POLAR: Adaptive and Non-invasive Join Order Selection via Plans of Least Resistance

David Justen[1,2], Daniel Ritter[3], Campbell Fraser[4], Andrew Lamb[5], Nga Tran[5], Allison Lee[6], Thomas Bodner[7,8], Mhd Yamen Haddad[9,10], Steffen Zeuch[1,2], Volker Markl[1,2], Matthias Boehm[1,2]

Corresponding author's email: david.justen@tu-berlin.de

Motivation

Wrong cardinality estimates lead to **suboptimal join orders**

Adaptive Query Processing (AQP) measures cardinalities to reorganize joins during execution

Despite decades of AQP research, **low adoption in practice**

- System and integration **complexity**
- Large **performance overheads**

Design Objectives

Non-invasive Integration, Overhead Mitigation

Separation of compilation and execution phase

Reuse host system optimizer and operators

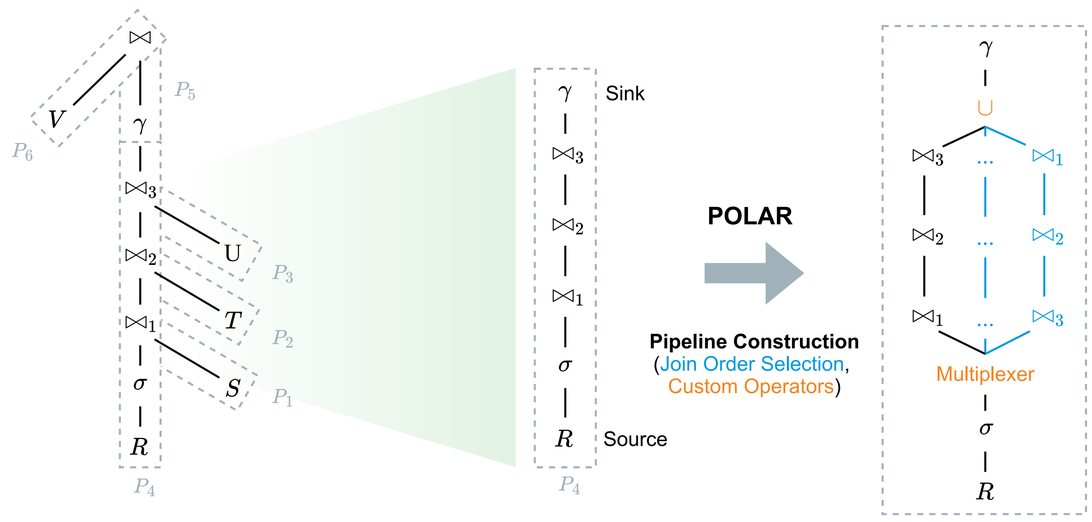
Allow **fallback** on original plan

Bound exploration overhead

System Overview

```
SELECT ...
  FROM R, S, T, U, V
  WHERE ...
  GROUP BY ...
```

Unmodified Optimizer



Enhance optimized query pipelines with **additional join orders**

Extensible **multiplexer** operator with several **routing strategies** and probabilistic **regret bound**

Open-source **DuckDB** prototype

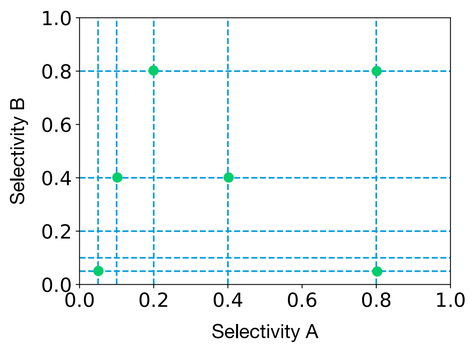
Extension in development

Join Order Selection

Form **d-dimensional grid** from base table predicate and join selectivities

Discretize selectivities with **exponential decay**

Sample grid uniformly and invoke DPsize with sampled points

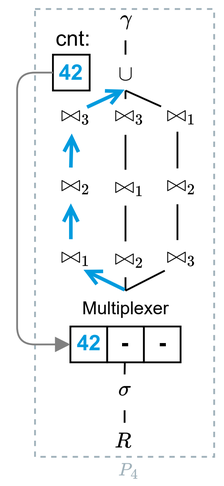


Adaptive Pipeline Execution

Route bags of tuples consecutively through pre-selected join orders

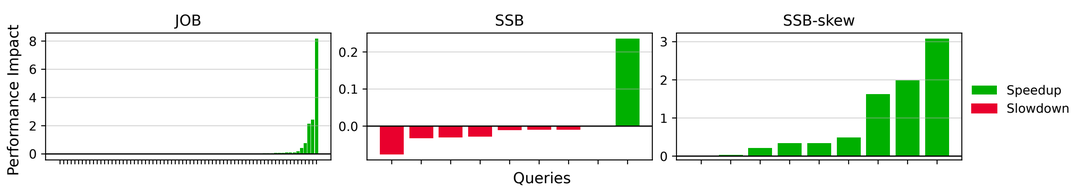
Use **path resistance** (i.e., intermediate results per input tuple) for routing decisions

Probabilistic Regret Bound
Trade-off between *exploiting* well-performing join orders and *exploring* weaker paths based on past observations



Experiments

Individual Performance Impact

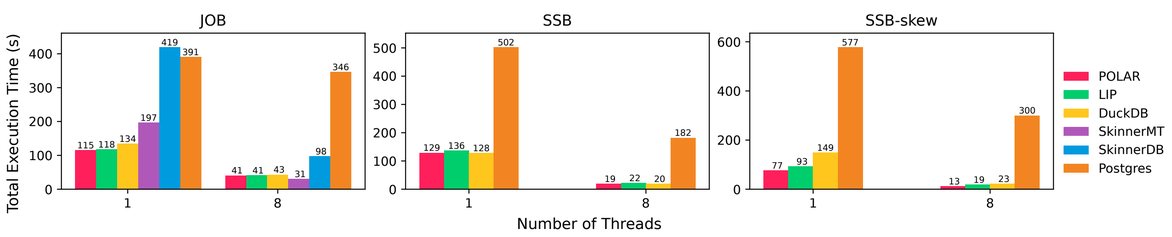


JOB: occasional performance improvements up to 9x

SSB: small impact and moderate overhead up to 7%

SSB-skew: improvement in almost every query

Total Execution Times



POLAR: total execution time consistently ≤ DuckDB

Large benefits from reusing DuckDB components

1 Technische Universität Berlin 2 BIFOLD 3 SAP 4 Google 5 influxdata® 6 snowflake 7 HPI 8 Universität Potsdam 9 Inria 10 École Polytechnique Université Paris-Saclay