Towards a Learning Optimizer for Shared Clouds*

Chenggang Wu, Alekh Jindal, Saeed Amizadeh, Hiren Patel,
Wangchao Le, Shi Qiao, Sriram Rao

February 8, 2019

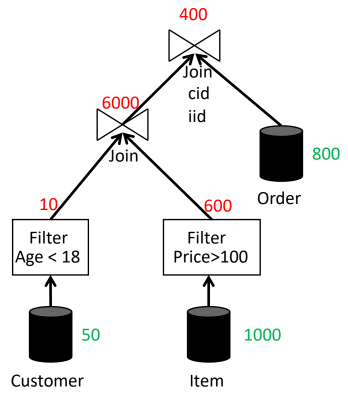
* *C. Wu, A. Jindal, S. Amizadeh, H. Patel, W. Le, S. Qiao, and S. Rao. Towards a Learning Optimizer for Shared Clouds. In PVLDB,* 12(3): 210–*222, 2018.*

Rise of Big Data Systems

- Hive
- Spark
- Flink
- Calcite
- BigQuery
- Big SQL
- HDInsight
- SCOPE
- Etc.

Declarative query interface
Cost-based query optimizer (CBO)

```
SELECT Customer.cname, Item.iname
   FROM Customer
   INNER JOIN Order
   ON Customer.cid == Order.cid
   INNER JOIN Item
   ON Item.iid == Order.iid
   WHERE Item.iprice > 100
   AND Customer.cage < 18;
```



Good plan => Good performance
Problem: CBO can make mistakes
esp. *Cardinality Estimation*

Rise of Big Data Systems

Hive
Spark
Flink
Calcite
BigQuery
Big SQL
HDInsight
SCOPE
Etc.

*The **root of all evil**, the Achilles Heel of query optimization, is the estimation of the size of intermediate results, known as **cardinalities**.* – [Guy Lohman, SIGMOD Blog 2014]

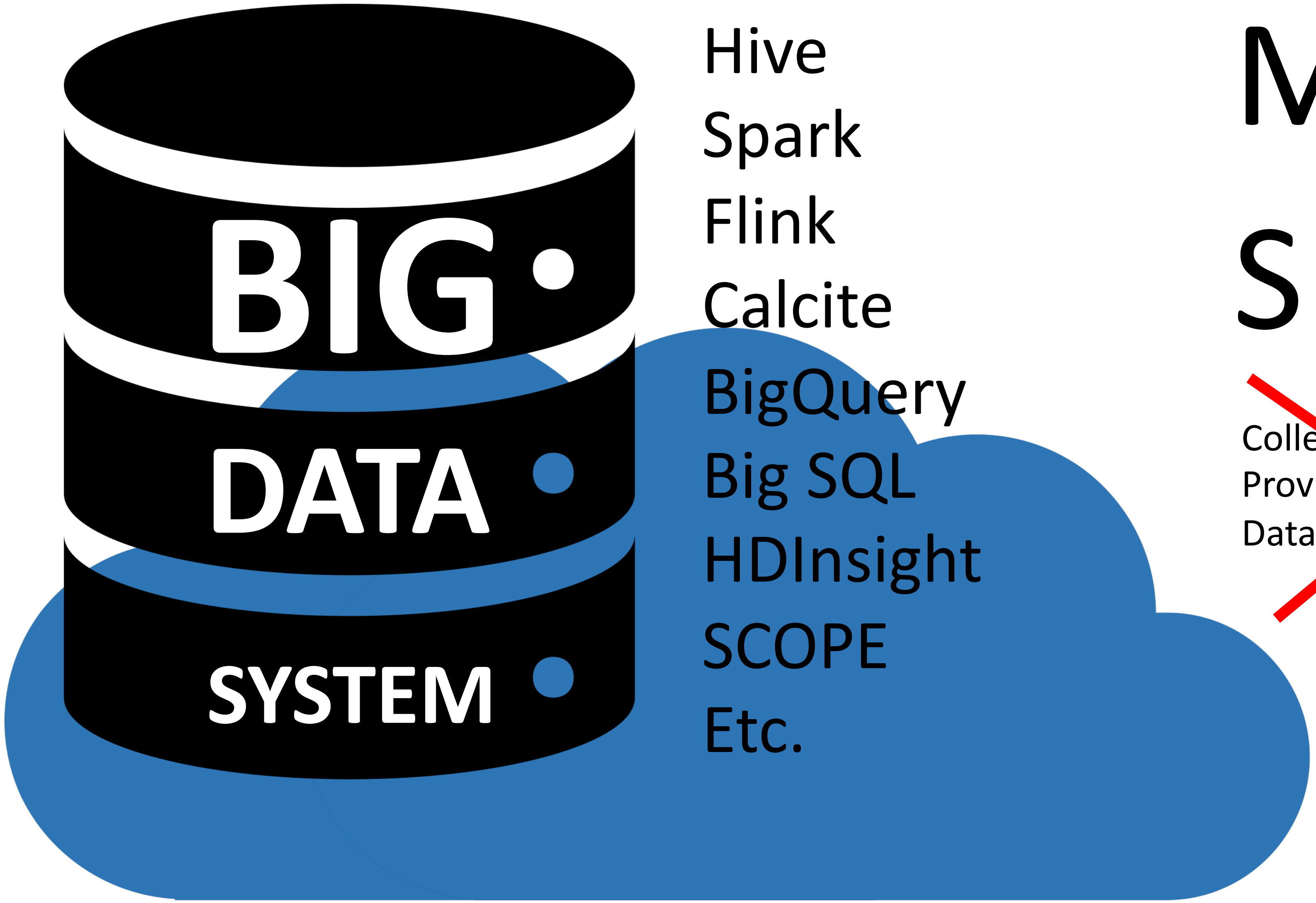
Rise of Big Data Systems

Hive
Spark
Flink
Calcite
BigQuery
Big SQL
HDInsight
SCOPE
Etc.

TUNING!

Collecting Statistics
Providing Query Hints
Database Administration

Rise of the Clouds



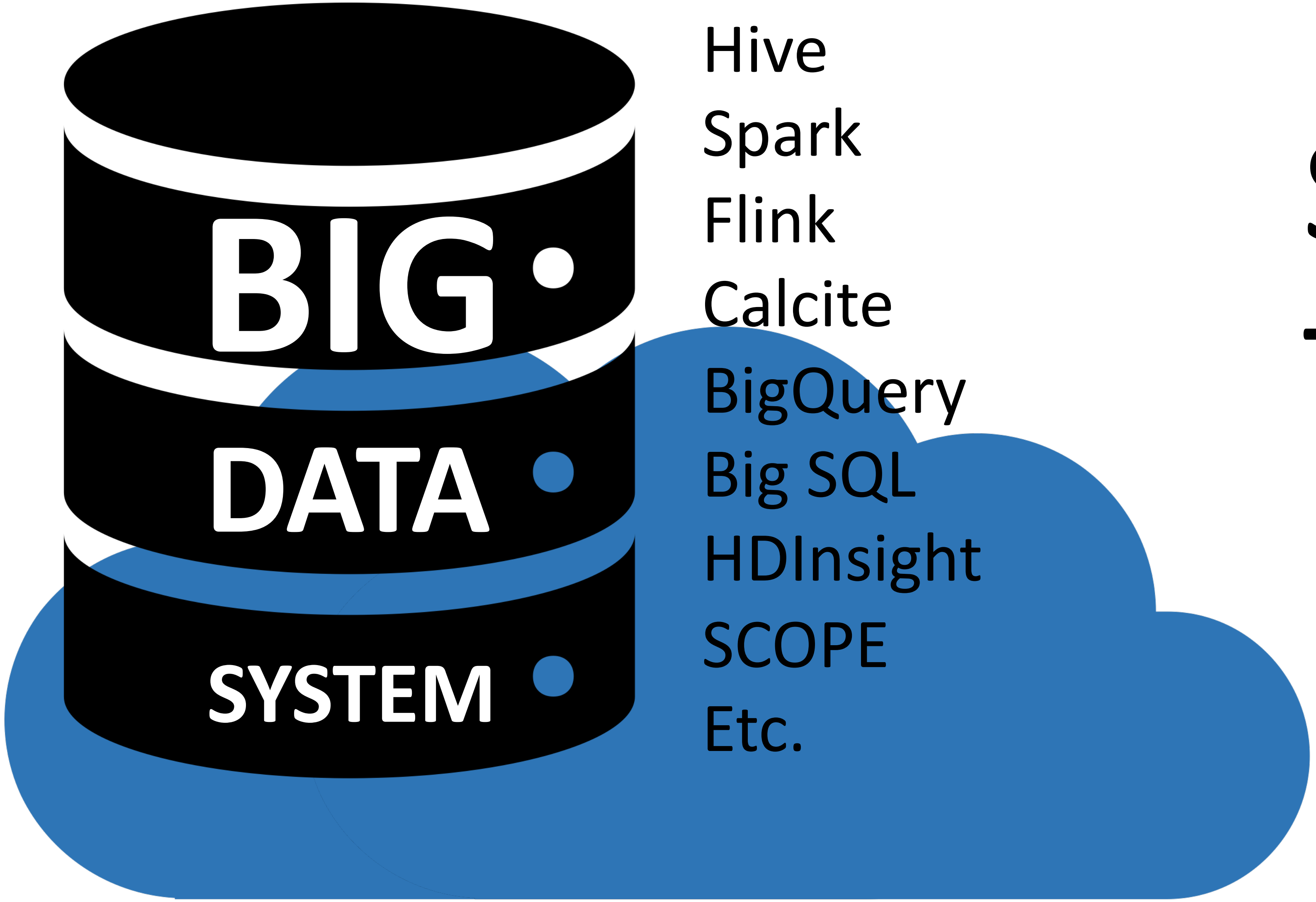
Hive
Spark
Flink
Calcite
BigQuery
Big SQL
HDInsight
SCOPE
Etc.

MANAGED
SERVERLESS

~~Collecting Statistics~~
~~Providing Query Hints~~
~~Database Administration~~

No Admin
No Expertise
No Control

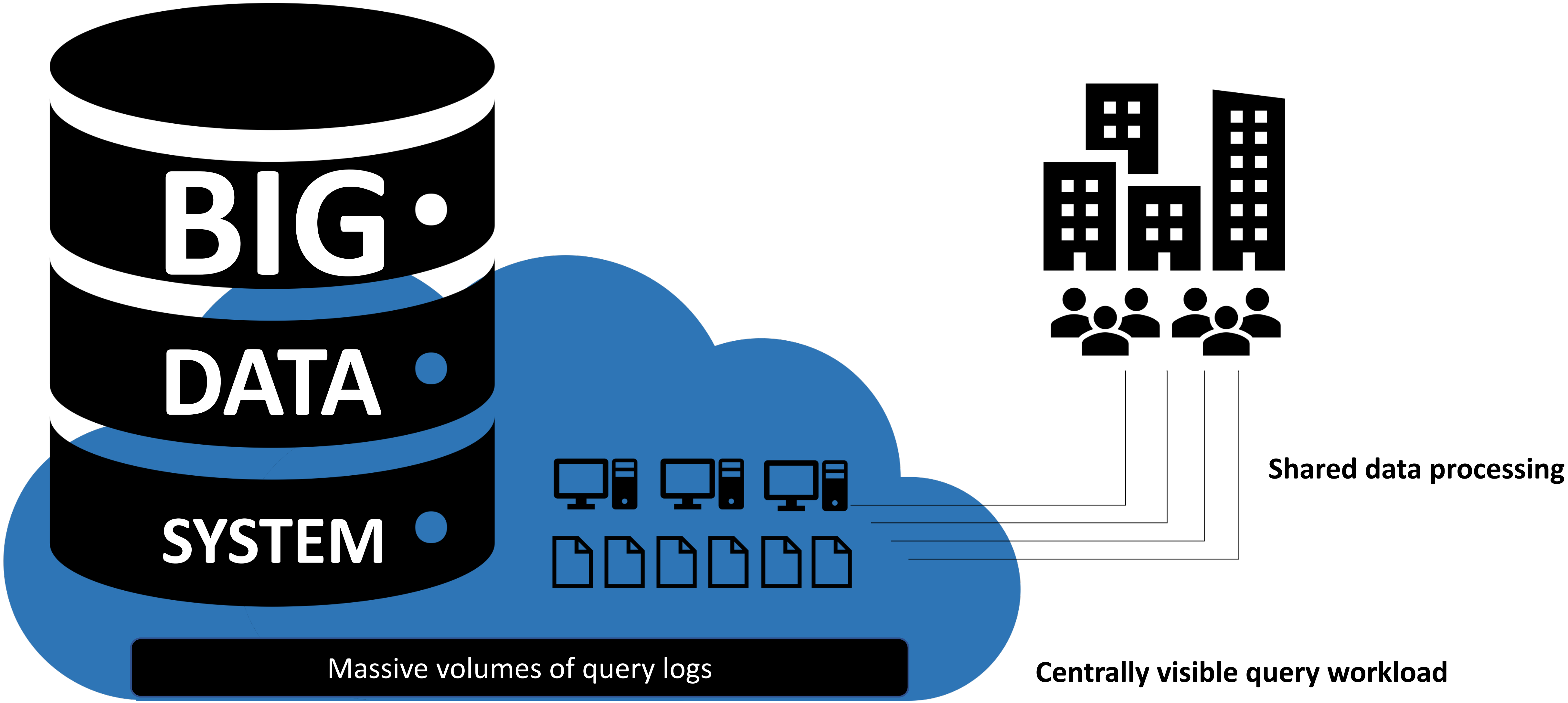
Rise of the Clouds



Hive
Spark
Flink
Calcite
BigQuery
Big SQL
HDInsight
SCOPE
Etc.

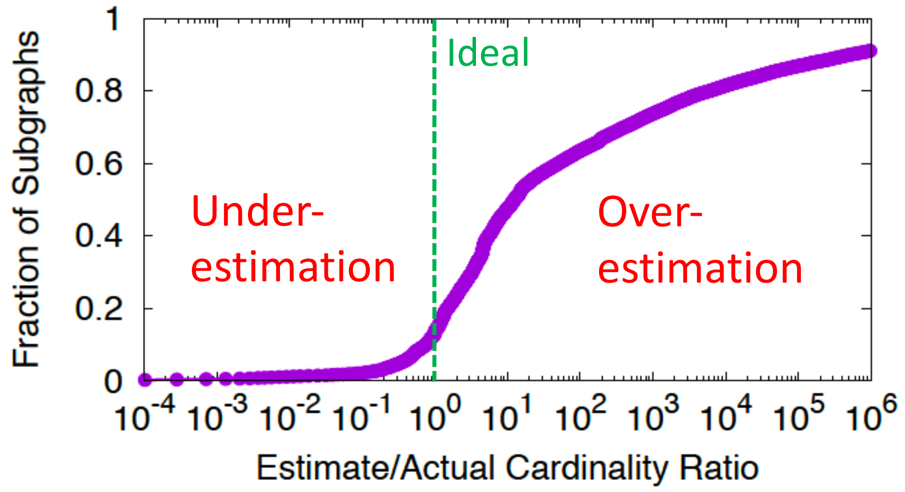
SELF
TUNING!

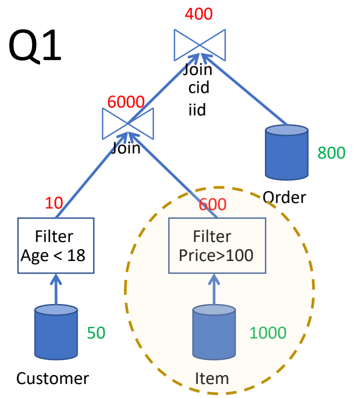
Hope: Shared Cloud Infrastructures

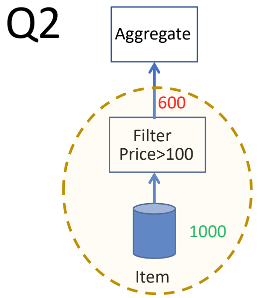


Cosmos: shared cloud infra at Microsoft

- SCOPE Workloads:
  - Batch processing in a job service
  - 100Ks jobs; 1000s users; EBs data; 100Ks nodes
- Cardinality estimation in SCOPE:
  - 1 day's log from Asimov
  - Lots of constants for best effort estimation
  - Big data, unstructured Data, custom code
- Workload patterns
  - Recurring jobs
  - Shared query subgraphs
- Can we *learn* cardinality models?







Learning Cardinality Model

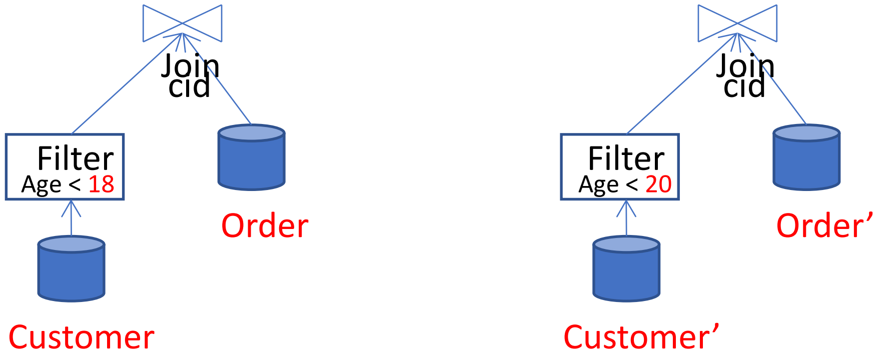
- Strict: cache previously seen values
  - Low coverage
  - Online feedback
- General: learning a single model
  - Hard to featurize
  - Hard to train
  - Prediction latency
  - Low accuracy
- Template: learning a model per subgraph template

  => No one-size-fits-all

Subgraph Type	Logical Expression	Parameter Values	Data Inputs
Strict	Fixed	Fixed	Fixed
General	Variable	Variable	Variable
Template	Fixed	Variable	Variable

Learned Cardinality Models

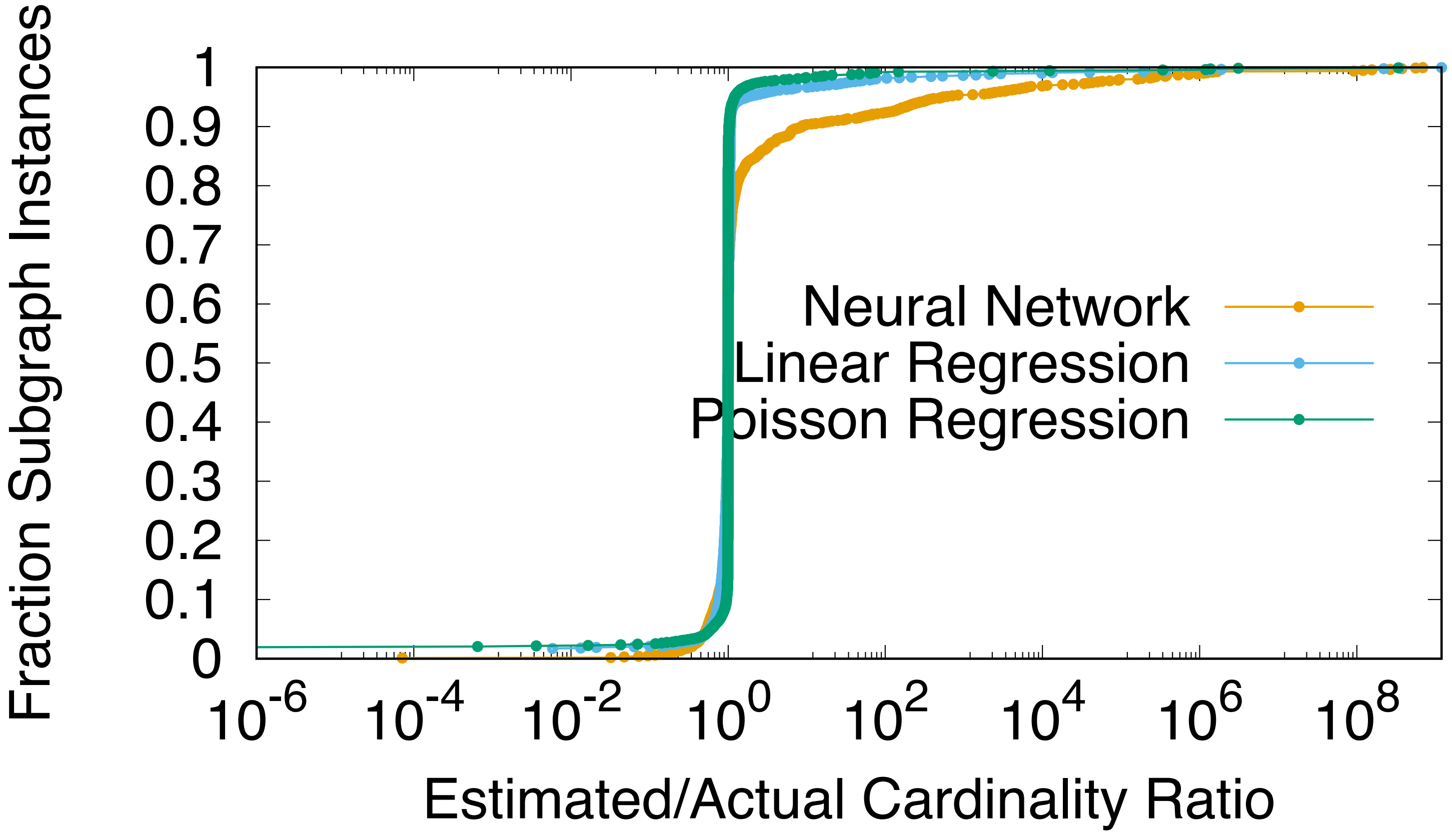
- Subgraph Template:
  - Same logical subexpression
  - Different physical implementation
  - Different parameters and inputs
- Feature Selection
- Model Selection
  - Generalized liner models due to their interpretability
  - More complex models, such as multi-layer perceptron harder to train



Name	Description
JobName	Name of the job containing the subgraph
NormJobName	Normalize job name
InputCardinality	Total cardinality of all inputs to the subgraph
Pow(InputCardinality, 2)	Square of InputCardinality
$Sqrt$(InputCardinality)	Square root of InputCardinality
Log(InputCardinality)	Log of InputCardinality
AvgRowLength	Average output row length
InputDataset	Name of all input datasets to the subgraph
Parameters	One or more parameters in the subgraph

Model	Percentage Error	Pearson Correlation
Default Optimizer	2198654	0.41
Adjustment Factor (LEO)	1477881	0.38
Linear Regression	11552	0.99
Neural Network	9275	0.96
Poisson Regression	696	0.98

Accuracy: 10-fold cross validation

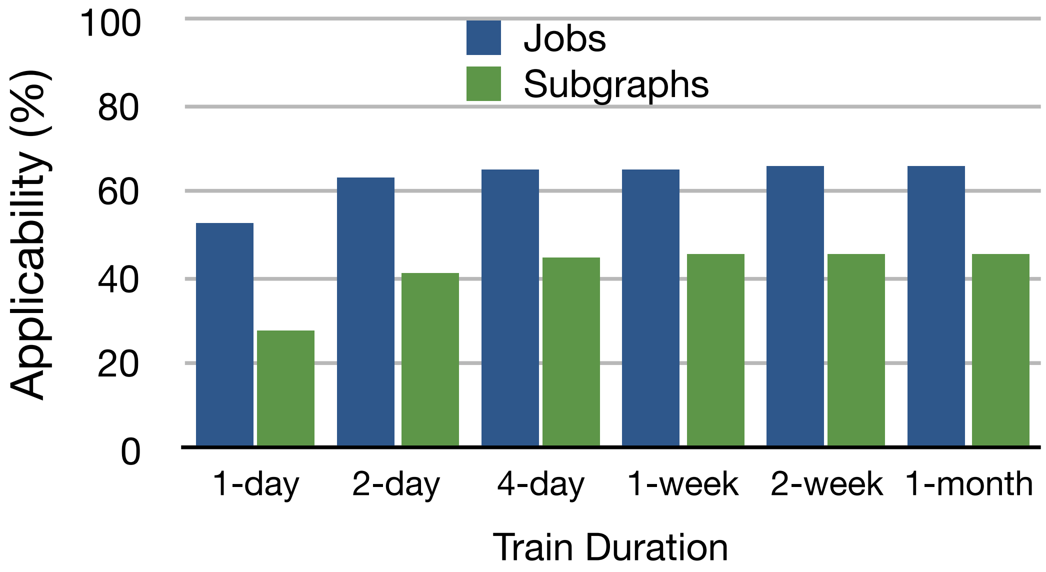


Model	75th Percentile Error	90th Percentile Error
Default SCOPE	74602%	5931418%
Poisson Regression	1.5%	32%

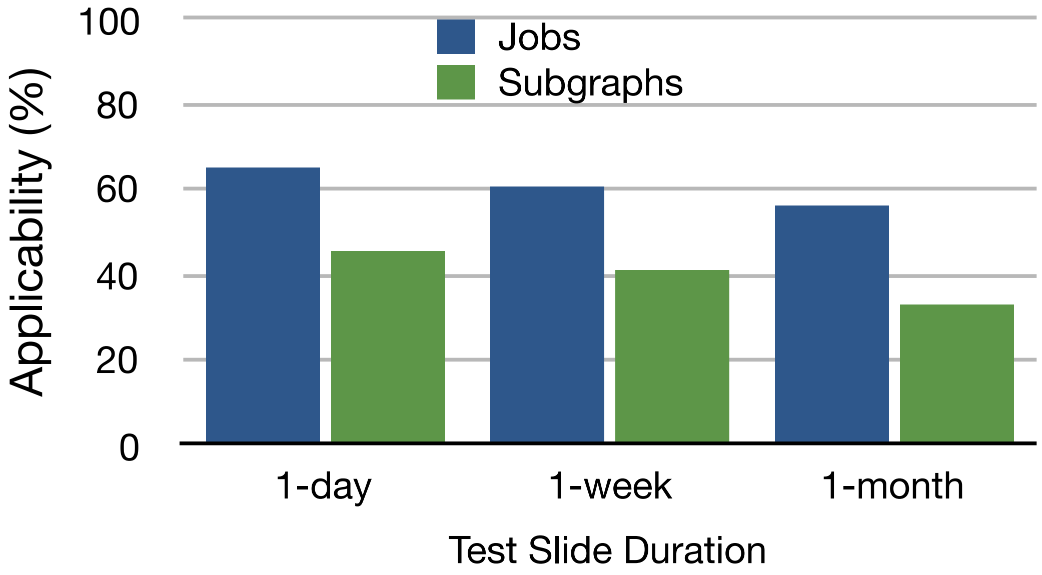
Note: Neural network overfits due to small observation and feature space per model

Applicability: %tage subgraphs having models

Varying Training Window



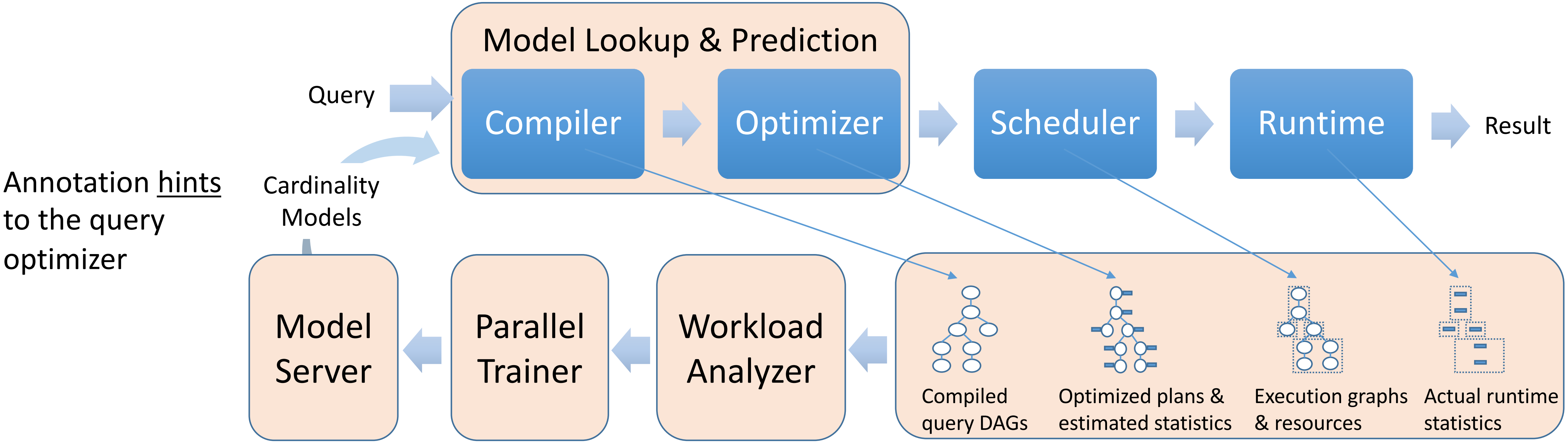
Sliding Test Window



End-to-end Feedback Loop

Easy to featurize with low overhead

Accurate and easy to understand

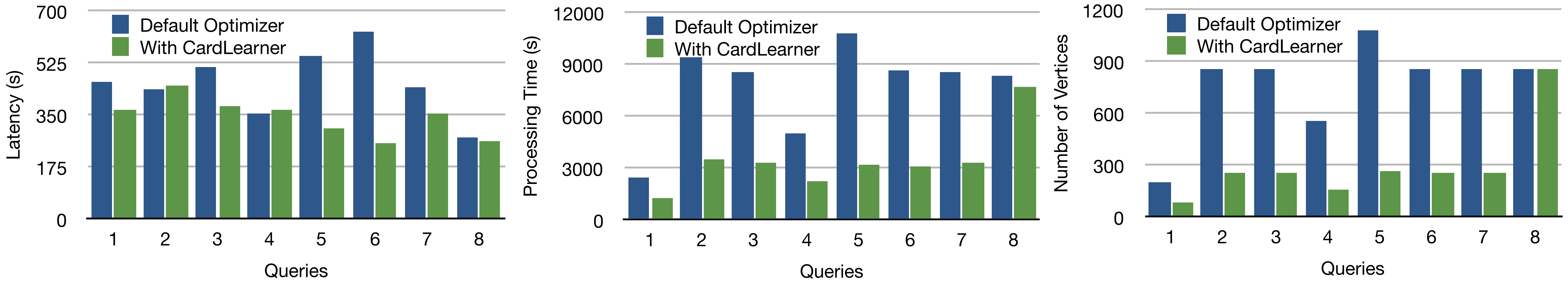


Trained offline over new batches of data

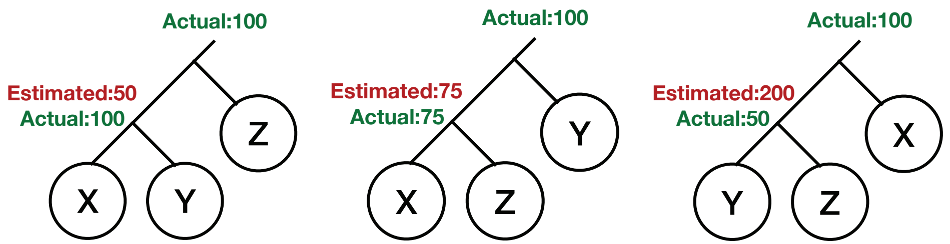
Large number of smaller, highly accurate models

Performance

- Subset of hourly jobs from Asimov
- These queries process unstructured data, use SPJA operators, and a UDO
- Re-ran the queries over same production data, but with redirected output



Avoiding Learning Bias



- Learning only what is seen
- Exploratory join ordering
  - Actively try different join orders
  - Pruning: discard plans with subexpressions that are more expensive than at least one other plan
  - Maximize new observations when comparing plans
- Execution strategies
  - Static workload tuning
  - Using sample data
  - Leveraging recurring/overlapping jobs

Takeaways

- Big data systems increasingly use cost-based optimization
- Users cannot tune these systems in managed/serverless services
- Hard to achieve a one-size-fits-all query optimizer
- Instance optimized systems are more feasible
- Very promising results from SCOPE workloads:
  - Could achieve very high accuracy
  - Reasonably large applicability, could further apply exploration
  - Performance gains, most significant being less resource consumption
- Learned cardinality models a step towards self-learning optimizers