Practical Aspects of Systems that Learn

Ambition Vs Reality

Alekh Jindal, Microsoft

So, you want to build a learning system?

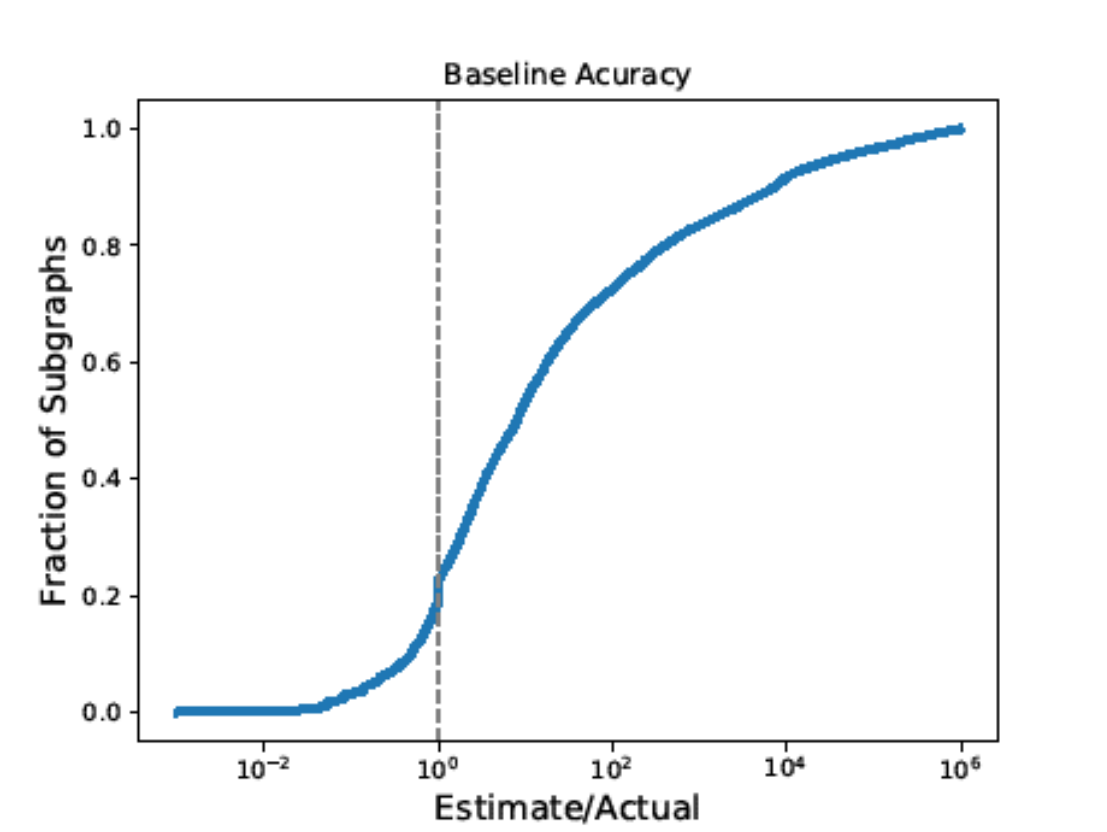
Really?

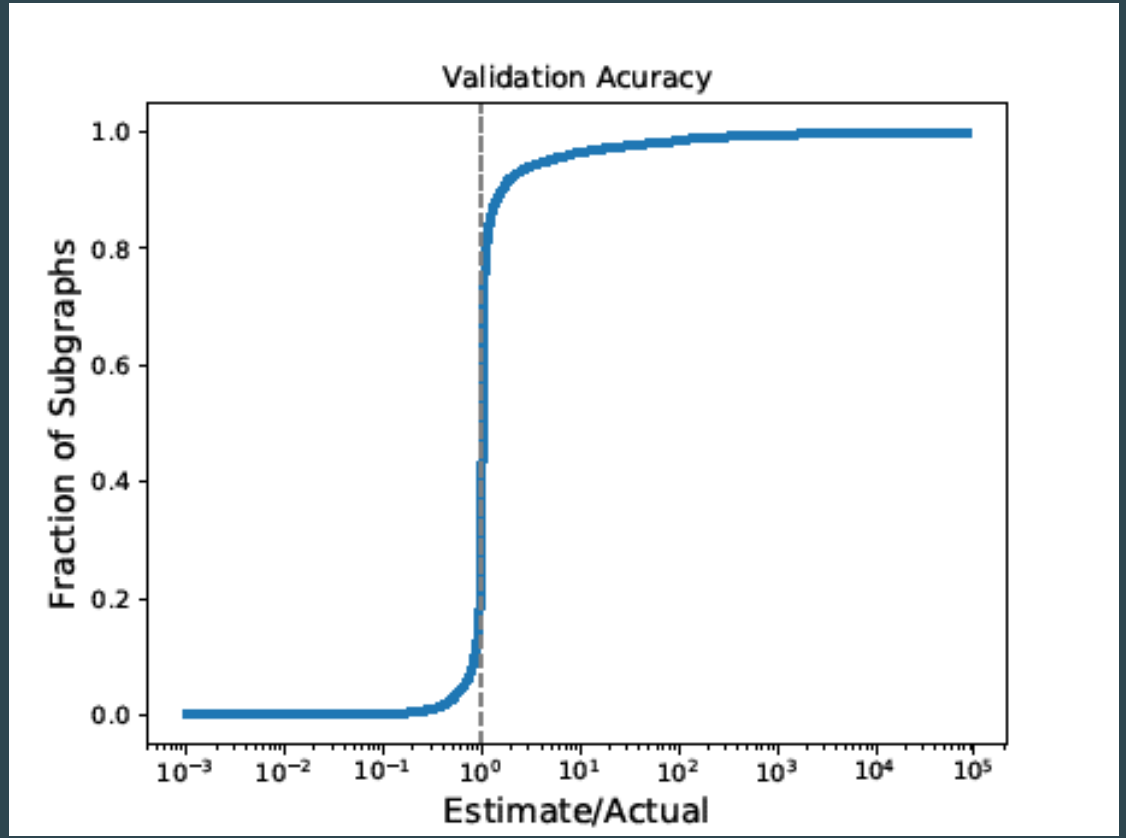
Learned Query Optimizer in Microsoft's Cosmos

2017-present: journey so far ..

- Learned Cardinality [VLDB'19]
- Learned Cost Model [SIGMOD'20]
- Learned Query Planner [SIGMOD'21]
- Learned Parallelism [VLDB'20]
- Learned Checkpointing [VLDB'21]
- Learned MQO

Learned Cardinality in Cosmos

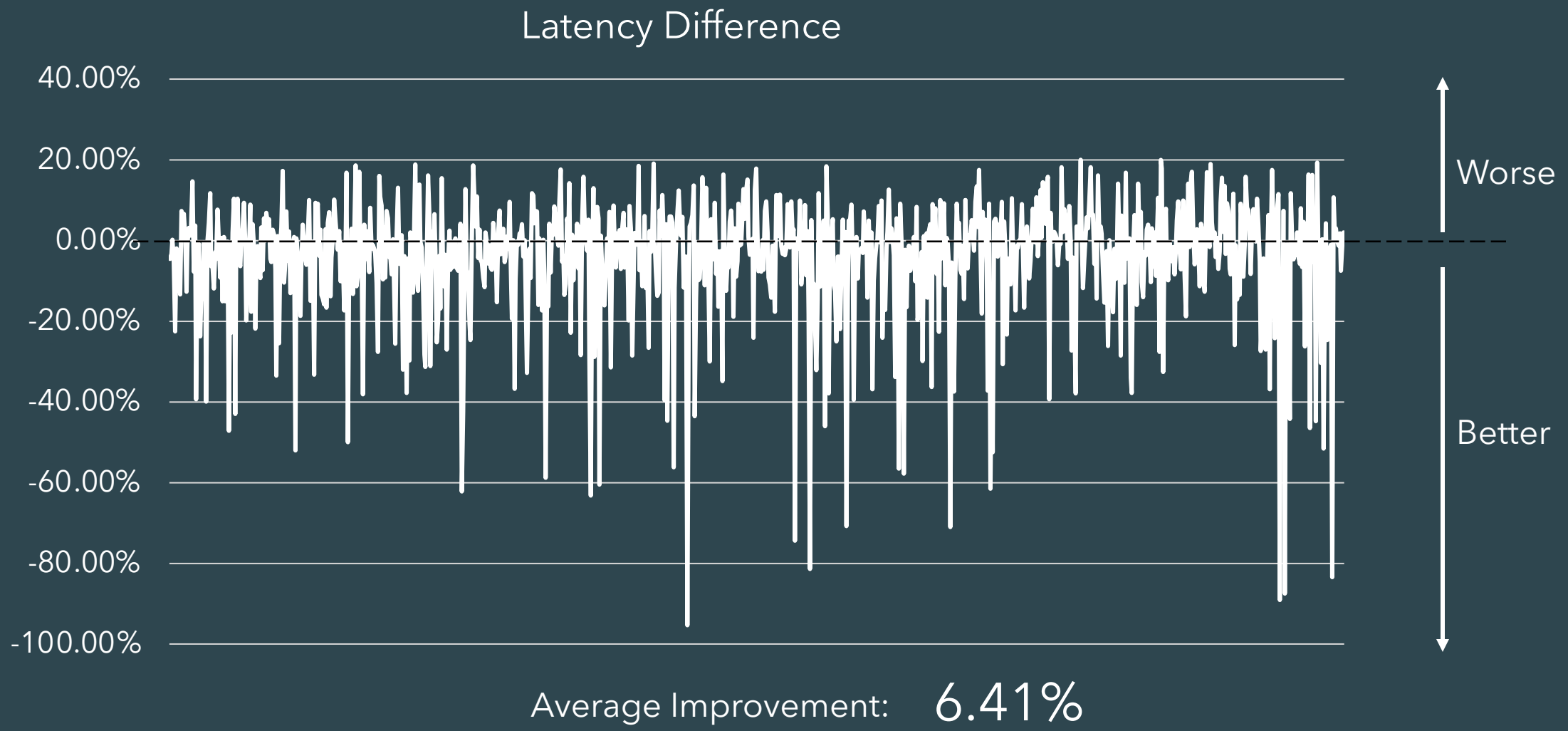




95th Percentile Error: 465711%

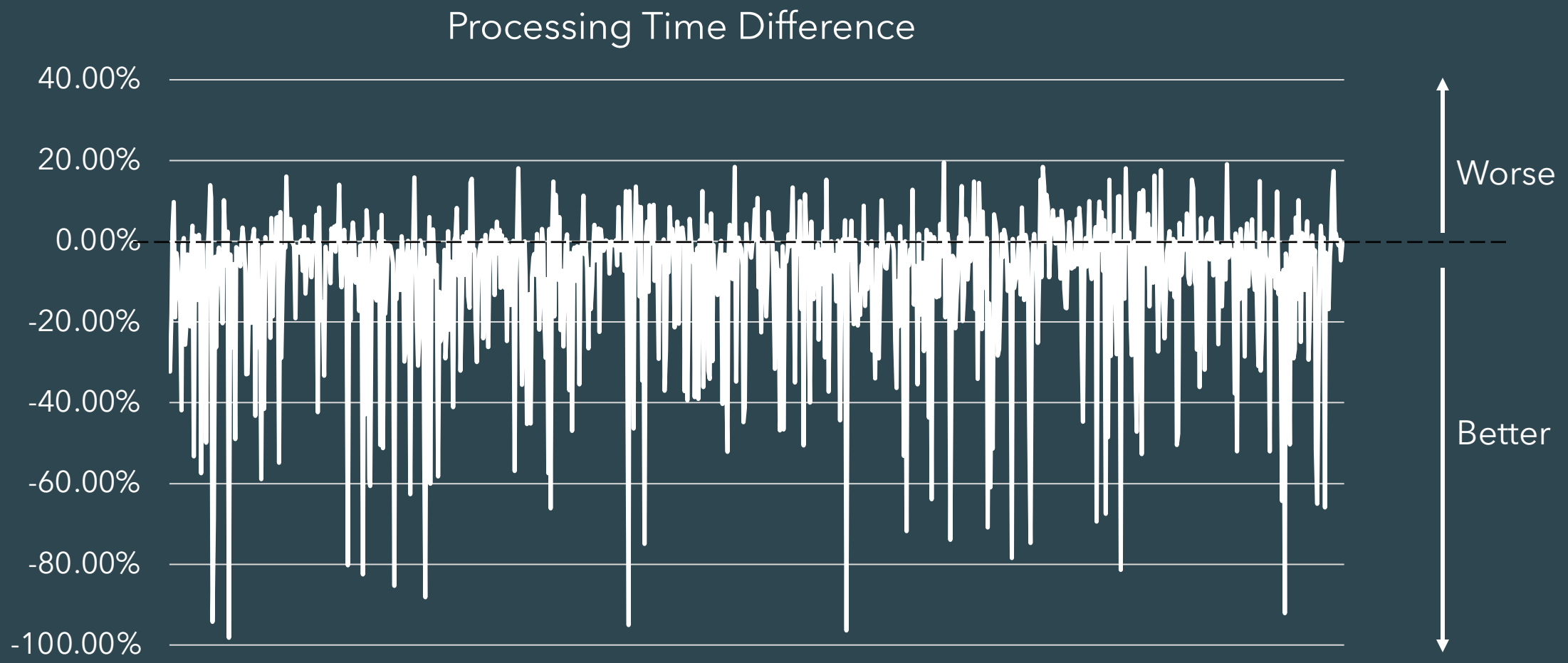
95th Percentile Error: 1%

Learned Cardinality in Cosmos



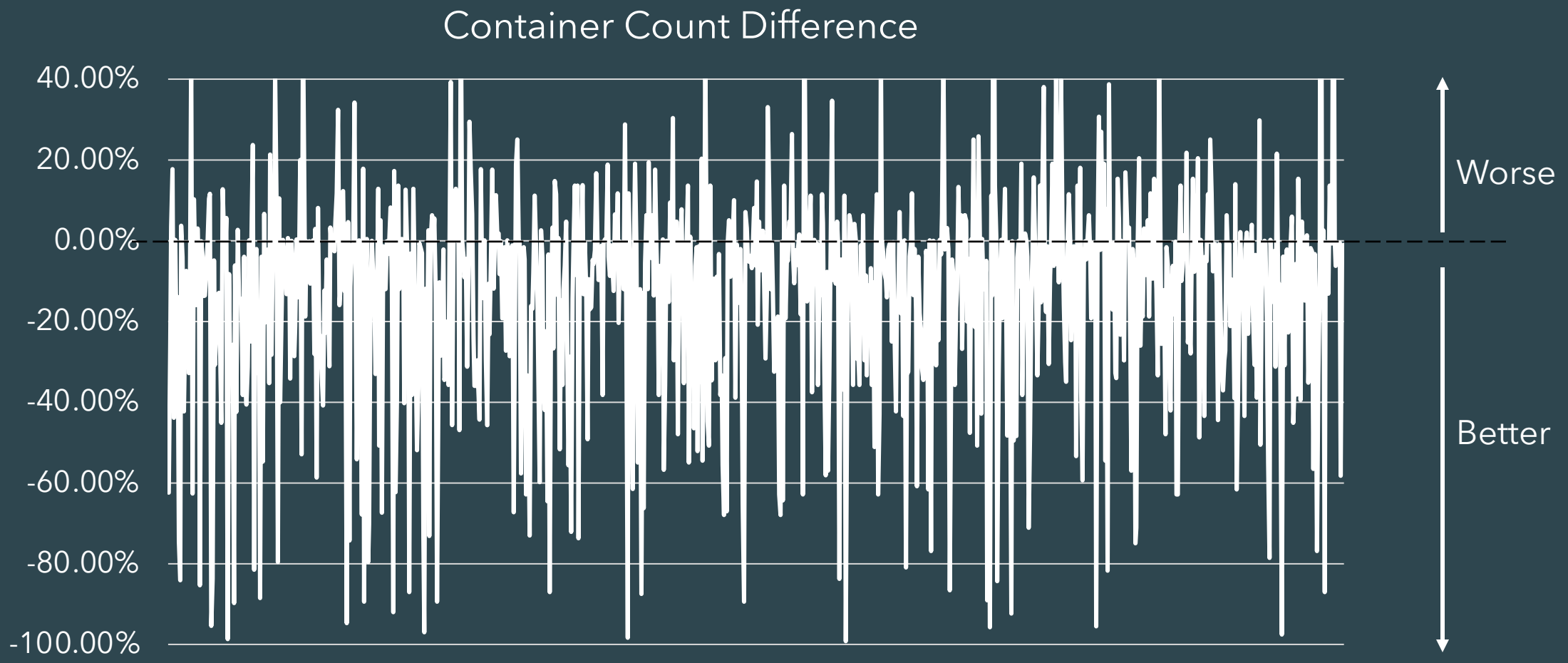
Average Improvement: 6.41%

Learned Cardinality in Cosmos



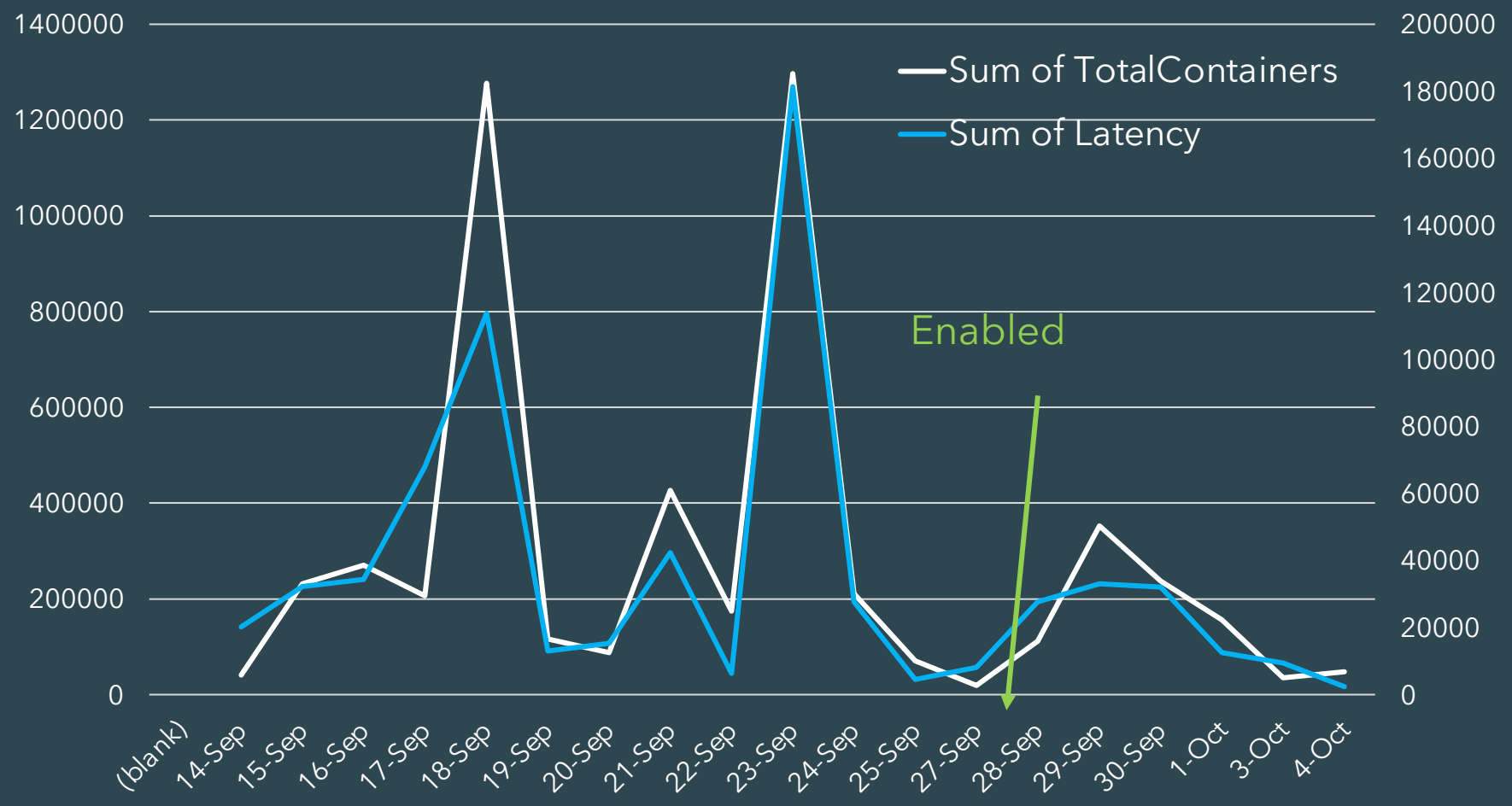
Average Improvement: 6.90%

Learned Cardinality in Cosmos



Average Improvement: 8.29%

Deployment!



2017:
Spring: problem
Summer: intern
Fall: code cleanup

2018:
System integration
E2E testing
Paper writing

2019:
Paper@VLDB'19
Plan comparison
Perf regressions

2020:
Release flighting
Deployment

2021:
industry@ICDE'21

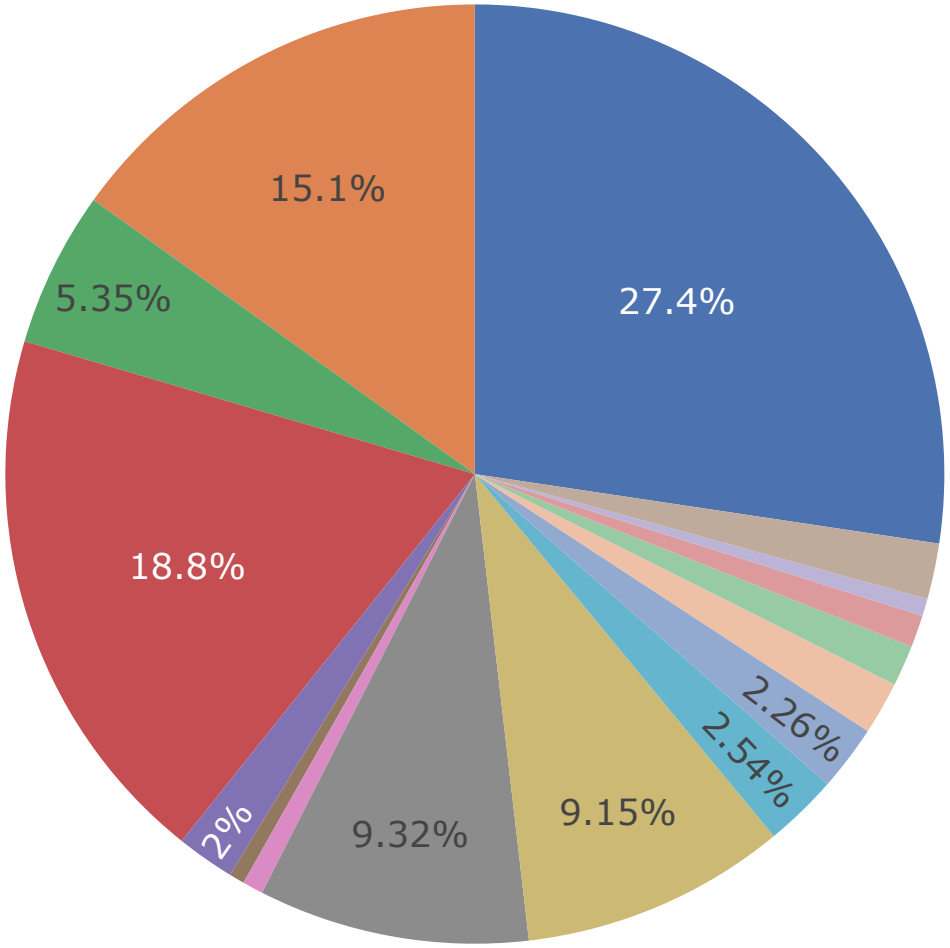
7 Practical Aspects

1. Telemetry Collection

- Operator-level telemetry
- Customer privacy
- Global telemetry
- Value of production workloads

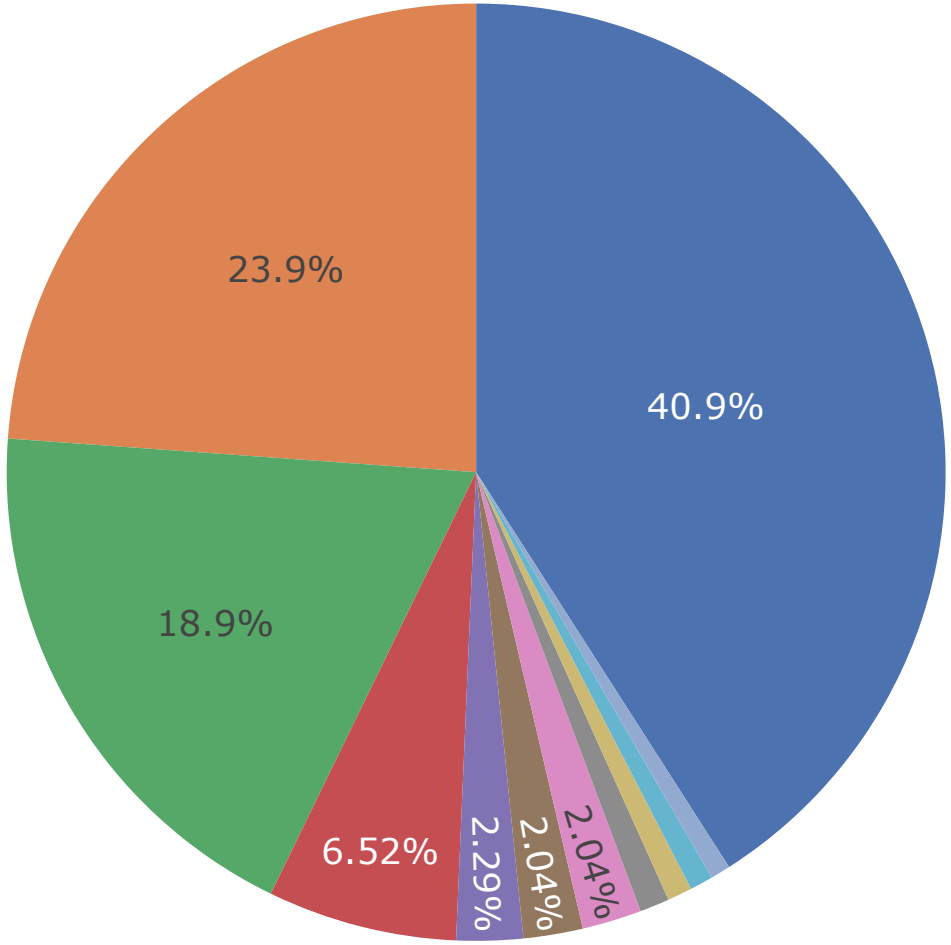
Synapse Spark Operator Distribution

Production



Project
Filter
Join
Aggregate
Sort
Union
Window
LocalLimit
GlobalLimit
SerializeFromObject
DeserializeToObject
MapElements
WriteToDataSourceV2
InsertIntoHadoopFsRel
Repartition
Other

TPC-DS

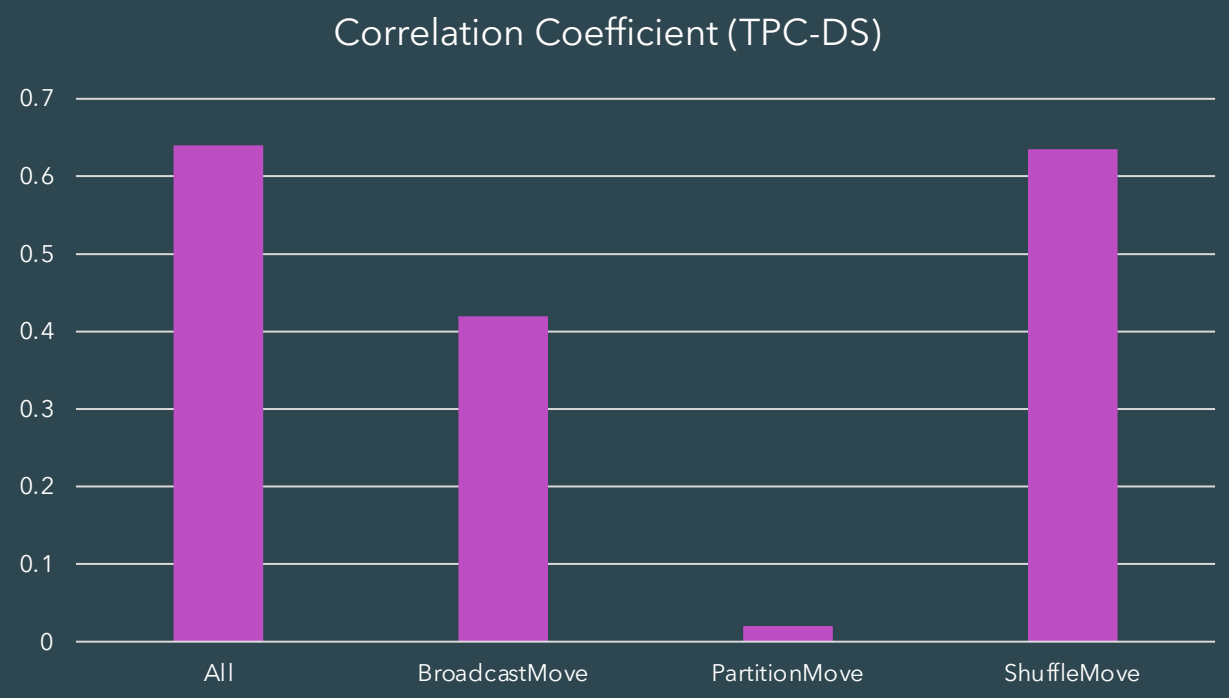


* *SparkCruise: Workload Optimization in Managed Spark Clusters at Microsoft*. Abhishek Roy et. al. VLDB 2021.

Synapse SQL Cardinality Correlation

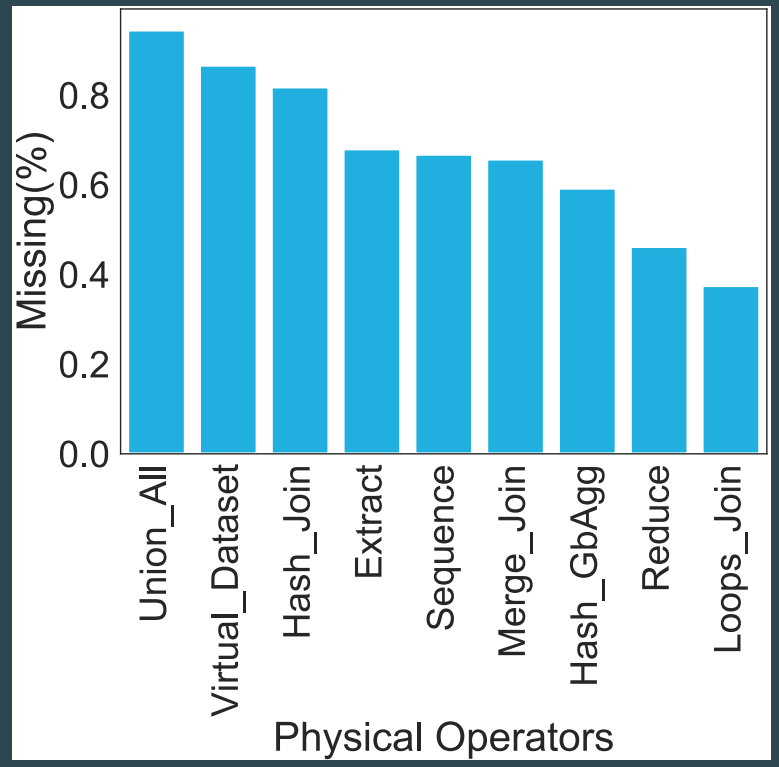
Production

TPC-DS



2. Training Data

- Tabular data
- Plan Encoding
- Connecting logical/physical
- Handling missing values

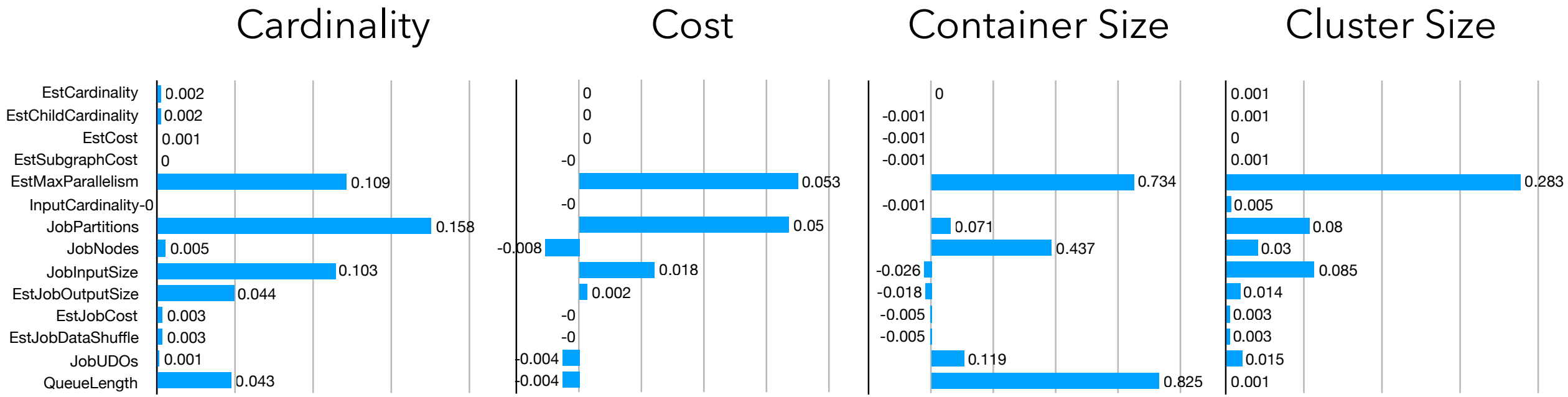


Missing Operator Statistics in Cosmos

3. Model Training

- Traditional vs deep learning
- Scalable training
- Accuracy vs Coverage
- Handling heterogeneity

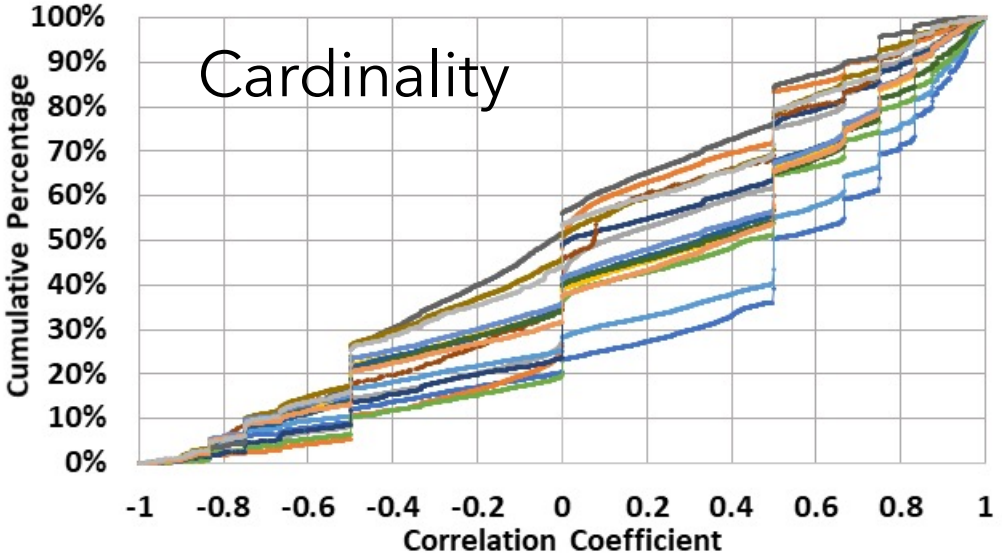
Global single-attribute correlations

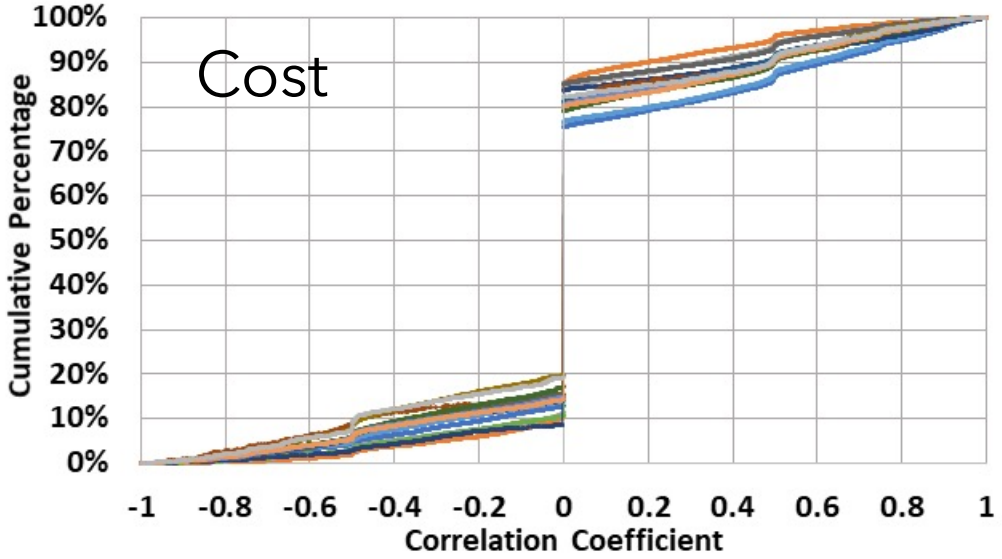


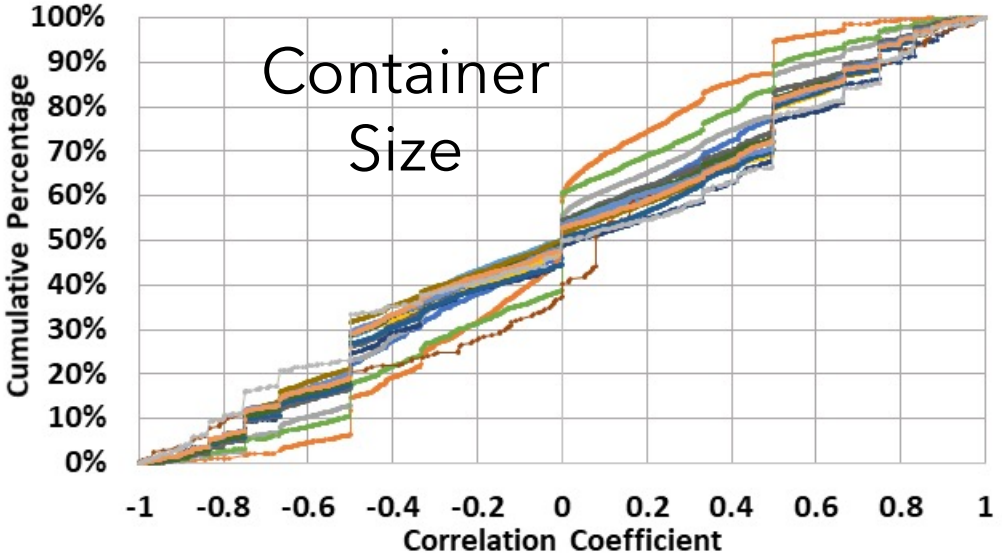
- Workload diversity, complexity, evolution!

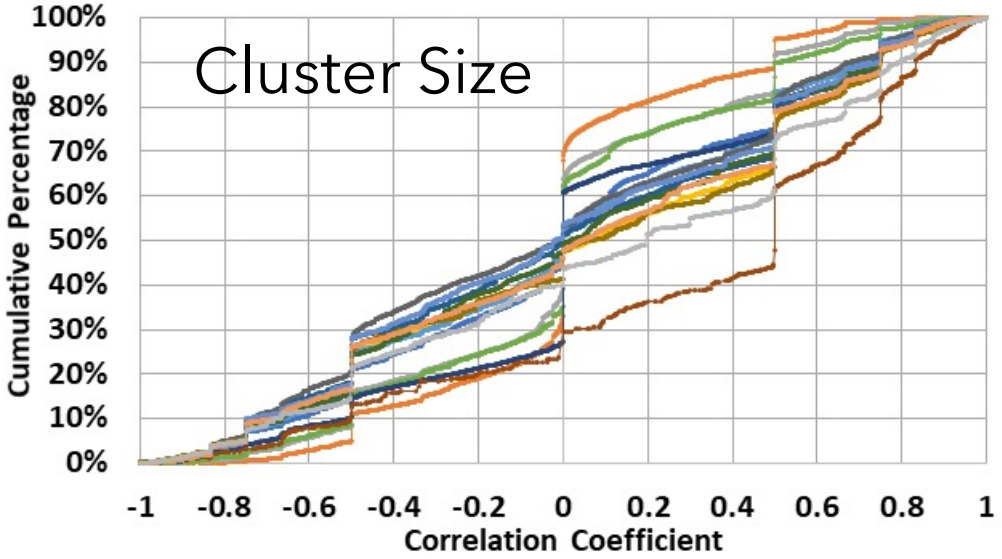
…. Oops!

Fine-grained learning









4. Feedback Loop

- External vs in-process scoring
- Gauging the impact of feedback
- Lightweight inference
- Replacing code with model predictions

Replacing code with model predictions

Use learned cardinality wherever available

```
HashmapLearnedModels learnedModels = pdhint->LearnedModels();
HashmapPwszFeatureVal hmFeatureValues;
CTreeHandle trh(pgrp->PgexprFirstLogExpr());
CAutoRefc<COptExpr> a_exprInput = trh.PexprExtract(true, true);
OptimizerUtil::GetLogicalFeatures(pmo, a_exprInput, hmFeatureValues);

// set predicted cardinality.
try
    {
    CLearnedModel *cardLearned = learnedModels[x_learnedCard];
    if (cardLearned && !cardLearned->FEmpty())
        {
        CARD cardPredicted = cardLearned->Predict(hmFeatureValues);
        pdhint->SetCardExpected(cardPredicted);
        QP_OPTCONTEXT->TraceCounter(CQO_COUNTER_PREDICT_CARD);
        QP_OPTCONTEXT->SetLearnedResourceApplied(x_learnedCard);
        }
    }
catch (...)
    {
    QP_OPTCONTEXT->LogDiagnostic(CQO_WARNING_FAILED_TO_PREDICT_CARD);
    QP_OPTCONTEXT->TraceCounter(CQO_COUNTER_FAILED_TO_PREDICT_CARD);
    }
```

Custom predict function

```
//---------------------------------------------------------------------
// CardModel::CardPredict
//  Return the predicted value of cardinality
//
CARD CLearnedModel::Predict(HashmapPwszFeatureVal &featureValues) const
    {
    CARD predictedValue = 0;
    for (auto &featureWeight : m_hmFeatureWeights)
        {
        std::unordered_map<std::wstring, FeatureVal>::iterator it = featureValues.find(featureW
        if (it != featureValues.end())
            {
            predictedValue += it->second * featureWeight.second;
            }
        }

    switch (m_modelType)
        {
        case x_linearModel:
            Assert(HUGE_VAL != predictedValue && HUGE_VAL != -predictedValue);  // ensure there
            break;

        case x_poissonModel:
            // For poisson model, return natural exponential function of the predictedValue.
            predictedValue = exp(predictedValue);
            Assert(HUGE_VAL != predictedValue && HUGE_VAL != -predictedValue);  // ensure there
            break;

        default:
            AssertSz(fFalse, "Unknown learned model type.");
            break;
        }

    return predictedValue;
    }
```

5. Experimentation

- Does feedback lead to better Performance?
- Is the plan change good?
- Discarding noise
- Replaying production workloads

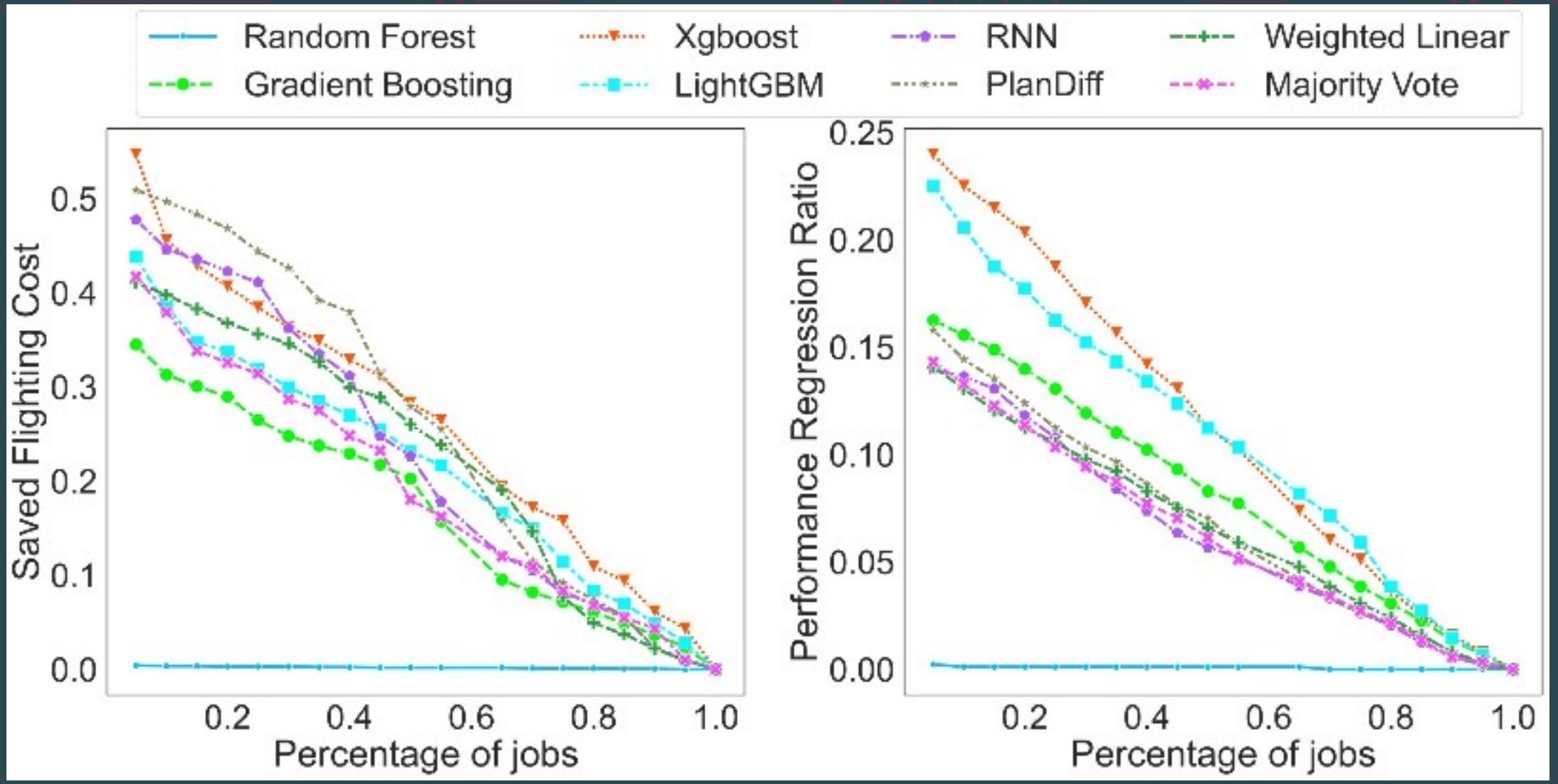
6. Deployment

- Opt-in:
  - Per-query
  - Per-account
- Opt-out
  - Per-cluster
  - Per-region

7. Performance Regression

- Causes: inaccurate models, component interactions
- Understanding and explaning
- Mitigation: disable model/query
- Avoiding regressions
  - More diversified training data
  - Online learning (customer expectations)
  - Fixing other components
  - Predicting the impact of plan changes

PerfGuard



PerfGuard: Deploying ML-for-Systems without Performance Regressions, Almost! . Remmelt Ammerlaan et. al. VLDB 2022 (to appear).

Open Questions

- How to cope the old school with the new one?
- Is the rich getting richer?
- What are we really learning?
- How do we address learning bias?
- What is cost of learning?
- ML vs Heuristics?

Thanks!