

PVLDB 2024

POLAR: Stop Guessing, Start Adapting

David Justen^{1,2}, Daniel Ritter³, Campbell Fraser⁴, Andrew Lamb⁵, Nga Tran⁵, Allison Lee⁶, Thomas Bodner^{7,8}, Mhd Yamen Haddad^{9,10}, Steffen Zeuch^{1,2}, Volker Markl^{1,2}, Matthias Boehm^{1,2}



> whoami



David Justen

Since 2022
Everywhere
DuckDB Enthusiast

More info:

 d-justen.github.io

 david.justen@tu-berlin.de

Cardinality Estimation Problem

Join Ordering Problem

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

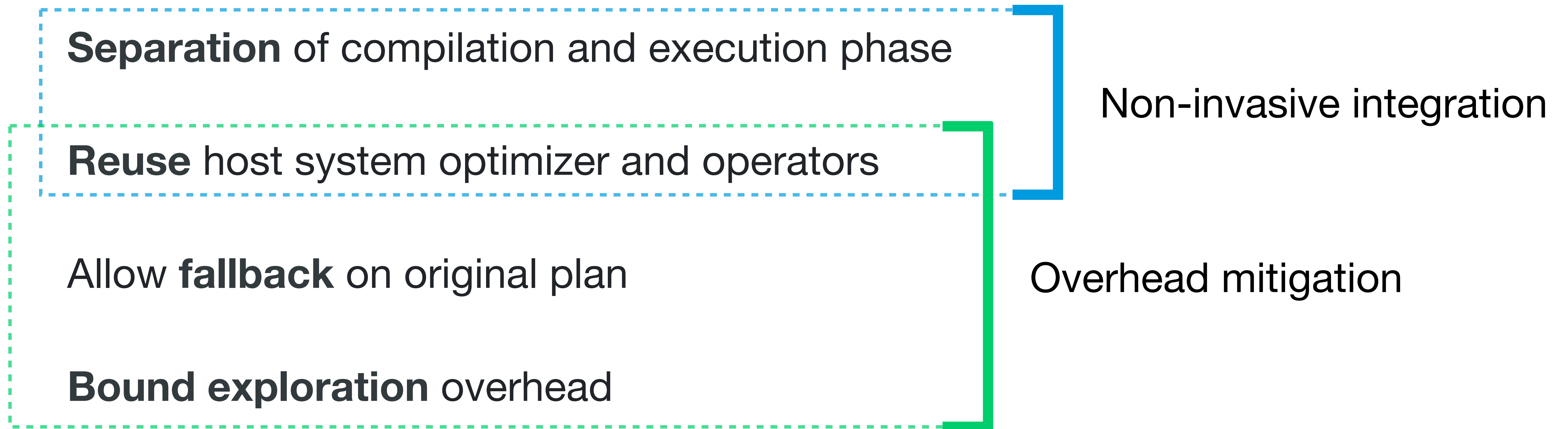
Adaptive Query Processing (AQP)

- 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



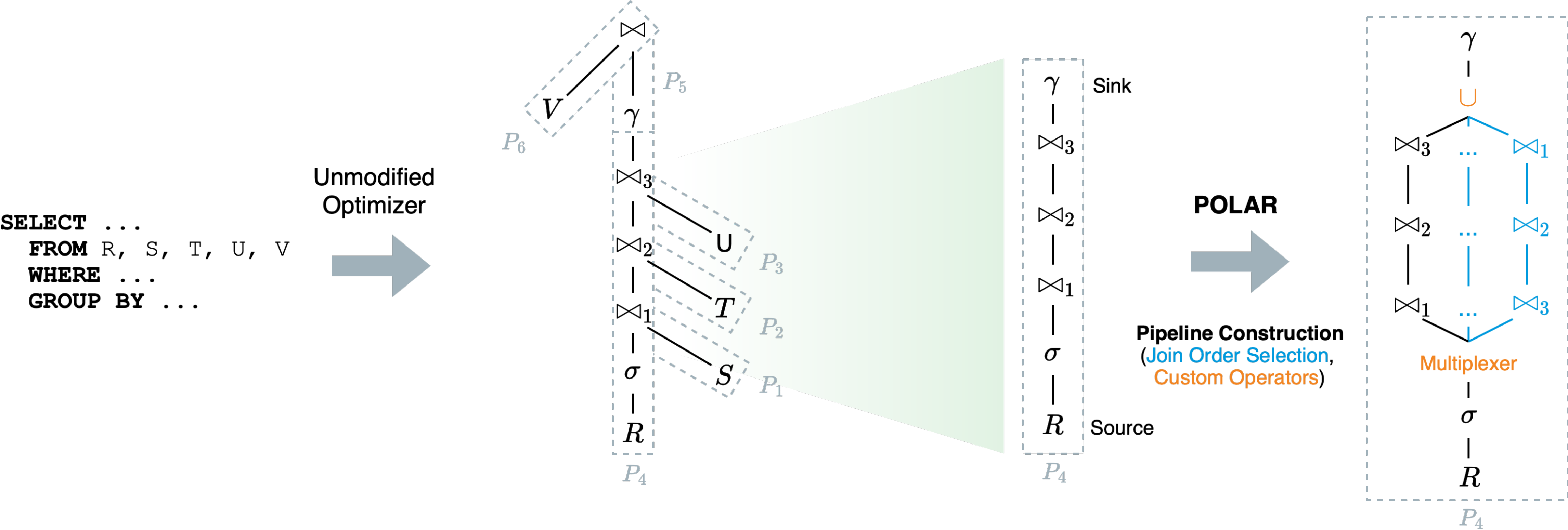
Contributions

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



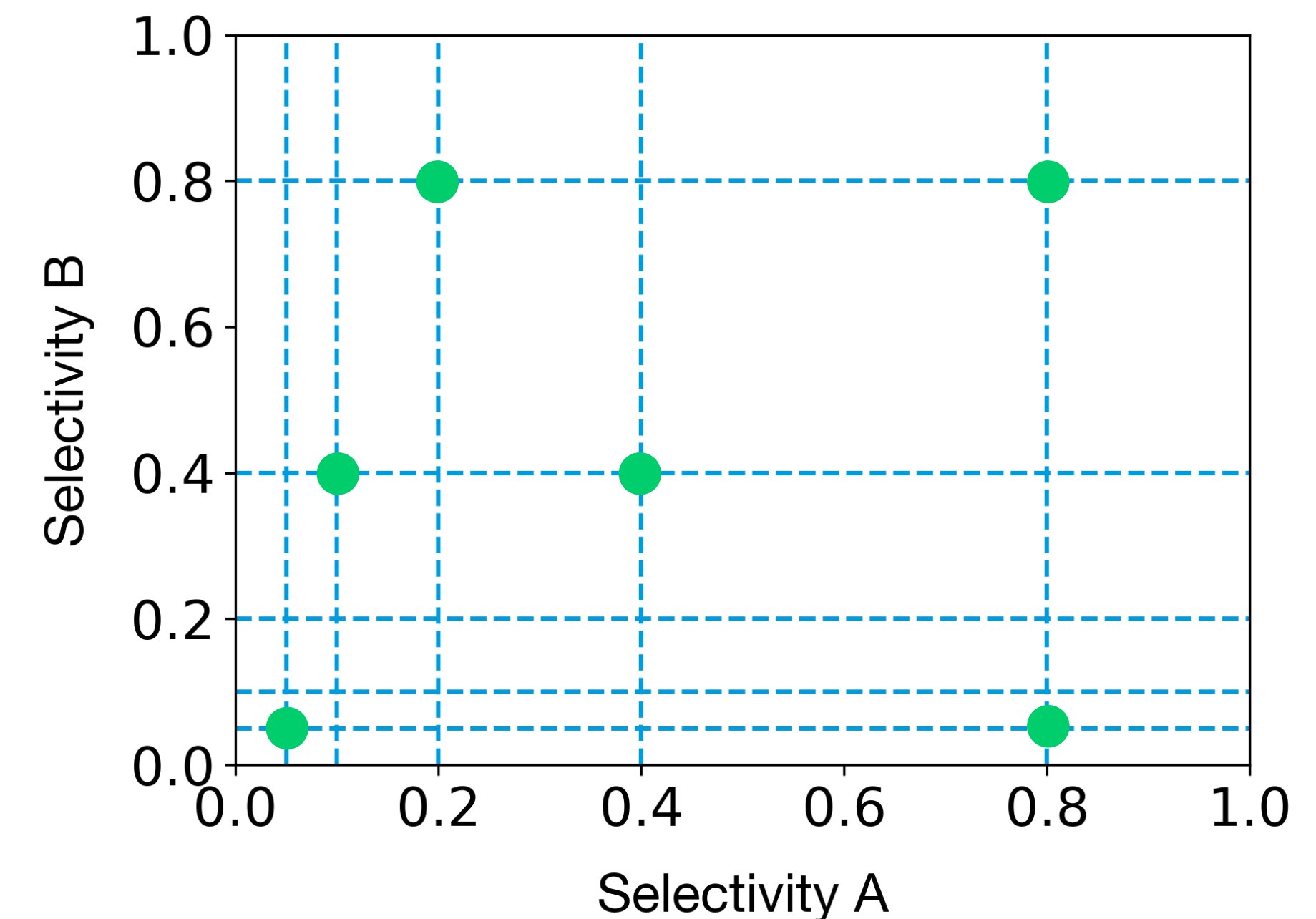
Join Order Selection

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:

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

POLAR as a DuckDB Extension

Optimizer Rule

Identify join pipelines

Replace joins with POLAR operator taking ownership of the joins

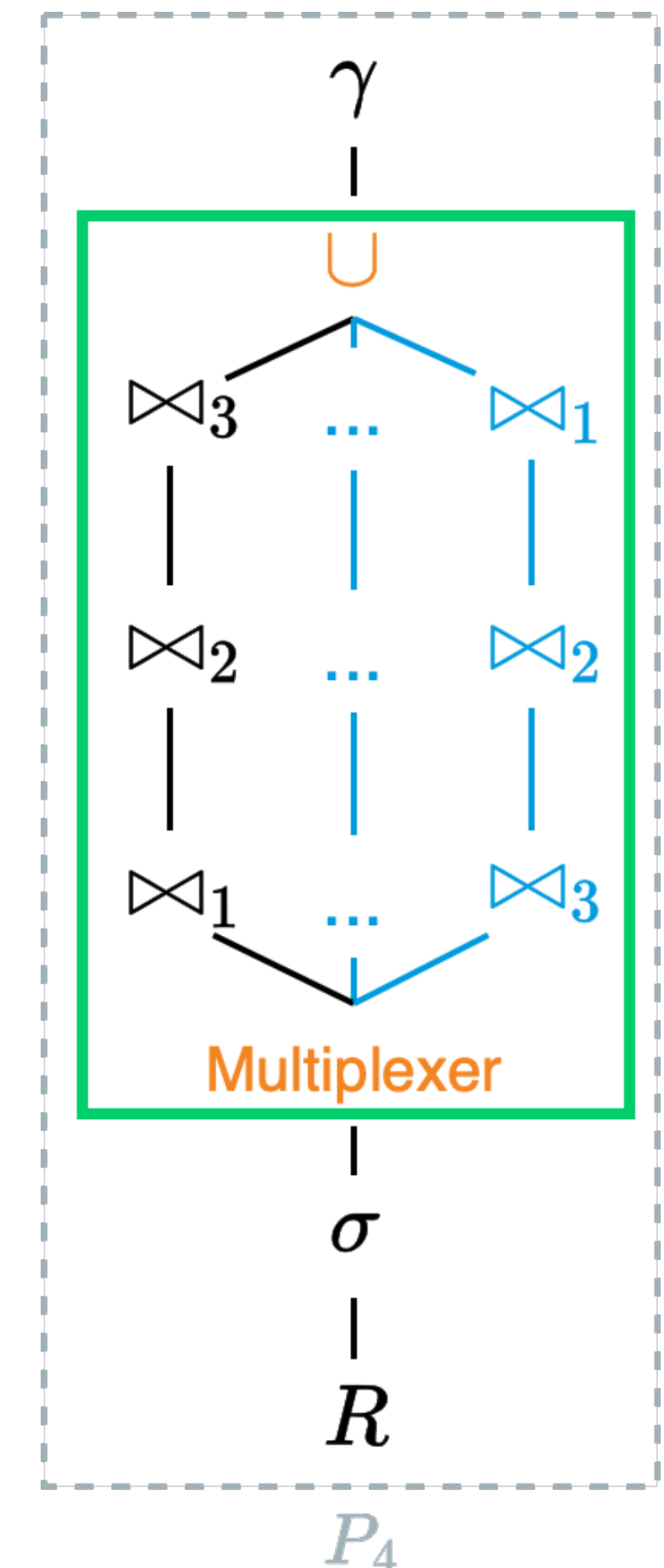
Enhance operator with additional join orders

Operator Extension

Internally contains multiplexer, joins, and adaptive union

Works like a pipeline, but is not a pipeline breaker

Encapsulated
POLAR
Operator



Applicability Study

Benchmarks

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

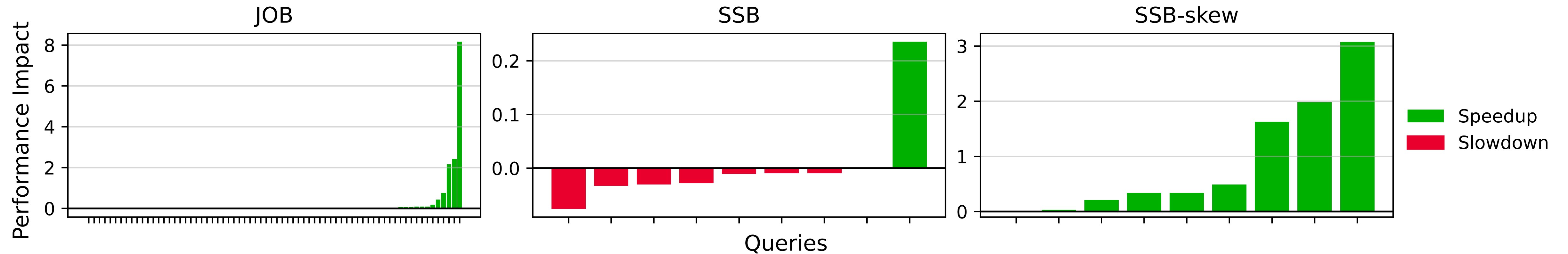
Star Schema Benchmark (sf=100): long pipelines, easy to estimate ➡ 73% coverage*

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

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

Experiments run on DuckDB v1.0

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

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



Contact

 d-justen.github.io

 david.justen@tu-berlin.de