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]

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

50th International Conference on Very Large Databases | Guangzhou, China | August 26-30, 2024

Cardinality Estimation Problem

Join Ordering Problem

- Join orders are crucial for analytical query performance
- Traditional cardinality estimation is stubbornly difficult

Adaptive Query Processing (AQP)

- Measure performance indicators (e.g., cardinalities) mid-flight
- Continuously adapt query plan during execution

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

- AQP system complexity (compile/execution phase intertwining)
- Potentially large performance overheads

System Design Objectives

Separation of compilation and execution phase

Reuse host system optimizer and operators

Allow **fallback** on original plan

Bound exploration overhead

Non-invasive integration

Overhead mitigation

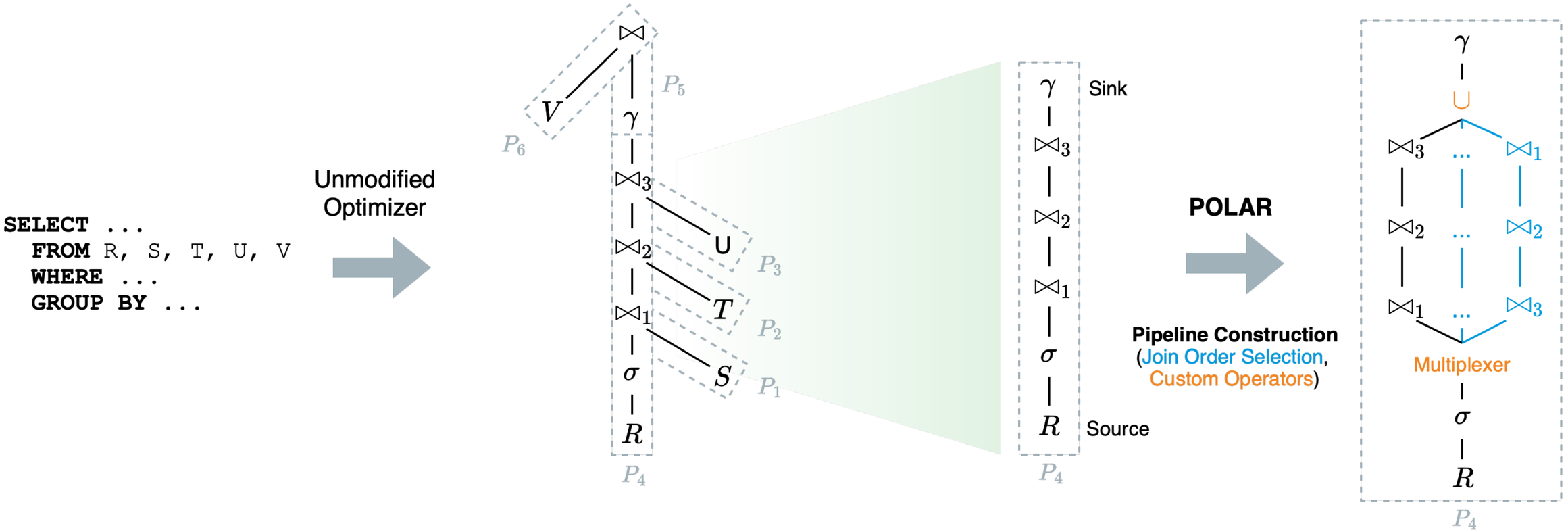
Contributions

POLAR, an intra-pipeline AQP technique enhancing join pipelines with alternative join plans to find and exploit the Plan of Least Resistance.

- Non-invasive pipeline design
- Extensible multiplexer operator with several routing strategies and probabilistic bounded regret
- SSB-skew: AQP system benchmark with a star-schema and skewed data

Open-source prototype, DuckDB extension in development ⚙️

System Overview



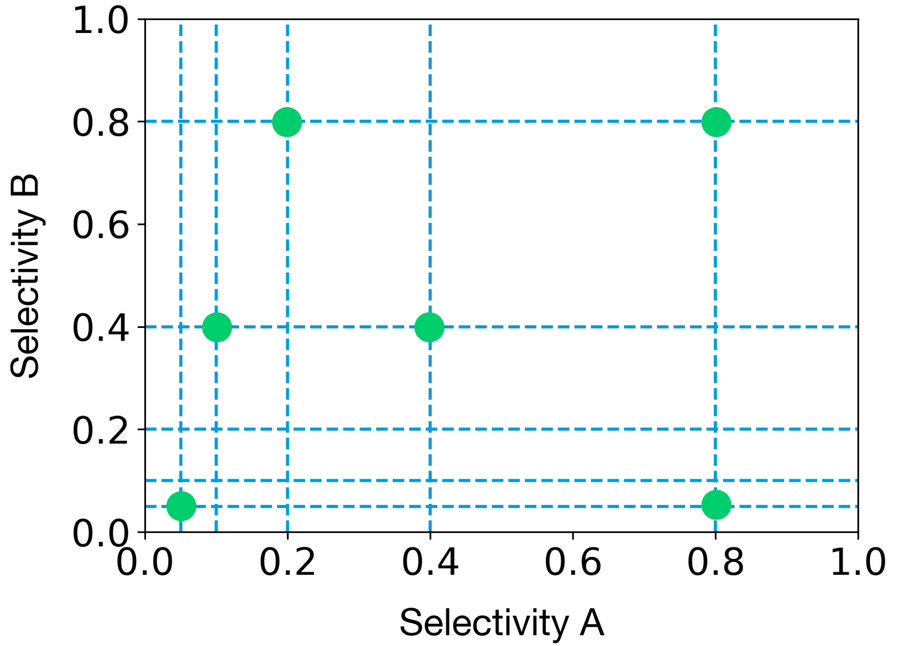
Join Order Selection

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

Stop after max iterations or max join orders generated



Adaptive Pipeline Execution

Multiplexer
Route tuple vectors consecutively through join orders

POLAR Pipeline Executor
Measure *path resistance* = intermediate results per input tuple

Use resistance to make routing decisions in multiplexer

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

Given regret budget *b* and resistances *R*, find path weights *P* so that:

$$\sum p_i * r_i \leq \min(R) * (1 + b)$$

Experiment Setup

Benchmarks

Join Order Benchmark: short join pipelines ➡️ 36% coverage*

Star Schema Benchmark (sf=100): long pipelines, easy to estimate ➡️ 73% coverage*

SSB-skew (sf=100): long pipelines, skewed and correlated data ➡️ 99% coverage*

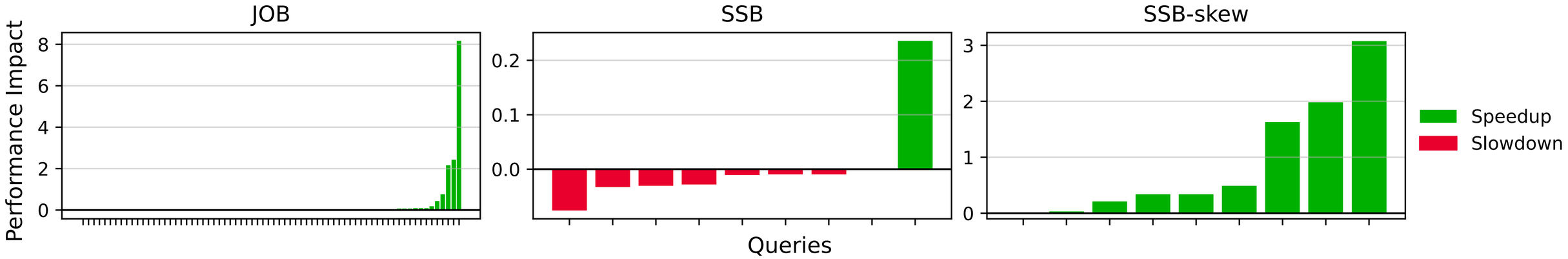
* relative amount of execution time spent in POLAR-amenable join pipelines

Comparing Systems

DuckDB-based: DuckDB, Lookahead Information Passing (LIP)

External: SkinnerDB, SkinnerMT, Postgres

Individual Query Performance Impact

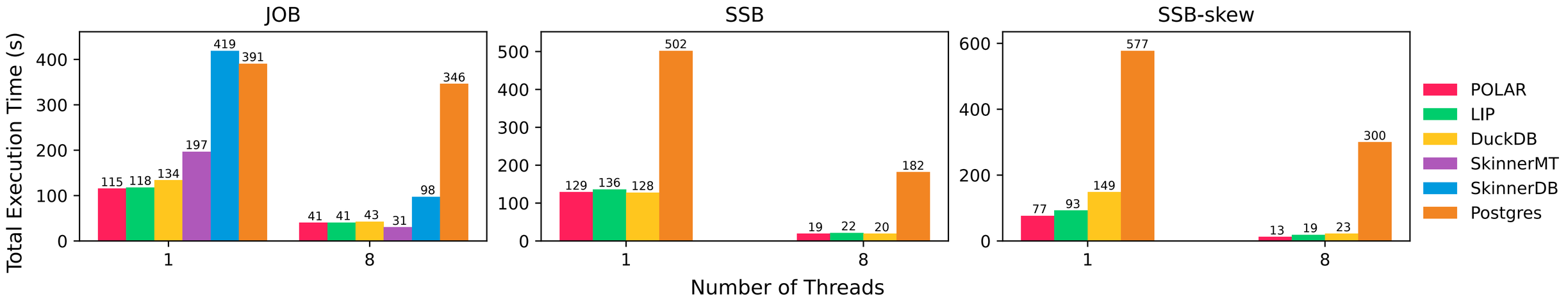


JOB ➡️ occasional performance improvements up to 9x

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

LIP ➡️ competitive performance but occasional overheads of up to 2x

SkinnerDB/MT ➡️ out-of-memory for SSB(-skew), high execution time for sf=10

Conclusion

Focus on join pipelines lowers applicability for **minimal overhead** (up to 7%)

Substantial improvements for **individual queries** (up to 9x)

Most applicable to **skewed, correlated data in star schema**

Stay tuned for the **DuckDB extension** 🦆

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

David Justen, Daniel Ritter, Campbell Fraser, Andrew Lamb, Nga Tran, Allison Lee, Thomas Bodner, Mhd Yamen Haddad, Steffen Zeuch, Volker Markl, Matthias Boehm

Link to paper

Contact

Email: david.justen@tu-berlin.de

Web: d-justen.github.io