CS 265

Stratos Idreos

BIG DATA SYSTEMS

NoSQL | Neural Networks | Image AI | LLMs | Data Science

HOW TO JUDGE A DESIGN?

HOW TO JUDGE A DESIGN?

1

COMPLEXITY ANALYSIS

HOW TO JUDGE A DESIGN?

1

COMPLEXITY ANALYSIS

2

IMPLEMENTATION & TESTING

HOW TO JUDGE A DESIGN?

1

COMPLEXITY ANALYSIS

2

IMPLEMENTATION & TESTING

3

GENERALIZED MODELS

HOW TO JUDGE A DESIGN?

1

COMPLEXITY ANALYSIS

2

IMPLEMENTATION & TESTING

3

GENERALIZED MODELS

This sounds ideal: is it possible?

ACCESS PATH SELECTION in ANALYTICAL SYSTEMS

scan vs secondary index selection

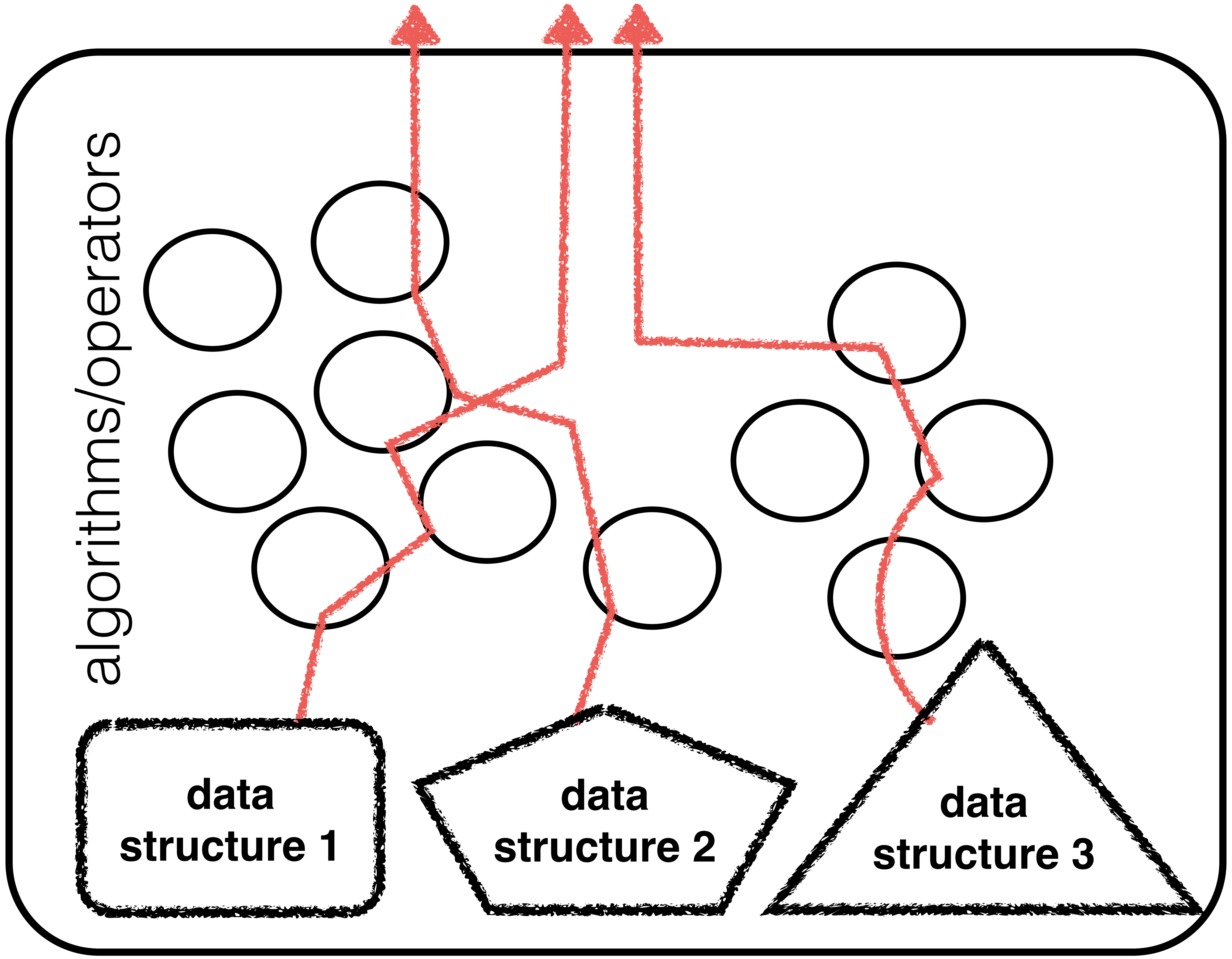
@SIGMOD 2017

data* system

ACCESS PATH SELECTION in ANALYTICAL SYSTEMS

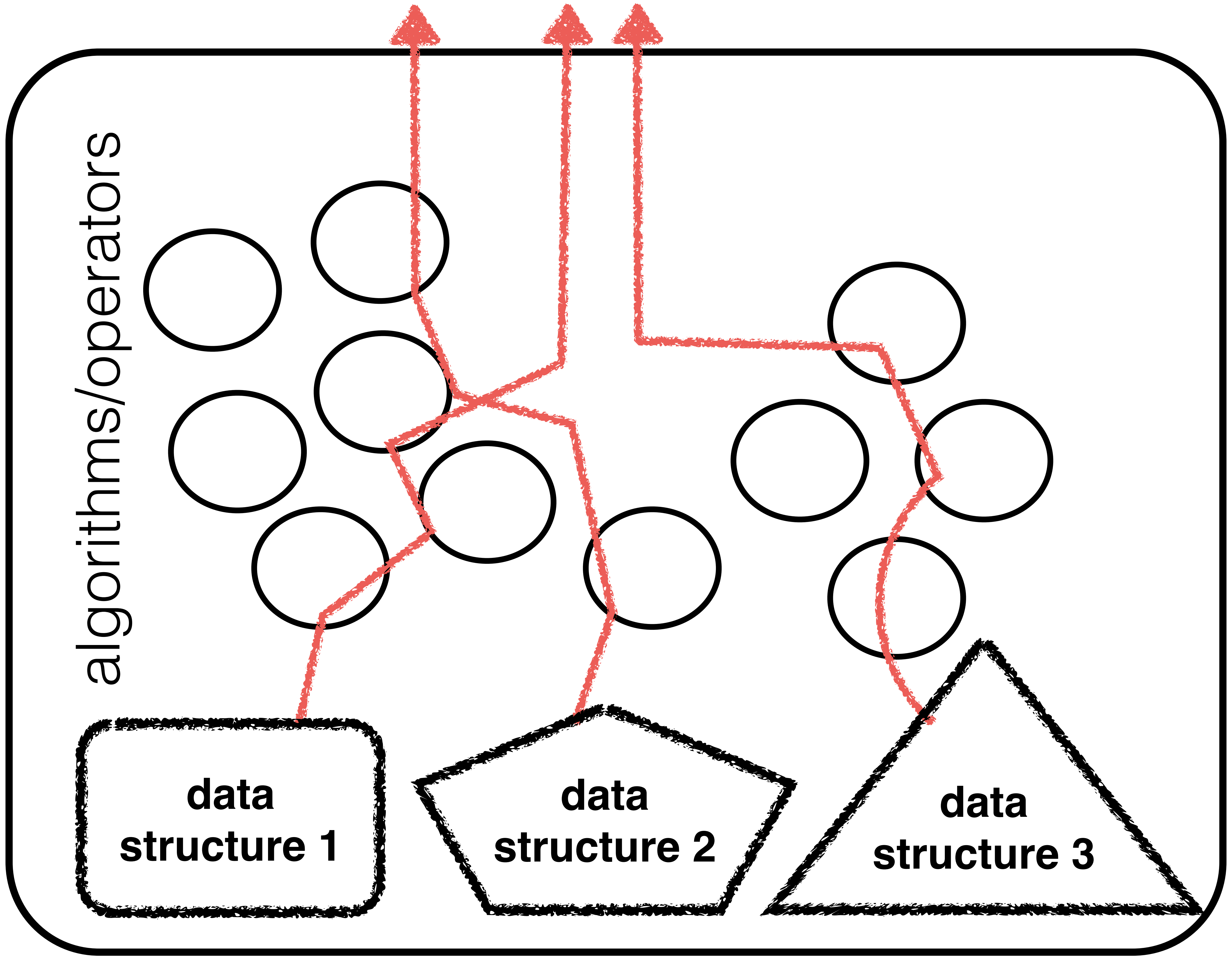
scan vs secondary index selection

@SIGMOD 2017

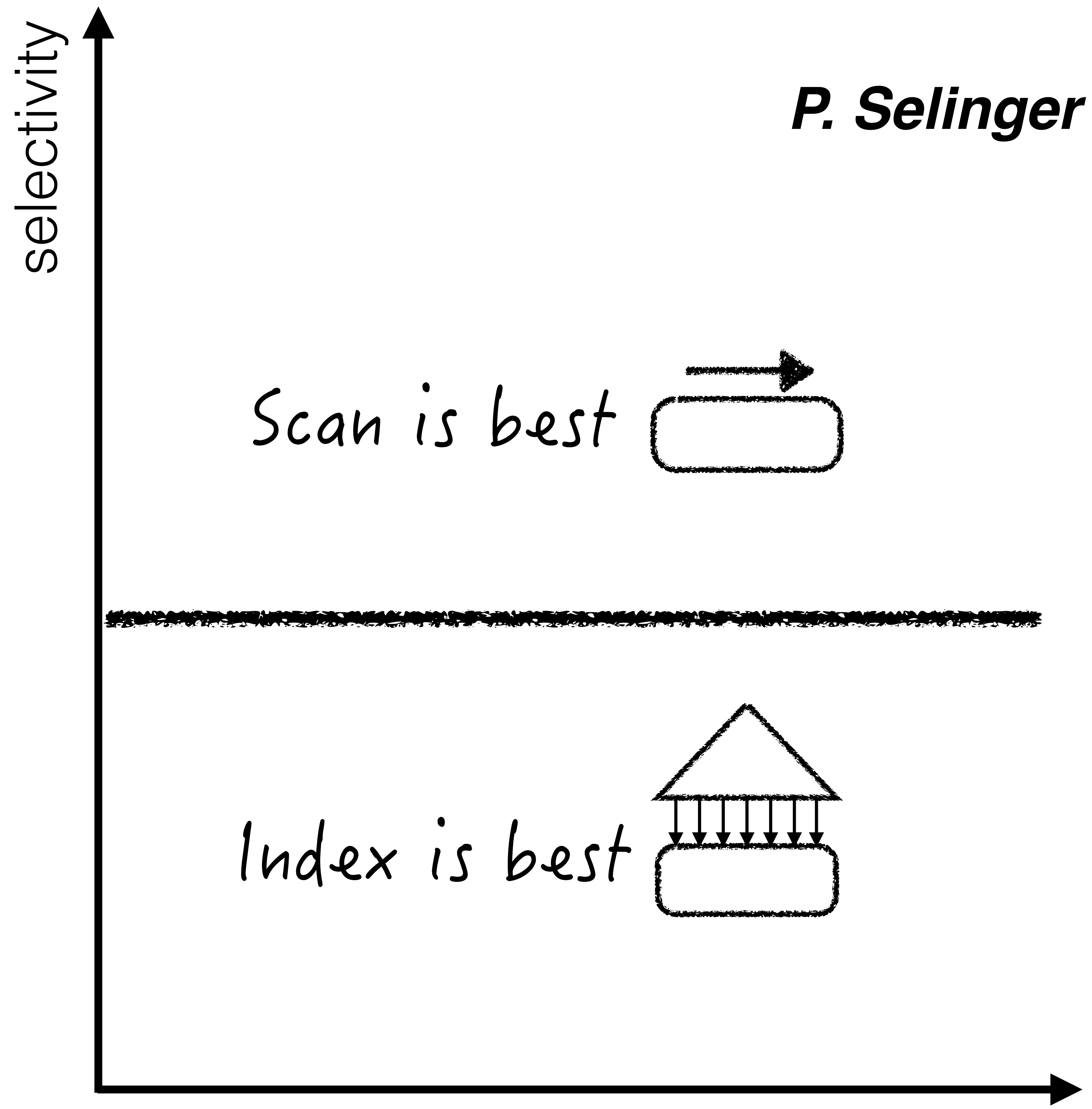


ACCESS PATH SELECTION

scan vs secondary index selection



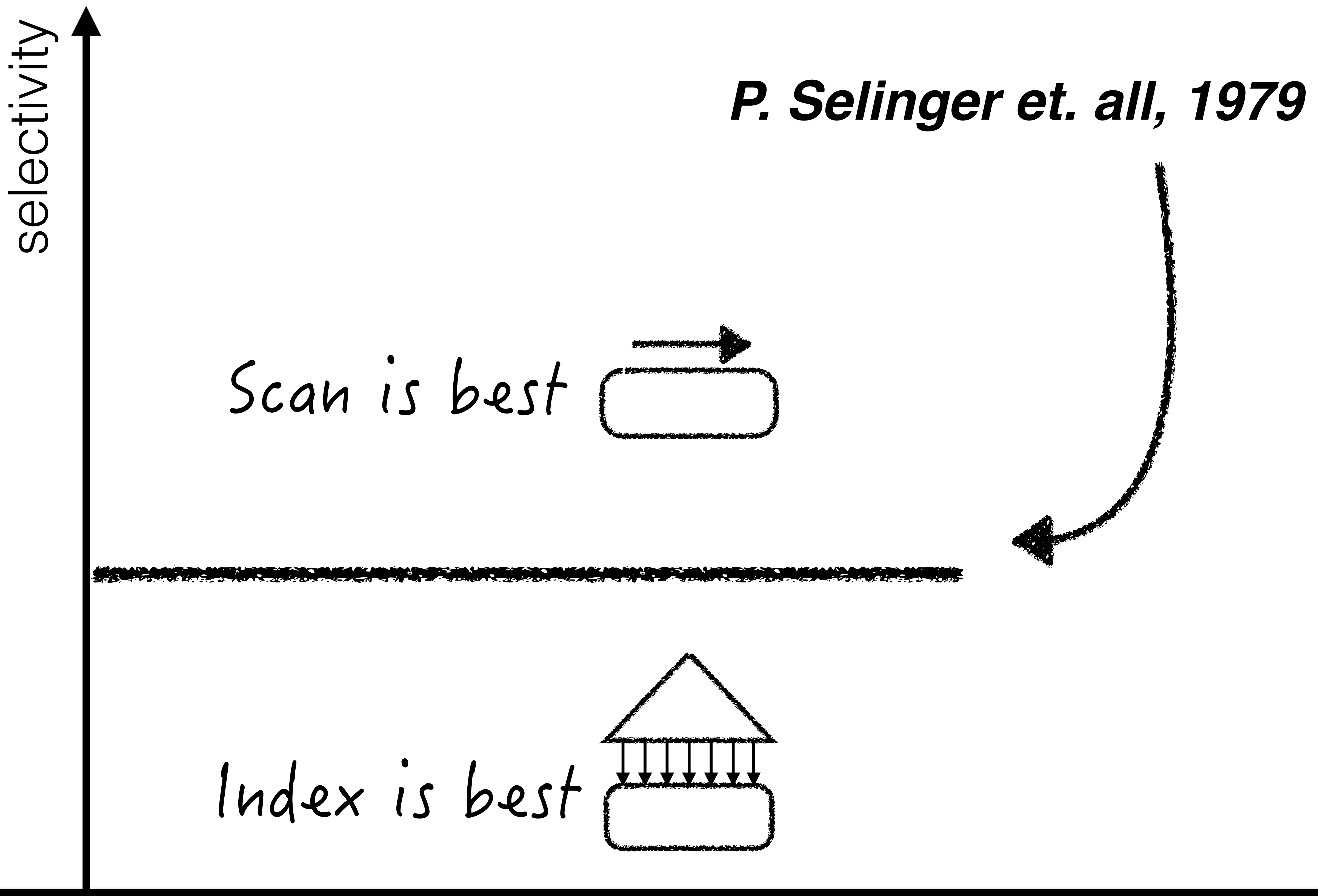
ACCESS PATH SELECTION
scan vs secondary index selection



DASlab @ Harvard SEAS

ACCESS PATH SELECTION

scan vs secondary index selection

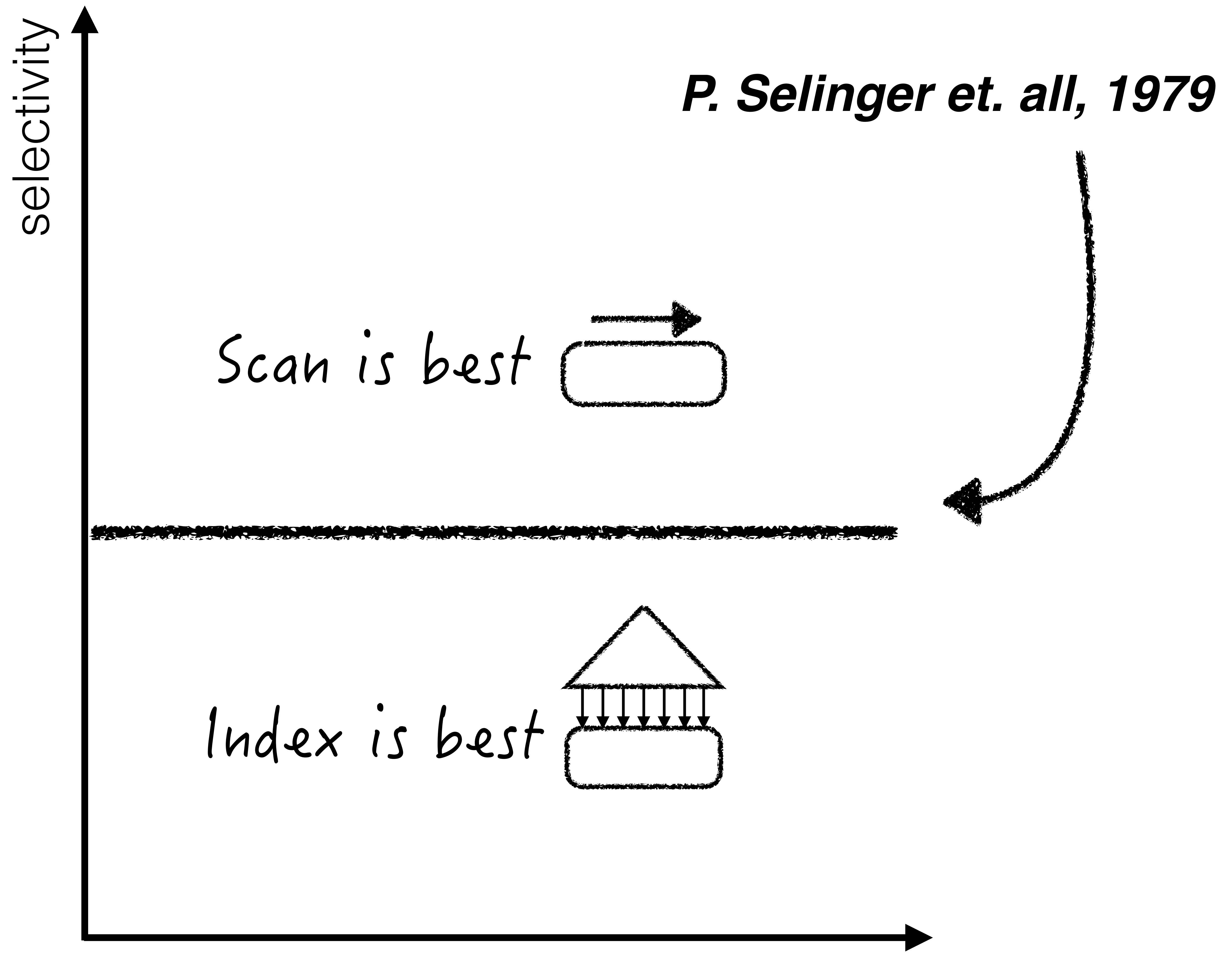


DO WE STILL NEED INDEXING? (AND IF YES HOW DO WE CHOOSE)

ACCESS PATH SELECTION in ANALYTICAL SYSTEMS

scan vs secondary index selection

@SIGMOD 2017

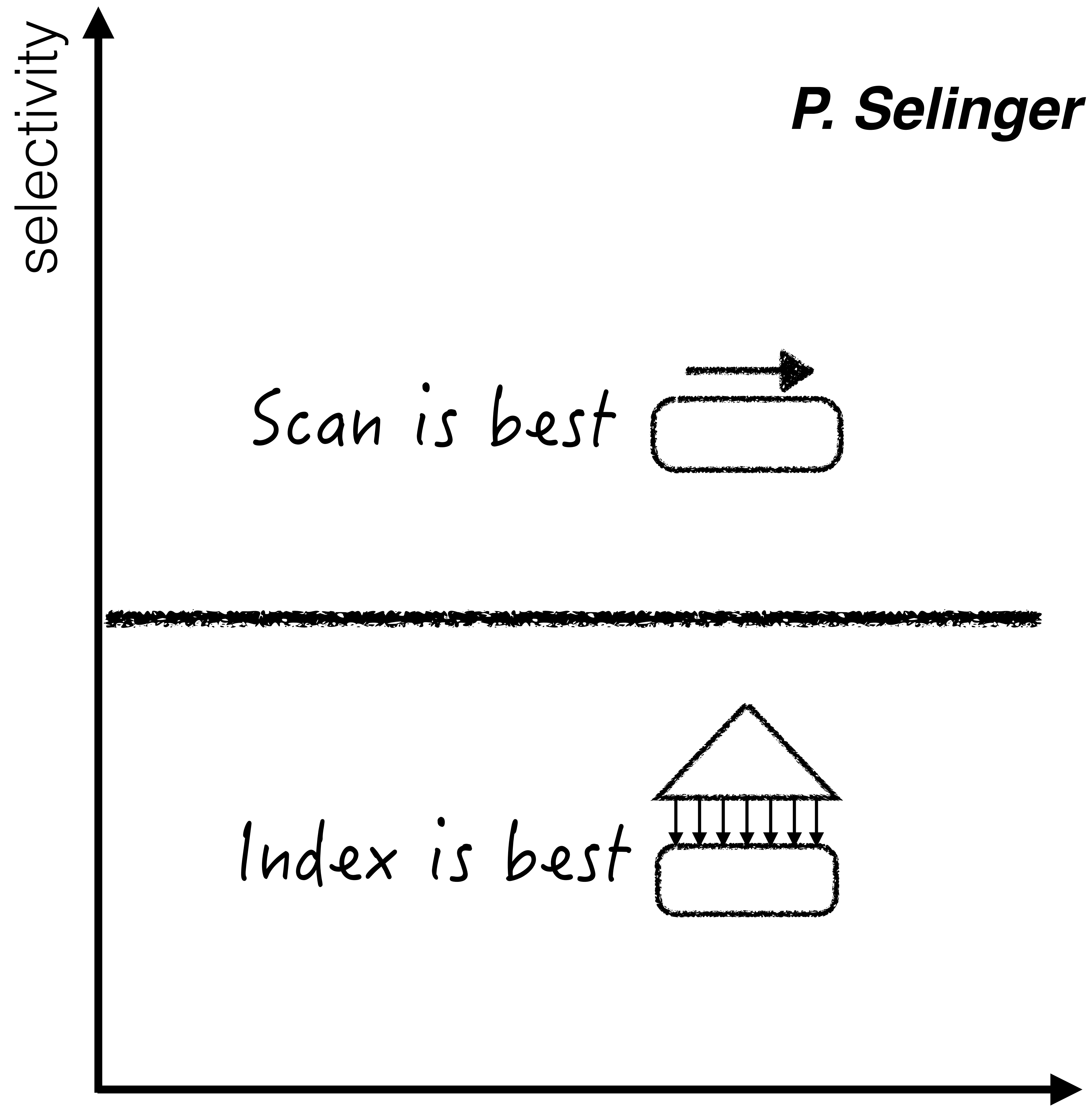


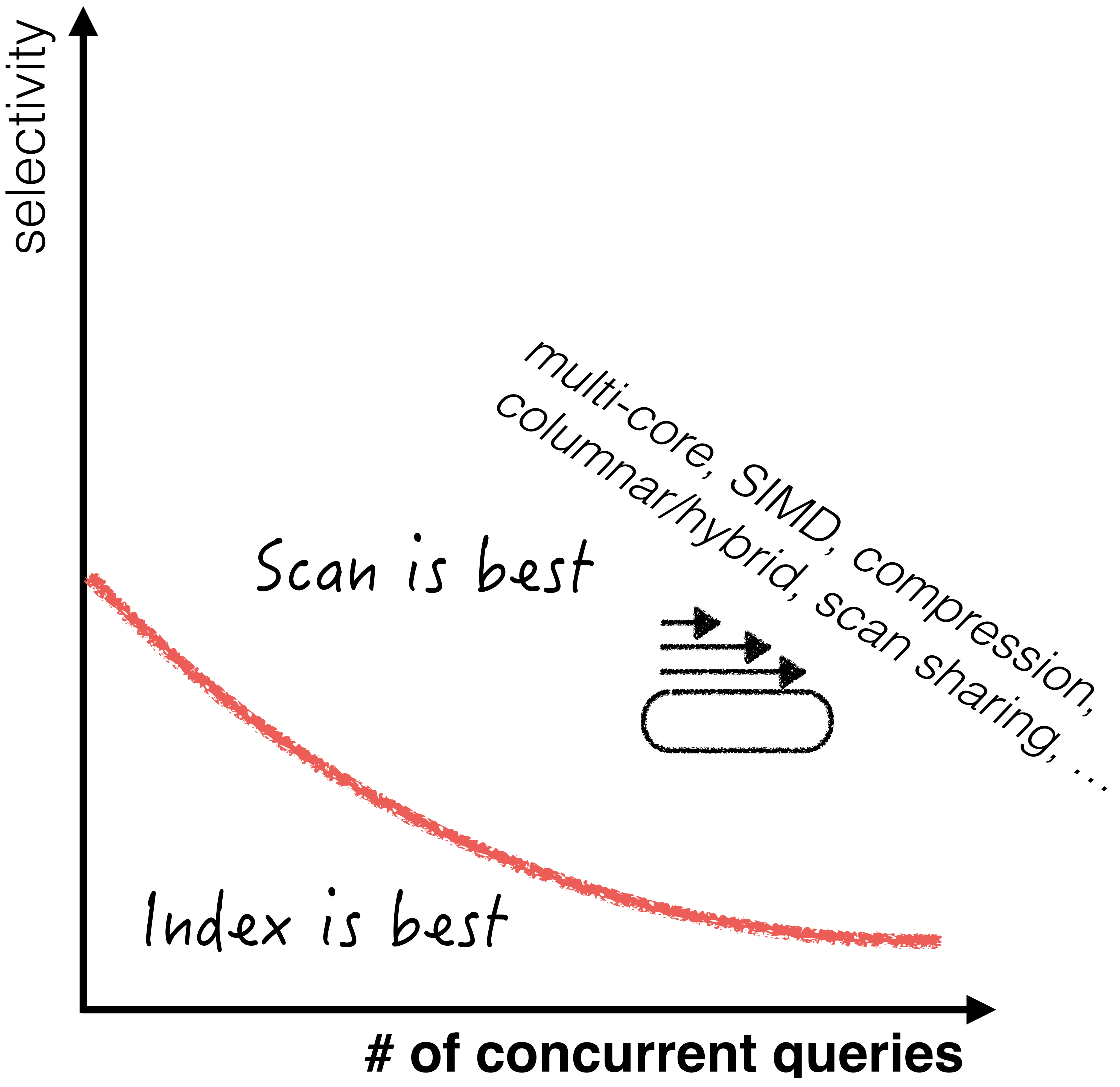
DASlab @ Harvard SEAS

ACCESS PATH SELECTION in ANALYTICAL SYSTEMS

scan vs secondary index selection

@SIGMOD 2017

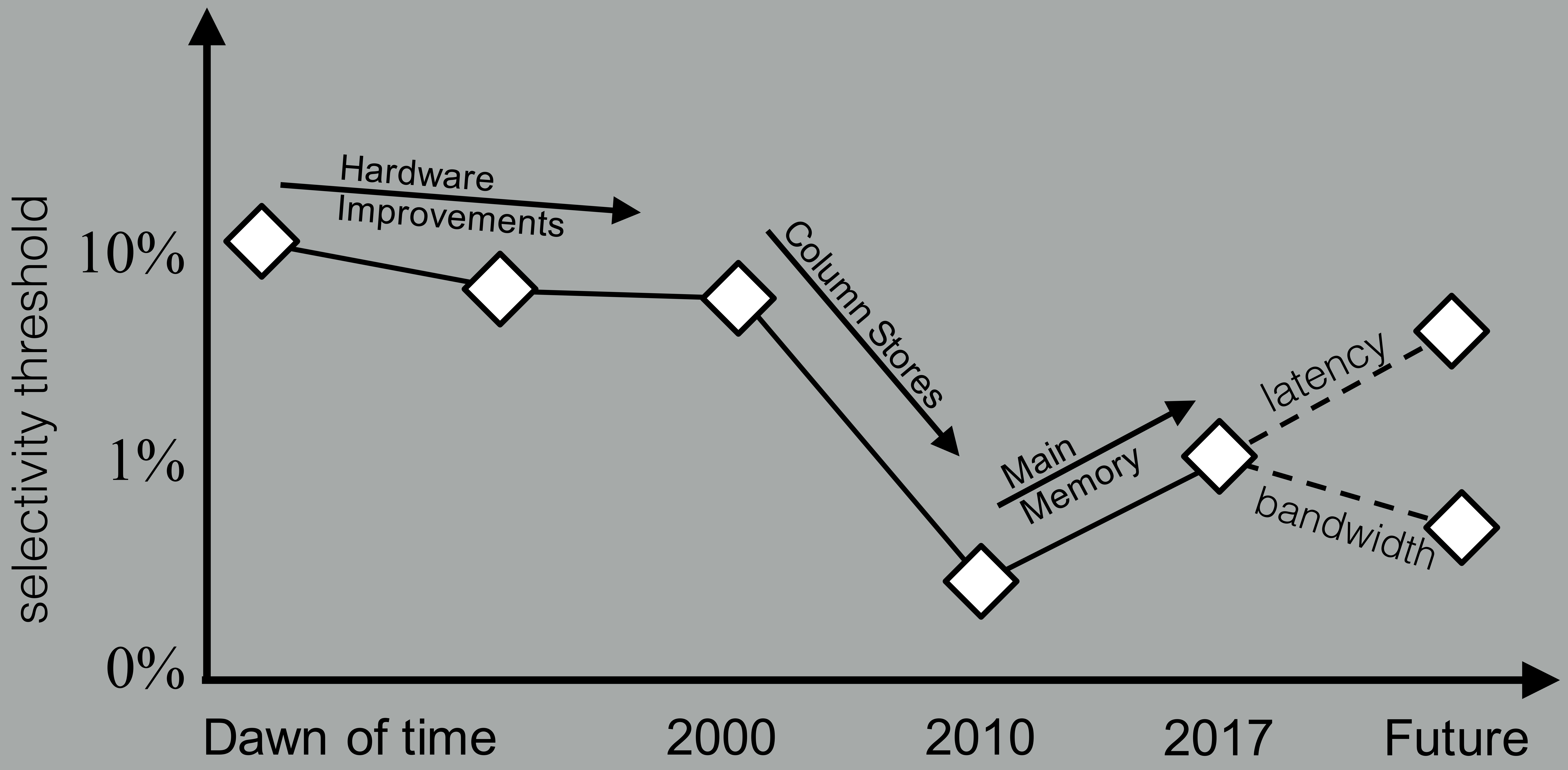




ACCESS PATH SELECTION in ANALYTICAL SYSTEMS

scan vs secondary index selection

@SIGMOD 2017

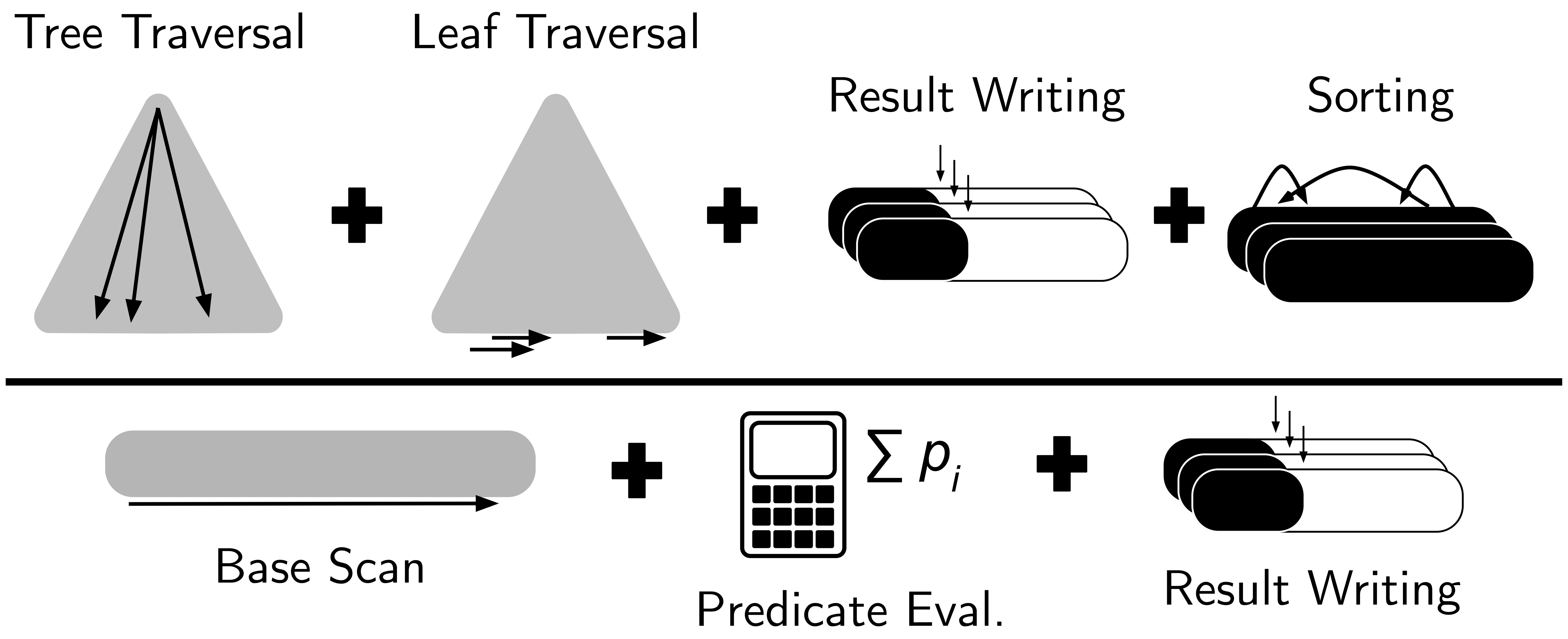


$$APS(q,S_{tot}) = \frac{q \cdot \frac{1+\lceil log_b(N) \rceil}{N} \cdot \left(BW_S \cdot C_M + \frac{b \cdot BW_S \cdot C_A}{2} + \frac{b \cdot BW_S \cdot f_p \cdot p}{2} \right)}{max\left(ts, 2 \cdot f_p \cdot p \cdot q \cdot BW_S\right) + S_{tot} \cdot rw \cdot \frac{BW_S}{BW_R}}$$

$$+ \frac{S_{tot} \left(\frac{BW_S \cdot C_M}{b} + (aw + ow) \cdot \frac{BW_S}{BW_I} + rw \cdot \frac{BW_S}{BW_R} \right)}{max\left(ts, 2 \cdot f_p \cdot p \cdot q \cdot BW_S\right) + S_{tot} \cdot rw \cdot \frac{BW_S}{BW_R}}$$

$$+ \frac{S_{tot} \cdot log_2\left(S_{tot} \cdot N\right) \cdot BW_S \cdot C_A}{max\left(ts, 2 \cdot f_p \cdot p \cdot q \cdot BW_S\right) + S_{tot} \cdot rw \cdot \frac{BW_S}{BW_R}}$$

scan vs secondary index selection @SIGMOD 2017

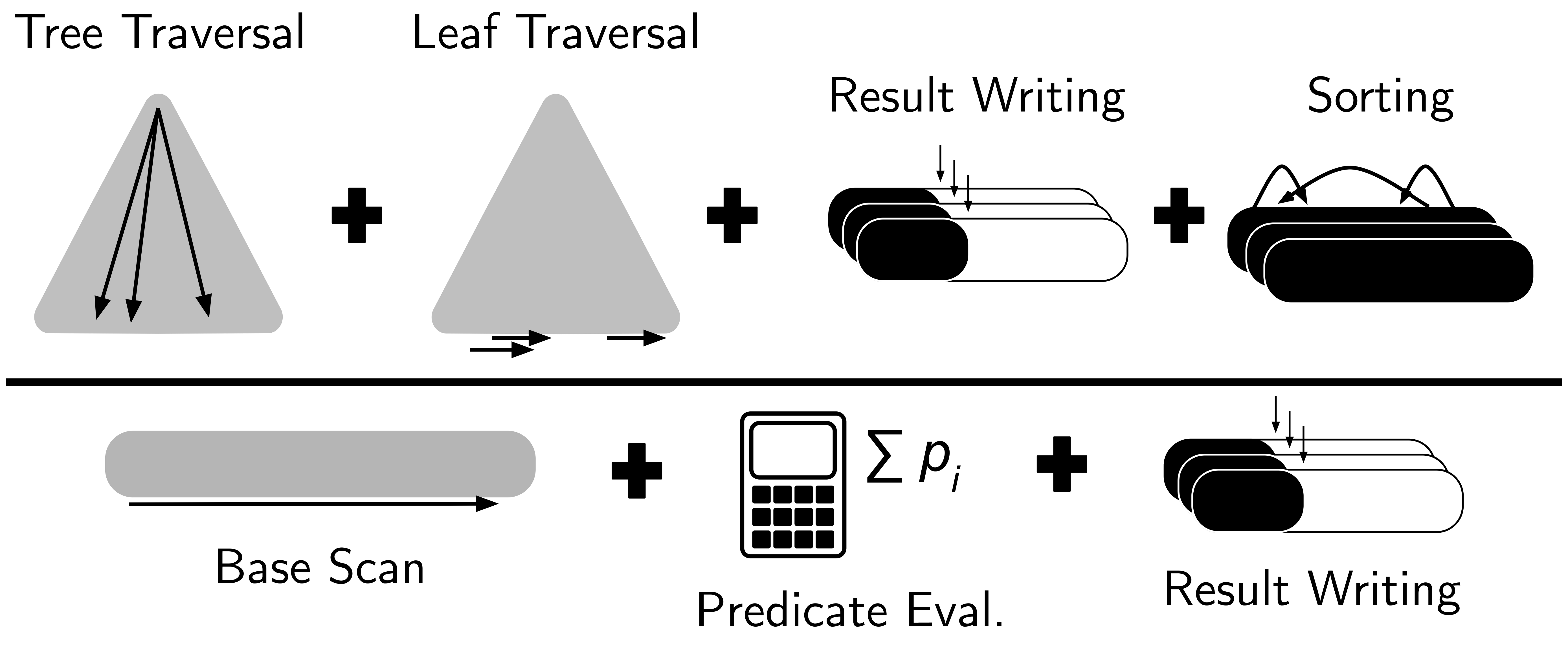


Workload	q	number of queries
	s_i	selectivity of query i
	S_{tot}	total selectivity of the workload
Dataset	N	data size (tuples per column)
	ts	tuple size (bytes per tuple)
Hardware	C_A	L1 cache access (sec)
	C_M	LLC miss: memory access (sec)
	BW_S	scanning bandwidth (GB/s)
	BW_R	result writing bandwidth (GB/s)
	BW_I	leaf traversal bandwidth (GB/s)
	p	The inverse of CPU frequency
	f_p	Factor accounting for pipelining
Scan & Index	rw	result width (bytes per output tuple)
	b	tree fanout
	aw	attribute width (bytes of the indexed column)
	ow	offset width (bytes of the index column offset)

HARD & SLOW

$$APS(q, S_{tot}) = \frac{q \cdot \frac{1+\lceil log_b(N) \rceil}{N} \cdot \left(BW_S \cdot C_M + \frac{b \cdot BW_S \cdot C_A}{2} + \frac{b \cdot BW_S \cdot f_p \cdot p}{2} \right)}{max\left(ts, 2 \cdot f_p \cdot p \cdot q \cdot BW_S\right) + S_{tot} \cdot rw \cdot \frac{BW_S}{BW_R}}$$
$$+ \frac{S_{tot} \left(\frac{BW_S \cdot C_M}{b} + (aw + ow) \cdot \frac{BW_S}{BW_I} + rw \cdot \frac{BW_S}{BW_R} \right)}{max\left(ts, 2 \cdot f_p \cdot p \cdot q \cdot BW_S\right) + S_{tot} \cdot rw \cdot \frac{BW_S}{BW_R}}$$
$$+ \frac{S_{tot} \cdot log_2\left(S_{tot} \cdot N\right) \cdot BW_S \cdot C_A}{max\left(ts, 2 \cdot f_p \cdot p \cdot q \cdot BW_S\right) + S_{tot} \cdot rw \cdot \frac{BW_S}{BW_R}}$$

scan vs secondary index selection @SIGMOD 2017



Workload	q	number of queries
	s_i	selectivity of query i
	S_{tot}	total selectivity of the workload
Dataset	N	data size (tuples per column)
	ts	tuple size (bytes per tuple)
Hardware	C_A	L1 cache access (sec)
	C_M	LLC miss: memory access (sec)
	BW_S	scanning bandwidth (GB/s)
	BW_R	result writing bandwidth (GB/s)
	BW_I	leaf traversal bandwidth (GB/s)
	p	The inverse of CPU frequency
	f_p	Factor accounting for pipelining
Scan & Index	rw	result width (bytes per output tuple)
	b	tree fanout
	aw	attribute width (bytes of the indexed column)
	ow	offset width (bytes of the index column offset)

Access Path Selection in Main-Memory Optimized Data Systems: Should I Scan or Should I Probe? Michael Kester, Manos Athanassoulis, Stratos Idreos. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 2017

S. BING YAO

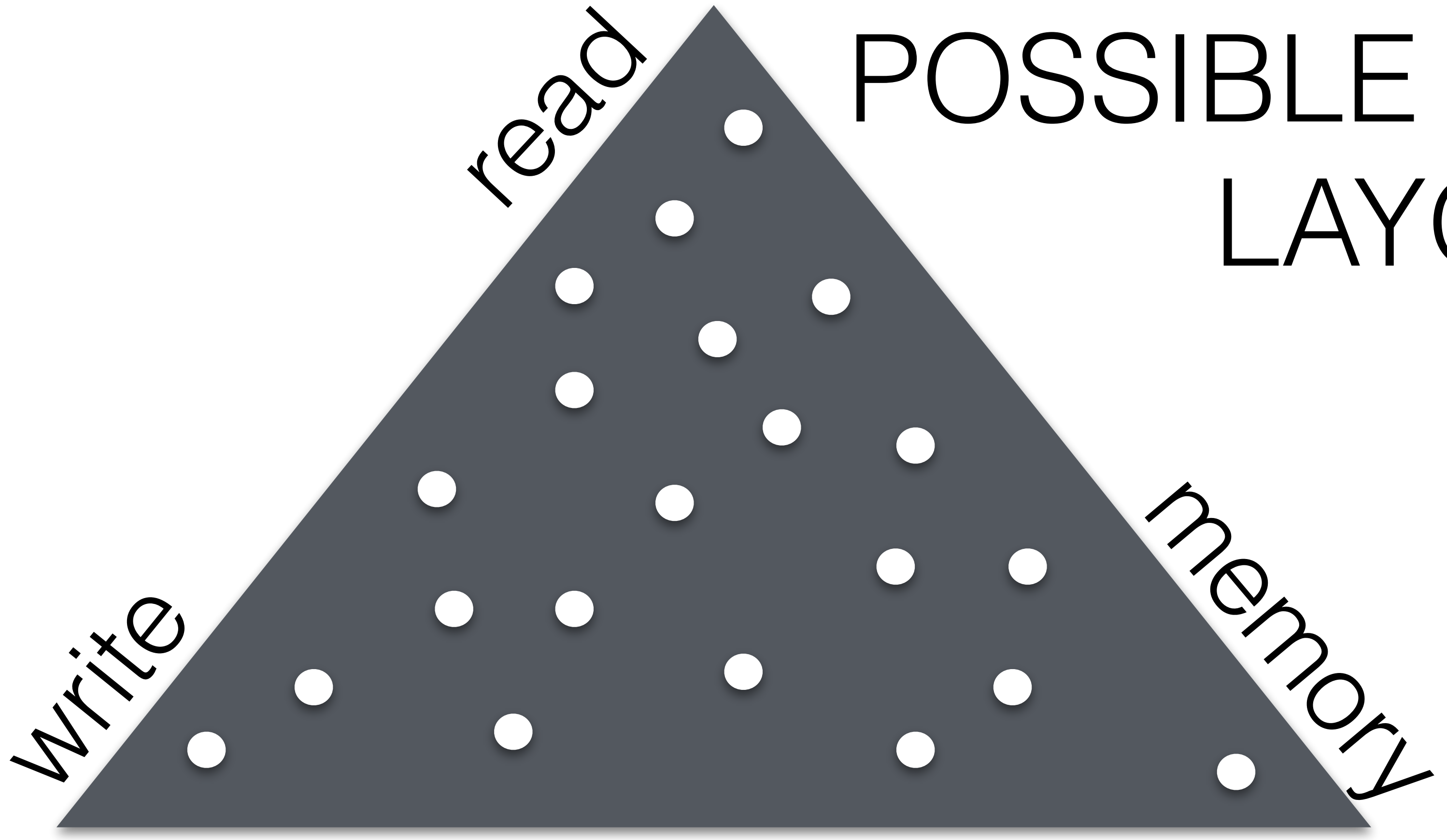
models/advisors

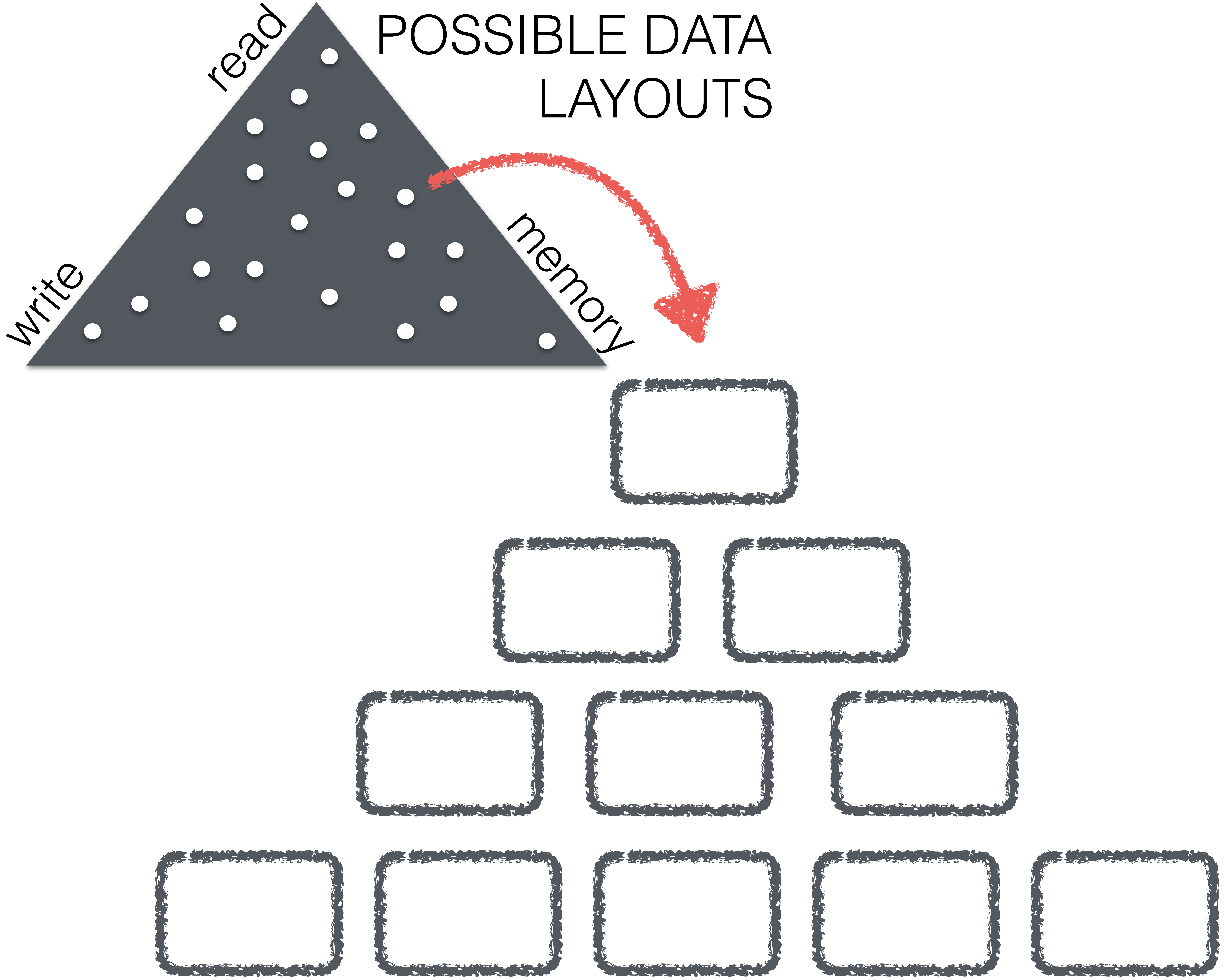
STEFAN MANEGOLD

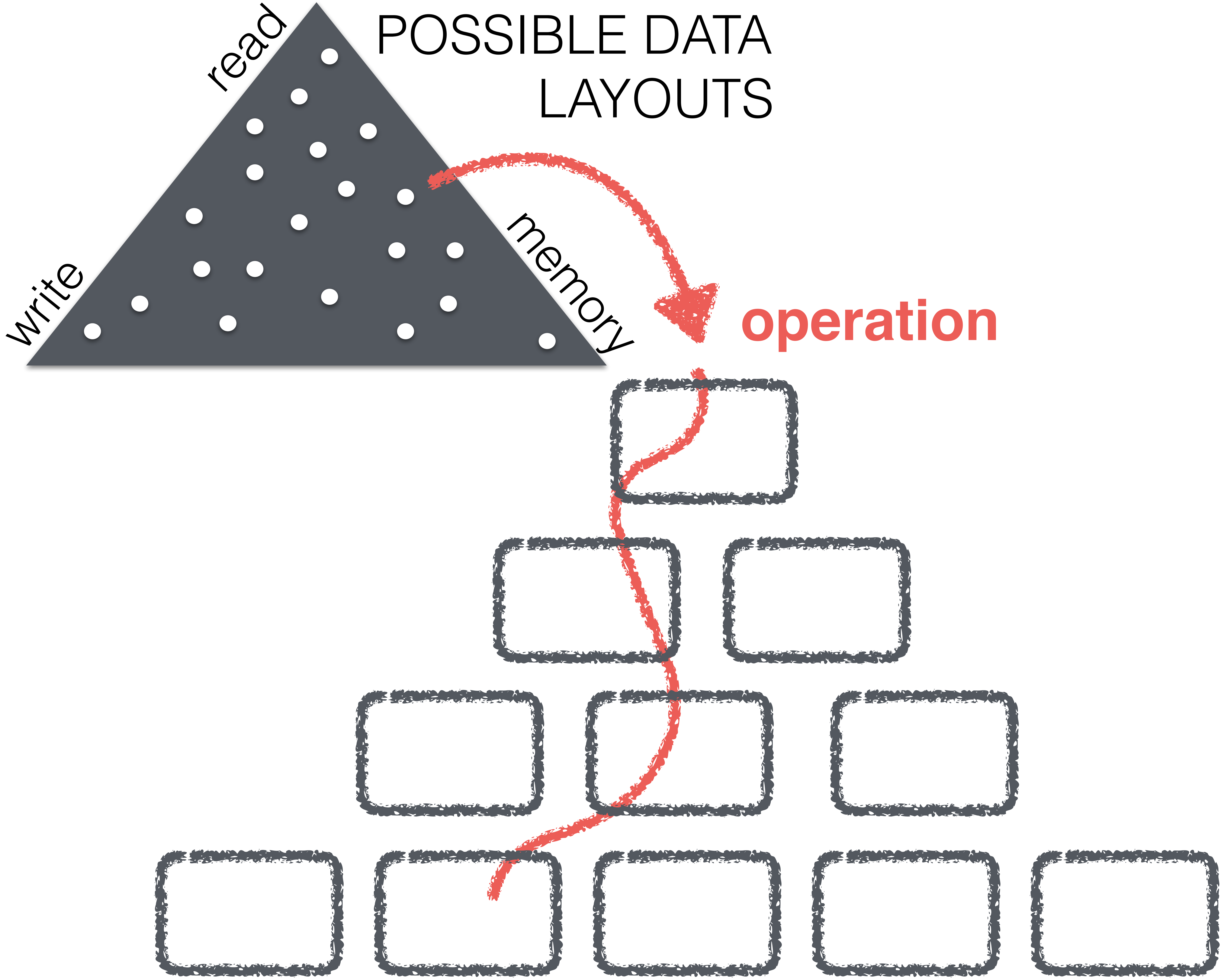
model synthesis

We need something else: Something more scalable & robust!

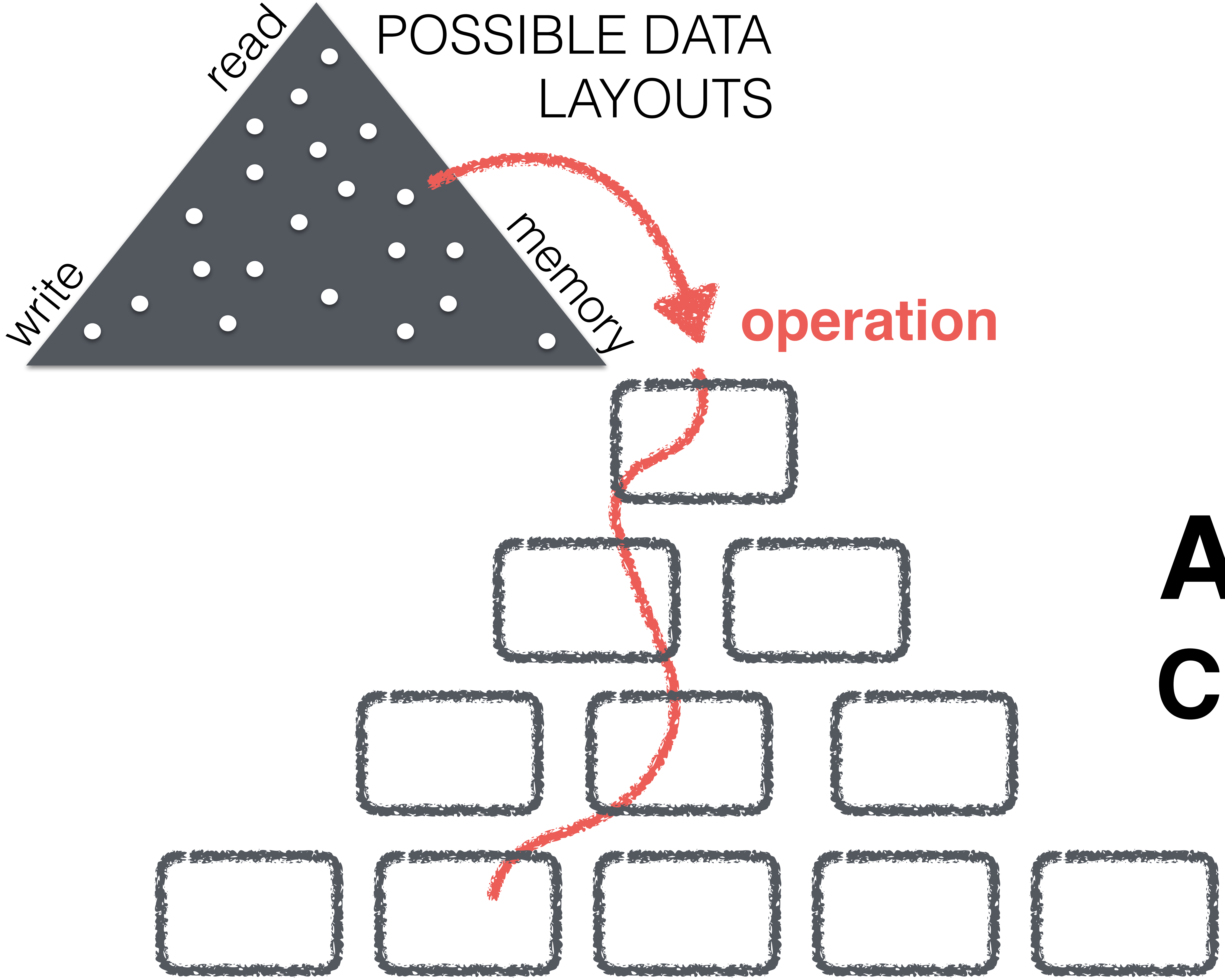
POSSIBLE DATA LAYOUTS



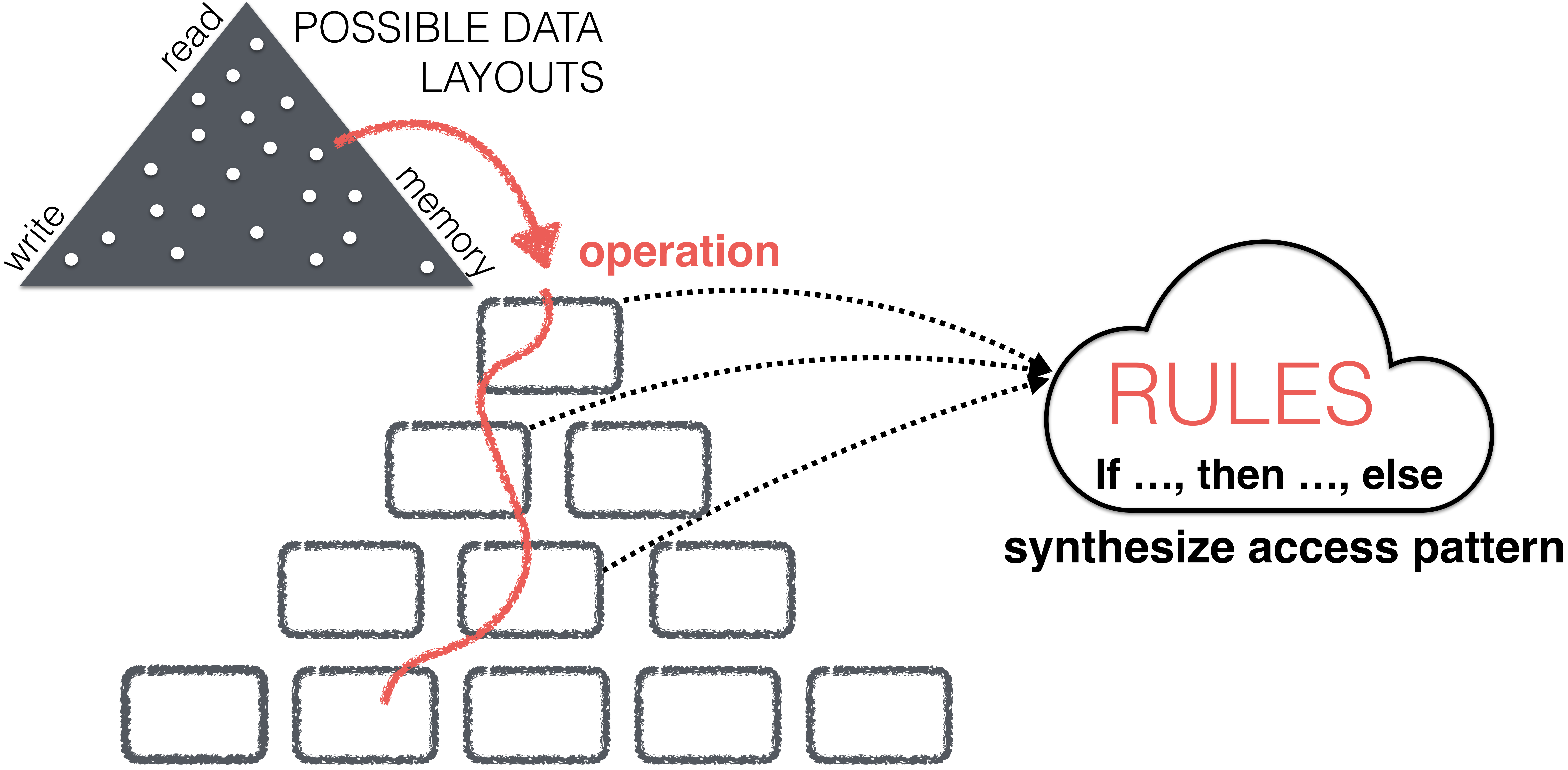

POSSIBLE DATA
LAYOUTS
read
write
memory



DASlab @ Harvard SEAS



ALGORITHM & COST SYNTHESIS

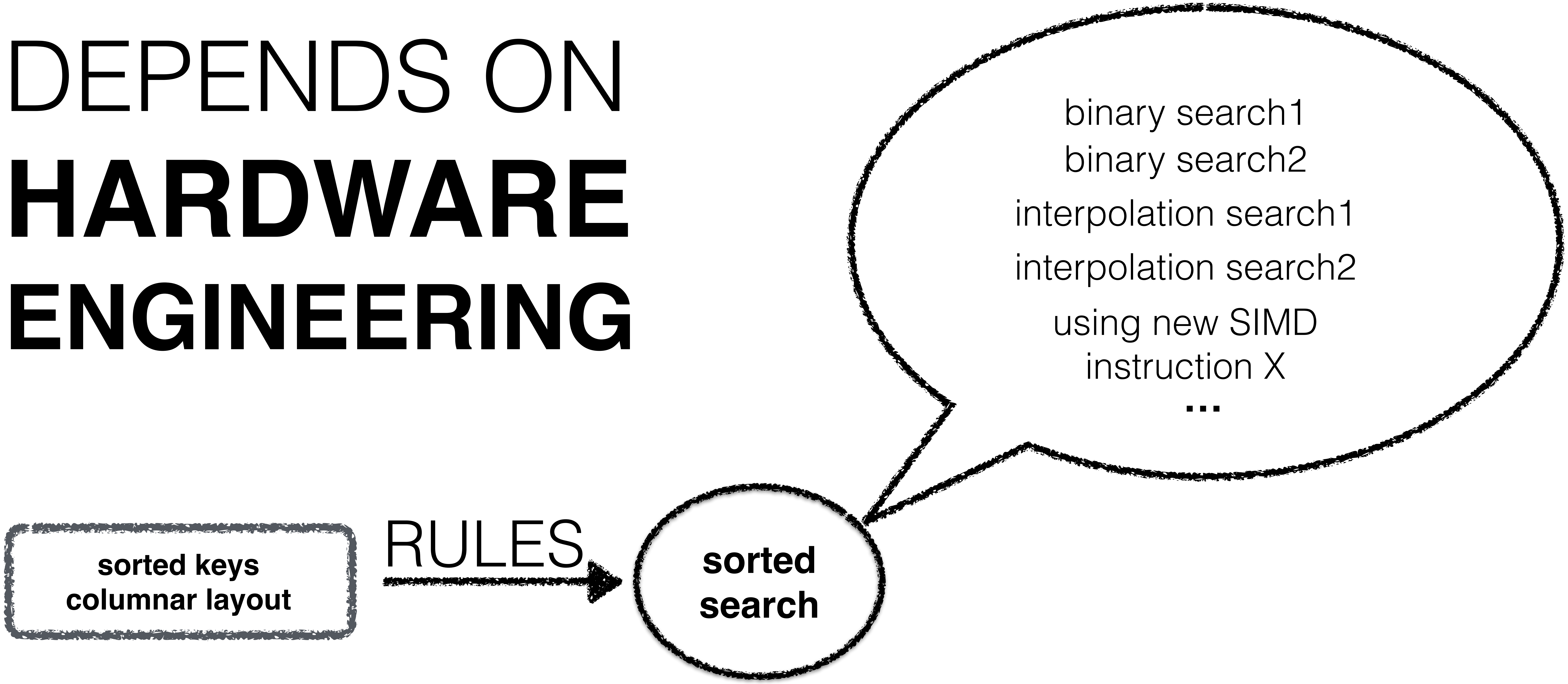


DASlab
@ Harvard SEAS

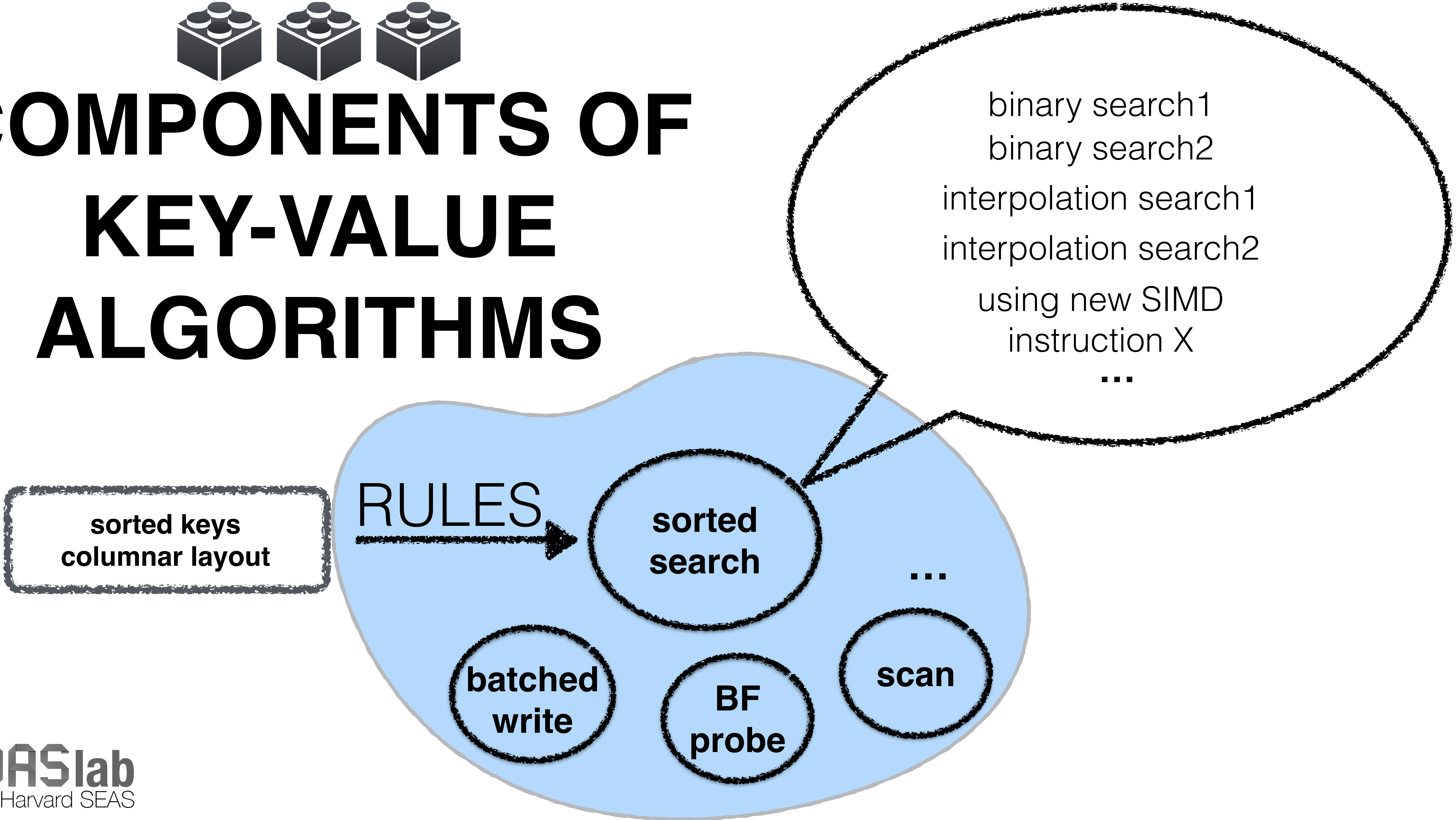
sorted keys
columnar layout


sorted keys
columnar layout
RULES
sorted
search

DEPENDS ON **HARDWARE ENGINEERING**

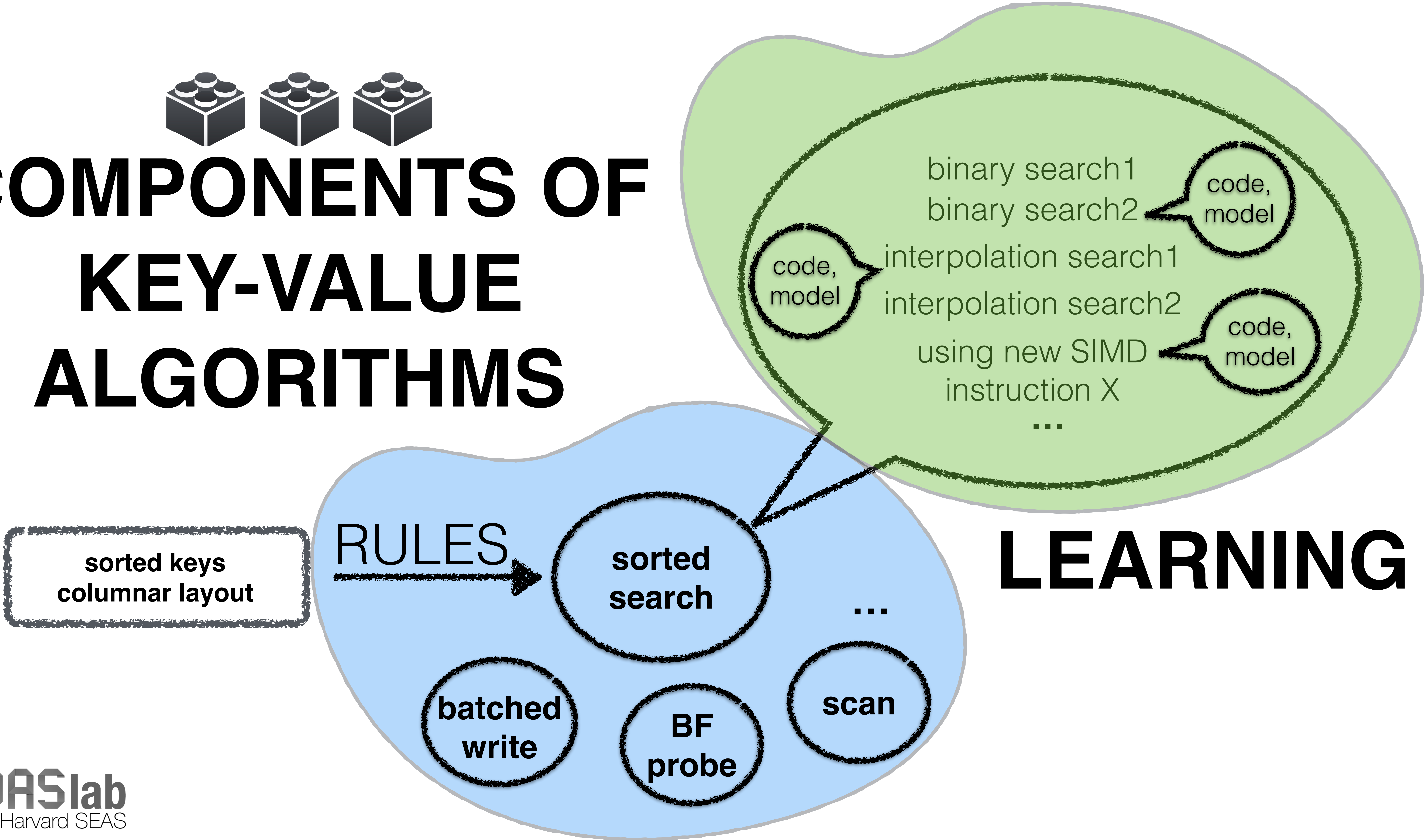


COMPONENTS OF KEY-VALUE ALGORITHMS



DASlab @ Harvard SEAS

COMPONENTS OF KEY-VALUE ALGORITHMS



DASlab @ Harvard SEAS

SYNTHESIS FROM LEARNED MODELS

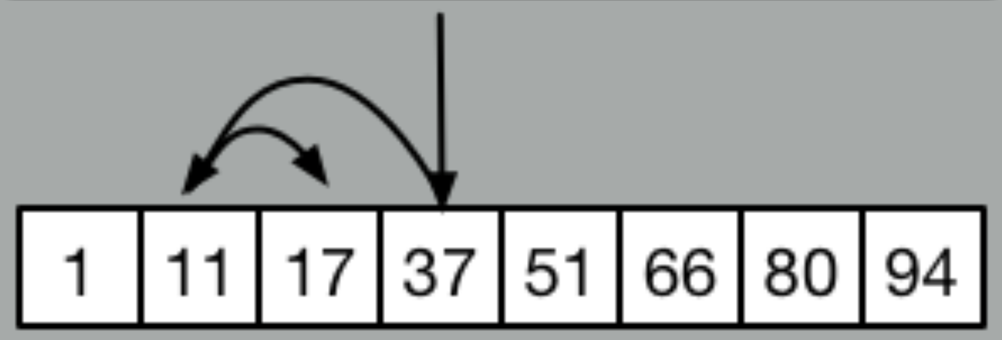
coding, modeling, generalized models, and a touch of ML

1. MINIMAL CODE

e.g., binary search

C++

```
if (data[middle] < search_val) {
    low = middle + 1;
} else {
    high = middle;
}
middle = (low + high)/2;
```



SYNTHESIS FROM LEARNED MODELS

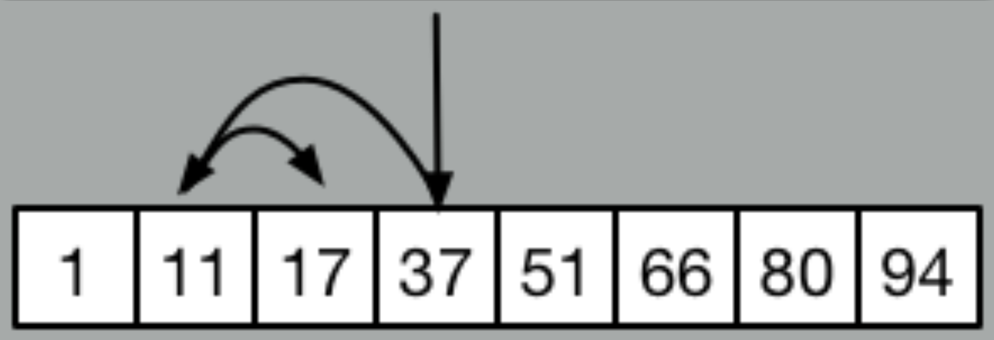
coding, modeling, generalized models, and a touch of ML

1. MINIMAL CODE

e.g., binary search

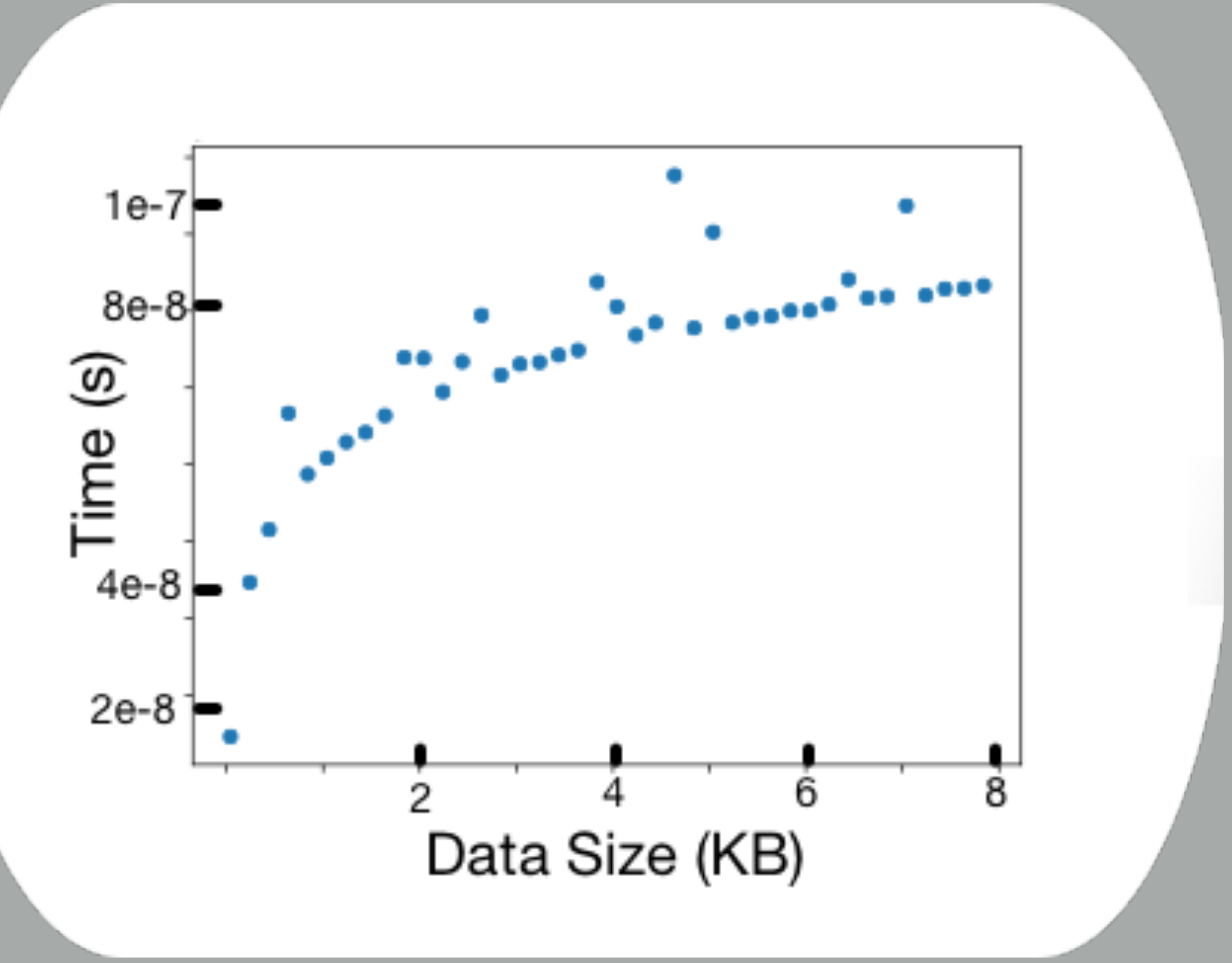
C++

```cpp
if (data[middle] < search_val) {
    low = middle + 1;
} else {
    high = middle;
}
middle = (low + high)/2;
```



Run

2. BENCHMARK



SYNTHESIS FROM LEARNED MODELS

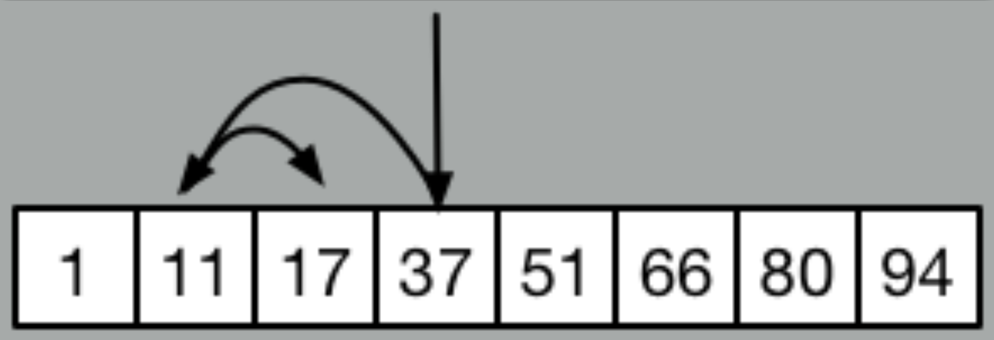
coding, modeling, generalized models, and a touch of ML

1. MINIMAL CODE

e.g., binary search

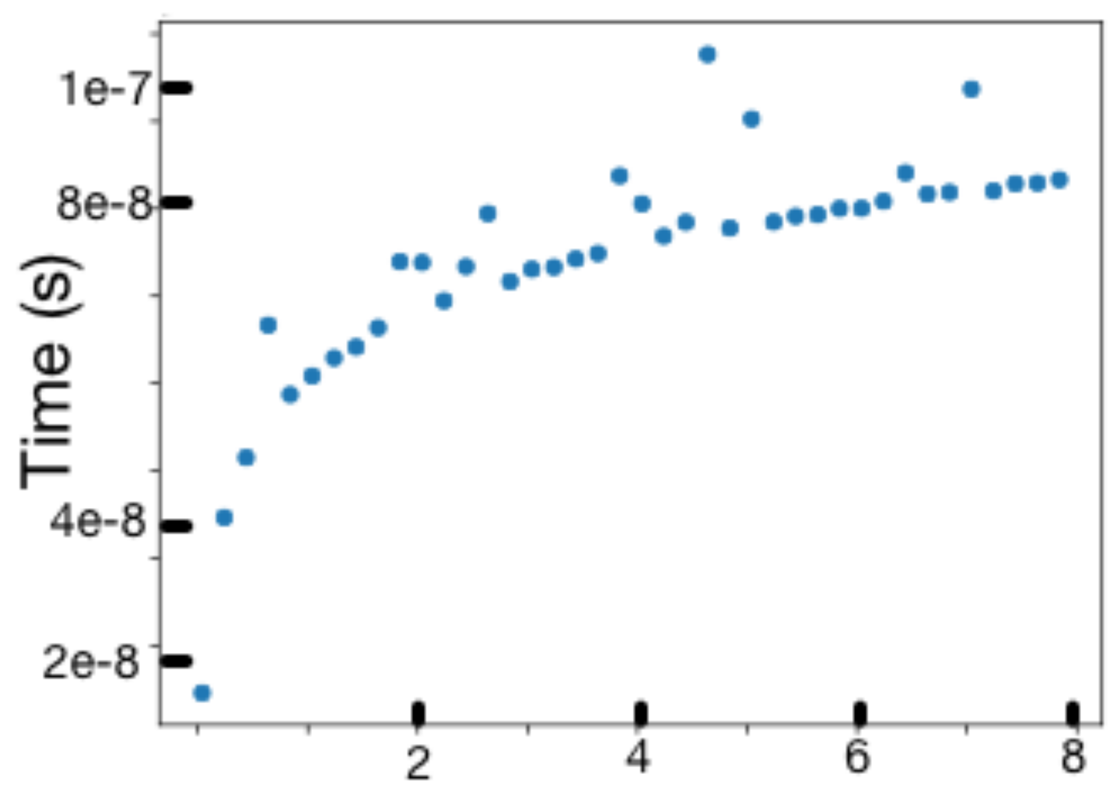
C++

```
if (data[middle] < search_val) {
    low = middle + 1;
} else {
    high = middle;
}
middle = (low + high)/2;
```



Run

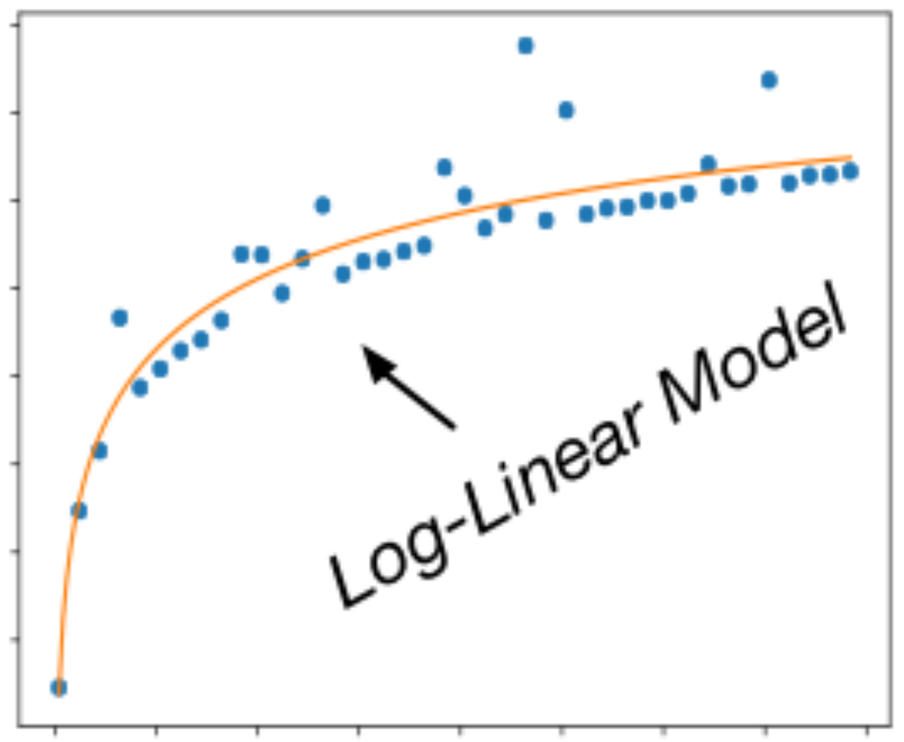
2. BENCHMARK



f(x)

Train

3. FIT MODEL



$$f(x) = ax + b \log x + c$$

DASlab @ Harvard SEAS

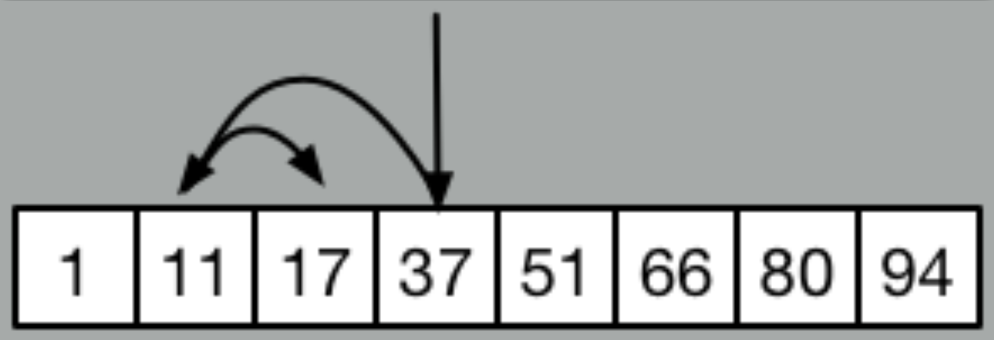
SYNTHESIS FROM LEARNED MODELS

coding, modeling, generalized models, and a touch of ML

1. MINIMAL CODE

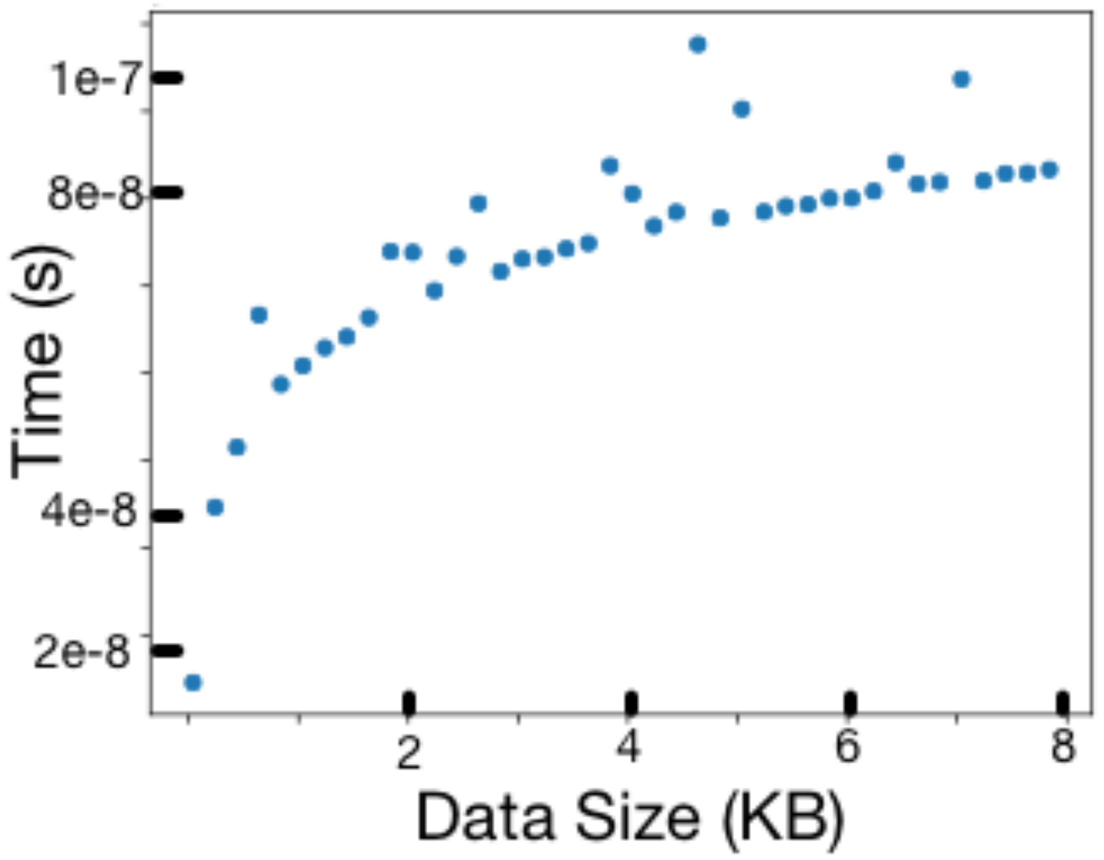
e.g., binary search

```
if (data[middle] < search_val) {
    low = middle + 1;
} else {
    high = middle;
}
middle = (low + high)/2;
```



Run

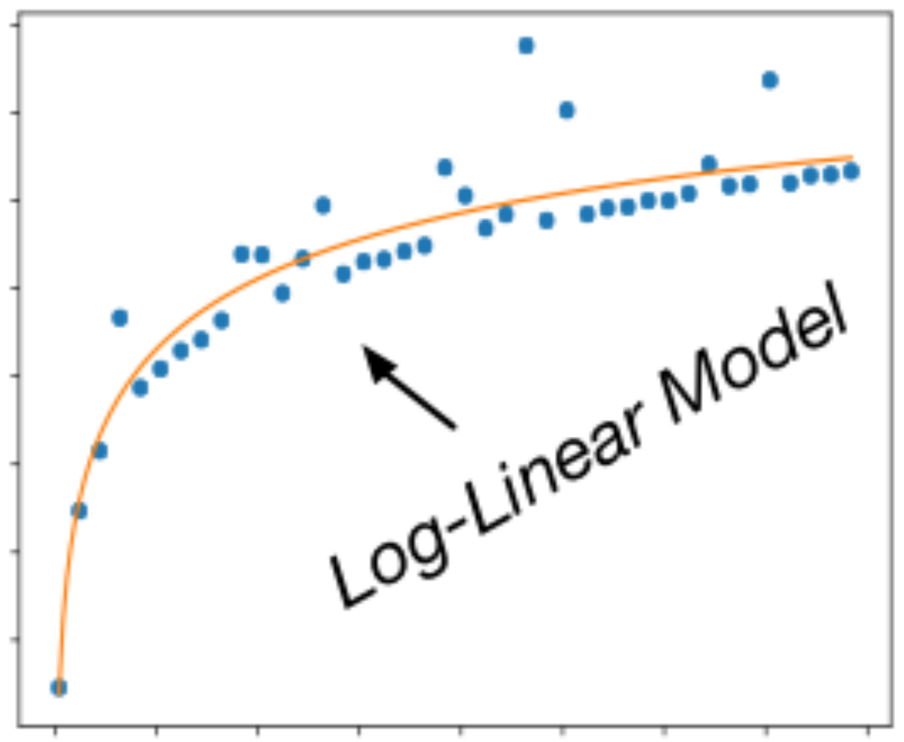
2. BENCHMARK



$f(x)$

Train

3. FIT MODEL



$$f(x) = ax + b\log x + c$$

FOLDING ALGORITHMIC, ENGINEERING, AND H/W, PROPERTIES INTO THE COEFFICIENTS

RULE/MODEL BASED SYSTEM SYNTHESIZES ALGORITHM AND COST

CS 265

Stratos Idreos

BIG DATA SYSTEMS

NoSQL | Neural Networks | Image AI | LLMs | Data Science