Interactive BI Analytics with Presto

Big Data Conference Europe 2020

Karol Sobczak, Software Engineer, Starburst
Łukasz Osipiuk, Software Engineer, Starburst

Łukasz Osipiuk

Software Engineer, Starburst

/in/lukasz-osipiuk-781903

@losipiuk

lukasz.osipiuk@starburstdata.com

Karol Sobczak

Software Engineer, Co-founder, Starburst

/in/karol-sobczak-a7b19a10

@sopel39

karol.sobczak@starburstdata.com

Agenda

1. Introduction to Presto
2. Presto in data analysis ecosystem
3. Under the hood
4. Demo

What is Presto?

presto

High performance MPP SQL engine

- Interactive ANSI SQL queries
- Proven scalability
- High concurrency

Community-driven open source project

Separation of compute & storage

- Scale storage & compute independently
- SQL-on-anything
- Federated queries

Deploy Anywhere

- Kubernetes
- Cloud
- On premises

Presto Users

presto

Facebook: 10,000+ of nodes, 1000s of users
Uber 2,000+ nodes, 160K+ queries daily

LinkedIn: 500+ nodes, 200K+ queries daily
Lyft: 400+ nodes, 100K+ queries daily

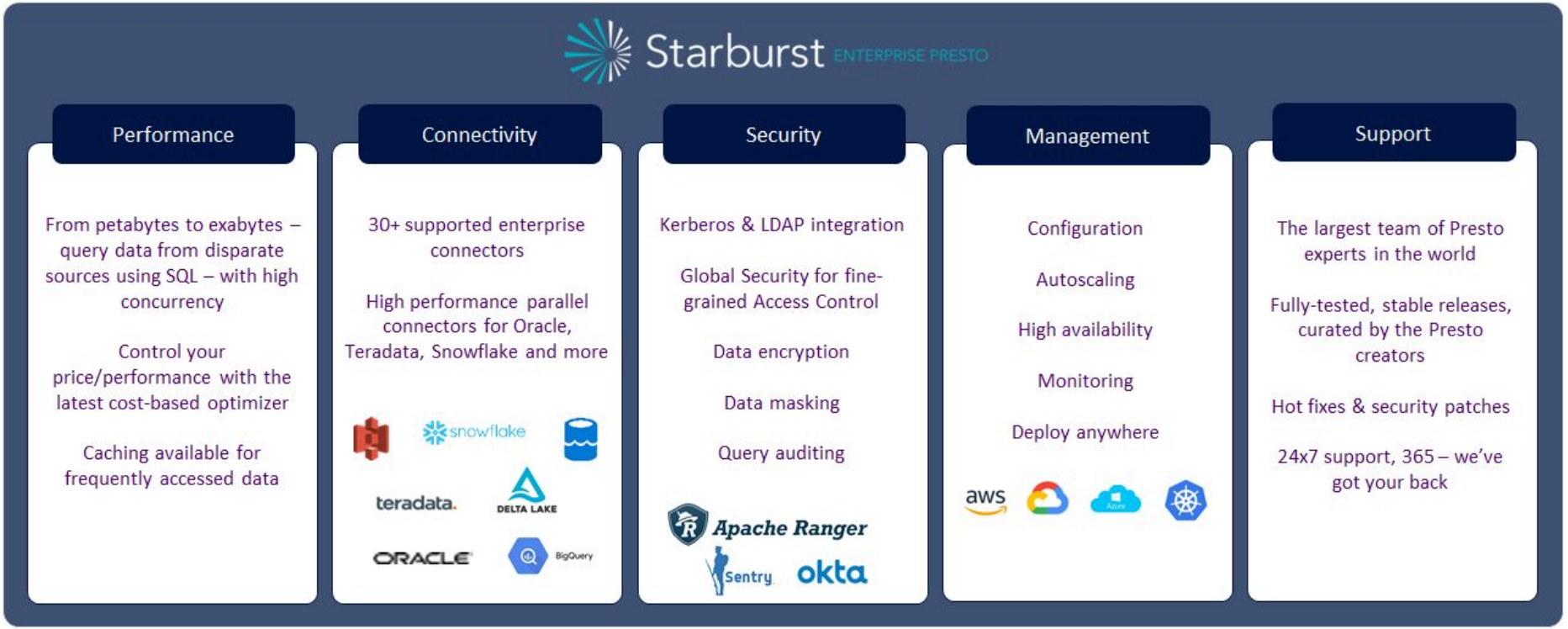
What is Starburst?

- Starburst Enterprise Presto - distribution
- Open core model
- Biggest open source Presto contributor

- Headquartered in Boston, MA
- Regional presence in EMEA - offices in Warsaw and London

Starburst

Starburst Enterprise Presto

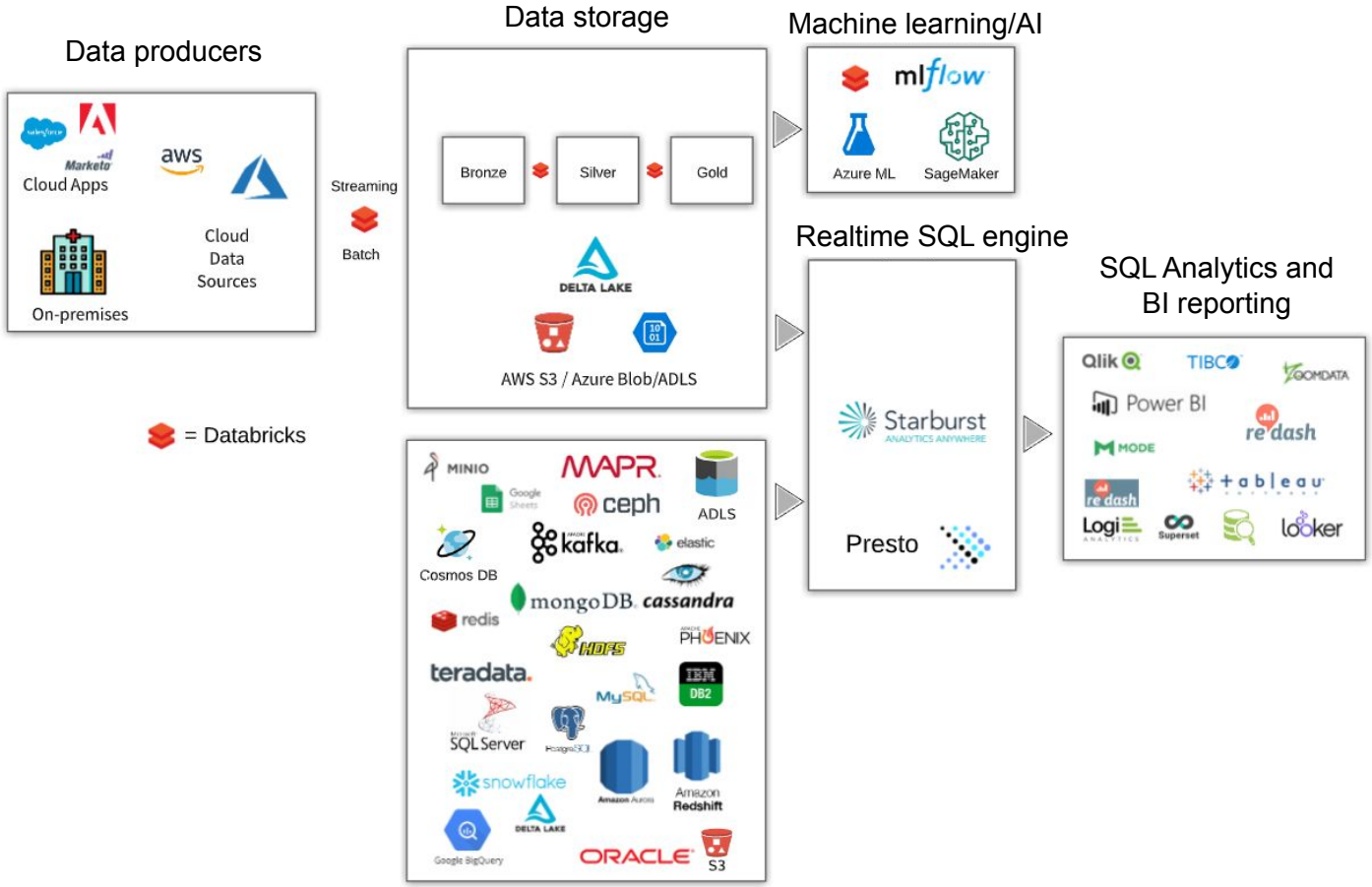


Data sources

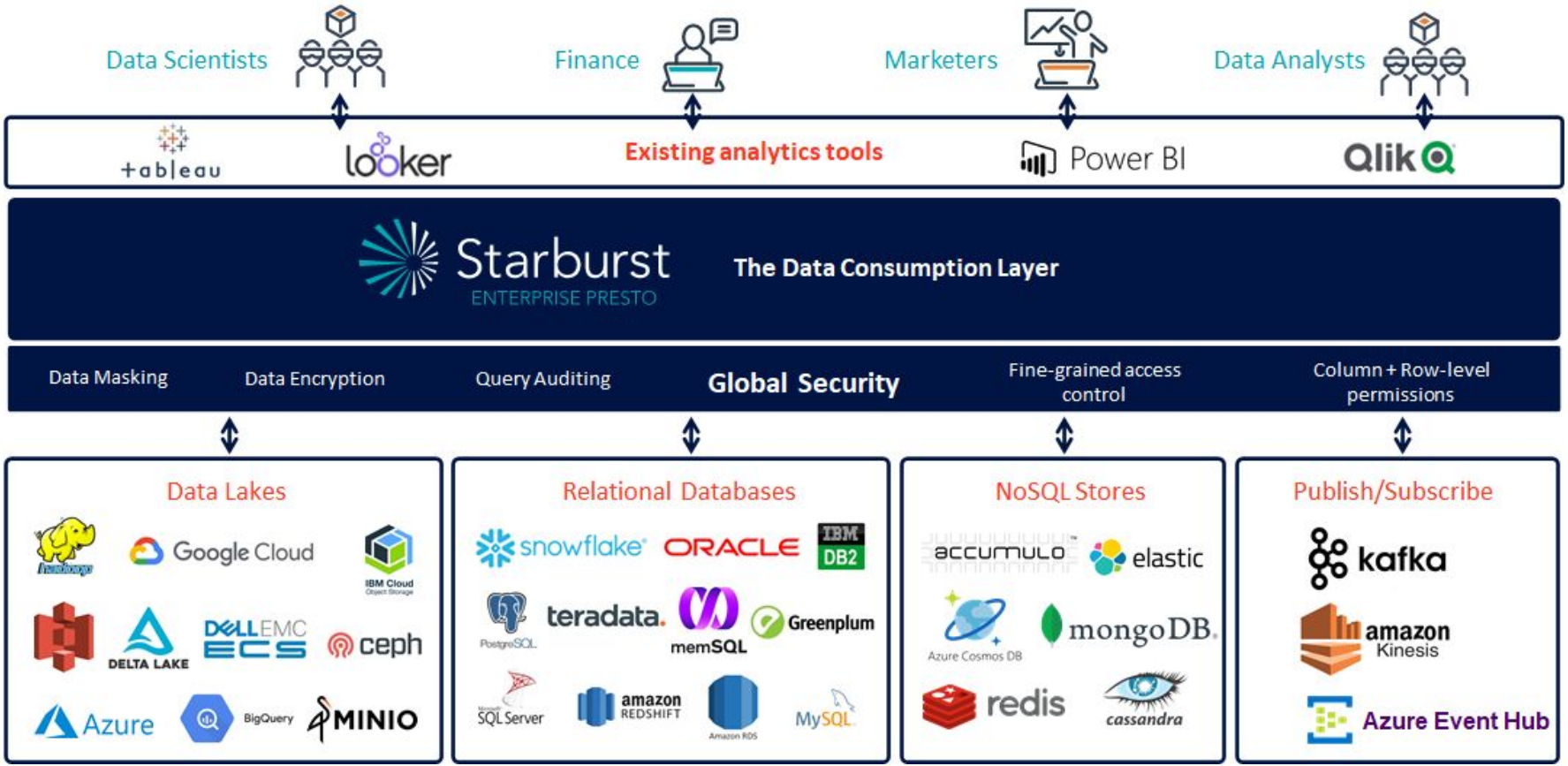
presto

Presto in data analysis ecosystem

Data Ingestion and Analytics Ecosystem

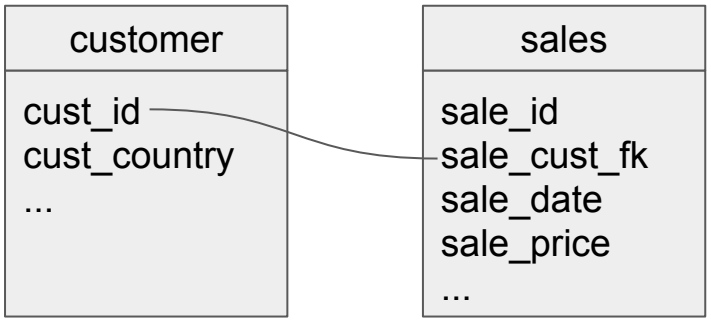


Starburst Platform



Under the hood

Static partition pruning



sales table partitioned by sale_date

Static partition pruning saves the day!

Only 31 partitions of sales table will be scanned

```
SELECT cust_country, sum(sale_price)
FROM customer JOIN sales ON cust_id = sale_cust_fk
WHERE sale_date >= date '2012-08-01' and sale_date <= date '2012-08-31'
GROUP BY cust_country
```

Dynamic partition pruning

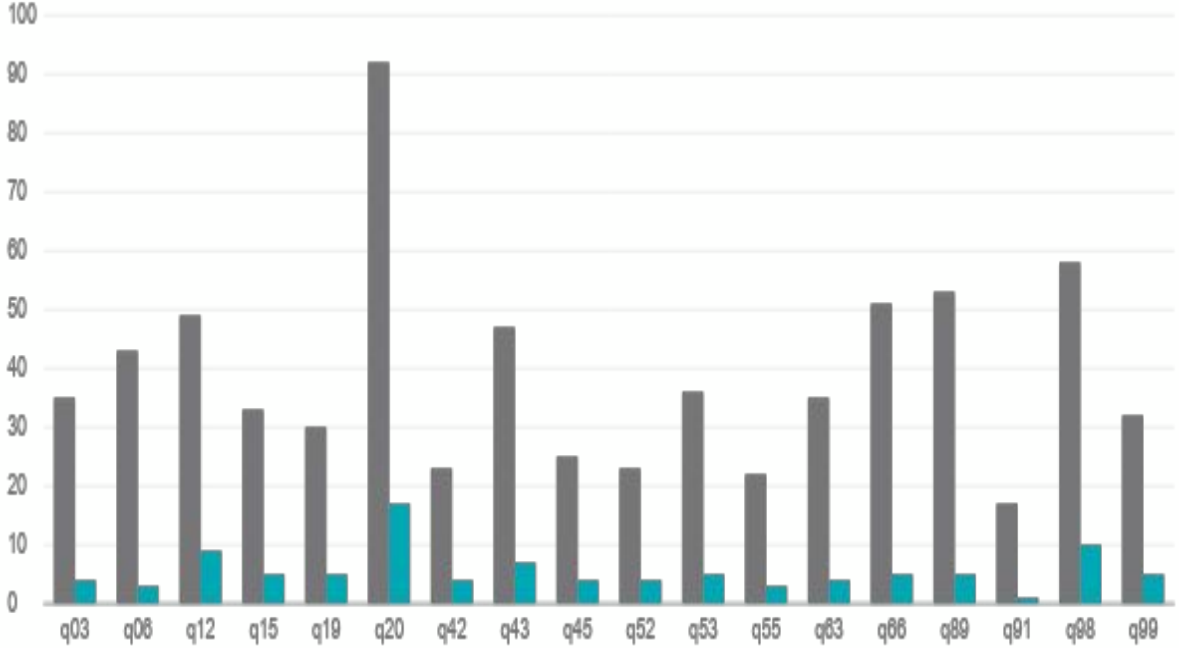
sales table partitioned by sale_date_fk

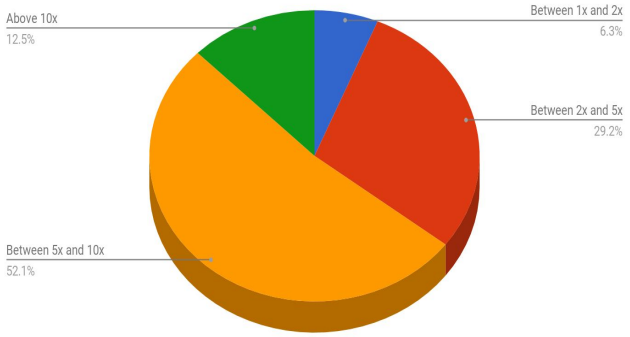
Dynamic partition pruning saves the day!

Only ~31 partitions of sales table will be scanned

```
SELECT cust_country, sum(sale_price)
FROM customer
    JOIN sales ON cust_id = sale_cust_fk
    JOIN date_dim ON sale_date_fk = date_id
WHERE date_date >= date '2012-08-01' and date_date <= date '2012-08-31'
GROUP BY cust_country
```

Cost based optimizer





TPC-DS queries, with CBO on and off

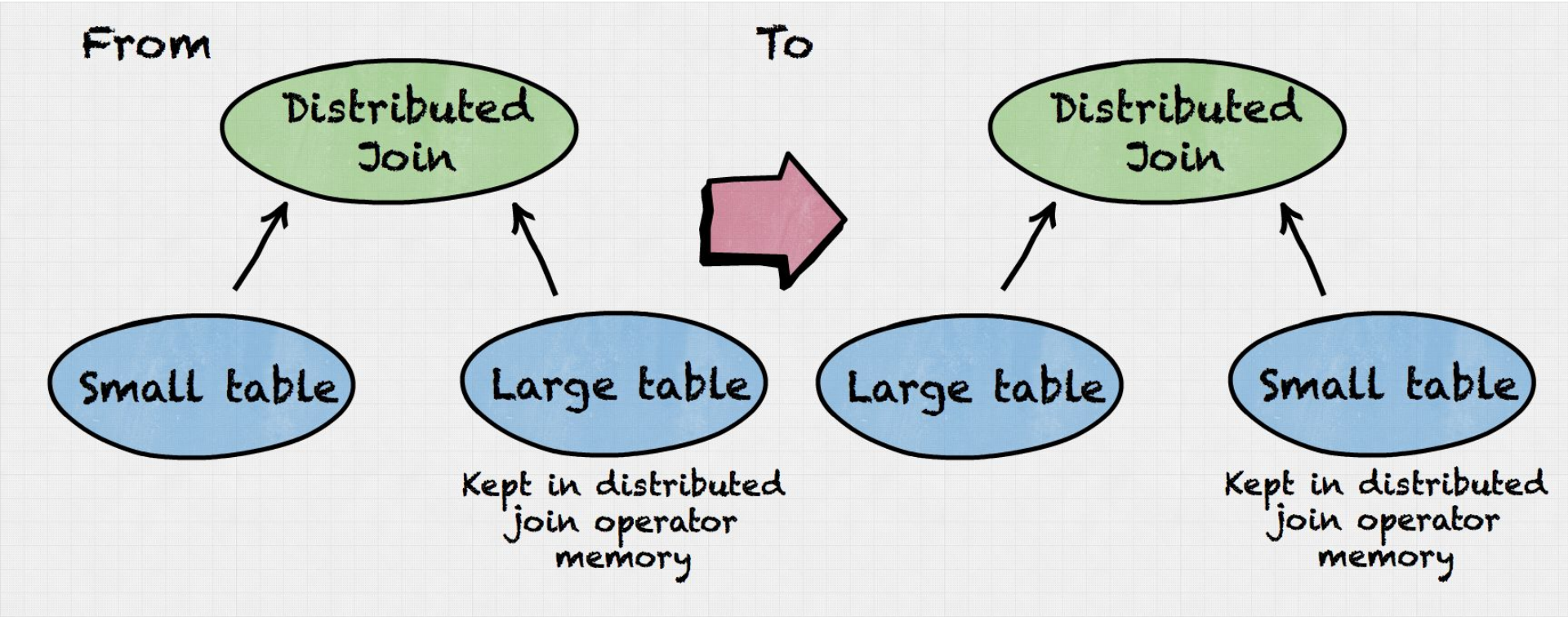
Cost based optimizer

Cost-Based Optimizer includes:

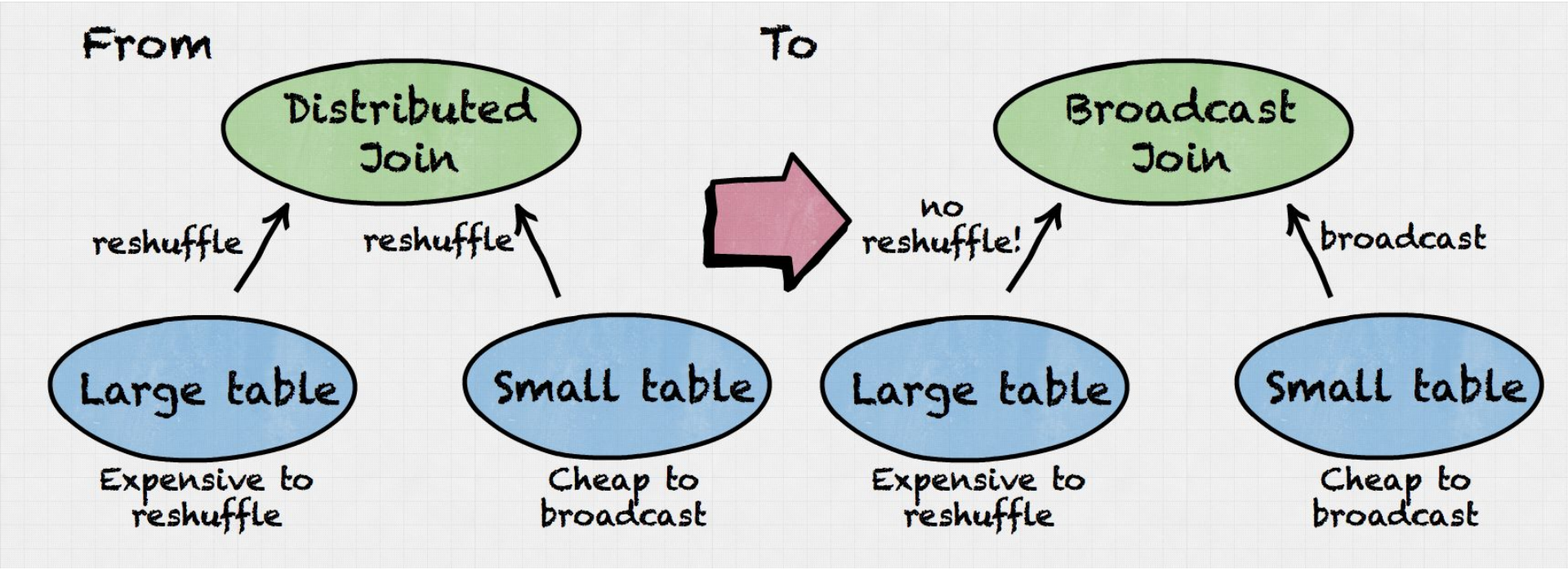
- **join reordering** based on selectivity estimates and cost
- automatic **join type** selection (repartitioned vs broadcast)
- automatic left/right **side selection** for joined tables
- support for **statistics** stored in Hive Metastore

https://www.starburstdata.com/technical-blog/

Join left/right side decision



Join type selection

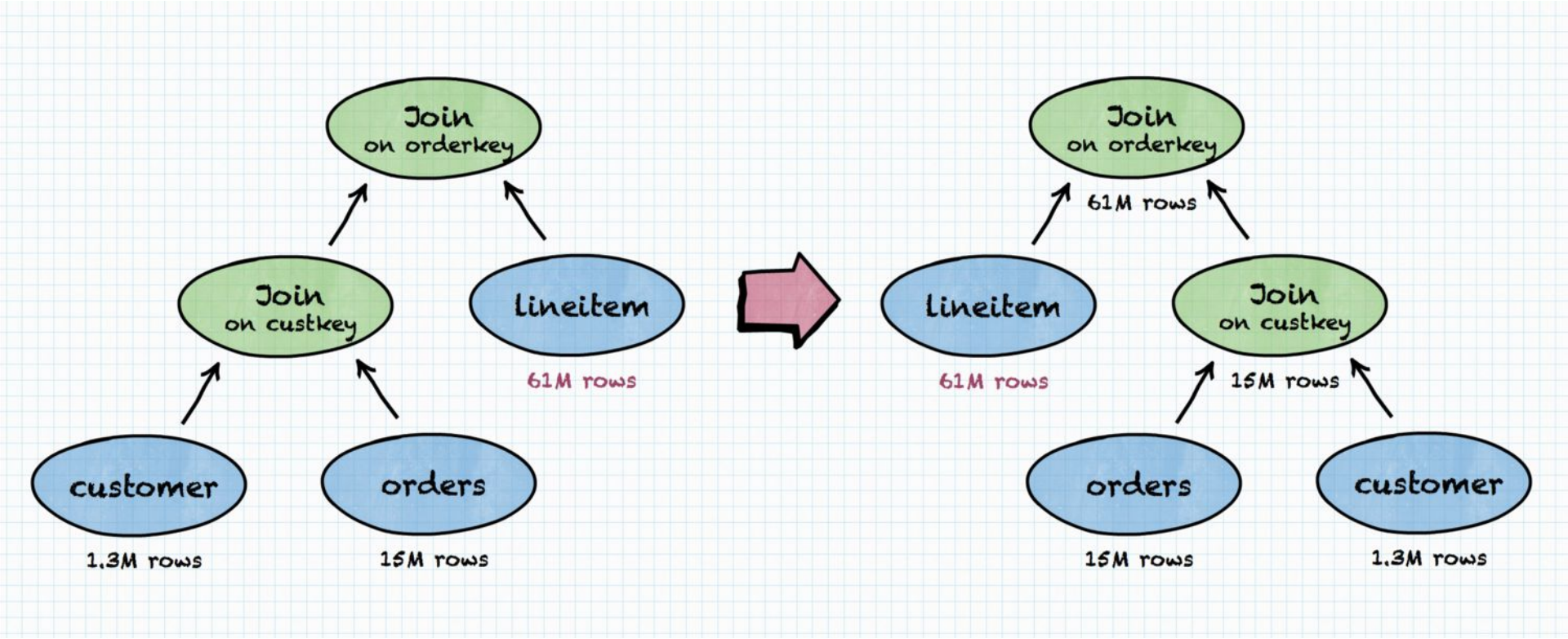


Join reordering

"Which customers are spending the most at our shop?"

```
SELECT c.custkey, sum(l.price)
FROM customer c
JOIN orders o ON c.custkey = o.custkey
JOIN lineitem l ON o.orderkey = l.orderkey
GROUP BY c.custkey
ORDER BY sum(l.price) DESC;
```

Join reordering

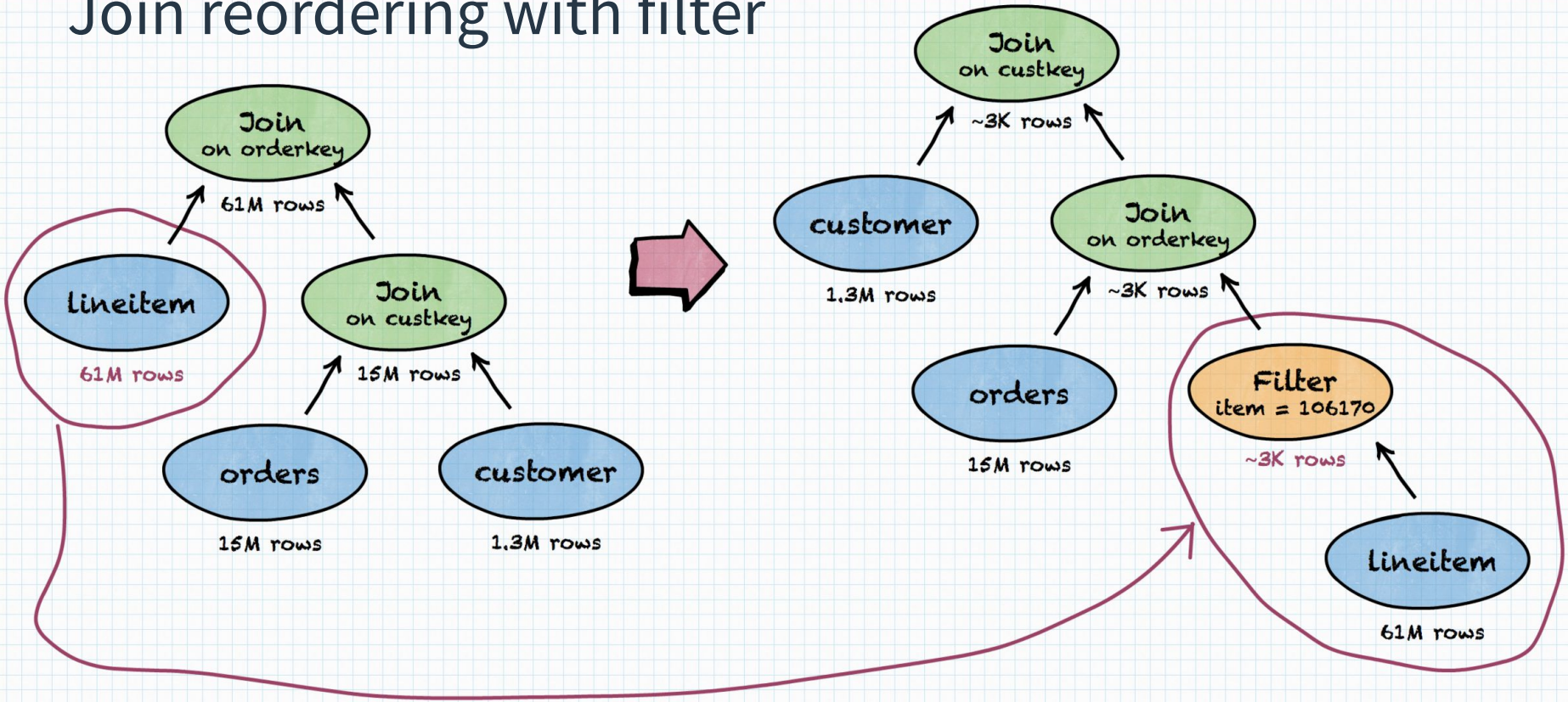


Join reordering with filter

"Which customers are spending the most on coffee?"

```
SELECT c.custkey, sum(l.price)
FROM customer c
JOIN orders o ON c.custkey = o.custkey
JOIN lineitem l ON o.orderkey = l.orderkey
WHERE l.item = 'coffee'
GROUP BY c.custkey
ORDER BY sum(l.price) DESC;
```

Join reordering with filter



Demo time

TPC-DS schema