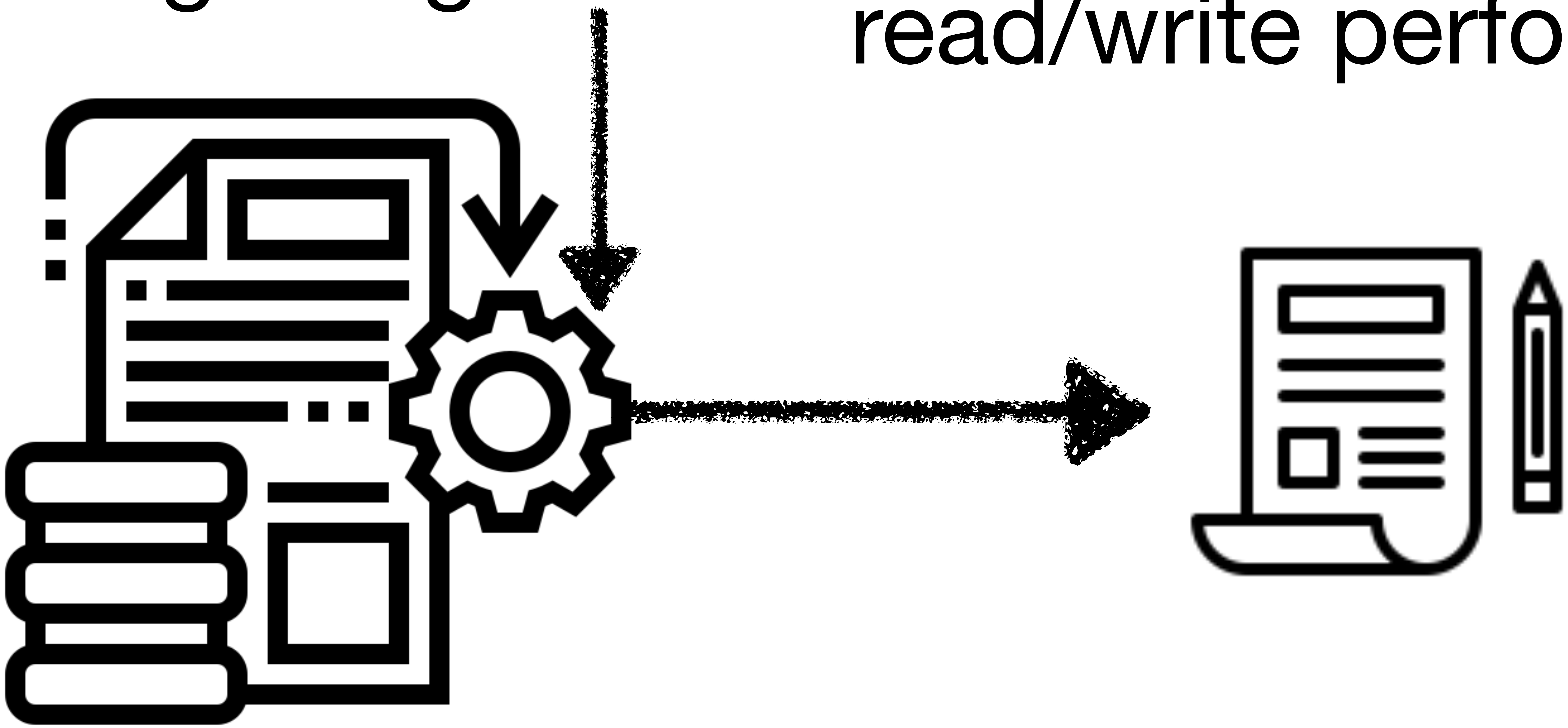
Cosine: A Cloud-Cost Optimized Self-Designing Key-Value Storage Engine

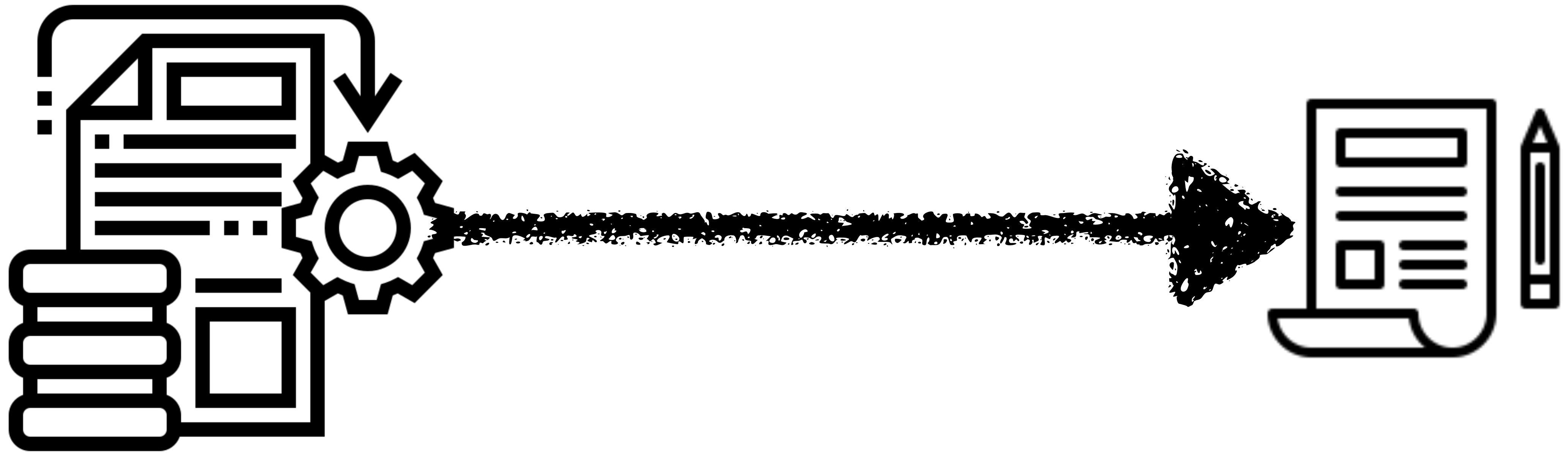
Subarna Chatterjee, Meena Jagadeesan,
Wilson Qin, Stratos Idreos

Data Systems Laboratory (DASLab)
Harvard University

DASlab @ Harvard SEAS

PVLDB 2022

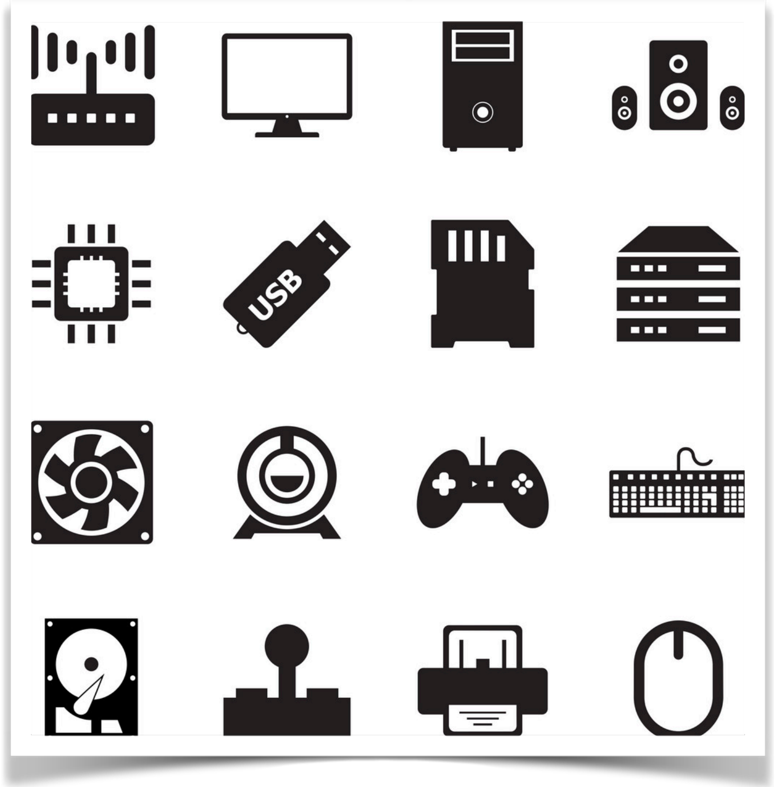

storage-engines
read/write performance
Data-systems



CONTEXT

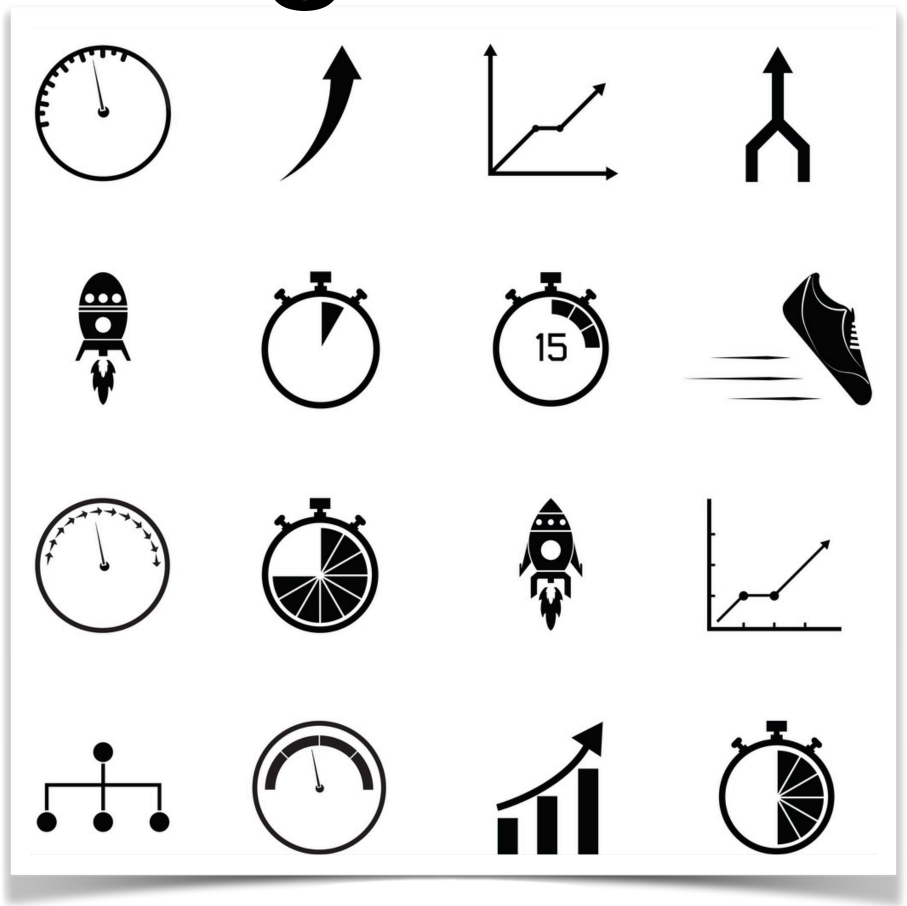
data

hardware



applications

performance goals



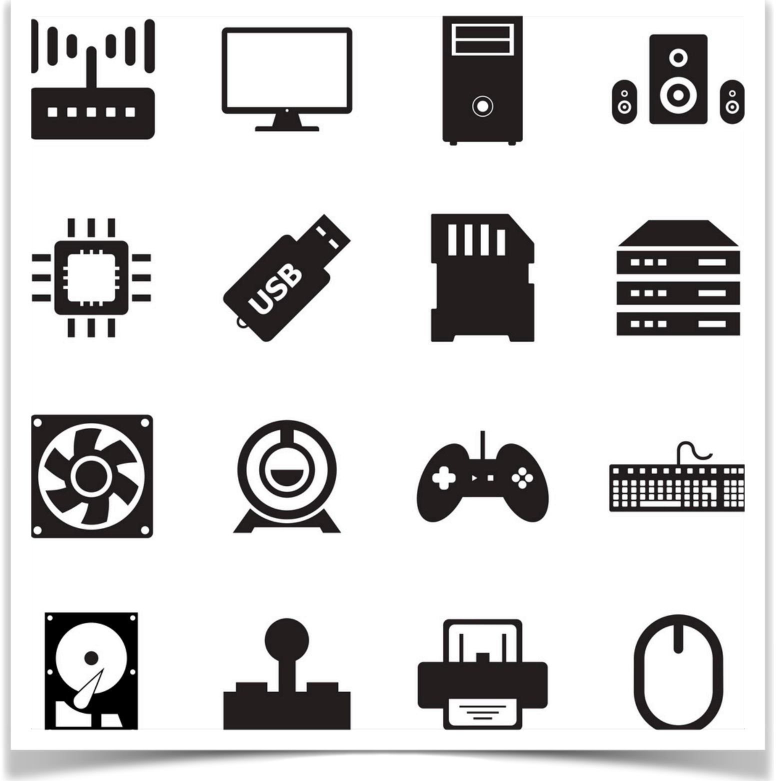
DASlab @ Harvard SEAS

The **CONTEXT** keeps changing …

The **CONTEXT** keeps changing …

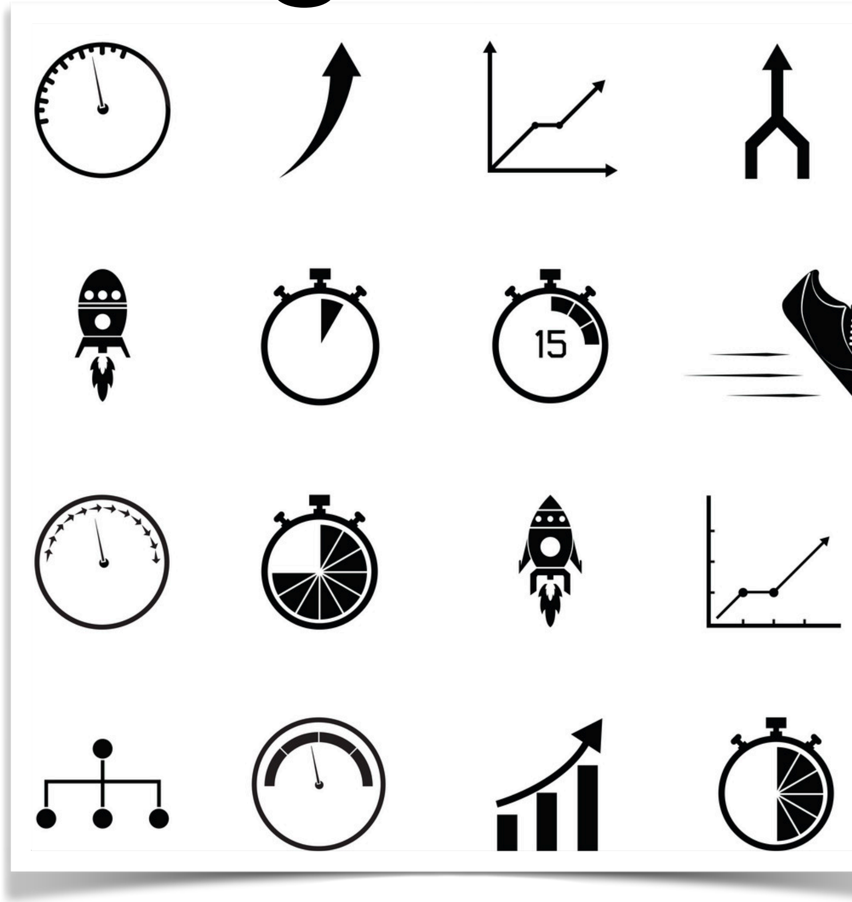
data

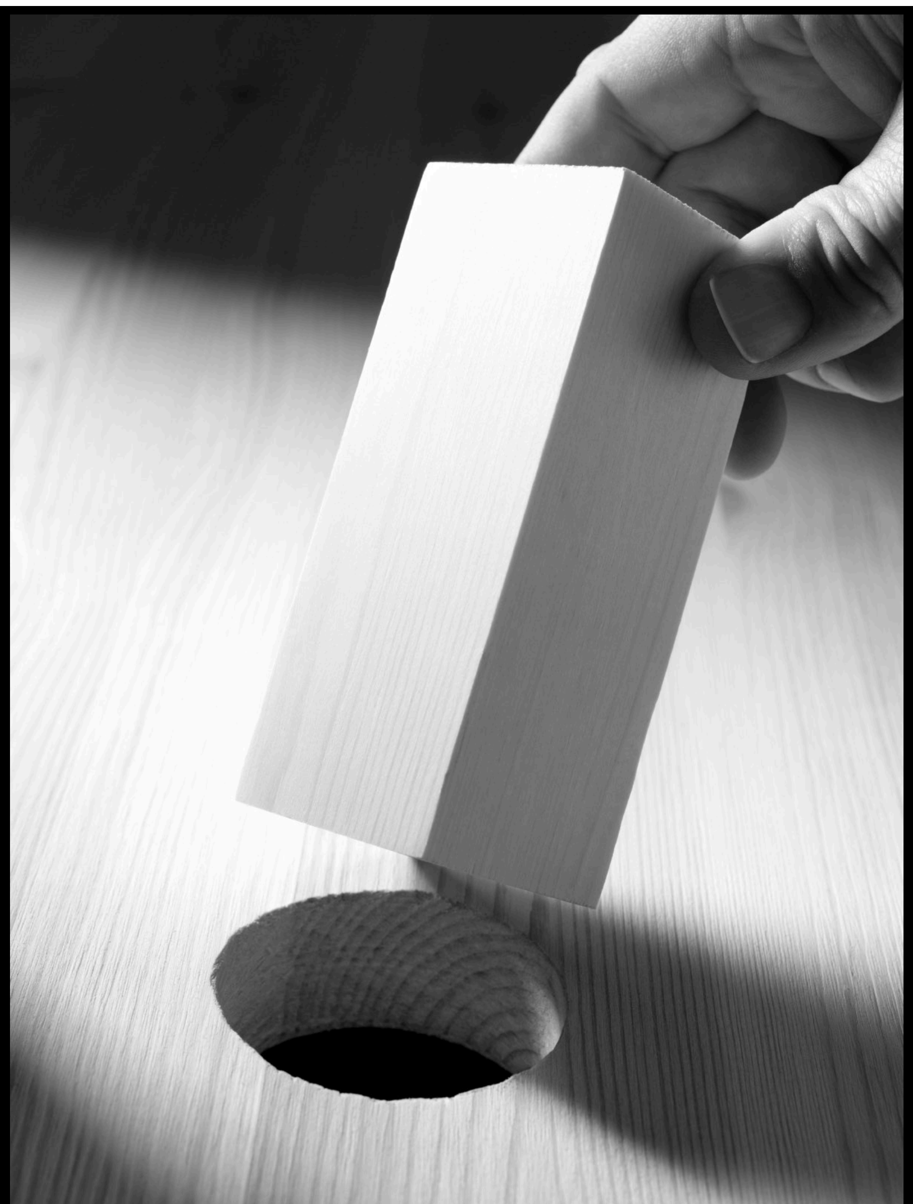
hardware



applications

performance goals

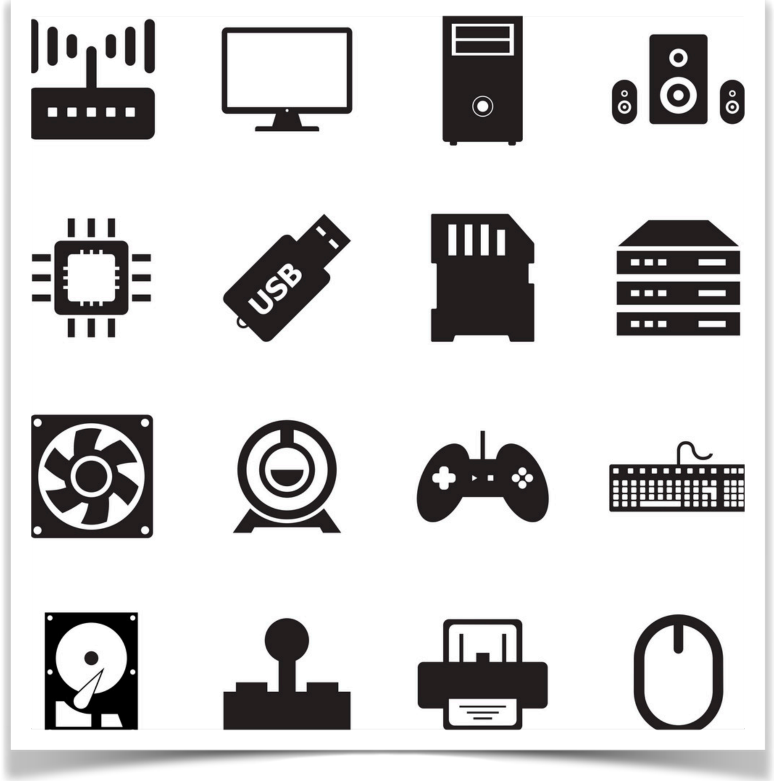




The **CONTEXT** keeps changing …

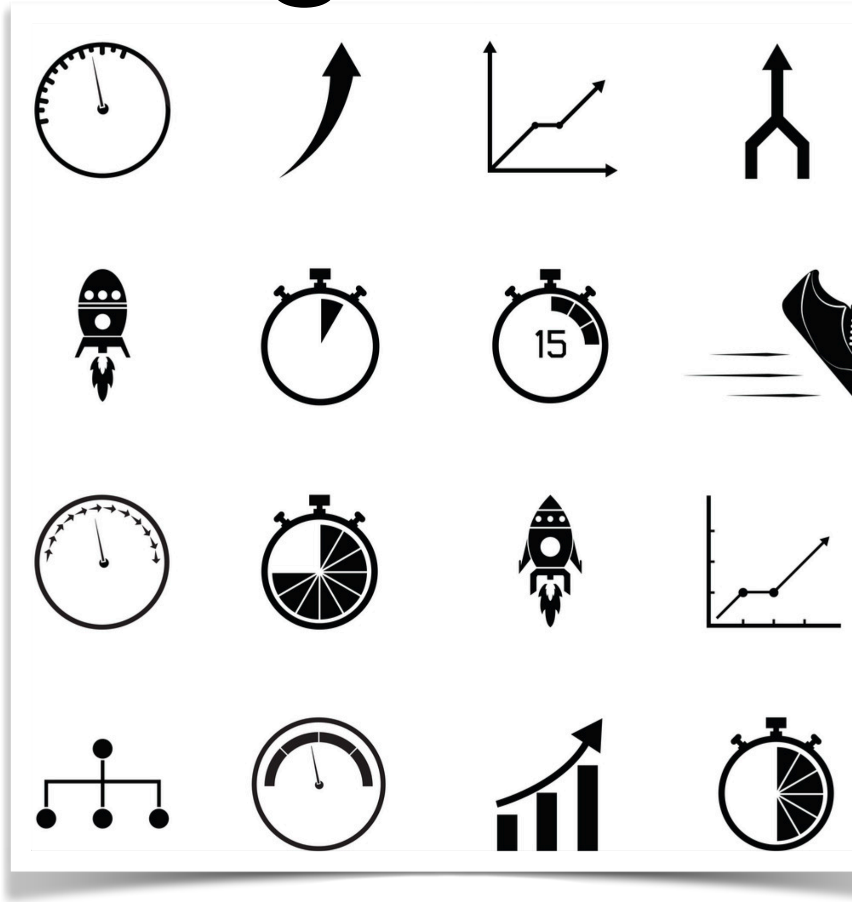
data

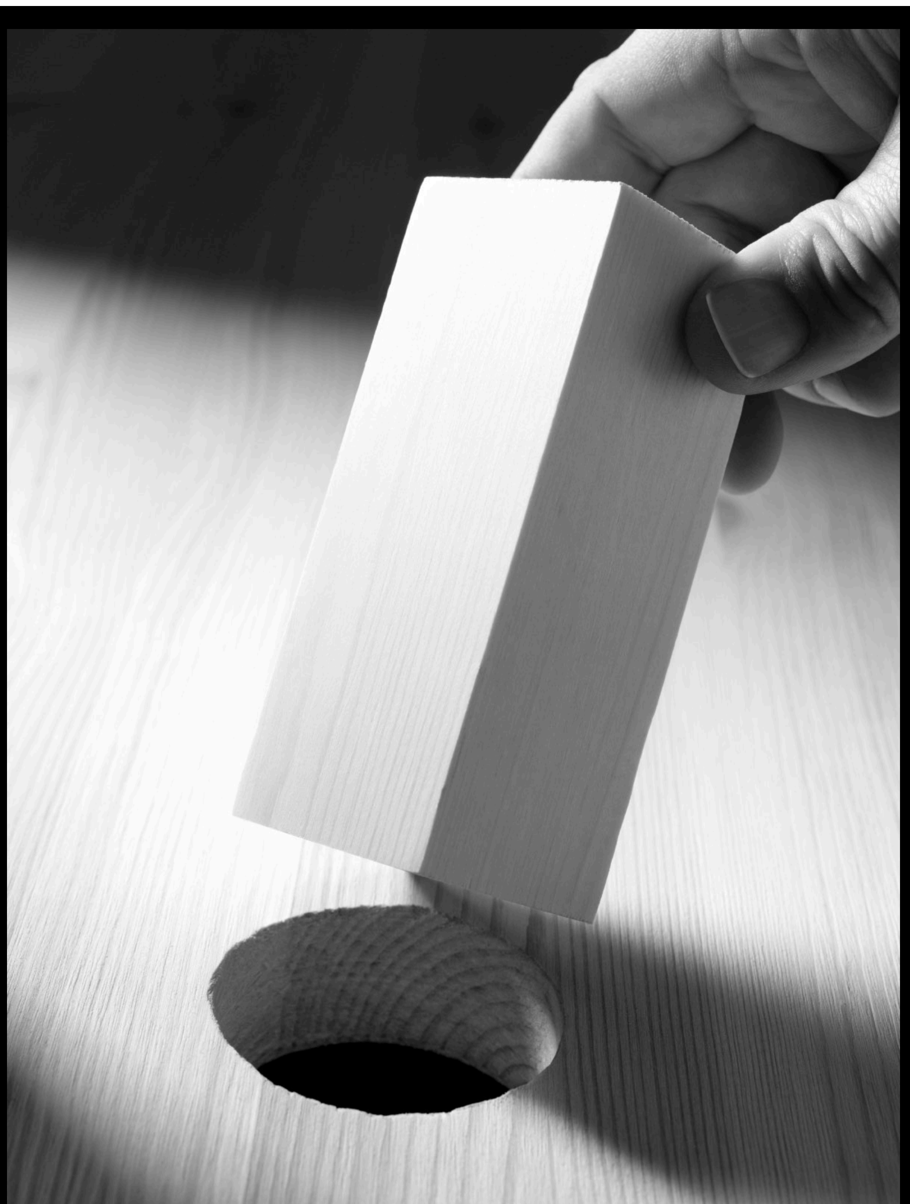
hardware



applications

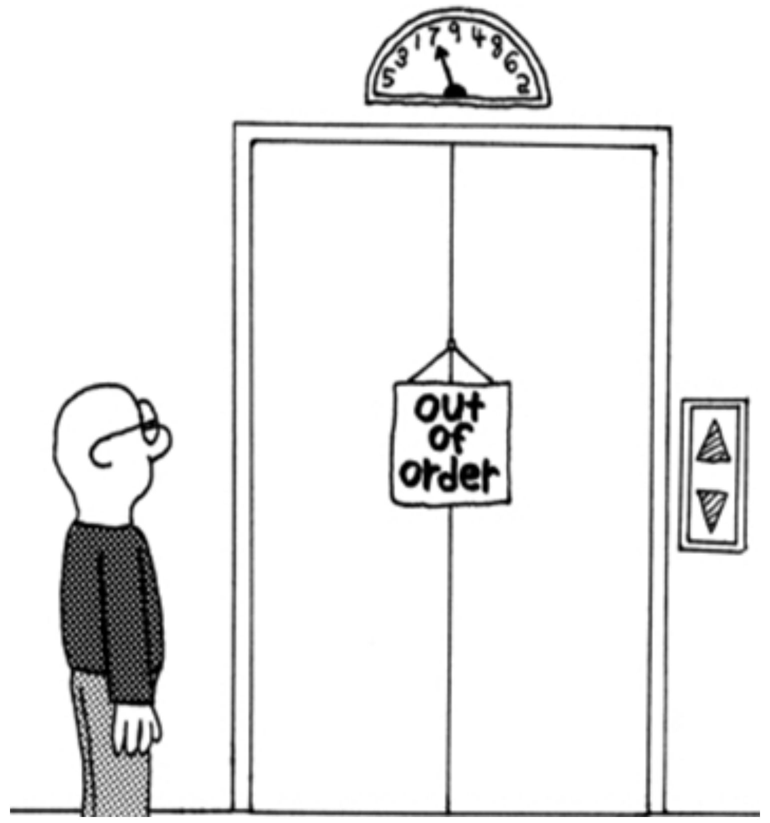
performance goals



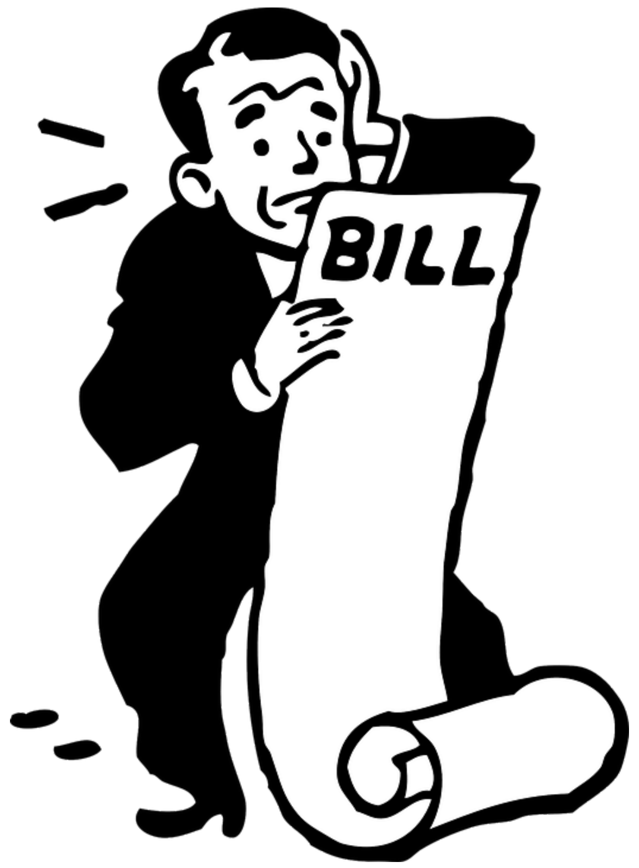


Bottleneck: Sub-Optimal Data Systems

Bottleneck: Sub-Optimal Data Systems



PERFORMANCE



COST

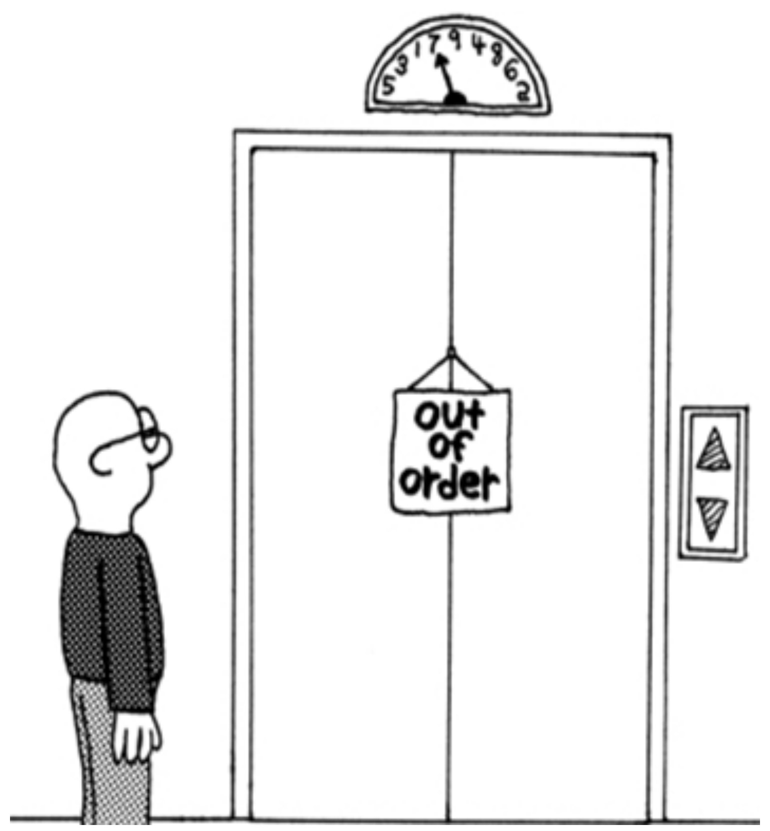
Ryan Booth @that1guy_15 · Mar 4
Oh fun! Another **high cloud bill**. This is exactly how I wanted to spend the rest of my week...

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

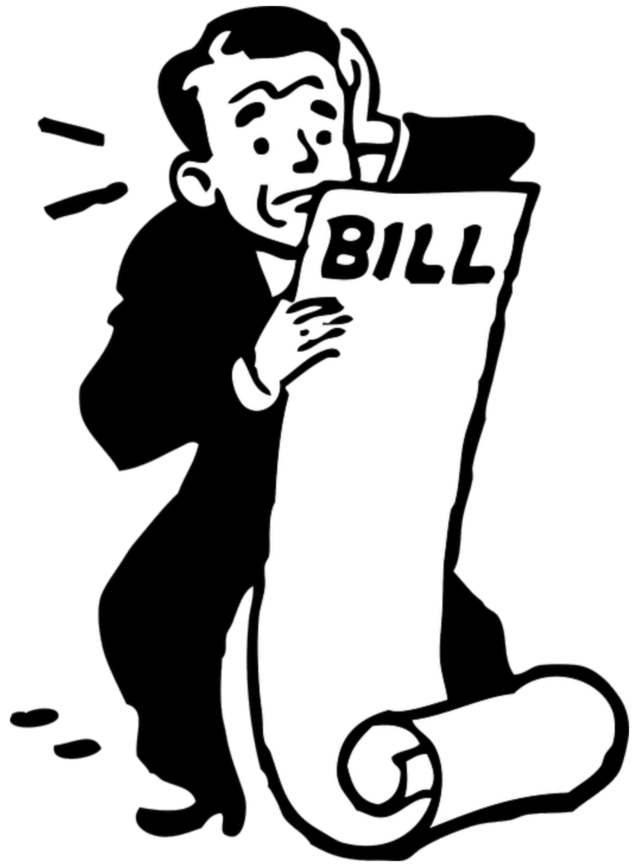
* for a tremendous amount of workloads

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

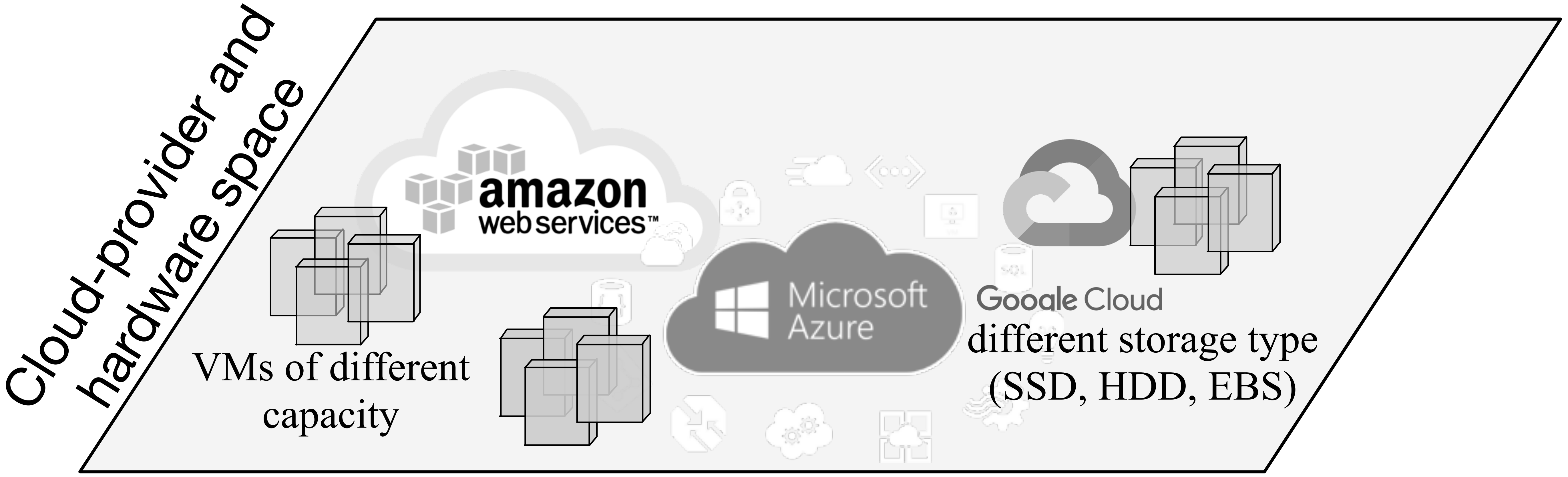
Translucent Computing @translucentcomp · Jan 20
#Kubernetes **cloud** costs are getting out of hand. Working with many clients and different **cloud** service providers, in more than 70% of cases, we see the VM cluster nodes are underutilized, which leads to a **high cloud bill**.

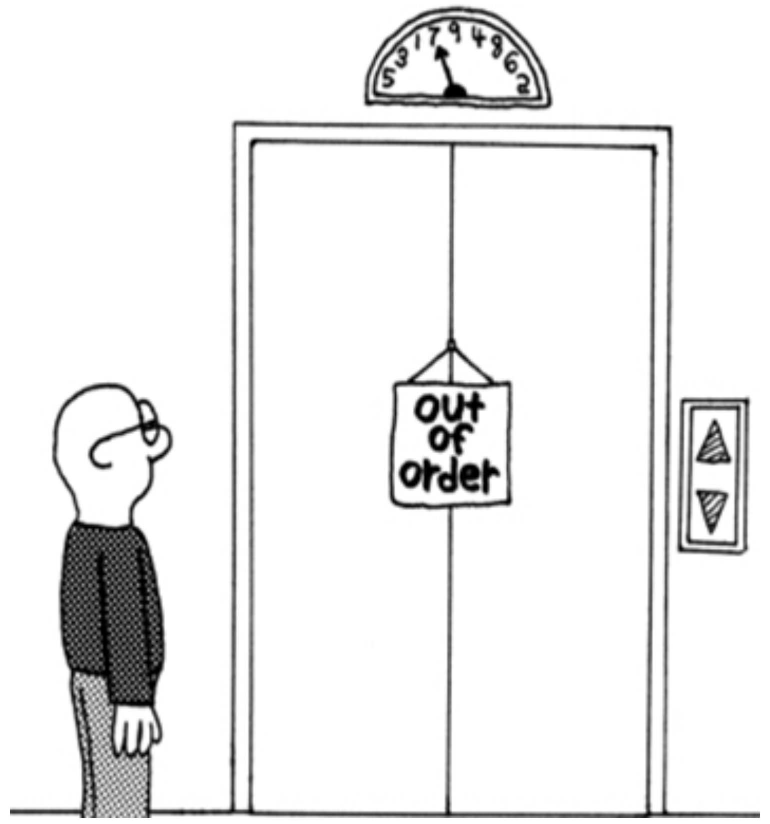


PERFORMANCE

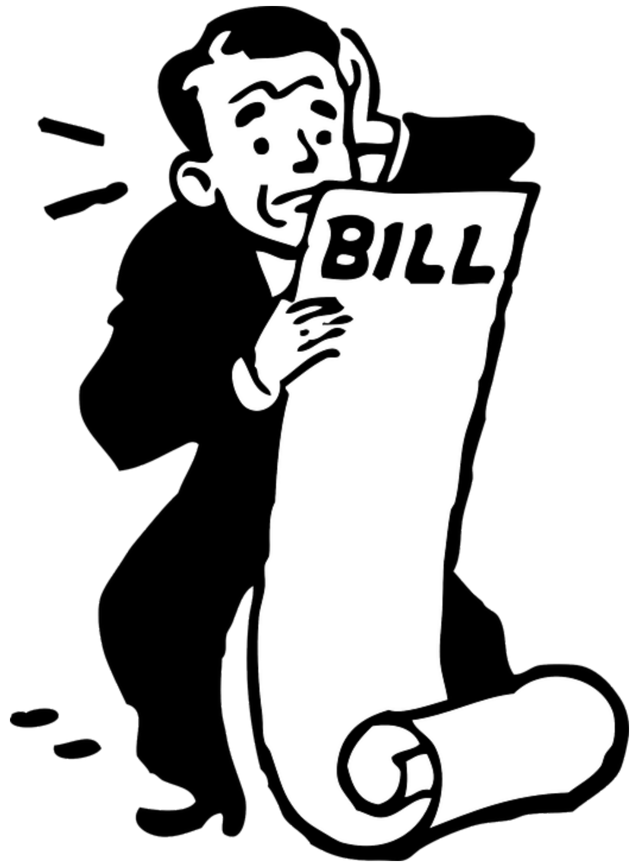


COST

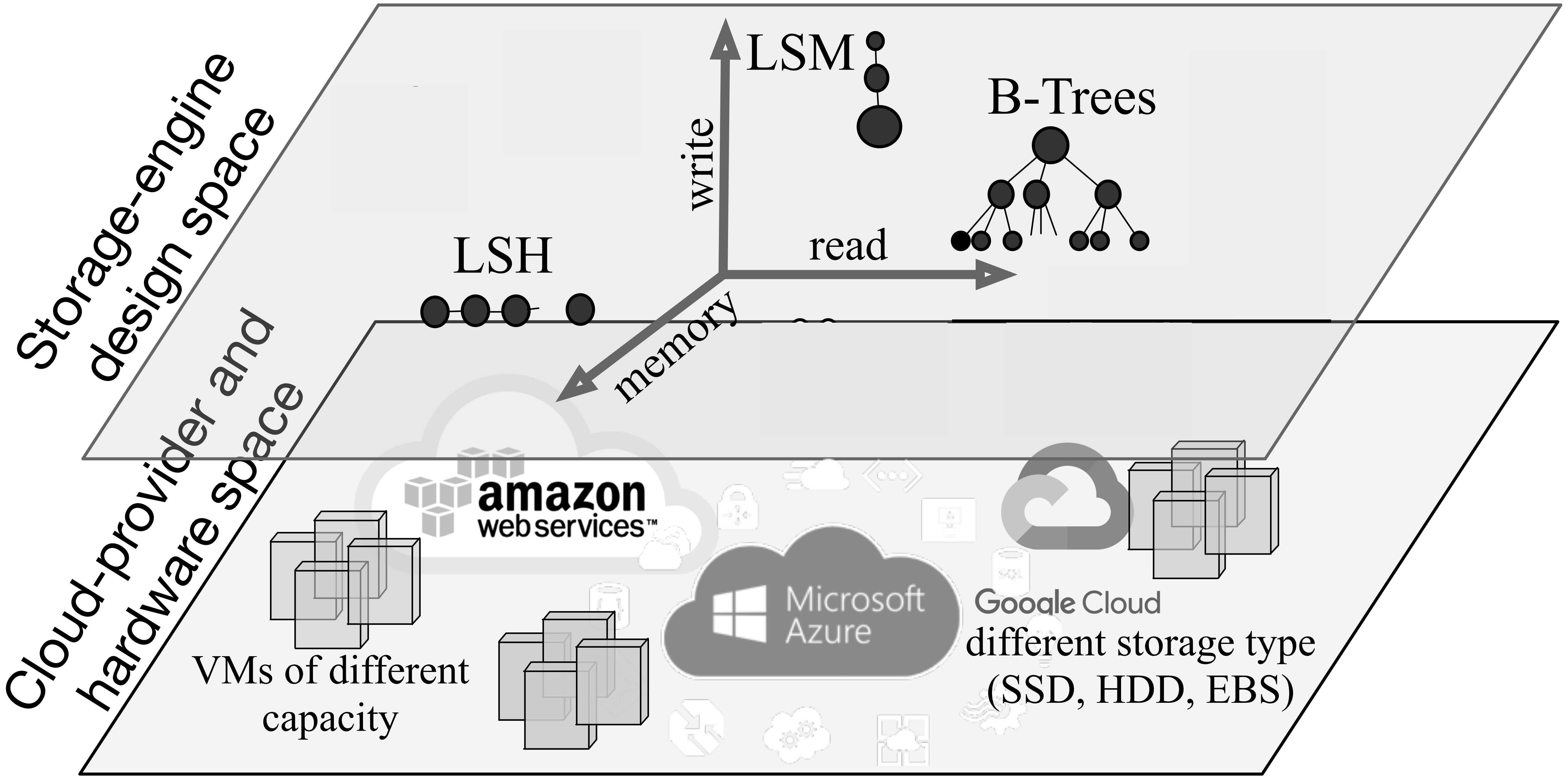




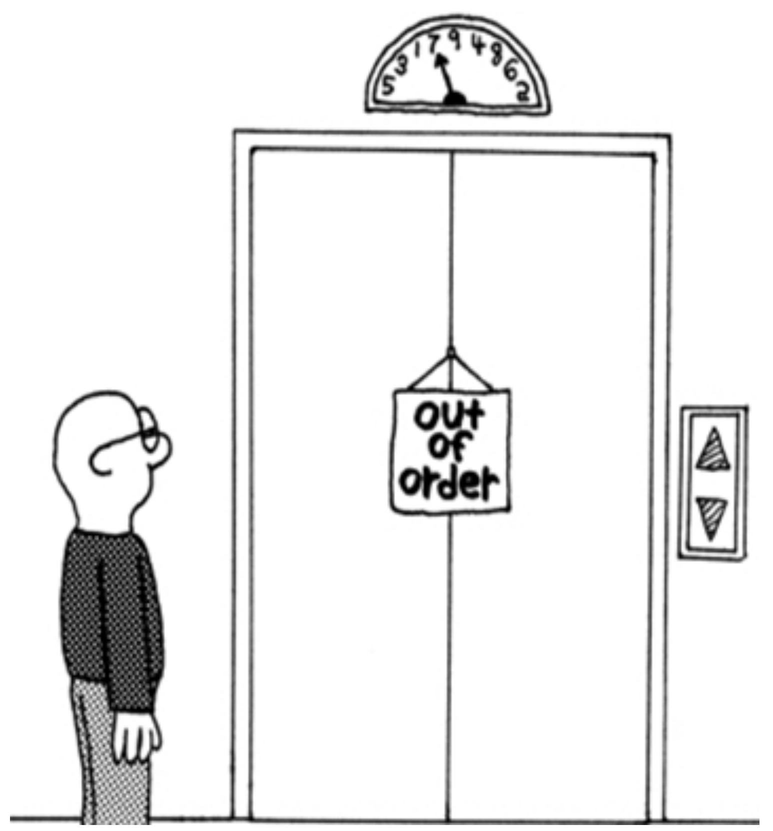
PERFORMANCE



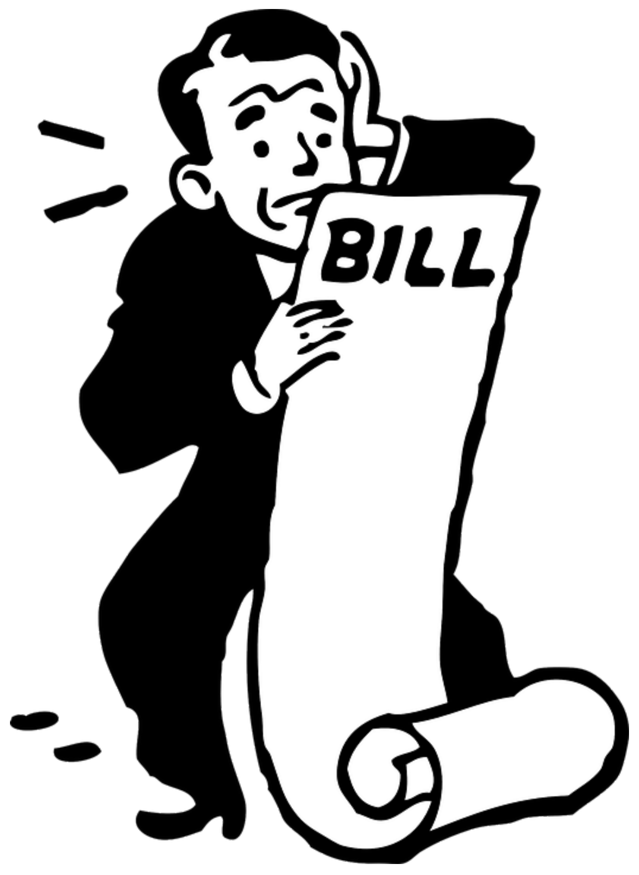
COST



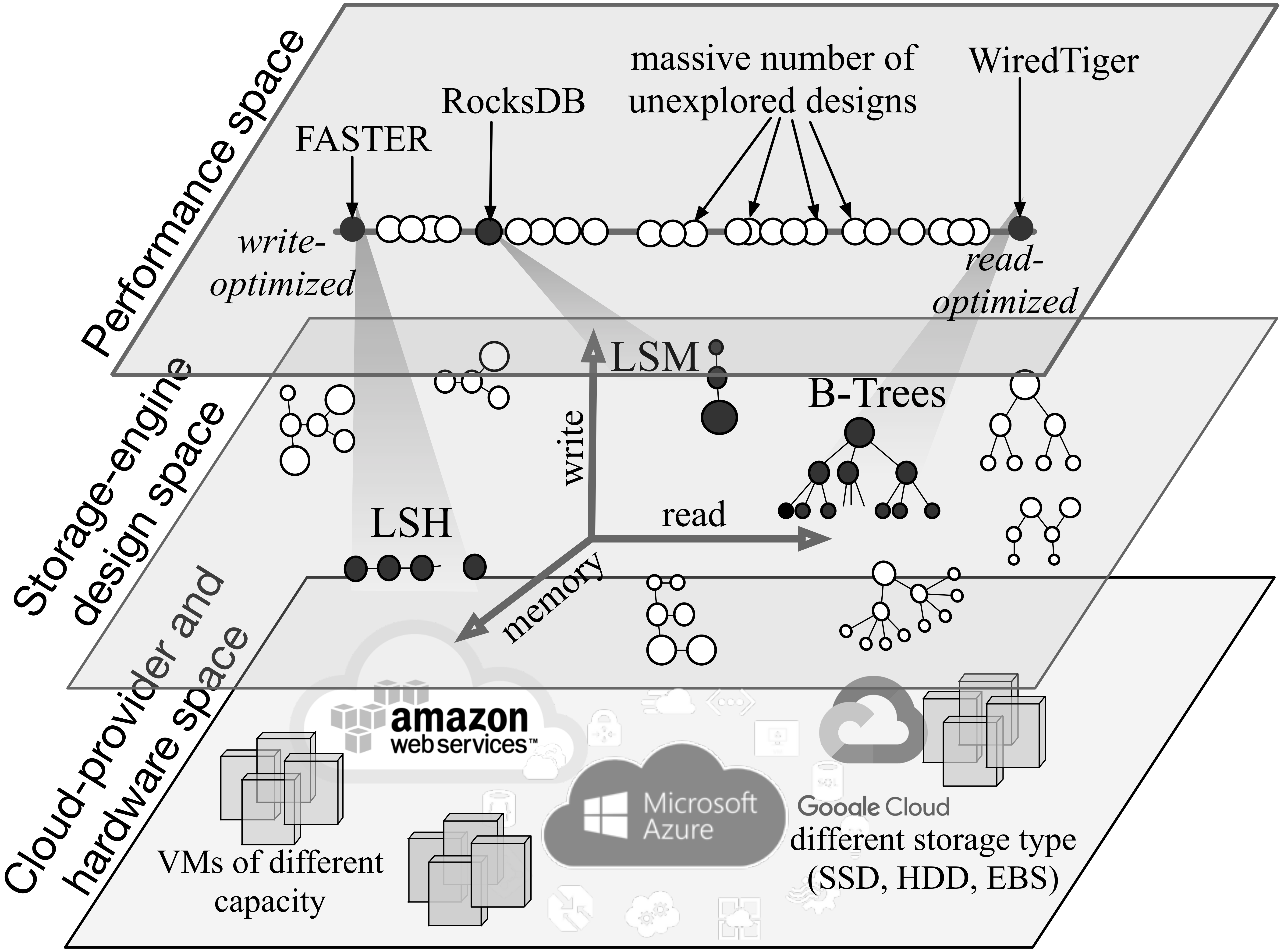
DASlab @ Harvard SEAS



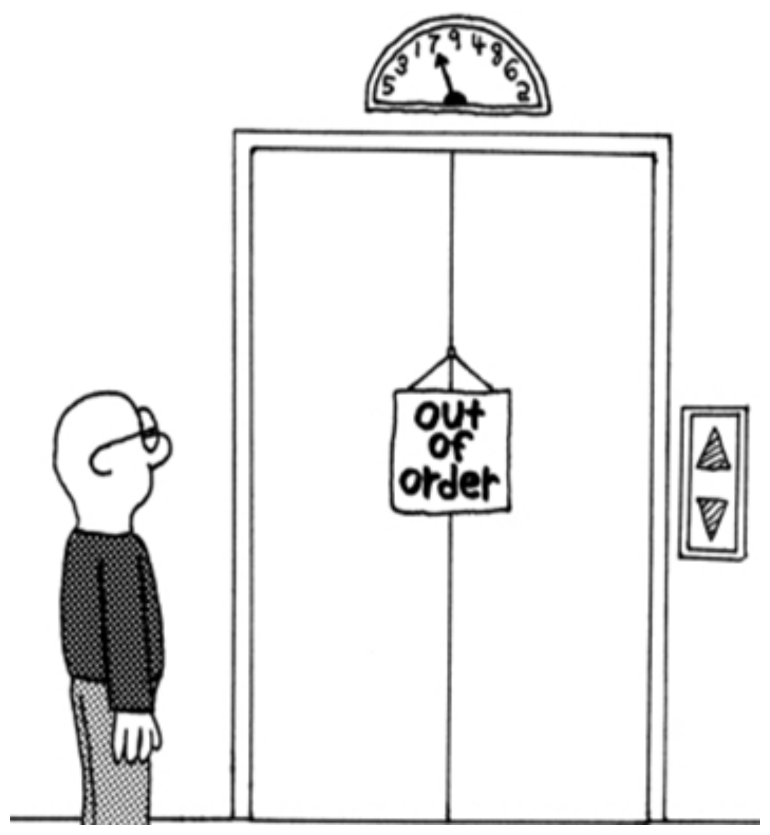
PERFORMANCE



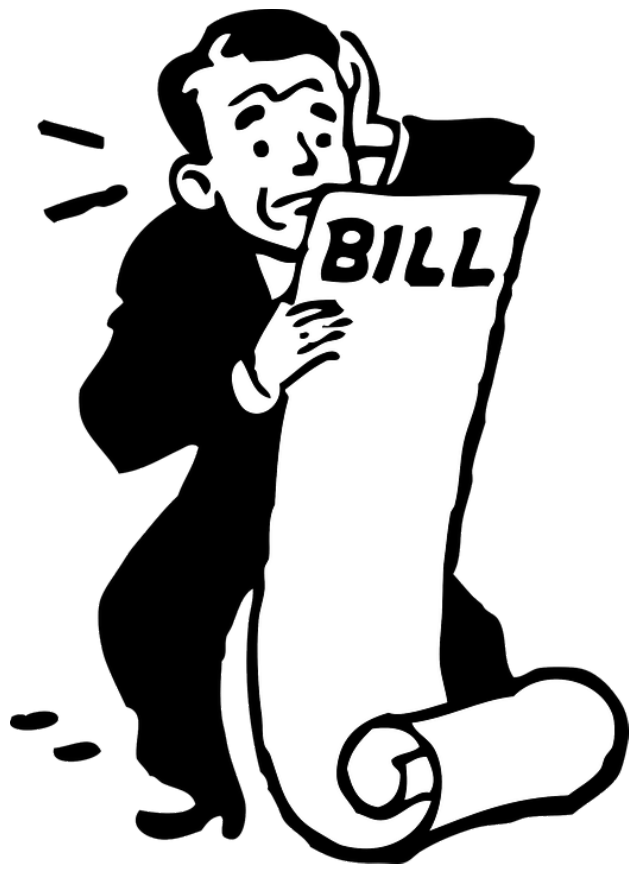
COST



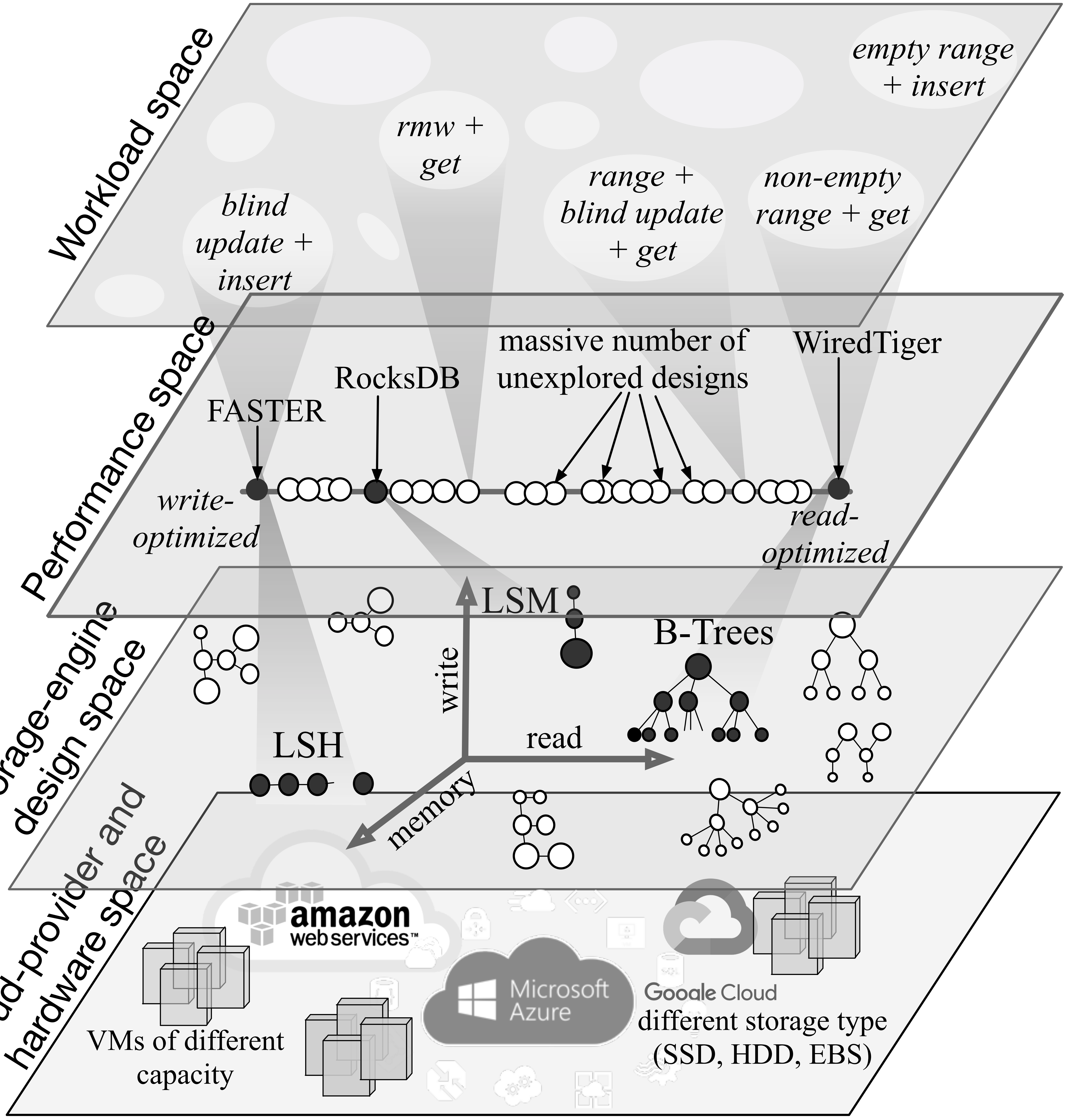
DASlab
@ Harvard SEAS



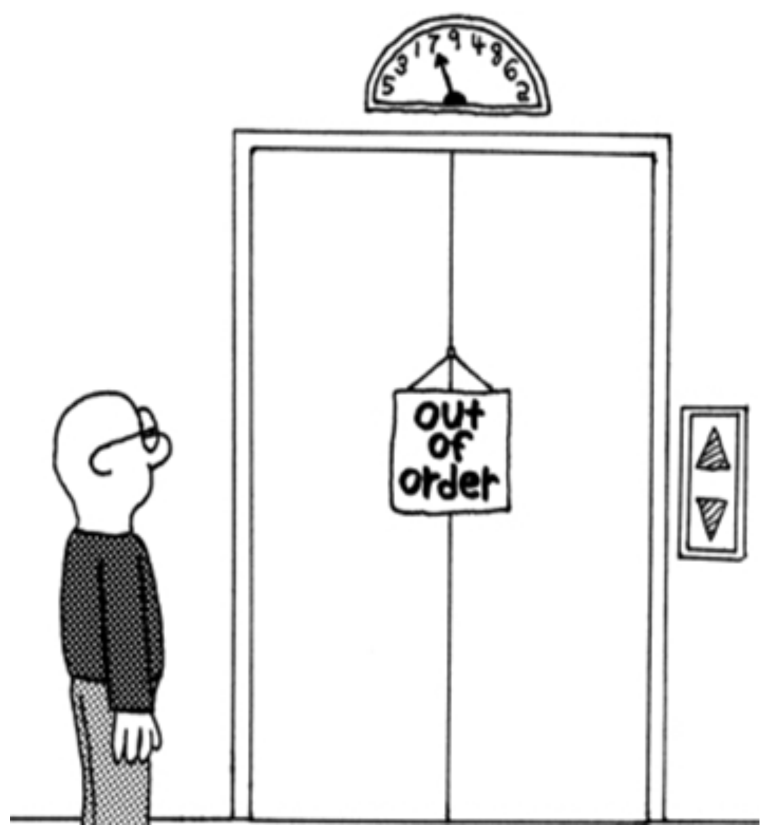
PERFORMANCE



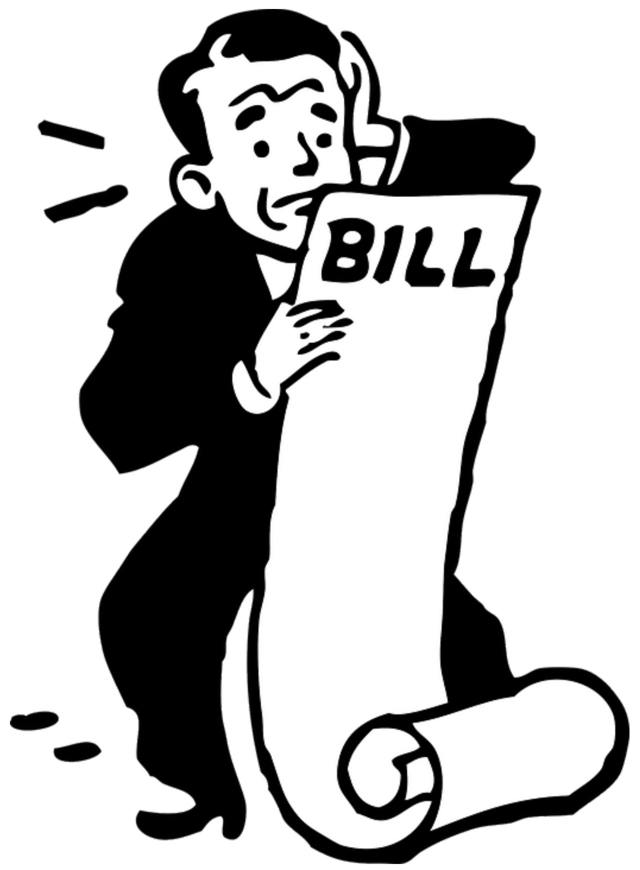
COST



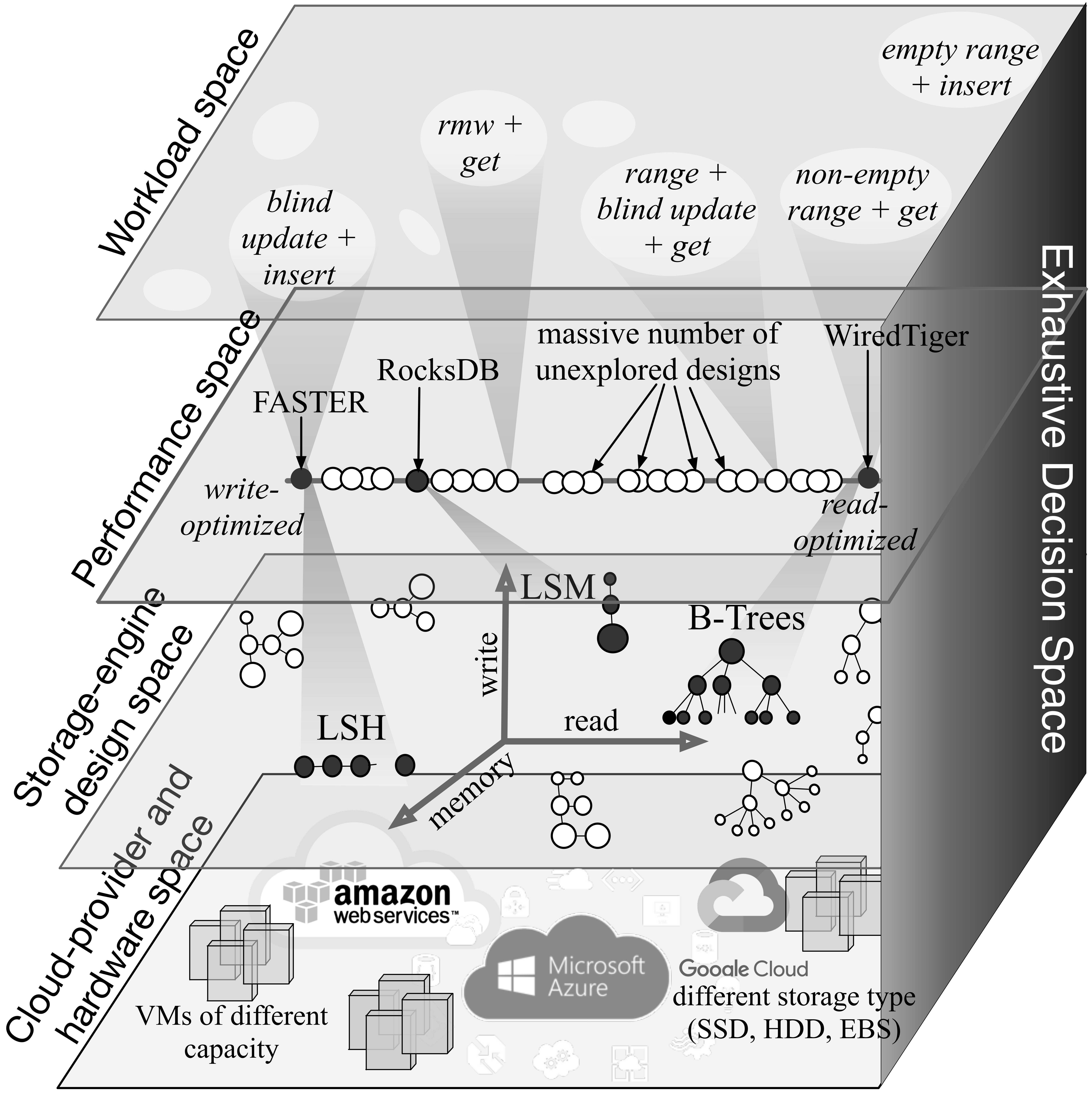
DASlab @ Harvard SEAS



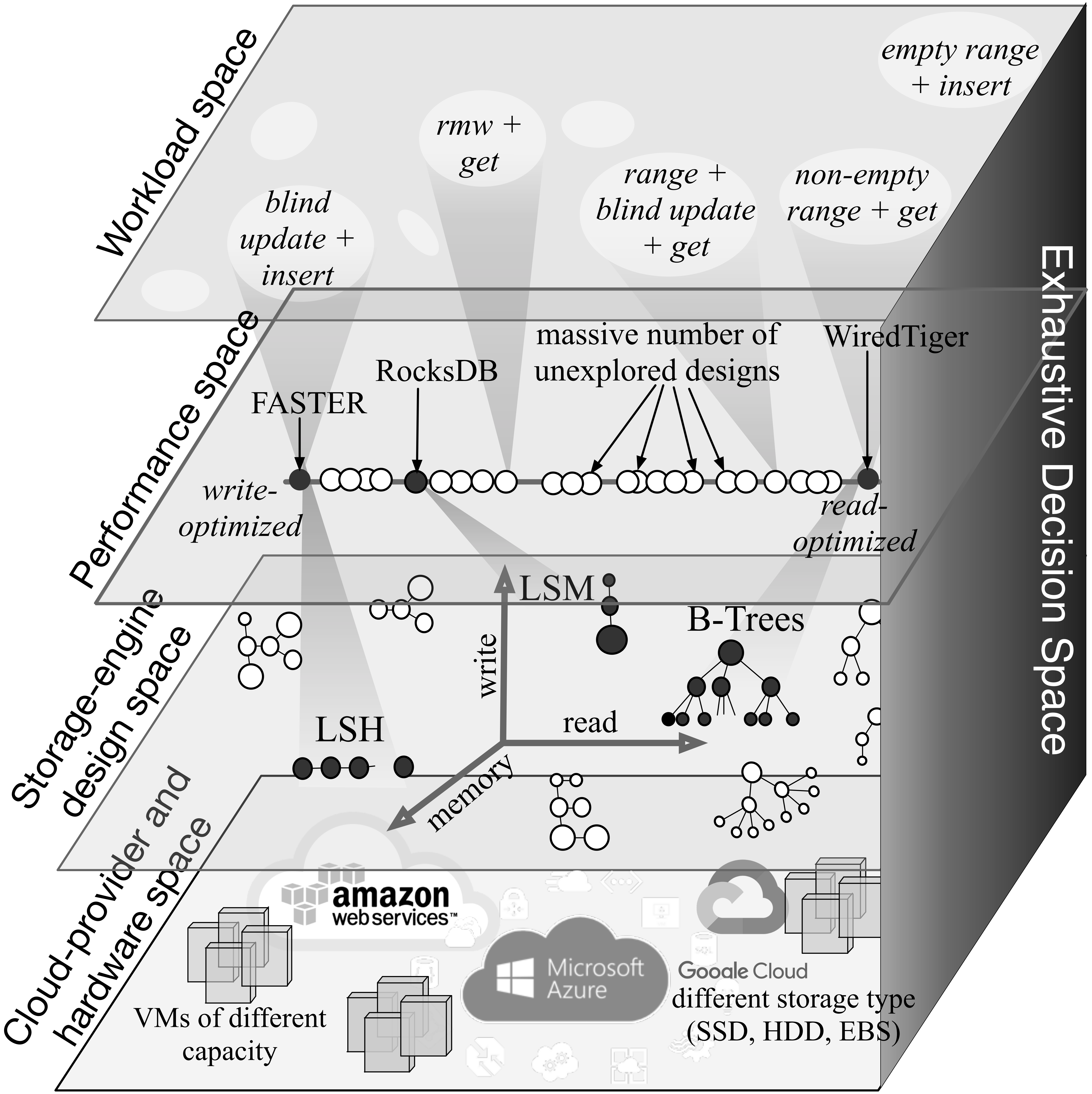
PERFORMANCE



COST



DASlab @ Harvard SEAS



Complex

Vast
(10^35 possibilities)

Manual
(Limited exploration of systems)

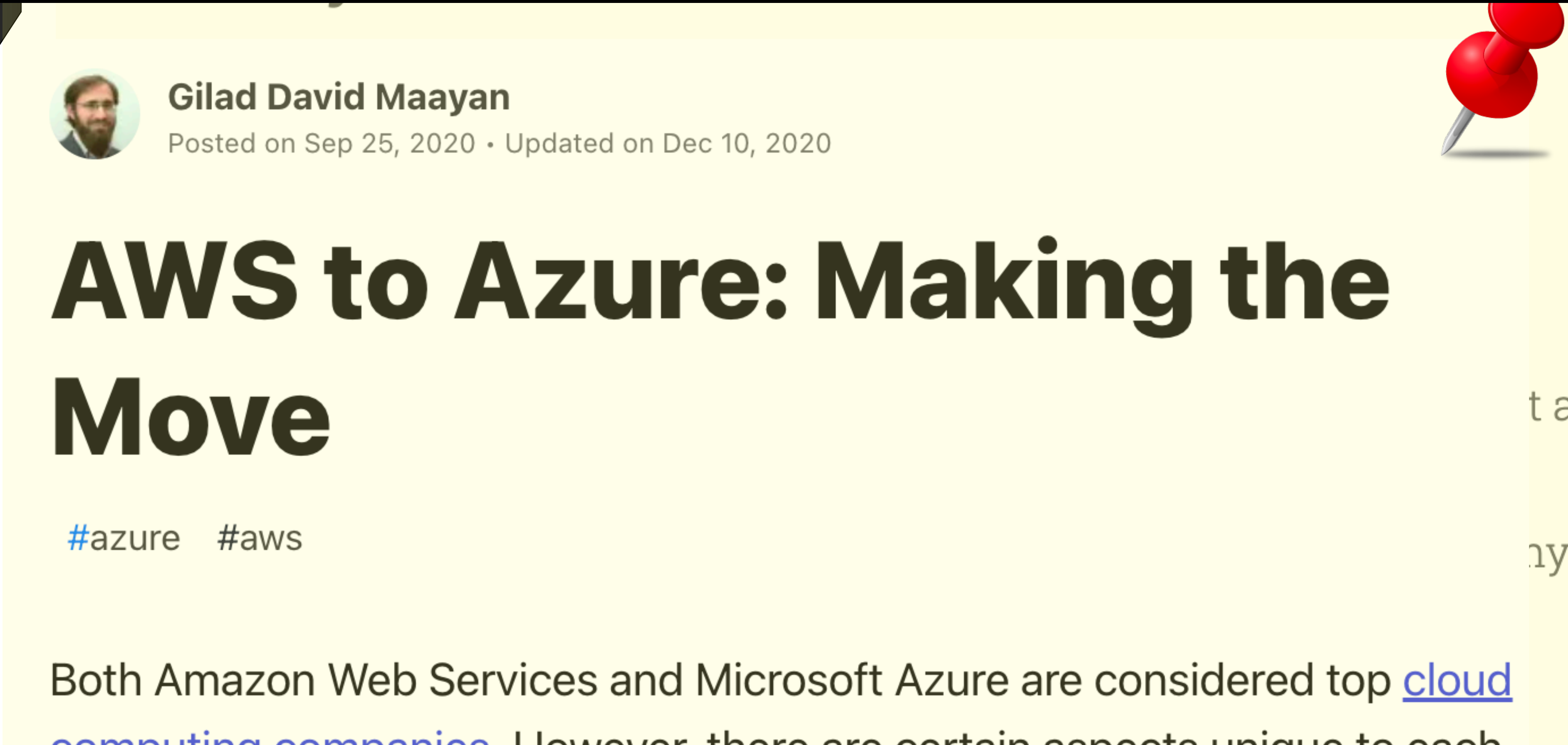
DASlab @ Harvard SEAS

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

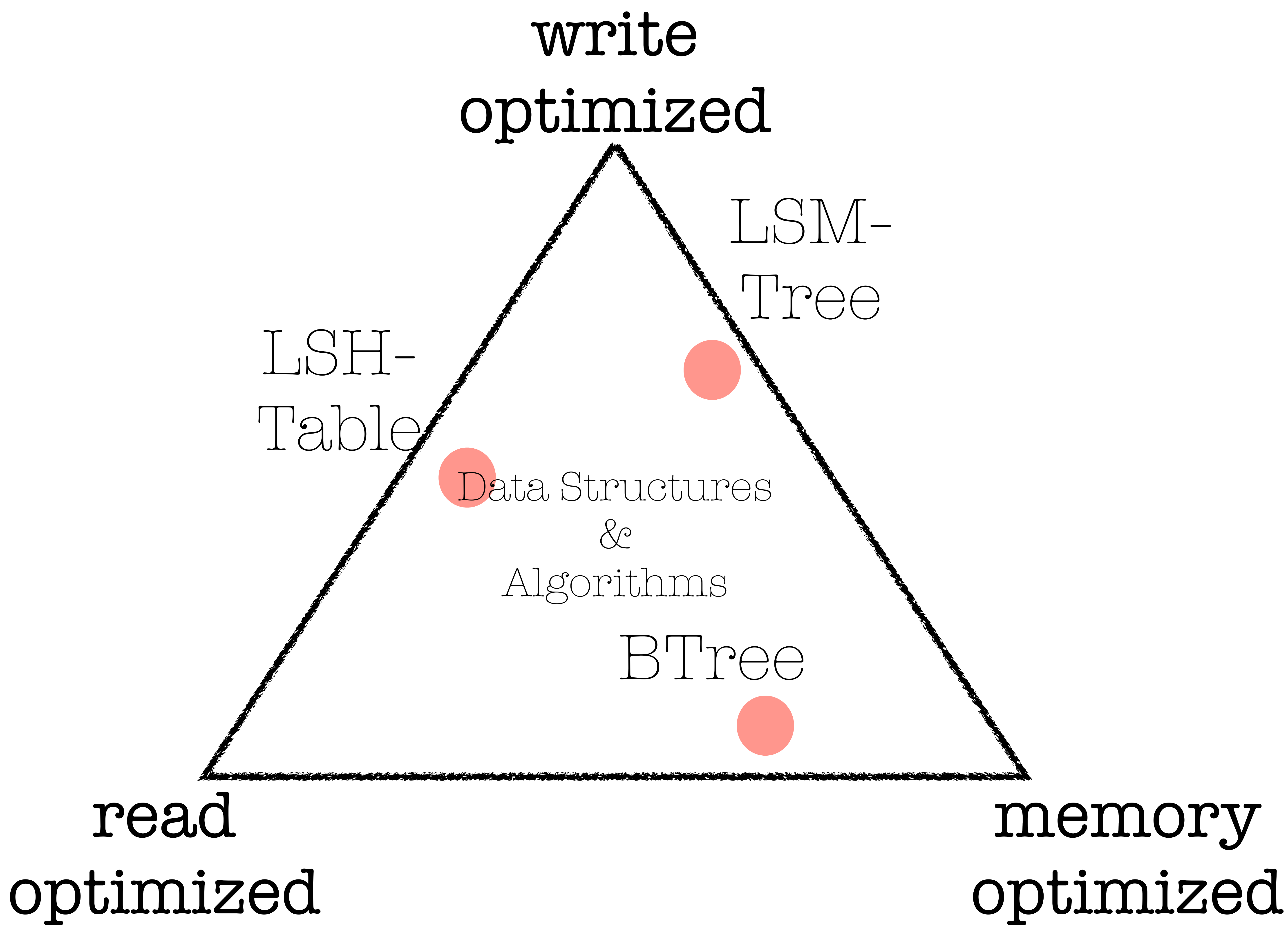
AWS to Azure: Making the Move

#azure #aws

Both Amazon Web Services and Microsoft Azure are considered top cloud

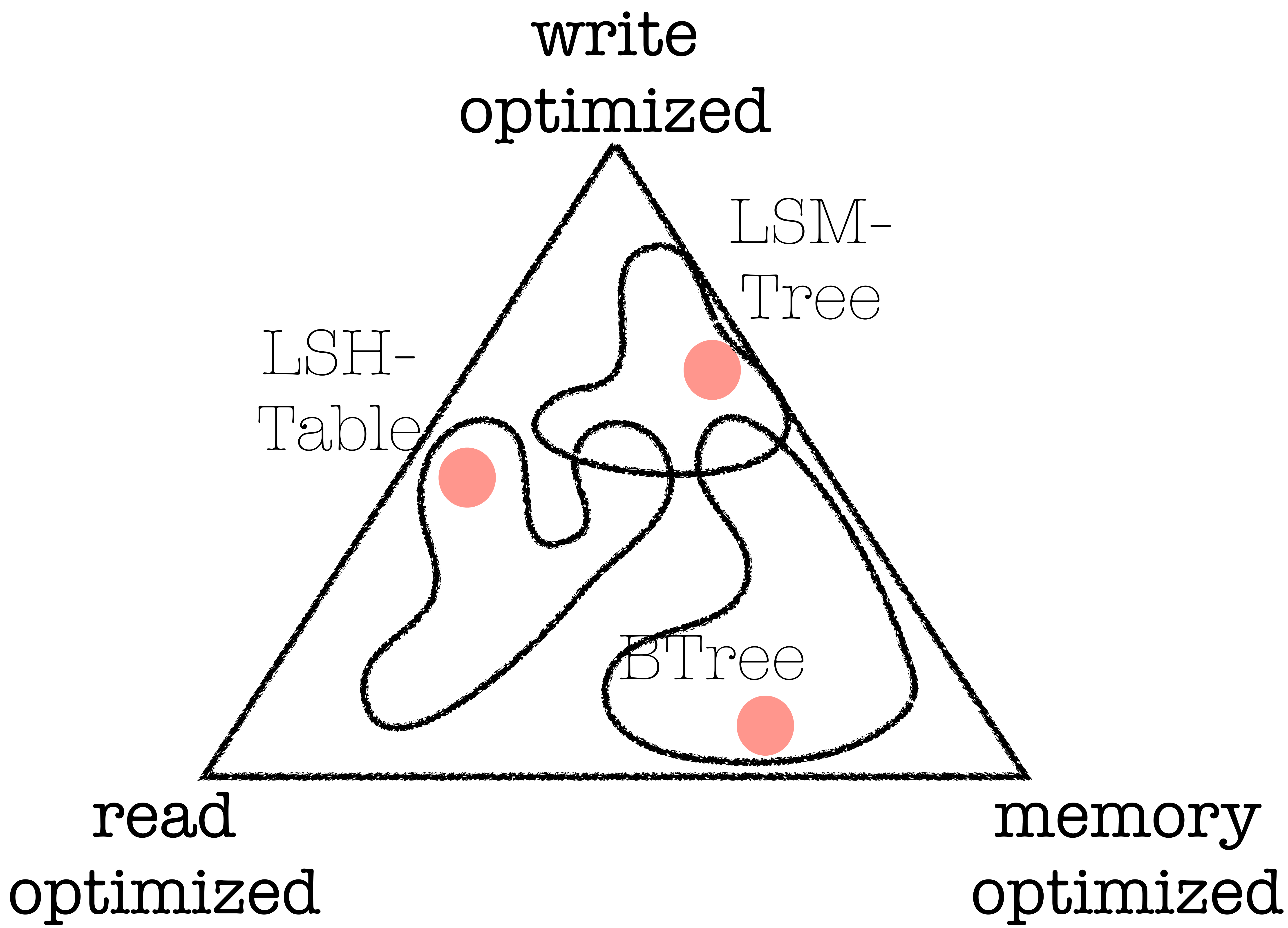
DASlab @ Harvard SEAS

Intuition



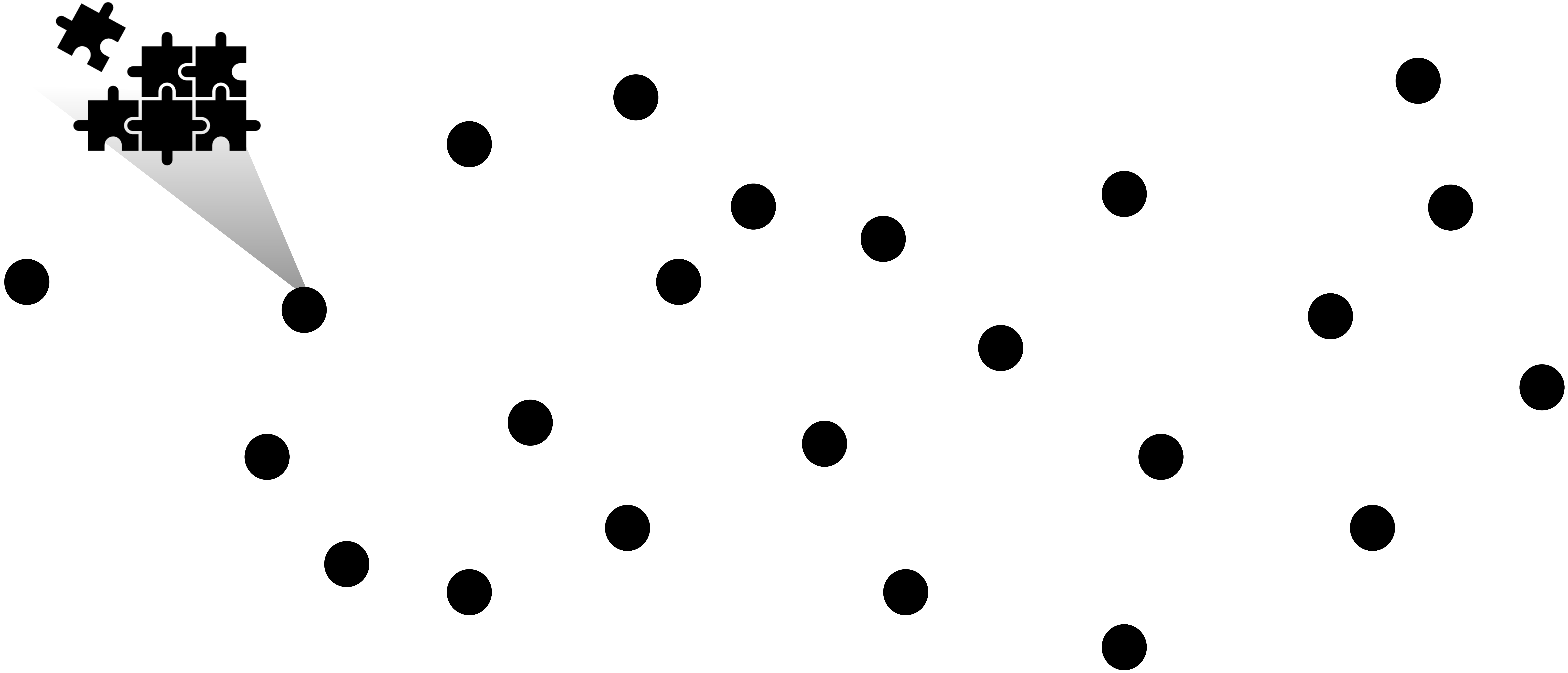
write
optimized
LSM-
Tree
LSH-
Table
Data Structures
&
Algorithms
BTree
read
optimized
memory
optimized

Intuition

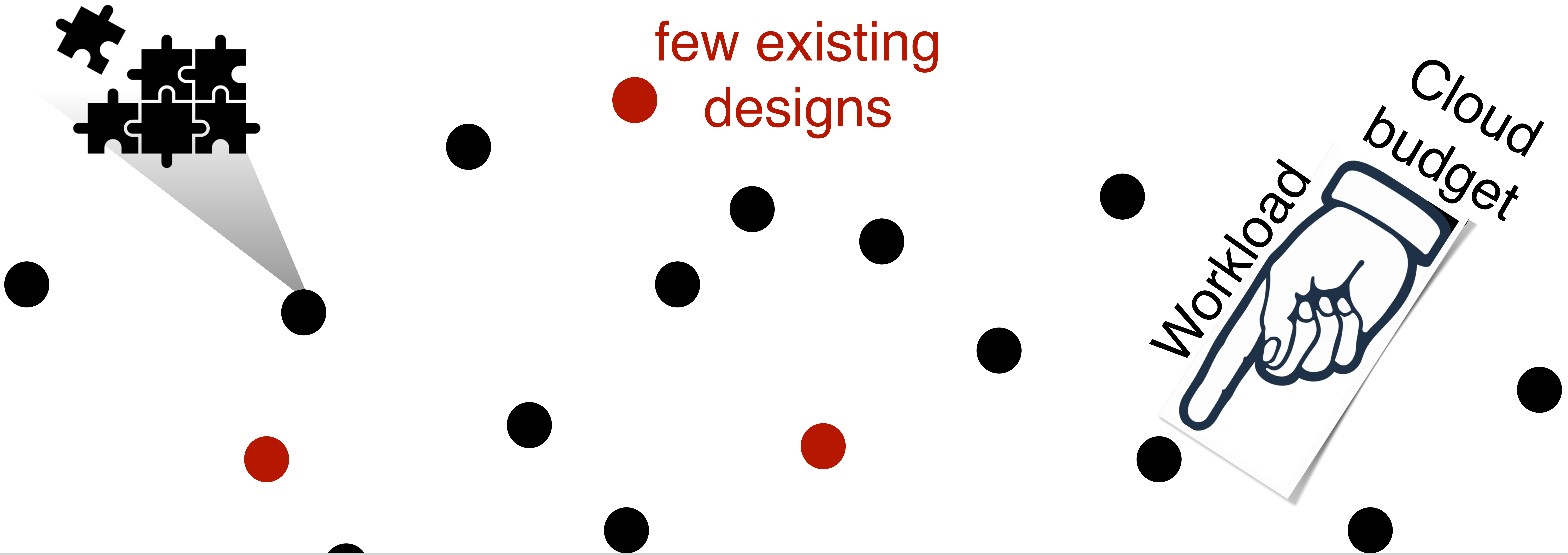


write optimized
LSM-Tree
LSH-Table
BTree
read optimized
memory optimized

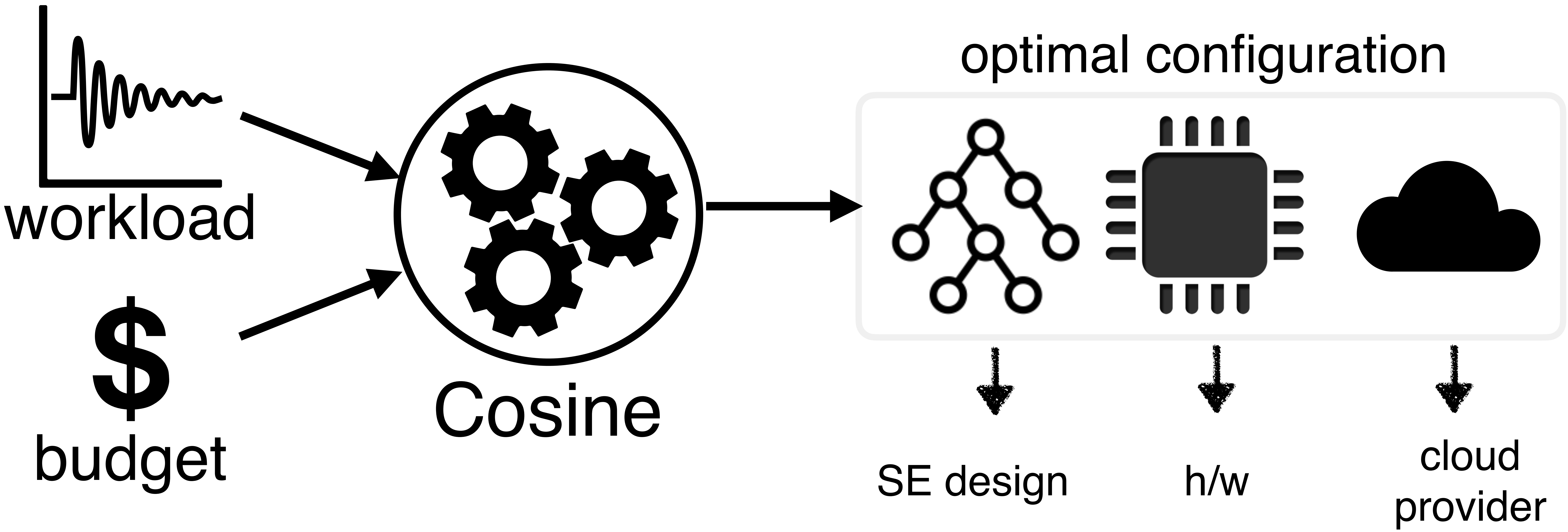
THE KEY **INTUITION**



THE KEY **INTUITION**

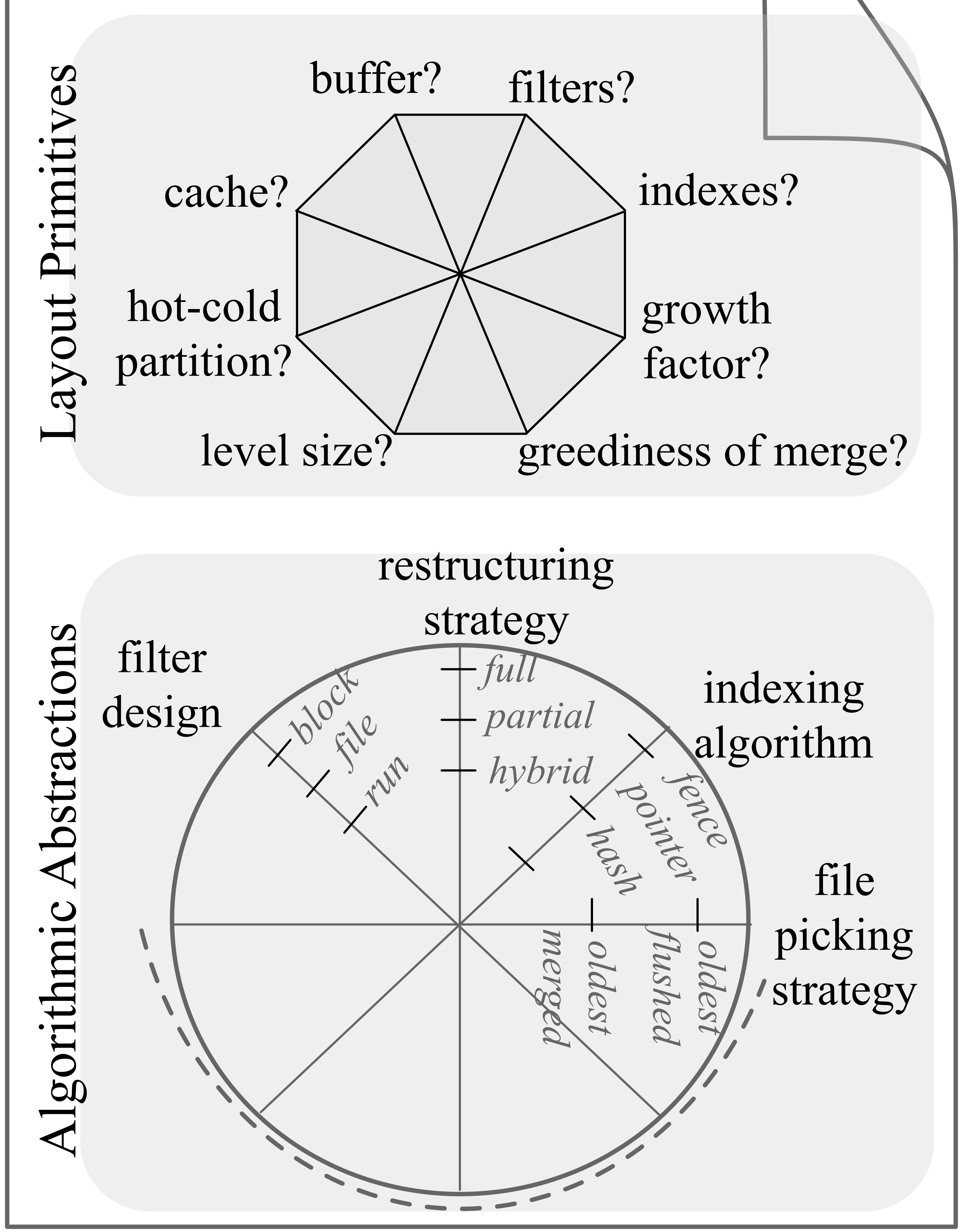


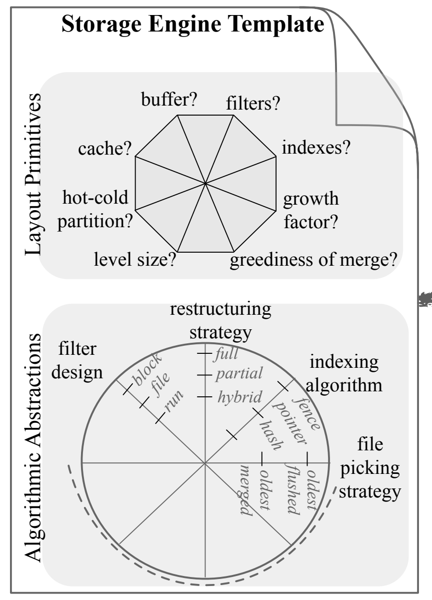
WHAT IF WE COULD **REASON** ABOUT THE MASSIVE **DESIGN SPACE** OF STORAGE ENGINES?



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
block
file
run
hash

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
buffer? [1..M]	storage pattern? {flat logs, hierarchical}
filters? Bloom? Cuckoo? … [1..M]	growth factor? [1, .., multiples of block size]
indexes? hash table? zone map? …	level size? [1, .., L]
cache? [1..M]	greediness of merge? [1 (high), .., T (low)]
	file size? MB … GB
	hot-cold partition?

Superstructure

MEMORY

buffer?

filters?

indexes?

cache?

DISK

storage pattern?

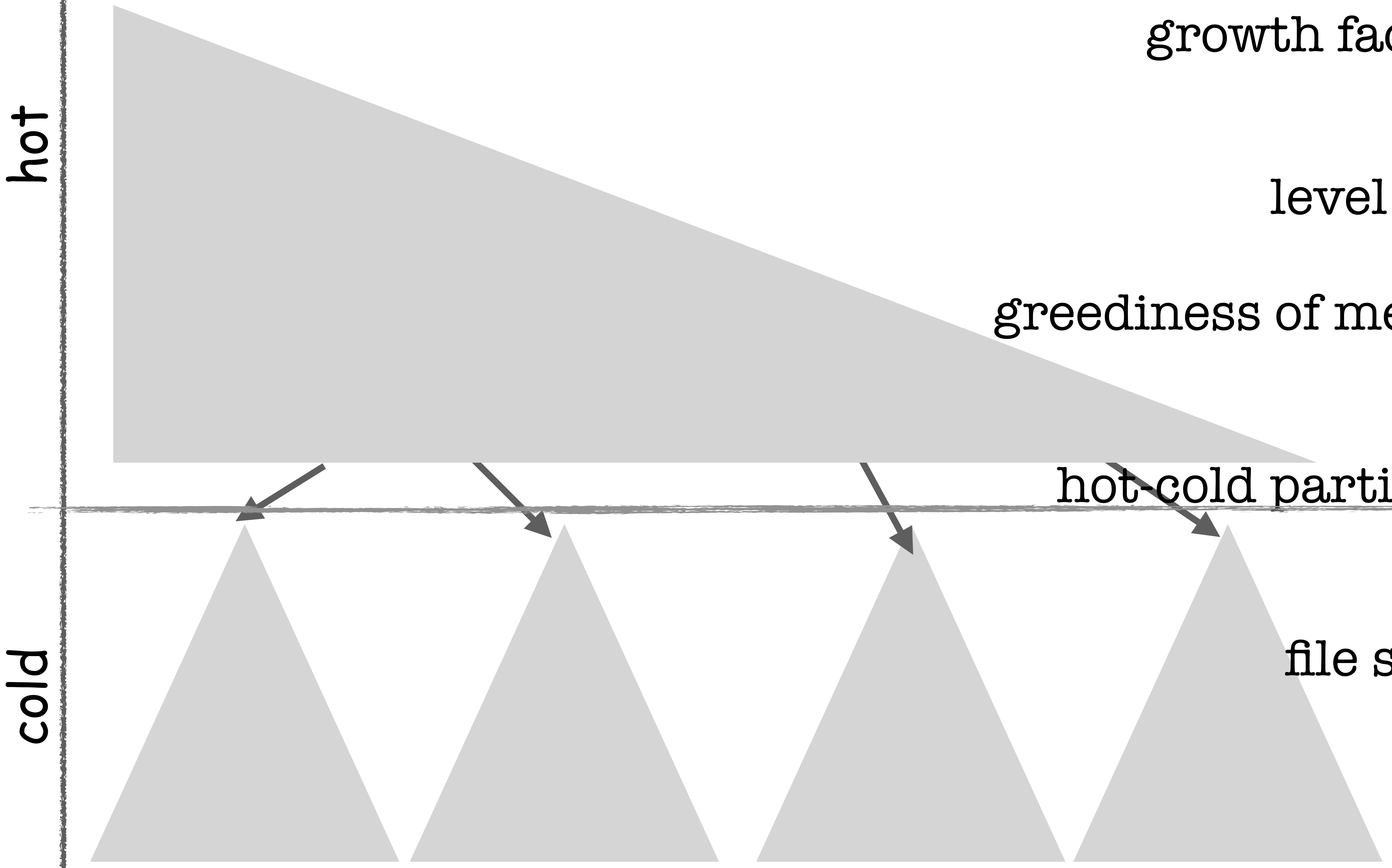
growth factor?

level size?

greediness of merge?

hot-cold partition?

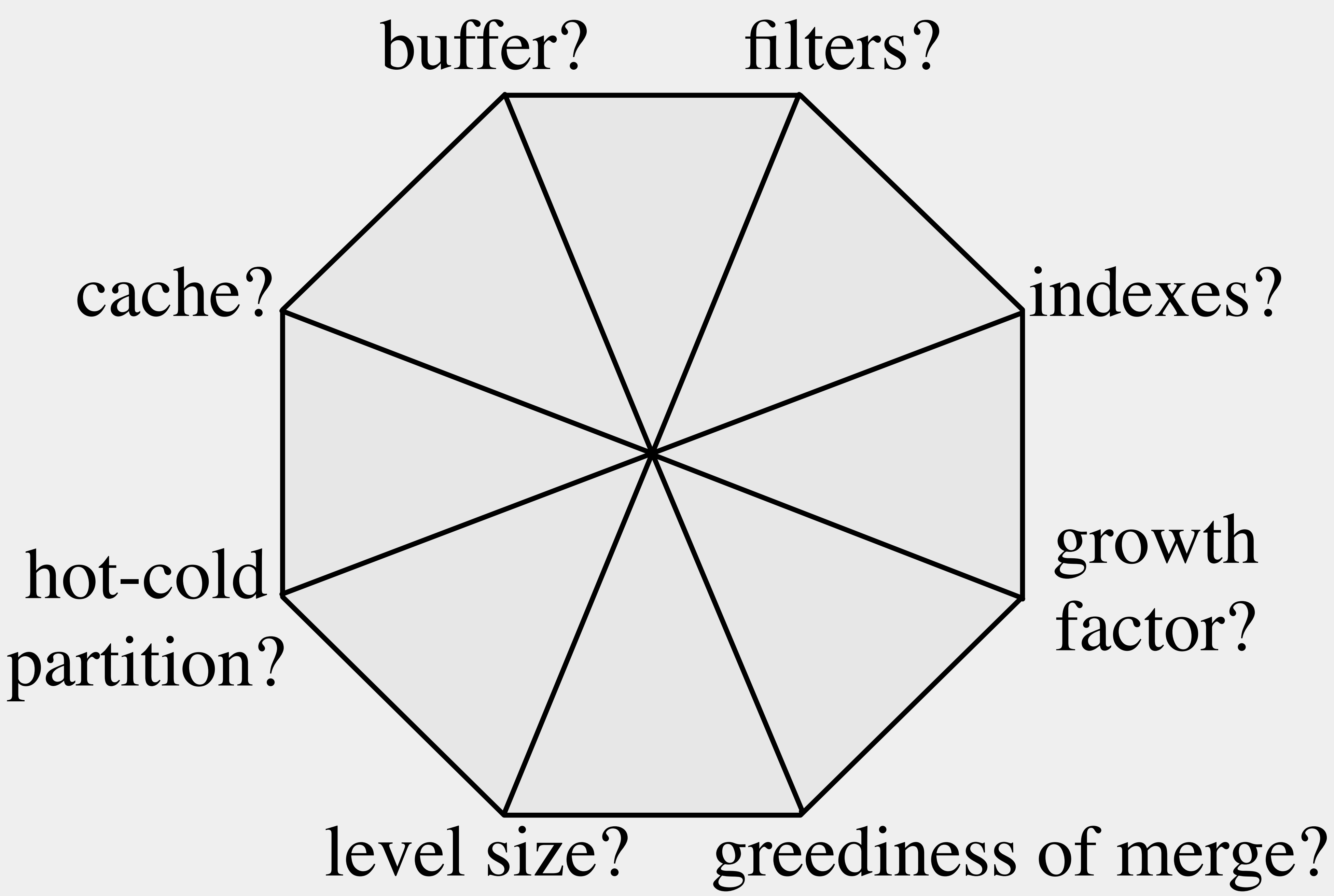
file size?



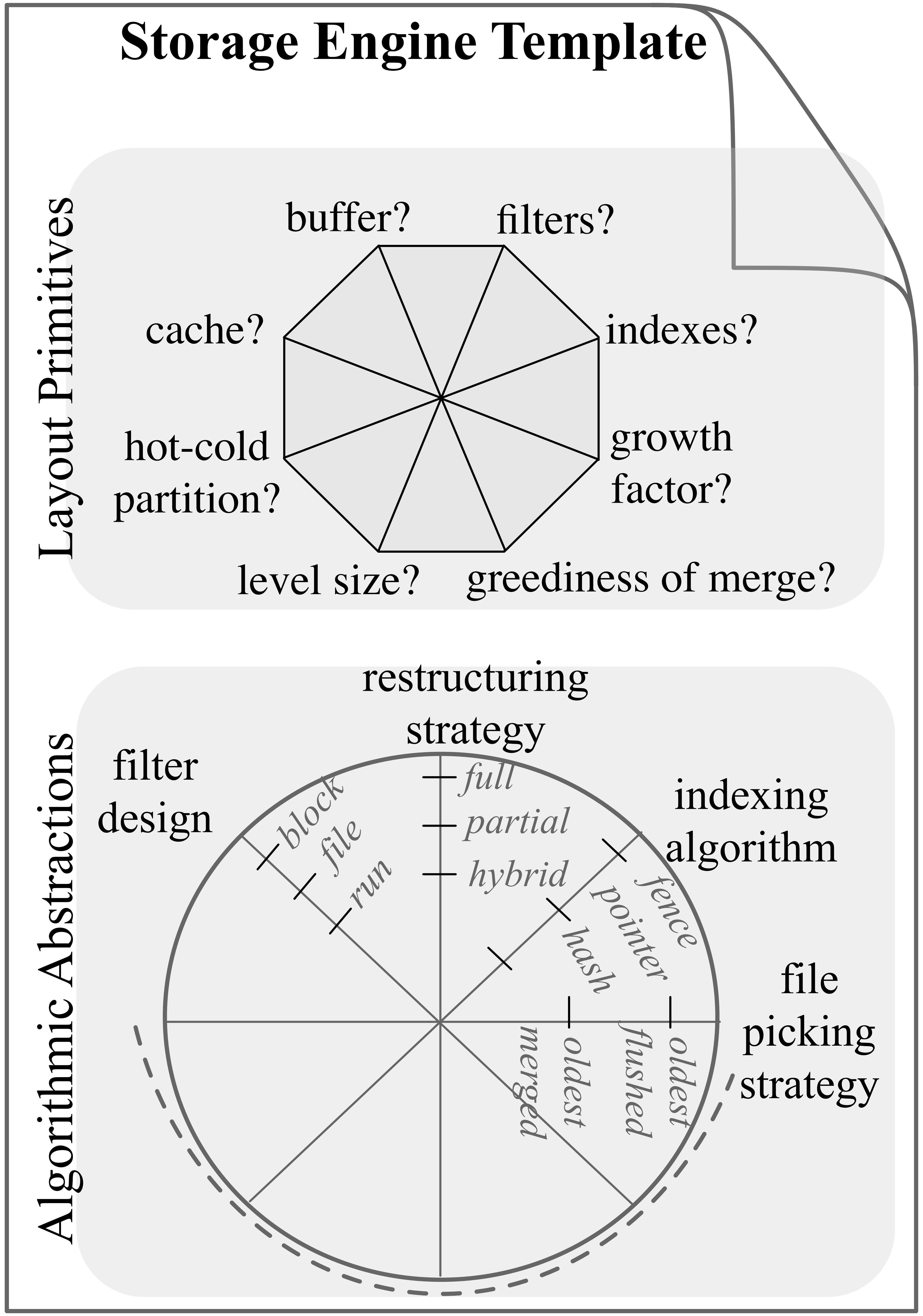
MEMORY
DISK
buffer?
filters?
indexes?
cache?
hot
cold
storage pattern?
growth factor?
level size?
greediness of merge?
hot-cold partition?
file size?
Leveled LSM
Tiered LSM
BTree
New layout

Storage Engine Template

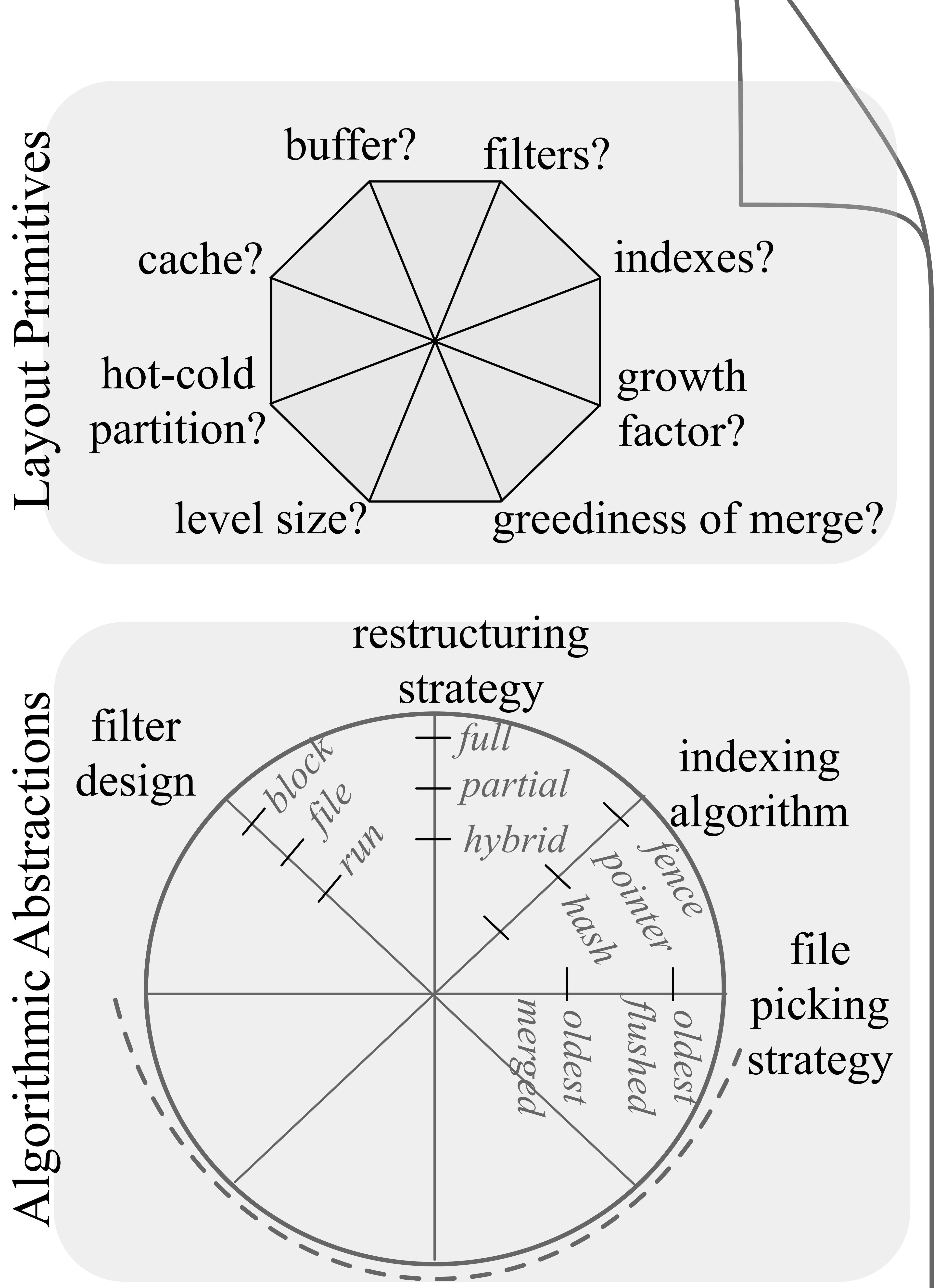
Layout Primitives

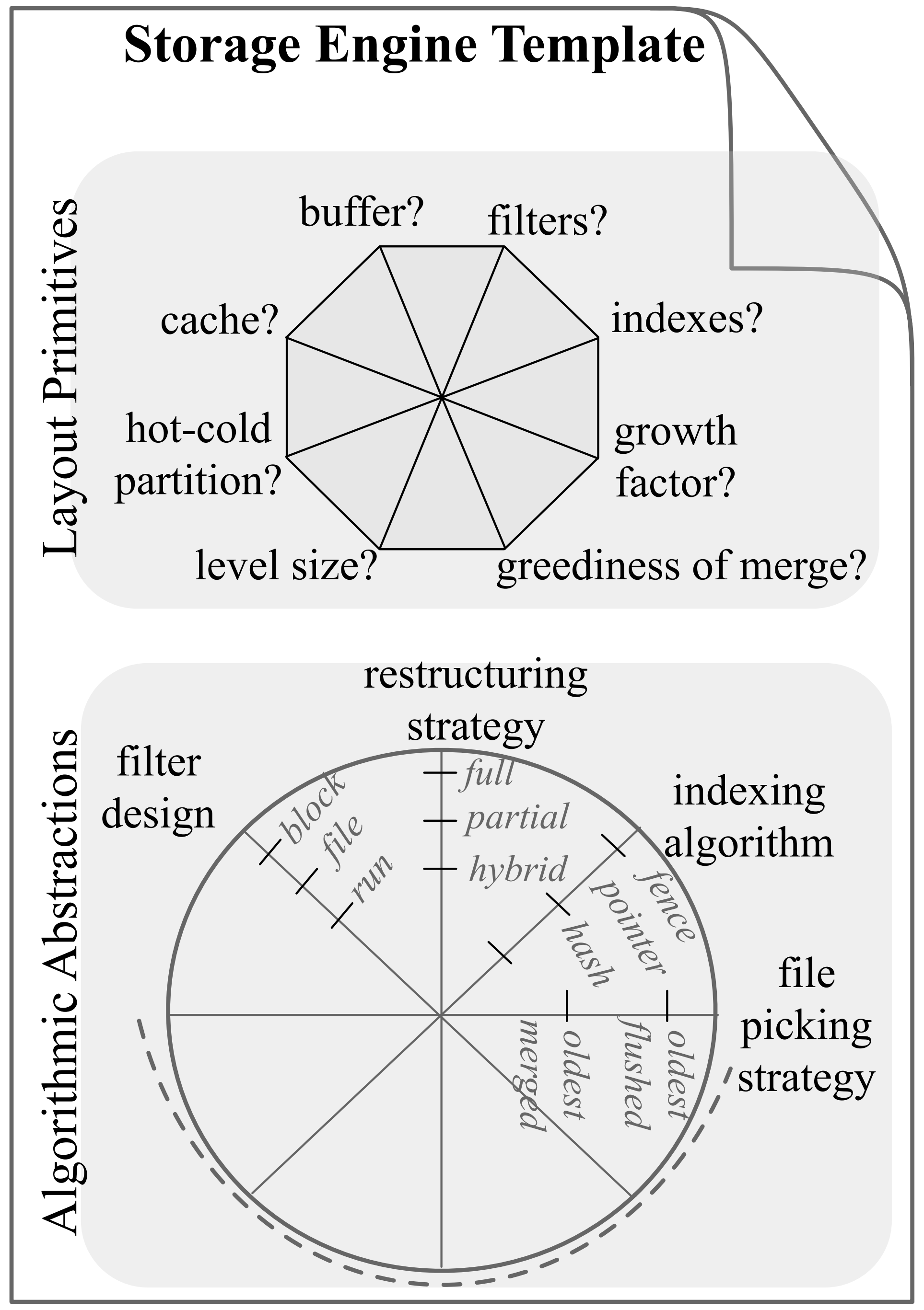


DASlab @ Harvard SEAS


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

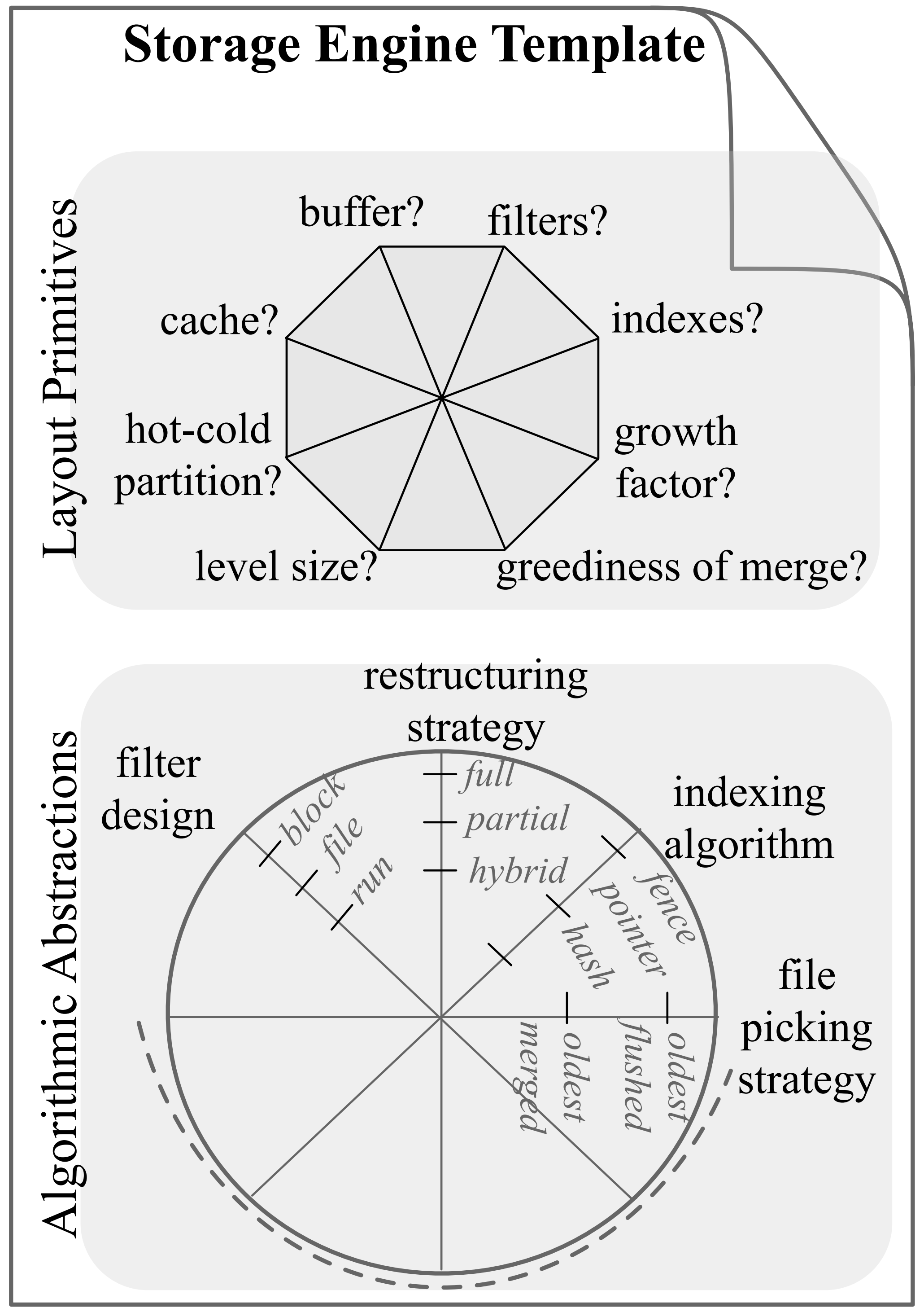
Storage Engine Template



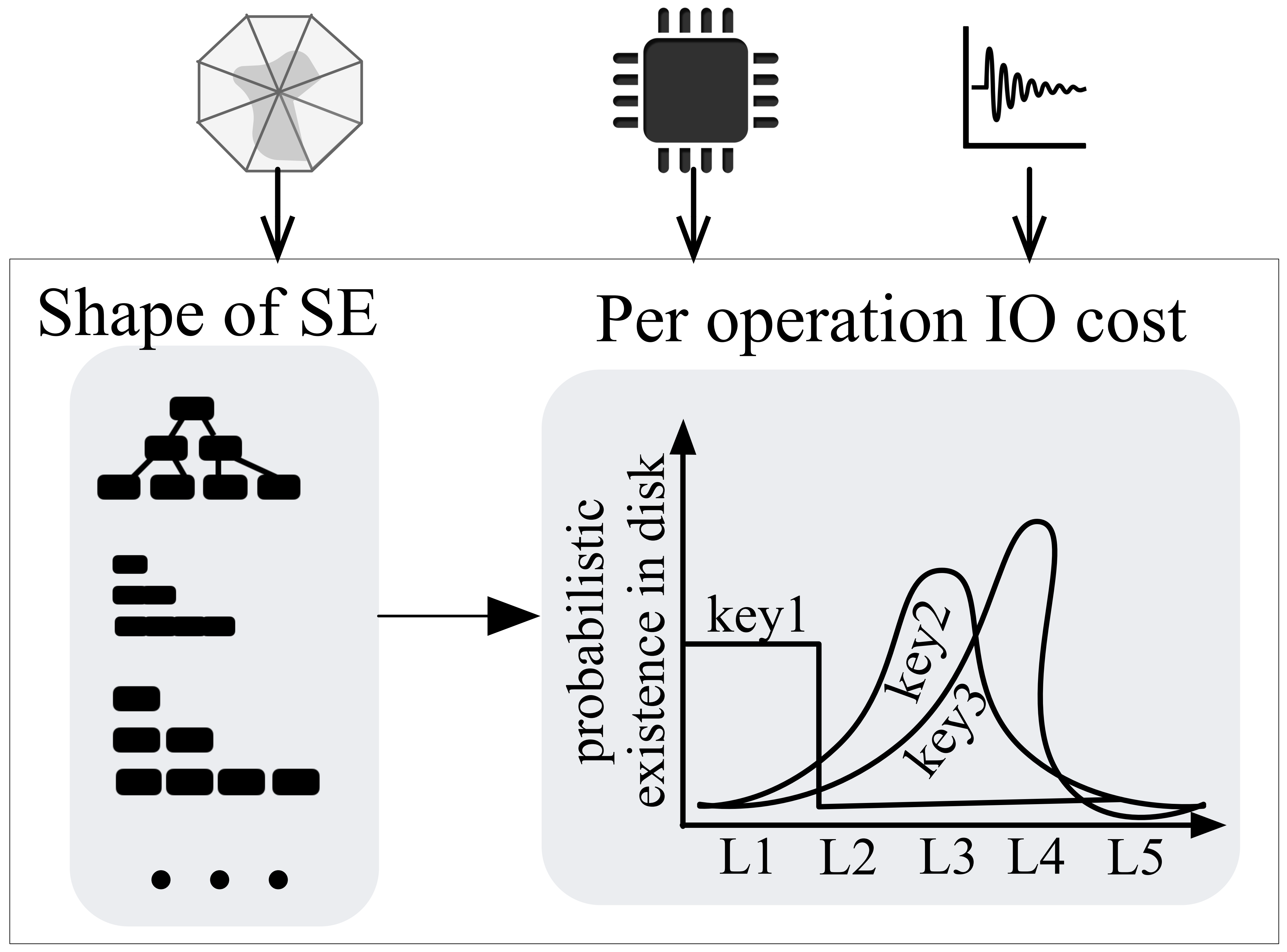


				Design Abstractions of Template	Type/Domain	Example templates for diverse data structures			
						RocksDB variants	WiredTiger variants	FASTER variants	A new design
LAYOUT PRIMITIVES	Design and hardware specification	*initialized by search through engine design space*	1.	**Key size:** Denotes the size 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
	Parallelism		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			

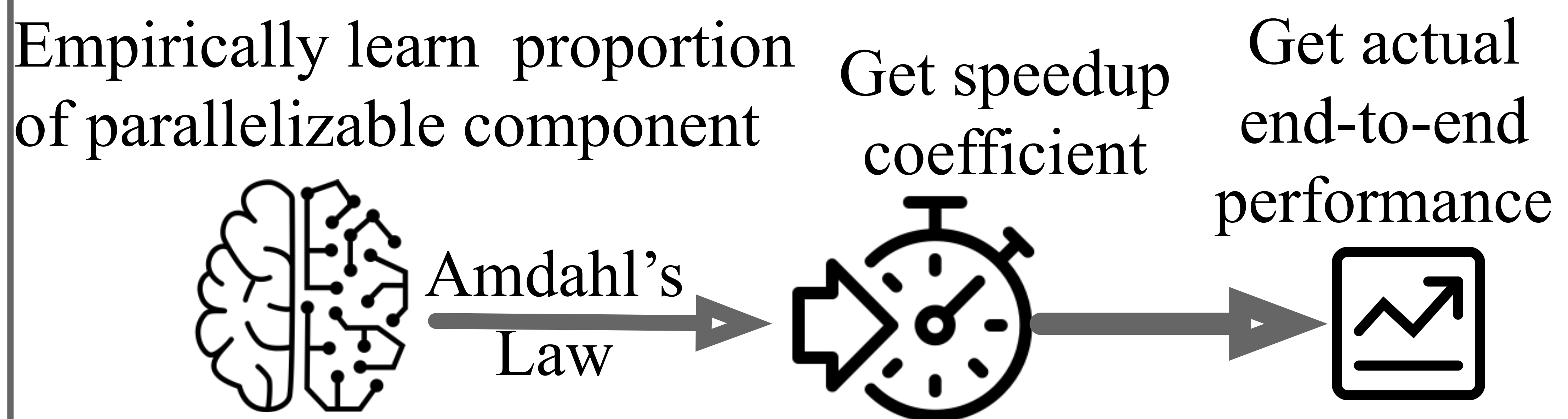
DASlab @ Harvard SEAS

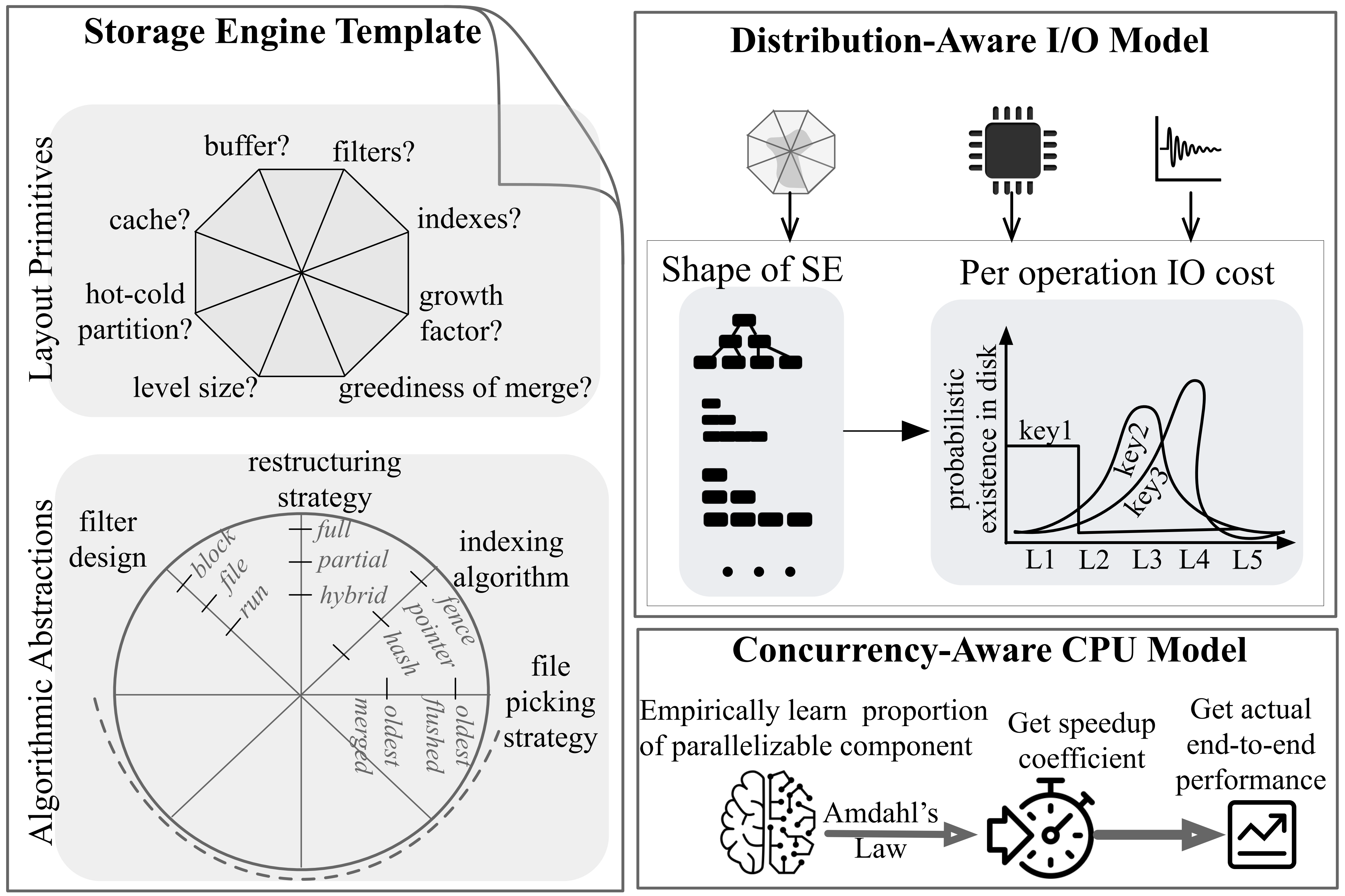


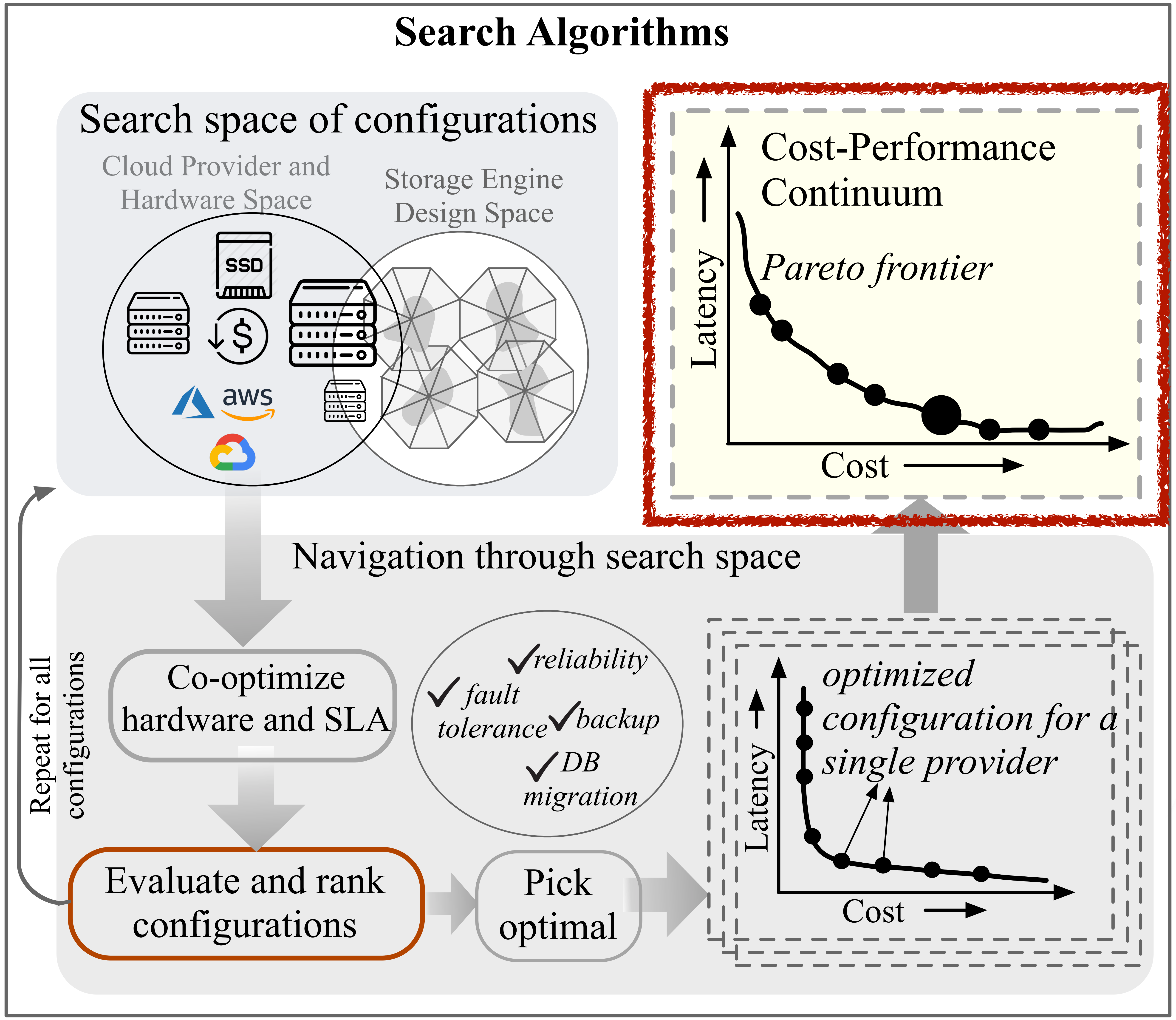
Distribution-Aware I/O Model



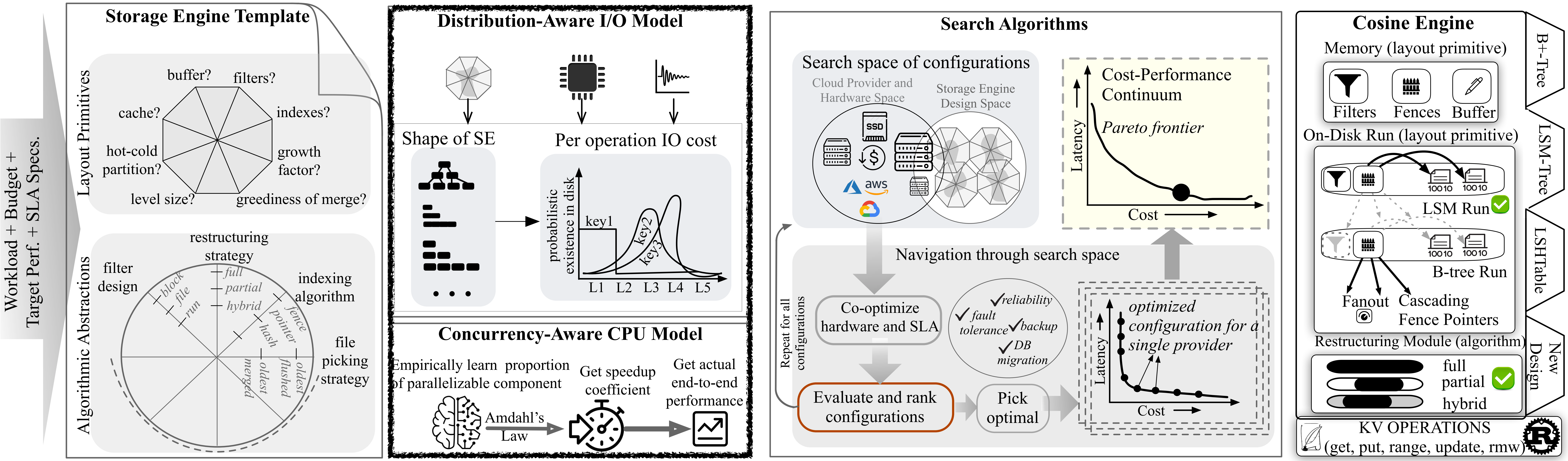
Concurrency-Aware CPU Model







DASlab @ Harvard SEAS



DASlab
@ Harvard SEAS