Technische Universität Berlin

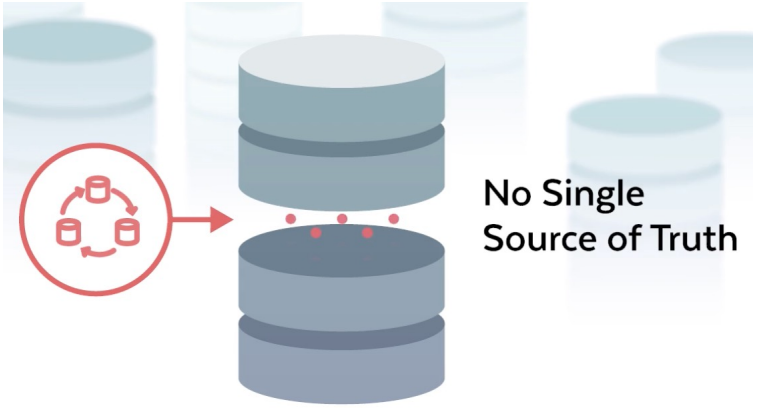
Mediatorless Cross-DB Query Execution

Jan Hanack | BDAPRO WiSe 21/22| 24.02.2022

Technische Universität Berlin

Recap & Relevance

- Use of Data Federations
  - Inter-Organizational
  - Intra-Organizational
- Enrich data analysis with external data
  - e.g. weather data for insurers, financial data, climate data for research
- Data Protection while cooperating
- Globalization & Big Data



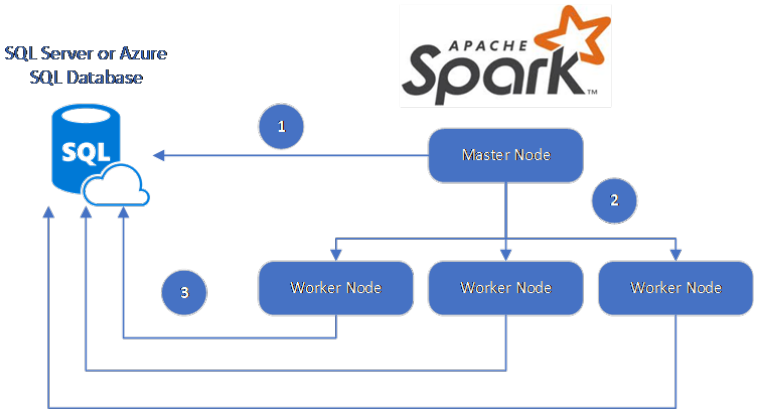
Technische Universität Berlin

Problem Statement

- Currently Mediator-Wrapper approach used
- Additional Data Transfer necessary

→ Increasing overhead and latency

→ Complicates Data Compliance



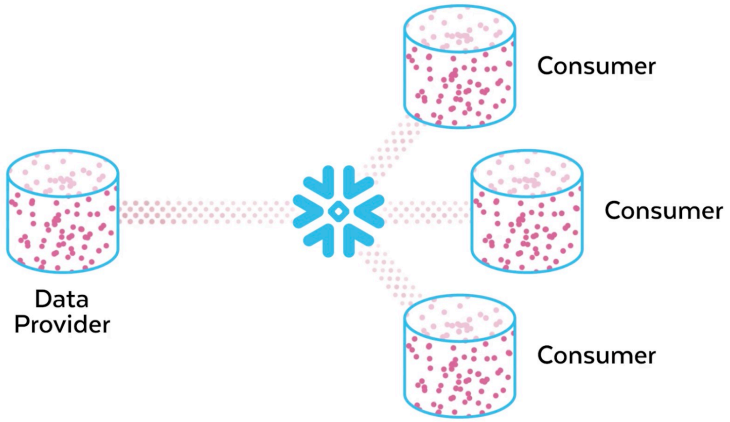
→ Bad fit for large scope, decentralized collaboration systems

- Akka's Actor Model non-ideal solution

Technische Universität Berlin

Goals

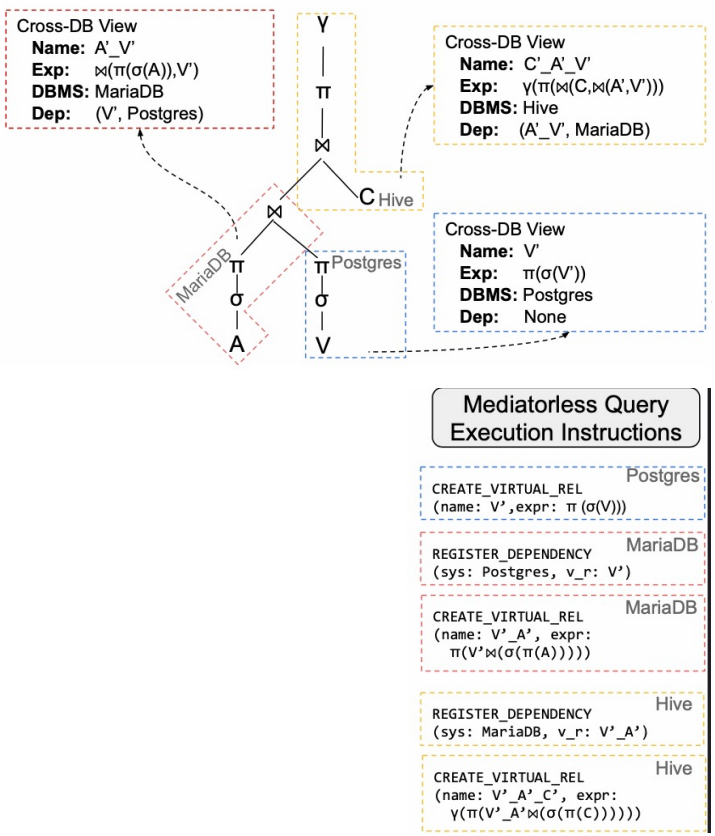
- Develop Mediator-less approach
- As little as possible data copied & transferred
- Interoperability (e.g PostgreSQL, MariaDB)
- Similar to AWS Data Exchange or Snowflake
- Use output of Agora's execution planner



AGORA
— A Data Ecosystem for AI Innovation —

Solution & Contributions - Mediatorless Cross-DB Query Execution

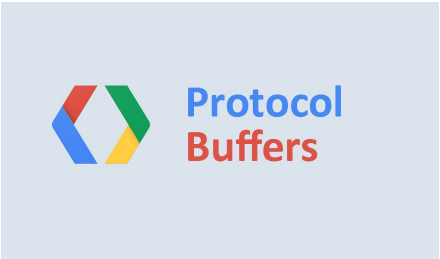
- Interoperability between different DBMS
- Distributed setting possible
- DBMS and nodes communicate directly
- Foreign Tables
  - Point to Table in different DB
- Views
  - Generate Sub-Query Result



Technische Universität Berlin

Solution & Contributions – Inter-Node/DB-Commuication

- Protocol Buffers
  - Serializes structured Data
  - Increases efficiency for transfer

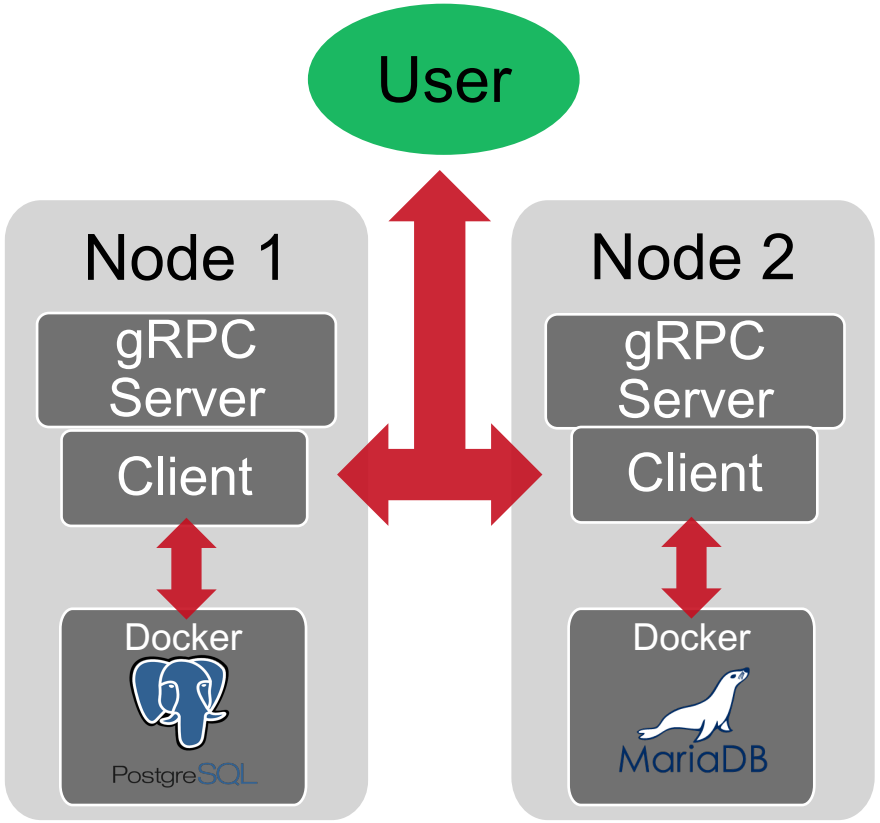


- gRPC
  - Uses Protocol Buffers
  - Client & Server Bindings
  - Easy Specification of services
  - Communication between Clients (DB Nodes)
  - Allows for millions of RPCs per second

Technische Universität Berlin

Solution & Contributions – Architecture

- Slim server implementation per node
- One Client per Node and user
- User calls createForeignTable() and createView()
- DBs communicate directly
- Final Query result sent to user
- One or many DBs per node



Technische Universität Berlin

Solution & Contributions – XDBX Implementation

- DBManager maps
  - DB ID ⇢ e.g. localhost:8080 ⇢ DB Type (e.g. POSTGRES)

- One connector class per DB brand
  - getColumns() converts to standard format and back

- Two connections maintained:
  1. User ↔ Node Client

```
new SimpleXDBConnection( address: "localhost",  port: 7000,  username: "max",  authentication: "123");
```

  2. Client ↔ Database
     - JDBC or ODBC ⇢

```
new PostgresInstance(clientFactory),
dbAddress: "localhost",
dbPort: 8081,
dbUsername: "bdapro_user",
dbPassword: "bdapro_password",
database: "bdapro_database");
```

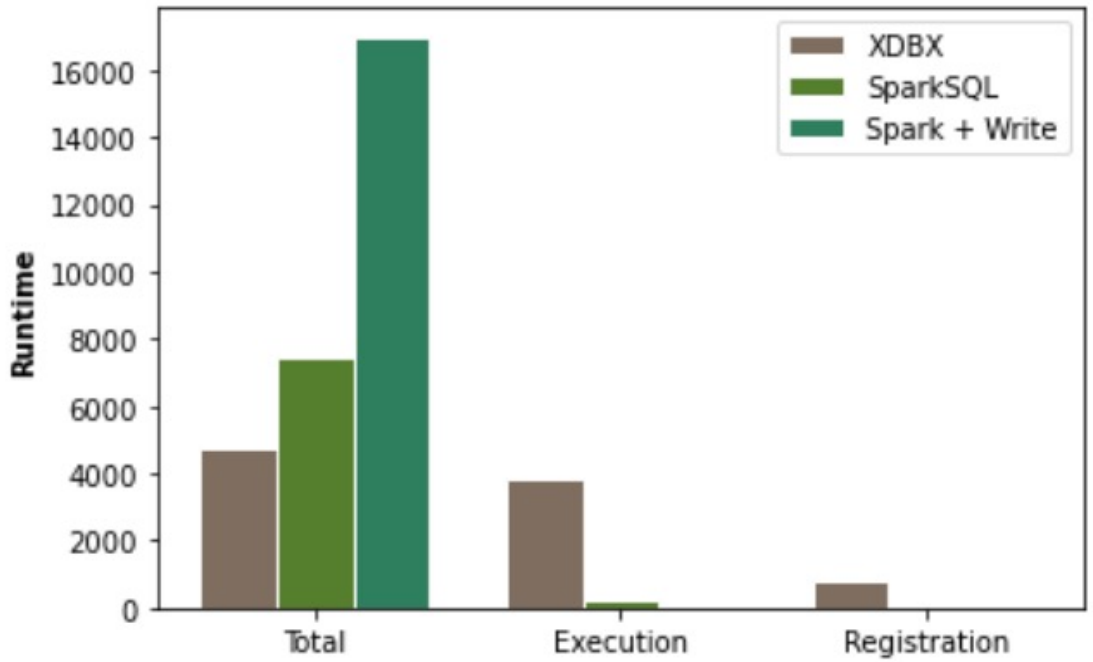
Technische Universität Berlin

Solution & Contributions – Experiments Setup

- Two virtual nodes
- 2-3 PostgreSQL containers via JDBC
- TPC-H Data & Queries
  - Common Benchmarking Tool
  - Scale Factor 1
  - Data distributed over DBs
- Benchmarks
  - Apache SparkSQL
  - Presto
  - Apache Calcite

Technische Universität Berlin

Solution & Contributions – Benchmarks - TPC-H Query 3



- Machine: 6 Core CPU, 16 GB RAM

Technische Universität Berlin

Solution & Contributions – Summary & Evaluation

- System works, query output correct
- Further improvement by use of indices
- For Report:
  - More TPC-H queries
  - Benchmark scale factor 10
  - Different DB setup
- Performance depends on workload (filters, or data manipulation, disk writes)
- Spark can be used on top of XDBX

Technische Universität Berlin

Thank You for your Attention!