Peregrine: workload optimization for cloud query engines*

Alekh Jindal

Gray Systems Lab

* **Peregrine: Workload Optimization for Cloud Query Engines**. Alekh Jindal, Hiren Patel, Abhishek Roy, Shi Qiao, Jarod Yin, Rathijit Sen, Subru Krishnan. *SOCC 2019*.


Engine
Workload
DBA

On-Premise

DBA

On-Premise

On-Premise

Cloud Query Engines

- Setup, installation, maintenance taken care of
- On-demand provisioning, pay as you go

Reality Check for providers:

- System developers == virtual DBAs!
- Too many cloud users, compared to system developers
- Too many support requests; often redundant
- Less time for feature development

Reality Check for customers:

- Lots of services to choose from (even within Azure, GCP, AWS)
- Lot of knobs to tune for **good perf** and **low cost**
- Lack of control; and lack of expertise
- And, **the DBA is gone**!

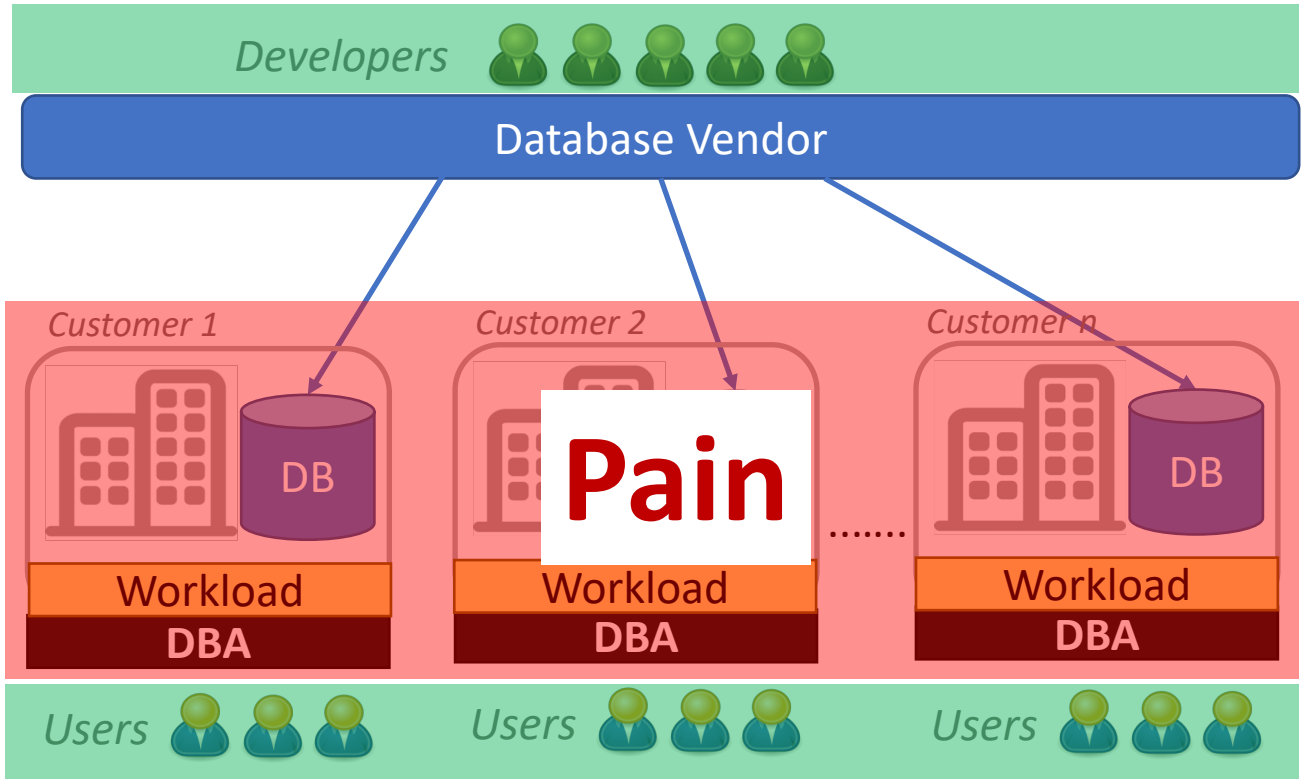
Cosmos: big data infra at Microsoft

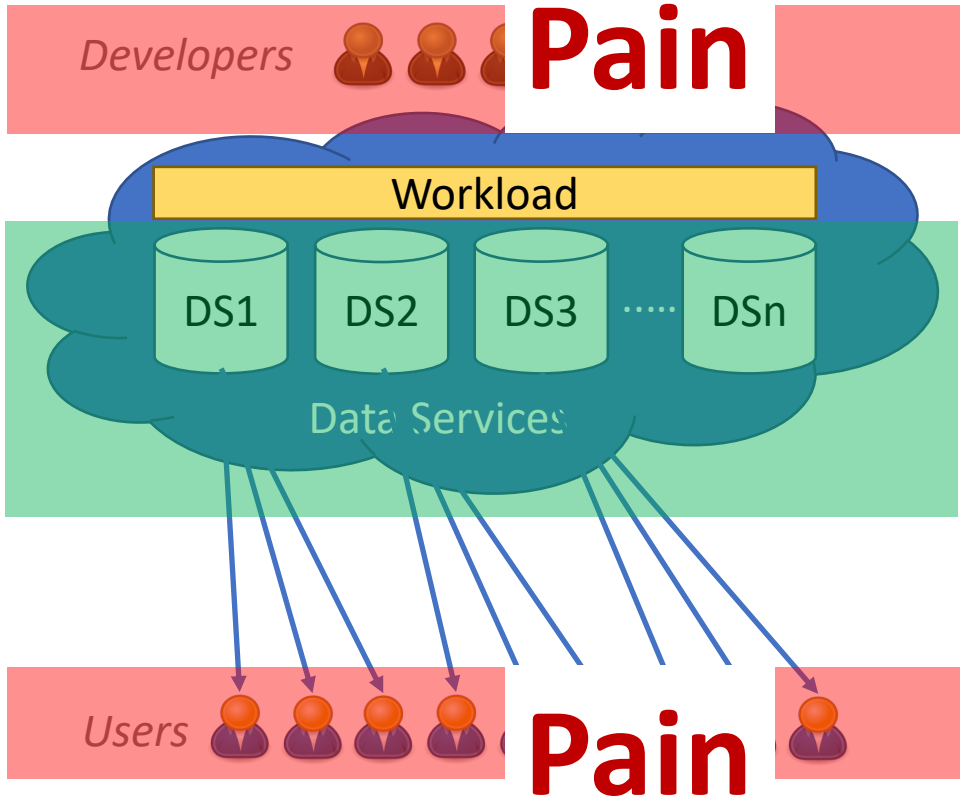
- 100s of thousands of machines
- Exabytes of data at rest; Petabytes ingress/egress daily
- 500k+ batch jobs / day
- 3B+ tasks executed / day
- 10s of millions interactive queries / day
- 10s of thousands of SCOPE developers
- 1000s of teams

The missing DBA and the growing pain in Cosmos

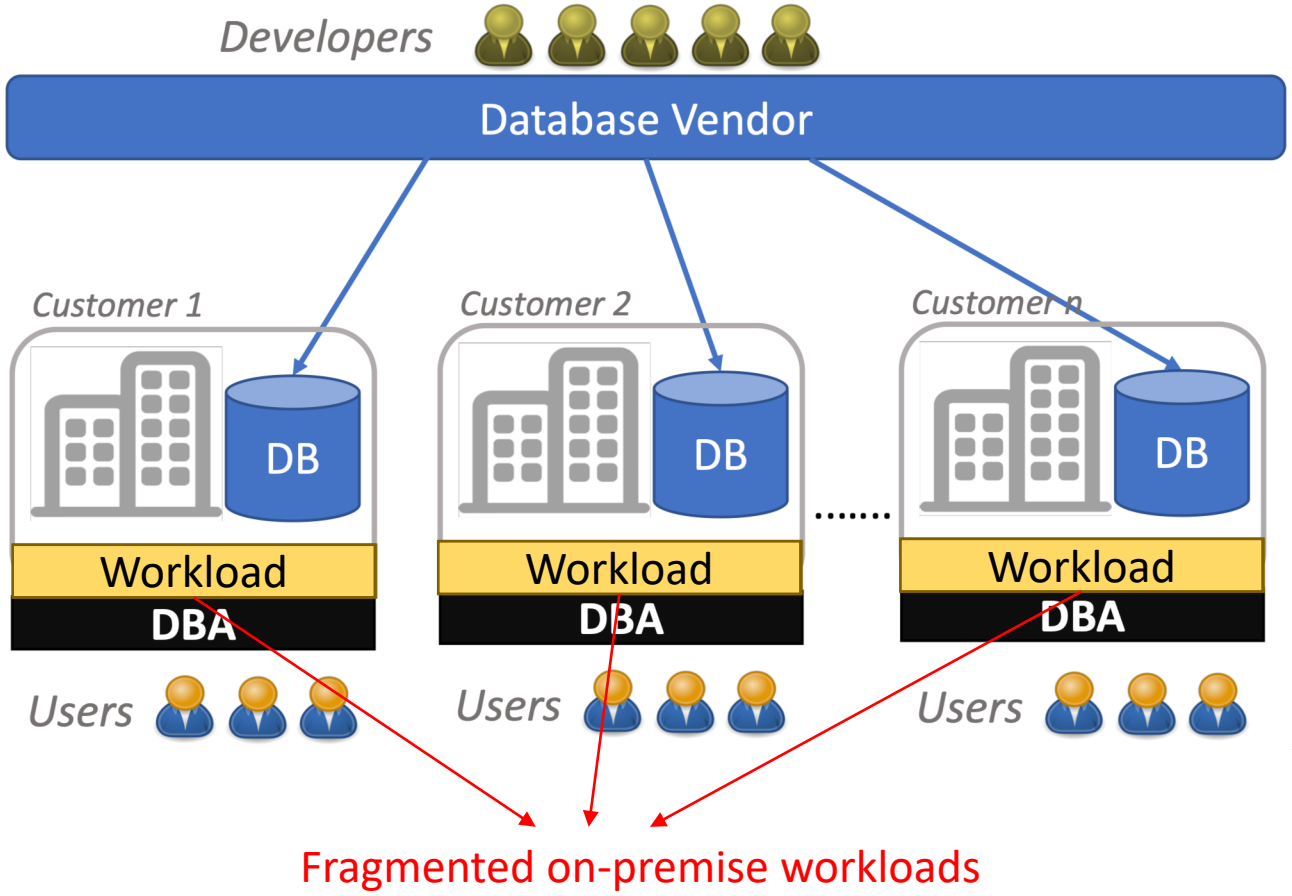
- Large number of knobs/hints at script, data, plan level
  - Only few expert users
  - Rest need guidance
  - Survey: better tooling for improving SCOPE queries
- Support challenge
  - 10s of thousands incidents / years
  - 10 incidents per system developer on call
  - 100x users compared to system developers
  - ~10% growth in SCOPE workload in 2019

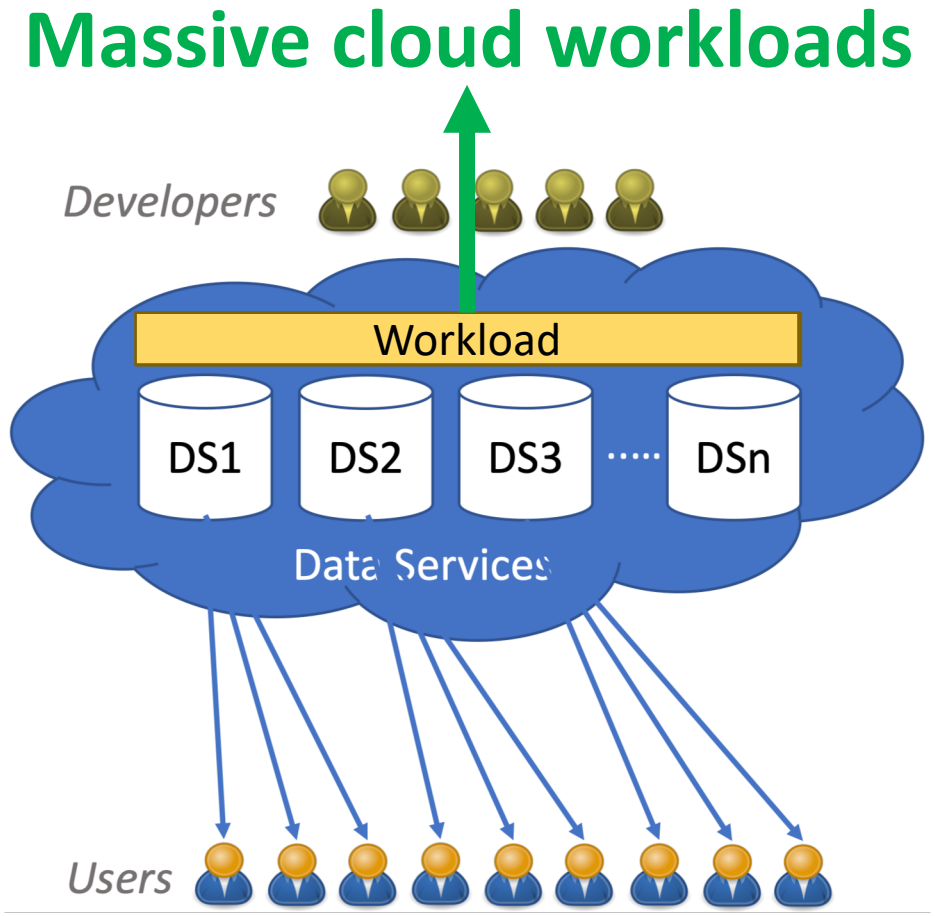
On-premise pain -> Cloud pain





The cloud opportunity





The Cosmos opportunity

Job metadata
name, user, account, submit/start/end times

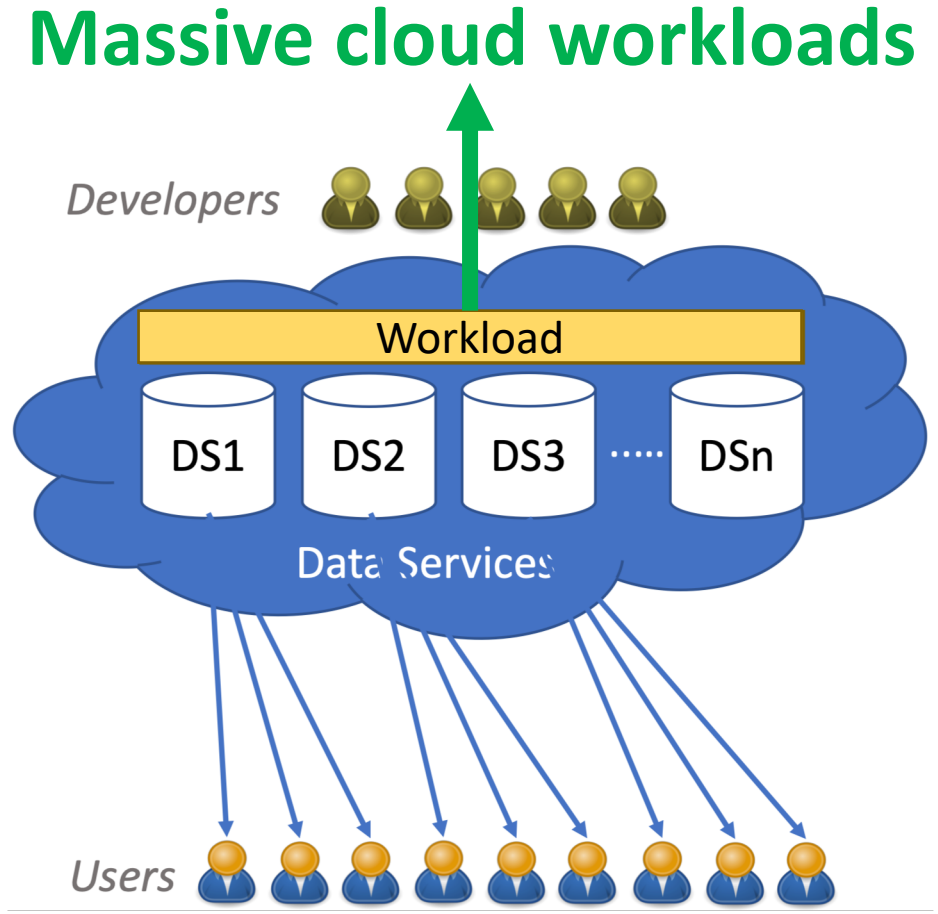
Query plans
logical, physical, stage graph, estimates

Runtime statistics
Operator-wise observables

Task level logs
start/end events

Machine counters
CPU, IO, etc.

Several TBs of metadata / day



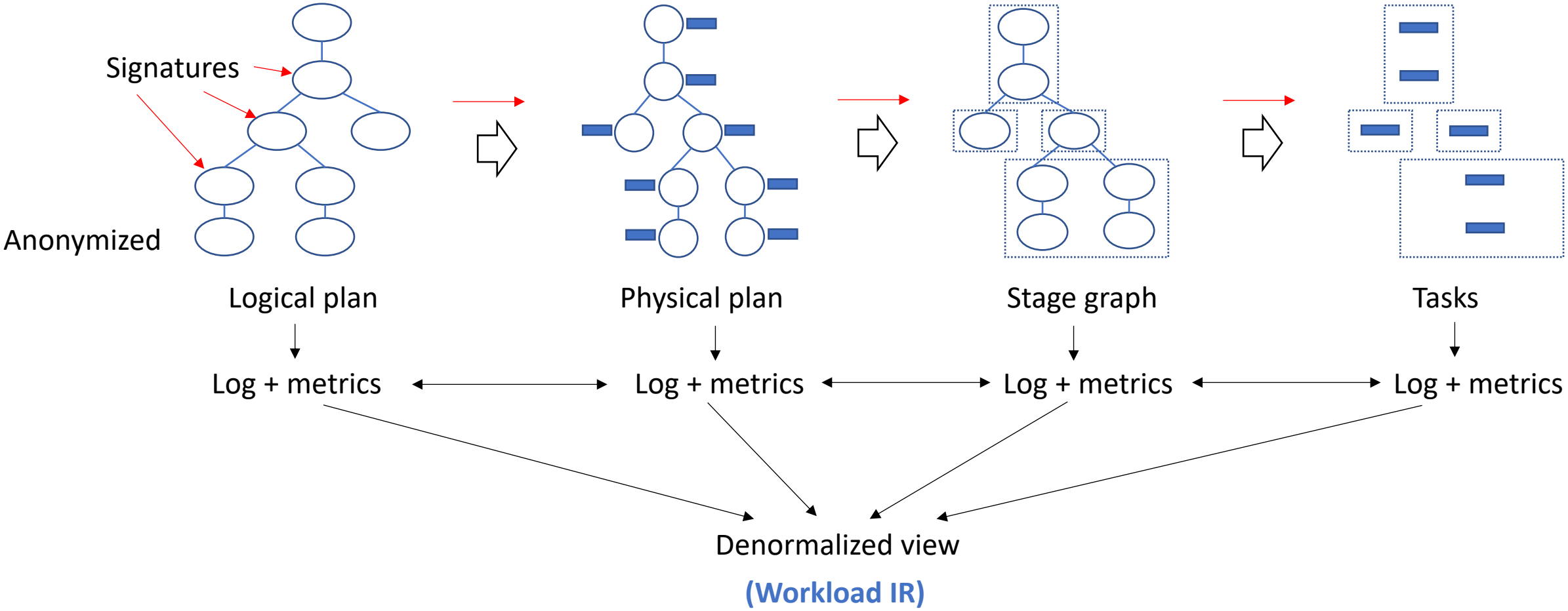
The case for a workload optimization platform

- DBA-as-a-Service
  - Another service in the cloud (easier integration)
  - Based on cloud workloads at hand (instance optimization)
- Engine agnostic
  - Not specific to different query engines, e.g., SCOPE, Spark, SQL DW, or etc.
  - E.g., view selection is still the same problem
- Global optimizations
  - Cloud workloads are organized into data pipelines
  - People often care about end-to-end aggregate costs in the cloud

Step 1: workload representation

Instrument, log, and collect workload characteristics

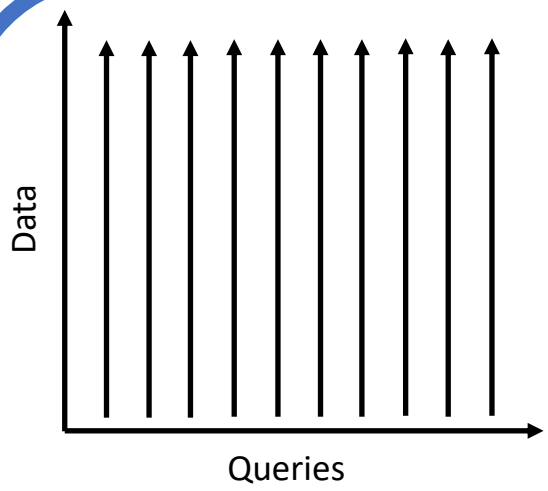
Engine-agnostic workload representation



Step 2: optimize for patterns

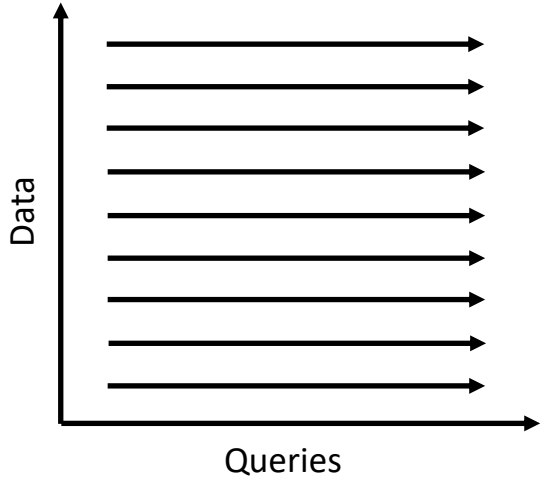
Typical workload patterns

- Consider a simplified 2D space of data and queries



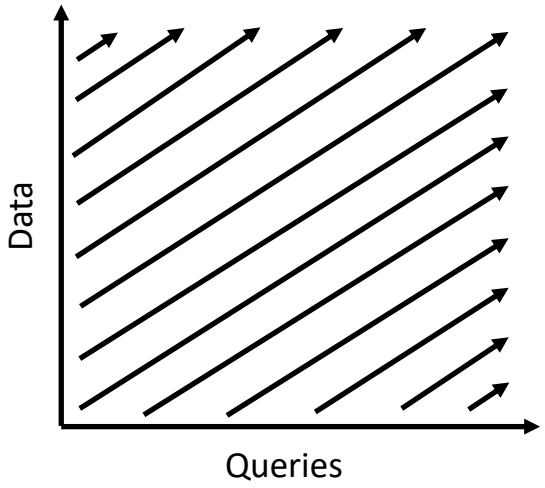
Recurring

Query templates appear over newer datasets



Similarity

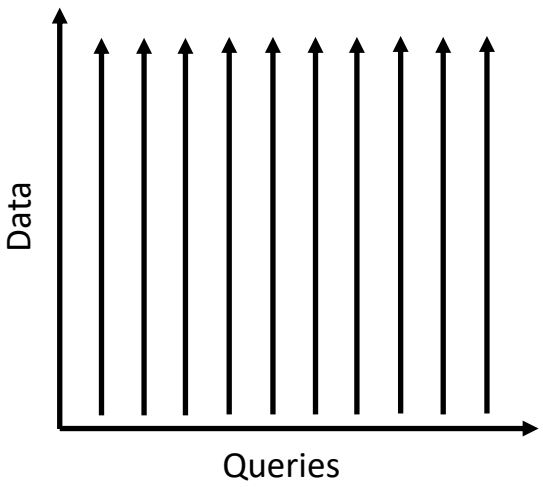
Queries over same datasets have similarities



Dependency

Queries depend on datasets produced by previous queries

Recurring pattern



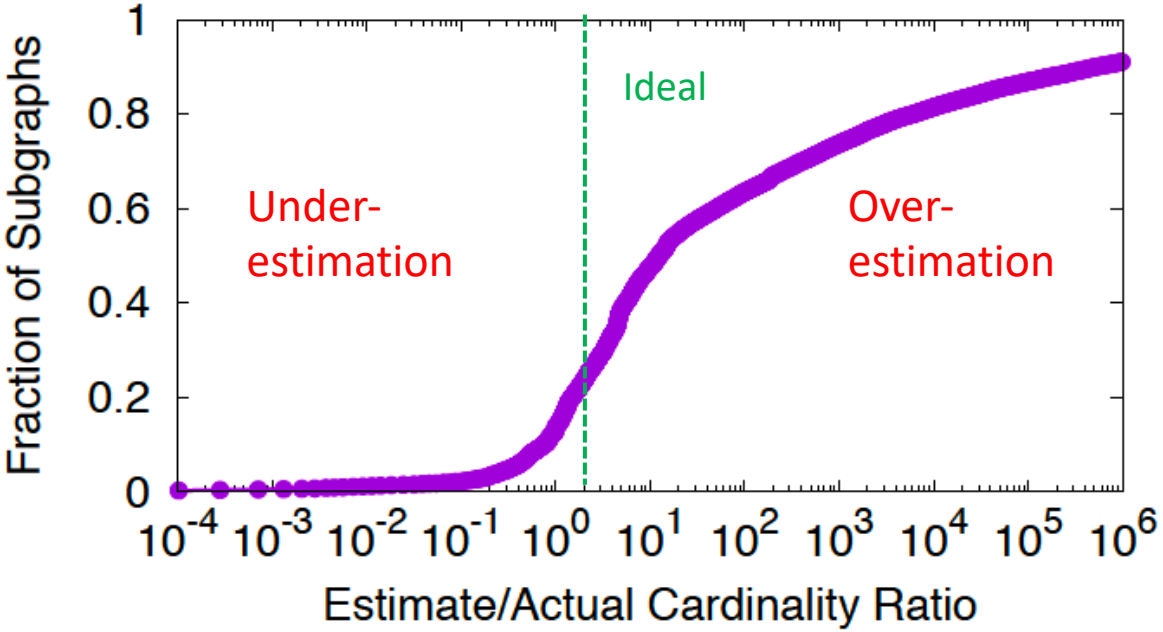
Recurring

Query templates appear over newer datasets

- Majority of production workloads
  - There is a regular ETL needed before other things can happen
- Opportunity to learn from the past
- Examples
  - Learned cardinality
  - Learned cost models
  - Learned resources
  - Learned etc.

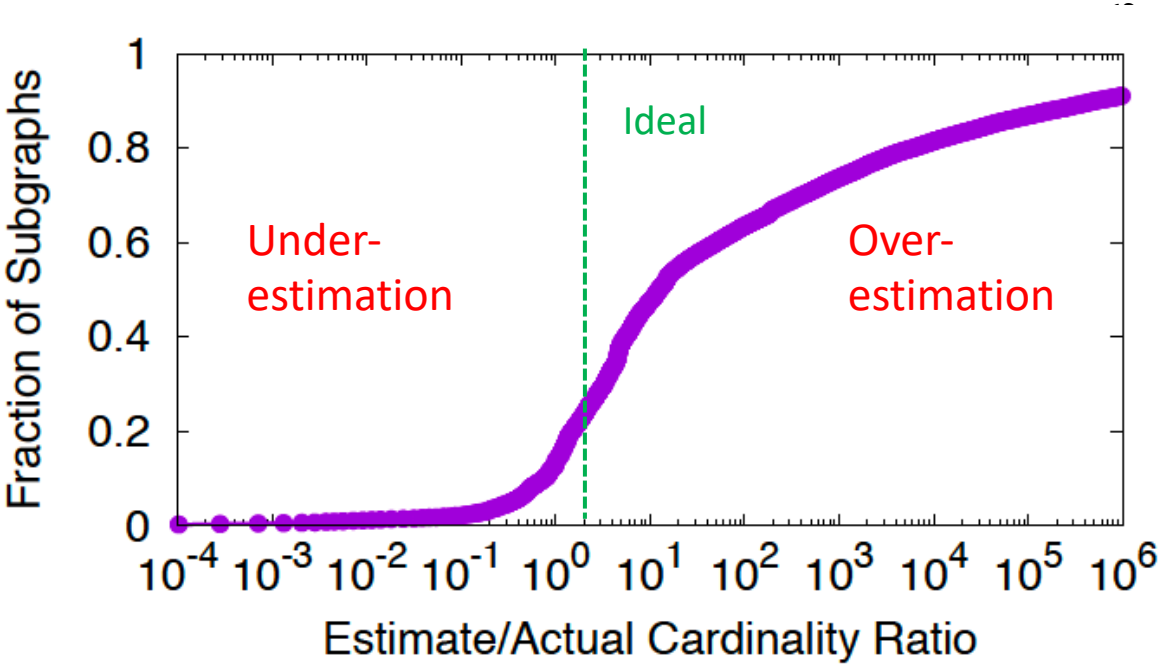
Recap from NWDS'19

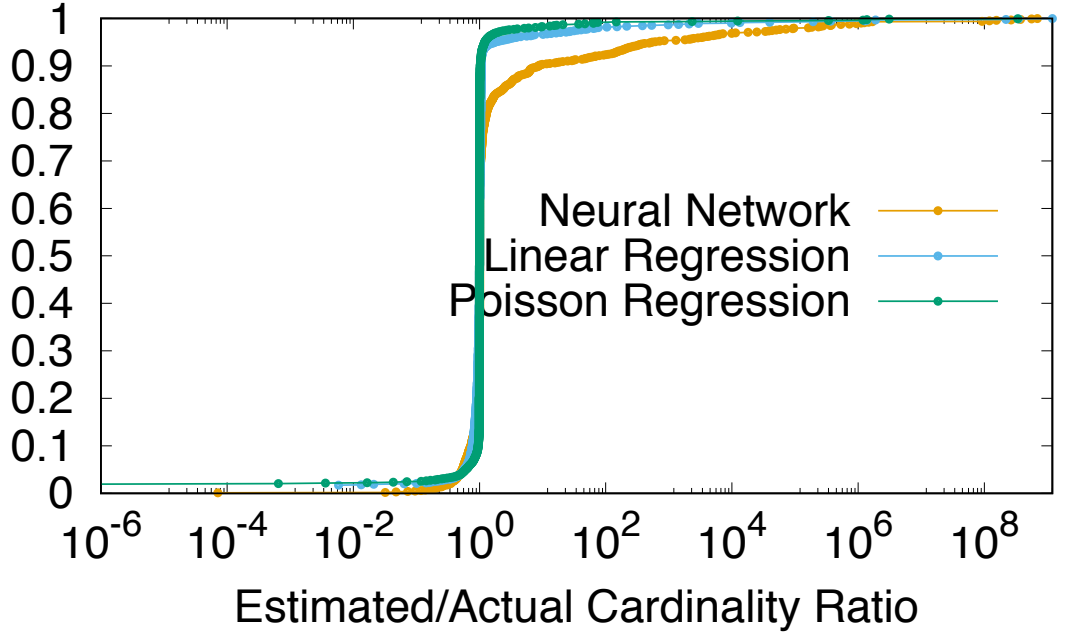
SCOPE Cardinality Estimation



Recap from NWDS'19

SCOPE Cardinality Estimation

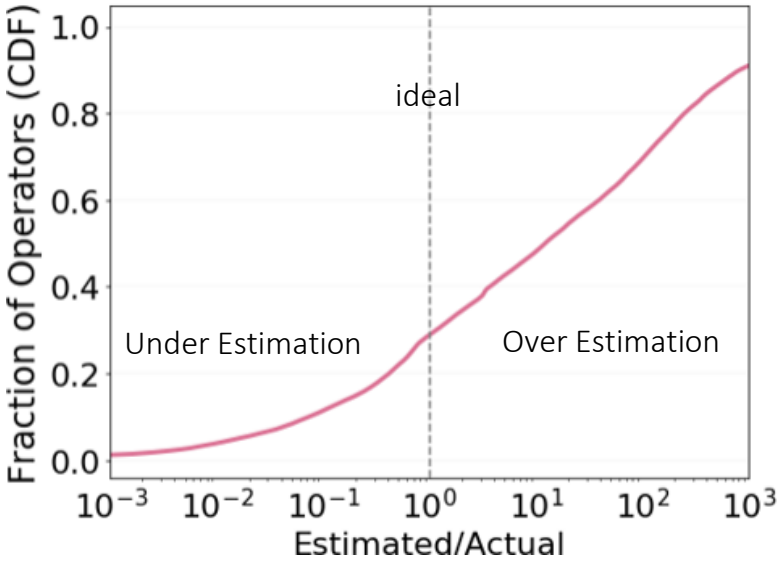


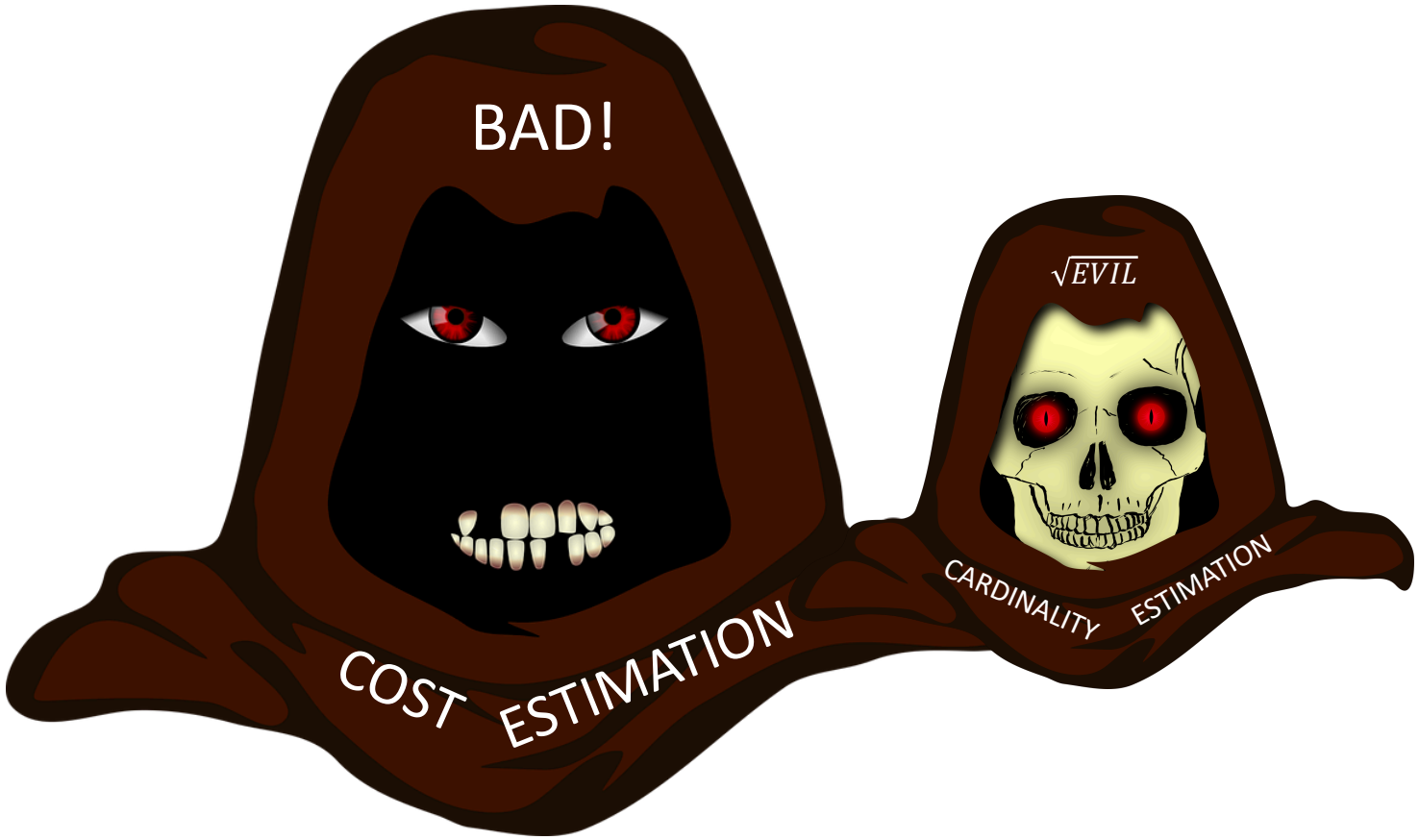


Towards a Learning Optimizer for Shared Clouds.
Chenggang Wu, Alekh Jindal, Saeed Amizadeh, Hiren Patel, Wangchao Le, Shi Qiao, Sriram Rao. *VLDB 2019*.

SCOPE Cost Estimation

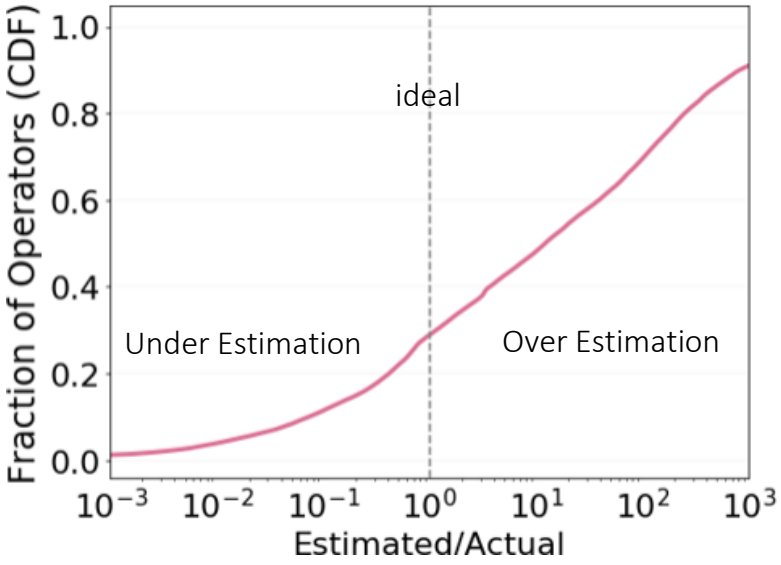
- Costs models are orders of magnitude off!





SCOPE Cost Estimation

- Costs models are orders of magnitude off!



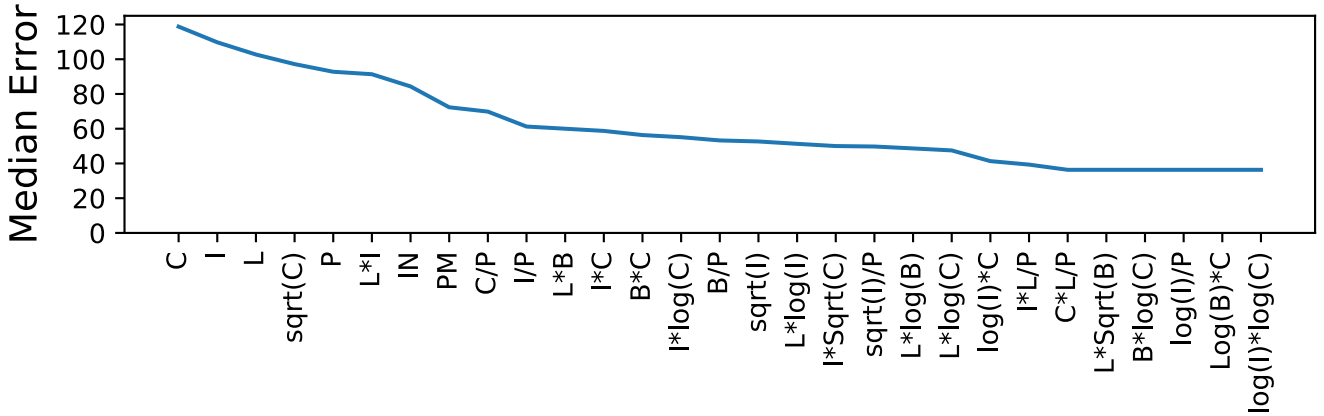
Manually tuned cost model

Feeding perfect cardinalities

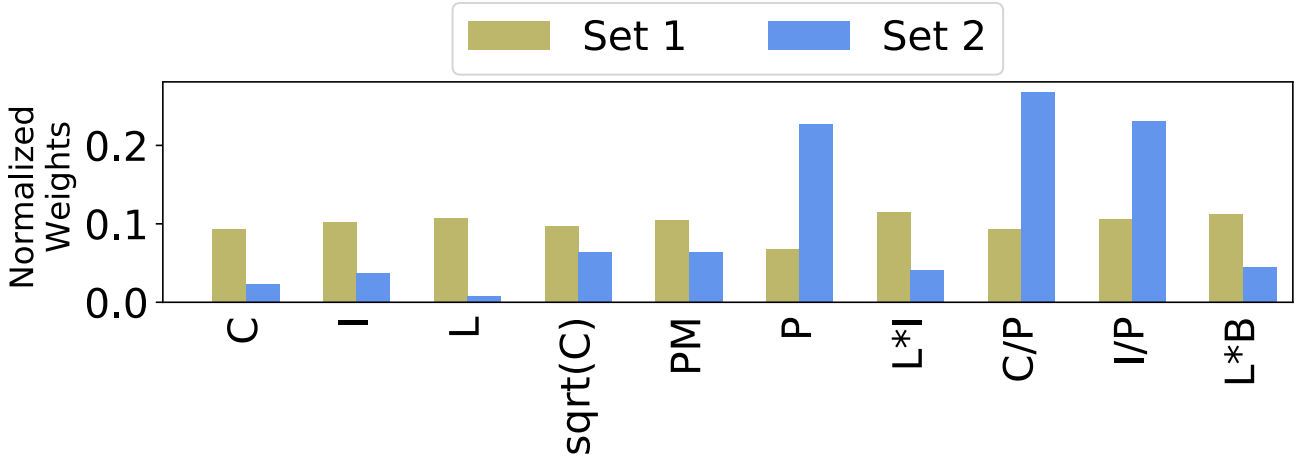
- Pervasive use of user defined functions
- Complexity of big data systems
- Variance in the cloud environments

Why cardinality is not enough?

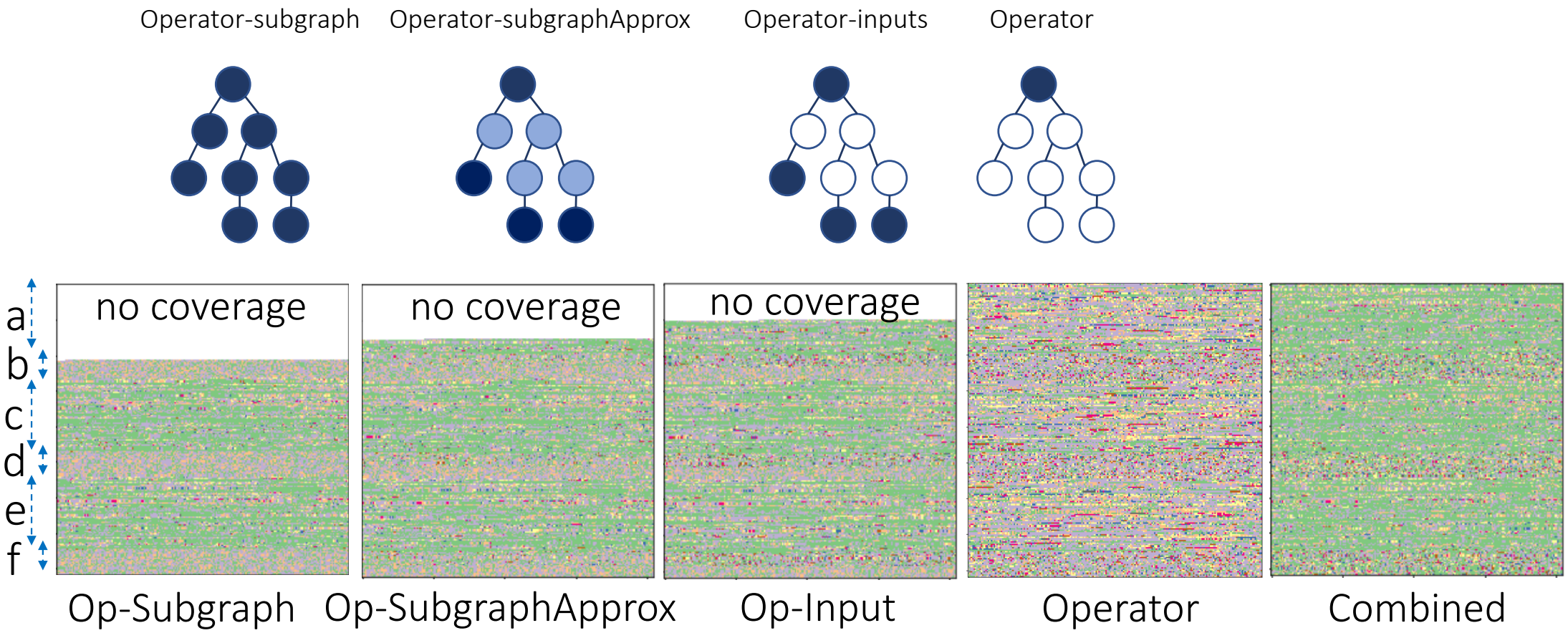
- Incrementally add features
- Error drops from 110% to 40%
- Additional transformations needed
- Hard to come up with such heuristics



- Two sets of hash join instances
- Different feature weights
- Hard to instance optimize manually

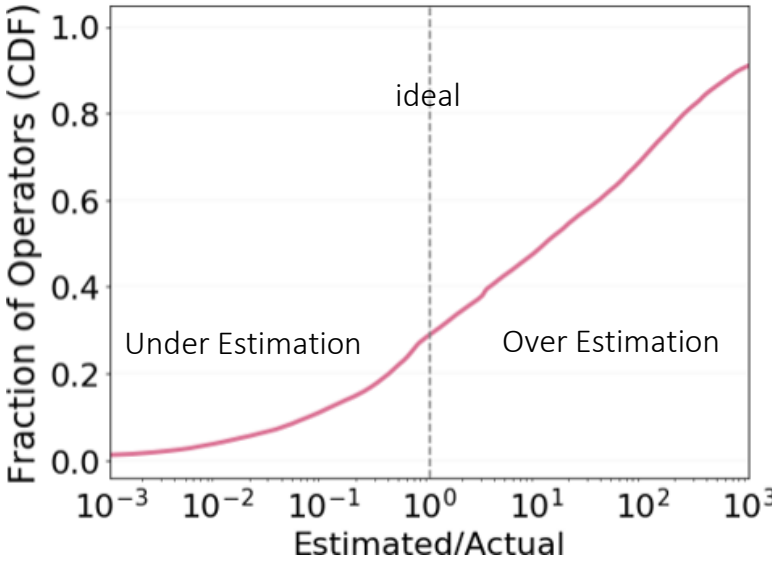


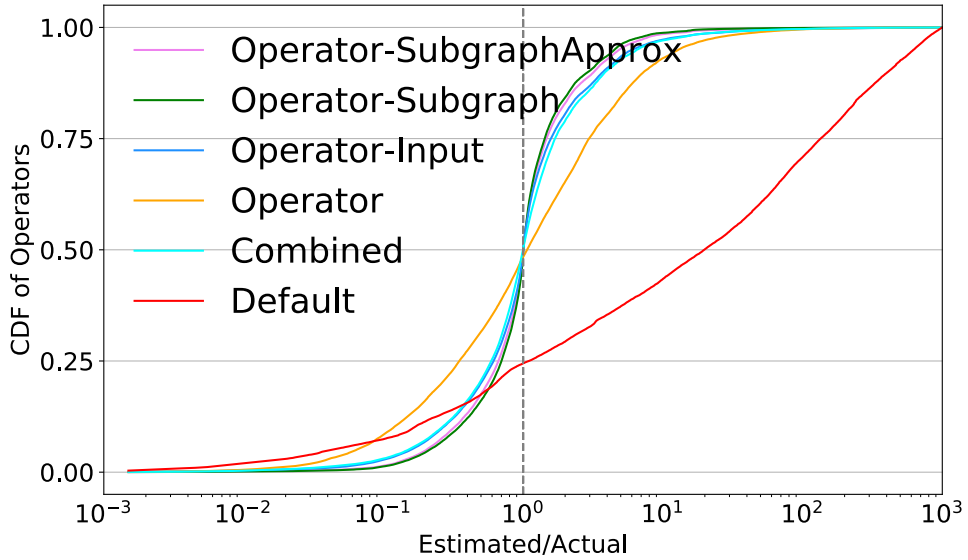
Ensemble of Models over Recurring Patterns



SCOPE Cost Estimation

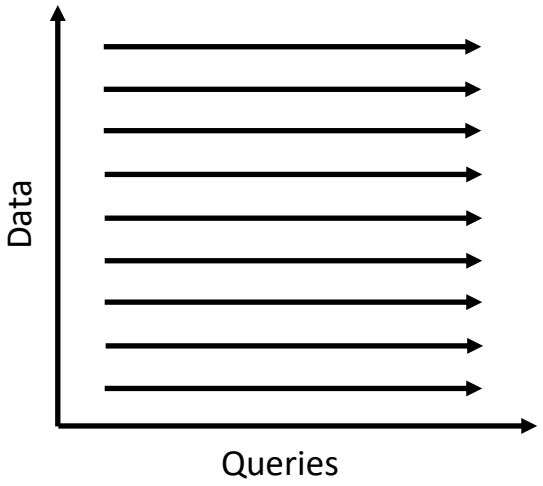
- Can learn pretty accurate cost models!





Cost Models for Big Data Query Processing: Learning, Retrofitting, and Our Findings.
Tarique Siddiqui, Alekh Jindal, Shi Qiao, Hiren Patel, Wangchao Le. *SIGMOD 2020 (to appear)*.

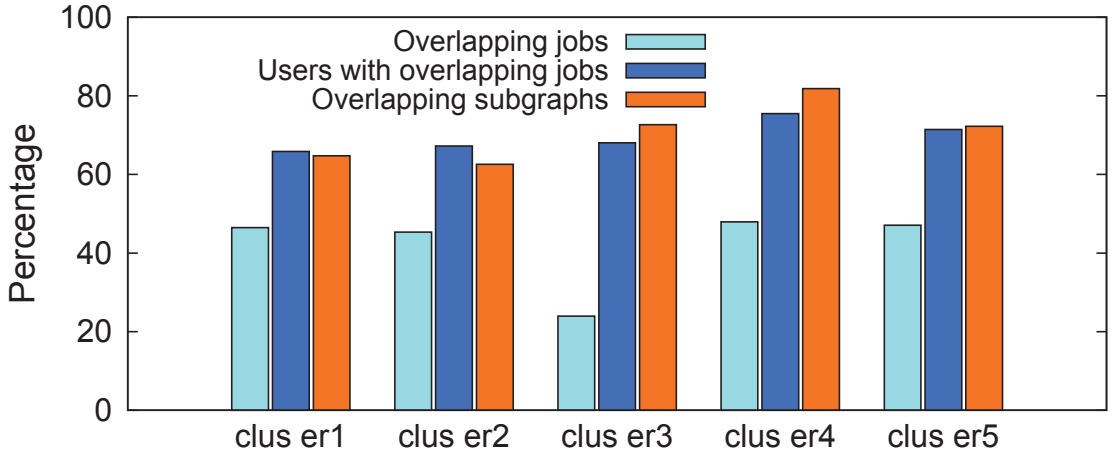
Similarity pattern



Similarity

Queries over same datasets have similarities

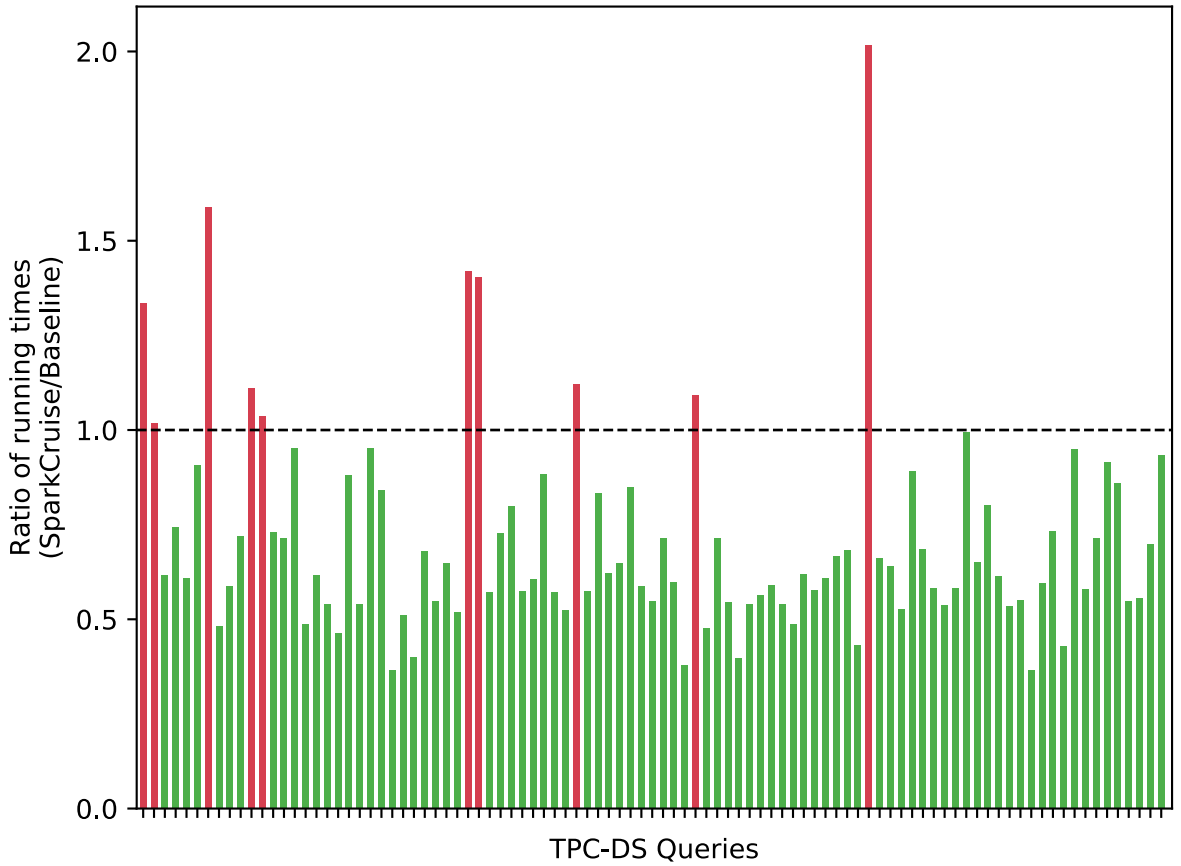
- Very typical in multi-user shared cloud environments
  - Cosmos, HDI, Ant Financial, ML workflows, etc.
- Opportunity for multi-query optimization
- SCOPE compute reuse



Computation Reuse in Analytics Job Service at Microsoft.
Alekh Jindal, Shi Qiao, Hiren Patel, Jarod Yin, Jieming Di, Malay Bag, Marc Friedman, Yifung Lin, Konstantinos Karanasos, Sriram Rao. *SIGMOD 2018*.

Spark Compute Reuse

- Instrument application log
- Analyze common subexpressions over Spark SQL plans
- Optimizer rules to automatically materialize/reuse in future queries
- Almost 30% improvement in total time on TPC-DS

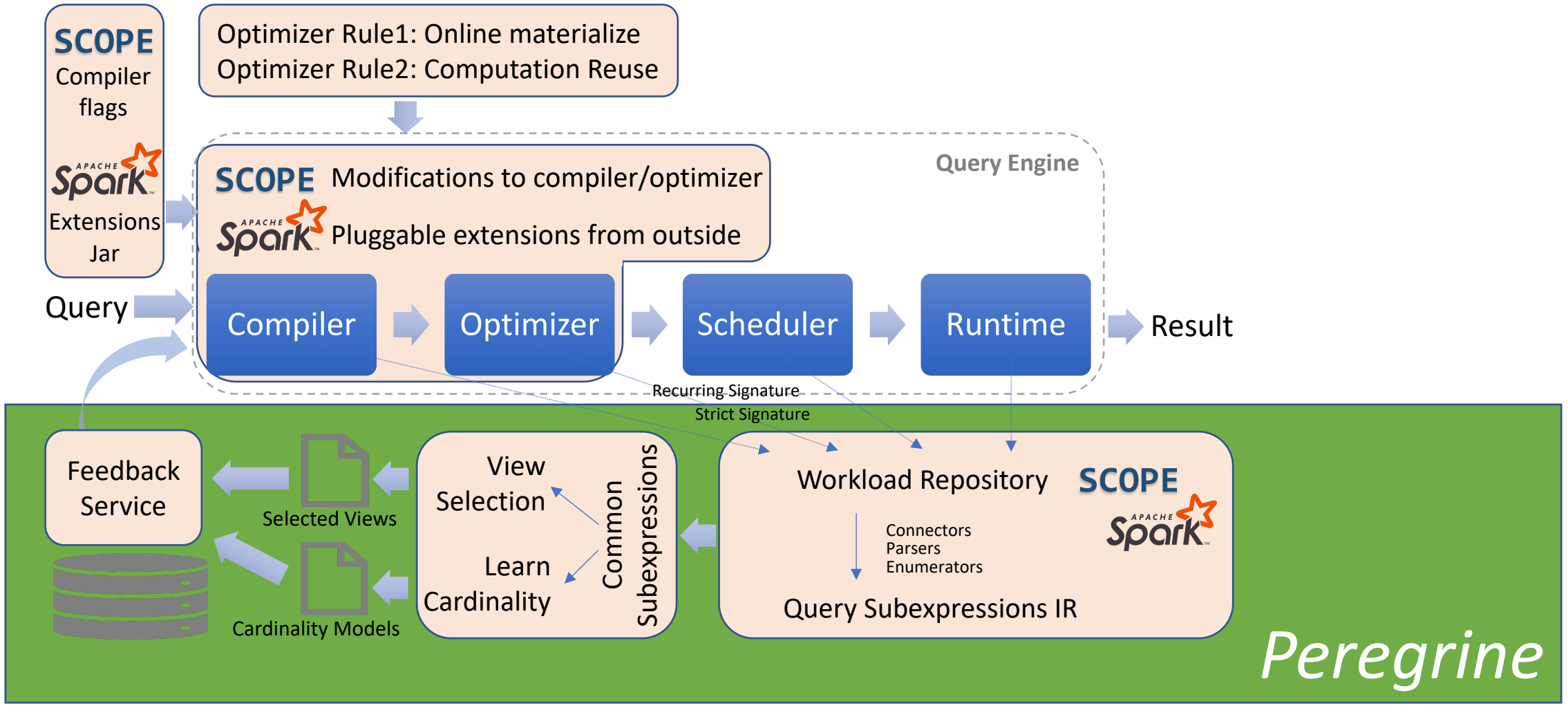


SparkCruise: Handsfree Computation Reuse in Spark.
Abhishek Roy, Alekh Jindal, Hiren Patel, Ashit Gosalia, Subru Krishnan, Carlo Curino. *VLDB 2019 (Demo)*.

Step 3: feeding it back

- Actions
  - Insights
  - Recommendations
  - Self-tuning

Illustration: Scope and Spark query engines



Peregrine Summary

- Easier to add newer features
- Easier to add newer engines
- Easier for people to participate
  - Researchers, developers, interns
  - Abstracts the painful steps
  - Build on top of each other
  - Focus on workload optimizations
- Gray Systems Lab: [aka.ms/gsl](aka.ms/gsl)