Learned Cost Models for Query Optimization: From Batch to Streaming Systems

VLDB 2025

Roman Heinrich
TU Darmstadt & DFKI

Xiao Li
ITU Copenhagen

Manisha Luthra Agnihotri
TU Darmstadt & DFKI

Zoi Kaoudi
ITU Copenhagen

TECHNISCHE UNIVERSITÄT DARMSTADT

dfki ai Deutsches Forschungszentrum für Künstliche Intelligenz *German Research Center for Artificial Intelligence*

IT UNIVERSITY OF COPENHAGEN

Presenters

Dr. Manisha Luthra Agnihotri

Athene Young Investigator (TU Darmstadt) & *Deputy Head* (DFKI GmbH)

Research focus:

- AI-enhanced streaming systems
- Multimodal streaming
- Benchmarking AI-enhanced streaming

Prof. Zoi Kaoudi

Associate Professor
(IT University of Copenhagen)

Research focus:

- ML-based query optimization
- Cross-engine data systems

Roman Heinrich

PhD Student
(TU Darmstadt and DFKI GmbH)

Research focus

- Learned cost models &
- Query optimization of data systems

Dr. Xiao Li

Postdoc
(IT University of Copenhagen)

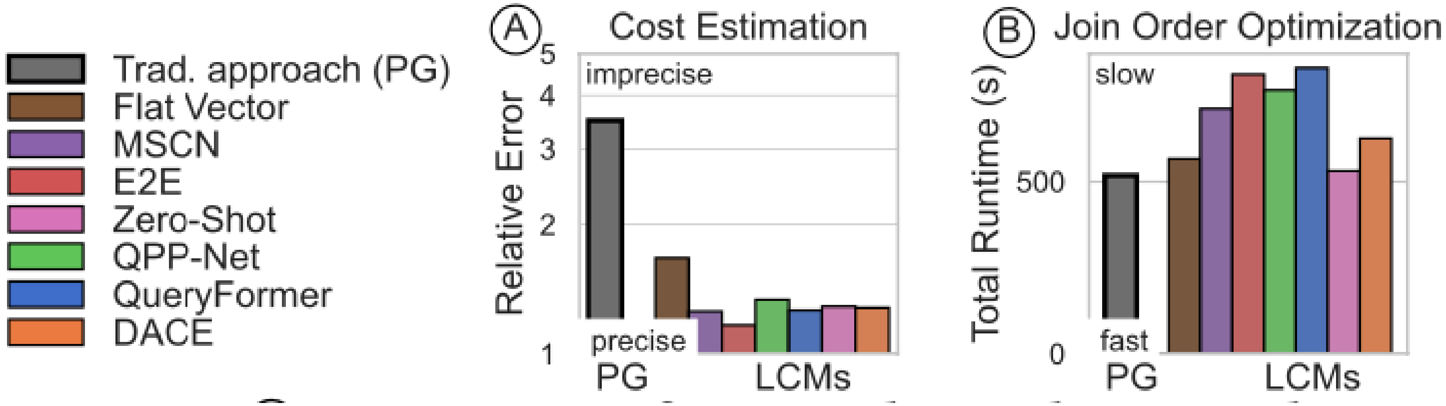
Research focus

- Query optimization with ML
- Data cleaning with ML

Why this Tutorial?

*No **unified** (batch & stream systems) **overview** of **Learned Cost Models for query optimizers** yet!*

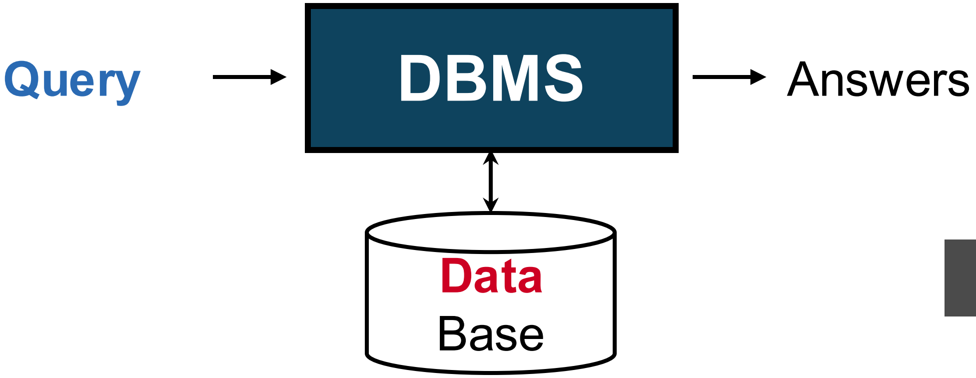
Spoiler: Batch systems



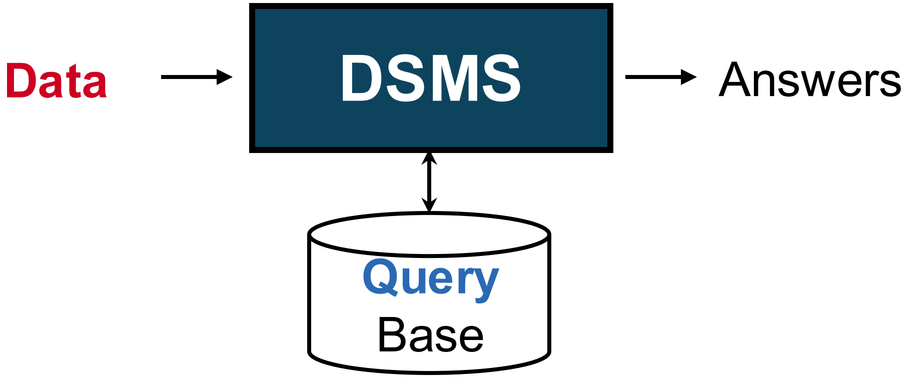
Heinrich R, Luthra M, Wehrstein J, Kornmayer H, Binnig C: How Good are Learned Cost Models, Really? Insights from Query Optimization Tasks, SIGMOD 2025

What are **Batch and Stream Systems?**

Database Management System



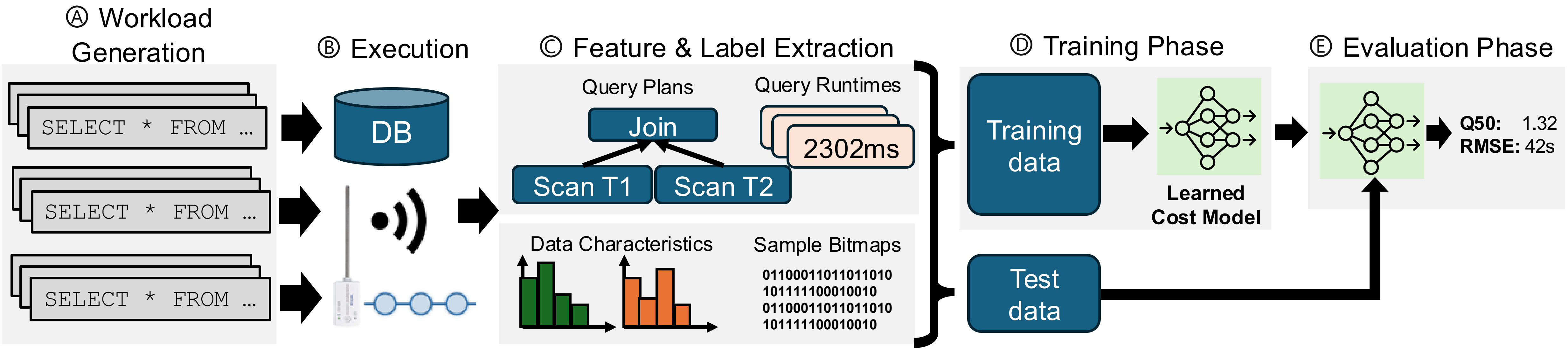
Data Stream Management System



Differences in workloads mean very different requirements for learned cost models

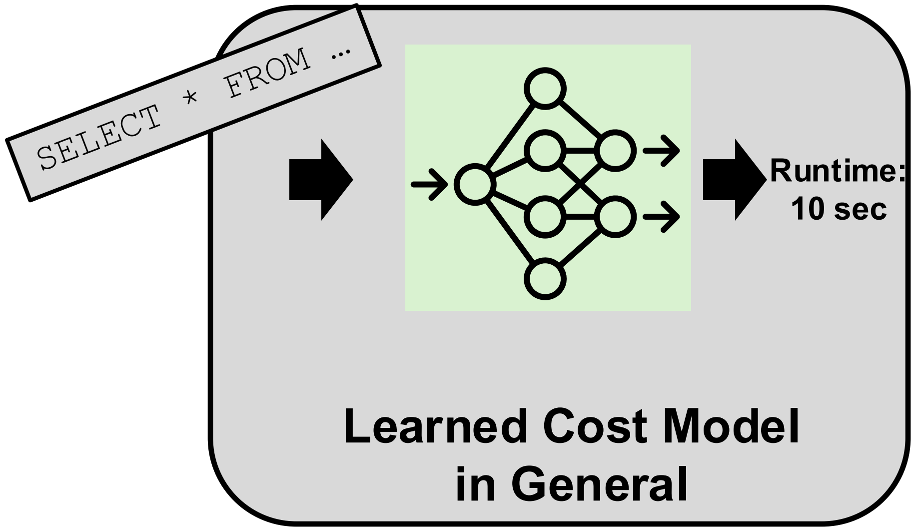
Emergence of Learned Cost Models

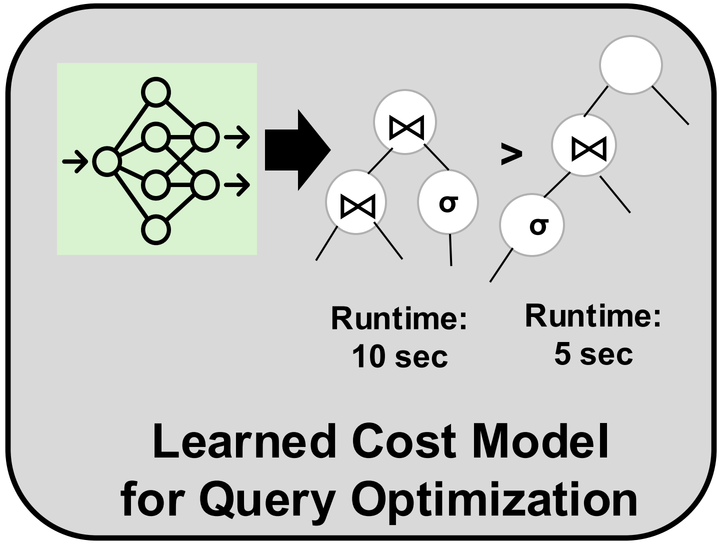
- **Learned cost models: powerful tool** overcoming limitations of traditional cost models
- Key Idea: **Instead** of relying on **hand-crafted analytical models**, let **data and ML guide the estimation**



Heinrich R, Luthra M, Wehrstein J, Kornmayer H, Binnig C: How Good are Learned Cost Models, Really? Insights from Query Optimization Tasks, SIGMOD 2025

Our Tutorial: **Cost Models (in) Query Optimizers**





For both Batch and Streaming Systems

Agenda

- LCMs in Batch Systems
- LCMs in Streaming Systems
- Road Ahead

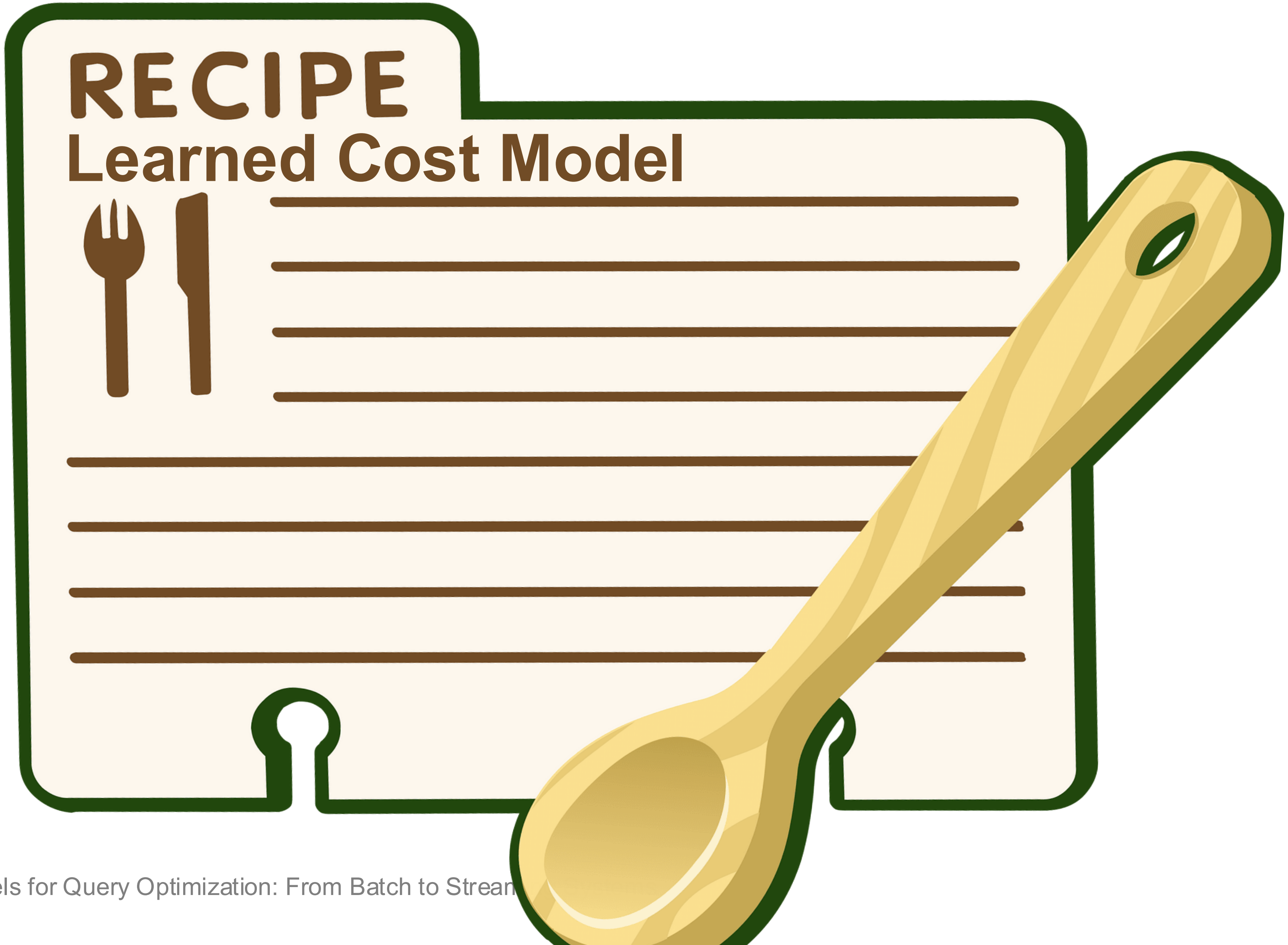
Agenda

- **LCMs in Batch Systems**

- LCMs in Streaming Systems

- Road Ahead

What's the ingredients of **Learned Cost Models?**



Learned Cost Models Ingredients

Featurization

- [ ] Featurization

ML Model

- [ ] ML Model

Training data collection

- [ ] Training data collection

What features can we build?

Featurization

- Query Encoding
- Plan Encoding
- Cardinality/Cost Estimates
- DB Statistics

Query encoding

Featurization

- Given a query SQL statement, convert it into a feature vector, e.g.,

```
SELECT * FROM A, B, C
WHERE A.a = B.a AND B.b = C.b AND A.a < 51
GroupBy C.c OrderBy A.a;
```

```
[0, 0, 1, …, 1, 0, 0.25, 0.73, 1, 0, 0, 0, …….]
```

Elements in a query

Featurization

```
SELECT * FROM A, B, C
WHERE A.a = B.a AND B.b = C.b AND A.a < 51
GroupBy C.c OrderBy A.a;
```

- Major elements in a query to consider
  - tables, columns, predicates, joins, aggregator (group by) /sorter (order by)

- Most of these elements are categorical variables
  - one-hot
  - multi-hot
  - learnable embedding

Encode elements of a query

Featurization

```
SELECT * FROM A, B, C
WHERE A.a = B.a AND B.b = C.b AND A.a < 51
GroupBy C.c OrderBy A.a;
```

Tables/columns: one-hot/multi-hot encoding

- table set: `[A, B, C, D]` => table “`A`”: `[1, 0, 0, 0]`
- column set: `[A.a, A.b, A.c, B.a, …]` => column “`A.a`”: `[1, 0, 0, 0, ….]`
- if multiple tables/columns are involved, multi-hot is used
  - `[A, B]` => `[1, 1, 0, 0]`
  - `[A.a, A.c]` => `[1, 0, 1, 0, 0, 0, …]`

Encode elements of a query

Featurization

```
SELECT * FROM A, B, C
WHERE A.a = B.a AND B.b = C.b AND A.a < 51
GroupBy C.c OrderBy A.a;
```

Predicates (<*column, predicate operator, value*> *triplet*): e.g., "`A.a < 51`"

- concatenate: one-hot column + one-hot operator + value, or
- directly one-hot encode the existence of a column predicate

A.a	A.b	A.c	B.a	…
1	0	0	0	…

column predicates

- obtain estimated selectivity of column predicates with histogram or bitmap
  - e.g., "`[1, 0, 0, 0, …, 1, …]`" replaced by "`[0.55, 0, 0, 0, …, 0.76, …]`"

- semantic embedding: map a predicate to an embedding vector

Encode elements of a query

Featurization

```
SELECT * FROM A, B, C
WHERE A.a = B.a AND B.b = C.b AND A.a < 51
GroupBy C.c OrderBy A.a;
```

- Joins: e.g., "`A.a = B.a`"
  - directly one-hot encode the joins: `[0,1]`
  - concatenate "`a`", "`A`", and "`B`"'s representation
  - embedding

- `Groupby` or `Orderby` operators
  - boolean indicator: 0/1 => the sql includes the operator or not

```
SELECT * FROM A, B, C
WHERE A.a = B.a AND B.b = C.b AND A.a < 51
GroupBy C.c OrderBy A.a;
```

Encode a query

Featurization

Encode a query:

- use element features individually

SELECT COUNT(*) FROM title t, movie_companies mc WHERE t.id = mc.movie_id AND t.production_year > 2010 AND mc.company_id = 5

Table set {[0 1 0 1 ... 0], [0 0 1 0 ... 1]} Join set {[0 0 1 0]} Predicate set {[1 0 0 0 0 1 0 0 0.72], [0 0 0 1 0 0 1 0 0.14]}

table id samples join id column id value operator id

- or concatenate the representation of considered elements

There is no consideration of the query/join graph structure!

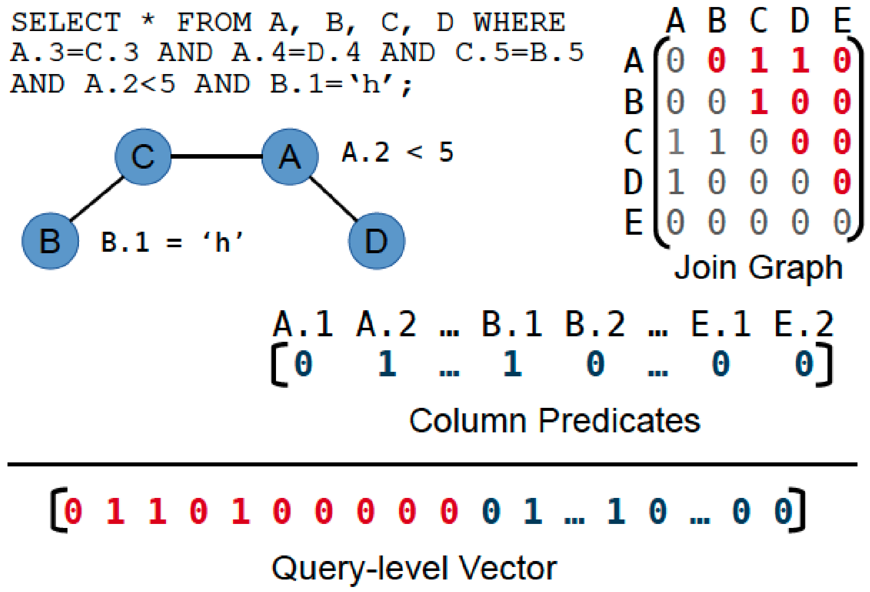
Andreas Kipf et al. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. CIDR 2019.

Encode a query

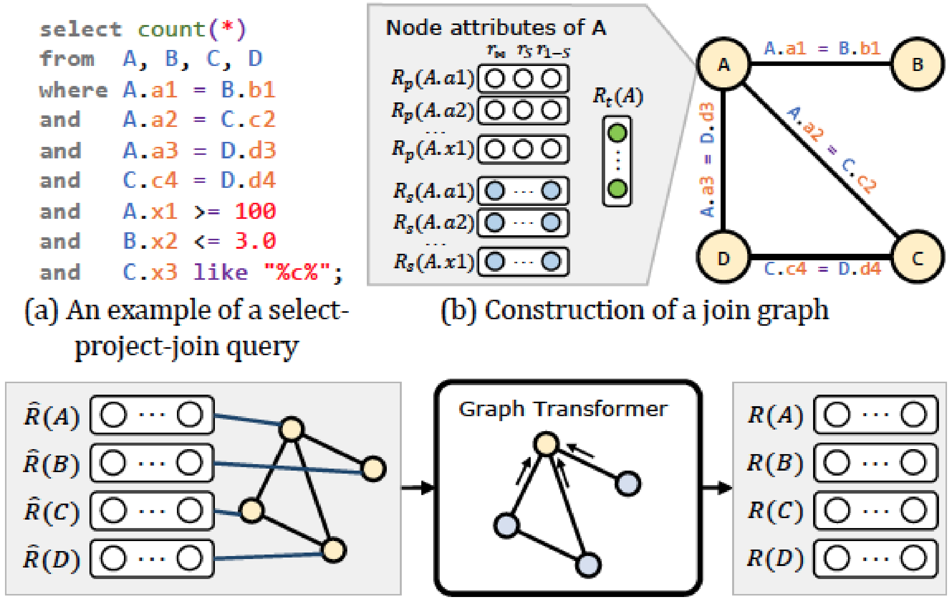
Featurization

Join graph: a table as a node and a join (e.g., A.a = B.a) as an edge

Adjacency matrix



Graph embedding



(a) An example of a select-project-join query (b) Construction of a join graph

Embed the structure info of join graph into the query encoding!

Ryan Marcus et al. NEO: A Learned Query Optimizer. VLDB 2019.

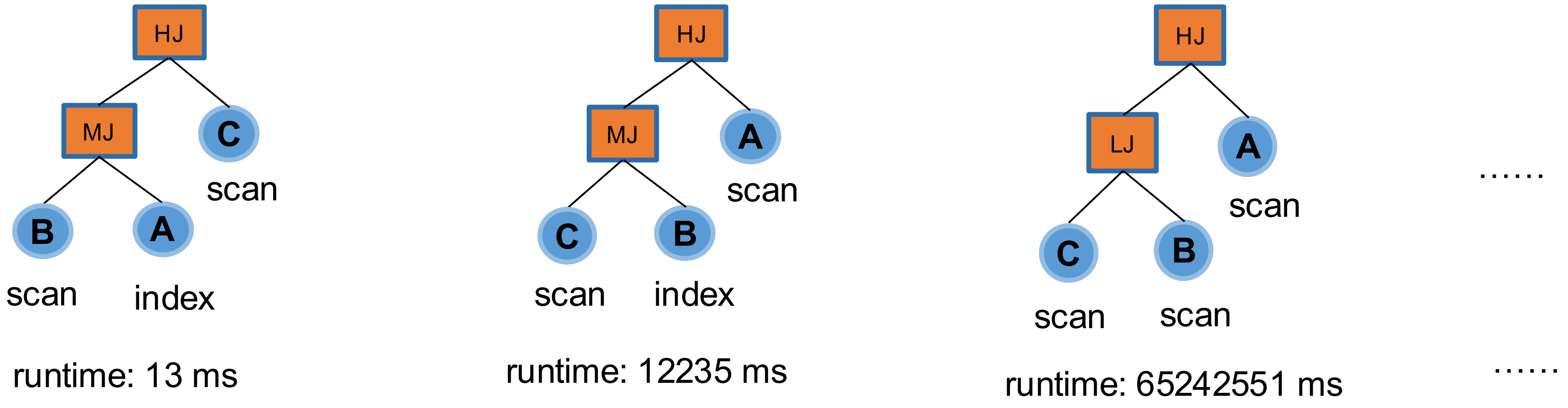
Tianyi Chen et al. LOGER: A Learned Optimizer Towards Generating Efficient and Robust Query Execution Plans. VLDB2023.

Plan encoding

Featurization

```
SELECT * FROM A, B, C
WHERE A.a = B.a AND B.b = C.b AND A.a < 51
GroupBy C.c OrderBy A.a;
```

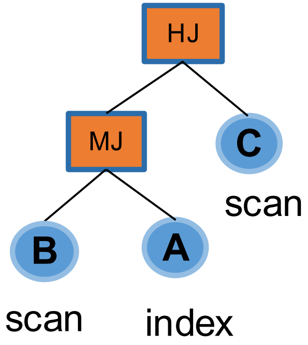
DBMS execute the query as per a tree-structured query plan.



It is the execution plan that determines the cost of executing a query!

How to encode a query plan

Featurization



Major elements in a query plan:

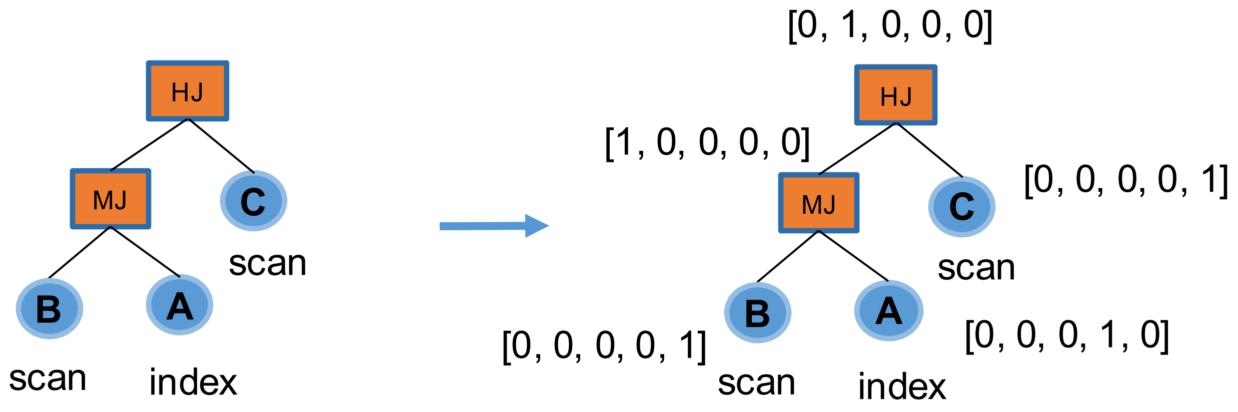
- operators, e.g.,
  - join operator: e.g., hash join(HJ), merge join (MJ), loop join (LJ)
  - data access operator: e.g., index scan (index), seq scan (scan)
  - aggregate (hash/stream)
- tables and columns to be joined
- join order: embodied by the tree structure in a bottom-up way

How to encode a query plan

Featurization

Procedures to encode a query plan:

1) encode the node:
   - one-hot encode the operators
   - aggregate other node features such as tables/columns (discussed before)



How to encode a query plan

Featurization

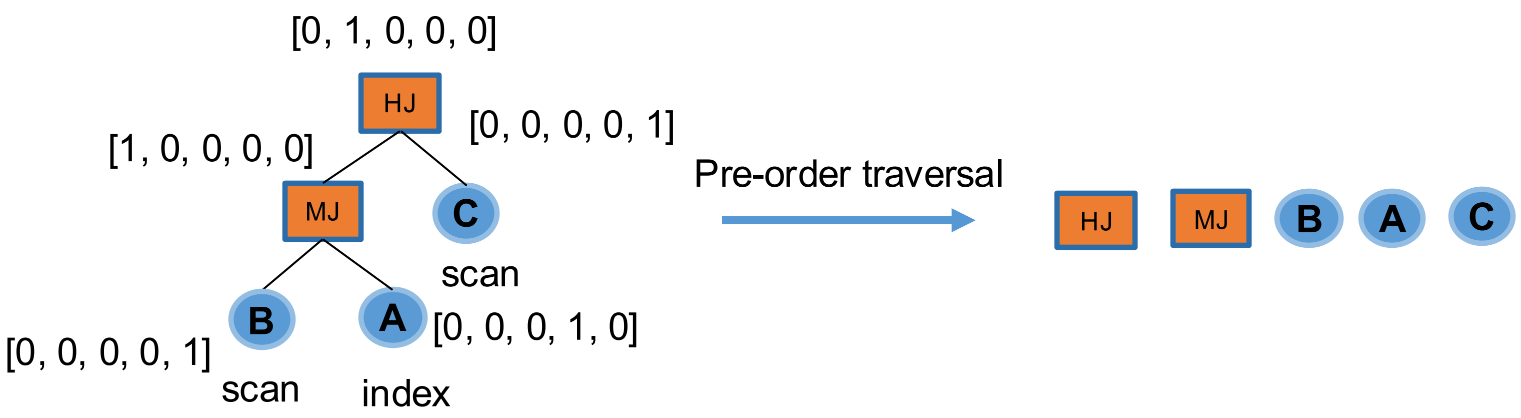
2) exploit the structure info (multiple choices).

- flatten it into a **vector** with tree traversing algorithm, e.g., depth first searching
- preserve the **tree** shape for the subsequent tree-based NN
- extend it to a more informative structure such as a **graph**

Examples of plan encoding

Featurization

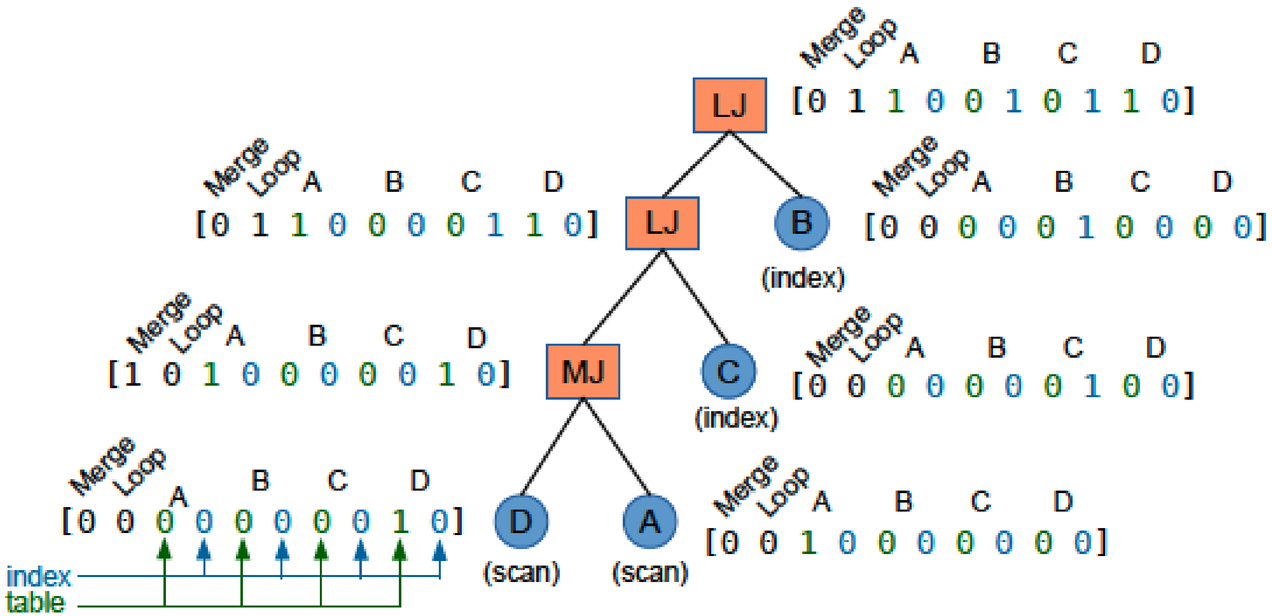
Flat vector



Examples of plan encoding

Featurization

Vector tree



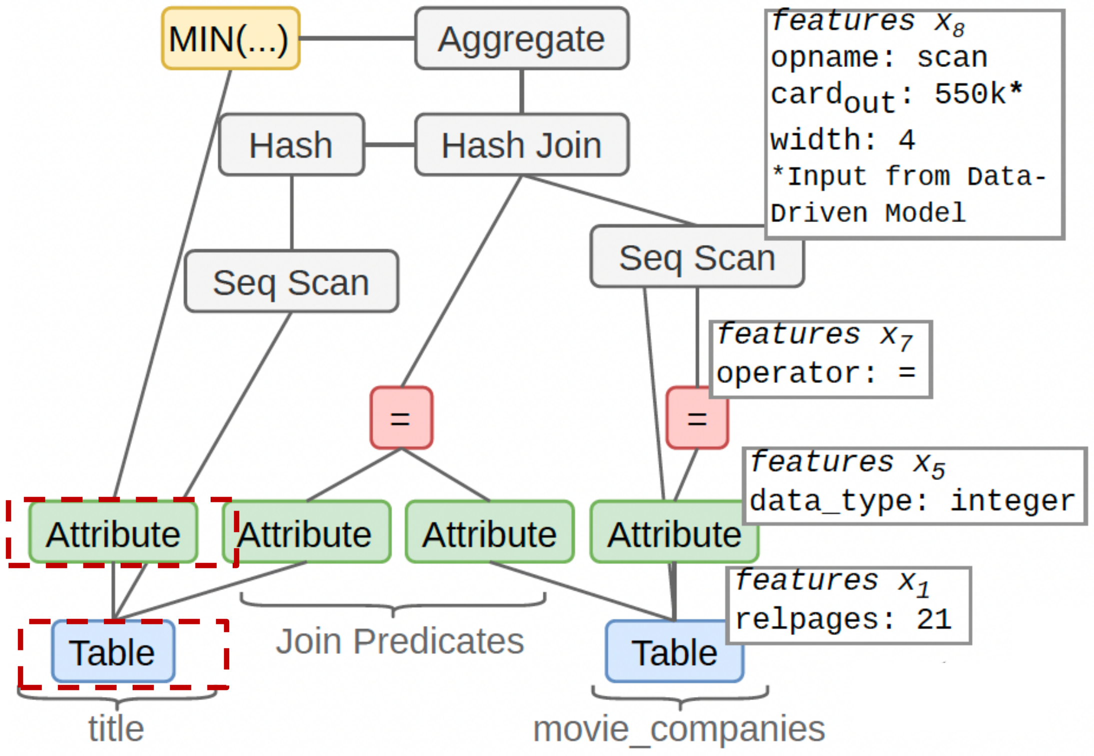
Keep the tree structure and take it into the subsequent tree neural network

Ryan Marcus et al. NEO: A Learned Query Optimizer. VLDB 2019.

Examples of plan encoding

Featurization

Graph



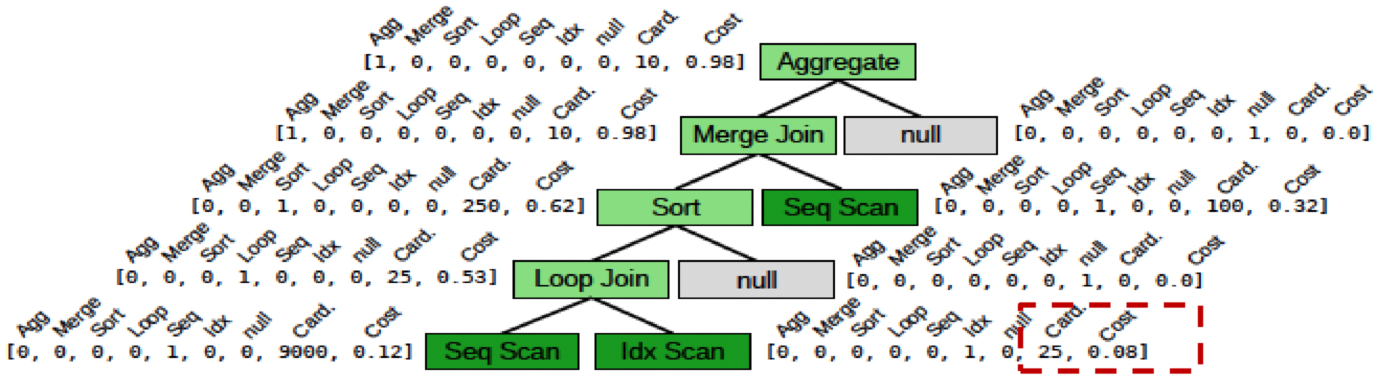
Besides the tree nodes, there are more nodes like attributes and tables.

Benjamin Hilprecht et al. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction. VLDB 2022.

Cardinality & cost estimates

Featurization

- Obtain these estimates from traditional cost-based optimizer
- Incorporate these estimates into the plan encoding



Bao takes cardinality and cost estimates into plan encoding

Ryan Marcus et al. Bao: Learning to Steer Query Optimizers. SIGMOD 2021.

DB statistics

Featurization

- Database statistics: histograms and sample bitmaps
  - combine their usage with table/column encoding or predicate encoding

- Other statistical features from DB
  - the number of rows in a table
  - the number of unique values in a column
  - the number of null values in a column
  - ….

Notes on featurization

Featurization

Transferability in the features:

- DB-specific features such as table/column name/identifier which cannot transfer across DBs
- transferable features such as card./cost estimates and statistics

Feature selection:

- some features may be redundant such as predicate encoding and cardinality estimates
- query encoding only / plan encoding only / both encoding

Plan encoding is often there!

Learned Cost Models Ingredients

- [x] Featurization
- [ ] ML Model
- [ ] Training data collection

What kinds of models can we build?

ML Model

- Regression task: the target is runtime (more common)
- The architecture of cost models are related to the shape of input features

- Cost model for flat features:
    - FlatVector [1] leverage a regression tree model
    - An multi-layer perceptron (MLP) even can do this task or more competent design, e.g., MSCN [2]

[1] Archana Ganapathi et al. Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning. ICDE 2009.

[2] Andreas Kipf et al. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. CIDR 2019.

Cost models for tree-shape features

ML Model

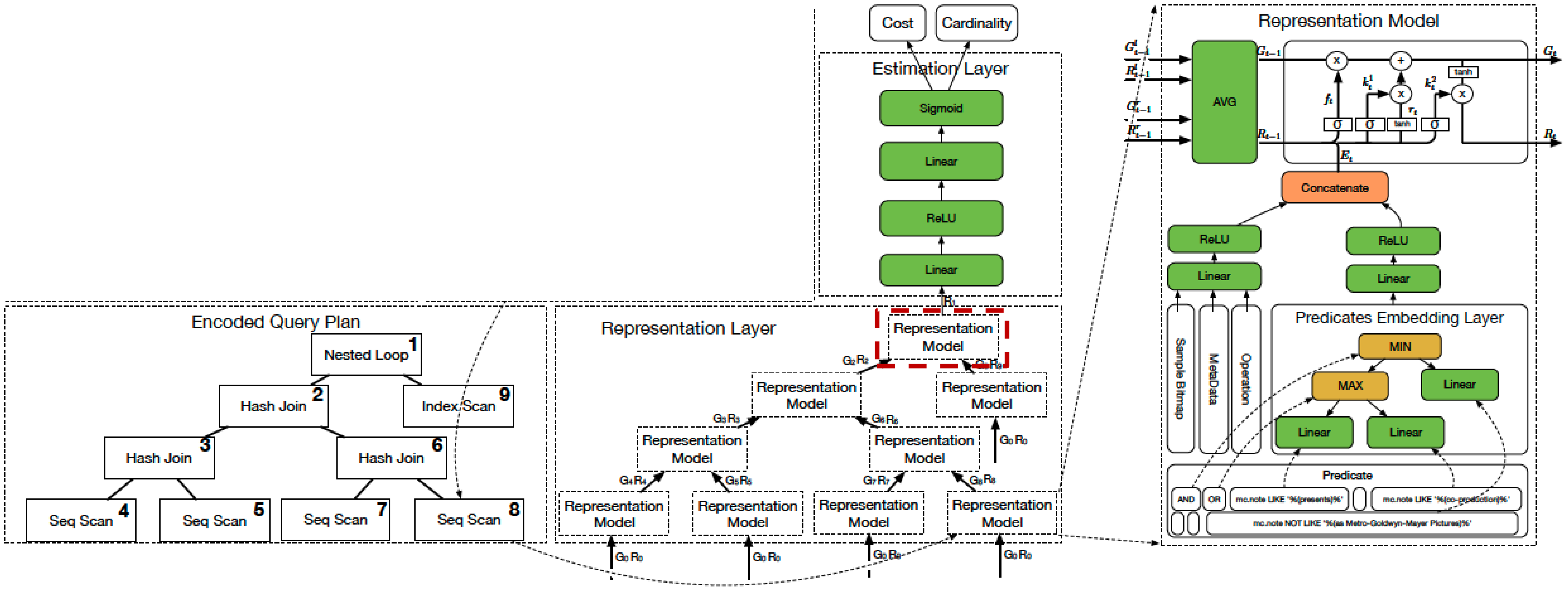
Cost model for tree-shape features:

- the model aims to capture the useful relations in the tree structure of the input
- the **popular** model architectures in LCMs: e.g.,
  - treeLSTM
  - treeCNN
  - tree-based transformer

Cost models for tree-shape features

ML Model

TreeLSTM



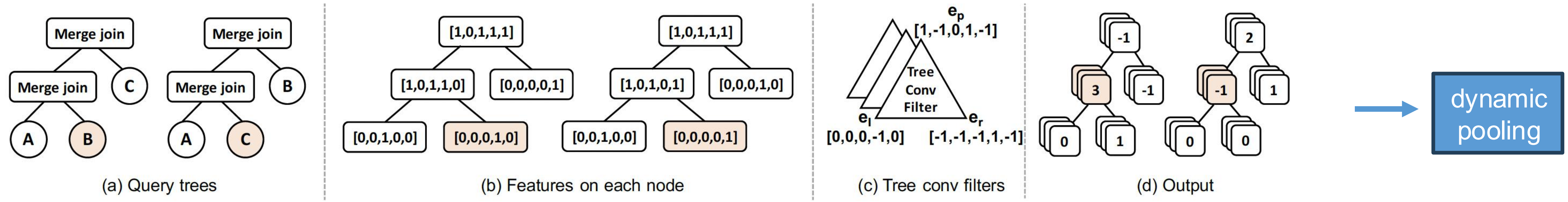
Each node in a plan tree as a LSTM unit and passing the state to their parent node in bottom-up way!

Ji Sun et al. An End-to-End Learning-based Cost Estimator. VLDB 2019.

Cost models for tree-shape features

ML Model

TreeCNN



Ryan Marcus et al. NEO: A Learned Query Optimizer. VLDB 2019.

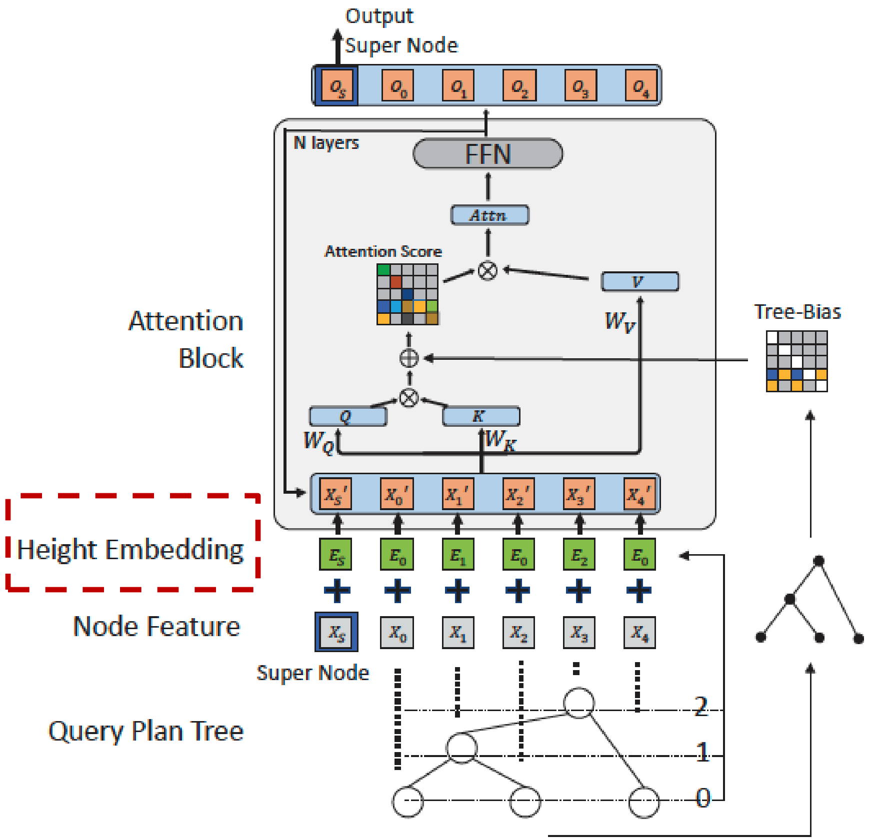
treeConv filters slide in the plan tree to get a convolved representation of vector tree

Cost models for tree-shape features

ML Model

Tree-transformer

Height embedding similar to position embedding in transformer records a node's position in a plan tree.

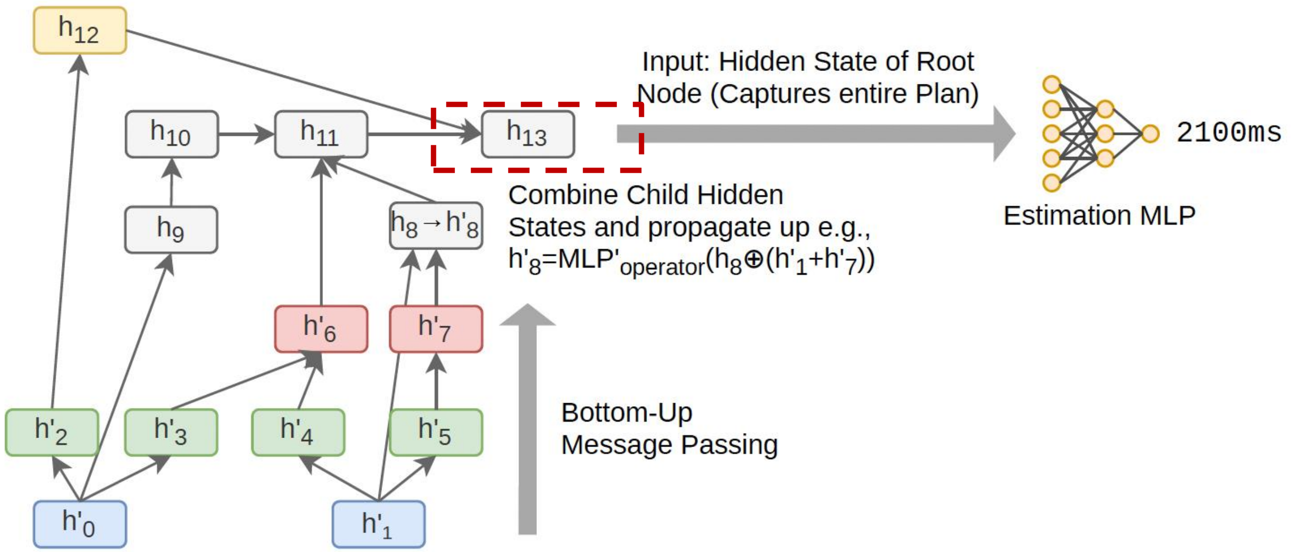


Yu Zhao et al. QueryFormer: A Tree Transformer Model for Query Plan Representation. VLDB 2022.

Cost models for graph-shape features

ML Model

Cost model for graph-structured features: graph neural network (GNN)



bottom-up message passing

Benjamin Hilprecht et al. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction. VLDB 2022.

General LCMs vs. LCMs for query optimizer

ML Model

Architecture differences

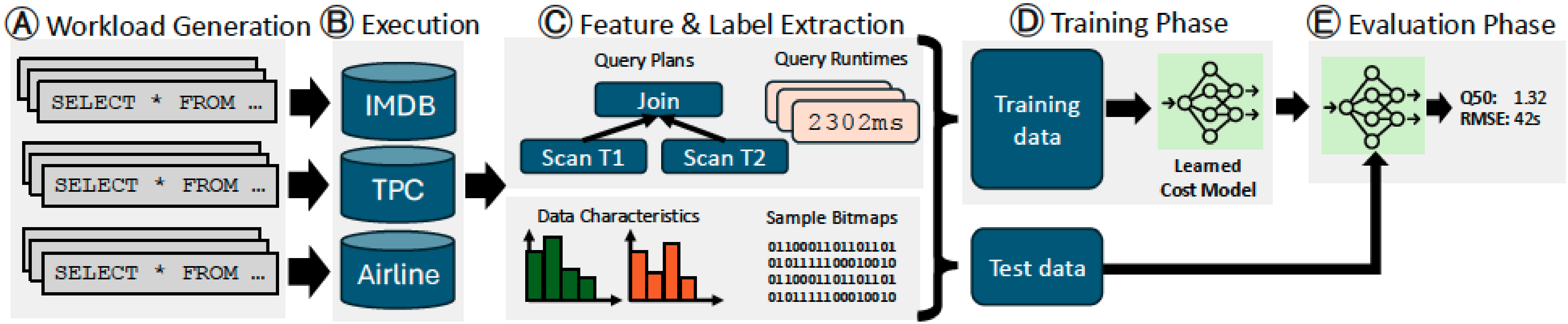
Research Topic	Existing Work	Architecture of Its Learned Cost Model
General LCMs	FlatVector MSCN End-to-End QPP-Net QueryFormer Zero-shot DACE	RegressionTree Deep multisets TreeLSTM TreeNN Transformer GNN Transformer
Learned Query Optimizer	NEO RTOS Bao Balsa HybridQO LEON LOGER	Tree-CNN Tree-LSTM Tree-CNN Tree-CNN Tree-LSTM Tree-CNN Tree-LSTM

General LCMs vary in the model architectures, while LCMs for query optimizers mostly use tree-CNN or tree-LSTM!

General LCMs vs. LCMs for query optimizer

ML Model

- Workflow differences
  - General LCMs:

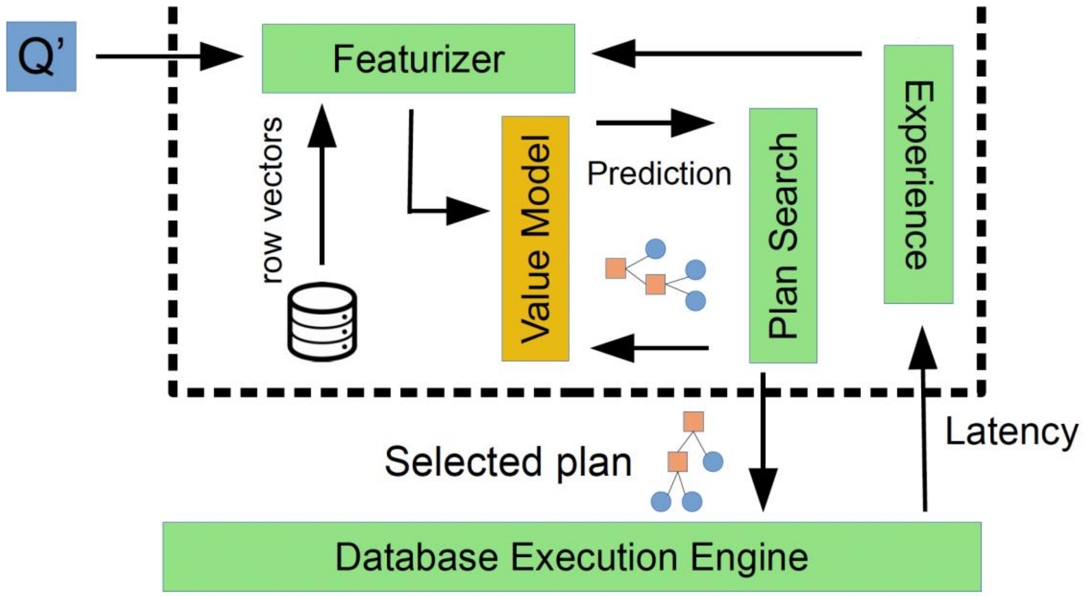


- follow typical **"two-stage"** working flow: i.e., training + testing
- training data and testing data are both **pre-optimized plans** obtained by DBMS
- the input of the model is **a complete plan** which corresponds to an input query

General LCMs vs. LCMs for query optimizer

ML Model

- LCMs for query optimizers



- works as a **value model**, embedded in RL framework
- the input plan to the model may be **not pre-optimized**
- the input plan to the model can be a **subplan**

Learned Cost Models Ingredients

- ☑ Featurization
- ☑ ML Model
- ☐ Training data collection

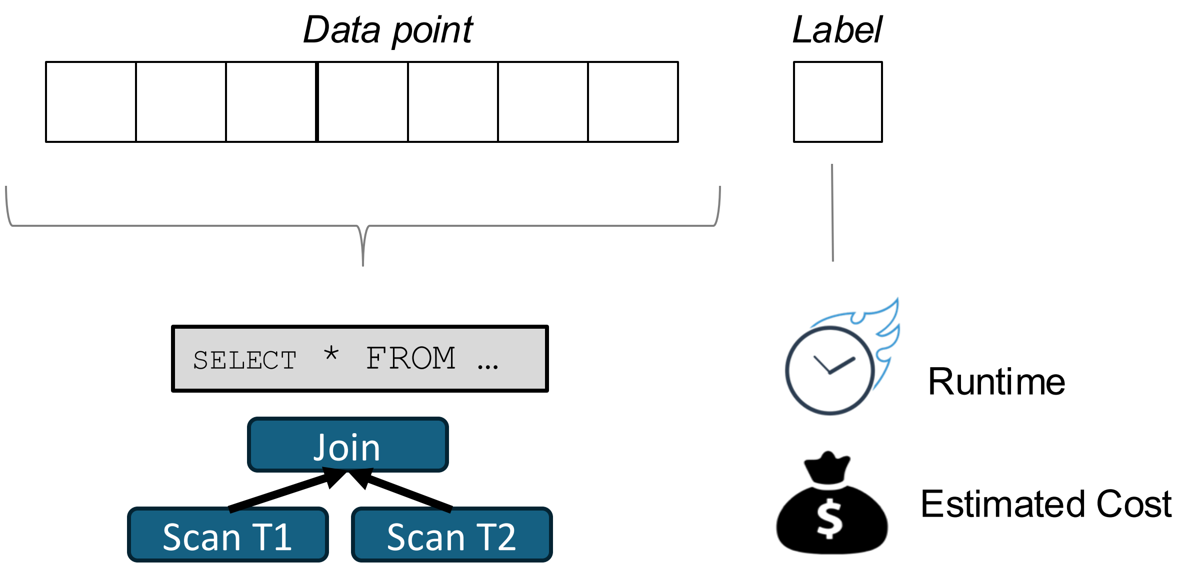
Training data vital for ML models

Training data collection

ML models are data-hungry

What is training data in LCMs?

Training data collection

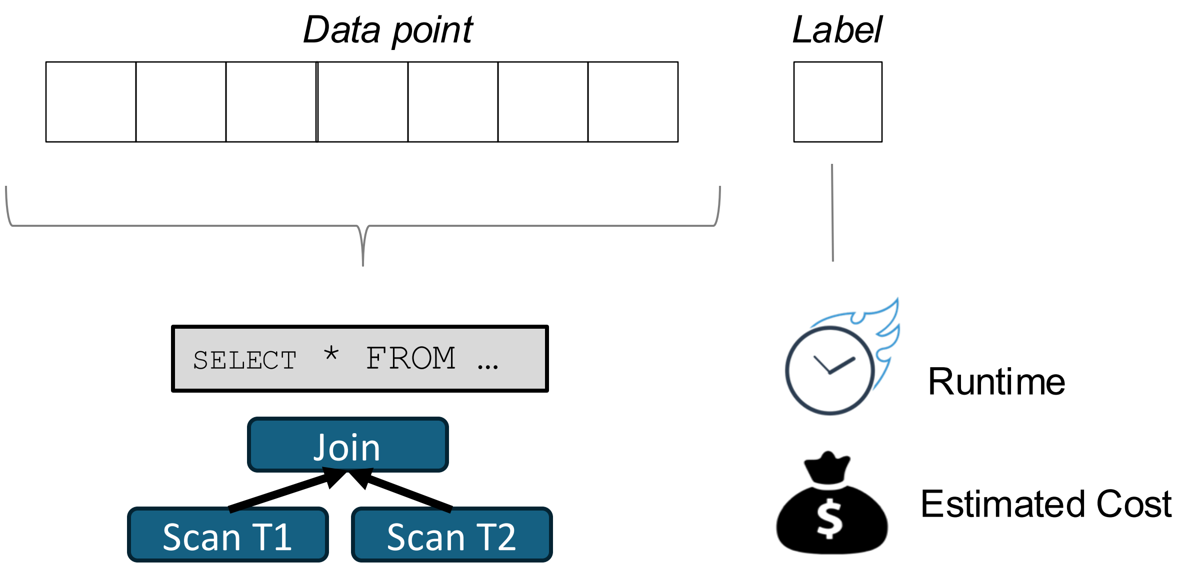


- How to find thousands of SQL queries and plans?

- How to obtain their label?

What is training data in LCMs?

Training data collection



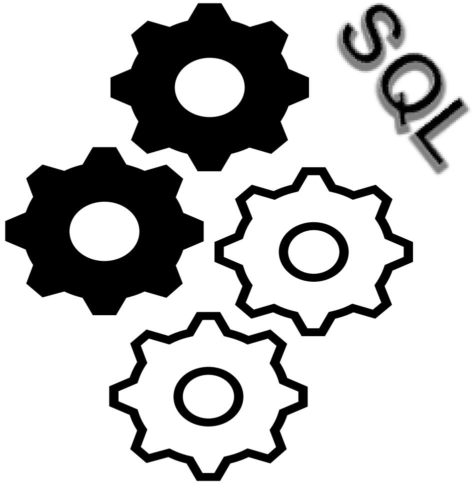
- **How to find thousands of SQL queries and plans?**

- How to obtain their label?

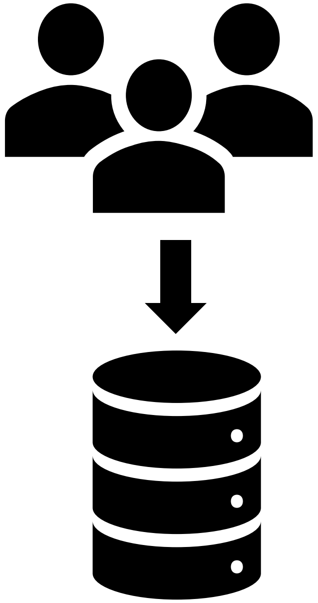
SQL queries

Training data collection

Benchmarks



Synthetic
query generators

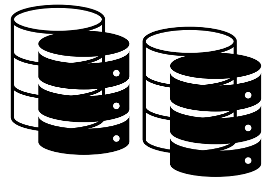


Real user queries

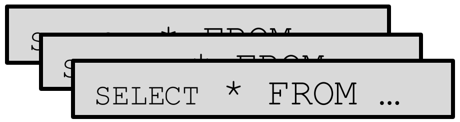
SQL query **generation** for zero-shot model

Training data collection

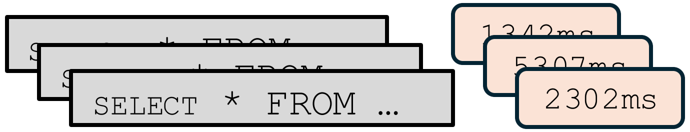
- **Data**: 20 databases from real-world datasets & benchmarks



- **Queries**:
  - Benchmark queries
  - Workload generator
    - Standard mode → SPJA queries with conjunctive predicates
    - Complex mode → SPJA with disjunctive complex predicates (e.g., IN)
    - Index mode → random indices in foreign keys and predicate columns



- **Bonus**: Workload traces (queries with runtimes)



B. Hilprecht et al. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction. VLDB 2022.

SQL query **collection** in Amazon Redshift

Training data collection

- Training data collected as queries run in **production**
- **Sliding window** (one query in, one out)

Problem

Mostly short-running queries

↓

Catastrophic predictions for **long-running** queries

Solution

Partition training set into buckets, e.g.:

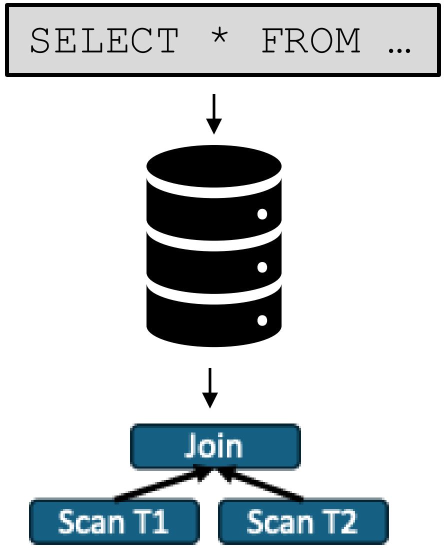
- Bucket 1: 0-10 sec
- Bucket 2: 10-30 sec etc.

Training happens in the production cluster!

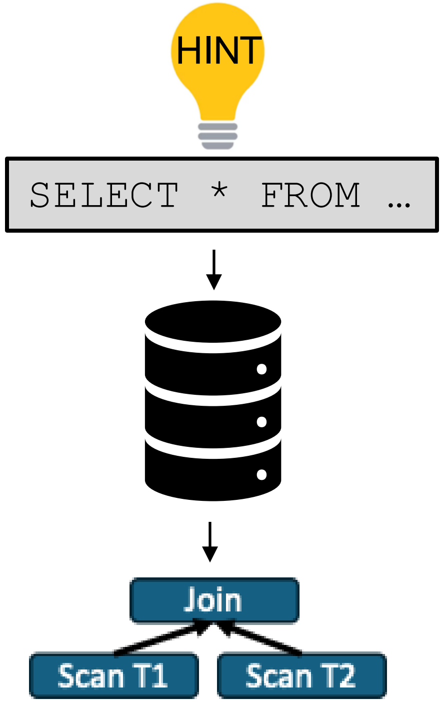
G. Saxena et al. Auto-WLM: Machine Learning Enhanced Workload Management in Amazon Redshift. SIGMOD 2023.

Query plans from SQL workloads

Training data collection

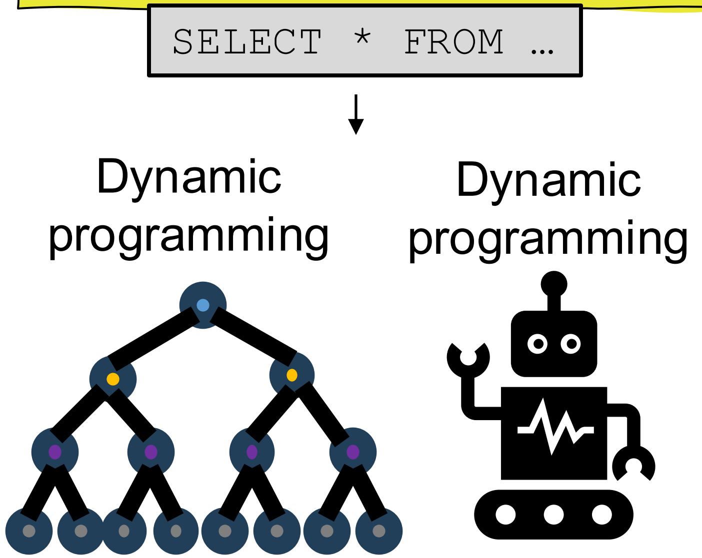


SQL query execution



SQL queries with hints

For LCMs in QO only



Balsa, LTR

Neo, Bao, LOGER

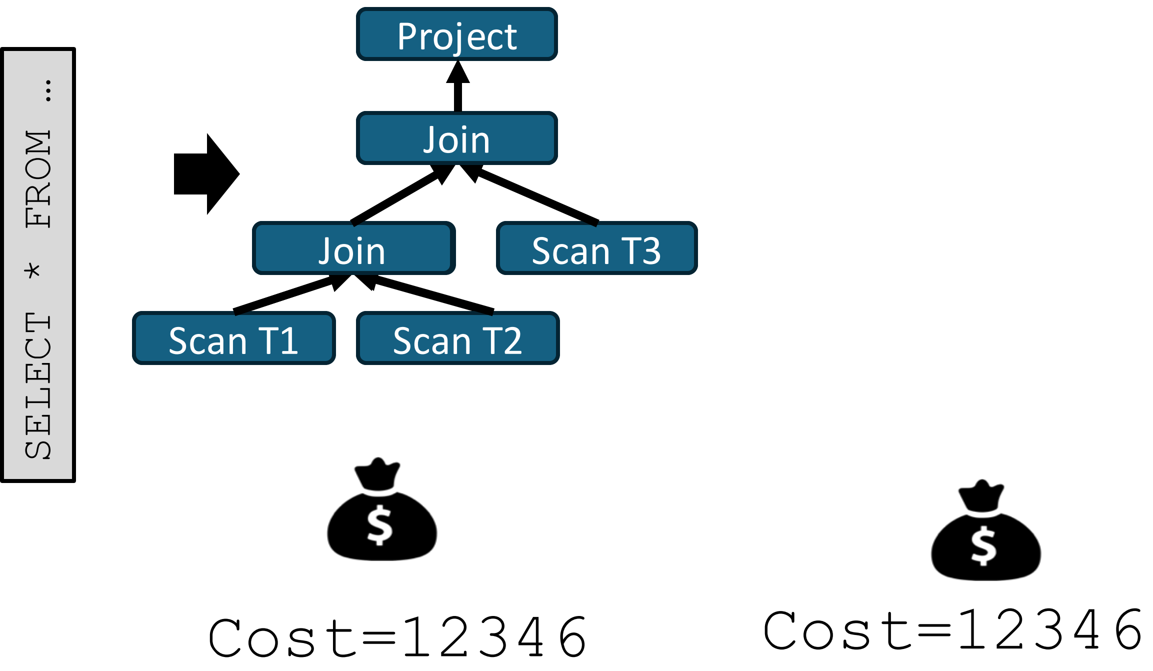
Plan enumeration

[Balsa] Z. Yang et al. Balsa: Learning a Query Optimizer Without Expert Demonstrations. SIGMOD 2022.
[LTR] H. Behr et al. Learn What Really Matters: A Learning-to-Rank Approach for ML-based Query Optimization. BTW 2023.
[Neo] R. Marcus et al. Neo: a learned query optimizer. PVLDB 2019
[Bao] R. Marcus et al. Bao: Making Learned Query Optimization Practical. SIGMOD 2021.
[LOGER] T. Chen et al. LOGER: A Learned Optimizer Towards Generating Efficient and Robust Query Execution Plans. PVLDB 2023

Query plan **augmentation**

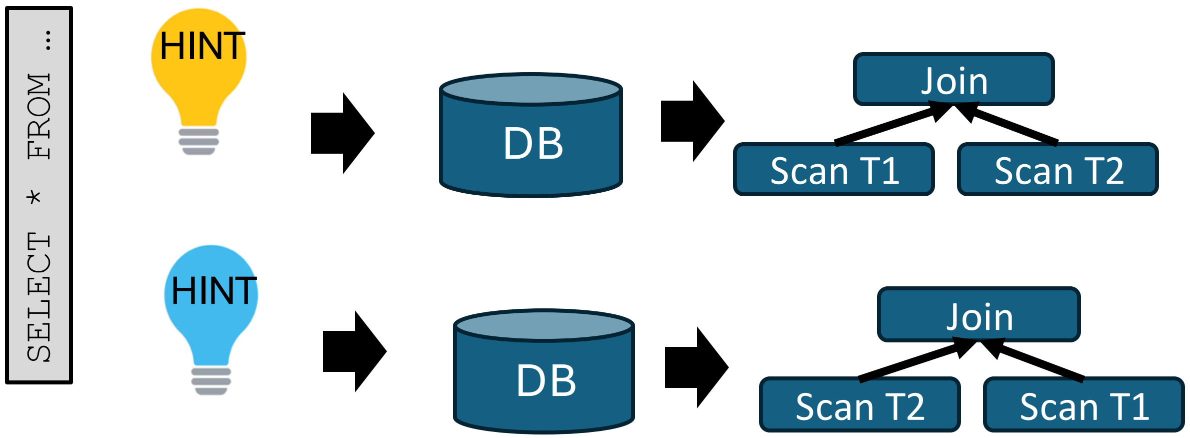
Training data collection

Use subplans



*[Balsa] Z. Yang et al. **Balsa**: Learning a Query Optimizer Without Expert Demonstrations. SIGMOD 2022.*

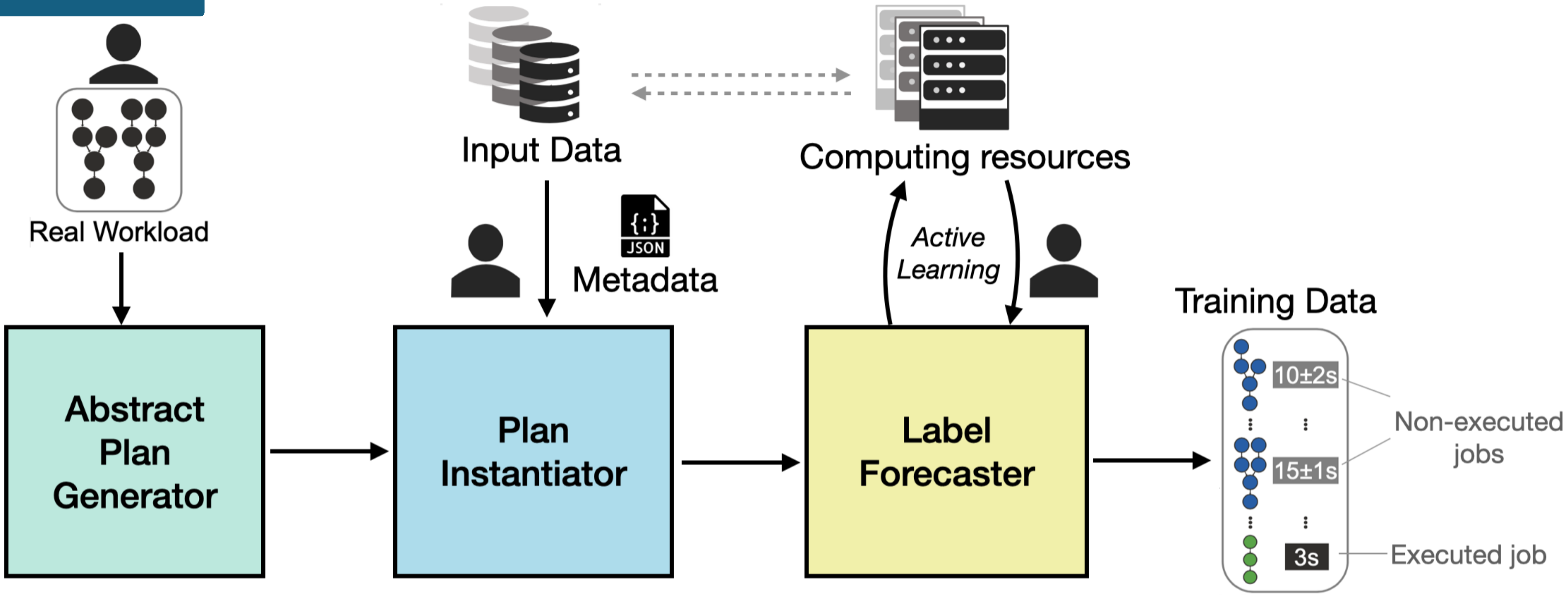
Change cardinalities



*R. Zhu et al. Balsa: **Lero**: A Learning-to-Rank Query Optimizer. PVLDB 2023.*

Synthetic plan generation with DataFarm

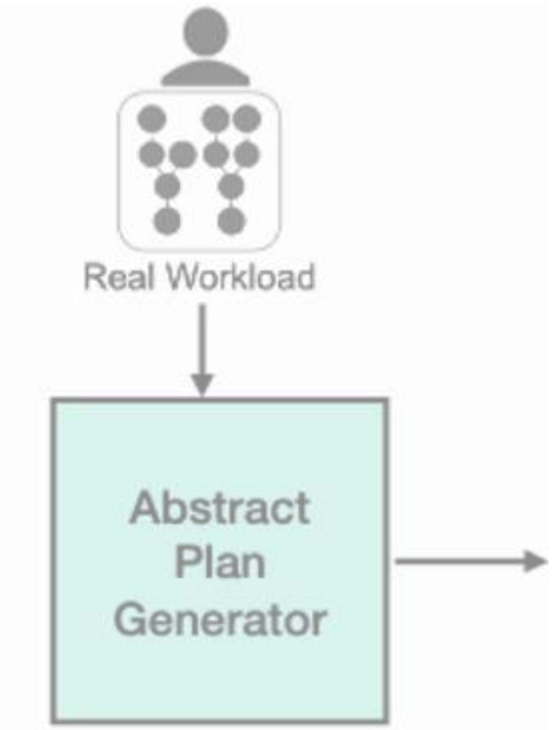
Training data collection

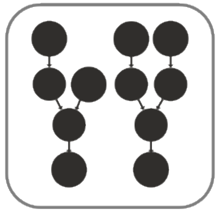


F. Ventura et al. Expand your Training Limits! Generating Training Data for ML-based Data Management. SIGMOD 2021.
R. van de Water. DataFarm: Farm Your ML-based Query Optimizer's Food! – Human-Guided Training Data Generation. CIDR 2022
R. van de Water. Farming Your ML-based Query Optimizer's Food. ICDE 2022 (best demo award)

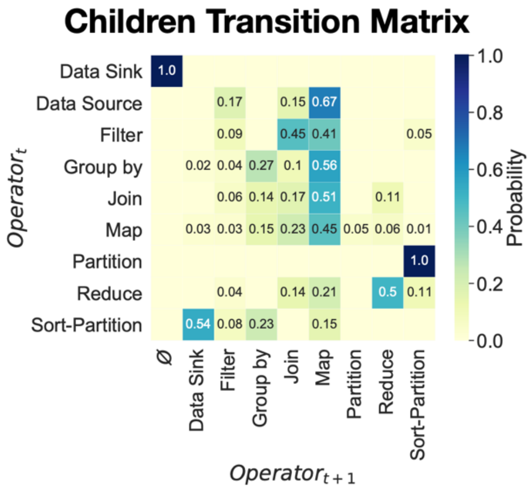
Synthetic plan generation with DataFarm

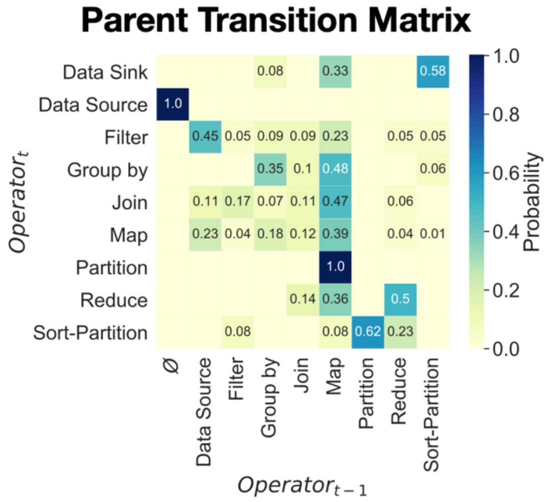
Training data collection





Real Workload





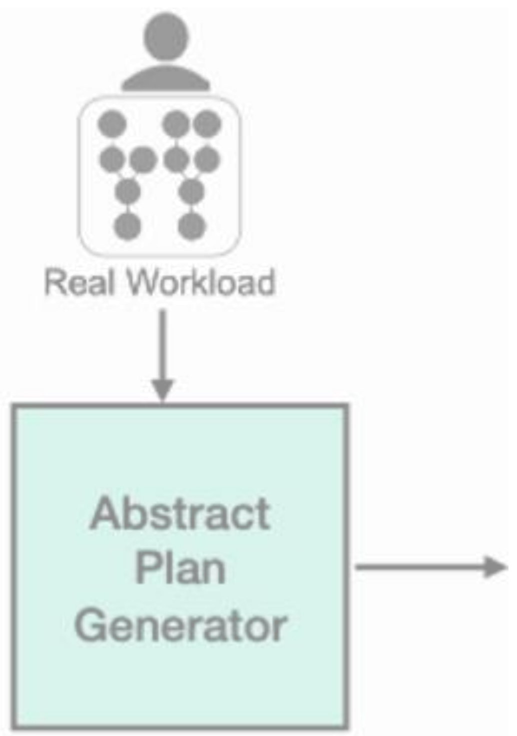
Learns real execution patterns

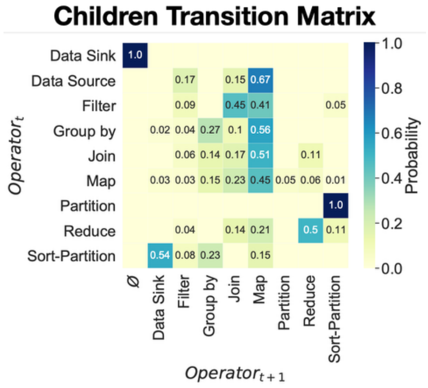
Generates new representative plans

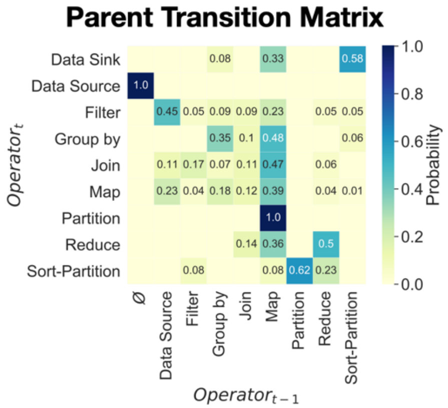
F. Ventura et al. Expand your Training Limits! Generating Training Data for ML-based Data Management. SIGMOD 2021.
R. van de Water. DataFarm: Farm Your ML-based Query Optimizer's Food! – Human-Guided Training Data Generation. CIDR 2022
R. van de Water. Farming Your ML-based Query Optimizer's Food. ICDE 2022 (best demo award)

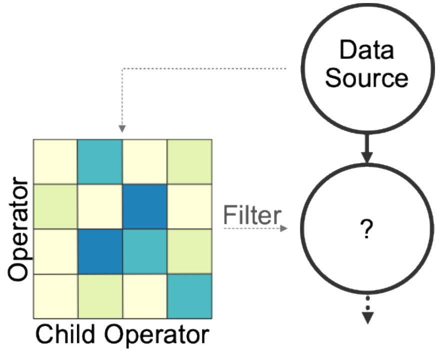
Synthetic plan generation with DataFarm

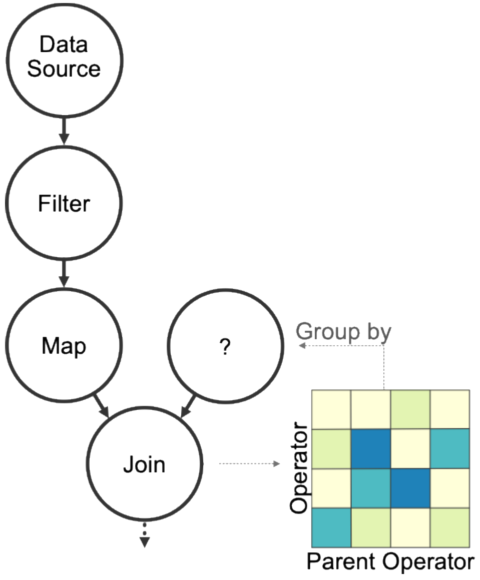
Training data collection







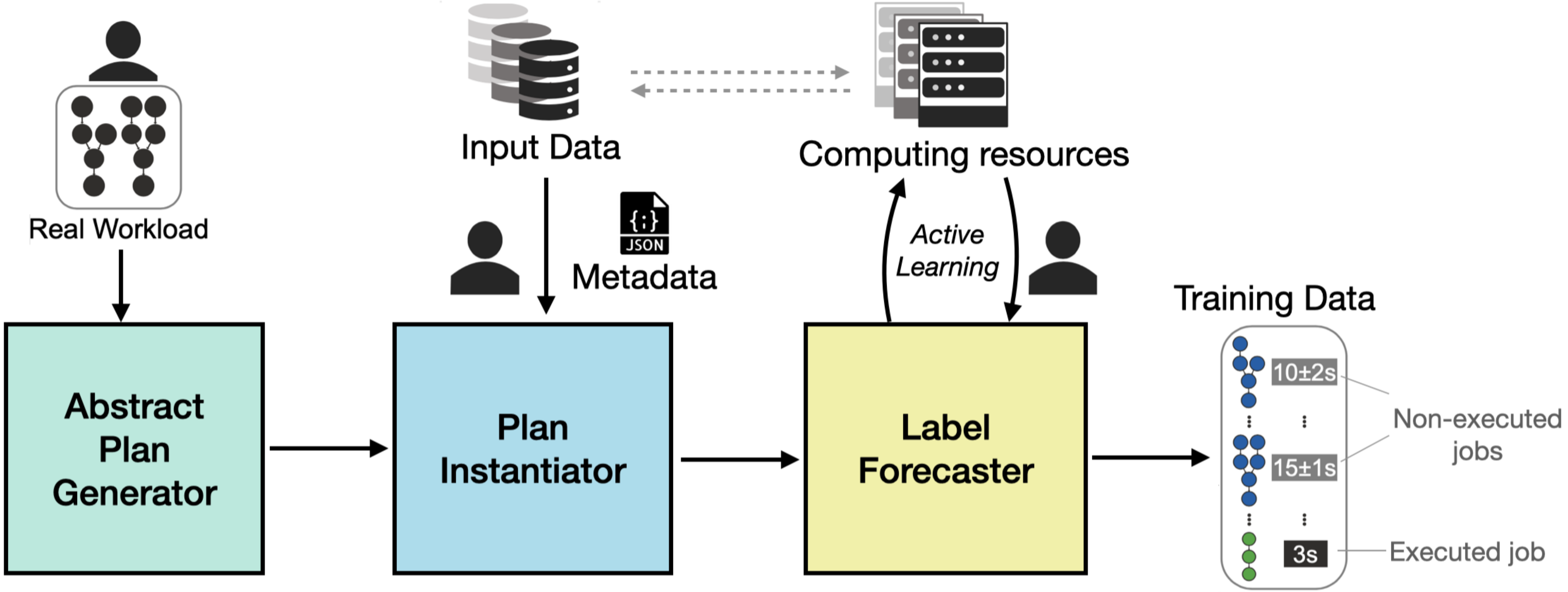




F. Ventura et al. Expand your Training Limits! Generating Training Data for ML-based Data Management. SIGMOD 2021.
R. van de Water. DataFarm: Farm Your ML-based Query Optimizer's Food! – Human-Guided Training Data Generation. CIDR 2022
R. van de Water. Farming Your ML-based Query Optimizer's Food. ICDE 2022 (best demo award)

Synthetic plan generation with DataFarm

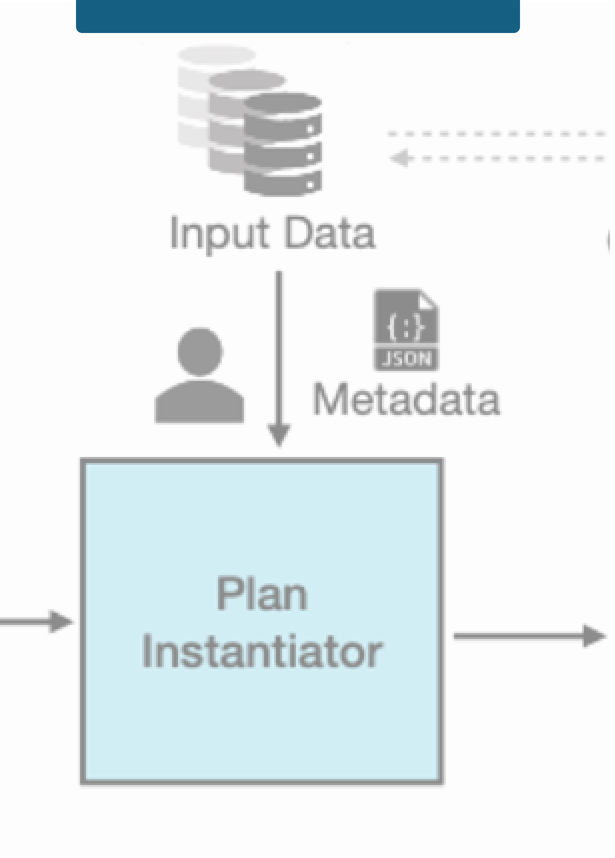
Training data collection

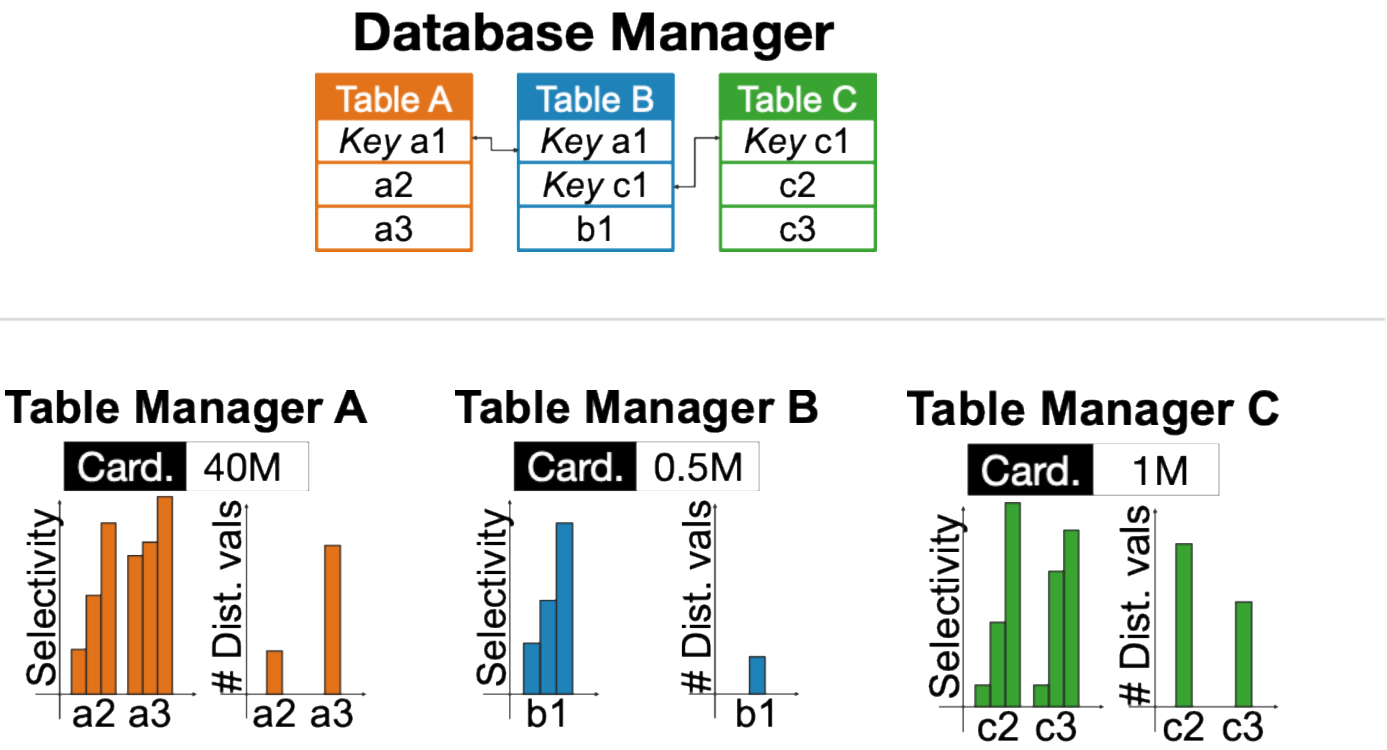


F. Ventura et al. Expand your Training Limits! Generating Training Data for ML-based Data Management. SIGMOD 2021.
R. van de Water. DataFarm: Farm Your ML-based Query Optimizer's Food! – Human-Guided Training Data Generation. CIDR 2022
R. van de Water. Farming Your ML-based Query Optimizer's Food. ICDE 2022 (best demo award)

Synthetic plan generation with DataFarm

Training data collection

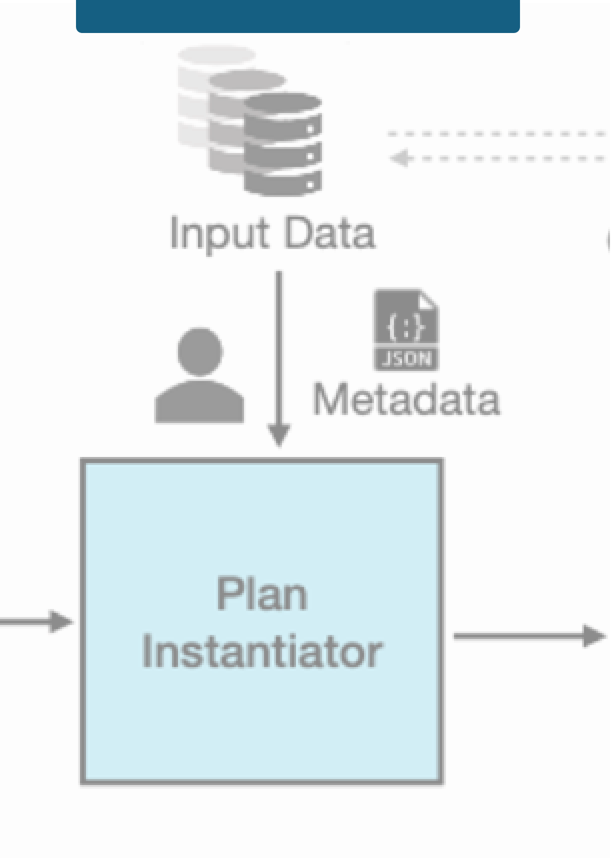


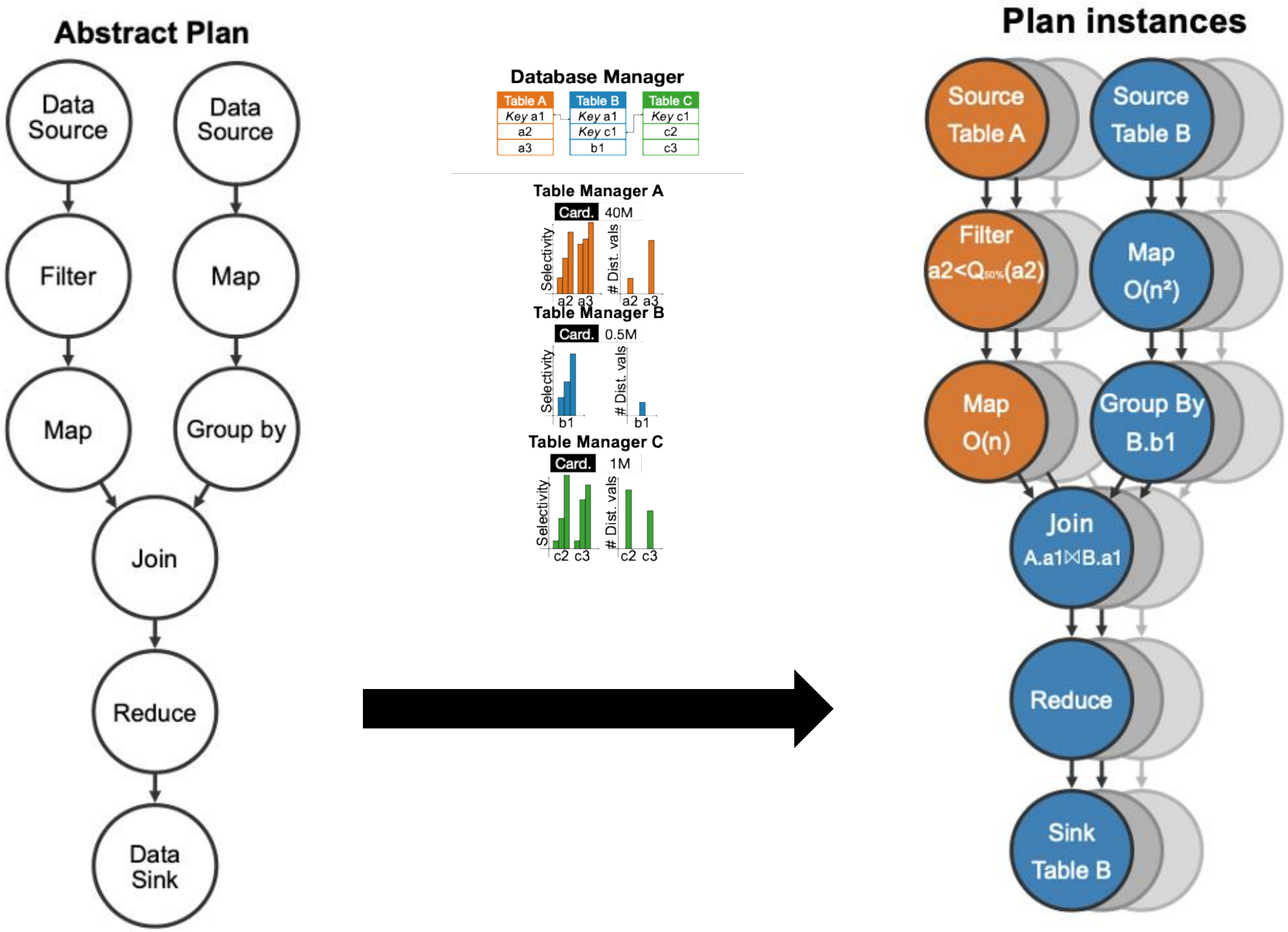


F. Ventura et al. Expand your Training Limits! Generating Training Data for ML-based Data Management. SIGMOD 2021.
R. van de Water. DataFarm: Farm Your ML-based Query Optimizer's Food! – Human-Guided Training Data Generation. CIDR 2022
R. van de Water. Farming Your ML-based Query Optimizer's Food. ICDE 2022 (best demo award)

Synthetic plan generation with DataFarm

Training data collection

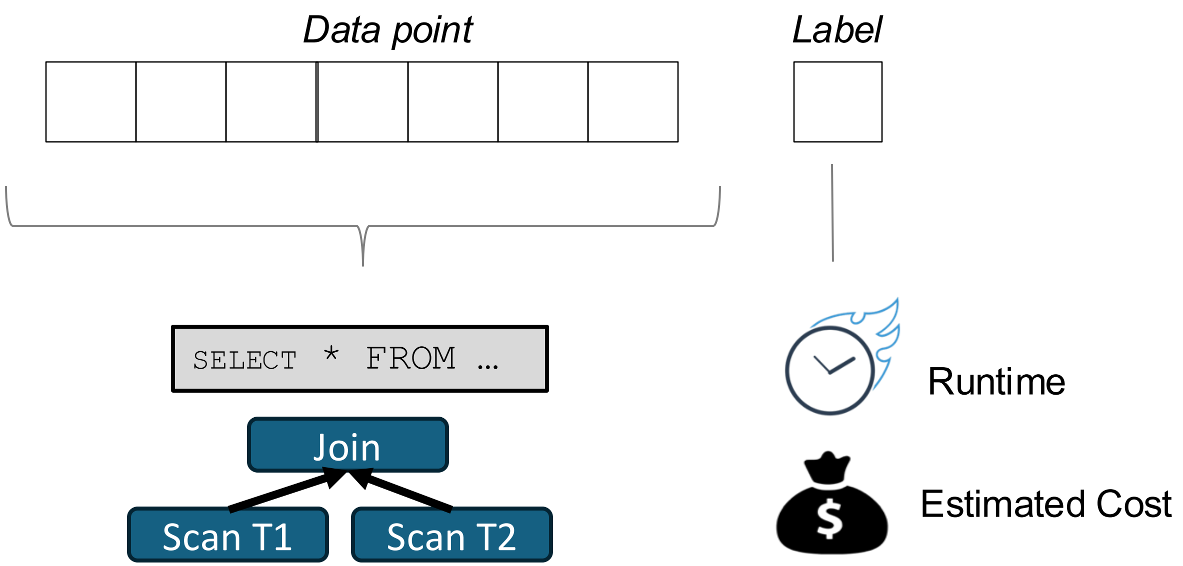




F. Ventura et al. Expand your Training Limits! Generating Training Data for ML-based Data Management. SIGMOD 2021.
R. van de Water. DataFarm: Farm Your ML-based Query Optimizer's Food! – Human-Guided Training Data Generation. CIDR 2022
R. van de Water. Farming Your ML-based Query Optimizer's Food. ICDE 2022 (best demo award)

What is training data in LCMs?

Training data collection



- ➢ Where to find thousands of queries and plans?

- ➢ **How to obtain their label?**

Runtime collection

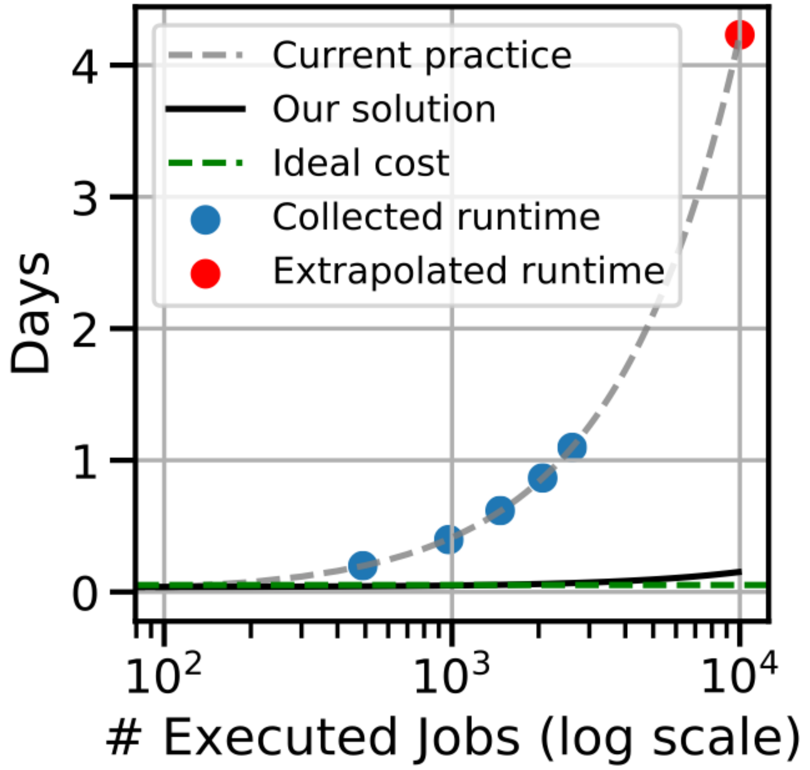
Training data collection

Executing queries can be very time-consuming!

Runtime collection - examples

Training data collection

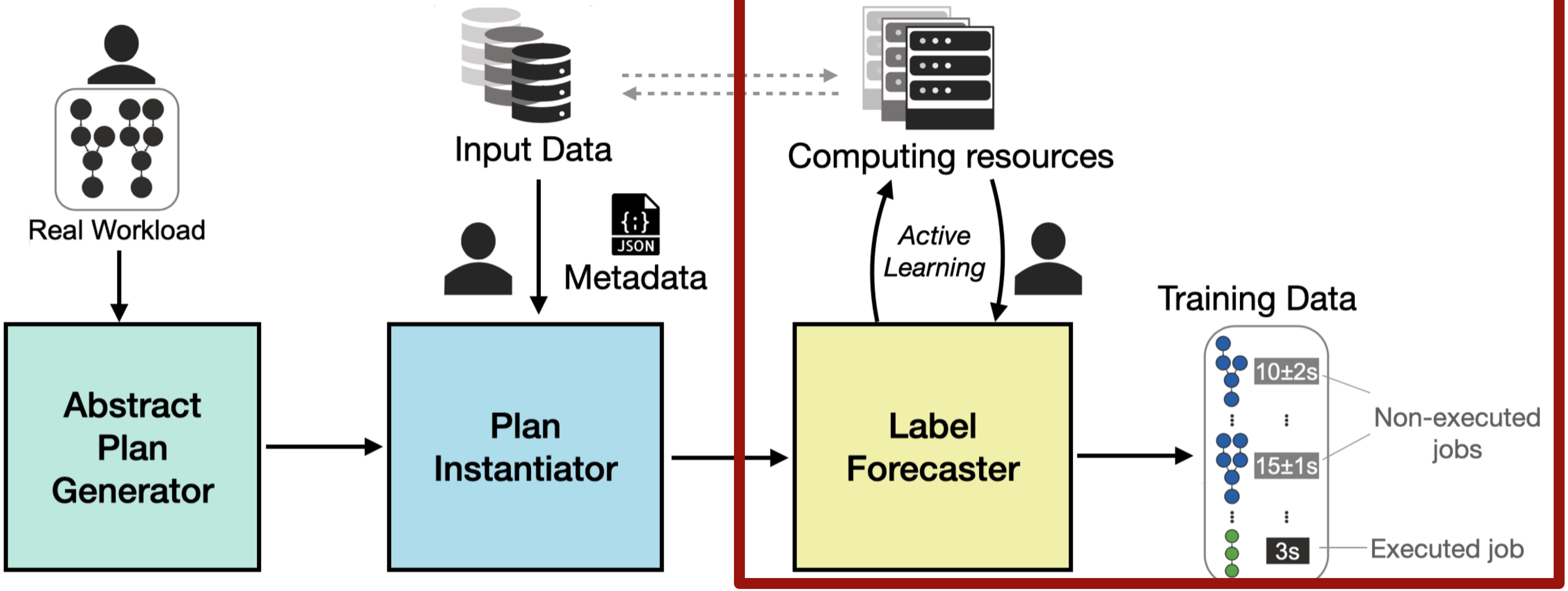
Zero-shot	DataFarm
3.9GB GB data *300k queries* *PostgreSQL*	*1GB data* *10k jobs* *Flink*
↓	↓ …
10 days	***5 days***
B. Hilprecht et al. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction. VLDB 2022.	*F. Ventura et al. Expand your Training Limits! Generating Training Data for ML-based Data Management. SIGMOD 2021.*



Extrapolated cost of 10,000 plans with 1TB input data > 6 months*

Label collection in DataFarm

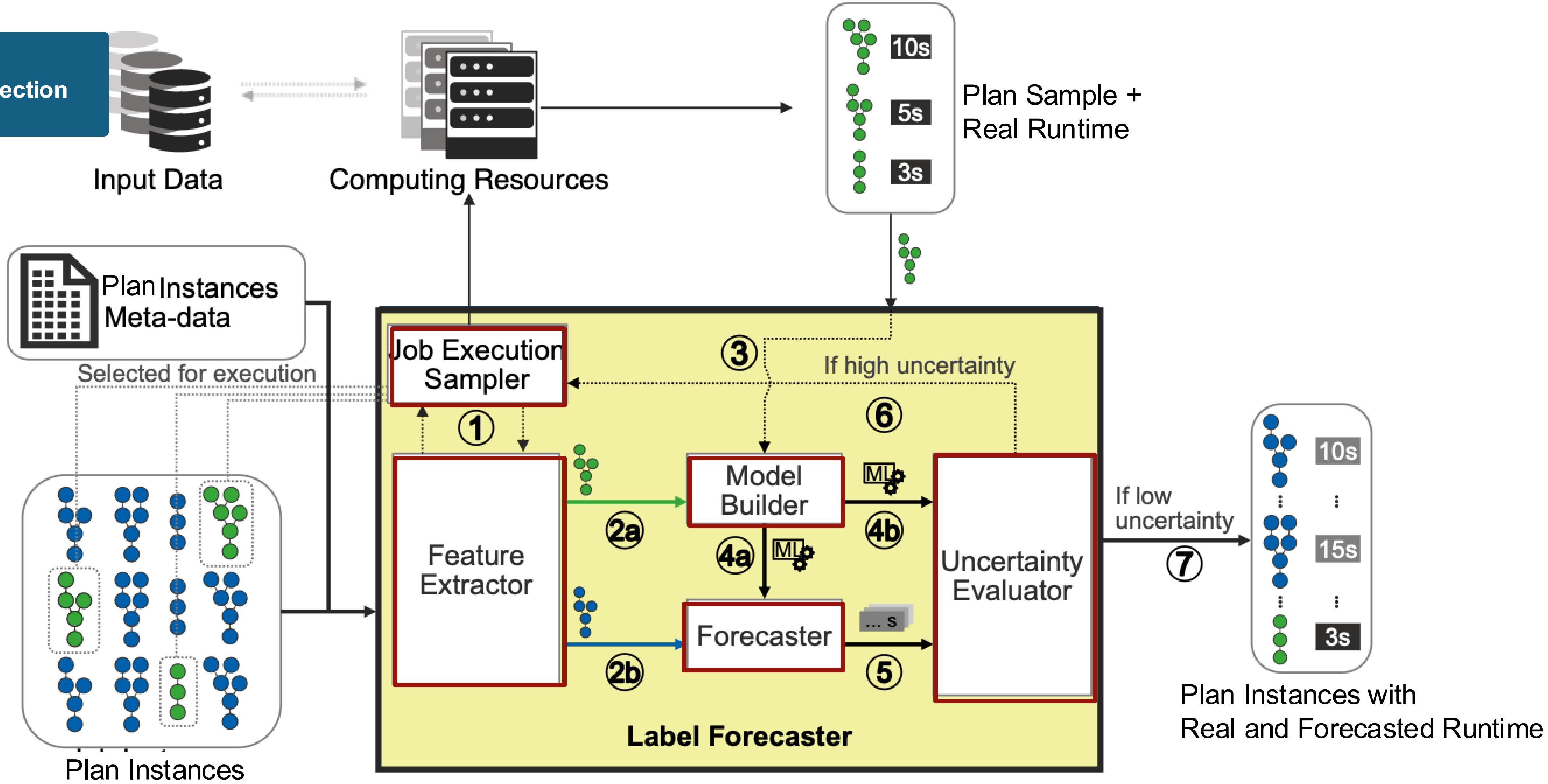
Training data collection



F. Ventura et al. *Expand your Training Limits! Generating Training Data for ML-based Data Management. SIGMOD 2021.*
R. van de Water. *DataFarm: Farm Your ML-based Query Optimizer's Food! – Human-Guided Training Data Generation. CIDR 2022*
R. van de Water. *Farming Your ML-based Query Optimizer's Food. ICDE 2022 (best demo award)*

Label collection in DataFarm

Training data collection



F. Ventura et al. Expand your Training Limits! Generating Training Data for ML-based Data Management. SIGMOD 2021.
R. van de Water. DataFarm: Farm Your ML-based Query Optimizer's Food! – Human-Guided Training Data Generation. CIDR 2022
R. van de Water. Farming Your ML-based Query Optimizer's Food. ICDE 2022 (best demo award)

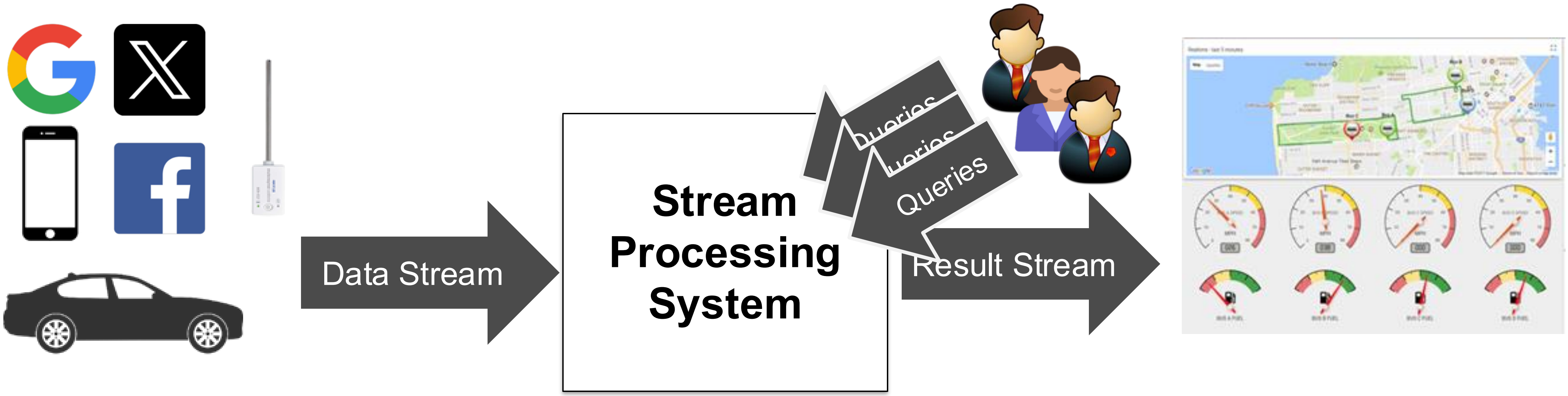
Learned Cost Models Ingredients

- ☑ Featurization
- ☑ ML Model
- ☑ Training data collection

Agenda

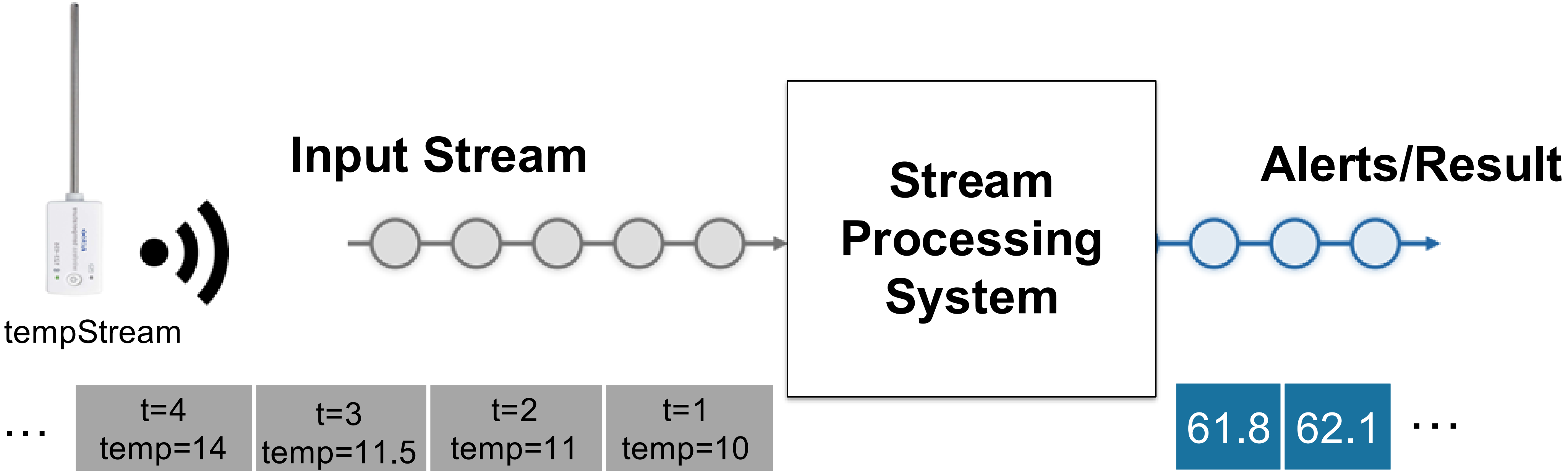
- LCMs in Batch Systems
- **LCMs in Streaming Systems**
- Road Ahead

What is Stream Processing in a Nutshell?



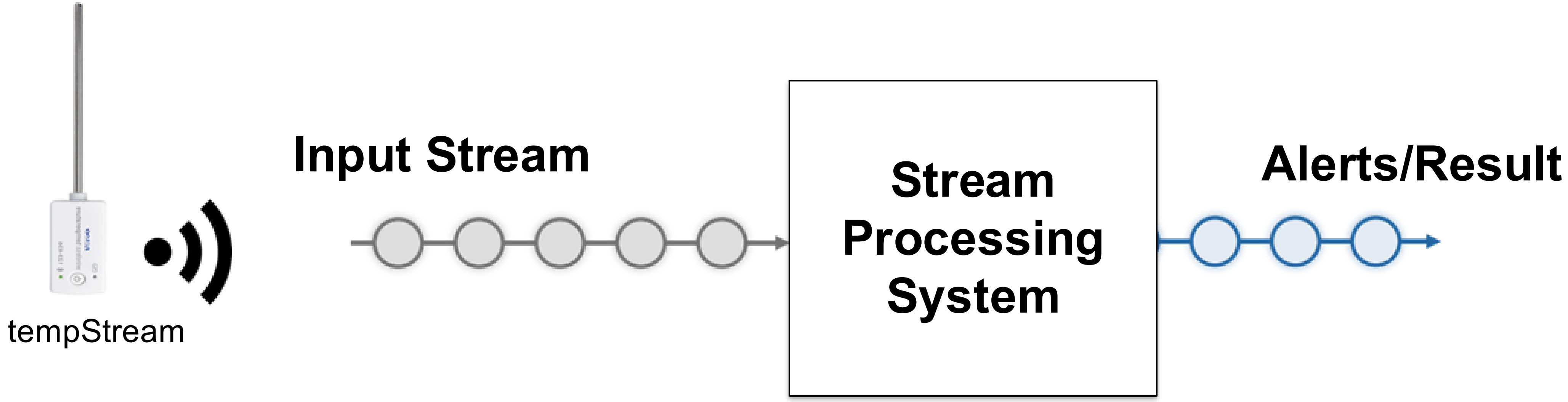
- **Take inputs: Continuous data** from devices (cars/buses, health devices, card transactions, social networks, sensors)
- **AND Standing queries** for monitoring (e.g., positions/speed/# of cars)
- **Output: Continuous results on standing queries** (time-series)
- **Objectives**: (often) **low latency** and **high throughput**

Stream Processing 101



Query: Notify when average values of temperature is higher than 60°C (in the last minute, for the last three sensor values, …)?

Stream Processing 101



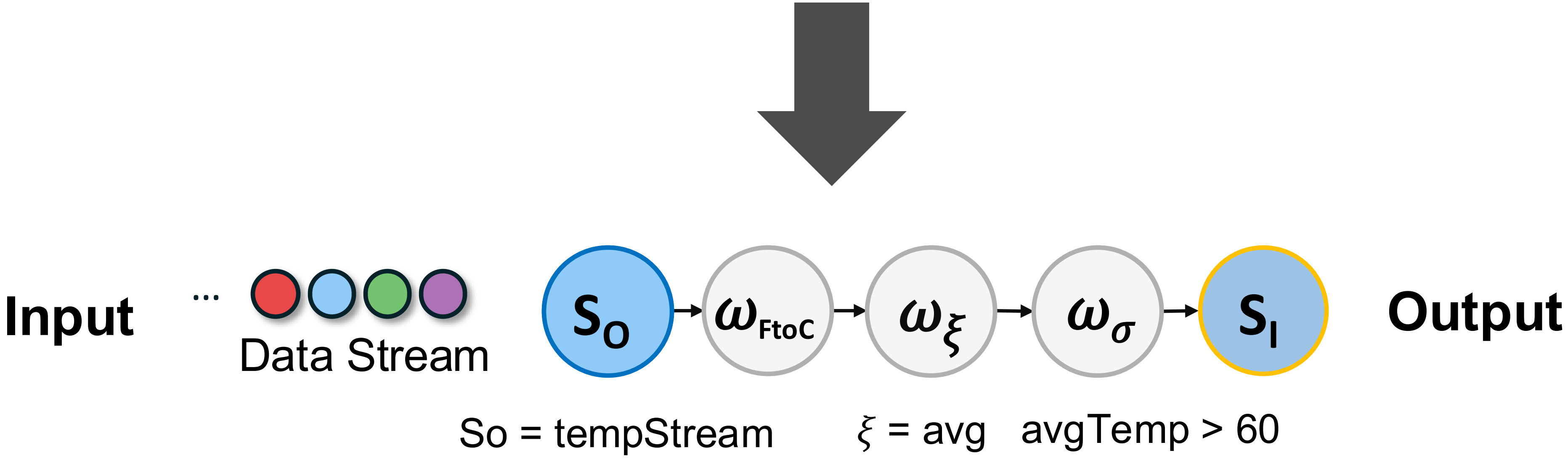
[1]
```
SELECT AVG(FtoC(temp)) as avgTempStream
FROM tempStream [ROWS 3, ADVANCE BY 1 MIN]
HAVING avgTemp > 60
```

[1] Query expressed in CQL (continuous query language), a SQL-like query language for streaming.

Stream Processing 101

Queries are compiled **into data flow graph (DFG) of stream operators**

```
SELECT AVG(FtoC(temp)) as avgTempStream
FROM tempStream [ROWS 3, ADVANCE BY 1 MIN]
HAVING avgTemp > 60
```



Poll

*Can **traditional cost models** of databases be **adapted** to estimate costs of data flow graphs of **streaming** systems?*

Image Source: ChatGPT - DALLE

No Traditional Cost Models in Streaming!

A Catalog of Stream Processing Optimizations

MARTIN HIRZEL, IBM Watson Research Center
ROBERT SOULÉ, University of Lugano
SCOTT SCHNEIDER, IBM Watson Research Center
BUĞRA GEDIK, Bilkent University
ROBERT GRIMM, New York University

Avenues for future work. Finding the right sequence in which to apply optimizations is an interesting problem when there are variants of optimizations with complex interactions. Furthermore, while there is literature with cost models for individual optimizations, extending those to work on multiple optimizations is challenging; in part, that is because the existing cost models are usually sophisticated and cus for their optimization. Furthermore, models for optimizations must captu istics not just of the application, but also of the system and the input dat these characteristics accurately and with moderate cost is another aven work.

Apache Flink™: Stream and Batch Processing in a Single Engine

Paris Carbone[†] Stephan Ewen[‡] Seif Haridi[†]
Asterios Katsifodimos[*] Volker Markl[*] Kostas Tzoumas[‡]

[†]KTH & SICS Sweden [‡]data Artisans [*]TU Berlin & DFKI
parisc,haridi@kth.se first@data-artisans.com first.last@tu-berlin.de

and interesting-property propagation. However, the arbitrary UDF-heavy DAGs that make up Flink's dataflow programs do not allow a traditional optimizer to employ database techniques out of the box [17], since the operators hide their semantics from the optimizer. For the same reason, cardinality and cost-estimation methods are equally difficult to employ. Flink's runtime supports various execution strategies including repartition and

[2]

[1] Hirzel, M., Soulé, R., Schneider, S., Gedik, B., & Grimm, R. (2014). *A Catalog of Stream Processing Optimizations.* ACM Computing Surveys (CSUR), 46(4).
[2] Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). *Apache Flink™: Stream and Batch Processing in a Single Engine.* IEEE Data Engineering Bulletin.

Using Current Streaming Systems Feels Like

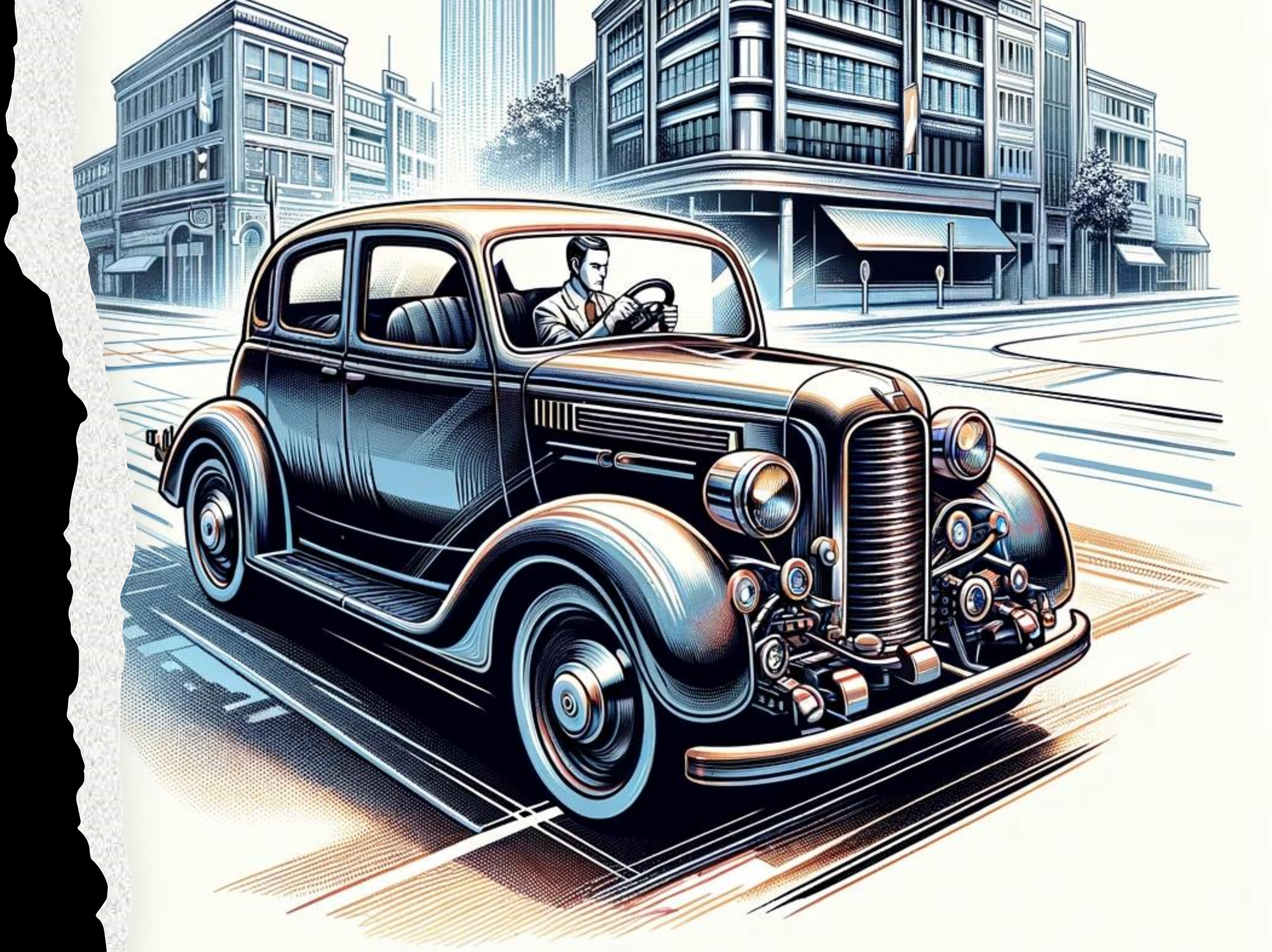
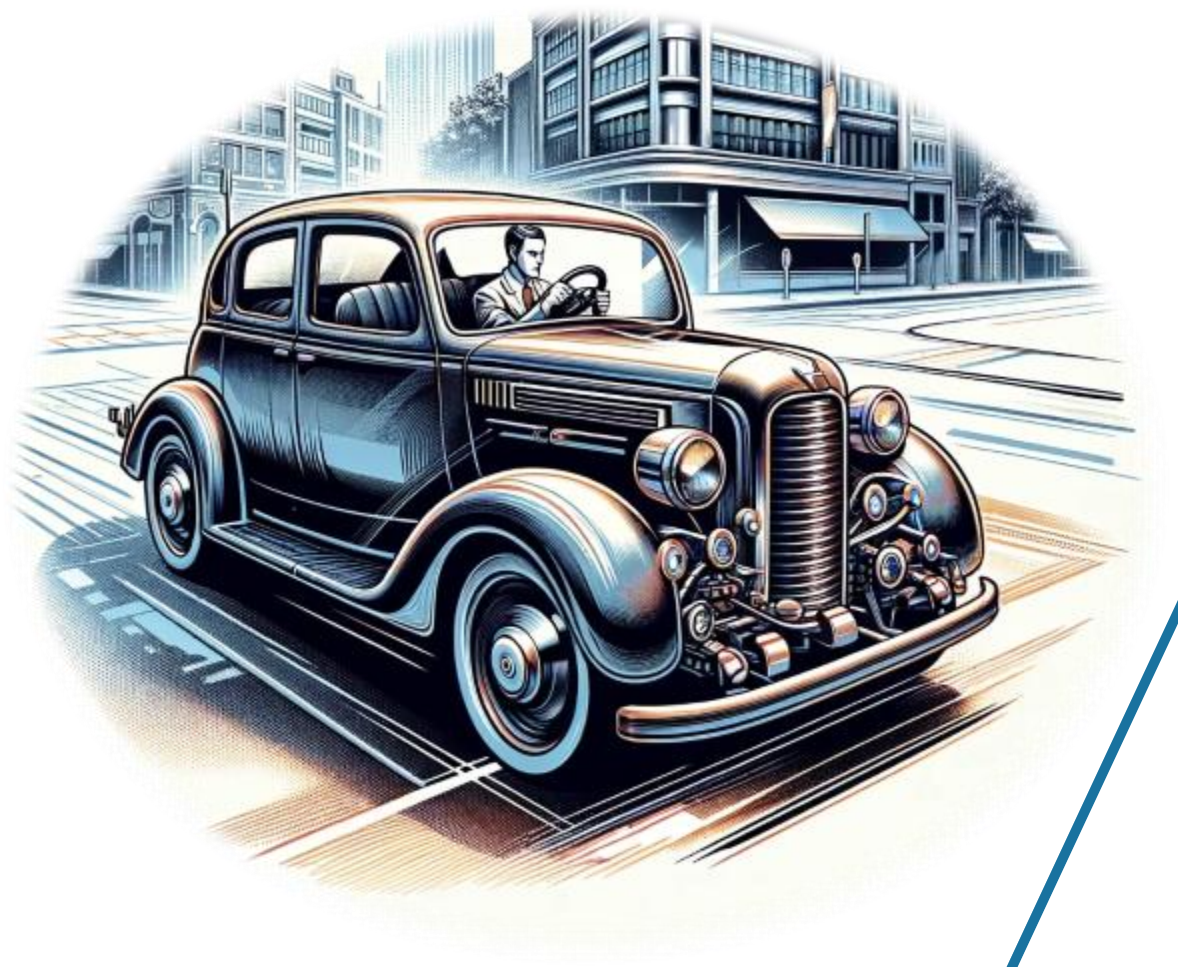
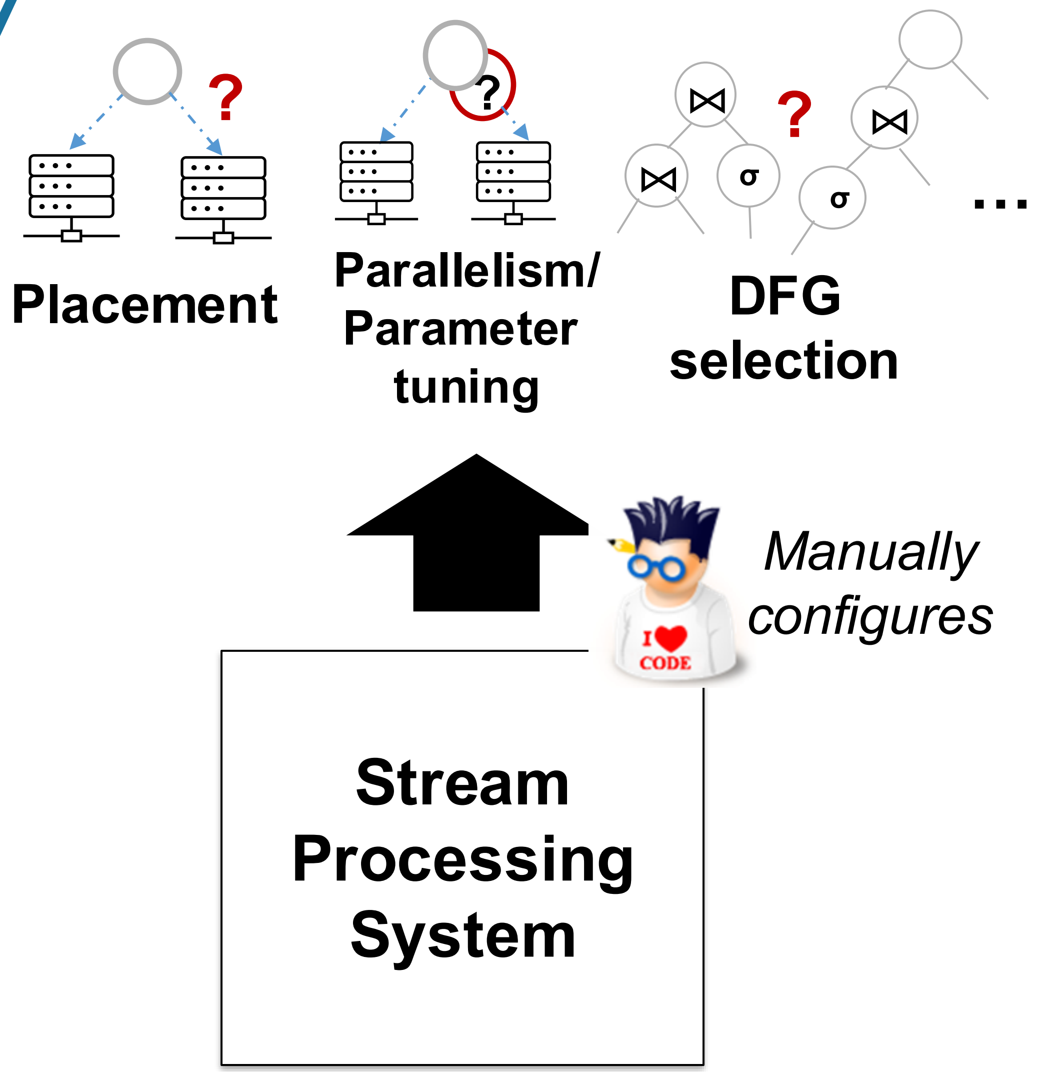


Image Source: ChatGPT - DALLE

Optimization Parameters in Stream Processing

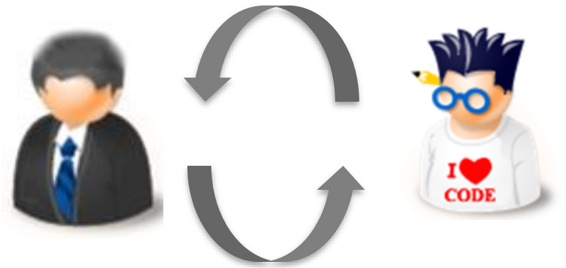




Expert Tuning in Streaming Systems

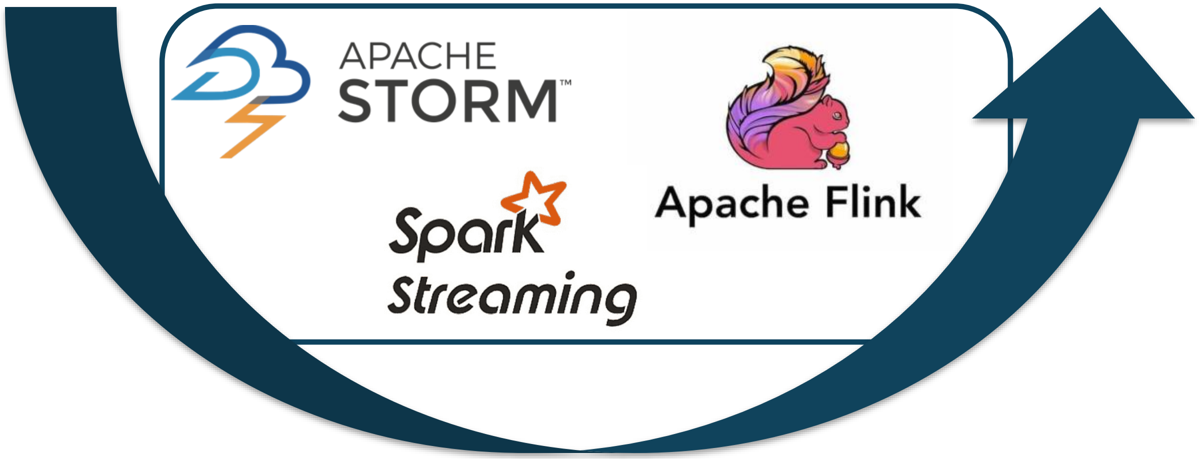
Extensive Tuning Needed!
→ Expert tuning to meet performance constraints

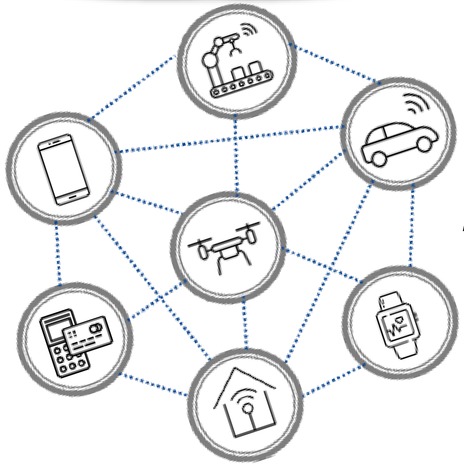
Notify when average number of car
on the street is greater than 60



Domain Expert

Data Engineer



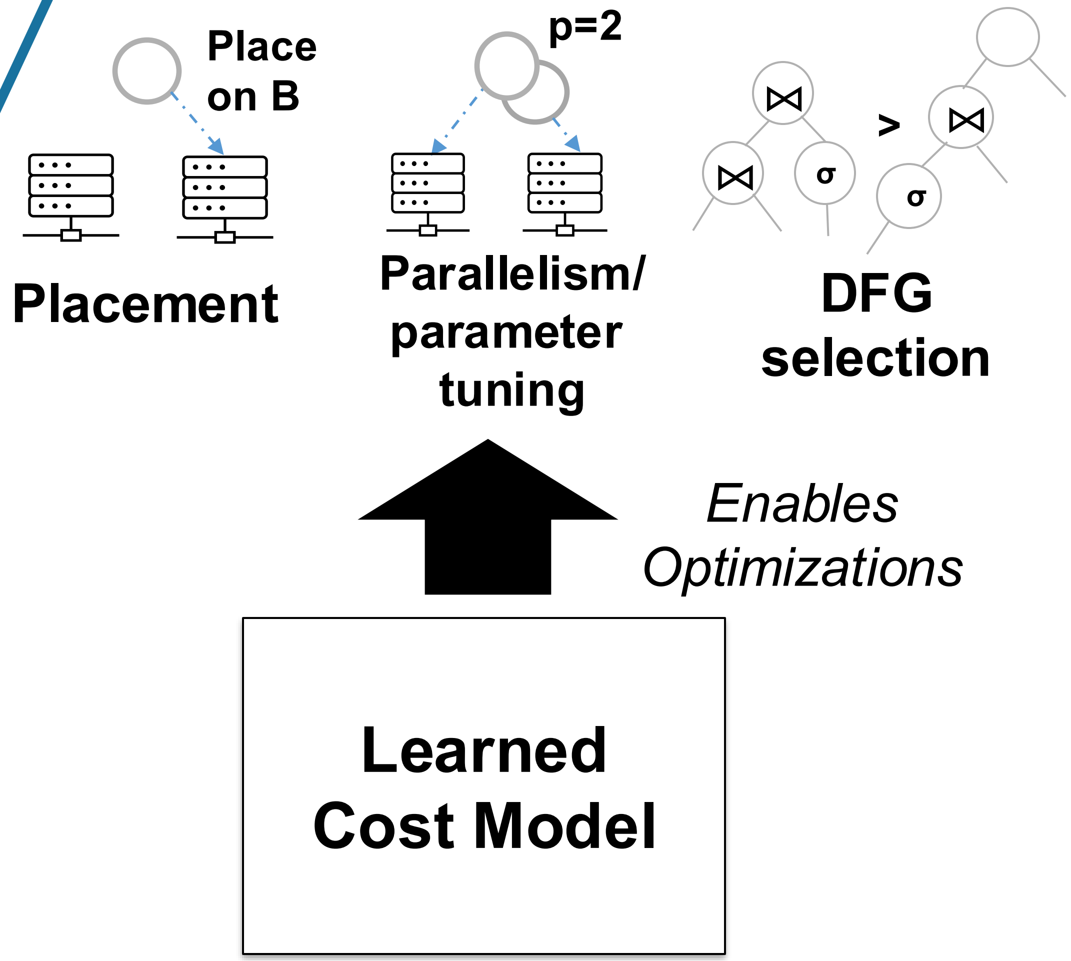


IoT infrastructure

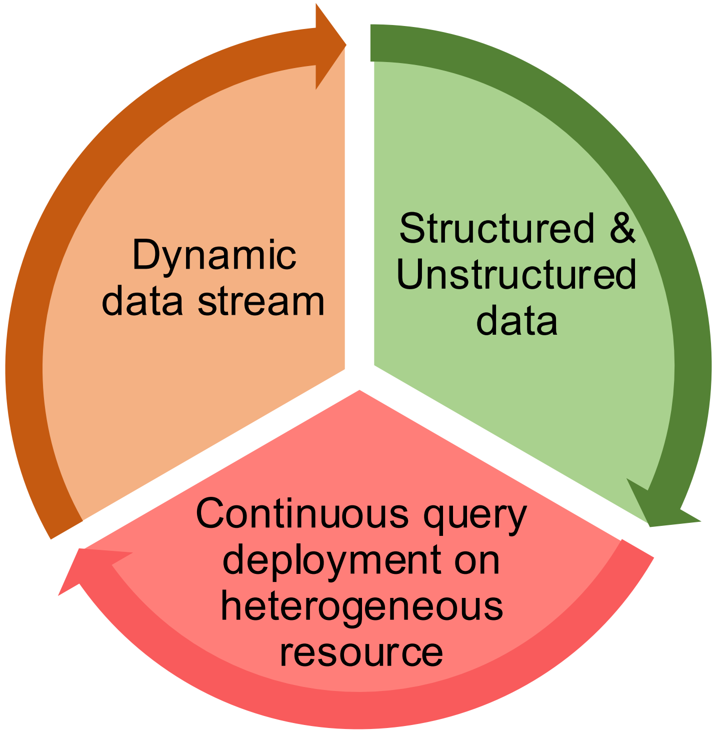
Learned Cost Models To the Rescue

Image Source: ChatGPT - DALLE

Opportunity: LCMs Enabled Optimizations

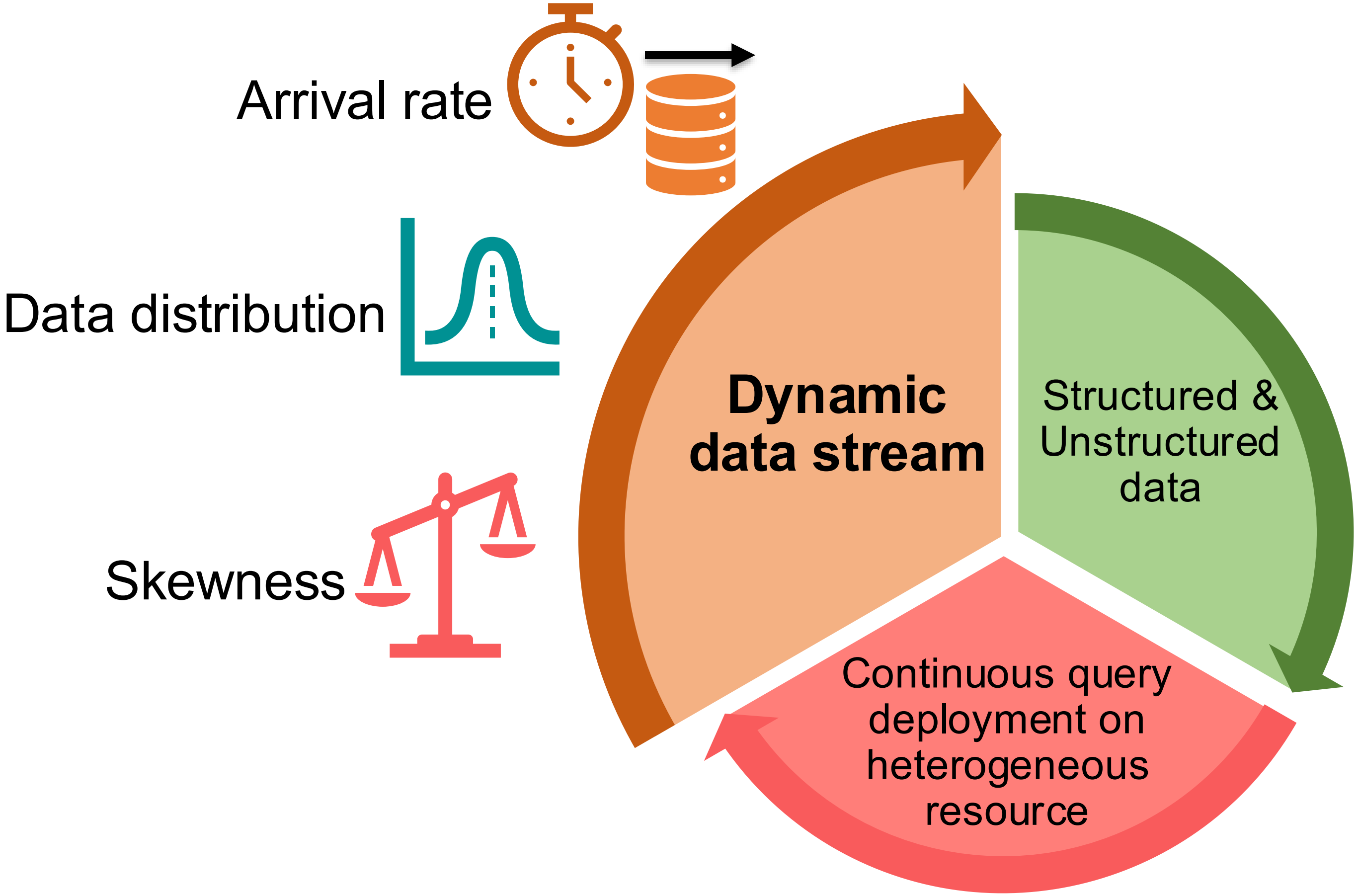


Challenges for LCMs for Streaming



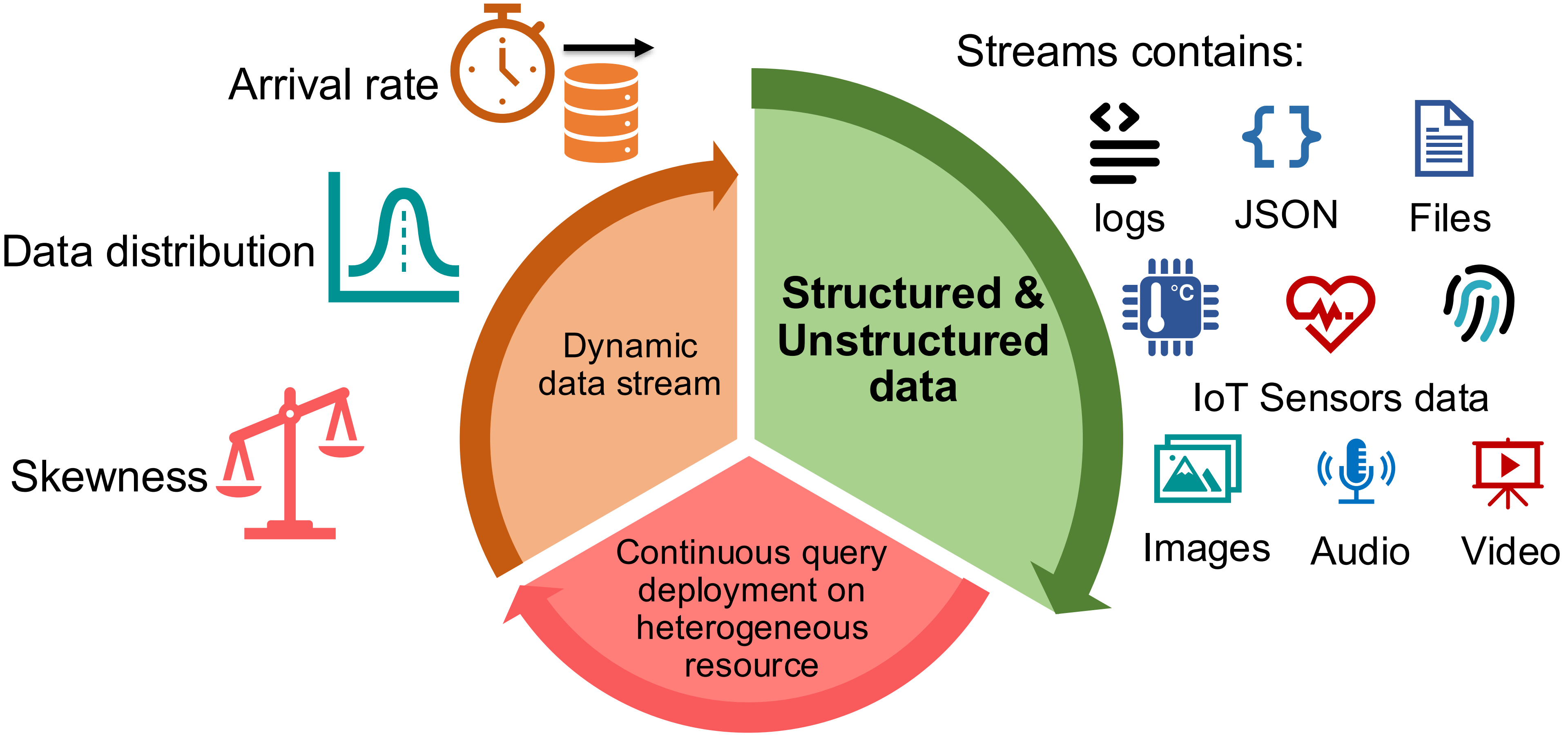
G Rosinosky, D Schmitz, and E Rivière. 2024. StreamBed: Capacity Planning for Stream Processing. DEBS '24

Challenges for LCMs for Streaming



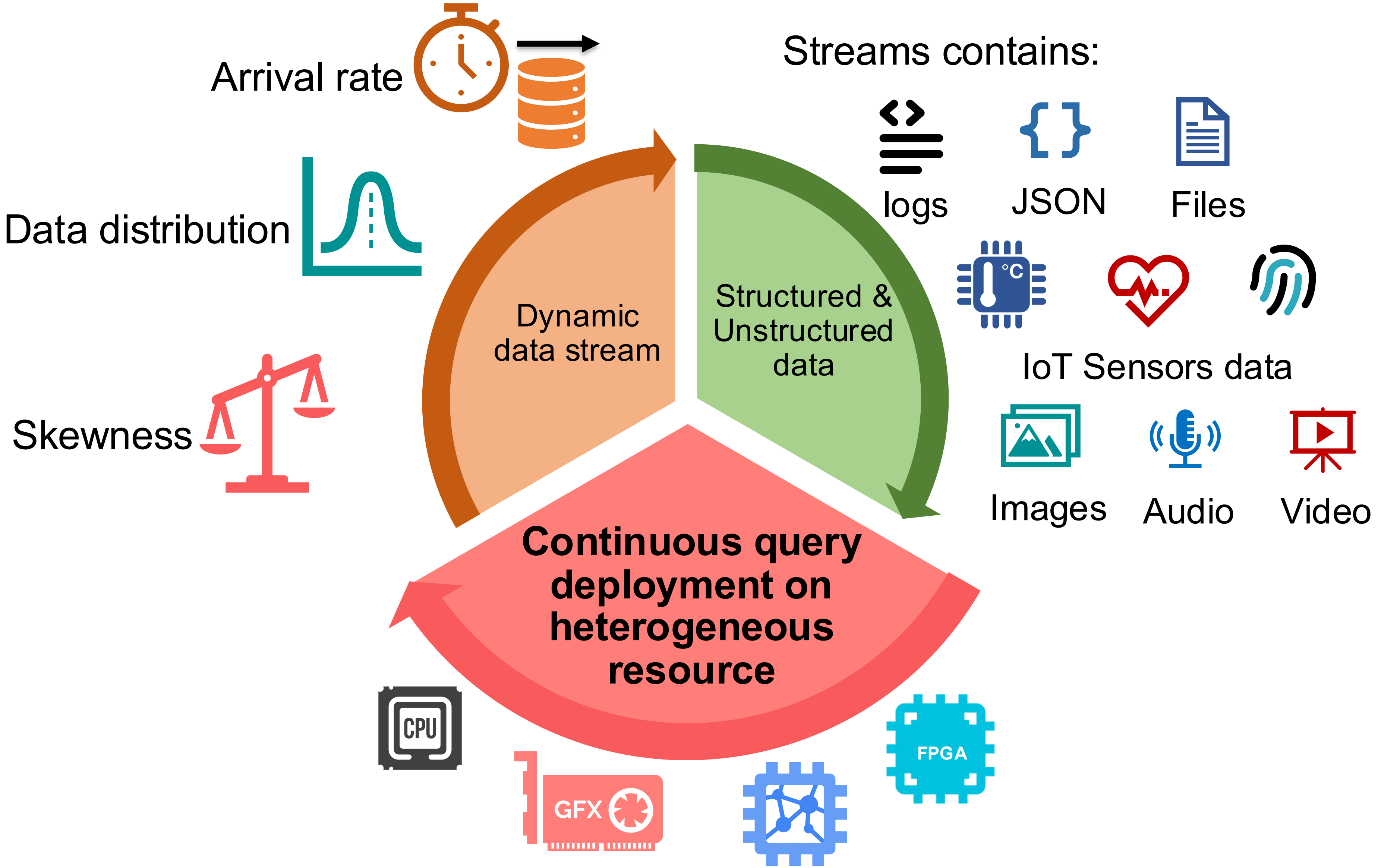
G Rosinosky, D Schmitz, and E Rivière. 2024. StreamBed: Capacity Planning for Stream Processing. DEBS '24

Challenges for LCMs for Streaming



G Rosinosky, D Schmitz, and E Rivière. 2024. StreamBed: Capacity Planning for Stream Processing. DEBS '24

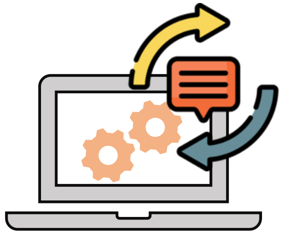
Challenges for LCMs for Streaming



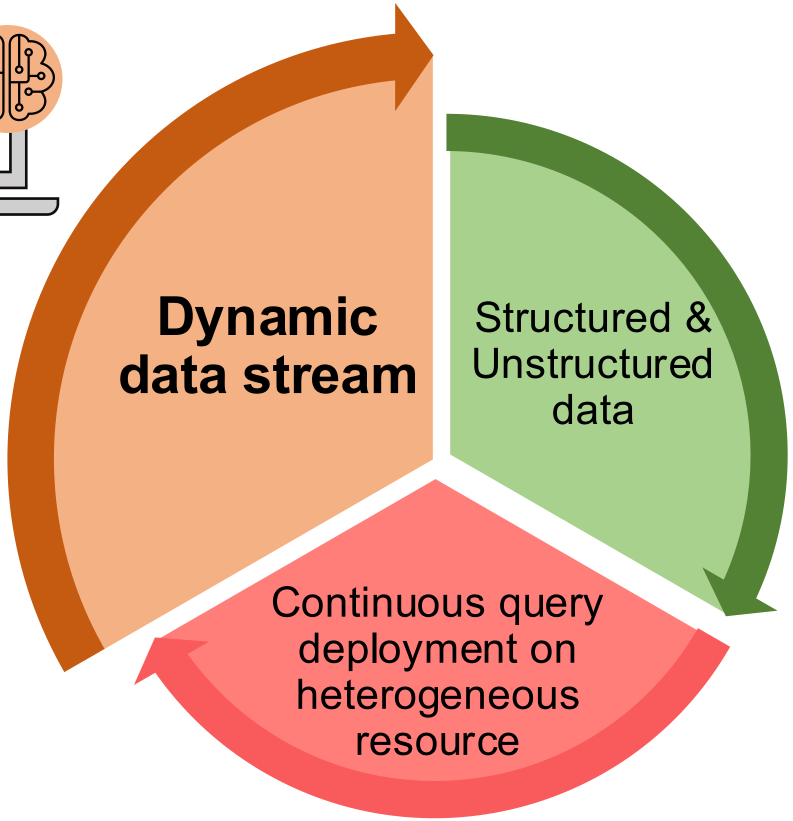
How existing LCMs deals with them?

LCMs are **(re)trained** or **fine-tuned** on dynamic data

Learn from **feedback loops** (monitoring)



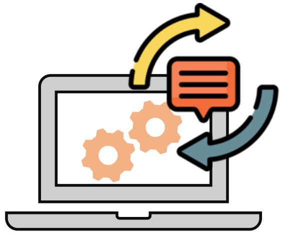
E.g., RL approaches like Decima

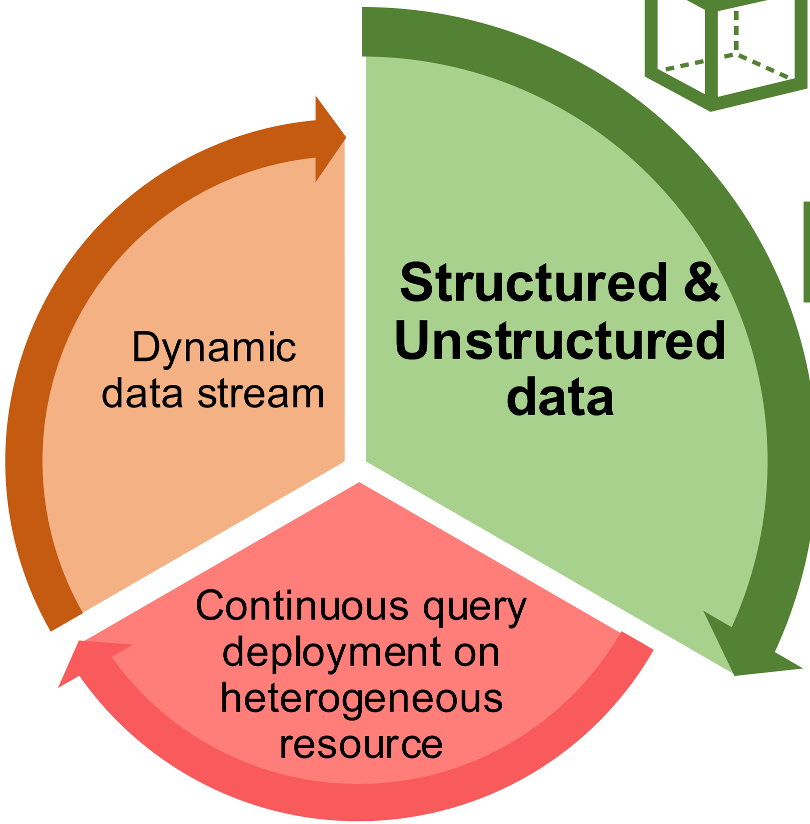


How existing LCMs deals with them?

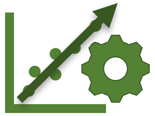
LCMs are **trained** or **fine-tuned** on dynamic data

Learn from **feedback loops** (monitoring)





Embed data into **feature vectors**

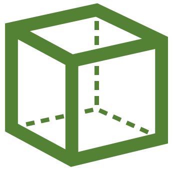


LCMs **map features** to **operator runtime costs**

But, LCMs **do not yet fully understand** unstructured data!
(not our focus here)

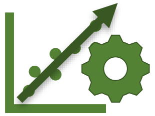
E.g., Regression models like Moira

How existing LCMs deals with them?

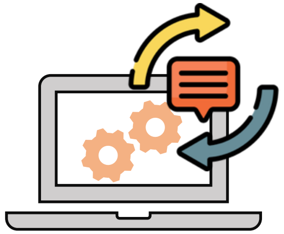


Embed data into **feature vectors**

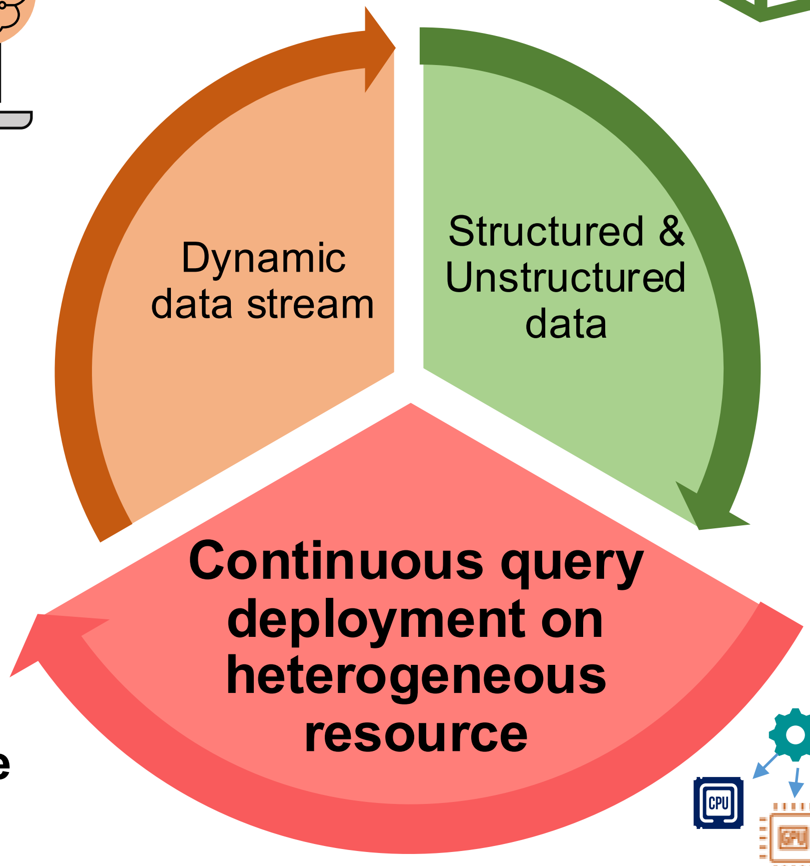
LCMs are **trained** or **fine-tuned** on dynamic data

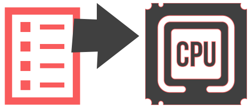


LCMs **map features** to **operator runtime costs**

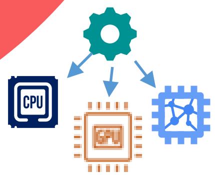


Learn from **feedback loops** (monitoring)





LCMs **include hardware descriptors**



LCMs **can guide placement decisions**

E.g., Optimization oriented LCMs like COSTREAM

Taxonomy of LCMs in Streaming

	Model	Intended Task	Model Architecture	Input Features			
				Stream Statistics	Hardware Charact.	Hardware Monitoring	Query Plan
General LCMs	Moira	Cost Estimation (latency, throughput)	SVM	✓		✓	
	Imai et al.	Cost Estimation (throughput)	Linear Reg.		✓	✓	
	Li et al. 2014	Cost Estimation (latency)	SVR		✓	✓	
	Zero-shot	Cost Estimation (latency, throughput)	GNNs	✓			✓
Optimization-oriented LCMs	ZeroTune	Operator Parallelism	GNNs	✓	✓		✓
	COSTREAM	Operator Placement	GNNs	✓	✓		✓
	Li et al. 2016	Operator Placement	RL	✓		✓	
	Decima	Operator Placement	RL	✓		✓	
	Ni et al.	Operator Placement	RL	✓		✓	

Generalization vs. Specialization

Generalizable LCM (e.g., Zero-shot) VS **Specialized LCM (e.g. RL tuned for workload)**

🎯 Transfer across workloads and hardware
- Use **transferable features**
- Shows high **accuracy** on **unseen workloads**

✅ **Advantages**
can better deal with workload drifts, adaptable

❌ **Disadvantages**
high (one-time) training effort

🎯 Optimize for a given workload/task
- Use **runtime-driven features**
- Shows high **accuracy on known workloads**

✅ **Advantages**
better accuracy (overfit to workload), adapts online

❌ **Disadvantages**
retraining required to deal with workload drift

Era of **Generalizable Cost Models** for Streaming

Zero-Shot Cost Models for Distributed Stream Processing

Roman Heinrich
DHBW Mannheim

Manisha Luthra
Technical University of Darmstadt

Harald Kornmayer
DHBW Mannheim

Carsten Binnig
Technical University of Darmstadt & DFKI

ABSTRACT

This paper proposes a learned cost estimation model for Distributed Stream Processing Systems (DSPS) with an aim to provide accurate cost predictions of executing queries. A major premise of this work is that the proposed learned model can generalize to the dynamics of streaming workloads *out-of-the-box*. This means a model once trained can accurately predict performance metrics such as *latency* and *throughput* even if the characteristics of the data and workload or the deployment of operators to hardware changes at runtime. That way, the model can be used to solve tasks such as optimizing the placement of operators to minimize the end-to-end latency of a streaming query or maximize its throughput even under varying conditions. Our evaluation on a well-known DSPS, Apache Storm, shows that the model can predict accurately for unseen workloads and queries while generalizing across real-world benchmarks.

CCS CONCEPTS

• **Information systems → Stream management**.

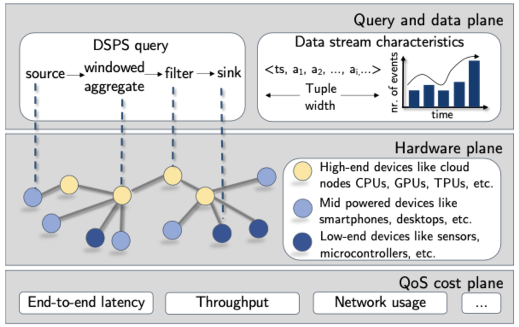
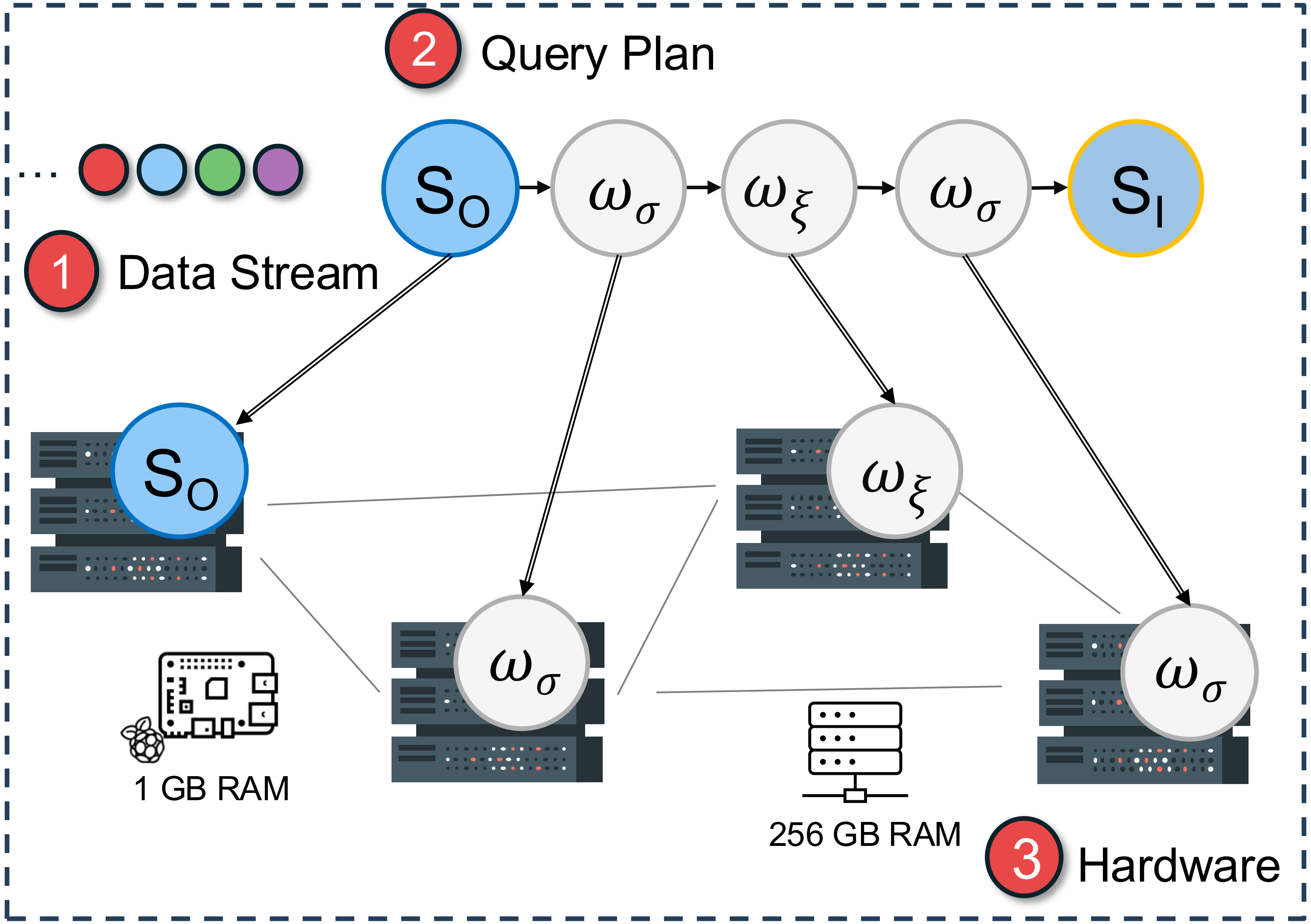


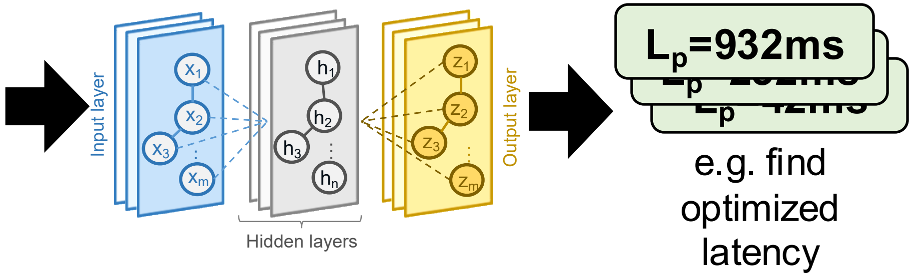
Figure 1: A DSPS has to provide guarantees in terms of one or more quality-of-service (QoS) cost metrics such as latency and throughput. The challenge is that DSPS serve a wide range of workloads on potentially diverse hardware, which makes the cost estimation harder.

Typically, a DSPS provides QoS guarantees using optimization mechanisms such as *operator placement* that usually monitors the costs to decide on the mapping of operators to hardware as shown in Figure 1 [2]. Moreover, frequent reconfigurations of the operator placement are required based on the observed changes of the work-

Zero-shot Cost Model in a Nutshell

Generalizable LCM
(e.g., Zero-shot)





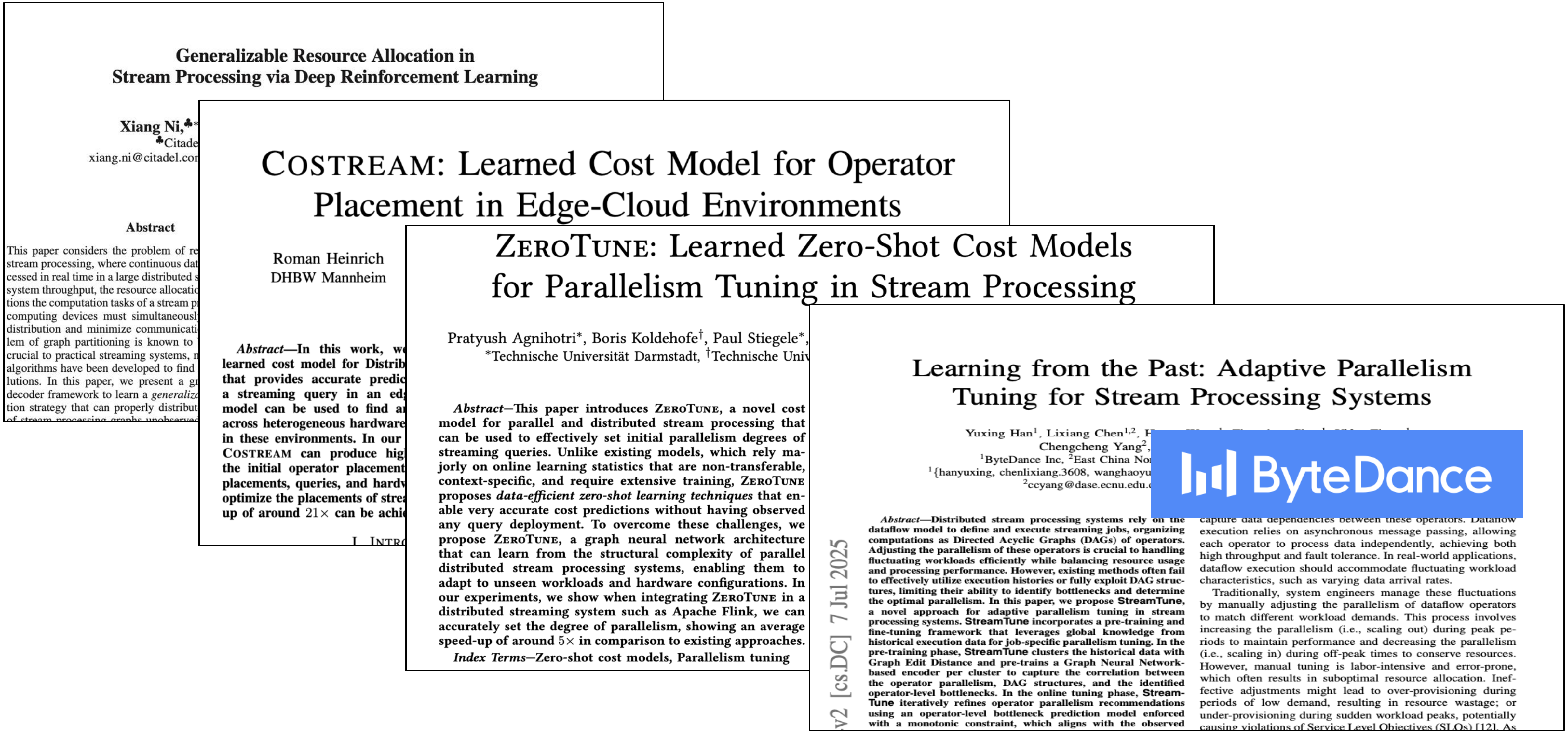
GNN model

Cost Predictions

Latency, Throughput,
Backpressure, …

R. Heinrich, C. Binnig, H. Kornmayer & M. Luthra, *Costream: Learned Cost Model for Operator Placement in Edge-Cloud Environments*, ICDE 2024.
P. Agnihotri, B. Koldehofe, P. Stiegele, R. Heinrich, C. Binnig & M. Luthra, *ZeroTune: Learned Zero-Shot Cost Models for Parallelism Tuning*, ICDE 2024.

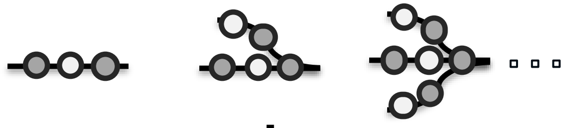
Generalizable Models for Optimizations



Zero-Shot Cost Model: **Training**

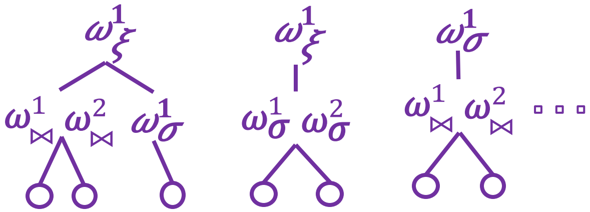
Training Zero-Shot Cost Model

Data Stream



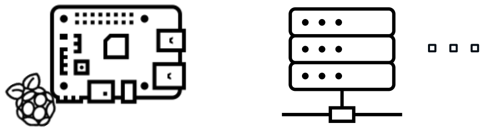
+

Query Workload



+

Hardware Resources



Transferable Features

- Event rate
- ⋮
- Filter function
- Window length
- Parallelism degree
- ⋮
- CPU cores
- ⋮

Filter function “>”

Filter literal data type

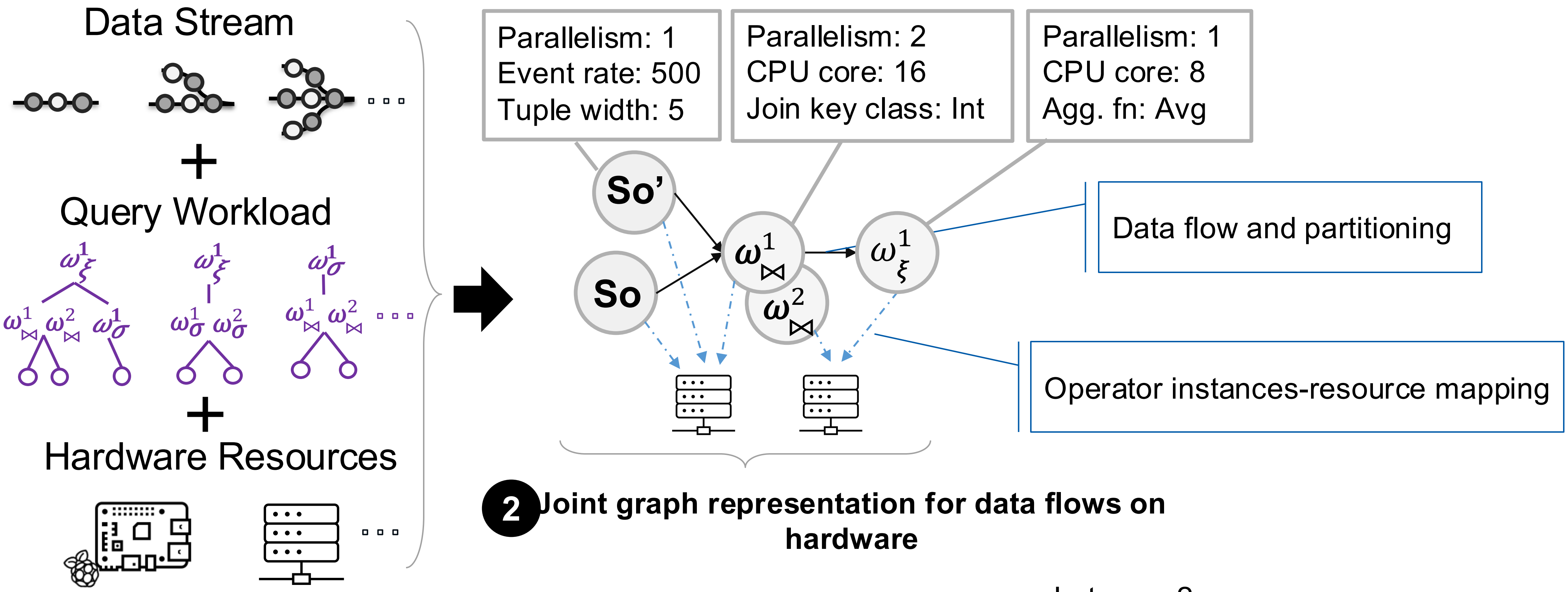
Age > 10

Filter literal value

1 Broad training dataset

Zero-Shot Cost Model: **Training**

Training Zero-Shot Cost Model



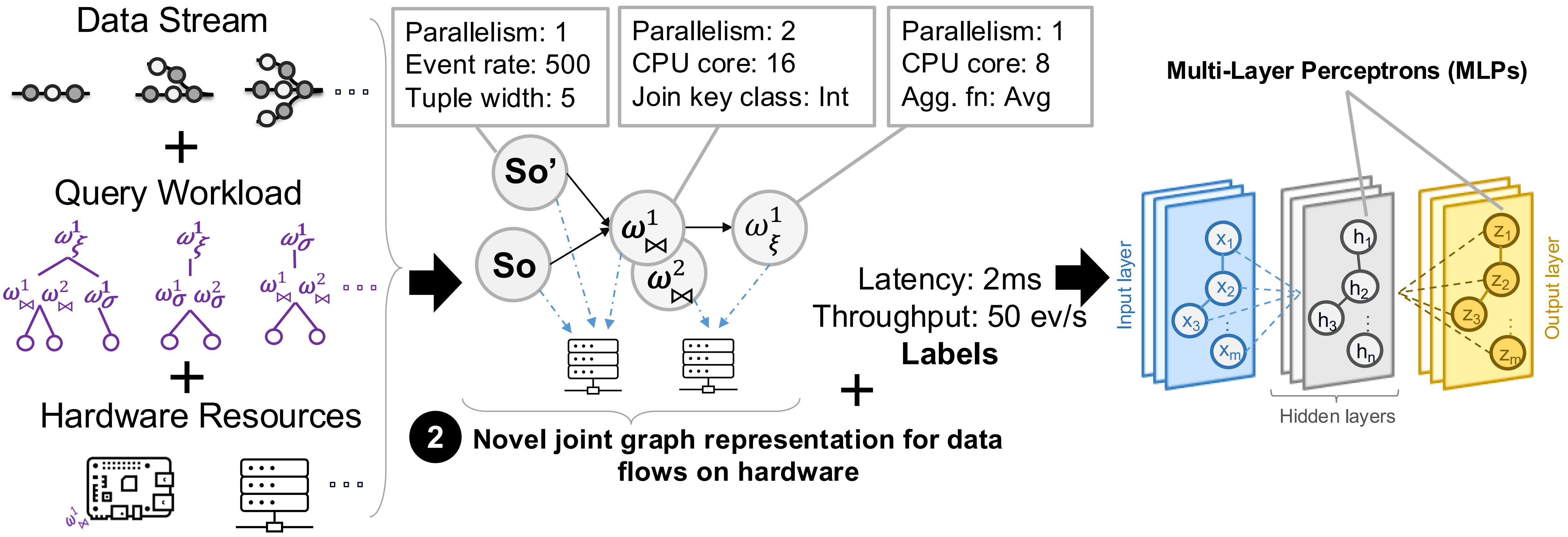
2 **Joint graph representation for data flows on hardware**

1 **Broad training dataset**

3 **Transferable features and query labels** + Latency: 2ms
Throughput: 50 ev/s
Labels

Zero-Shot Cost Model: **Training**

Training Zero-Shot Cost Model



1 **Broad training dataset**

2 **Novel joint graph representation for data flows on hardware**

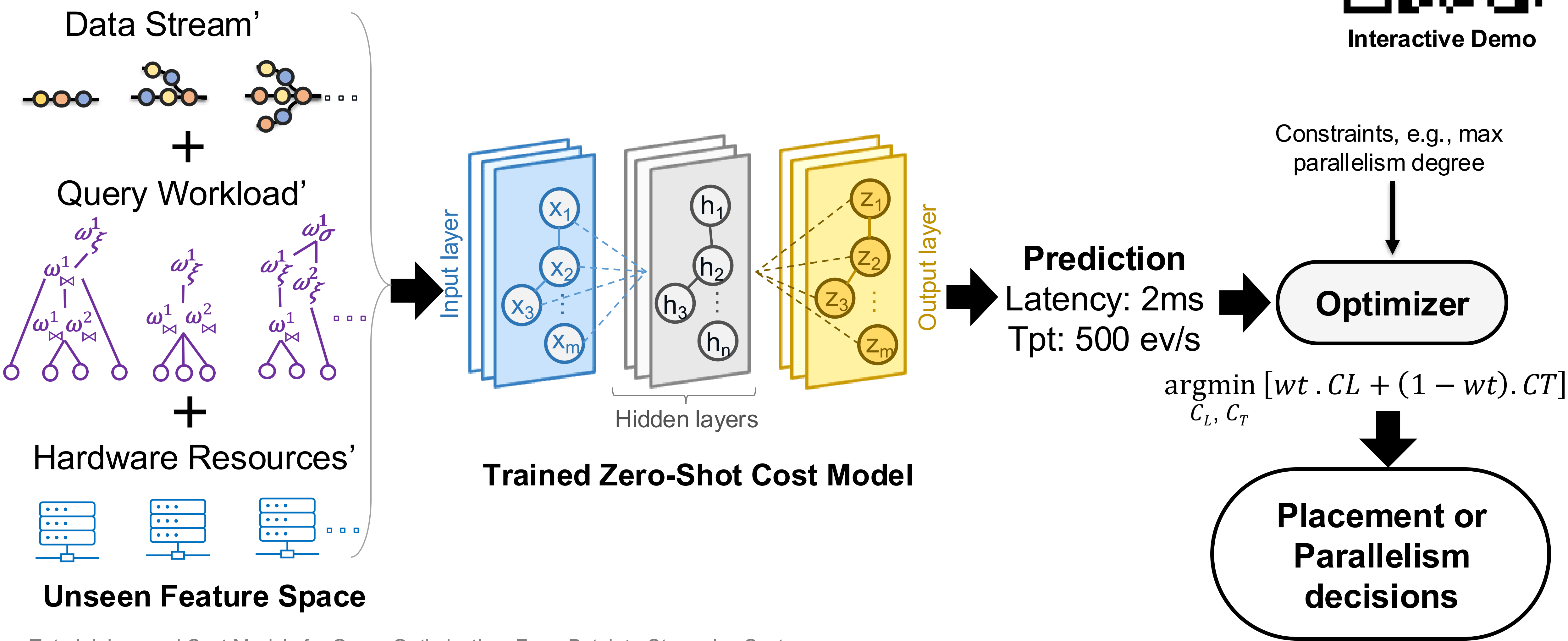
3 **Transferable features and query labels**

Train Zero-Shot Cost Model

Zero-Shot Cost Model: **Inference**

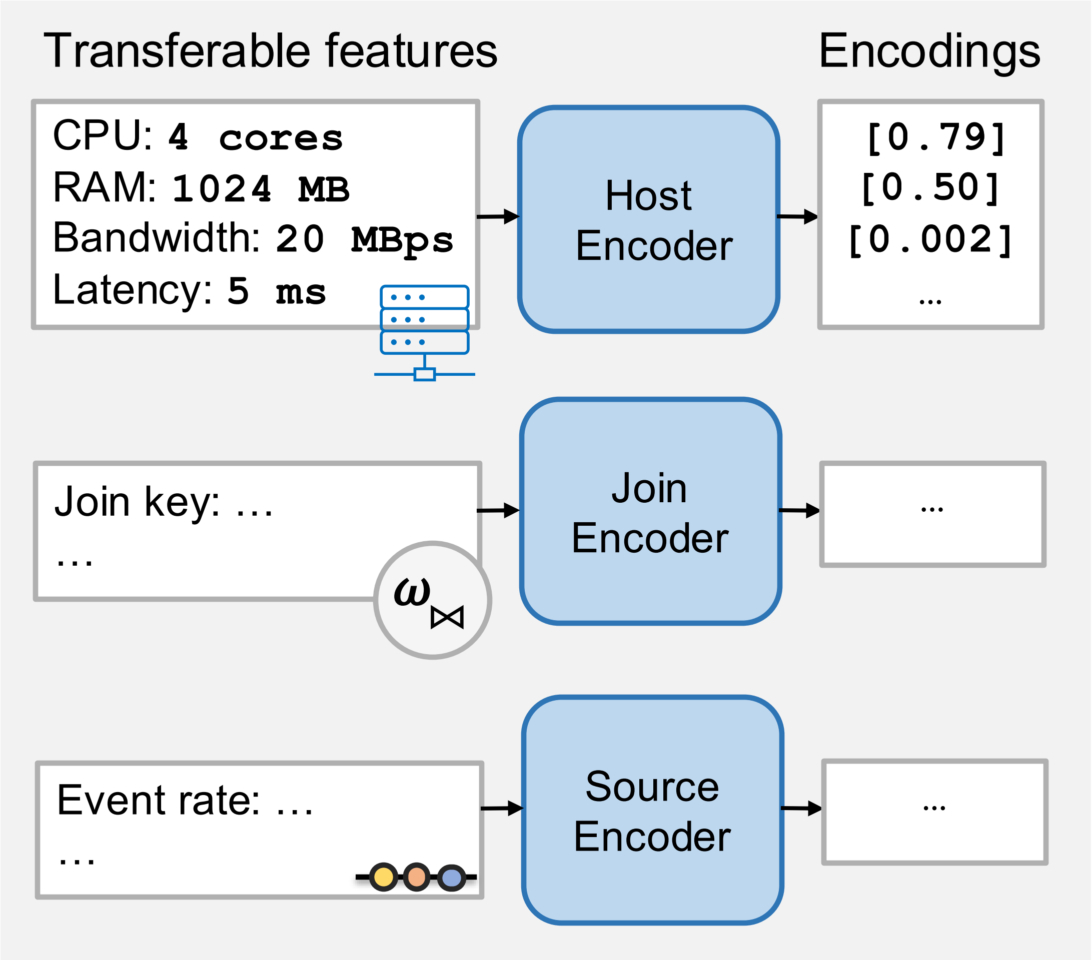
Interactive Demo

Inference and Optimization using Zero-Shot Cost Model

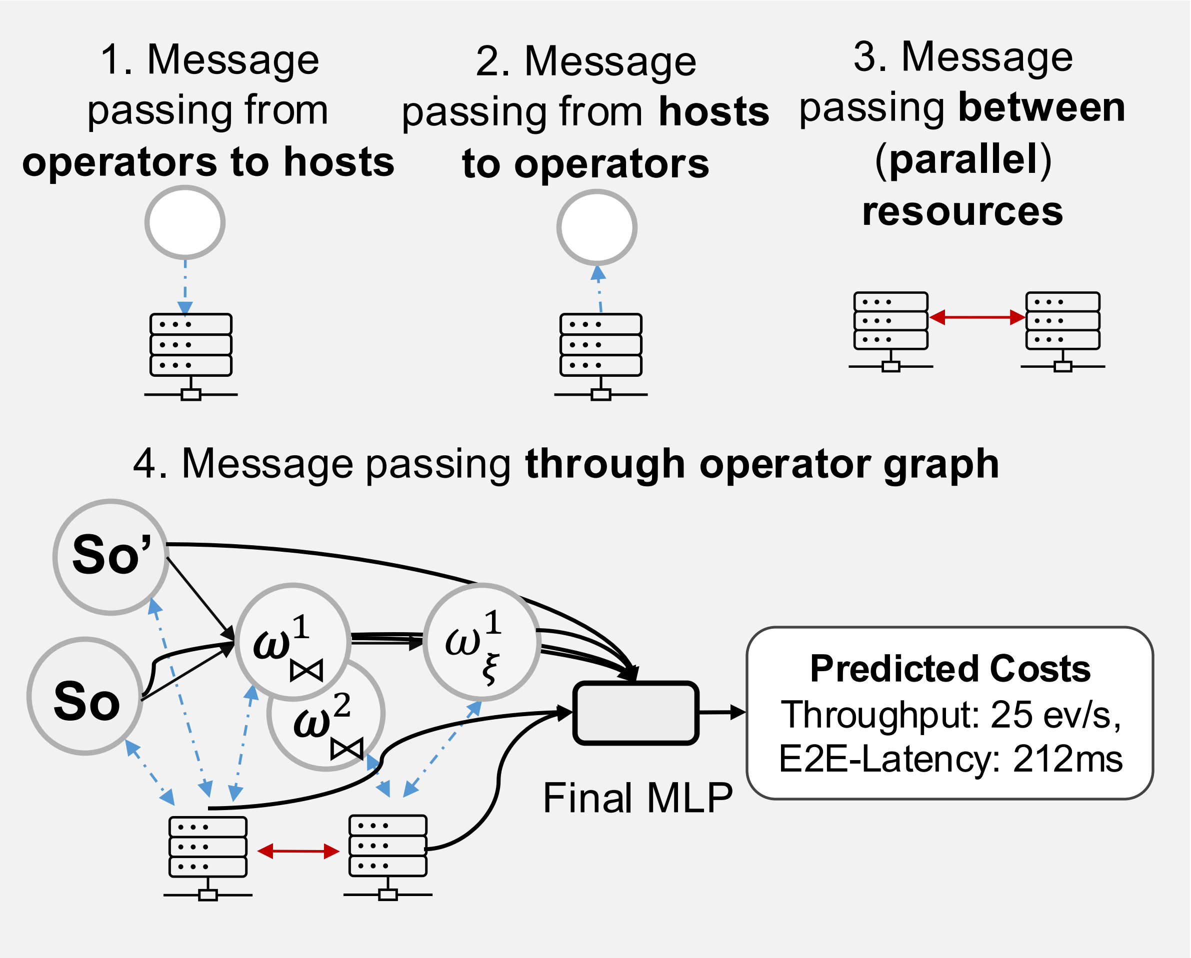


Learning Query Placement Costs with GNN

Neural Encoding



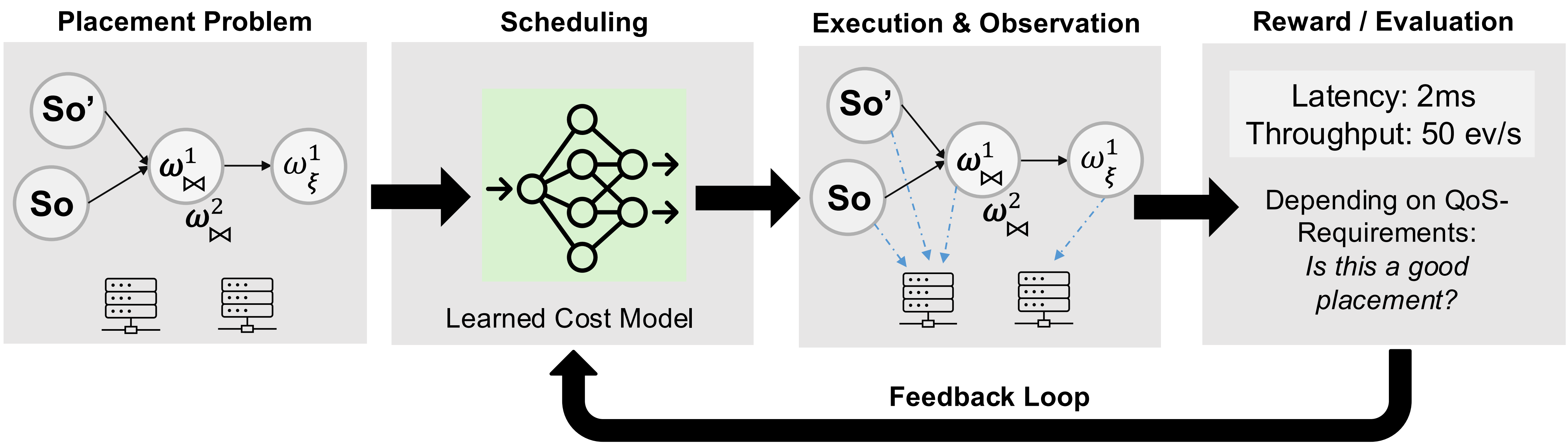
Novel Neural Message Passing



Specialized Models in a Nutshell

Specialized LCM
(e.g. RL tuned for workload)

Intuition: Improving model over time for a given workload by monitoring the results and iteratively updating the model

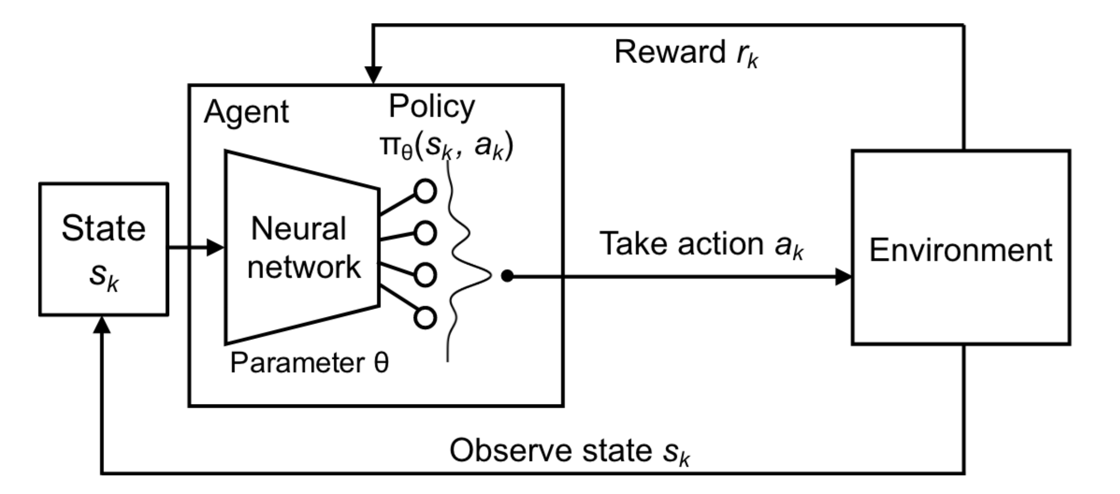


✅ **Advantages**
No human interaction required as policy improves over time
Avoids the massive collection of training data

❌ **Disadvantages**
Model gets tied towards seen workloads and does not **generalize**
Retraining required if workloads change

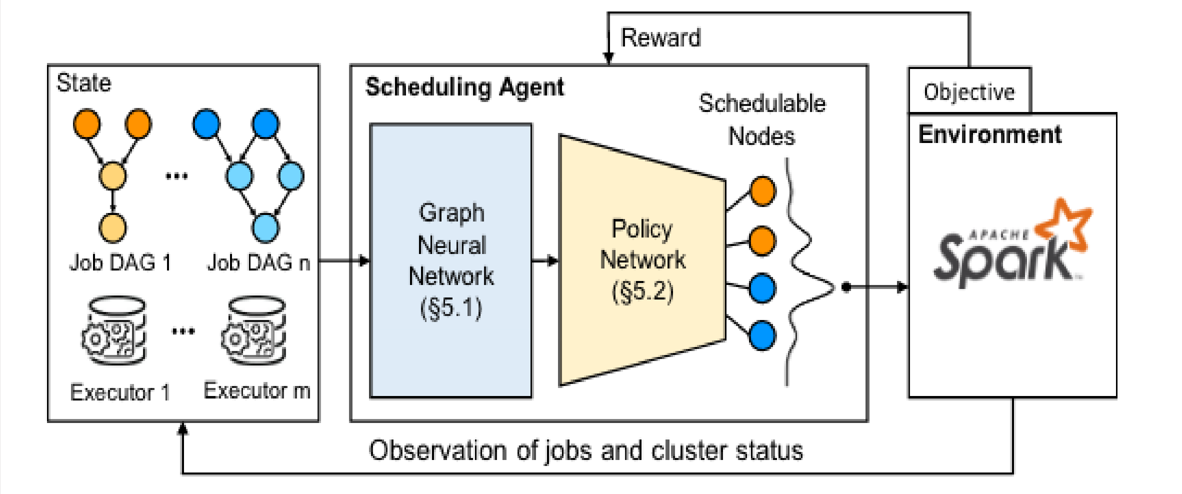
LCMs using **Reinforcement Learning**

Background: Reinforcement Learning



- Learning an **agent** by interacting with the environment
- Learning **policy** over time: Which actions to take given a system state?
- Assuming markov process: Actions are conditionally independent of the past

Decima: Learning Scheduling Algorithms with RL



- For a given query an agent uses a GNN and a policy network to come up with a schedule
- The schedule is executed on a spark cluster and observed
- The agent is updated by learning from a reward function of the given placement

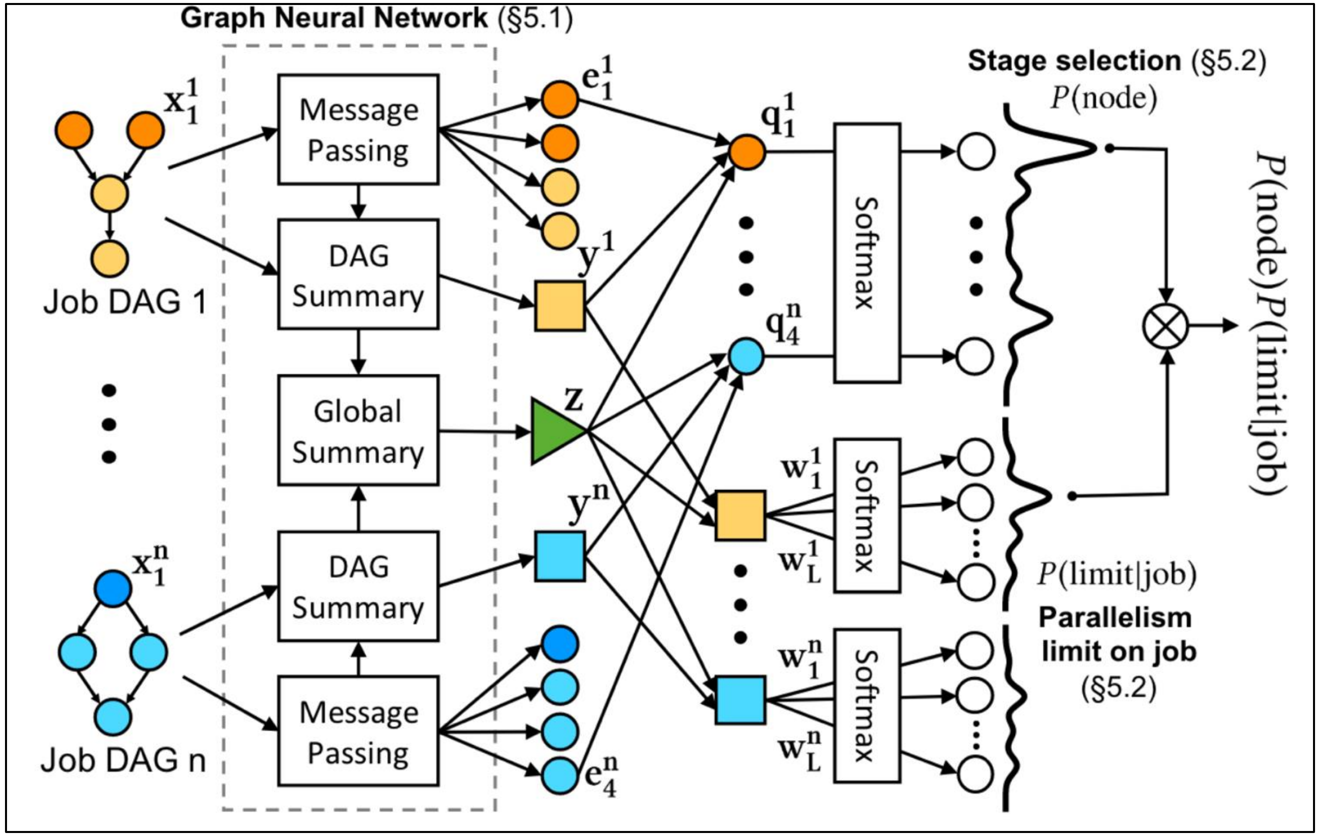
H.Mao, M. Schwarzkopf, S.B. Venkatakrishnan, Z. Meng, M. Alizadeh
Learning Scheduling Algorithms for Data Processing Clusters SIGCOMM 2019.

More approaches follow this idea:
- Moira (Foroni et al)
- Li et al

Decima: Learning Placement Costs

Features:
- number of tasks within operator
- average task duration
- number of executors working on the node
- available local executors

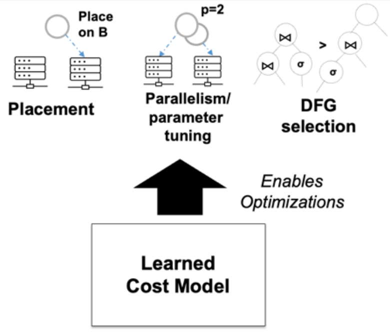
Embeddings:
- per-node embedding (e)
- per query embeddings (y)
- global summary



Output:
- the score of the schedule
- maximal parallelism degree

H.Mao, M. Schwarzkopf, S.B. Venkatakrishnan, Z. Meng, M. Alizadeh
Learning Scheduling Algorithms for Data Processing Clusters SIGCOMM 2019.

Summary: **LCMs for Stream Processing**



Role of LCMs in Optimizing Stream Processing Systems

	Model	Intended Task	Model Architecture	Input Features			
				Stream Statistics	Hardware Charact.	Hardware Monitoring	Query Plan
General LCMs	Moira	Cost Estimation (latency, throughput)	SVM	✓		✓	
	Imai et al.	Cost Estimation (throughput)	Linear Reg.		✓	✓	
	Li et al.	Cost Estimation (latency)	SVR		✓	✓	
	Zero-shot	Cost Estimation (latency, throughput)	GNNs	✓			✓
Optimization-oriented LCMs	ZeroTune	Operator Parallelism	GNNs	✓	✓		✓
	COSTREAM	Operator Placement	GNNs	✓	✓		✓
	Li et al.	Operator Placement	RL	✓		✓	
	Decima	Operator Placement	RL	✓		✓	
	Ni et al.	Operator Placement	RL	✓		✓	

Taxonomy on existing work

Generalizable LCM
(e.g., Zero-shot)

Specialized LCM
(e.g. RL tuned for workload)

Key Dimension:
Generalizability vs. Specialization

Open problems for batch and streaming systems

Training Data Collection

- *How can we collect the right labels efficiently?*
- *Do we need both optimal and non optimal query plans?*
- *How do we capture load fluctuations (streaming)?*

LCMs for Query Optimization

- *Which are the right models for query optimization?*
- *Which are the right metrics for LCMs beyond Q-error?*

Open problems for **batch** and **streaming** systems

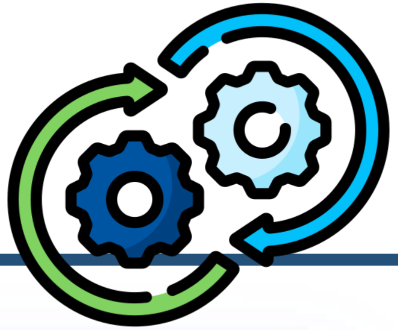
LCMs Evaluation

- *Which is a right benchmark with fixed training/validation/testing split?*
- *What are good metrics that reflect the downstream task?*

LCMs Interpretability & Explainability

- *Shall we aim for white-box models instead of NNs?*
- *What's the trade-off between "accuracy" and interpretability?*
- *How do we explain the results stemming from a black-box LCM?*

Open problems for **batch** and **streaming** systems

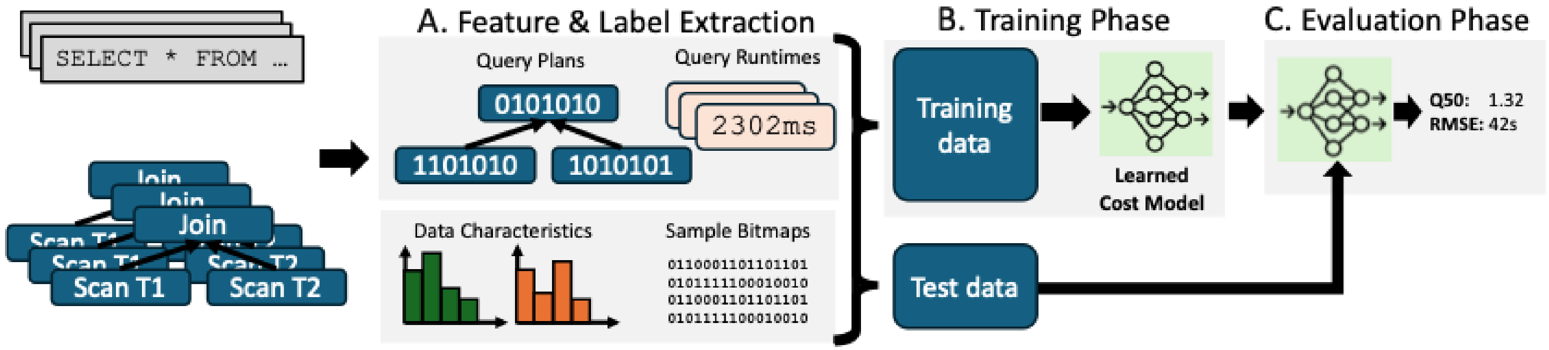


LCMs for Hybrid Workloads

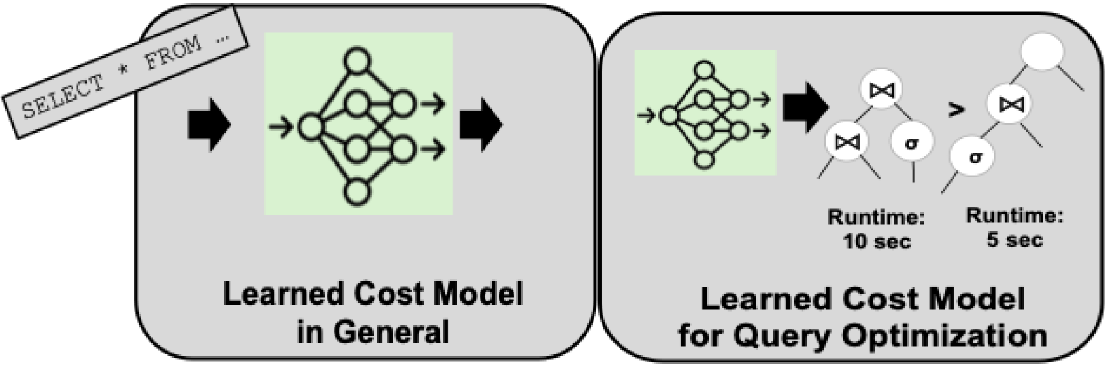
- *How can we build LCMs that support batch-stream workloads, commonly found in data lake settings?*
- *Do we need specialized LCMs per type, or could one be used?*

Summary

Learned Cost Models for Batch and Streaming Systems



- Featurization
- Model architecture
- Training data collection



Generalizable LCM (e.g., Zero-shot)

Specialized LCM (e.g. RL tuned for workload)