Learned Indexing and Sampling for Improving Query Performance in Big-Data Analytics

Kexin Rong

VMware Research Group | Georgia Tech SCS

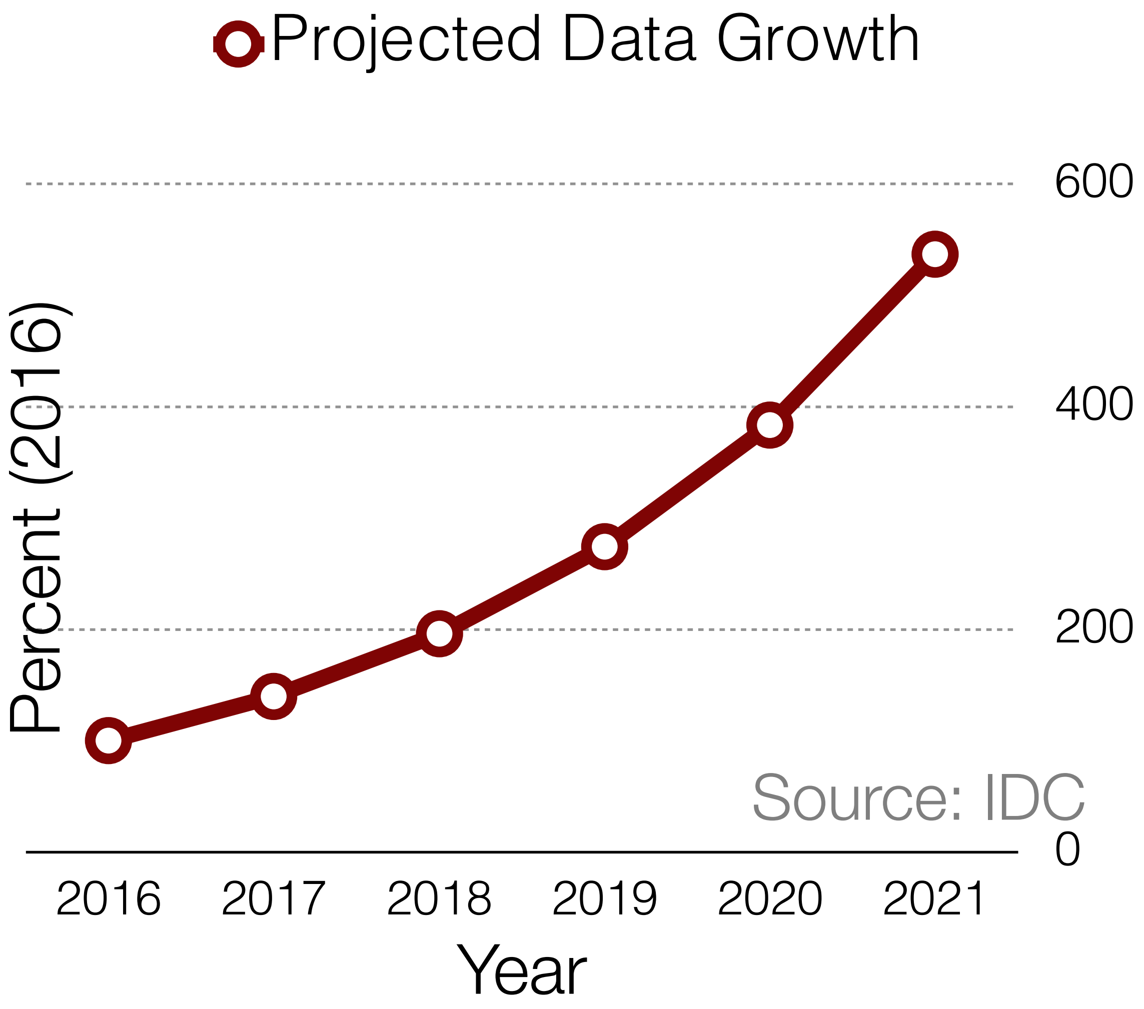
Stanford MLSys Seminar

04/14/22

vmware®

Georgia Tech | School of Computer Science
College of Computing

Data is growing exponentially



Increased automated processes (e.g., sensors, devices) to collect data

Reduced storage costs due to Big Data systems (e.g., HDFS, S3), cloud

Data partition as a basic unit for storage

Data partition as a basic unit for storage

host	metric	time
s1	3	2015
s2	5	2015
s1	500	2016
s3	100	2017
s1	50	2019
…	…	…

Partition 1

host	metric	time
s1	3	2015
s2	5	2015
…		

128 MB

Partition 2

host	metric	time
s1	500	2016
s3	100	2017
…		

128 MB

…

Partition N

host	metric	time
s1	50	2019
…		

128 MB

- many rows

Data partition as a basic unit for I/O

host	metric	time
s1	3	2015
s2	5	2015
s1	500	2016
s3	100	2017
s1	50	2019
…	…	…

Partition 1

host	metric	time
s1	3	2015
s2	5	2015
…		

128 MB

Partition 2

host	metric	time
s1	500	2016
s3	100	2017
…		

128 MB

…

Partition N

host	metric	time
s1	50	2019
…		

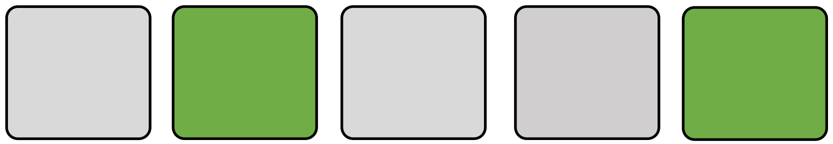
128 MB

- many rows
- columnar compression
- files on disk/cloud

How to process SQL queries efficiently?

Two classic ideas:

#1 Sampling



#2 Indexing

Before: row-level sampling for approximation

host	metric
server1	3
server1	5
server2	50000
server3	100
server3	50
…	

→

host	metric
server1	3
server1	5
server2	50000
server3	100
server3	50
…	

→

Aggregate Query

```
SELECT
  SUM(metric)
GROUP BY host
```

Now: row-level sampling is expensive

host	metric
server1	3
server1	5
server2	50000
server3	100
server3	50
…	

Partition 1

host	metric
server1	3
server1	5
…	

128 MB

Partition 2

host	metric
server1	10
server2	50000
…	

128 MB

…

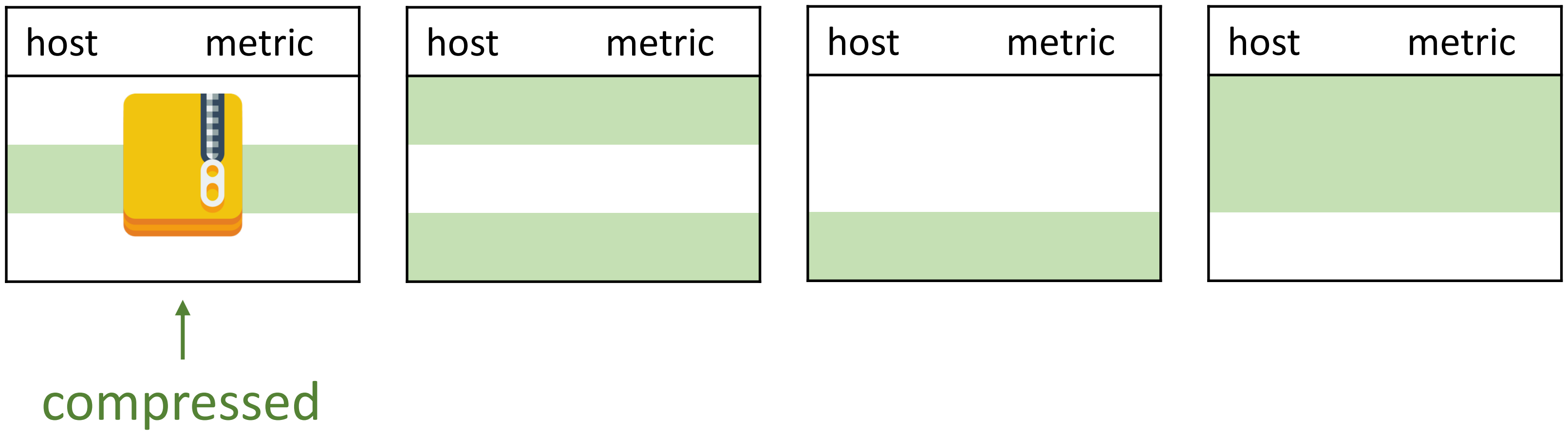
Partition N

host	metric
server3	100
server3	50
…	

128 MB

Now: row-level sampling is expensive

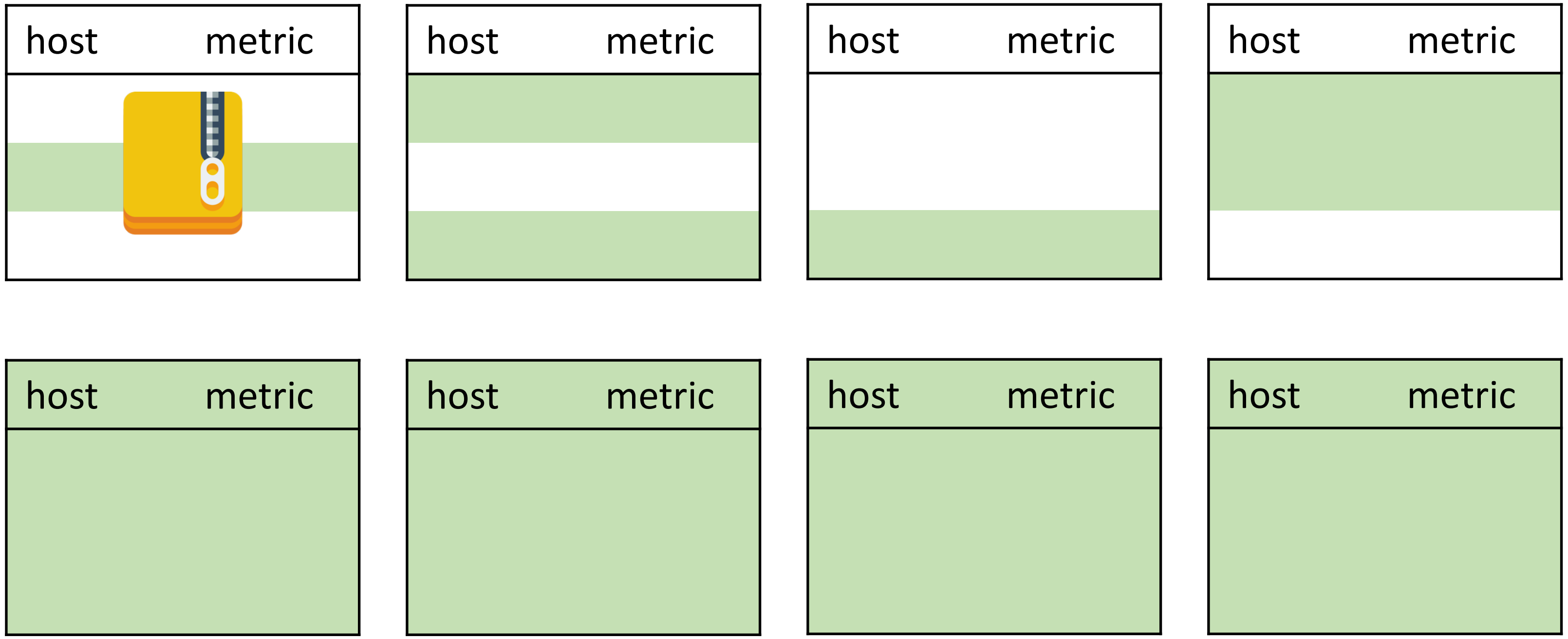
row samples



Now: row-level sampling is expensive

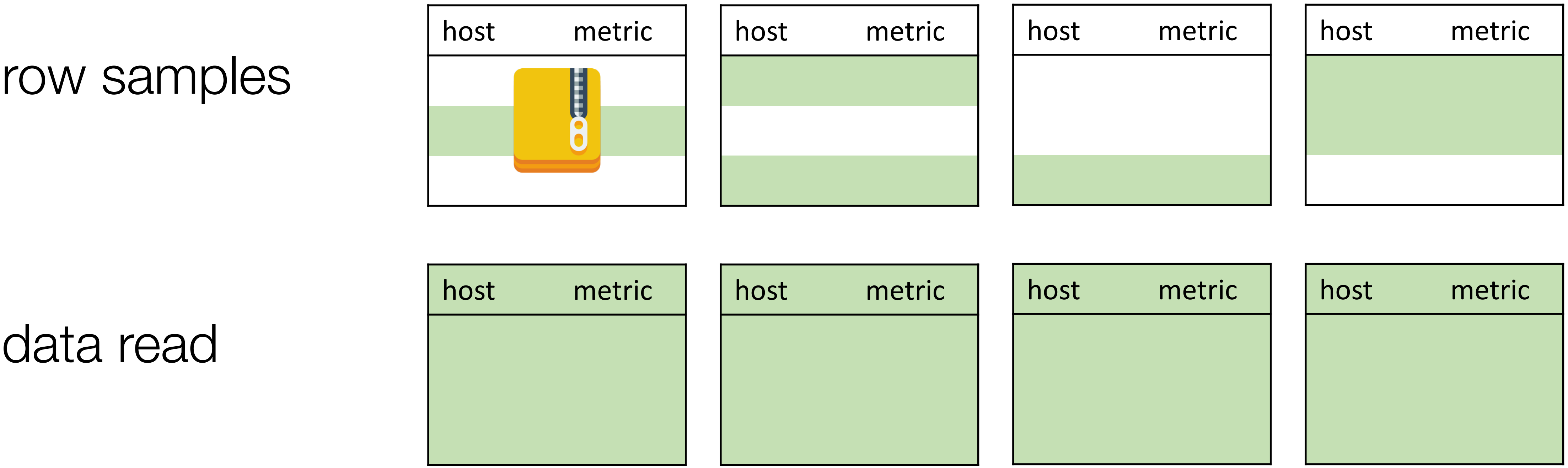
row samples

data read



Sampling one row => Reading one partition

Now: row-level sampling is expensive

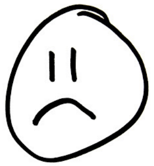


Suppose each partition has 100 rows:

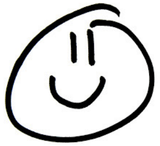
- 1% row sample => ~64% ($1-0.99^{100}$) of the partitions
- 10% row sample => almost every partition

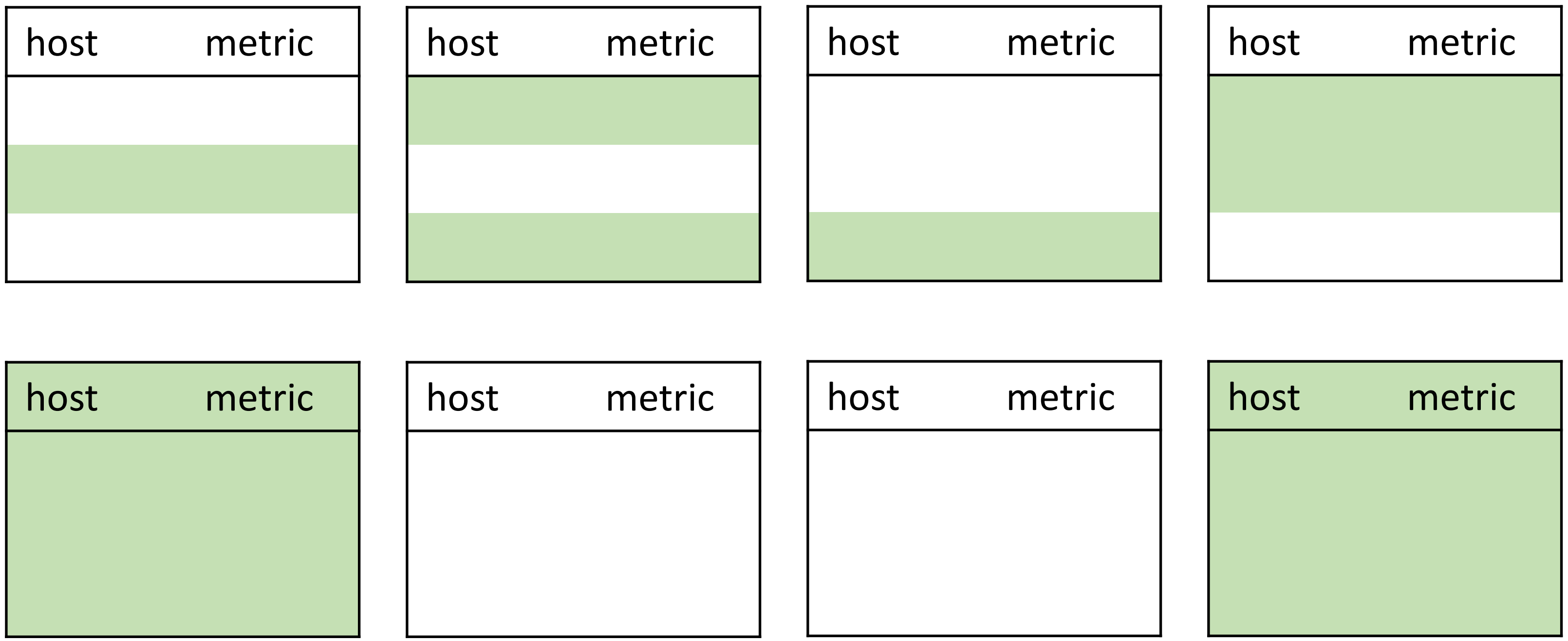
New Problem: partition-level sampling

row samples



partition samples



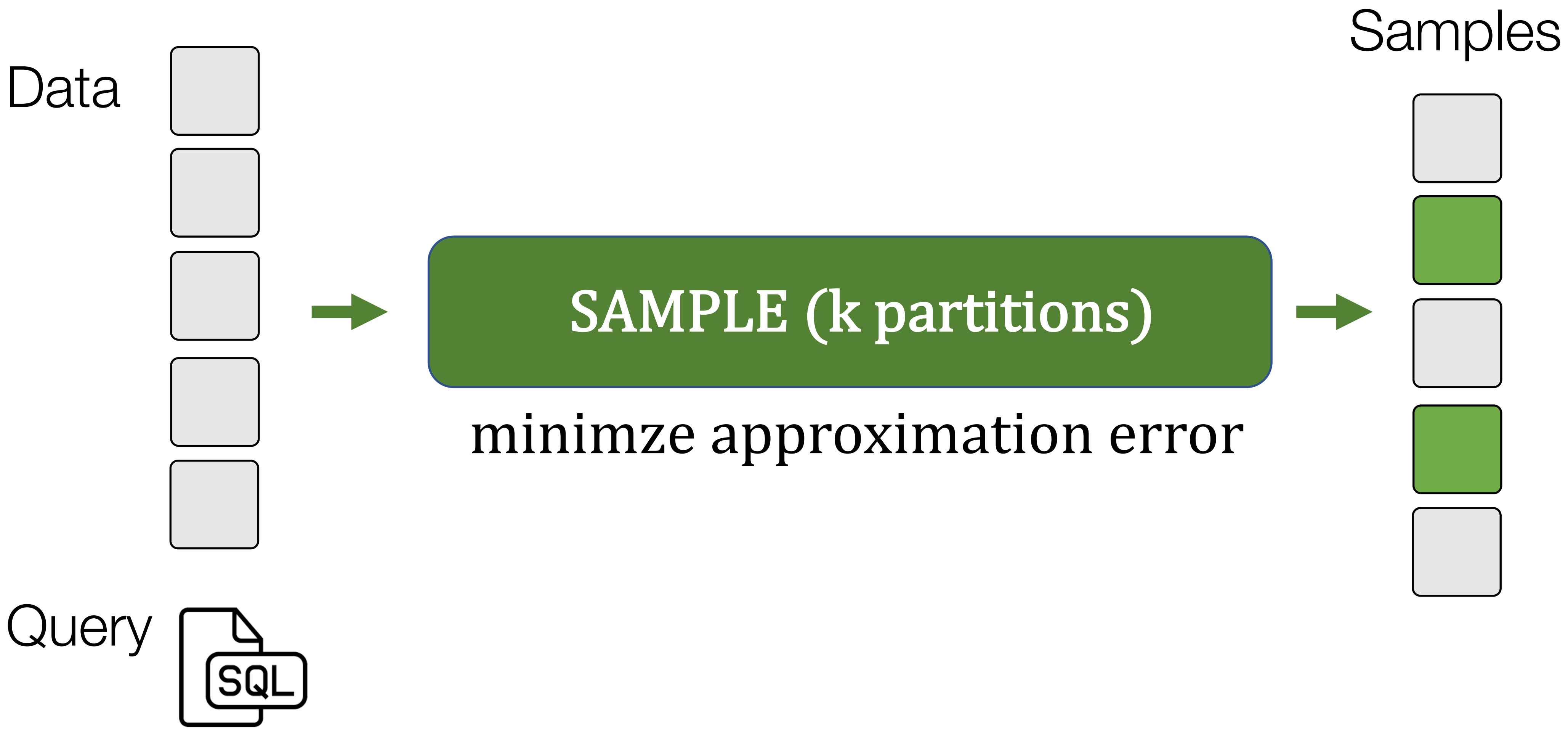


Either ALL or NONE of the rows in a partition are sampled

10% row samples => read **99.9%** of data

10% partition samples => read **10%** of data

New Problem: partition-level sampling



How to process SQL queries efficiently?

Two classic ideas:

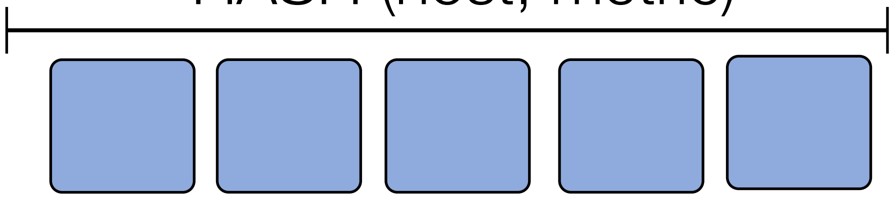
Partition is the new Row

#1 Sampling

SAMPLE (k partitions)

#2 Indexing

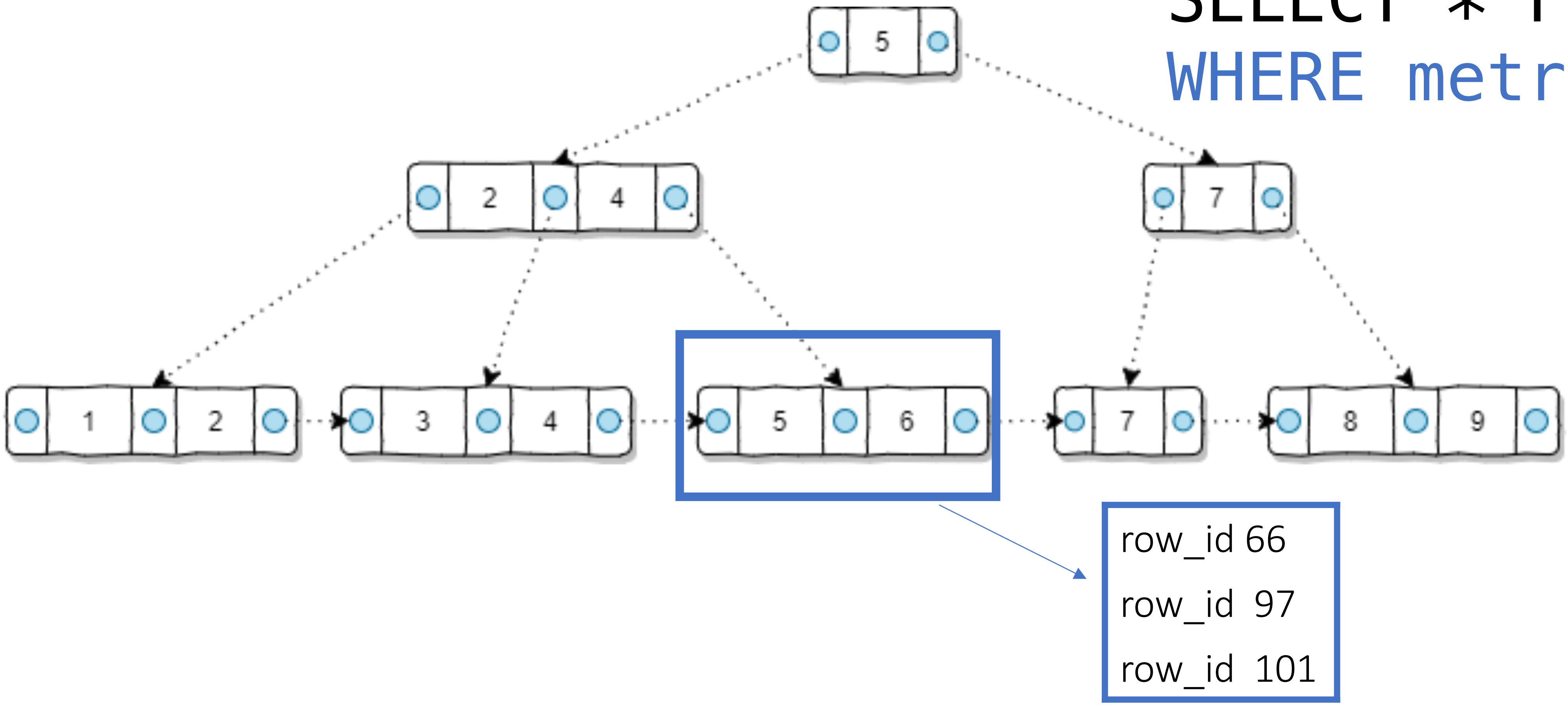
HASH (host, metric)



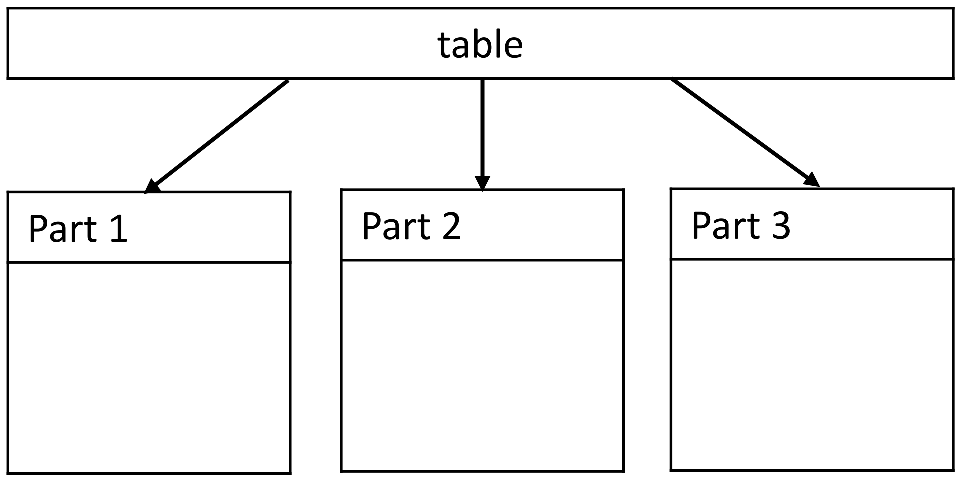
Before: row-level index

Locate rows quickly by avoiding sequential scans

```
SELECT * FROM tbl
WHERE metric = 5
```



Now: partition-level metadata as index

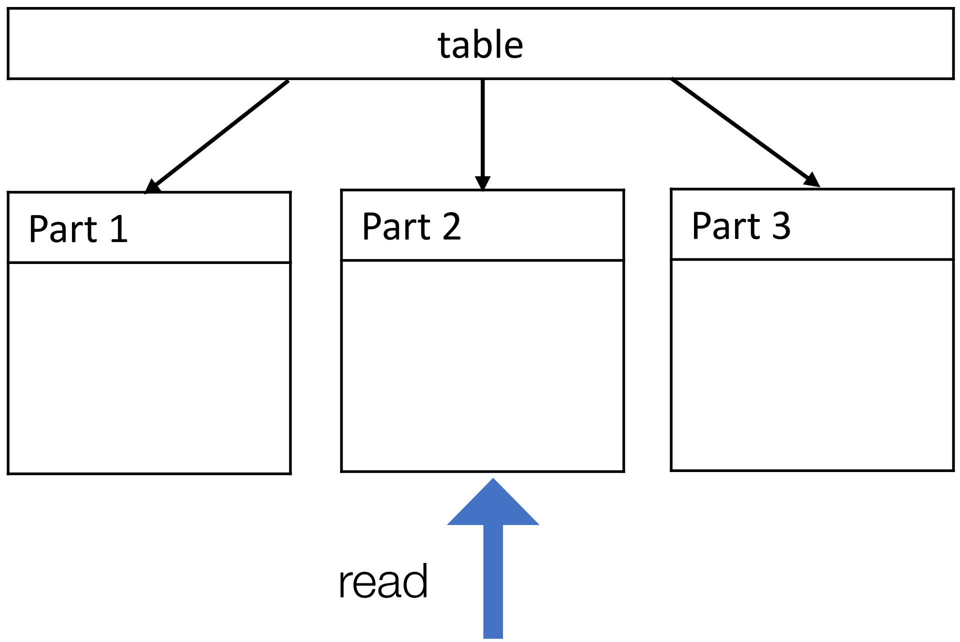


Min max of each column

Part	min(metric)	max(metric)	min(host)	max(host)
1	6	8	server1	server5
2	3	10	server1	server5
3	1	4	server1	server5

```
SELECT * FROM tbl
WHERE metric = 5
```

Now: partition-level metadata

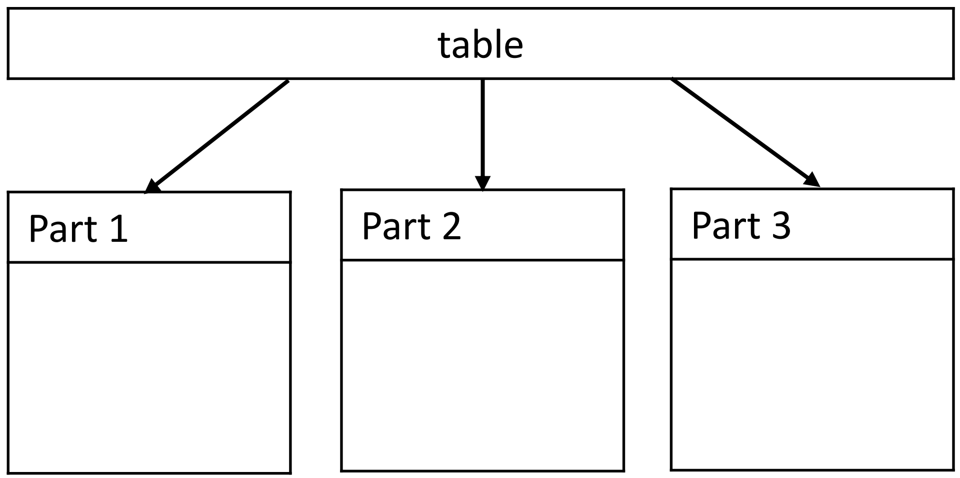


Min-max Index

Part	min(metric)	max(metric)	min(host)	max(host)
1	6	8	server1	server5
2	3	10	server1	server5
3	1	4	server1	server5

```
SELECT * FROM tbl
WHERE metric = 5
```

Now: partition-level metadata

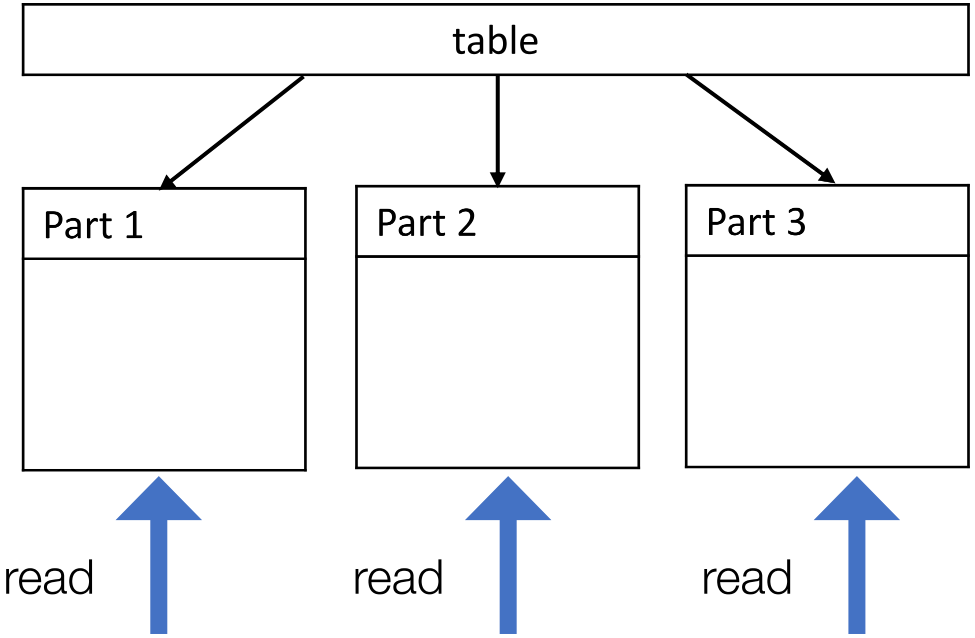


Min-max Index

Part	min(metric)	max(metric)	min(host)	max(host)
1	6	8	server1	server5
2	3	10	server1	server5
3	1	4	server1	server5

```
SELECT * FROM tbl
WHERE host = server2
```

Now: partition-level metadata

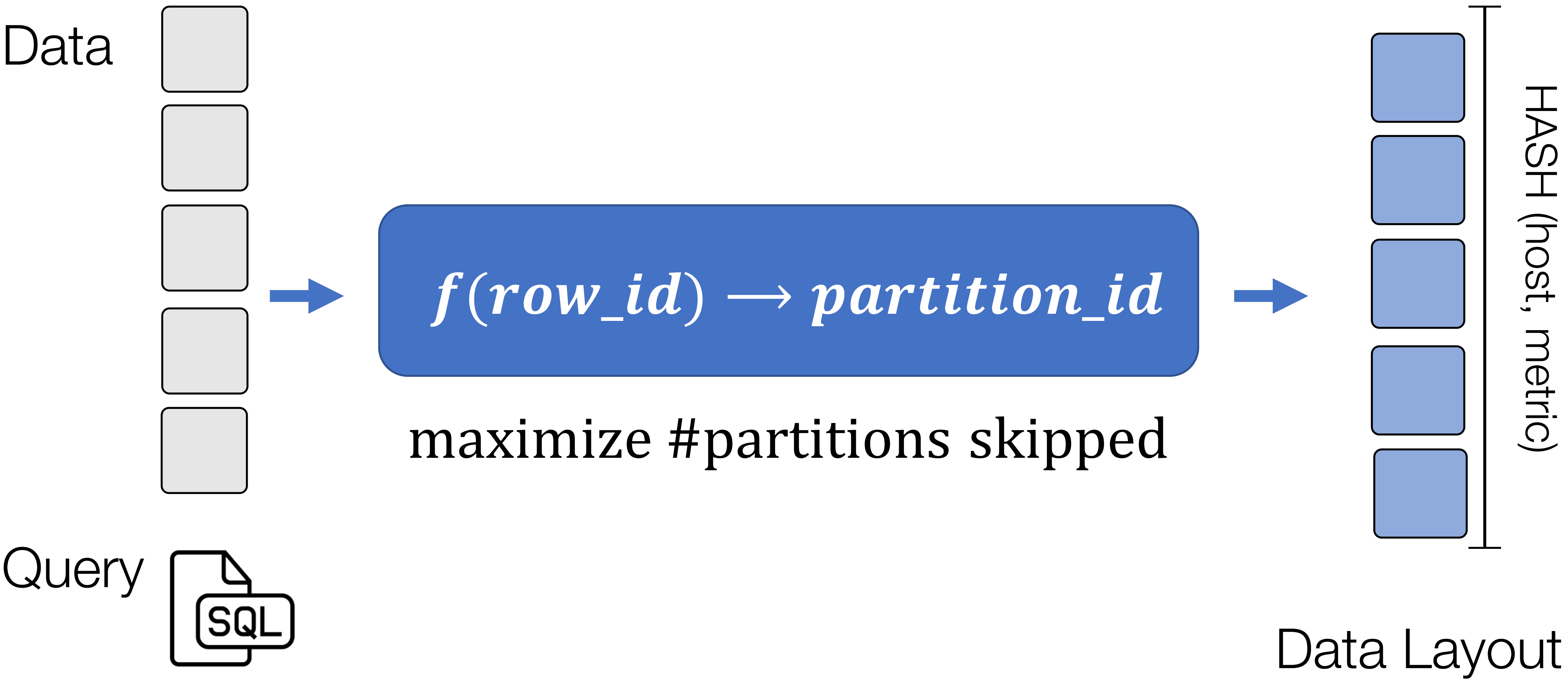


Min-max Index

Part	min(metric)	max(metric)	min(host)	max(host)
1	6	8	server1	server5
2	3	10	server1	server5
3	1	4	server1	server5

```
SELECT * FROM tbl
WHERE host = server2
```

New Problem: how to design partitions?



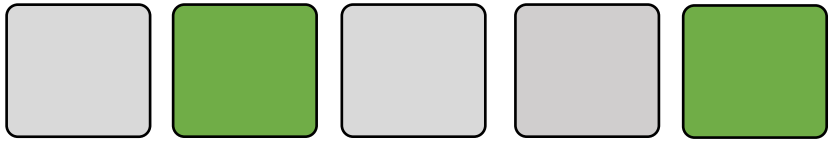
How to process SQL queries efficiently?

Two classic ideas:

Partition is the new Row

#1 Sampling

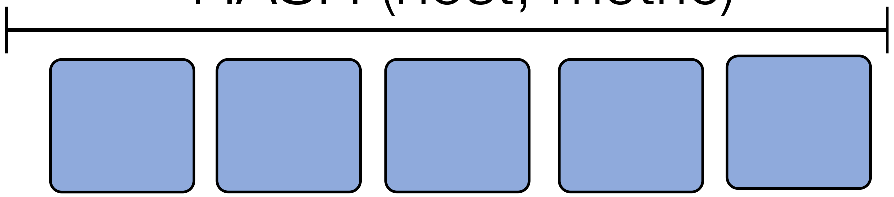
SAMPLE (k partitions)



#2 Indexing

$$f(row\_id) \longrightarrow part\_id$$

HASH (host, metric)

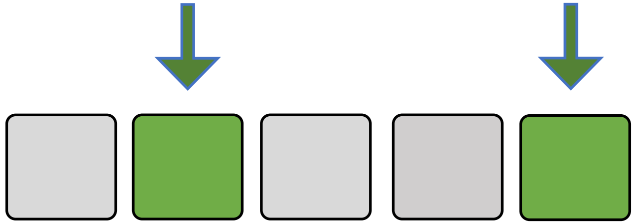


Talk Overview

#1 How to Sample?

PS3: weighted partition-level sampling

- 3-70x reduction in #partitions read



#2 How to Index?

OLO: online layout optimization

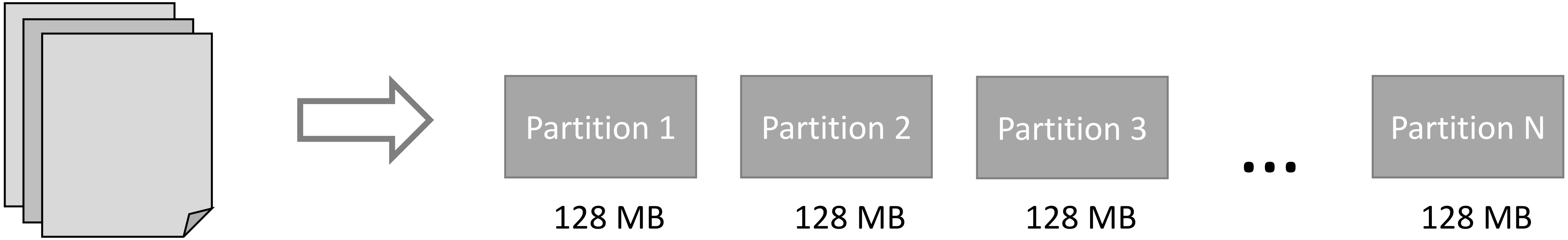
- 30% faster than a single layout

Approximate Partition Selection for Big-Data Workloads using Summary Statistics

Kexin Rong*, Yao Lu*, Peter Bailis, Srikanth Kandula*, Philip Levis

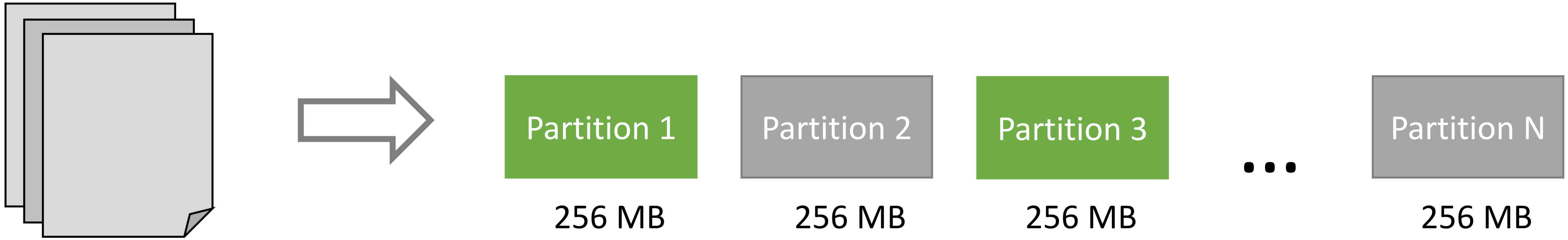
Stanford, Microsoft*

"Hidden" cost of row-level sampling



Media such as flat files in data lakes and columnar stores does not support *random access*

Partition-level Sampling



Sampling fraction ∝ I/O cost:

Either ALL or NONE of the rows in a partition are sampled

Uniform partition-level sampling is already supported in practice

Challenge: How to select partitions?

Partition 1

X	Y
ham	10
ham	2
…	

Partition 2

X	Y
ham	3
ham	5
…	

…

Partition 99

X	Y
ham	1
ham	5
…	

Partition 100

X	Y
spam	50000
spam	40000
…	

```
SELECT SUM(Y) GROUP BY X
```

- random partition-level sample ≠ random sample of the dataset
  - Rows in partition can be correlated

Challenge: How to select partitions?

Partition 1

X	Y
ham	10
ham	2
…	

Partition 2

X	Y
ham	3
ham	5
…	

…

Partition 99

X	Y
ham	1
ham	5
…	

Partition 100

X	Y
spam	50000
spam	40000
…	

```
SELECT SUM(Y) GROUP BY X
```

- random partition-level sample ≠ random sample of the dataset
  - Rows in partition can be correlated
- Unclear how to perform stratified/importance sampling
  - Needed by queries with **`GROUP BY`** or complex aggregates

Problem Statement

- Input:
  - A partitioning of the dataset
  - Sampling budget
  - Query from workload

Data

Budget 2 partitions

Query `SELECT SUM(Y) GROUP BY X`

* Supported Queries
  Aggregate: `SUM, COUNT(*)`
  Predicate: `AND, OR, NOT`
  Group by: groups with medium cardinality
  Join: deformalized table

* Workload Assumption
  known group by columns
  known aggregate functions

Problem Statement

- Input:
  - A partitioning of the dataset
  - Sampling budget
  - Query from workload
- Output:
  - Partitions selection + weights

Data

Budget 2 partitions

Query `SELECT SUM(Y) GROUP BY X`

Answer	(ham, 10)	(spam, 1000)
Weights	99	1

Estimate (ham, 10×99) (spam, 1000×1)

Problem Statement

- Input:
  - A partitioning of the dataset
  - Sampling budget
  - Query from workload
- Output:
  - Partitions selection + weights
- Goal: minimize error

Data: Partition 1, Partition 2, …, Partition 100

Budget: 2 partitions

Query: `SELECT SUM(Y) GROUP BY X`

	Partition 1	Partition 100
Answer	(ham, 10)	(spam, 1000)
Weights	99	1

Estimate	(ham, 990)	(spam, 1000)
Exact	(ham, 1000)	(spam, 1000)

PS^3: Partition Selection with Summary Statistics

Use case:

- *Read-only* and *append-only* data stores

Solution:

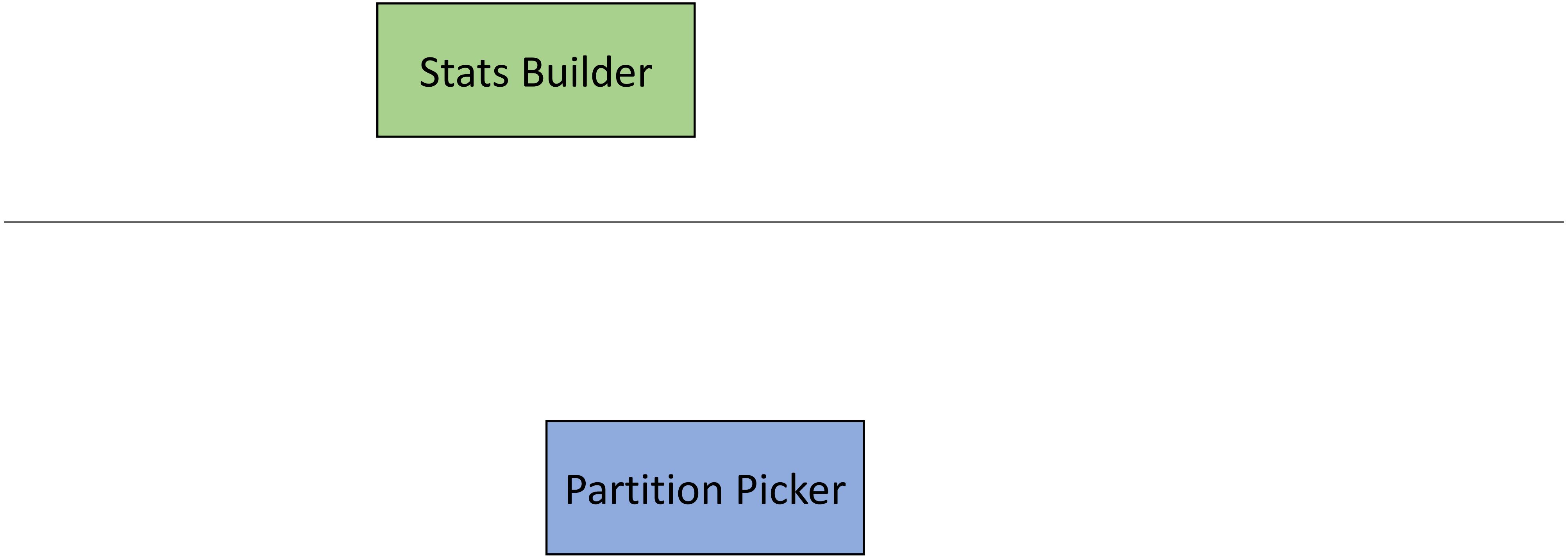
- Compute summary statistics offline
- Use statistics to select partitions online

Result:

- Between *2.7x-70x* reduction in number of partitions read to achieve the same relative error compared to random
- per partition storage overhead ≤ *100KB*

Overview of PS^3

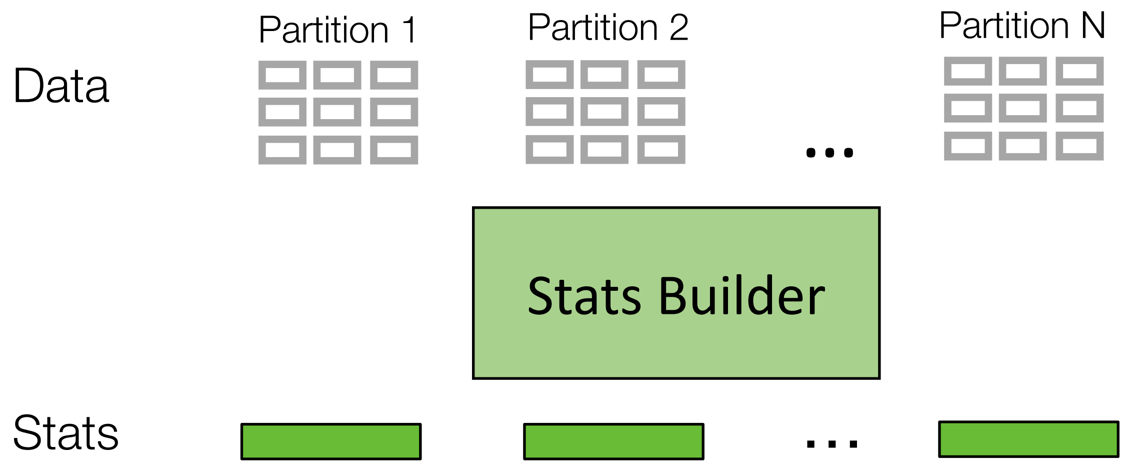
Offline



Online

Overview of PS^3

Offline

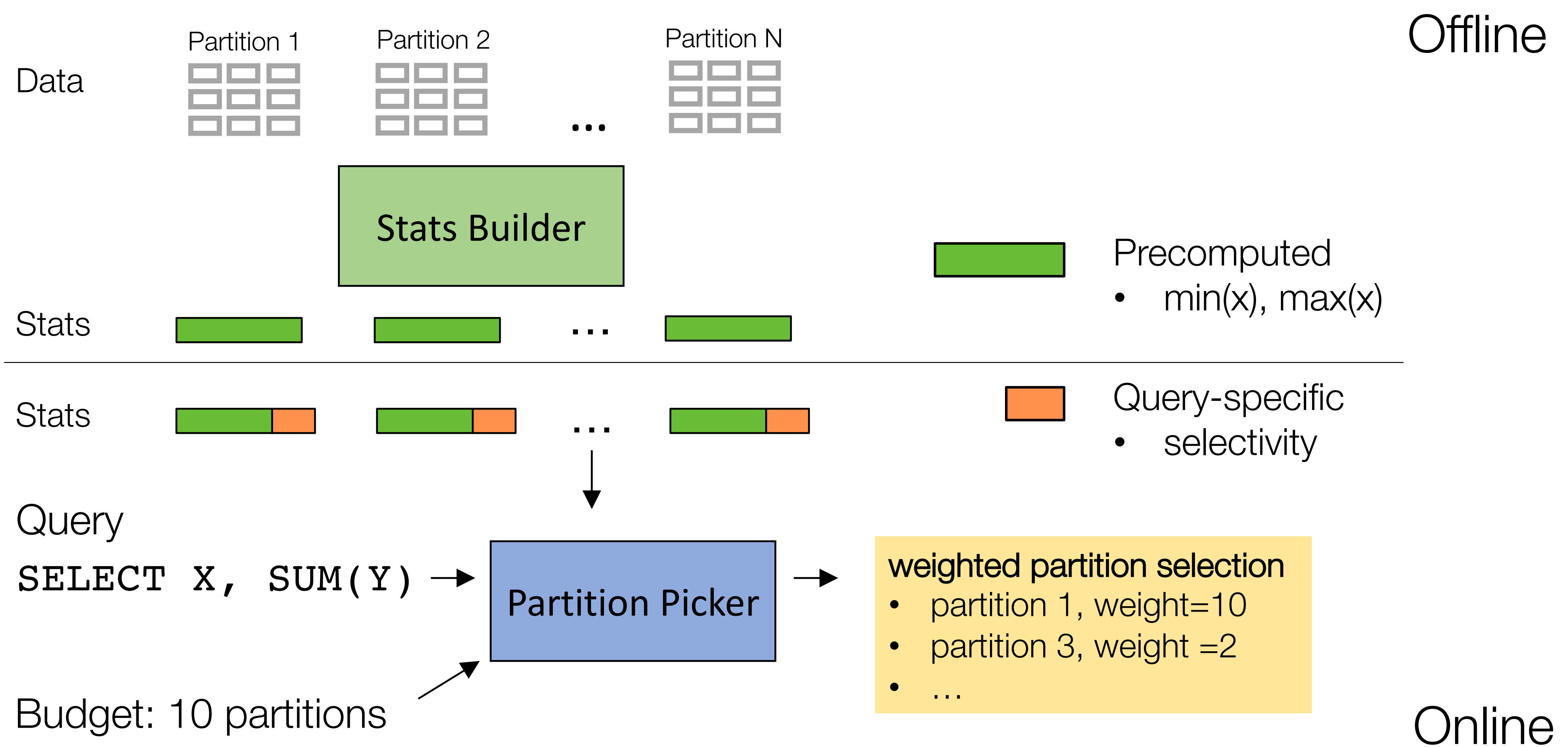


Precomputed

- min(x), max(x)

Online

Overview of PS^3



Statistics Builder: Which stats to store?

- Inspired by systems like Spark SQL, ZoneMaps

Sketches
Histograms
Measures
AKMV
Heavy Hitter

Statistics Builder: Which stats to store?

- Inspired by systems like Spark SQL, ZoneMaps

Sketches	Summary Statistics
Histograms	
Measures	min, max, moments, log moments…
AKMV	
Heavy Hitter	

Statistics Builder: Which stats to store?

- Inspired by systems like Spark SQL, ZoneMaps
- Summary statistics are different from query to query

Sketches	Summary Statistics
Histograms	selectivity estimates
Measures	min, max, moments, log moments…
AKMV	#dv, avg freq of dv …
Heavy Hitter	#hh, occurrence bitmap of hh …

- Details in the paper

Partition Picker: How to use stats?

- Idea #1: Distinguish partitions by *contribution* to the query
  - Sample more important partitions more frequently

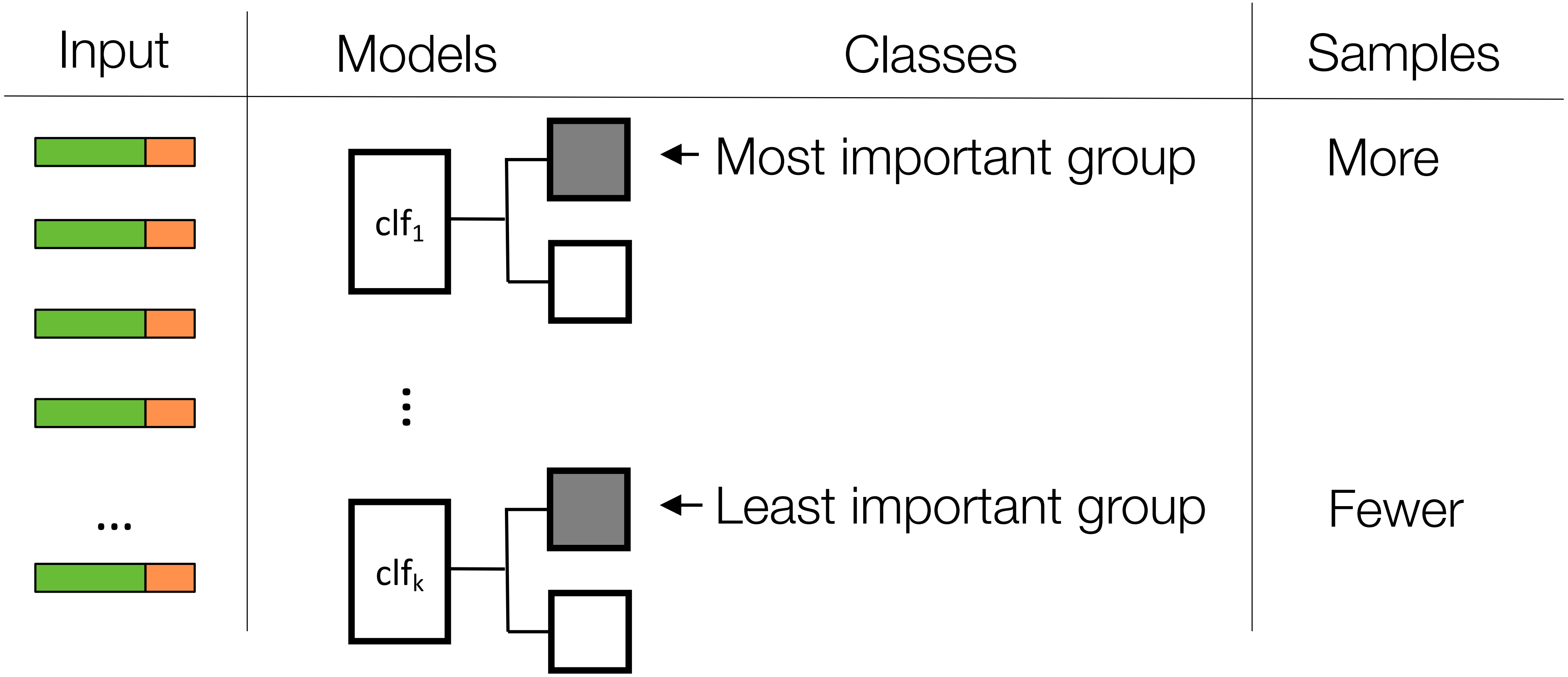
- Summary statistics is correlated with partition importance

```
SELECT SUM(Y) FROM table WHERE Z > 1 GROUP BY X
```

  - `SUM(Y) =>` max(Y), avg(Y)
  - `GROUP BY X =>` # distinct values in X
  - `WHERE Z>1 =>` selectivity

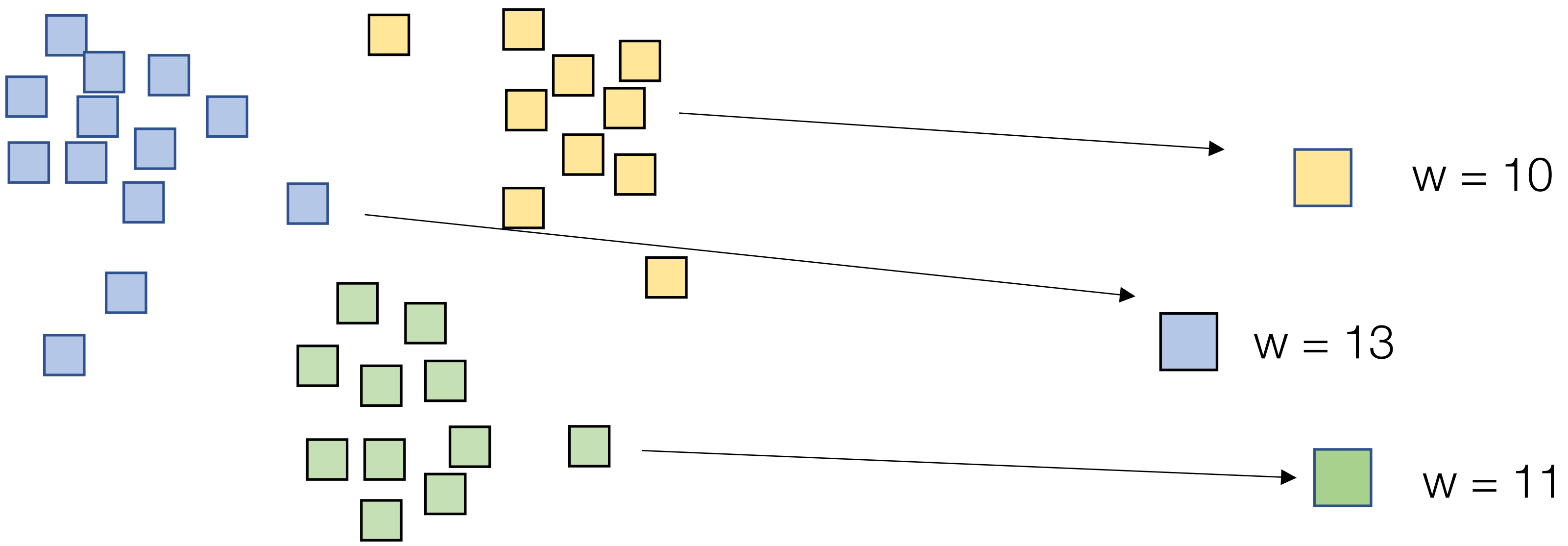
Partition Picker: How to use stats?

- Train models to classify partitions into importance groups
  - Trained per *workload*, data layout and dataset



Partition Picker: How to use stats?

- Idea #2: Leverage partition *redundancy*
  - Use clustering to choose dissimilar partitions



Evaluation: Accuracy

random	random+filter	LSS[1]	PS^3
↑	↑	↑	↑
Random partition level sampling	Random augmented with predicate filter enabled by summary statistics	modified prior work on Learned Stratified Sampling	our prototype

[1] B. Walenz, S. Sintos, S. Roy, and J. Yang. Learning to sample: Counting with complex queries. PVLDB, 13(3):390-402, 2019.

Evaluation: Accuracy

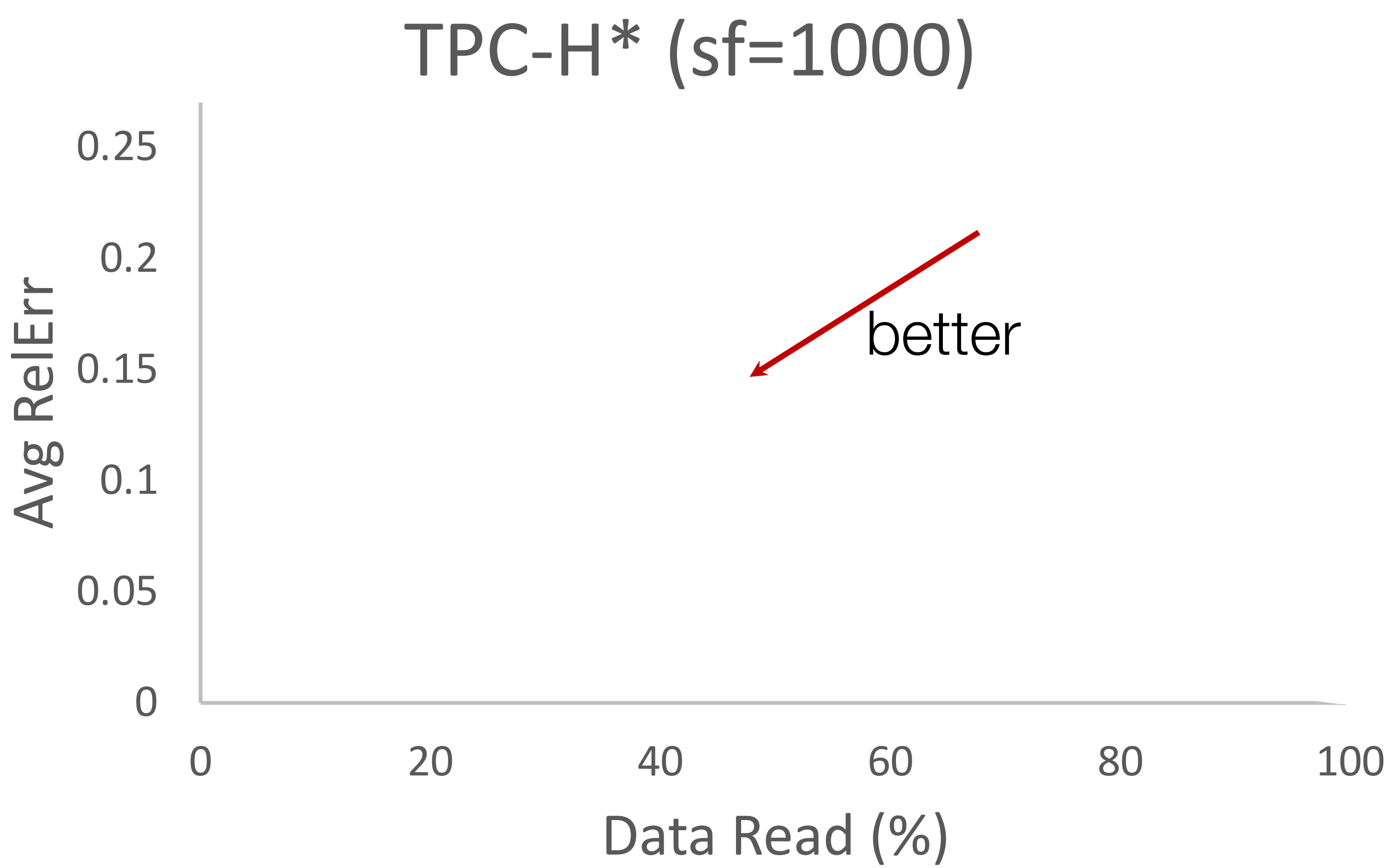
random   random+filter   LSS   PS^3

TPC-H* (sf=1000)

- Dataset
  - 2.5GB partitions × 3000
- Query

```
SELECT o_orderpriority,
       SUM(l_extendedprice*l_discount)
FROM tpch
WHERE r1_name = "EUROPE" AND
      p_size >7
GROUP BY o_orderpriority
```

Evaluation: Accuracy



data read (%) <=>
total compute hours

Evaluation: Accuracy

- PS^3 1% partition (1.5% error)
    - LSS 5% partition
    - random+filter 40% partition
    - random 70% partition

Evaluation: Overhead

- Per partition storage overhead

Aria	KDD	TPC-DS*	TPC-H*
18KB	12KB	103KB	84KB

  - Per partition storage overhead is constant

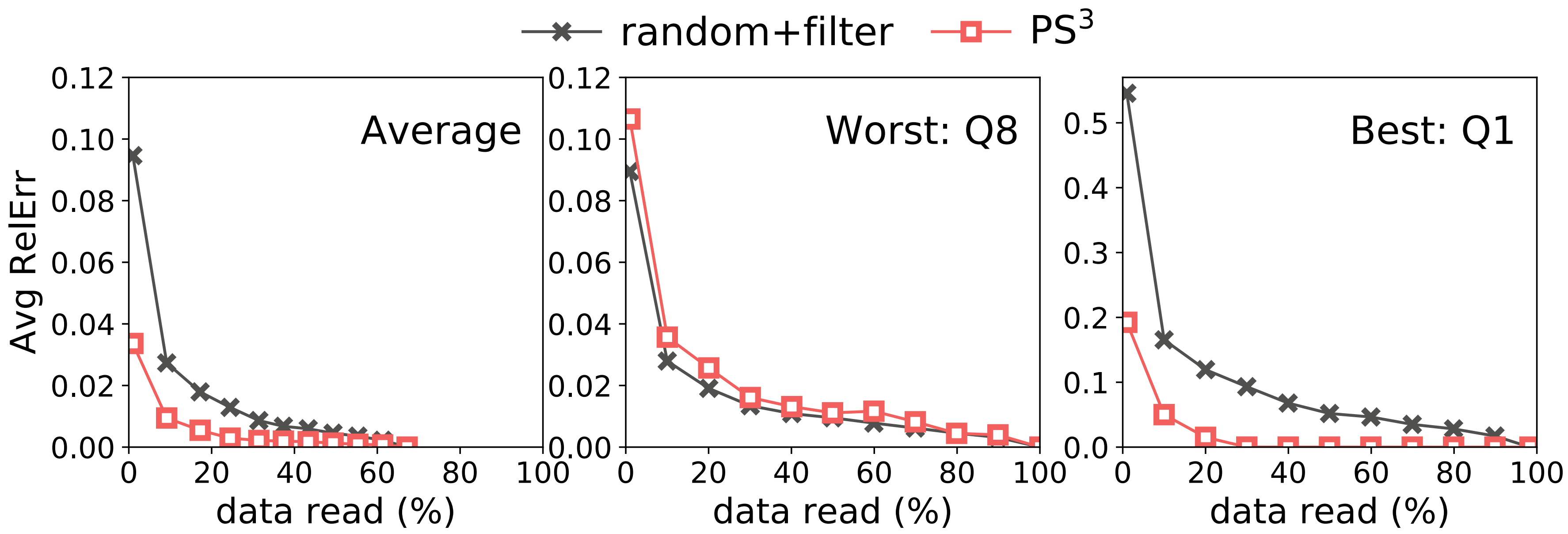
- Single-thread partition picker overhead

Aria	KDD	TPC-DS*	TPC-H*
90ms	106ms	220ms	1002ms

  - Can be further reduced via parallelization

More experiments in the paper

- Sensitivity analysis
  - Partition counts
  - Data layouts
  - Query selectivity
- Generalization to unseen TPC-H queries

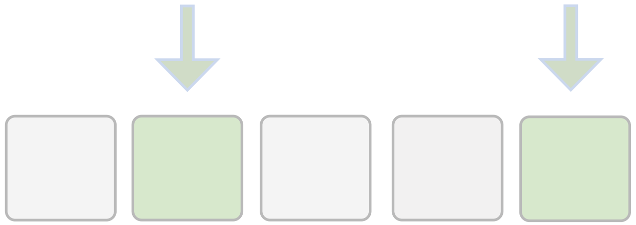


Talk Overview

#1 How to Sample?

PS3: weighted partition-level sampling

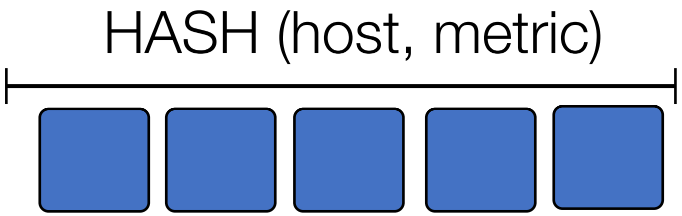
- 3-70x reduction in #partitions read



#2 How to Index?

OLO: online layout optimization

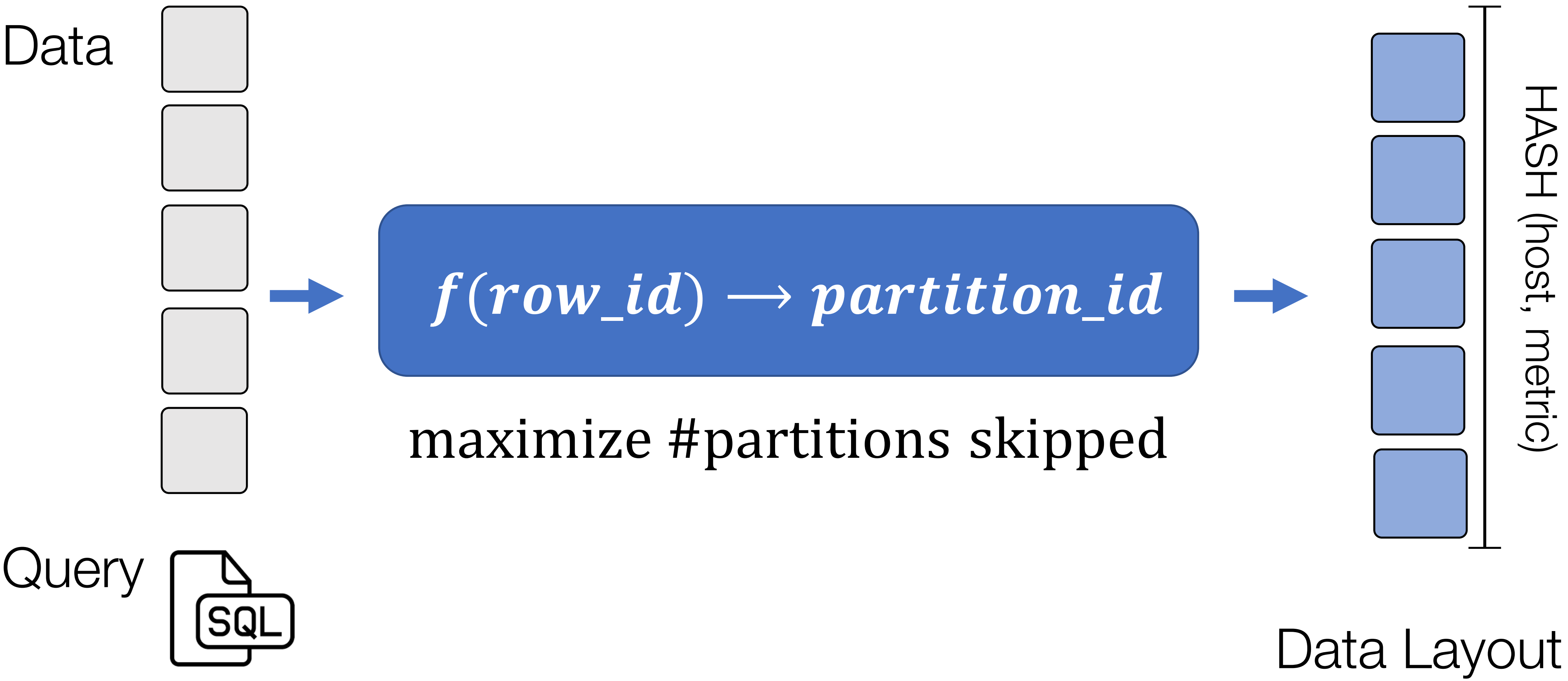
- 30% faster than a single layout



Online Data Layout Optimization via Metrical Task Systems

Kexin Rong, Paul Liu, Moses Charikar

Data layout affects query performance

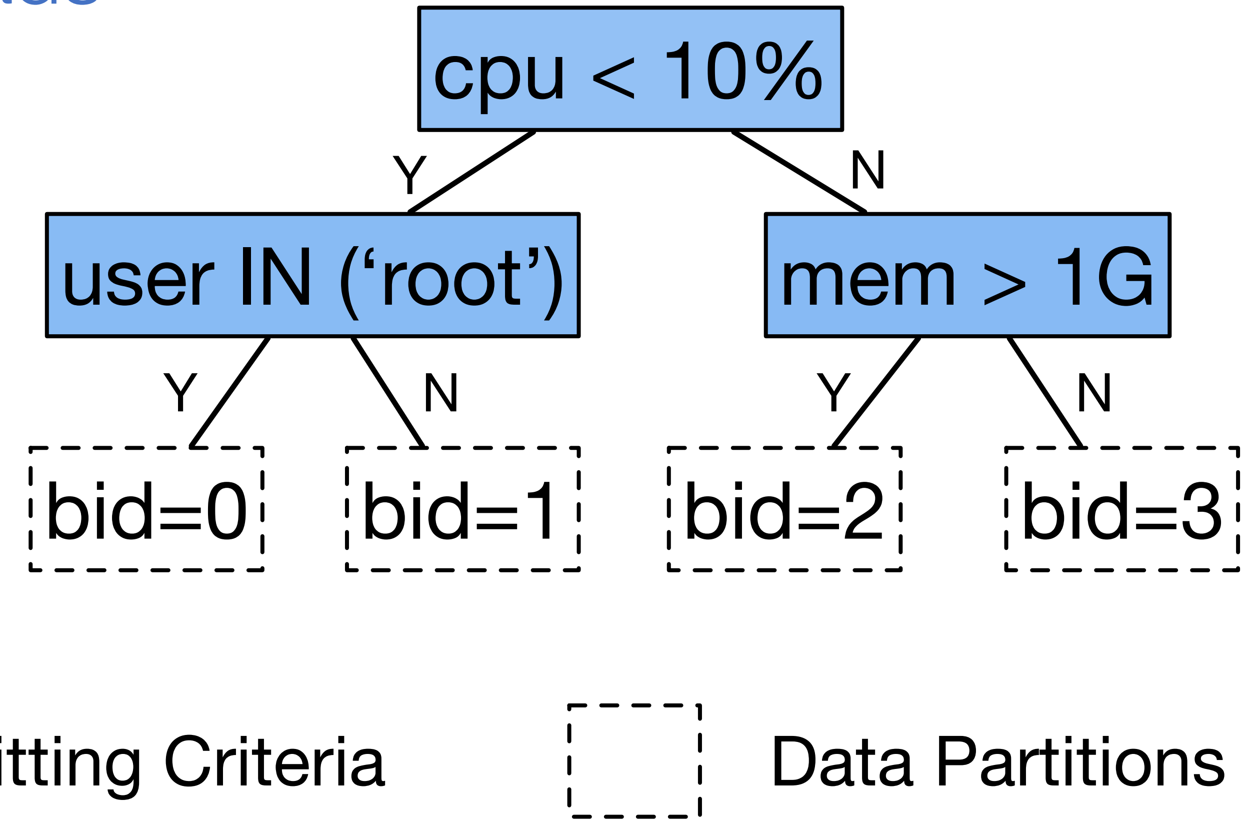


How to design layout to maximize skipping?

Specialize to query workloads

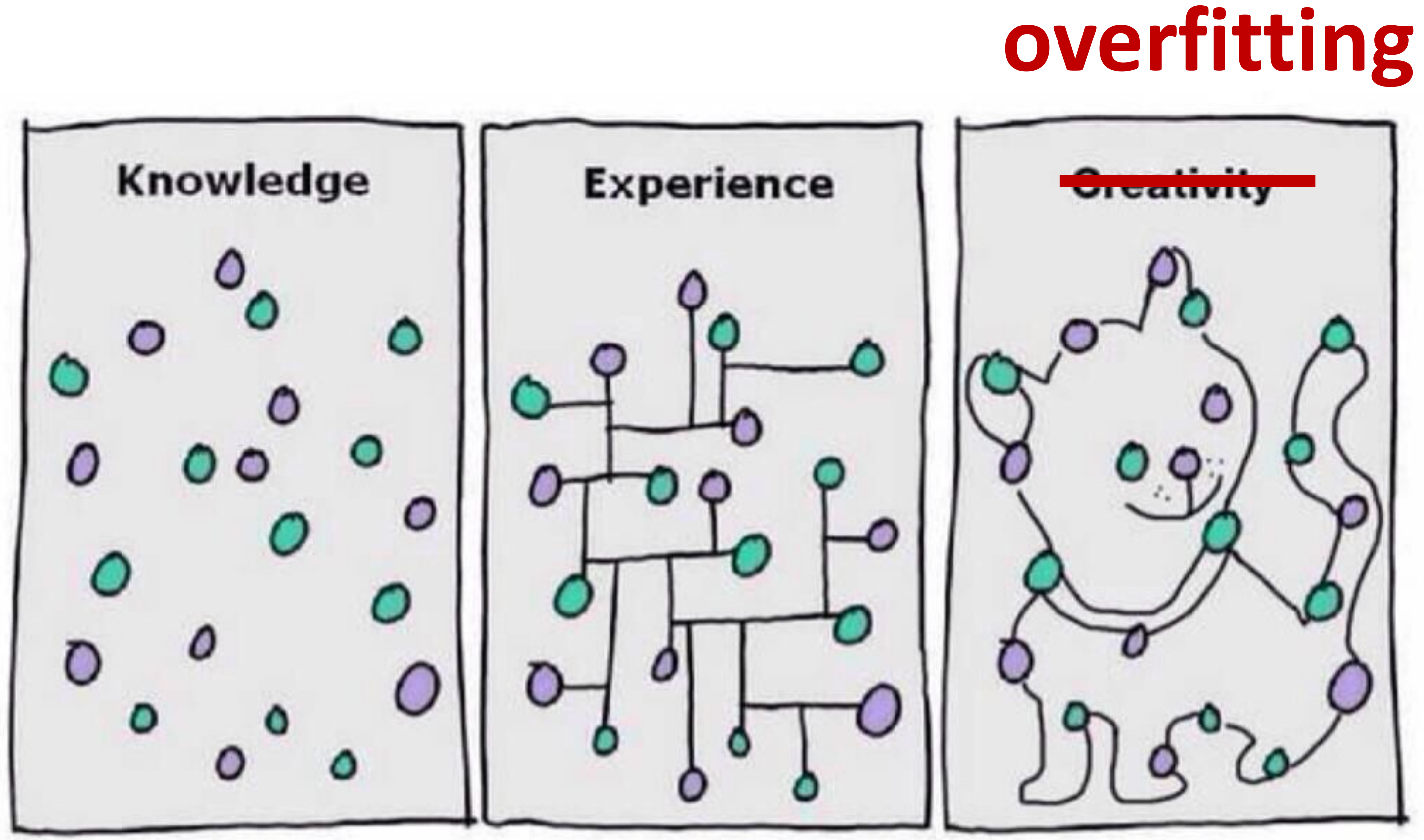
Qd-tree (SOTA) [1]

- Extract *predicates* from workloads as splitting criteria of the tree



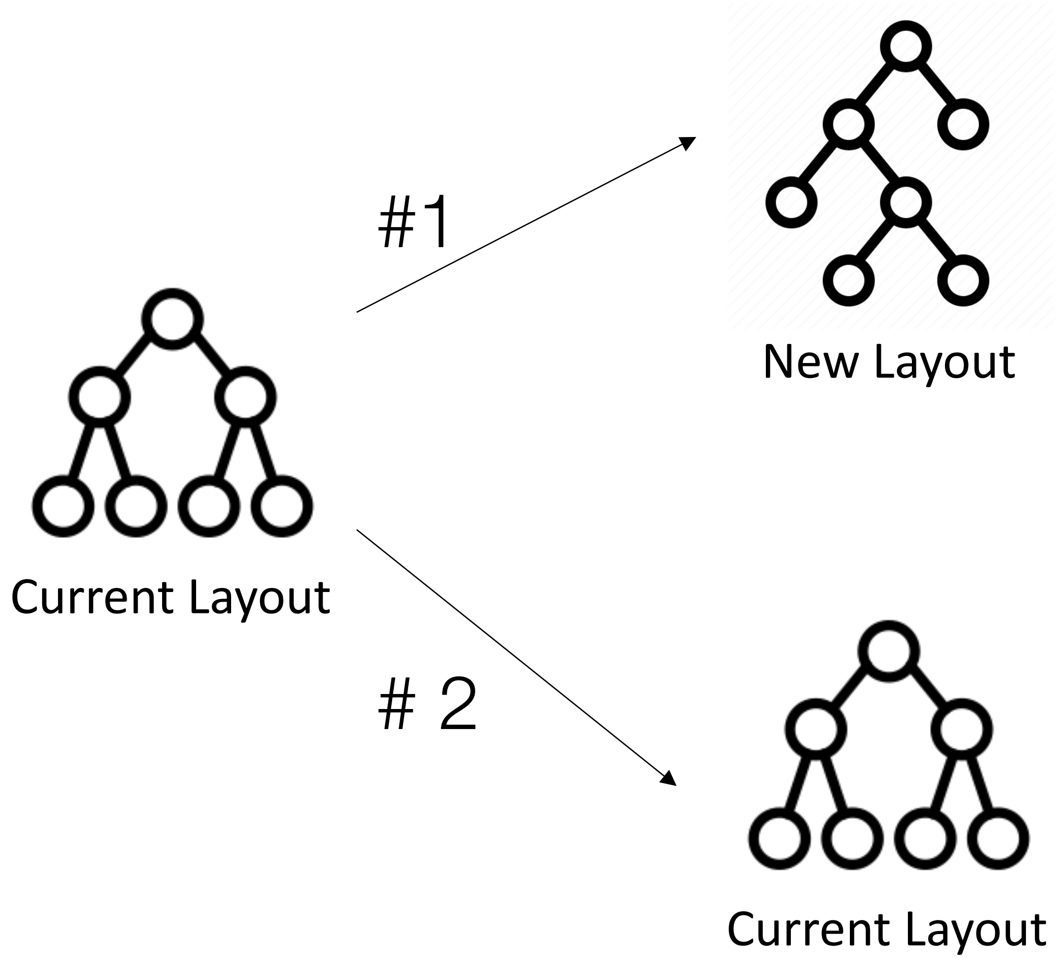
[1] Z. Yang, et al. Qd-tree: Learning Data Layouts for Big Data Analytics. In SIGMOD 2020.

Problem: layouts overfit to workloads



Performance subject to workload changes

What to do when workload changes?



Option 1: Change layout

Reorganization cost +

Query cost -

Option 2: Do nothing

Reorganization cost

Query cost +

Goal: Minimize query + reorganization costs

Input: unknown sequence of queries

Output: *when* and *how* to reorganize

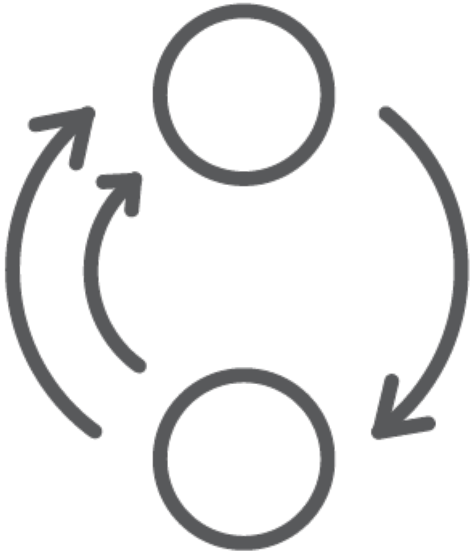
One approach: prediction task

Supervised learning

future workload

Reinforcement learning

reward of actions



Decisions rely on predictions of the future

Our approach: online algorithms

- Does NOT rely on predictions of future workload
- Provide guarantees in the form of competitive ratio

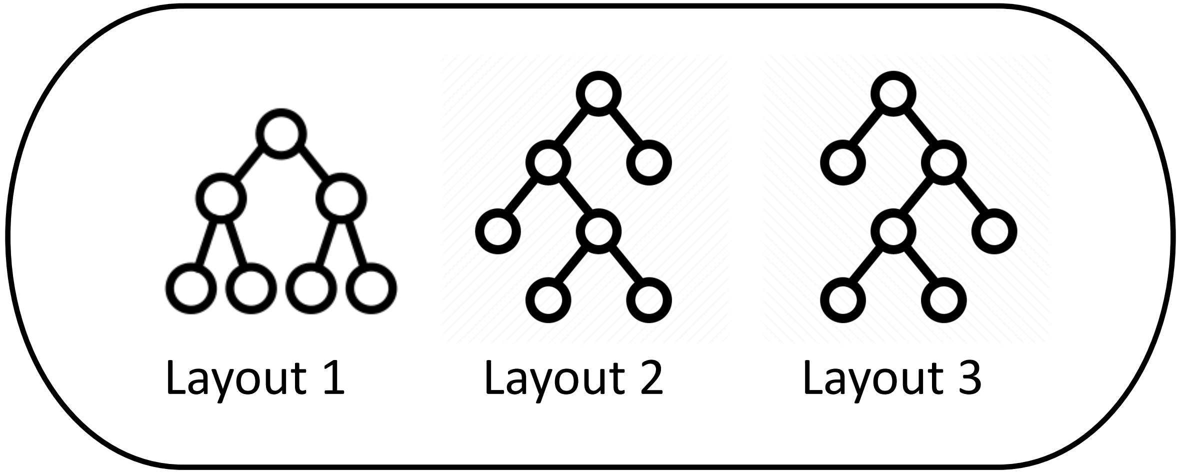
$$\sup_{I} \frac{cost(online\ algorithm)}{cost(offline\ algorithm)}$$

Our approach: online algorithms

- Does NOT rely on predictions of future workload
- Provide guarantees in the form of competitive ratio

$$\sup_{I} \frac{cost(online\ algorithm)}{cost(offline\ algorithm)} \sim \log(|S|)$$

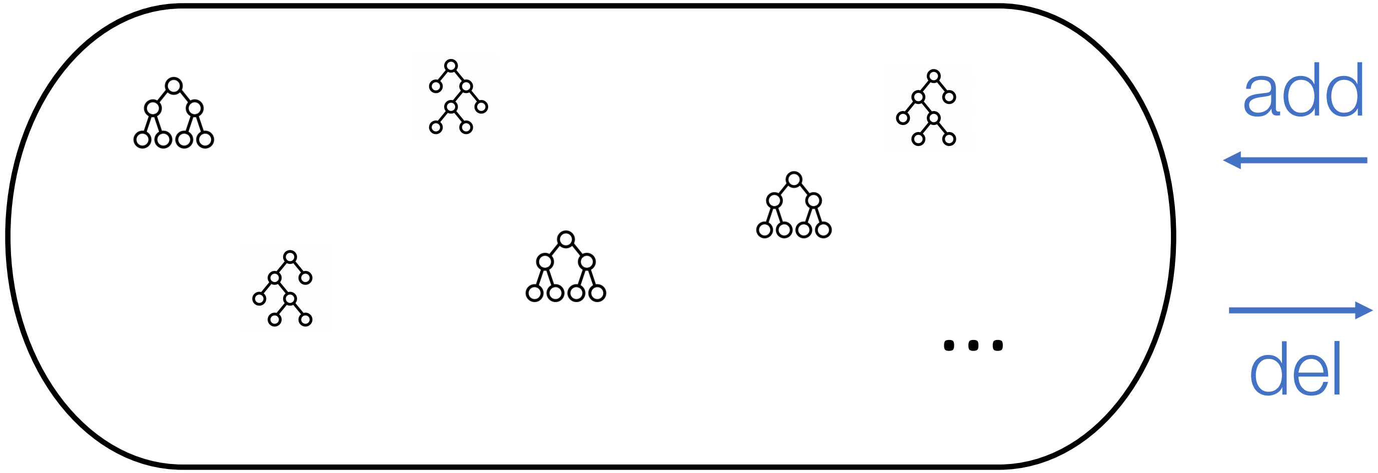
State Space S:



Challenge: intractable state space

$$f(row\_id) \longrightarrow partition\_id$$

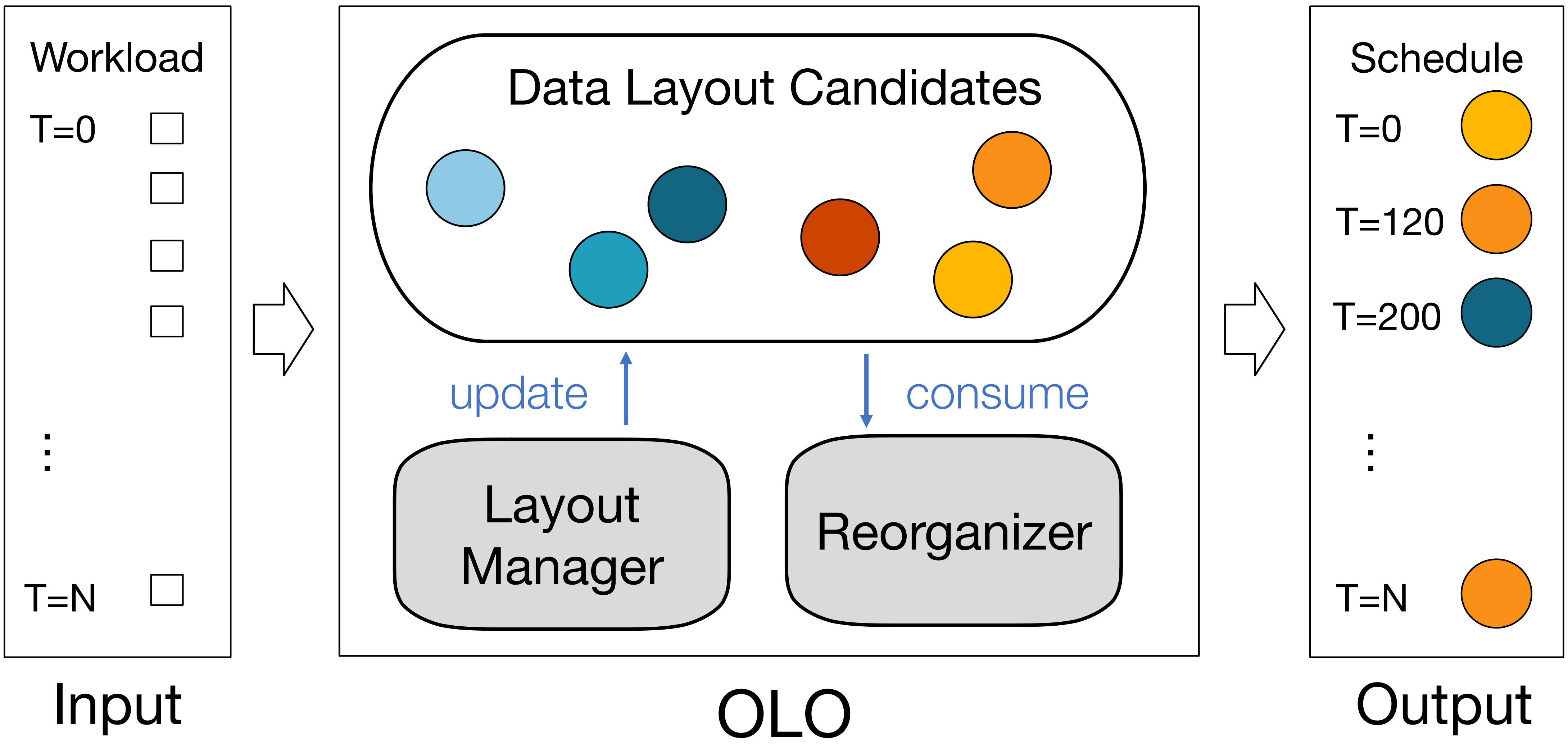
State Space S:



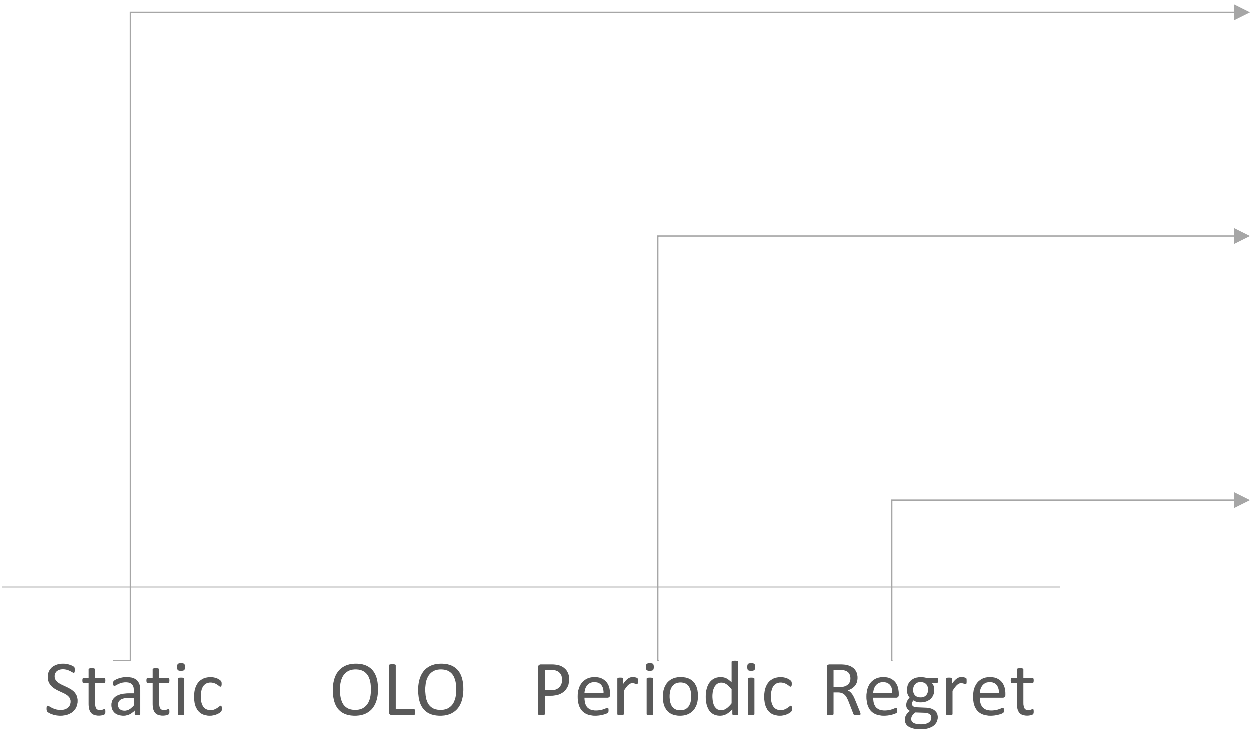
Insight: allow the state space to change over time

Result: competitive ratio $\sim \log(|S_{max}|)$

OLO Overview



Evaluation: End-to-end Time

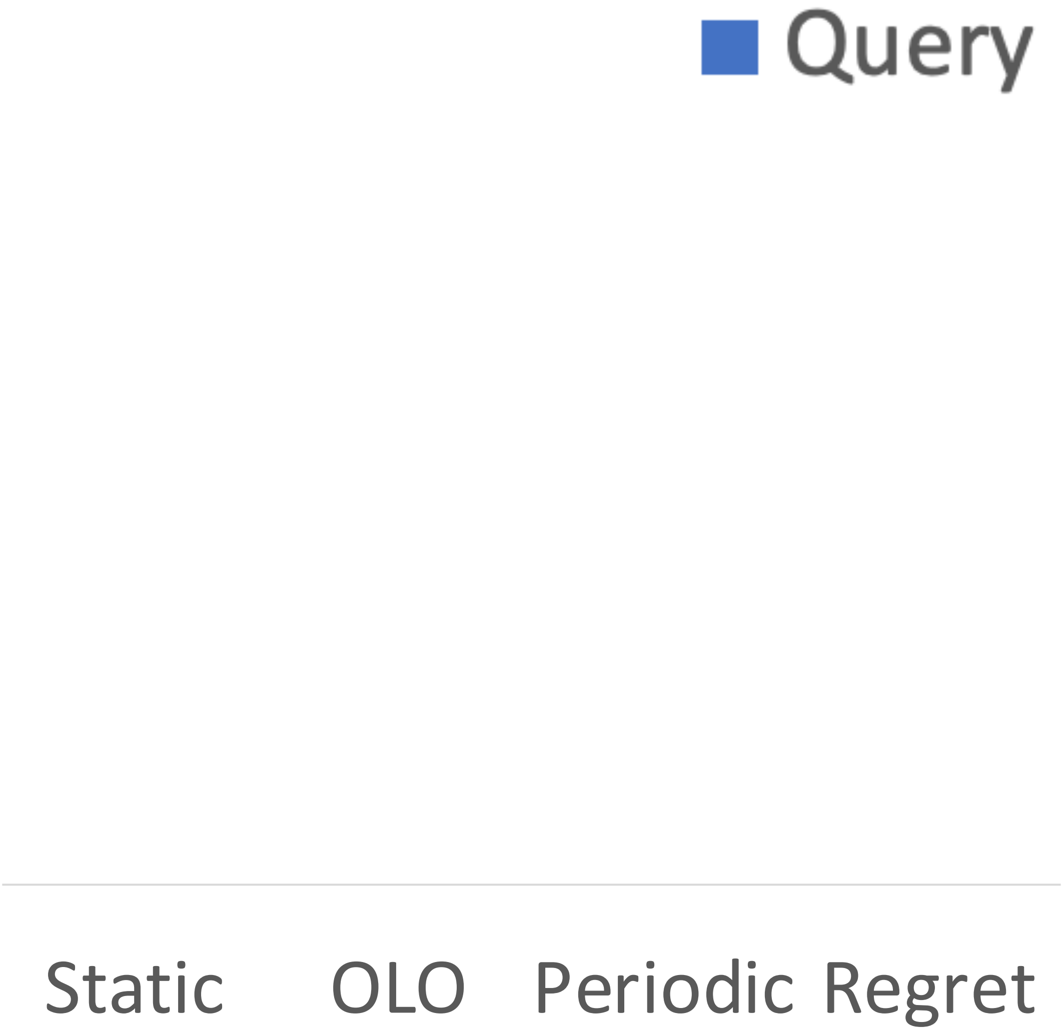


Best offline layout for the entire query workload

Switch to new layout if better

Switch to new layout if cumulative regret > reorganization cost

Evaluation: End-to-end Time

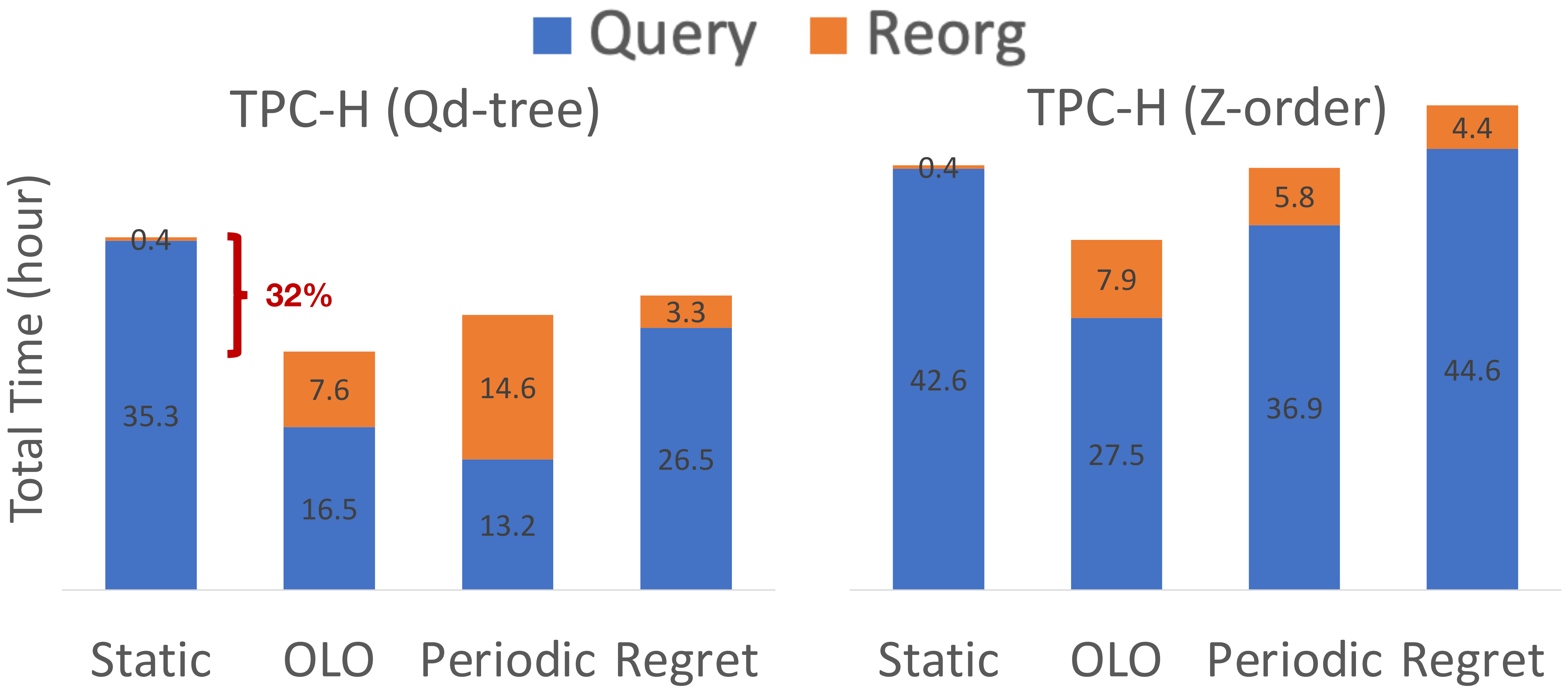


Dataset
- TPC-H

Workload
- 30k queries
- 20 templates

Metric:
- query + reorganization time

Evaluation: End-to-end Time

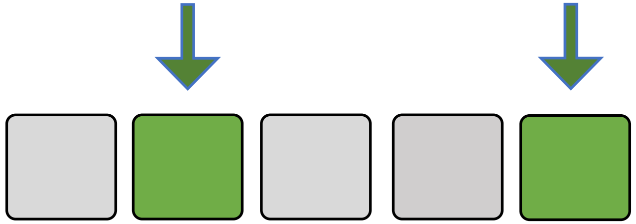


This Talk

#1 How to Sample?

PS3: weighted partition-level sampling

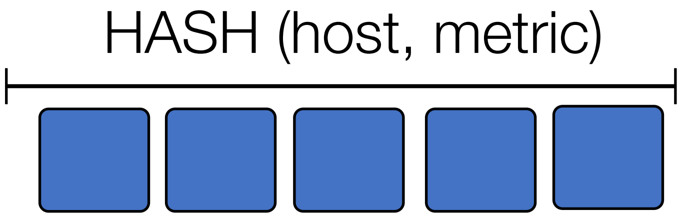
- 3-70x reduction in #partitions read



#2 How to Index?

OLO: online layout optimization

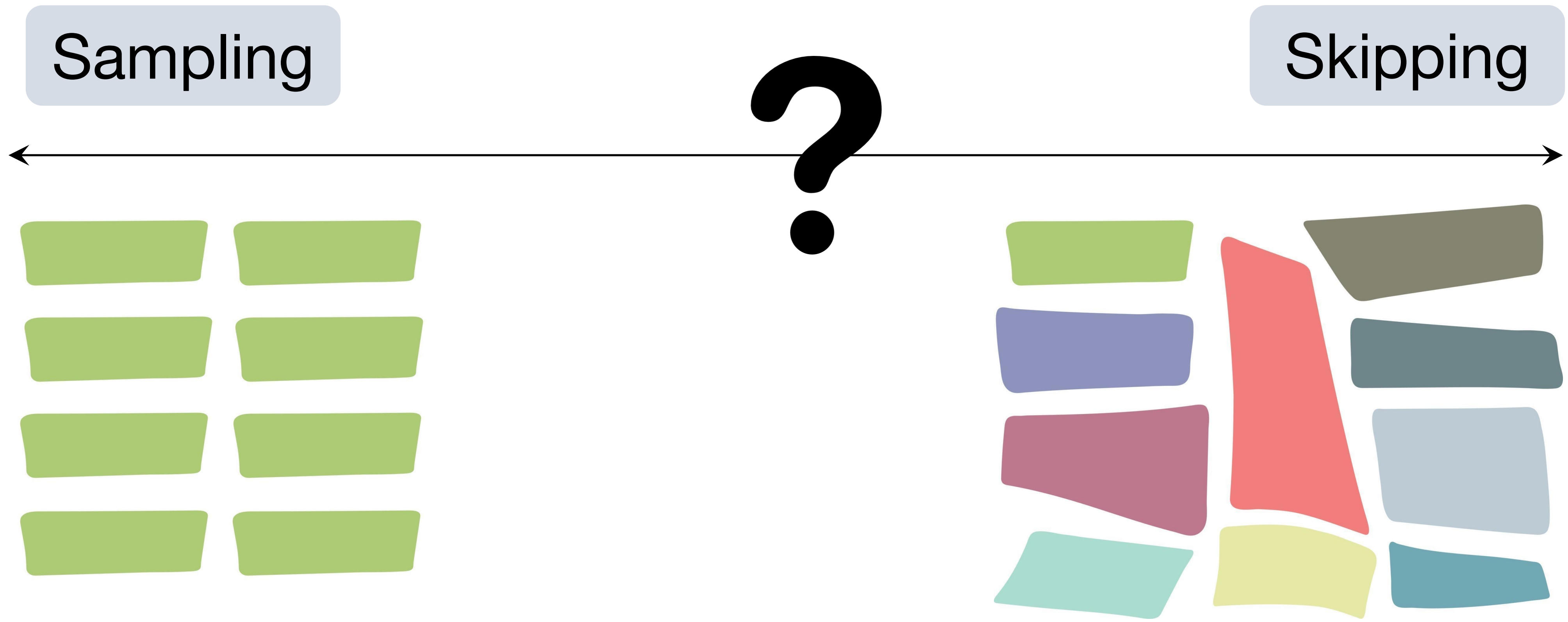
- 30% faster than a single layout



Kexin Rong krong@cs.stanford.edu

Question to think about

How to balance the needs between sampling and skipping?

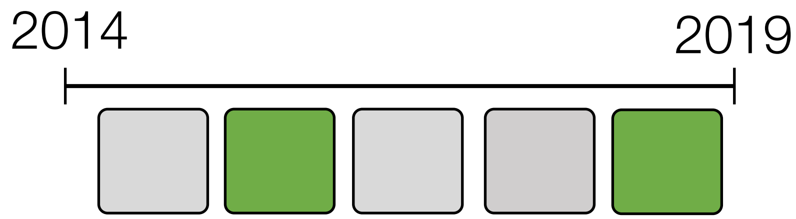


This talk

#1 How to Sample?

PS3: weighted partition-level sampling

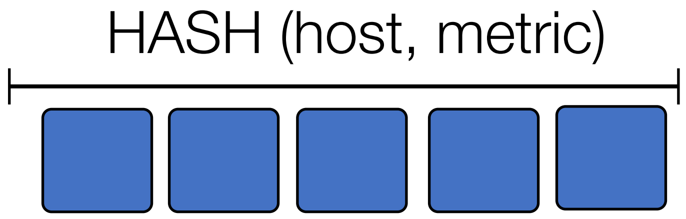
- 3-70x reduction in #partitions read



#2 How to Index?

OLO: online layout optimization

- 30% faster than a single layout



Kexin Rong krong@cs.stanford.edu