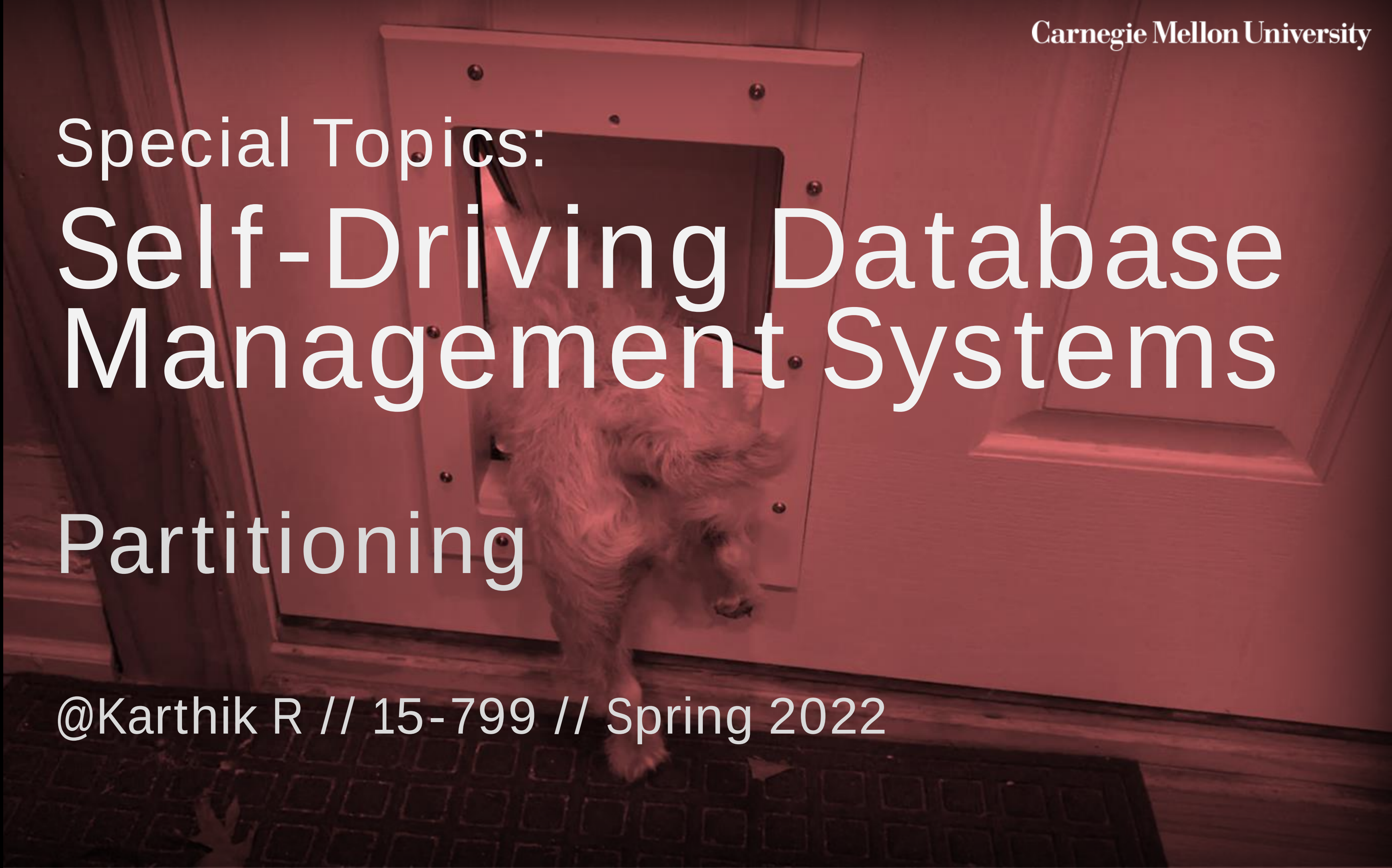
Lecture #10



Carnegie Mellon University

Special Topics:

Self-Driving Database Management Systems

Partitioning

@Karthik R // 15-799 // Spring 2022

LAST CLASS

- Design Choices in Knob Tuning
  - Knob Selection
  - Configuration Optimization
  - Knowledge Transfer
- Survey of Algorithms

TODAY'S AGENDA

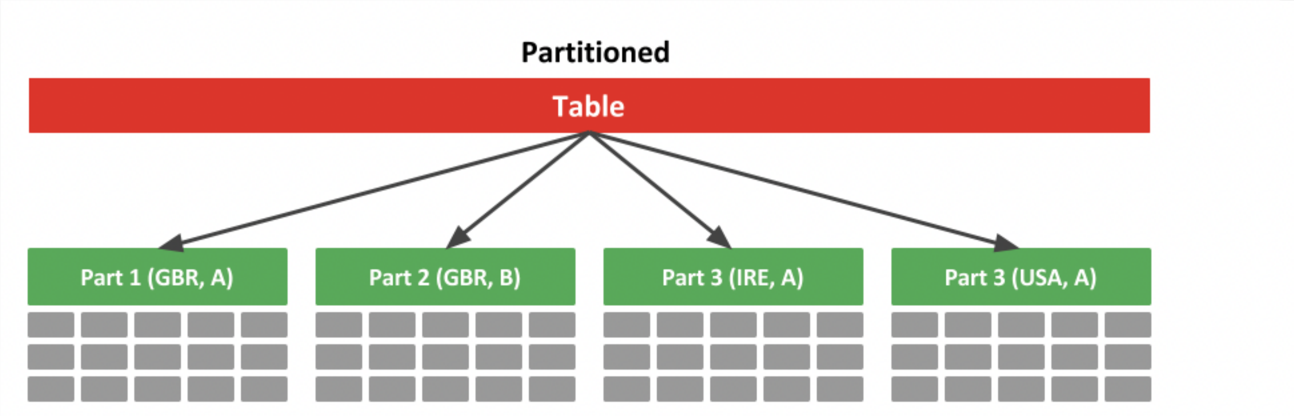
- Refresher on Partitioning
- What is Microsoft up to?
- The Partitioning Problem
- Approach
- Results
- Parting Thoughts

TODAY'S AGENDA

- Refresher on Partitioning
  - WHY, WHAT and HOW of Partitioning
  - The Big Picture
- What is Microsoft up to?
- The Partitioning Problem
- Approach
- Results
- Parting Thoughts

WHAT IS PARTITIONING?

- Partitioning splits database across multiple resources



Source: Oracle-base

NEED FOR PARTITIONING

Source: Reddit

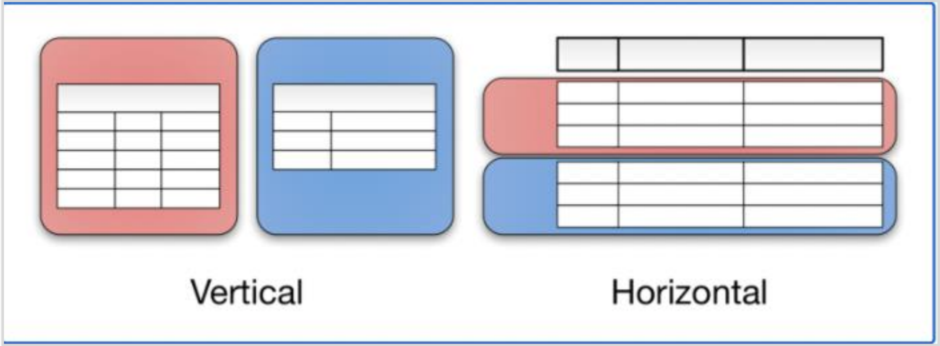
NEED FOR PARTITIONING

Improves:

- Scalability
  - →"Scale out" rather than "Scale up"
- Performance
  - → Queries run over subset of data
- Availability
  - → Removes single point of failure
  - → Related: Concept of "replicas" or redundancy

TYPES OF PARTITIONING

- By design
  - Horizontal, Vertical, …



- By method
  - Hash, Range, List, …
- By Subject
  - Tables, (n-Cl) Indexes, Materialized Views, Partitions

Source: Medium

THE BIG PICTURE

- Physical Database Design involves
  → Indexes, Materialized Views and Partitions
- Performance vs Manageability (for humans)
- Automated partition selection is NP-hard
  → Combinatorial explosion
  → Superficially similar to index selection

TODAY'S AGENDA

- Refresher on Partitioning
- What is Microsoft up to?
  - Timeline
  - Database Tuning Advisor
- The Partitioning Problem
- Approach
- Results
- Parting Thoughts

MICROSOFT'S TIMELINE

1997-98

- Cost-driver Index Selection
- Index Analysis

2000

- Selection of Materialized Views

2004

- Release of DB Tuning Advisor (DTA) for MSSQL 2005
- Integration of horizontal and vertical partitioning

DATABASE TUNING ADVISOR

Considerations

→ Co-location of Join columns (size and locality)

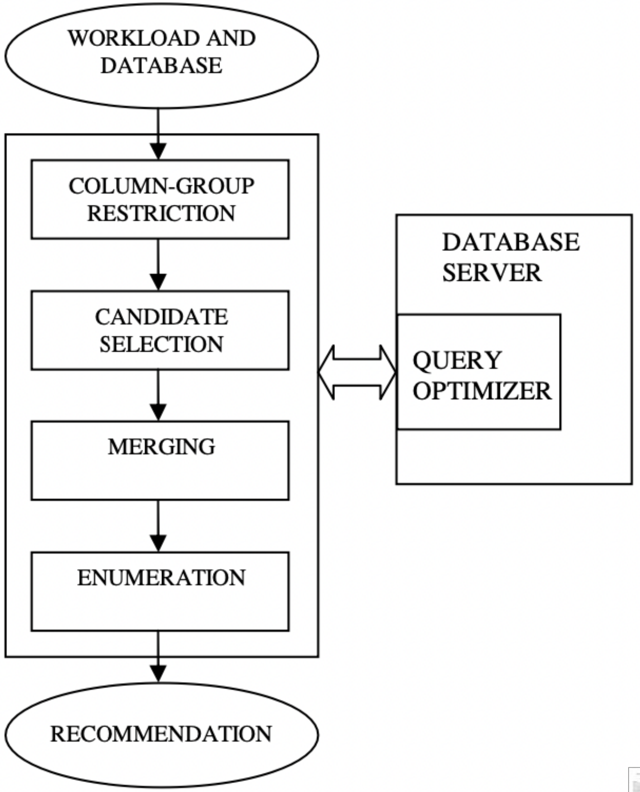
→ Mutual exclusivity (unlike indexes)

→ Specificity vs Generality

→ Storage and Update costs

→ Alignment

DATABASE TUNING ADVISOR



INTEGRATING VERTICAL AND HORIZONTAL PARTITIONING INTO AUTOMATED PHYSICAL DATABASE DESIGN
SIGMOD 2004

DATABASE TUNING ADVISOR

- Column Group Restriction
  → Simulating vertical partitions is hard!
  → Heuristic based
- Candidate Selection
  → Greedy (m, k) algorithm
- Merging
  → Over the entire workload
  → Involves indexes, MVs and partitions
- Enumeration

INTEGRATING VERTICAL AND HORIZONTAL PARTITIONING INTO AUTOMATED PHYSICAL DATABASE DESIGN
SIGMOD 2004

TODAY'S AGENDA

- Refresher on Partitioning
- What is Microsoft up to?
- The Partitioning Problem
- Approach
- Results
- Parting Thoughts

PARTITIONING FOR CLOUD DBs

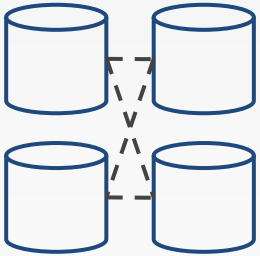
"Learn the optimal partitioning for each cloud customer, for a given database schema, for a given workload"

Suitability of Reinforcement Learning
→ Combinatorial Optimization problem
→ Exploration vs Exploitation instead of Greedy

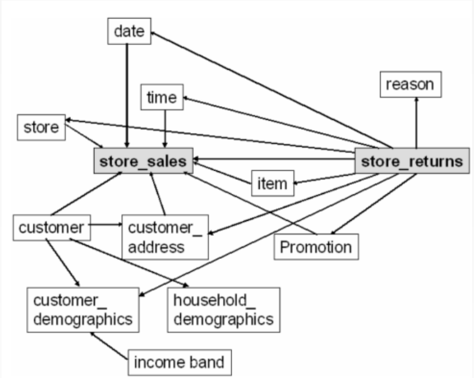
LEARNING A PARTITIONING ADVISOR FOR CLOUD DATBASES
SIGMOD 2020

PARTITIONING FOR CLOUD DBs

Diverse Hardware



+ Complex Schema



+ Arbitrary Workload

```sql
SELECT sr_customer_sk      AS ctr_customer_sk,
       sr_store_sk         AS ctr_store_sk,
       Sum(sr_return_amt)  AS ctr_total_return
FROM   store_returns,
       date_dim
WHERE  sr_returned_date_sk = d_date_sk
       AND d_year = 2001
GROUP  BY sr_customer_sk
```

(+ Various DBMS)

Postgres-XL

- Problems with existing approaches
  - → Cost estimates are coupled to hardware
  - → Cost estimates themselves are inaccurate

BENJAMIN HILPRECHT; LEARNING A PARTITIONING ADVISOR FOR CLOUD DATBASES
SIGMOD 2020

TODAY'S AGENDA

- Refresher on Partitioning
- What is Microsoft up to?
- The Partitioning Problem
- Approach
- Results
- Parting Thoughts

RL FOR PARTITIONING

- State:
  - → Given a table T_i with attributes $(a_{i1}, a_{i2}, \ldots., a_{in})$
  - → One-hot encoding as $(r_i, a_{i1}, a_{i2}, \ldots., a_{in})$

  - → "Edges" for co-partitioning
  - → Workload state
  - → All appended together

LEARNING A PARTITIONING ADVISOR FOR CLOUD DATBASES
SIGMOD 2020

RL FOR PARTITIONING

- Workload state modeling:
  → Encoding JOIN predicates, WHERE clauses etc …. Does not account for arbitrary nesting.

  → Nested query featurization …. complex encoding, larger training data

  → Keep It Simple Stupid! Encode only frequency info.
  $s(Q) = (f_1, f_2, \ldots, f_m)$

  → Bucketize queries based on Selectivity. How?

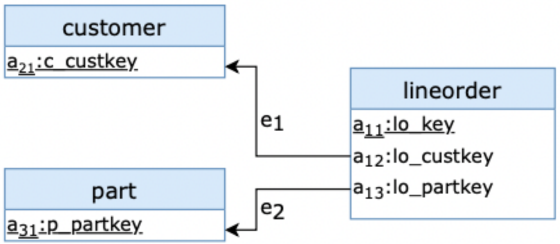
LEARNING A PARTITIONING ADVISOR FOR CLOUD DATBASES
SIGMOD 2020

RL FOR PARTITIONING

- Action:
  - → Q-learning … small state space desirable
  - → Partitioning OR Replication
  - → Only considers HASH Partitions
  - → Only considers horizontal partitioning with fixed no. of nodes
  - → One-hot encoding of (repl, partition, (de)activation) with at most one active

LEARNING A PARTITIONING ADVISOR FOR CLOUD DATBASES
SIGMOD 2020

RL FOR PARTITIONING



q_1: SELECT * FROM customer c, lineorder l
WHERE l.lo_custkey=c.c_custkey;

q_2: SELECT * FROM part p, lineorder l
WHERE l.lo_partkey=p.p_partkey;

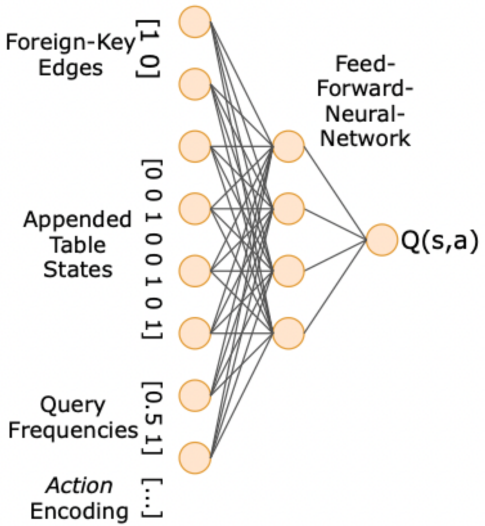
Foreign-Key Edges:
Edge e_1 for lo_custkey⟶ c_custkey: active
Edge e_2 for lo_partkey⟶ c_partkey: inactive
$s(E) = (e_1, e_2) = (1, 0)$

Table States:
lineorder partitioned by lo_custkey
$s(lineorder) = (r_1, a_{11}, a_{12}, a_{13}) = (0, 0, 1, 0)$

customer partitioned by c_custkey
$s(customer) = (r_2, a_{21}) = (0, 1)$

part replicated
$s(part) = (r_3, a_{31}) = (1, 0)$

Query Frequencies:
q_2 occurs twice as frequently as q_1
$s(Q) = (f_1, f_2) = (0.5, 1)$



(a) Database and Workload **(b) State Representation** **(c) Q-Network with Encoded State**

Figure 2: State Representation of Simplified SSB Schema and Workload.

RL FOR PARTITIONING

- Cost:
  - → "Network-centric" cost model for offline training
  - → Runtime of queries for online training
- Reward:
  - → Gain in performance for a workload
  - → $r = -\sum_{j=1}^{m} f_j * c(P, q_j)$
  - → Excludes cost of repartitioning for OLAP Workloads

LEARNING A PARTITIONING ADVISOR FOR CLOUD DATBASES
SIGMOD 2020

OFFLINE TRAINING

- Enumeration join orderings
- Estimate optimal join strategy for each join using cost model.
- Sum of costs (network + compute) is the cost of a query
- Each iteration ('episode') comprises of (t_{MAX} >= |T|) actions
- Train Q-network with SGD and loss

ONLINE TRAINING

- Runs on a copy of the database, workloads
- Cost model = true runtime
- Enumeration-based training → v.v. expensive!
- Optimization 1: Sampling
  - Use of a scale factor (per query) to weigh the costs
  - Very small samples can lead to suboptimal partitions
  - What should the rate/threshold be?

LEARNING A PARTITIONING ADVISOR FOR CLOUD DATBASES
SIGMOD 2020

ONLINE TRAINING

- Optimization 2: Query Runtime Caching
  - Maintain a cache of query runtimes per partition
  - Given two states s_a, s_b, their partitions P_a, P_b, a query q_i
  - We need to estimate costs, only if q_i queries $t \in |P_b - P_a|$
- Optimization 3: Lazy Repartitioning
  - Repartitioning take time!
  - We partition only if q_i queries $t \in |P_b - P_a|$
- Optimization 4: Timeouts
  - If a query costs more in P_b, it is not a good partition!

WORKLOAD CHANGES

- Committee of Experts:
  - Different queries favor different optimal partitions
  - Poison the freq. vector to extract these “reference partitions”
  - A query belongs to subspace of a reference partition if $P_i = argmax_{P_i \in \{P_1 \dots Pn\}} - \sum_{j=1}^{m} f_j S_j * c_{sample}(P, q_j)$
  - Train an agent for each subspace
- Incremental Training:
  - New query frequencies added to input state
  - Runtime Cache speeds things up
  - Might not need to train new reference partition

TODAY'S AGENDA

- Refresher on Partitioning
- What is Microsoft up to?
- The Partitioning Problem
- Approach
- Results
- Parting Thoughts

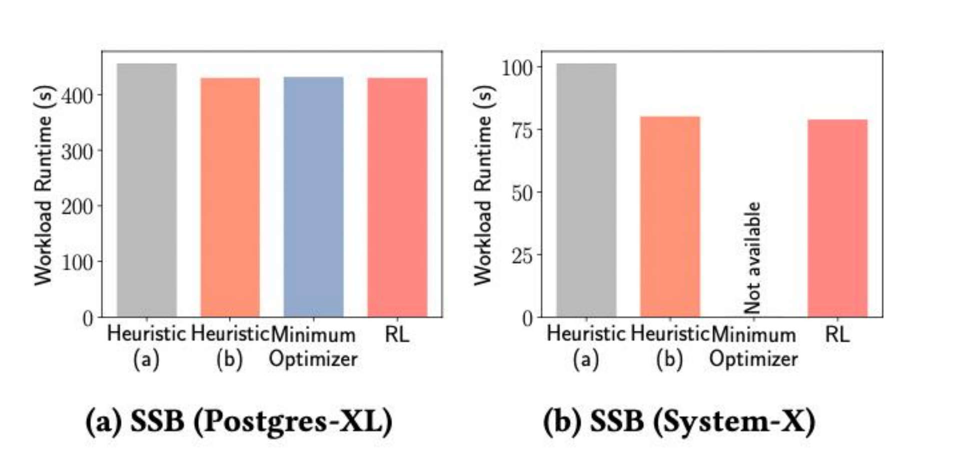
WORKLOADS AND SETUP

- Analytical Workloads:
  → Star Schema Benchmark
  → TPC-DS
  → TPC-CH

- Setup:
  → Postgres-XL (open source, disk-based)
  → System-X (commercial, memory-based)
  → 4-6 node clusters
  → 128GB RAM, 2*10-core Intel Xeon CPUs

BASELINE

- Compared to:
  - Heuristic A
  - Heuristic B
  - Minimum Optimizer



- Heuristic A: Most frequently joined dim. table
- Heuristic B: Largest table
- Minimum optimizer: Non ML opt. algorithm

OFFLINE TRAINING RESULTS

- For TPC-DS, DRL Agent suggests superior partitions
- For TPC-CH, DRL Agent optimizes for network costs, leading to the difference in runtimes

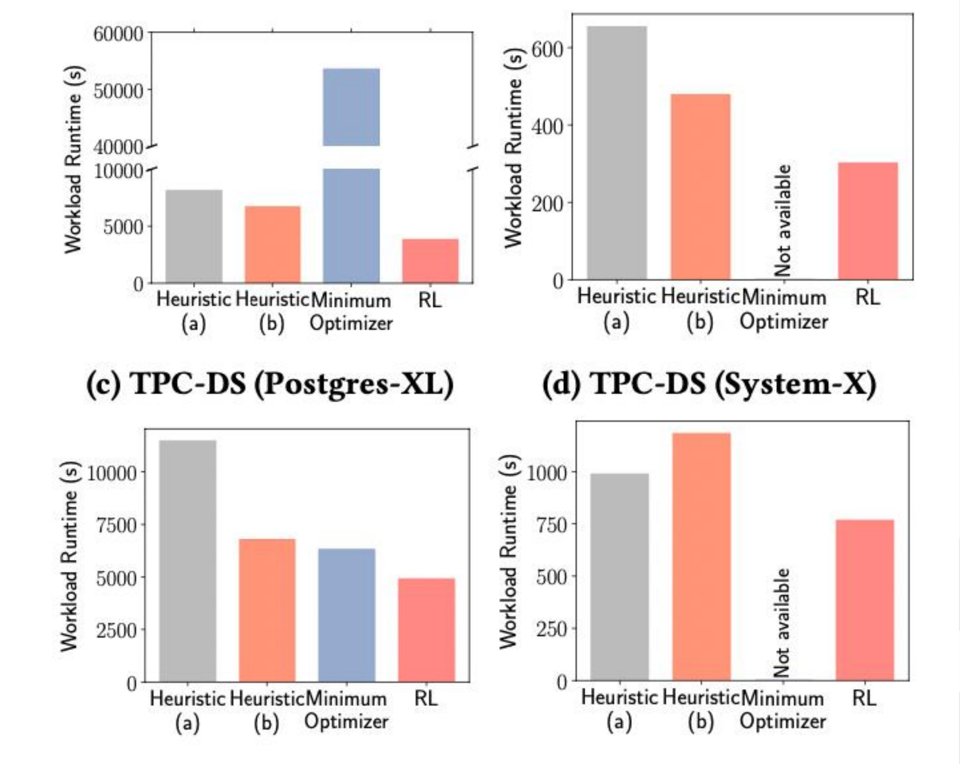
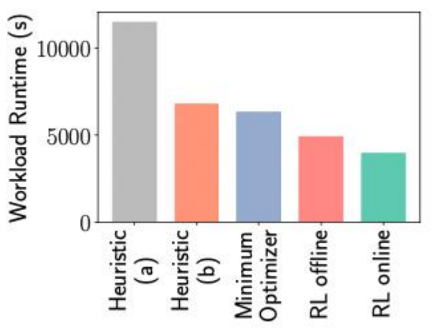


Figure 3: Offline RL vs. Baselines.

ONLINE TRAINING RESULTS

- Online training has better results because of a difference in cost model



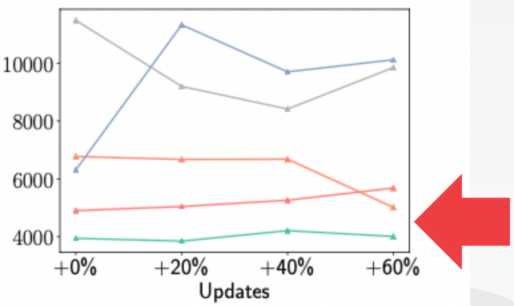
(a) TPC-CH

Optimizations	Training Time	Speedup
None	4621h	-
+ Runtime Cache	1160.4h	4.0
+ Lazy Repartitioning	60h	19.3
+ Timeouts	33.4h	1.8
+ Offline Phase	13.3h	2.5

Table 2: Training Time Reduction of Optimizations.

ADAPTIVITY TO CHANGE

- New data – Advisor performs fine for “non-significant” changes to data. Requires re-training otherwise.
- Changing Workload Mix – Committee of experts performs well.
- New queries – Requires incremental training; bootstrapped agent with low exploration.



(b) TPC-CH with Updates

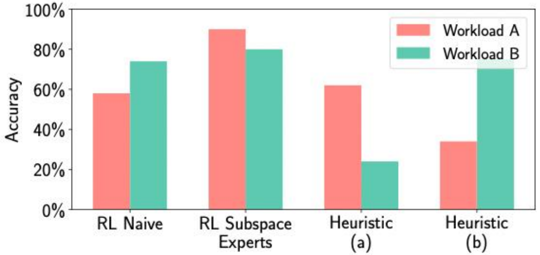
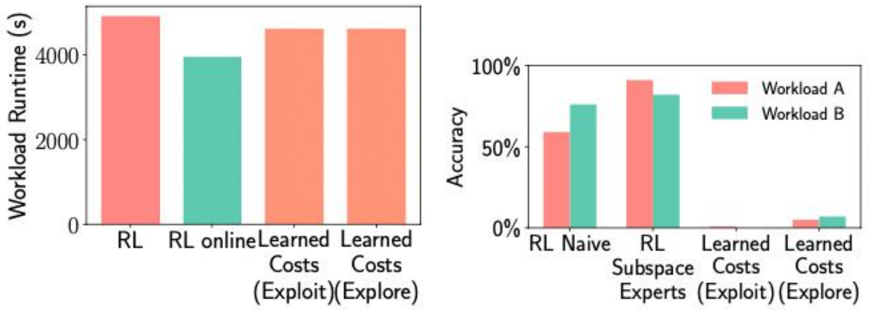


Figure 5: Best Partitioning found by Different Approaches for Varying Workloads (higher is better).

OTHER ML APPROACHES: COMPARISON

- Compared with NEO, a learned query optimizer

- Exp 1: Static Workload
- Exp 2: Workload Adaptivity



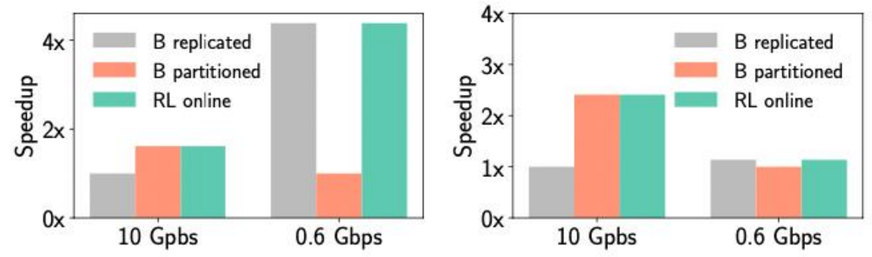
(a) TPC-CH Schema (b) Workload Adaptivity

Figure 7: RL vs. Neural Baselines.

NEO: A LEARNED QUERY OPTIMIZER
VLDB 2019

ADAPTIVITY TO DEPLOYMENT

- Run on System X, to avoid disk access costs
- Compares trade-off between Compute (co-location) and Network (parallelism)



(a) Standard HW (b) Slower Compute

Figure 8: Adaptivity to Deployment.

PARTING THOUGHTS

- Subset of the Partitioning Problem
  - → Only Horizontal, Hash Partitions
  - → Only evaluated on OLAP workloads
- Choice of Network Model
- Susceptible to Schema Changes?
- Accuracy as a Heuristic in Experiments
- Heterogenous Architecture/Hardware

NEXT CLASS

- Lin Ma's paper on Workload Modeling