned int	Use all available cores			
	17.	**No of threads:** Denotes how many threads are used to process the workload.	unsigned int	Use 1 thread per CPU core			

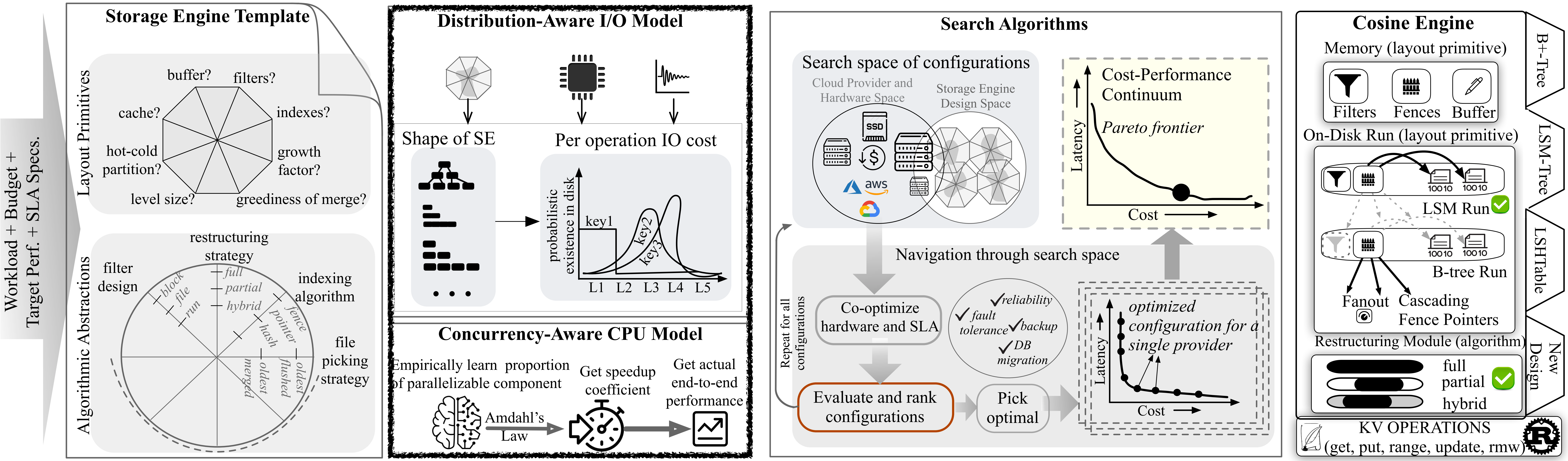




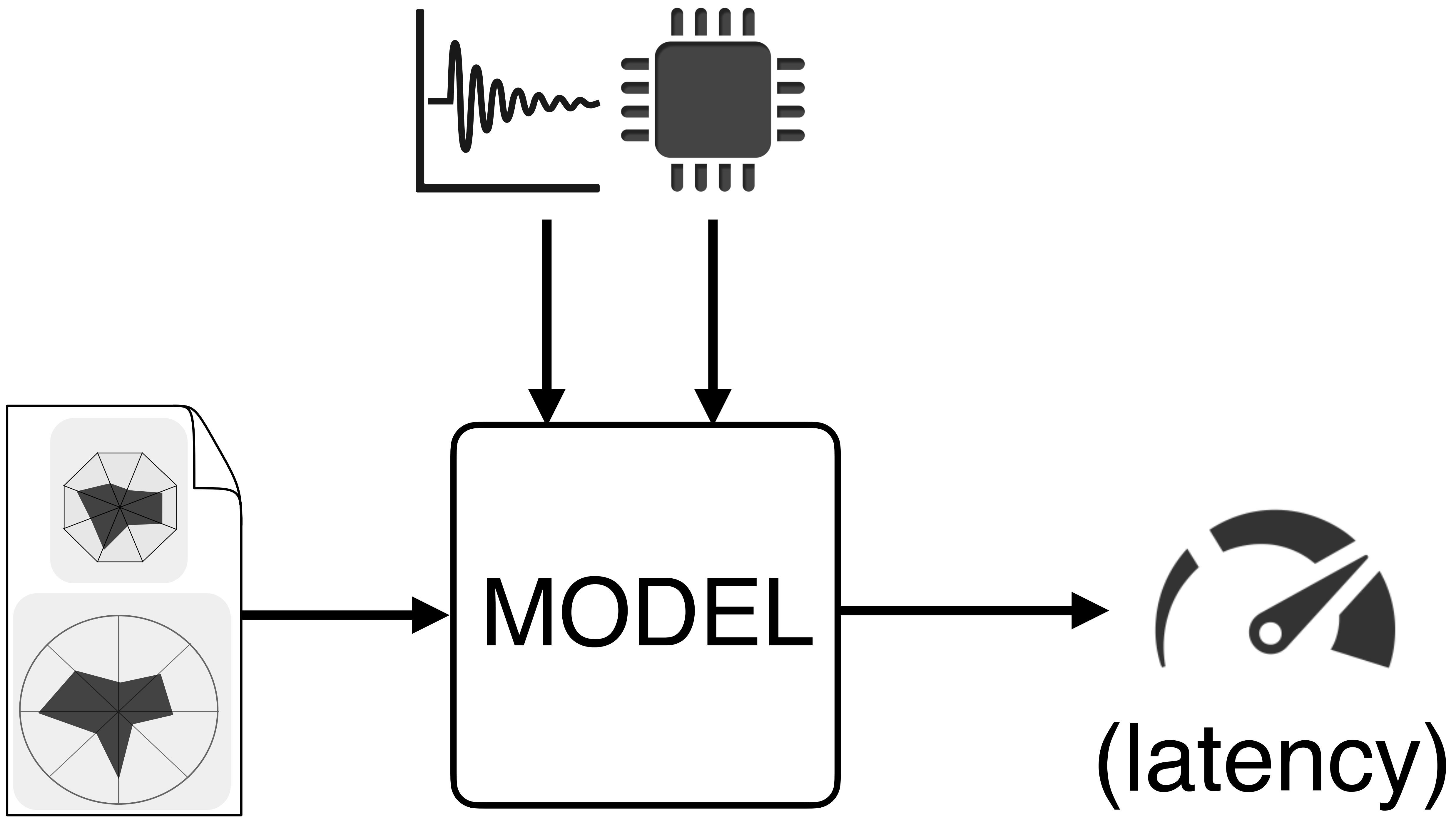


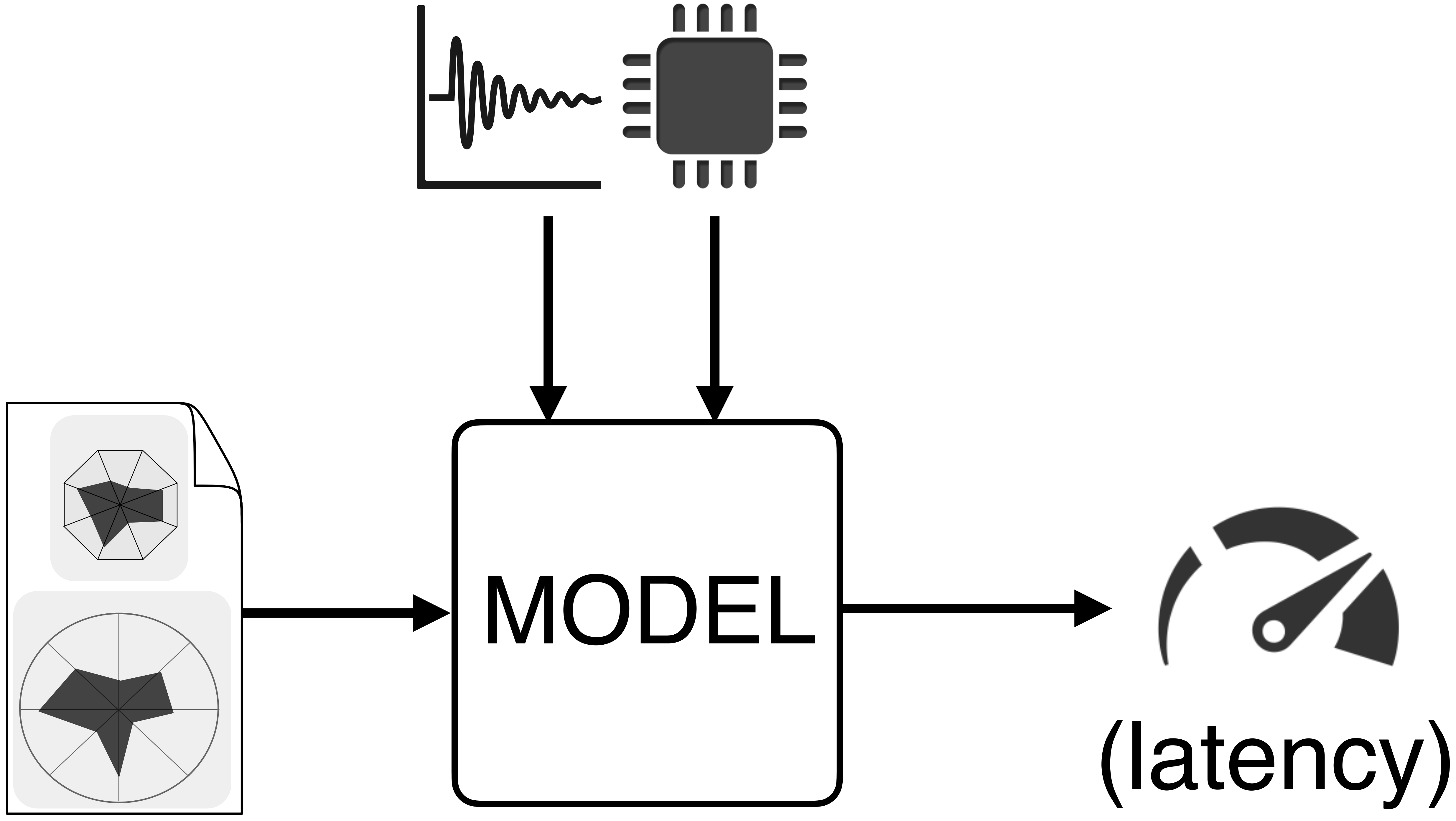
DASlab @ Harvard SEAS

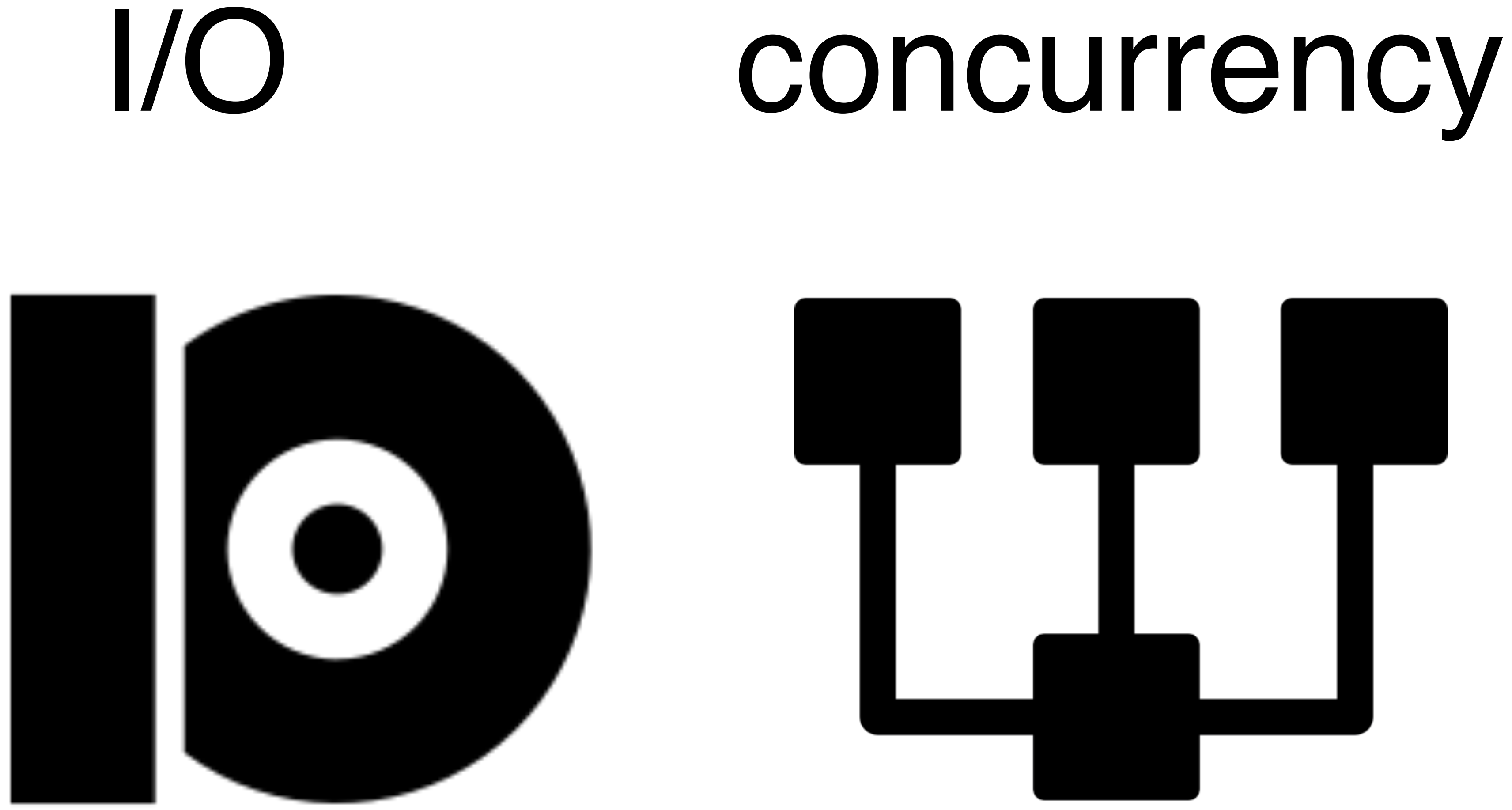

Storage Engine Template
Layout Primitives
buffer?
filters?
cache?
indexes?
hot-cold partition?
growth factor?
level size?
greediness of merge?
Algorithmic Abstractions
restructuring strategy
filter design
indexing algorithm
file picking strategy
Distribution-Aware I/O Model
Shape of SE
Per operation IO cost
probabilistic existence in disk
key1
key2
key3
L1 L2 L3 L4 L5
Concurrency-Aware CPU Model
Empirically learn proportion of parallelizable component
Amdahl's Law
Get speedup coefficient
Get actual end-to-end performance
Search Algorithms
Search space of configurations
Cloud Provider and Hardware Space
Storage Engine Design Space
SSD
aws
Cost-Performance Continuum
Pareto frontier
Latency
Cost
Navigation through search space
Repeat for all configurations
Co-optimize hardware and SLA
reliability
fault tolerance
backup
DB migration
Evaluate and rank configurations
Pick optimal
optimized configuration for a single provider
Latency
Cost

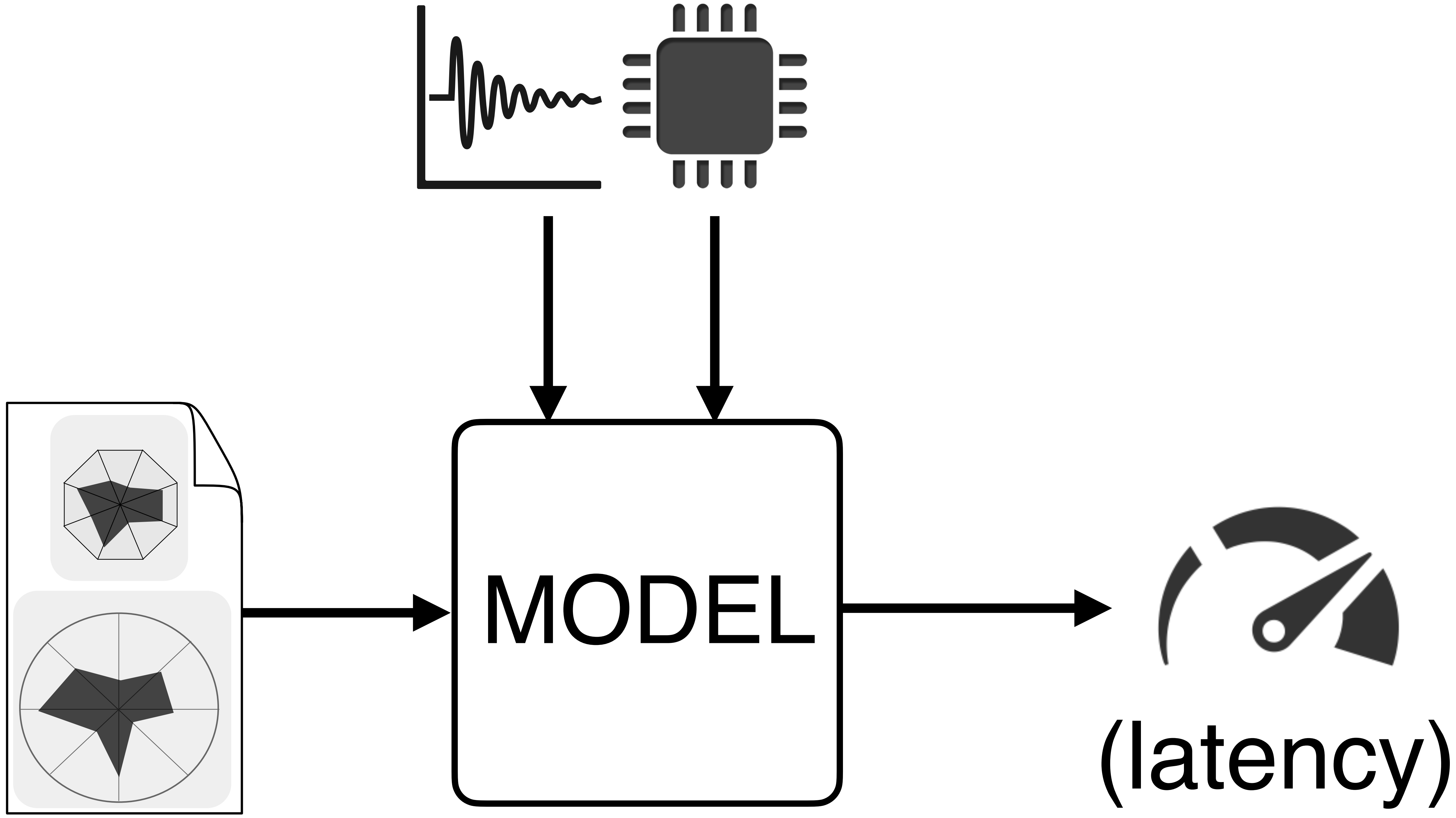


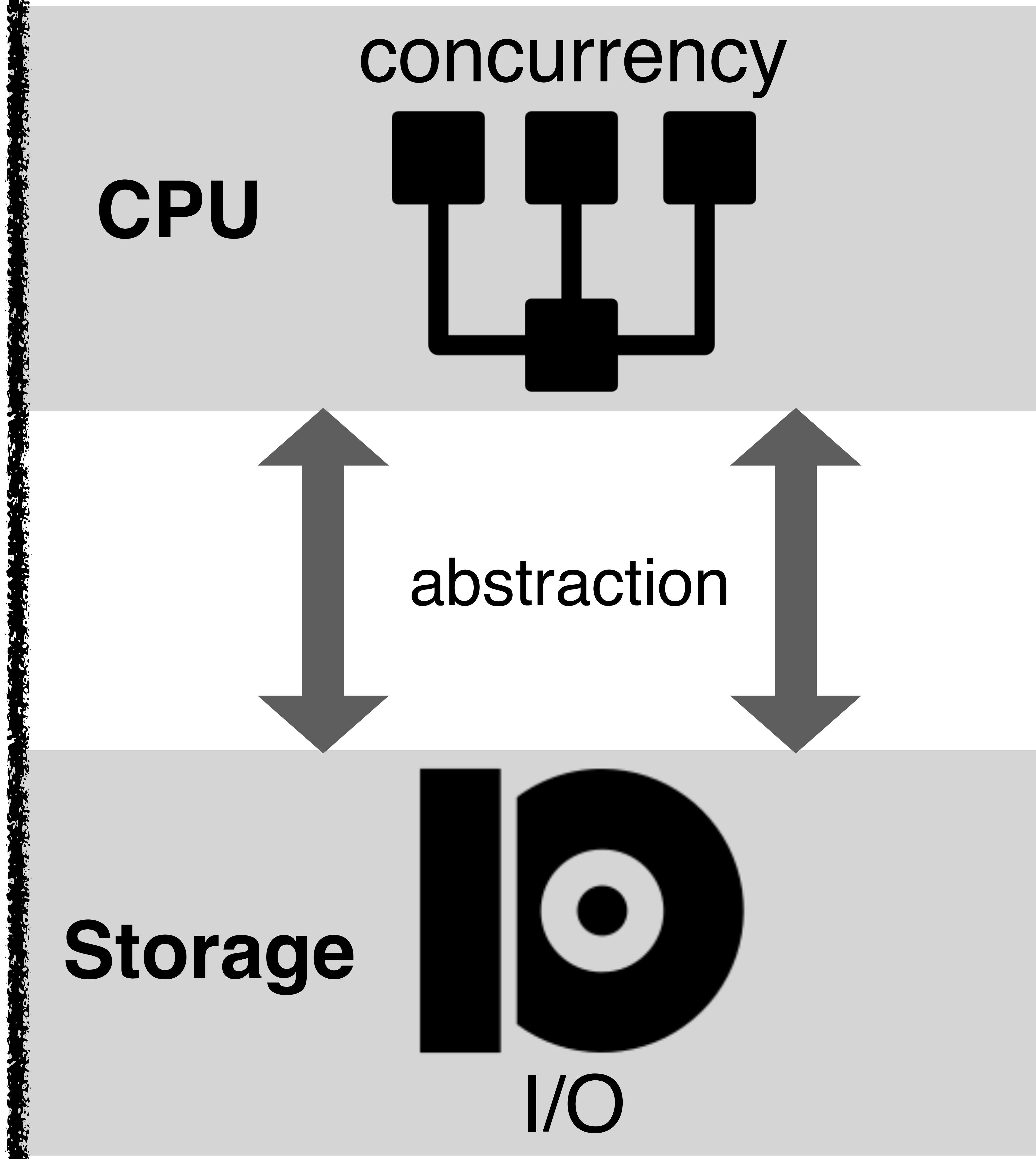
DASlab @ Harvard SEAS

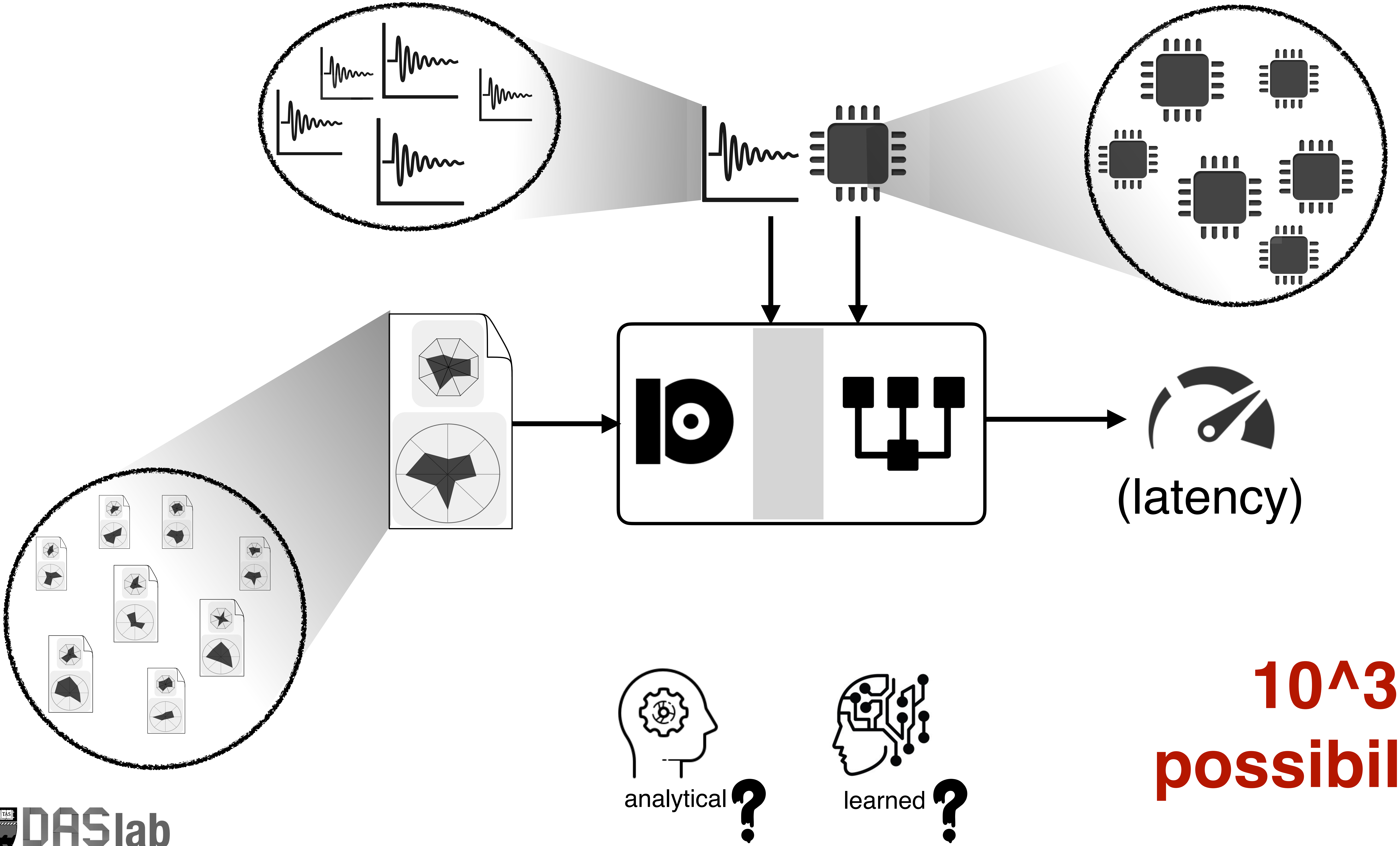

MODEL
(latency)


MODEL
(latency)


I/O
concurrency

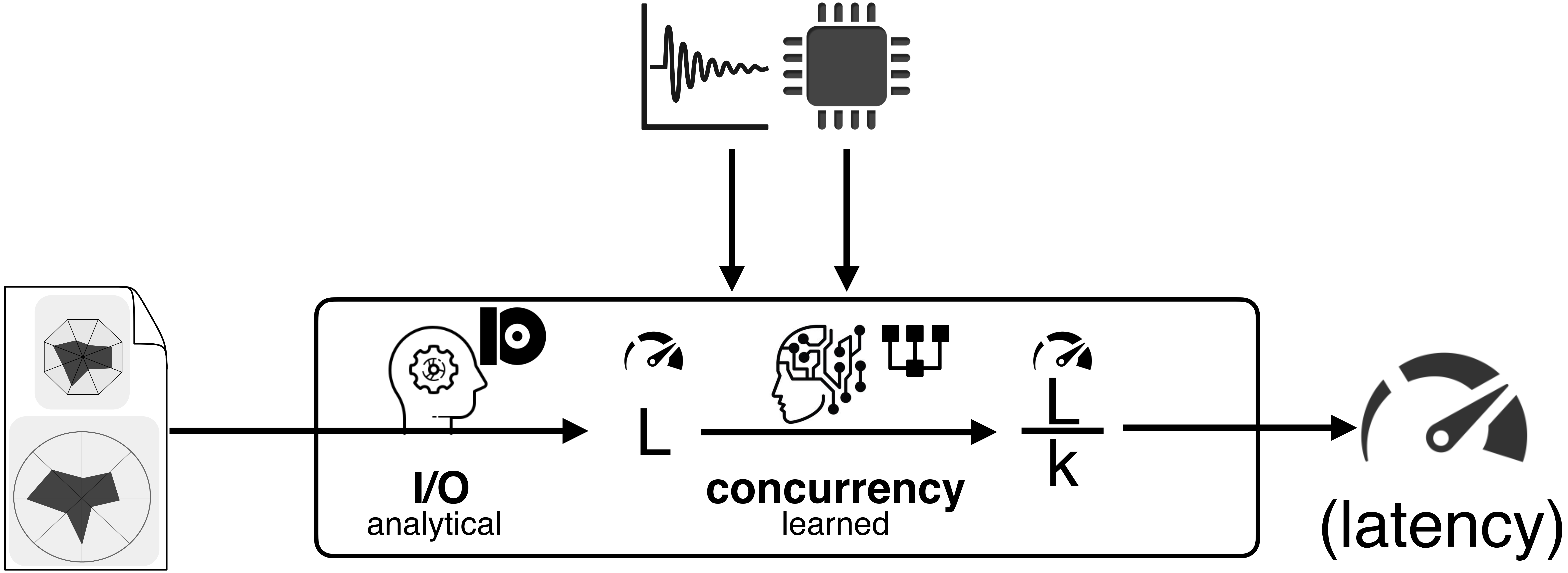

MODEL
(latency)


concurrency
CPU
abstraction
Storage
I/O



10^35 possibilities

DASlab @ Harvard SEAS


I/O
analytical
L
concurrency
learned
L
k
(latency)

1 design morphology

DASlab @ Harvard SEAS

1 design morphology

LAYOUT PRIMITIVES: Design and hardware specification — *initialized by search through engine design space* (rows 1–9)

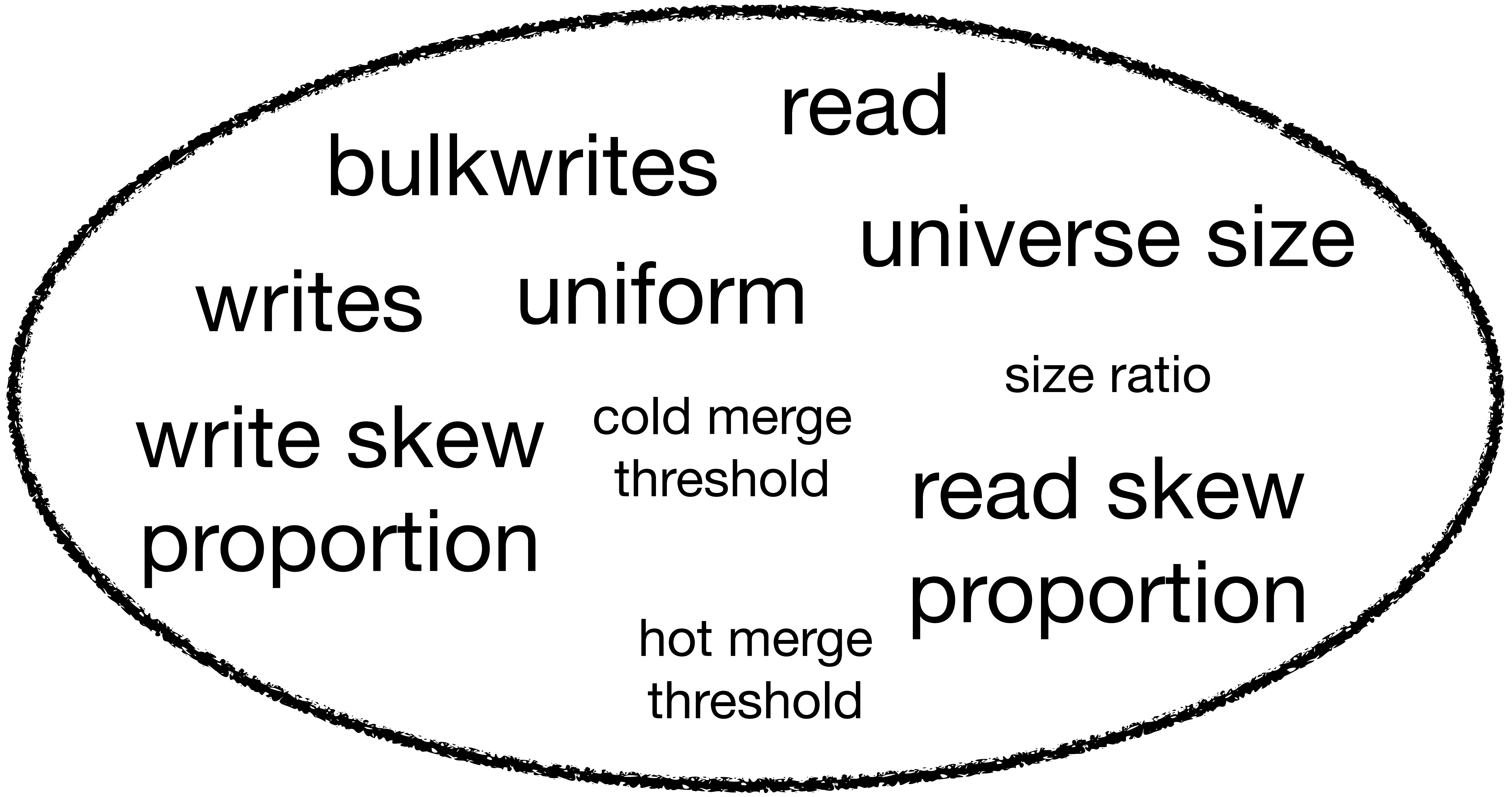
ALGORITHMIC ABSTRACTIONS: Data access, Parallelism — *derived with empirically verified rules* (rows 10–17)

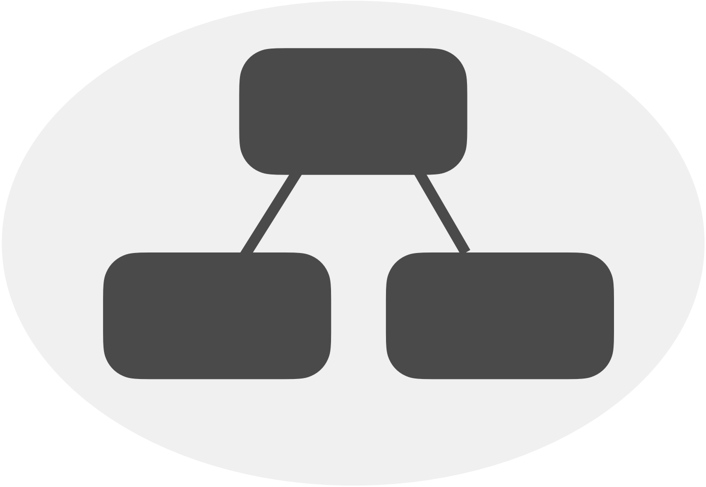
	Design Abstractions of Template	Type/Domain	Example templates for diverse data structures: LSM variants	B-Tree variants	LSH variants	A new design
1.	**Key size:** Denotes the size of keys in the workload.	unsigned int	auto-configured from the sample workload			
2.	**Value size:** Denotes the size of values in the workload. All values are accepted as variable-length strings.	string/slice *max size set to 1 GB*	auto-configured from the sample workload			
3.	**Size ratio (T):** The maximum number of entries in a block (e.g. growth factor in LSM trees or fanout of B-trees.	unsigned integer | function (func)	[2,.. 32]	[32, 64, 128, 256, ..]	[1000, 1001, …] (T is large)	2
4.	**Runs per hot level (K):** At what capacity hot levels are compacted. Rule: should be less than size ratio.	unsigned int	[1.. T]		[T-1]	7
5.	**Runs per cold level (Z):** At what capacity cold levels are compacted. Rule: should be less than size ratio.	unsigned int	[1.. T]	[1]		32
6.	**Logical block size (B):** Number of consecutive disk blocks.	unsigned int		[2048, 4096, …]		
7.	**Buffer capacity** (M_B): Denotes the amount of memory allocated to in-memory buffer/memtables. Configurable w.r.t file size.	64-bit floating point | function (func)	[64 MB, 128 MB, …]	[1 MB, 2 MB, …]	[64 MB, 128 MB, …]	h/w dependent
8.	**Indexes** (M_{FP}): Amount of memory allocated to indexes (fence pointers/hashtables).	64-bit floating point | function (func)	memory to cover L	memory for first level	memory for hash table	h/w dependent
9.	**Bloom filter memory** (M_{BF}): Denotes the bits/entry assigned to Bloom filters.	64-bit float | func(FPR)	10 bits/key			func(FPR)
10.	**Bloom filter design:** Denotes the granularity of Bloom filters, e.g., one Bloom filter instance per block or per file or per run. The default is file.	block | file | run	file			file
11.	**Compaction/Restructuring algorithm:** Full does level-to-level compaction; partial is file-to-file; and hybrid uses both full and partial at separate levels.	partial | full | hybrid	full, partial	partial	partial	hybrid
12.	**Run strategy:** Denotes which run to be picked for compaction (only for partial/ hybrid compaction).	first | last_full | fullest	first, fullest, last_full		first	fullest
13.	**File picking strategy:** Denotes which file to be picked for compaction (for partial/ hybrid compaction). For LSM-trees we set default to dense_fp as it empirically works the best. B-trees pick the first file found to be full. LSH-table restructures at the granularity of runs.	oldest_merged | oldest_flushed | dense_fp | sparse_fp | choose_first	dense_fp	choose_first		dense_fp (hot), choose_first (cold)
14.	**Merge threshold:** If a level is more than x% full, a compaction is triggered.	64-bit floating point	[0.7..1]	0.5		0.75
15.	**Full compaction levels:** Denotes how many levels will have full compaction (only for hybrid compaction). The default is set to 2.	unsigned integer | function (func)	[1..L]			L-Y (from optimal config)
16.	**No. of CPUs:** Number of available cores to use in a VM.	unsigned int		Use all available cores		
17.	**No of threads:** Denotes how many threads are used to process the workload.	unsigned int		Use 1 thread per CPU core		

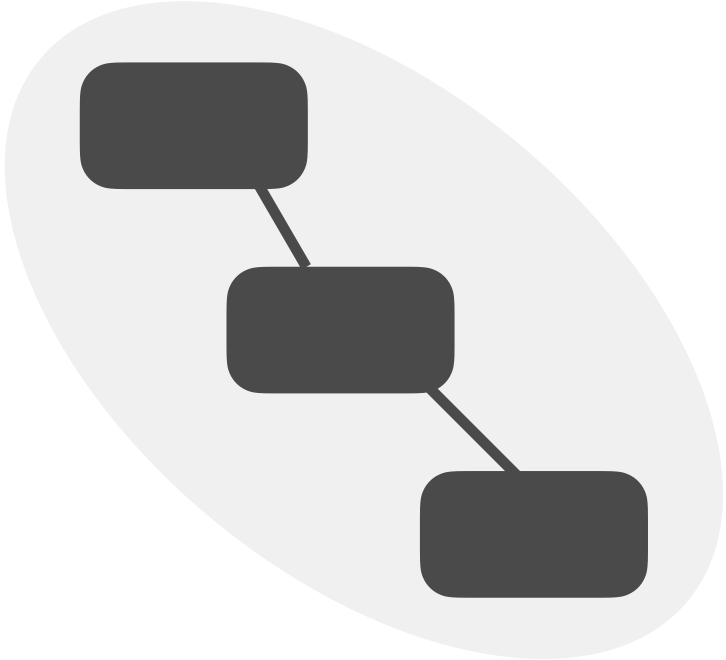
DASlab @ Harvard SEAS

❶ design morphology

Category	Group	Rule	#	Design Abstractions of Template	Type/Domain	Example templates for diverse data structures: LSM variants	B-Tree variants	LSH variants	A new design
LAYOUT PRIMITIVES	Design and hardware specification	*initialized by search through engine design space*	1.	**Key size:** Denotes the size of keys in the workload.	unsigned int	auto-configured from the sample workload			
			2.	**Value size:** Denotes the size of values in the workload. All values are accepted as variable-length strings.	string/slice *max size set to 1 GB*	auto-configured from the sample workload			
			3.	**Size ratio (T):** The maximum number of entries in a block (e.g. growth factor in LSM trees or fanout of B-trees.	unsigned integer | function (func)	[2,.. 32]	[32, 64, 128, 256, ..]	[1000, 1001, …] (T is large)	2
			4.	**Runs per hot level (K):** At what capacity hot levels are compacted. Rule: should be less than size ratio.	unsigned int	[1.. T]		[T-1]	7
			5.	**Runs per cold level (Z):** At what capacity cold levels are compacted. Rule: should be less than size ratio.	unsigned int	[1.. T]	[1]		32
			6.	**Logical block size (B):** Number of consecutive disk blocks.	unsigned int	[2048, 4096, …]			
			7.	**Buffer capacity** (M_B)**:** Denotes the amount of memory allocated to in-memory buffer/memtables. Configurable w.r.t file size.	64-bit floating point | function (func)	[64 MB, 128 MB, …]	[1 MB, 2 MB, …]	[64 MB, 128 MB, …]	h/w dependent
			8.	**Indexes** (M_{FP})**:** Amount of memory allocated to indexes (fence pointers/hashtables).	64-bit floating point | function (func)	memory to cover L	memory for first level	memory for hash table	h/w dependent
			9.	**Bloom filter memory** (M_{BF})**:** Denotes the bits/entry assigned to Bloom filters.	64-bit float | func(FPR)	10 bits/key			func(FPR)
ALGORITHMIC ABSTRACTIONS	Data access	*derived with empirically verified rules*	10.	**Bloom filter design:** Denotes the granularity of Bloom filters, e.g., one Bloom filter instance per block or per file or per run. The default is file.	block | file | run	file			file
			11.	**Compaction/Restructuring algorithm:** Full does level-to-level compaction; partial is file-to-file; and hybrid uses both full and partial at separate levels.	partial | full | hybrid	full, partial	partial	partial	hybrid
			12.	**Run strategy:** Denotes which run to be picked for compaction (only for partial/ hybrid compaction).	first | last_full | fullest	first, fullest, last_full		first	fullest
			13.	**File picking strategy:** Denotes which file to be picked for compaction (for partial/ hybrid compaction). For LSM-trees we set default to dense_fp as it empirically works the best. B-trees pick the first file found to be full. LSH-table restructures at the granularity of runs.	oldest_merged | oldest_flushed | dense_fp | sparse_fp | choose_first	dense_fp	choose_first		dense_fp (hot), choose_first (cold)
	Parallelism		14.	**Merge threshold:** If a level is more than x% full, a compaction is triggered.	64-bit floating point	[0.7..1]	0.5		0.75
			15.	**Full compaction levels:** Denotes how many levels will have full compaction (only for hybrid compaction). The default is set to 2.	unsigned integer | function (func)	[1..L]			L-Y (from optimal config)
			16.	**No. of CPUs:** Number of available cores to use in a VM.	unsigned int	Use all available cores			
			17.	**No of threads:** Denotes how many threads are used to process the workload.	unsigned int	Use 1 thread per CPU core			





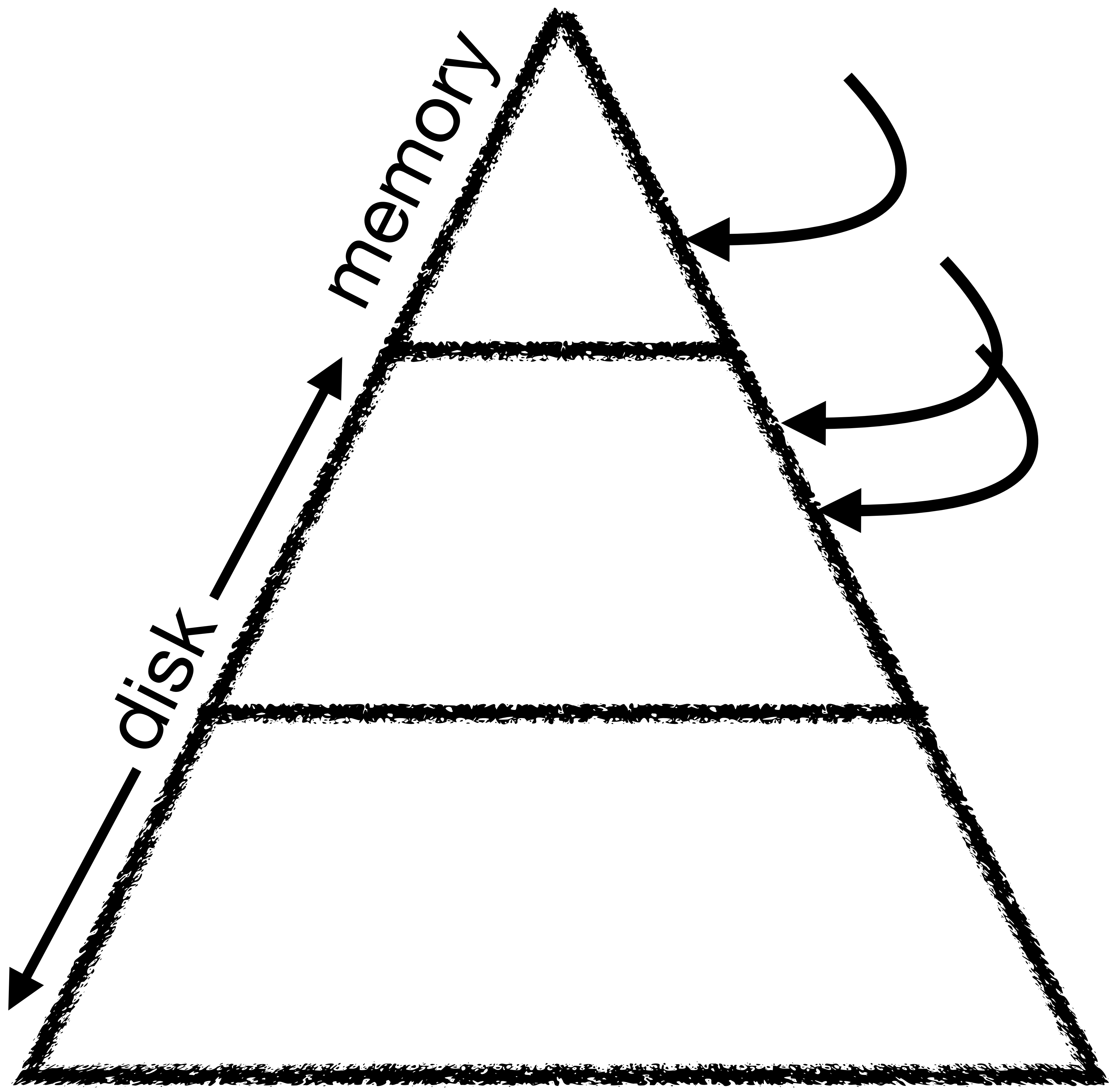


DASlab @ Harvard SEAS

1 design morphology

2 per operation I/O cost

❶ design morphology ❷ per operation I/O cost



early-stopping

infrequent merging

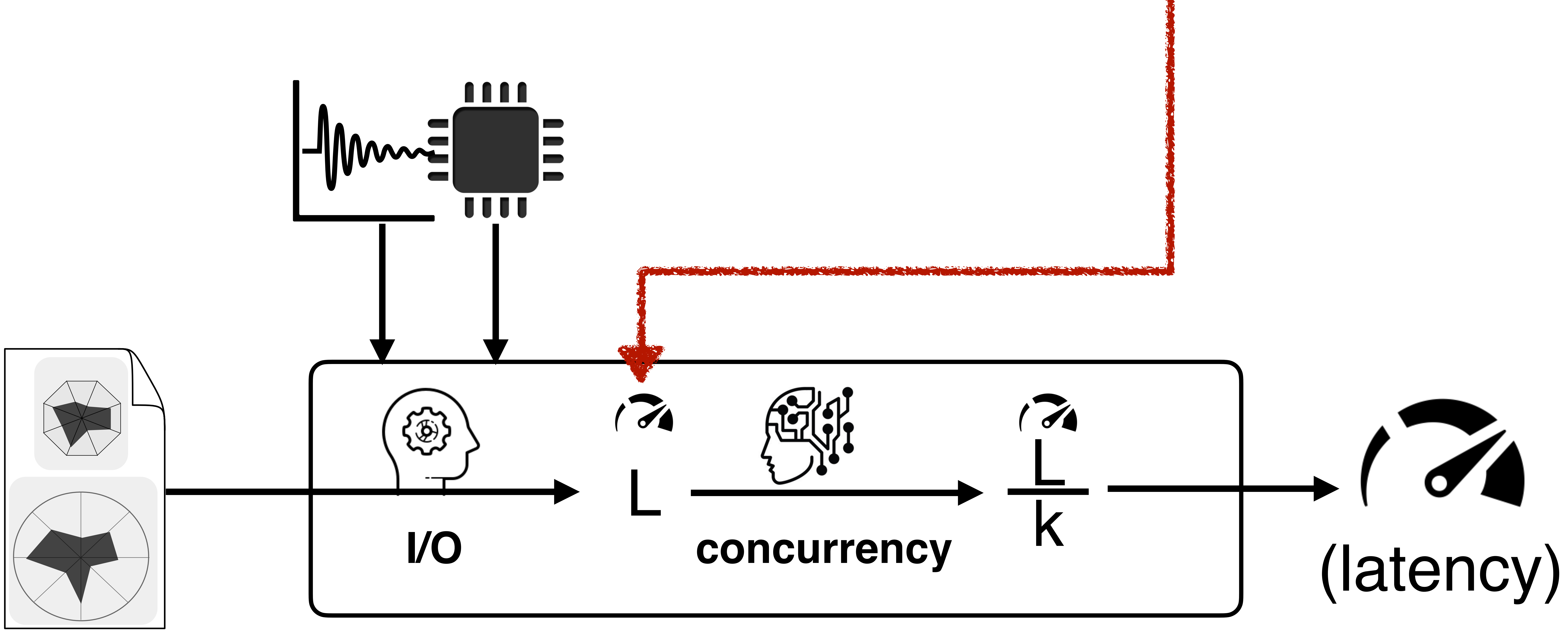
DASlab @ Harvard SEAS

❶ design morphology ❷ per operation I/O cost ❸ total cost

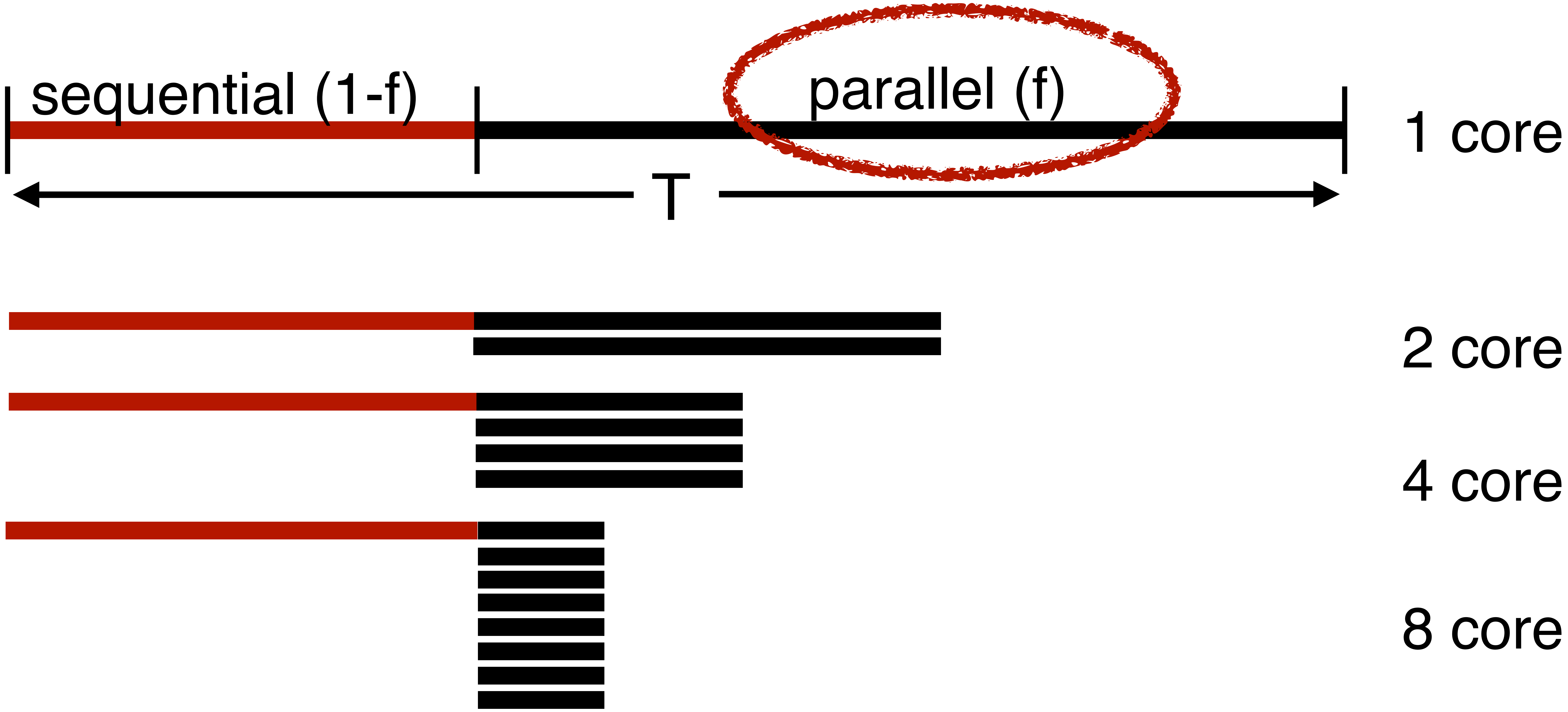
$$\text{latency} = \frac{\text{Total I/O}_{\text{workload}}}{\text{IOPS}} = L$$

❶ design morphology ❷ per operation I/O cost ❸ total cost

$$\text{latency} = \frac{\text{Total I/O}_{\text{workload}}}{\text{IOPS}} = L$$



Amdahl's Law (1967)



theoretical speedup $k = \dfrac{T}{T - fT + fT/n} = \dfrac{1}{1 - f(1 - 1/n)}$

DASlab @ Harvard SEAS

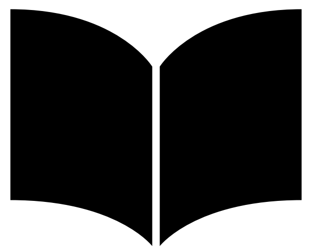
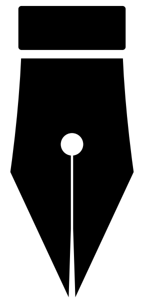
Design class				
LSM	Hybrid-1	Hybrid-2	B-tree	LSH
r1	r2	r3	r4	r5
w1	w2	w3	w4	w5

H/W 1: D16s v3 (16 vcpus, 64 GB RAM), 25600 IOPS, premium SSD
H/W 2: D8s v3 (8 vcpus, 32 GB RAM), 12800 IOPS, premium SSD
H/W 3: D16a v4 (16 vcpus, 64 GB RAM), 32x500 IOPS, non-premium SSD

(Base data, #reads)	H/W 1	H/W 2	H/W 3
10M, 1M	0.965	0.953	0.95
50M, 5M	0.94	0.94	0.92
100M, 10M	0.942	0.952	0.923

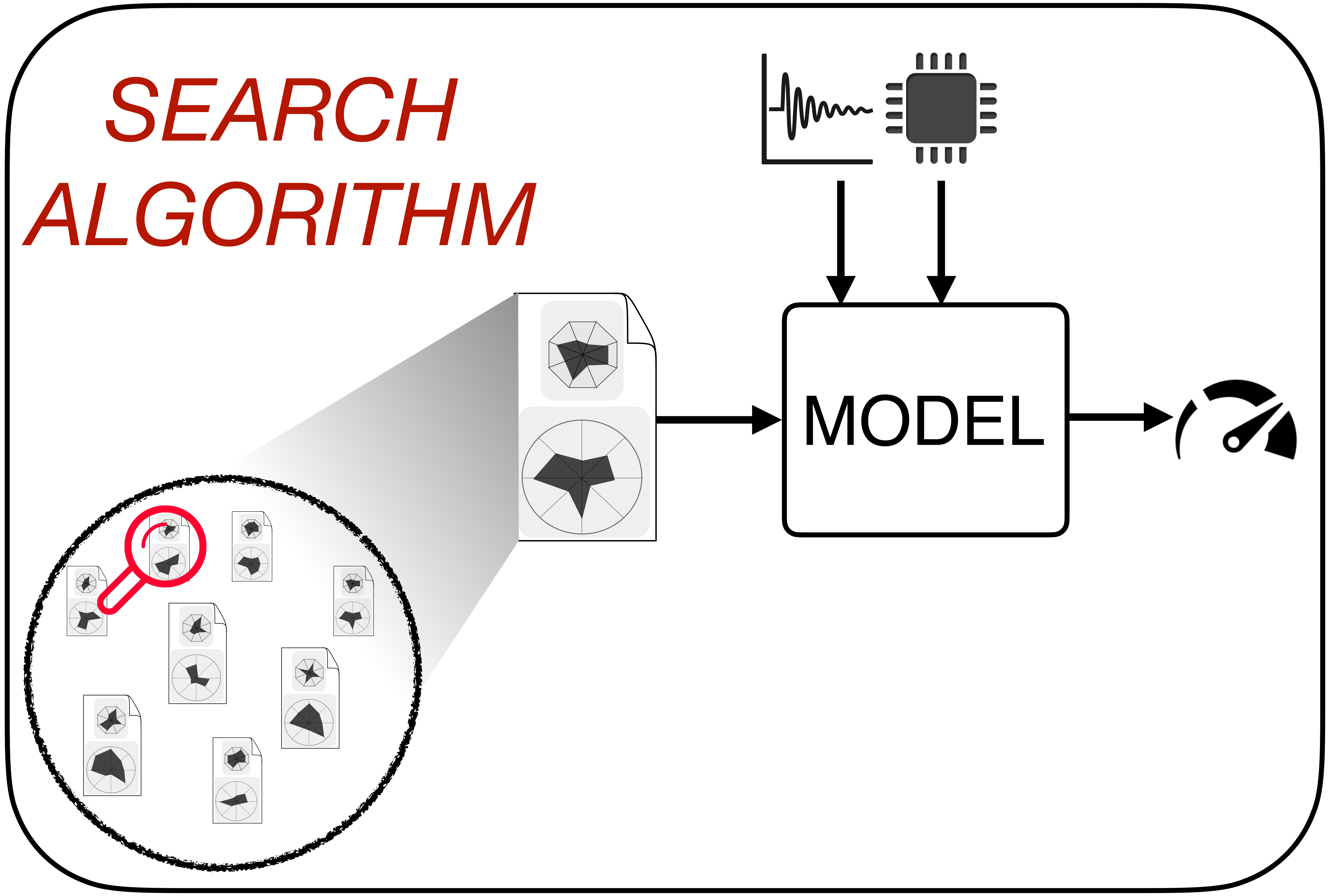
Ops	Design class				
	LSM	Hybrid-1	Hybrid-2	B-tree	LSH
	r1	r2	r3	r4	r5
	w1	w2	w3	w4	w5

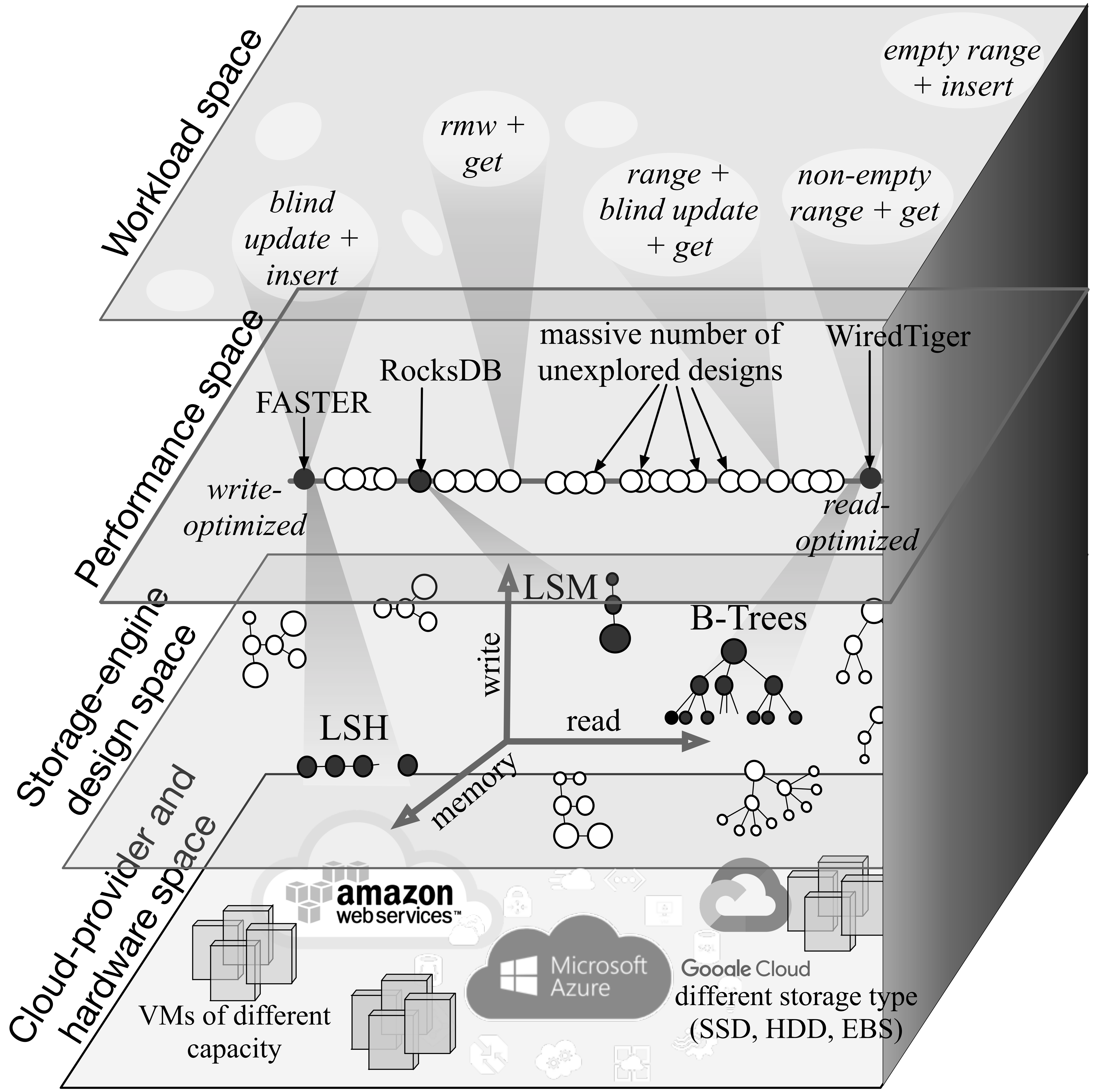
DASlab @ Harvard SEAS

Ops	Design class				
	LSM	Hybrid-1	Hybrid-2	B-tree	LSH
	0.91	0.93	0.92	0.94	0.97
	0.71	0.79	0.74	0.65	0.8

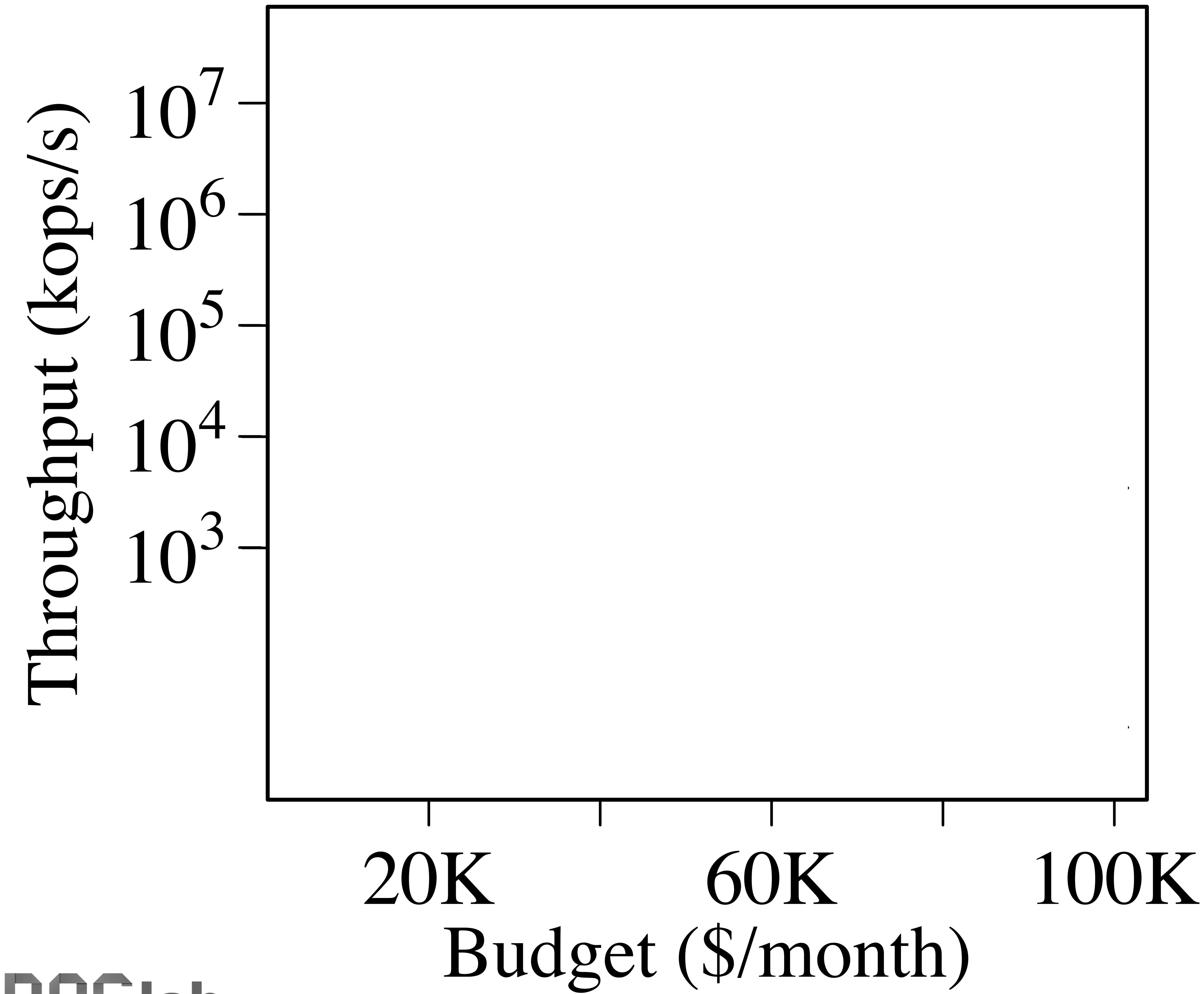
20% reads +
80% writes

proportion of parallelizable code = 0.2*0.91 + 0.8*0.71 = 0.75

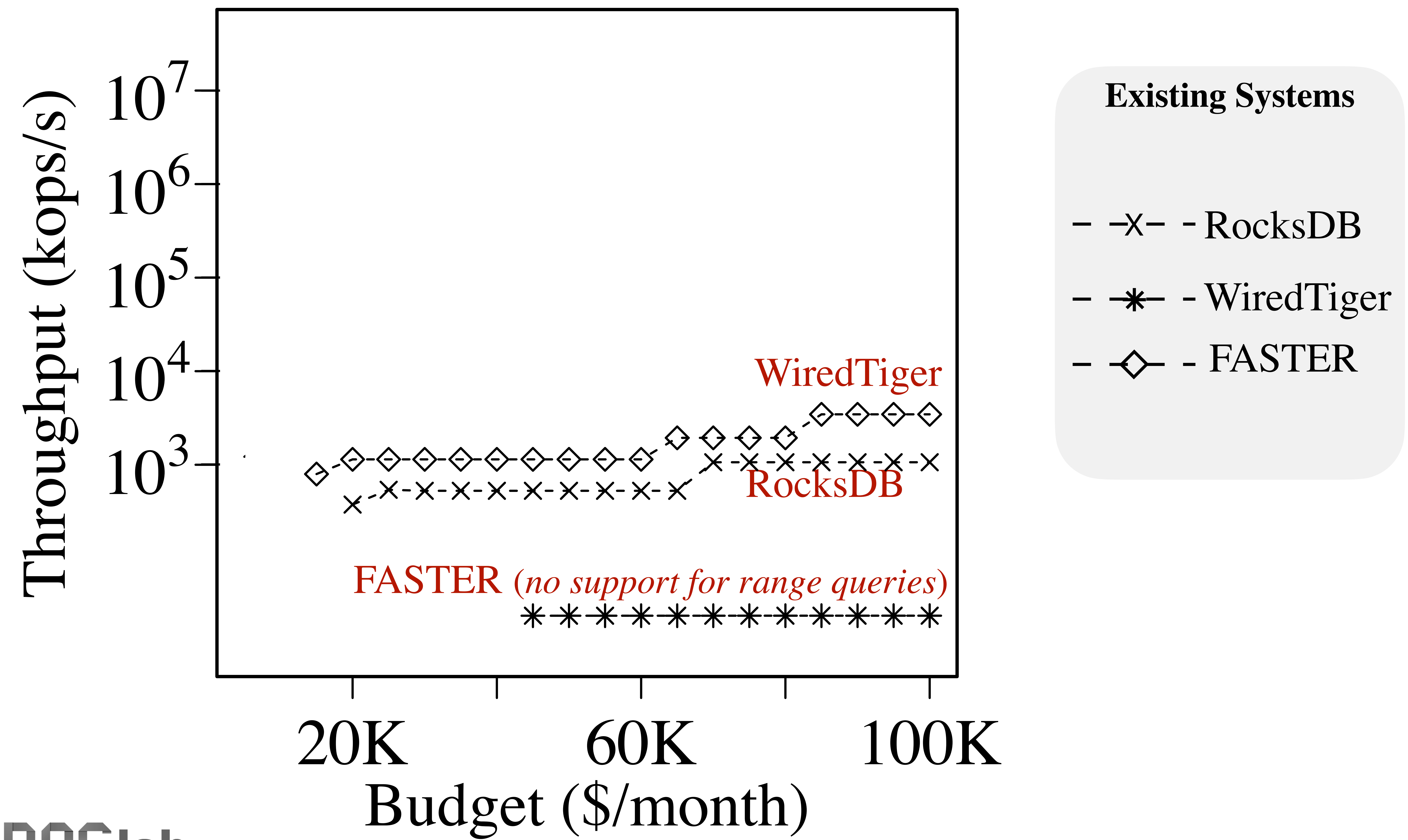

SEARCH
ALGORITHM
MODEL


Workload space
empty range + insert
rmw + get
blind update + insert
range + blind update + get
non-empty range + get
Performance space
massive number of unexplored designs
WiredTiger
RocksDB
FASTER
write-optimized
read-optimized
Storage-engine design space
LSM
B-Trees
write
read
memory
LSH
Cloud-provider and hardware space
amazon web services
Microsoft Azure
Google Cloud
VMs of different capacity
different storage type (SSD, HDD, EBS)

YCSB E variant (30% blind update, 20% non-empty range, 50% empty range)

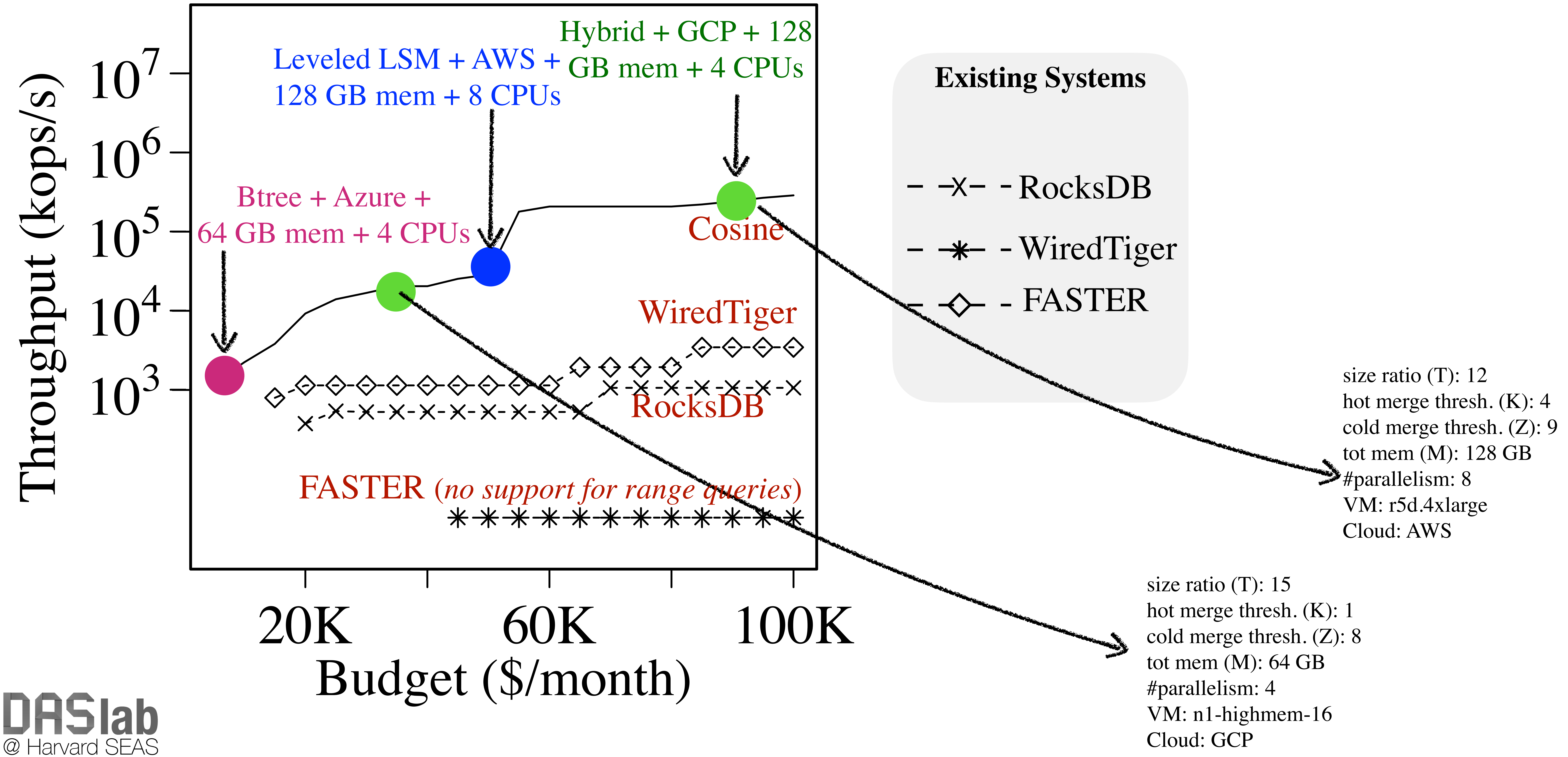


YCSB E variant (30% blind update, 20% non-empty range, 50% empty range)



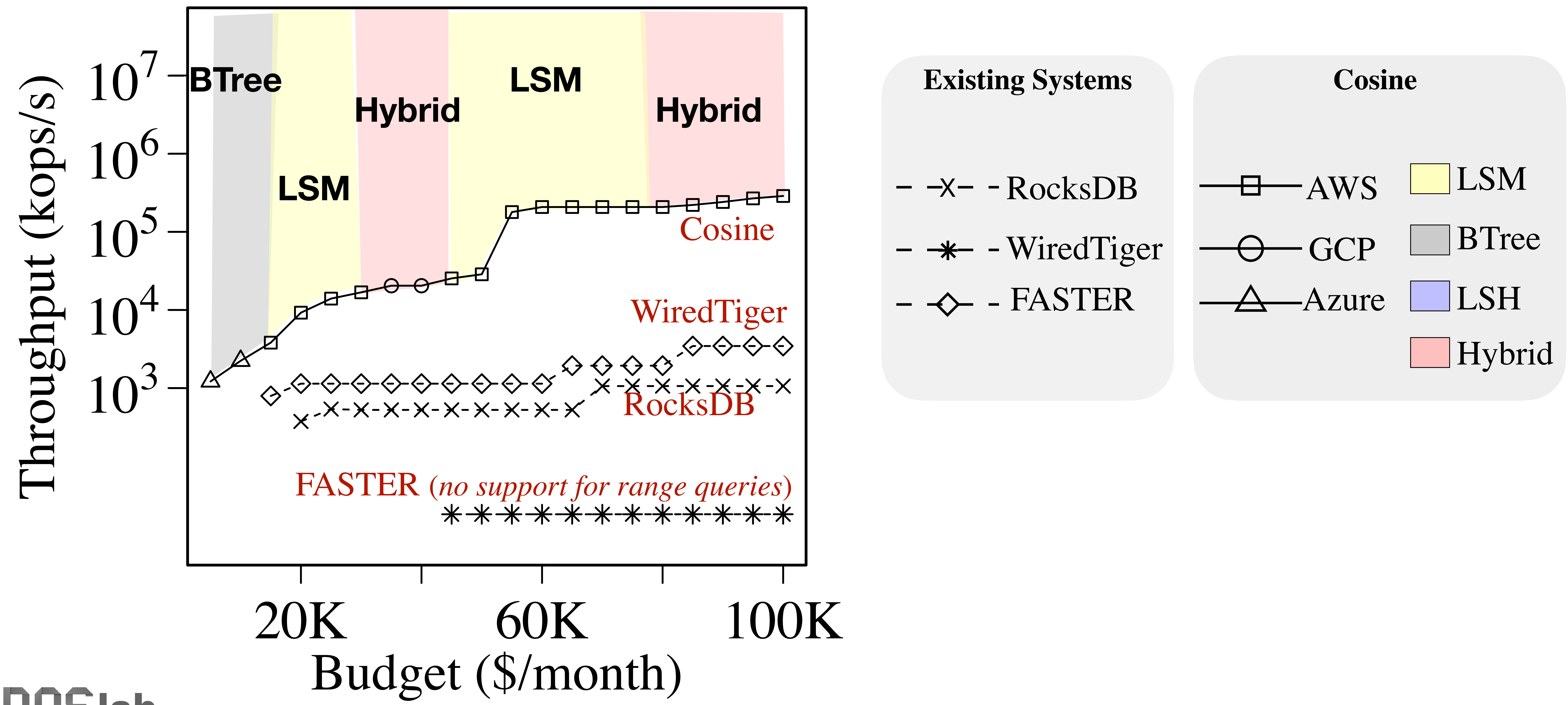
DASlab @ Harvard SEAS

YCSB E variant (30% blind update, 20% non-empty range, 50% empty range)



DASlab @ Harvard SEAS

YCSB E variant (30% blind update, 20% non-empty range, 50% empty range)



DASlab @ Harvard SEAS

DASlab @ Harvard SEAS

Self-Designing Systems

- robustness
- extensibility
- data types and designs
- complex operations
- IO and concurrency
- SLA and cloud
- adaptability
- workload forecasting

THANK YOU!

DASlab @ Harvard SEAS