Prestogres

PostgreSQL protocol gateway for Presto

Hacking PostgreSQL Internals to Solve Data Access Problems

Sadayuki Furuhashi

Treasure Data, Inc.
Founder & Software Architect

A little about me...

- **Sadayuki Furuhashi**
  - github/twitter: @frsyuki
- **Treasure Data, Inc.**
  - Founder & Software Architect
- **Open source hacker**

Open source

Fluentd - Unifid log collection infrastracture

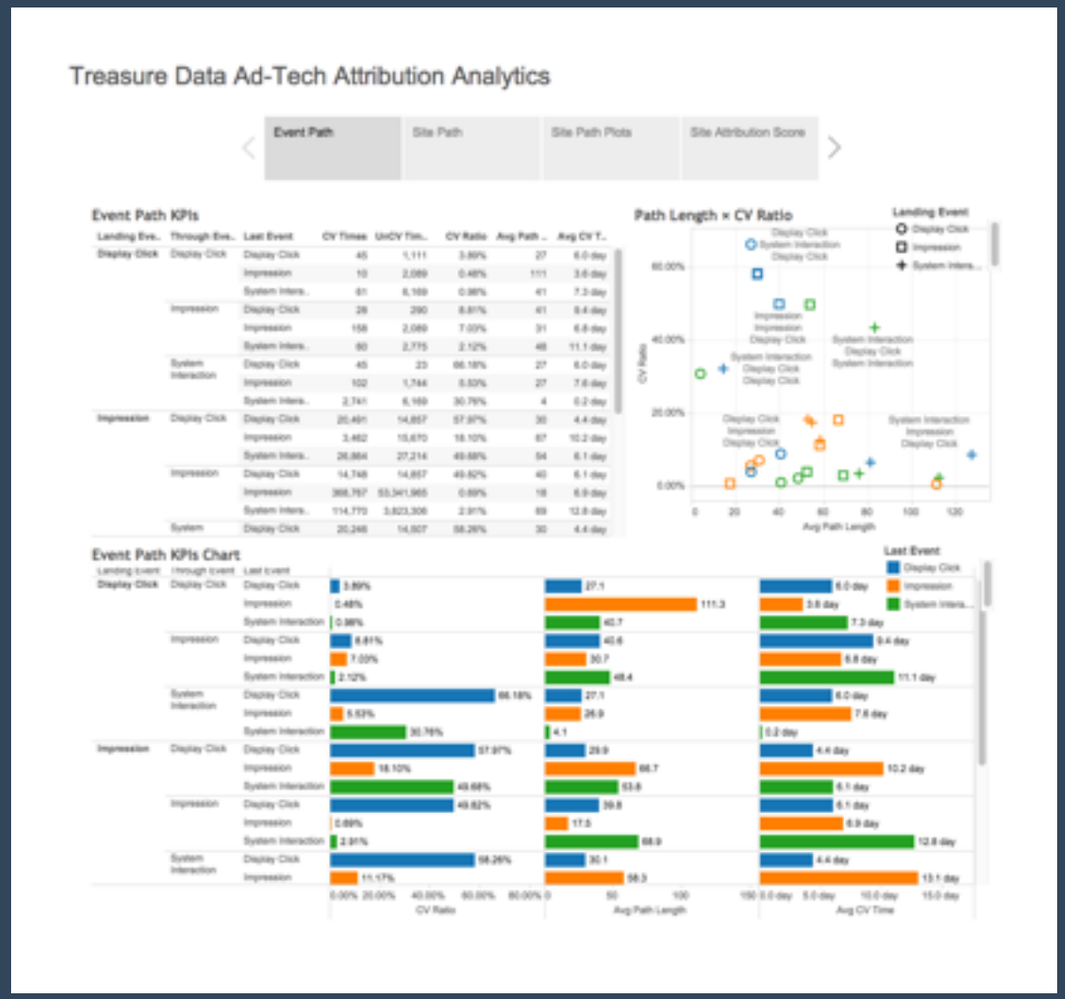
Embulk - Plugin-based parallel ETL

MessagePack - Schemaless serialization format

MessagePack

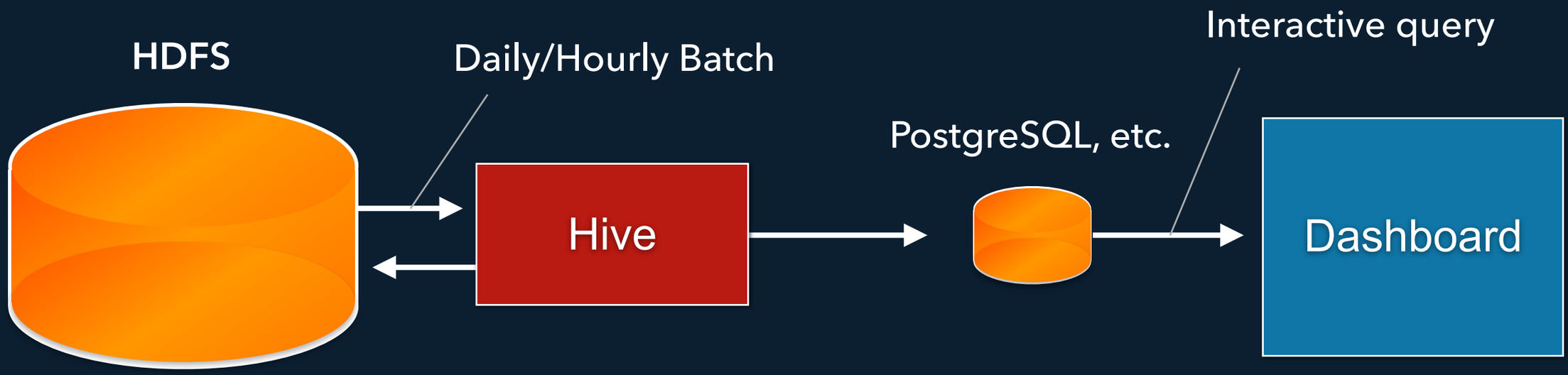
TREASURE DATA

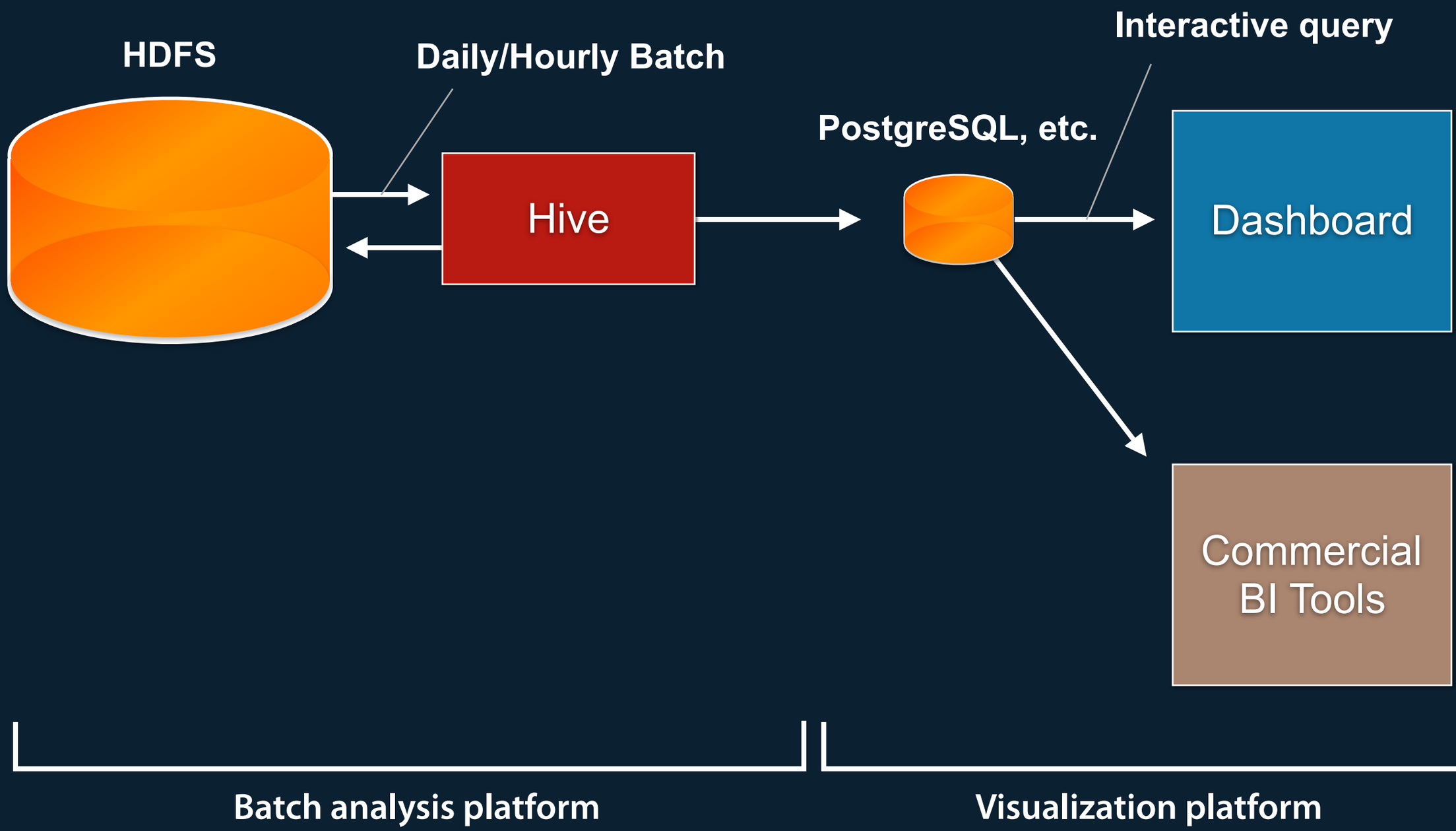
End-to-end data analytics pipeline on the cloud.

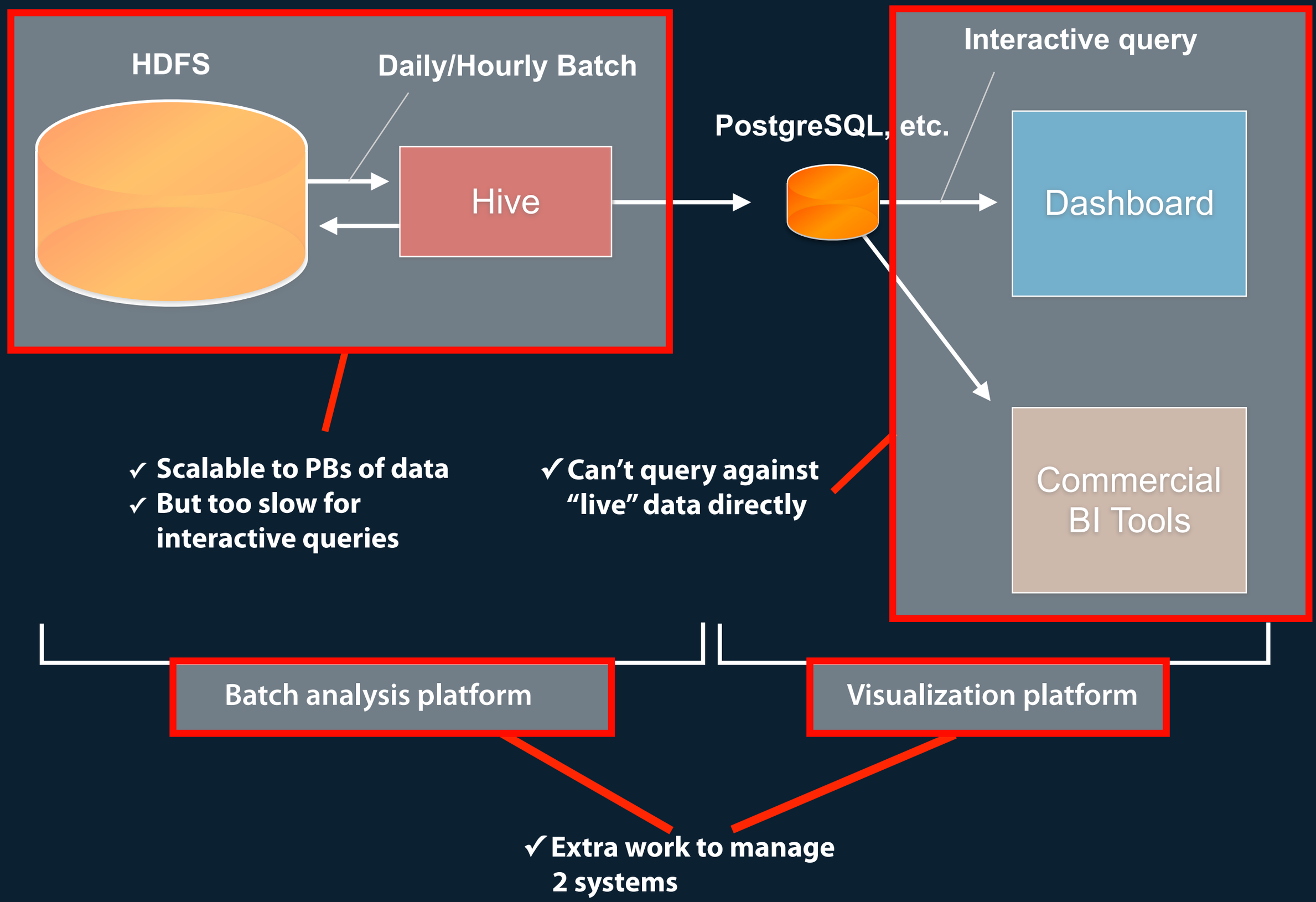


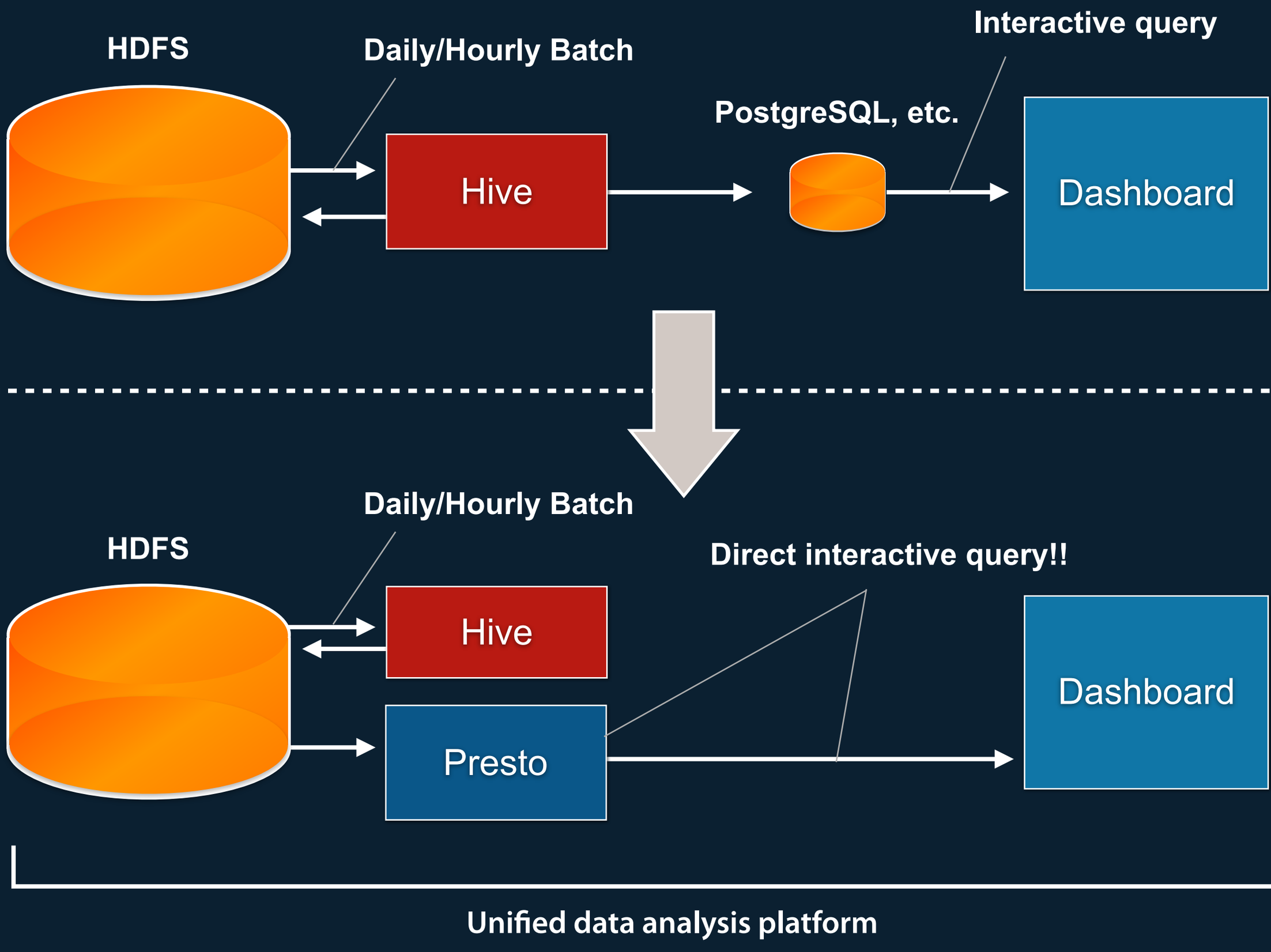
Motivation of Prestogres

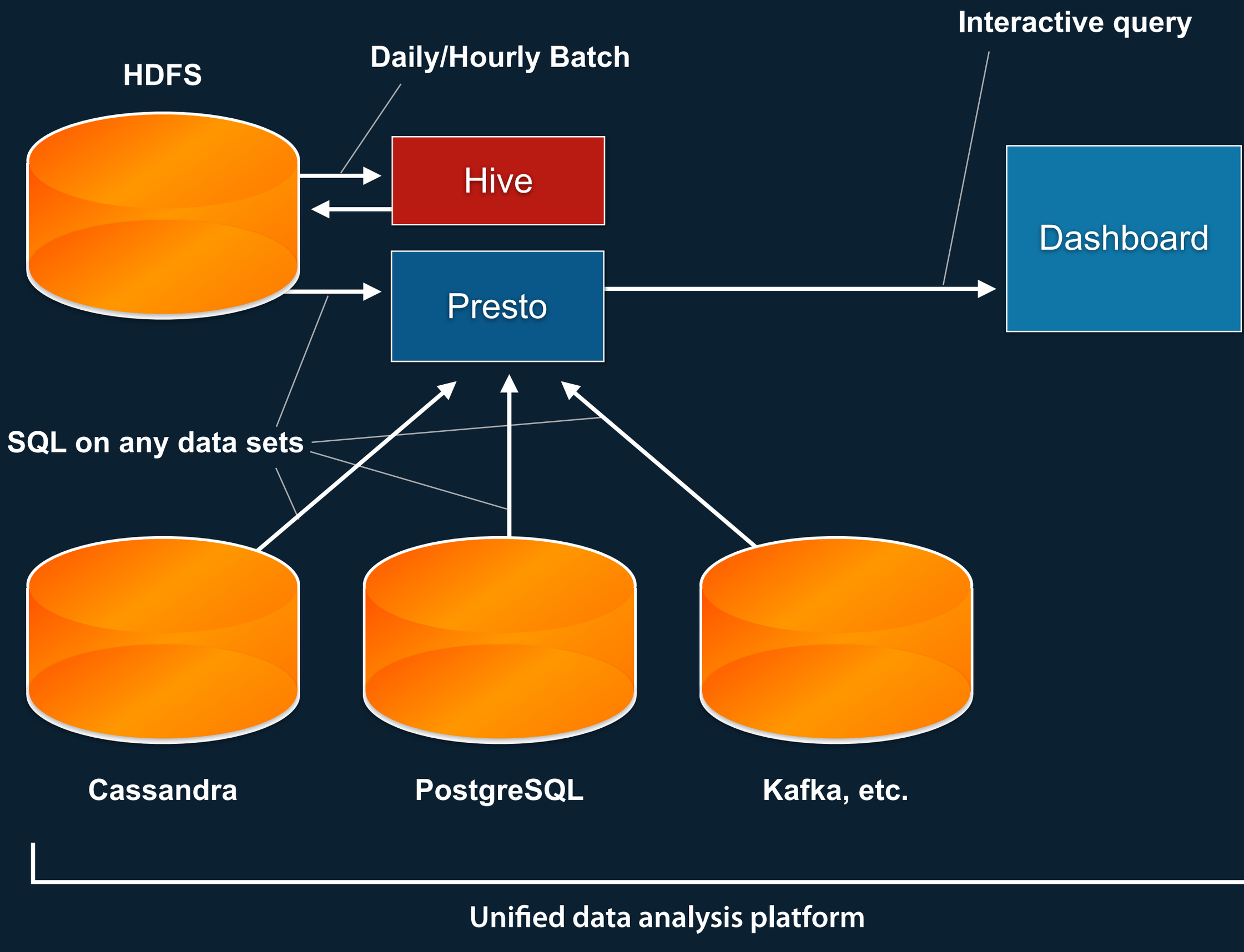
> **I want to build an open-source ODBC connectivity directly to a big data analytics infrastracture.**




HDFS
Daily/Hourly Batch
Hive
PostgreSQL, etc.
Interactive query
Dashboard
Commercial
BI Tools
Batch analysis platform
Visualization platform


HDFS
Daily/Hourly Batch
Hive
PostgreSQL, etc.
Interactive query
Dashboard
Commercial BI Tools
✓ Scalable to PBs of data
✓ But too slow for interactive queries
✓ Can't query against "live" data directly
Batch analysis platform
Visualization platform
✓ Extra work to manage 2 systems


HDFS
Daily/Hourly Batch
Hive
PostgreSQL, etc.
Interactive query
Dashboard
HDFS
Daily/Hourly Batch
Hive
Presto
Direct interactive query!!
Dashboard
Unified data analysis platform


HDFS
Daily/Hourly Batch
Interactive query
Hive
Presto
Dashboard
SQL on any data sets
Cassandra
PostgreSQL
Kafka, etc.
Unified data analysis platform

Interactive query

HDFS

Daily/Hourly Batch

Hive

Presto

Dashboard

ODBC

Commercial BI Tools

SQL on any data sets

- ✓ IBM Cognos
- ✓ Tableau
- ✓ ...

Cassandra

PostgreSQL

Kafka, etc.

Unified data analysis platform

Interactive query

HDFS

Daily/Hourly Batch

Hive

Presto

Dashboard

Prestogres

Commercial BI Tools

SQL on any data sets

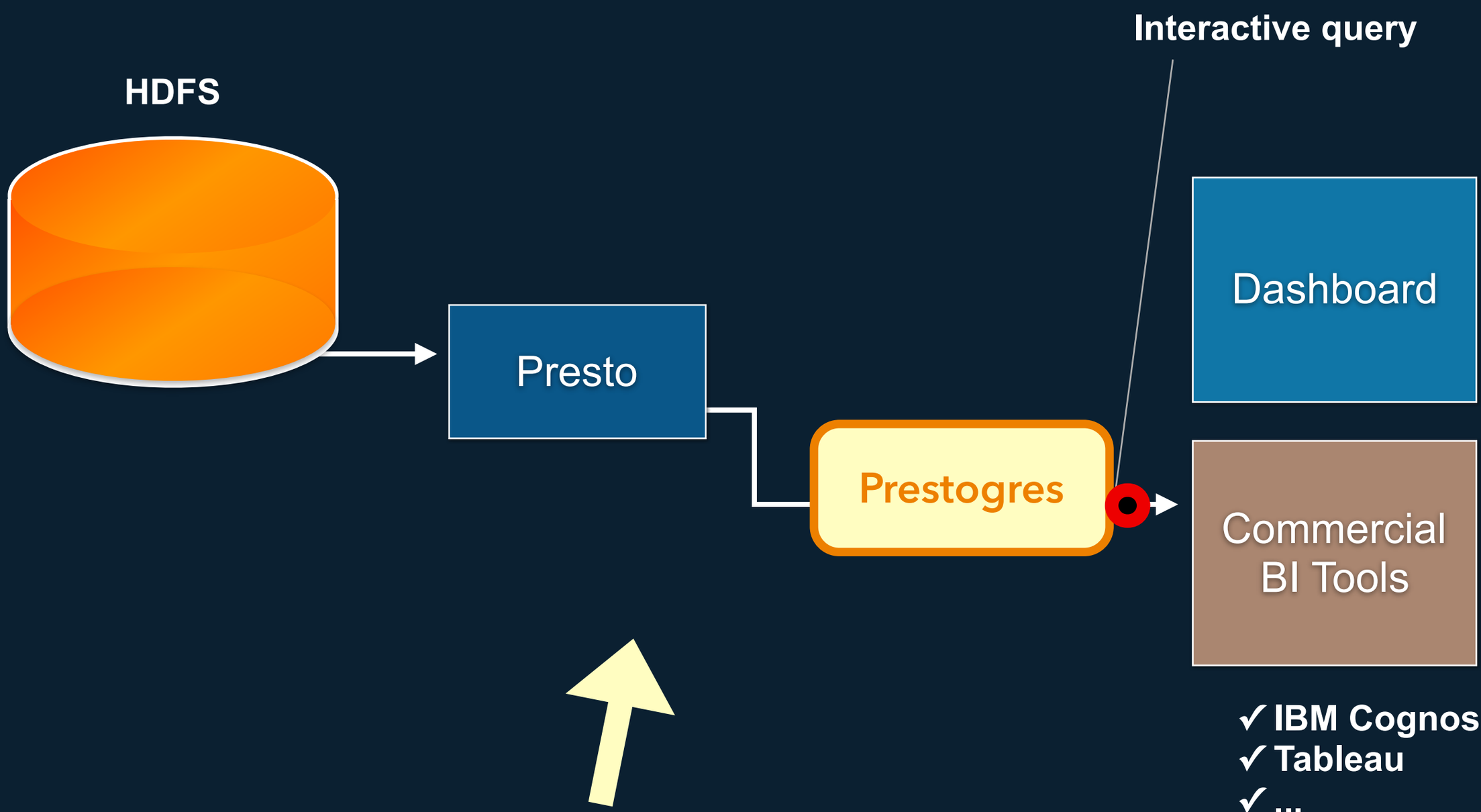
- ✓ IBM Cognos
- ✓ Tableau
- ✓ ...

Cassandra

PostgreSQL

Kafka, etc.

Unified data analysis platform

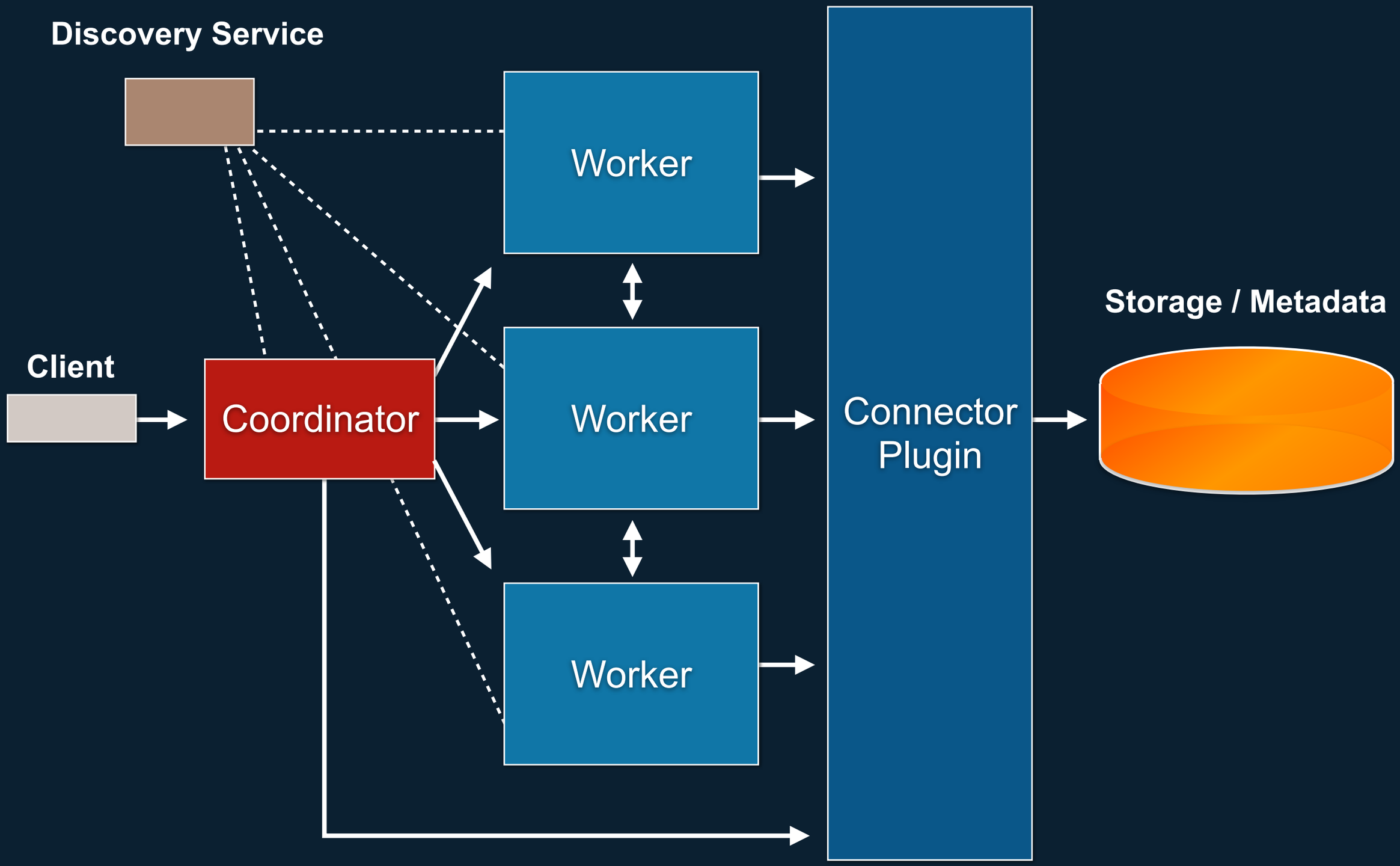


- ✓ IBM Cognos
- ✓ Tableau
- ✓ ...

Today's topic

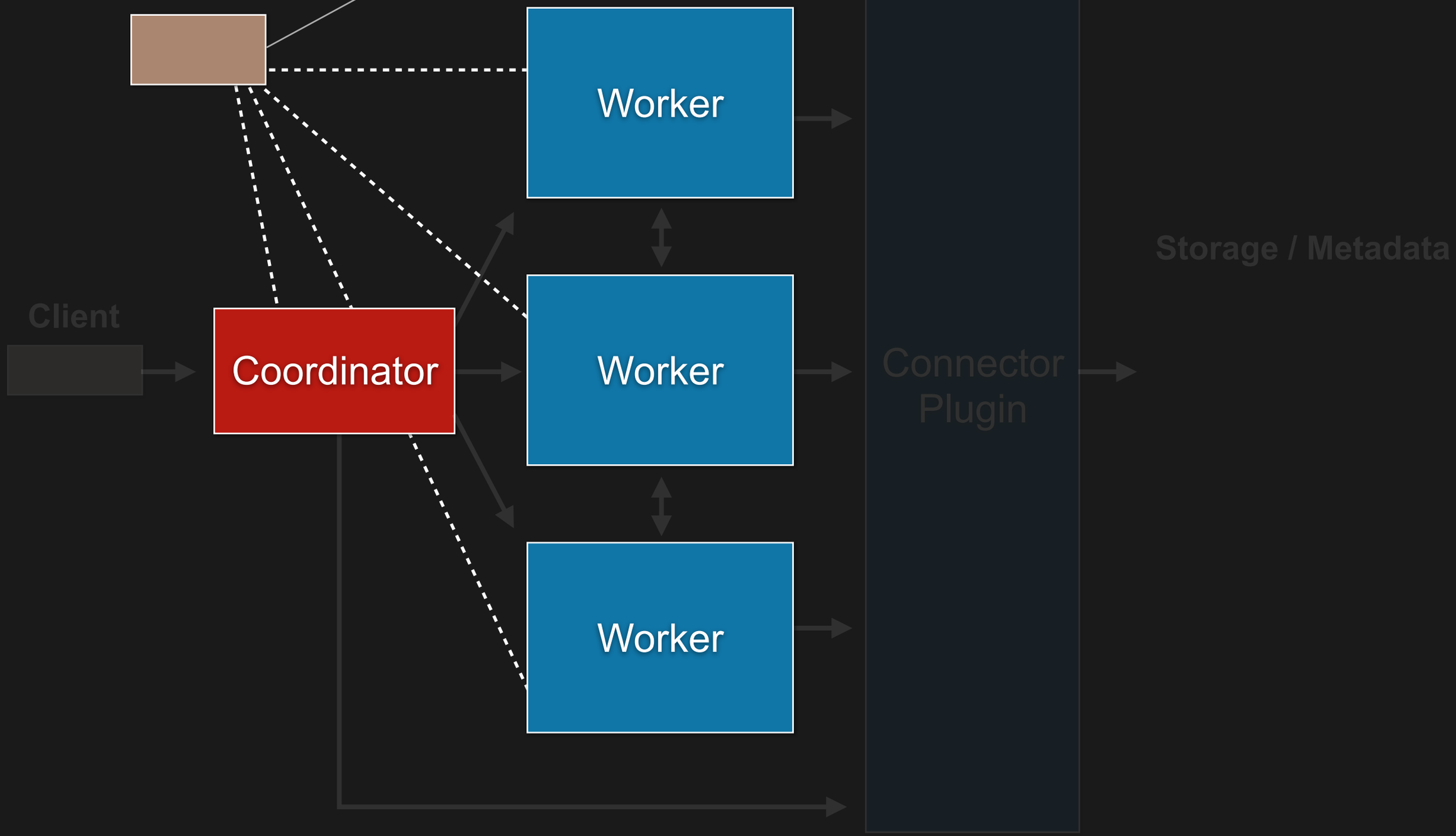
Q. Why do you choose Presto over other databases?

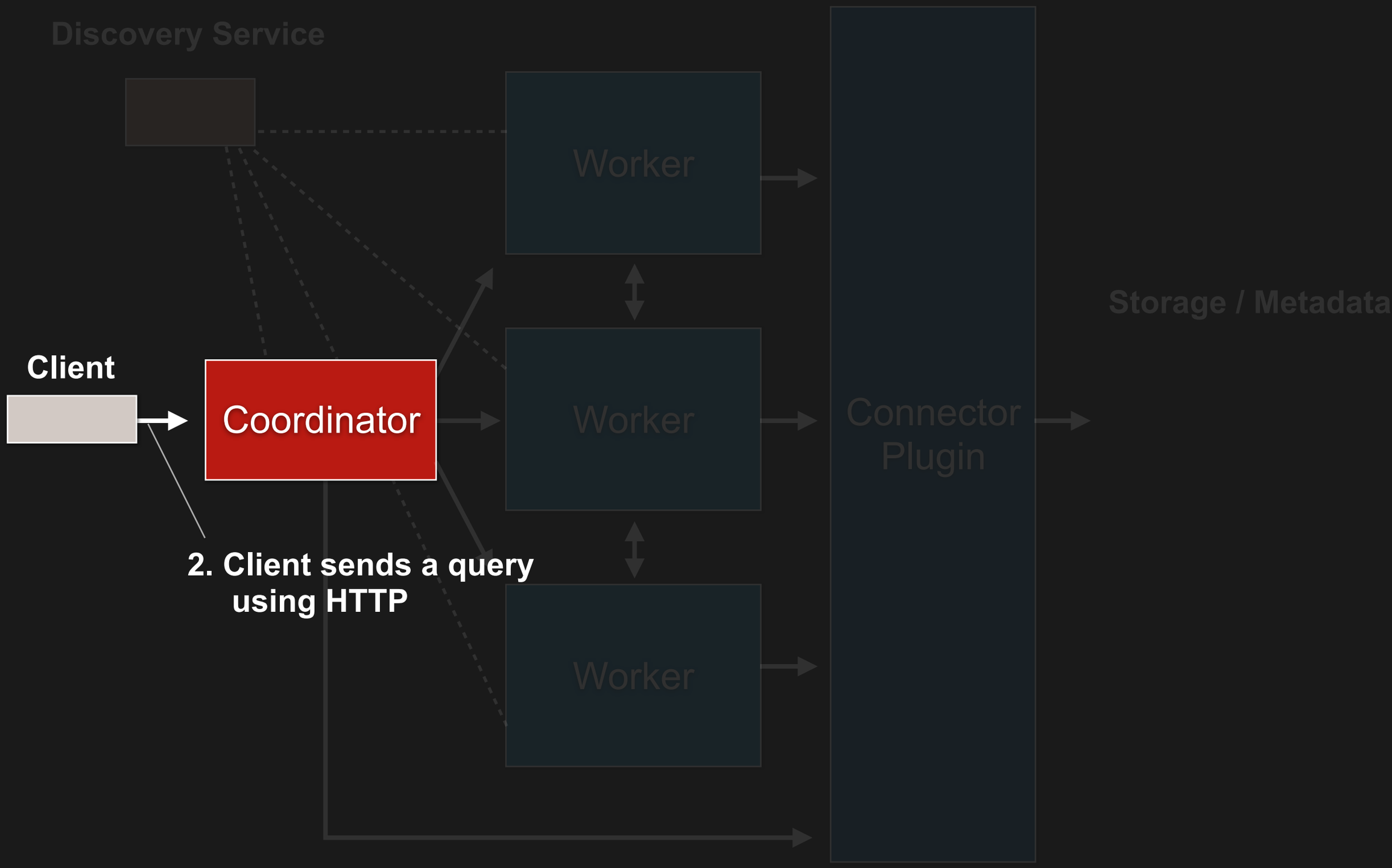
1. Why Presto? - Presto’s architecture

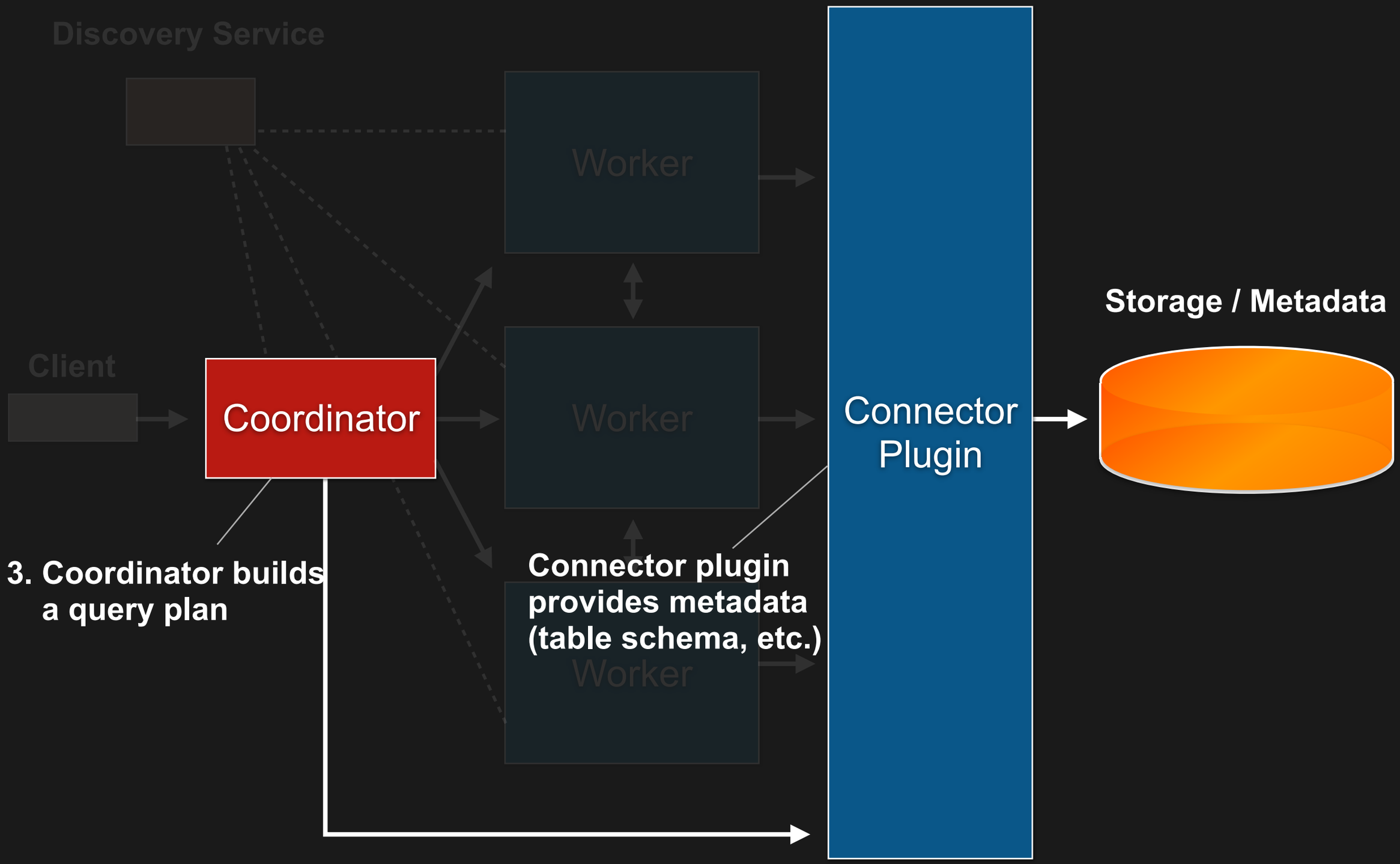

Discovery Service
Client
Coordinator
Worker
Worker
Worker
Connector Plugin
Storage / Metadata

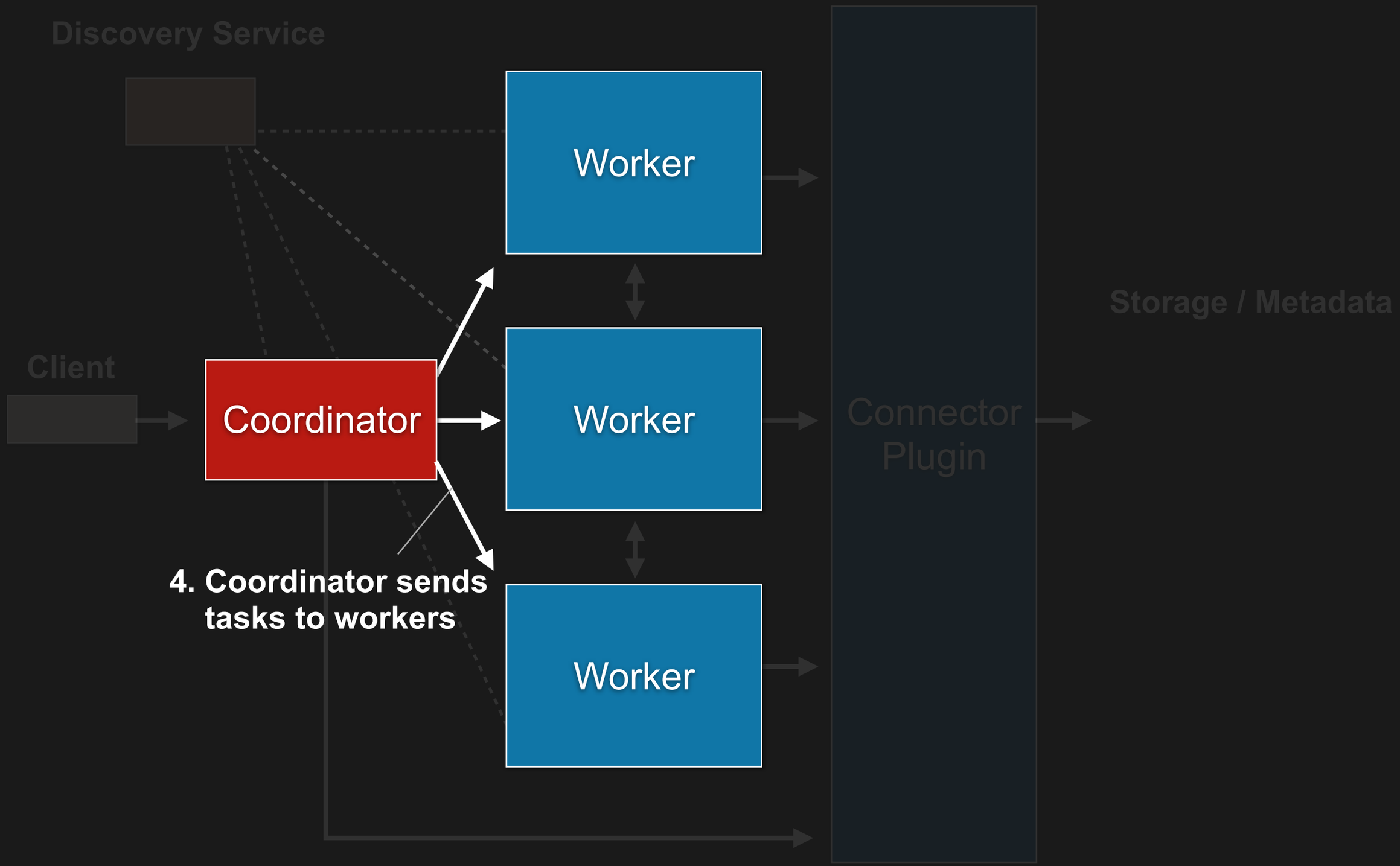
1. find servers in a cluster

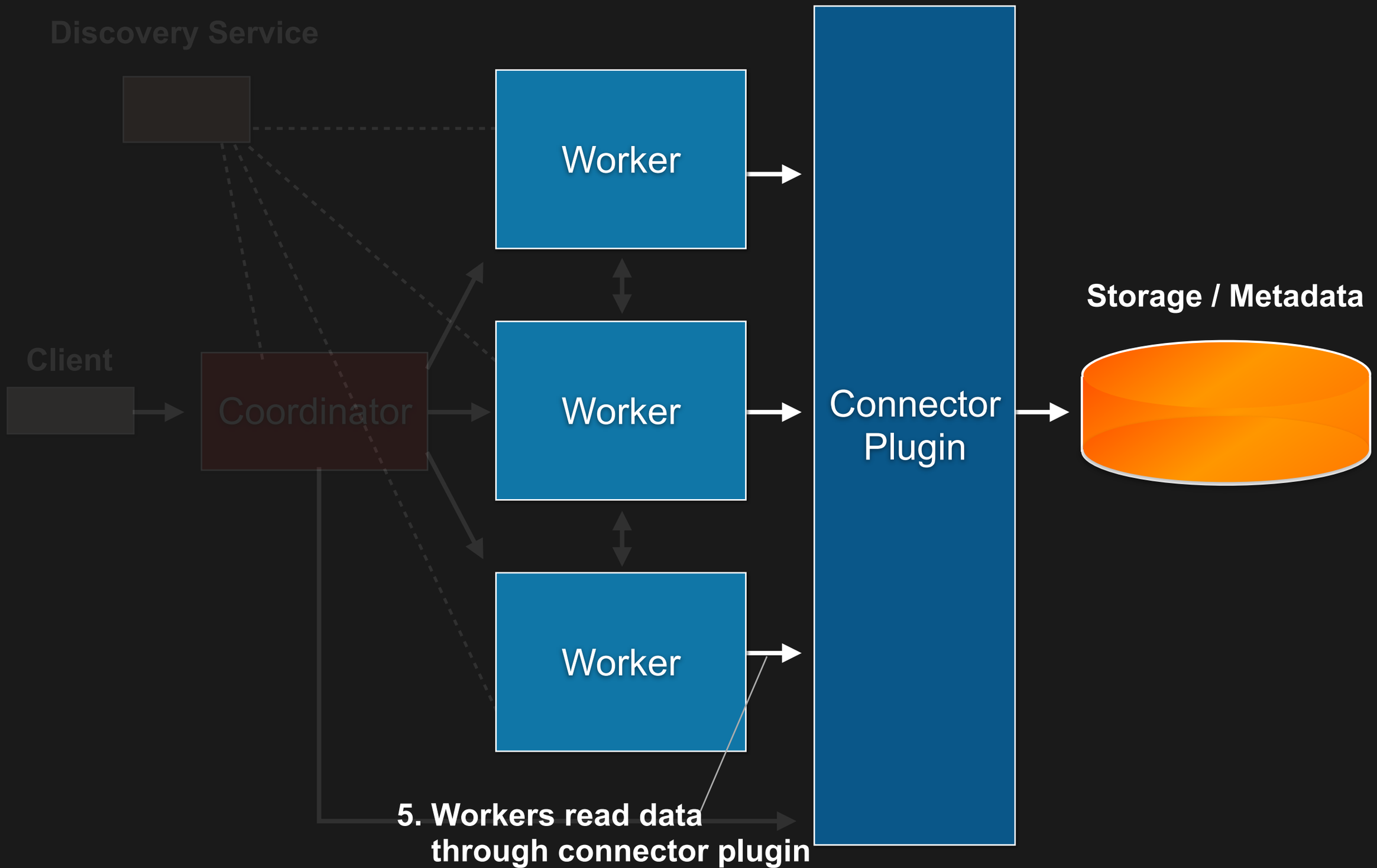
Discovery Service




Discovery Service
Worker
Storage / Metadata
Client
Coordinator
Worker
Connector Plugin
Worker
2. Client sends a query using HTTP


Discovery Service
Worker
Client
Coordinator
Worker
Connector Plugin
Storage / Metadata
3. Coordinator builds a query plan
Connector plugin provides metadata (table schema, etc.)
Worker


Discovery Service
Worker
Storage / Metadata
Client
Coordinator
Worker
Connector
Plugin
4. Coordinator sends
tasks to workers
Worker


Discovery Service
Client
Coordinator
Worker
Worker
Worker
Connector Plugin
Storage / Metadata
5. Workers read data through connector plugin

Discovery Service

Client

Coordinator

Worker

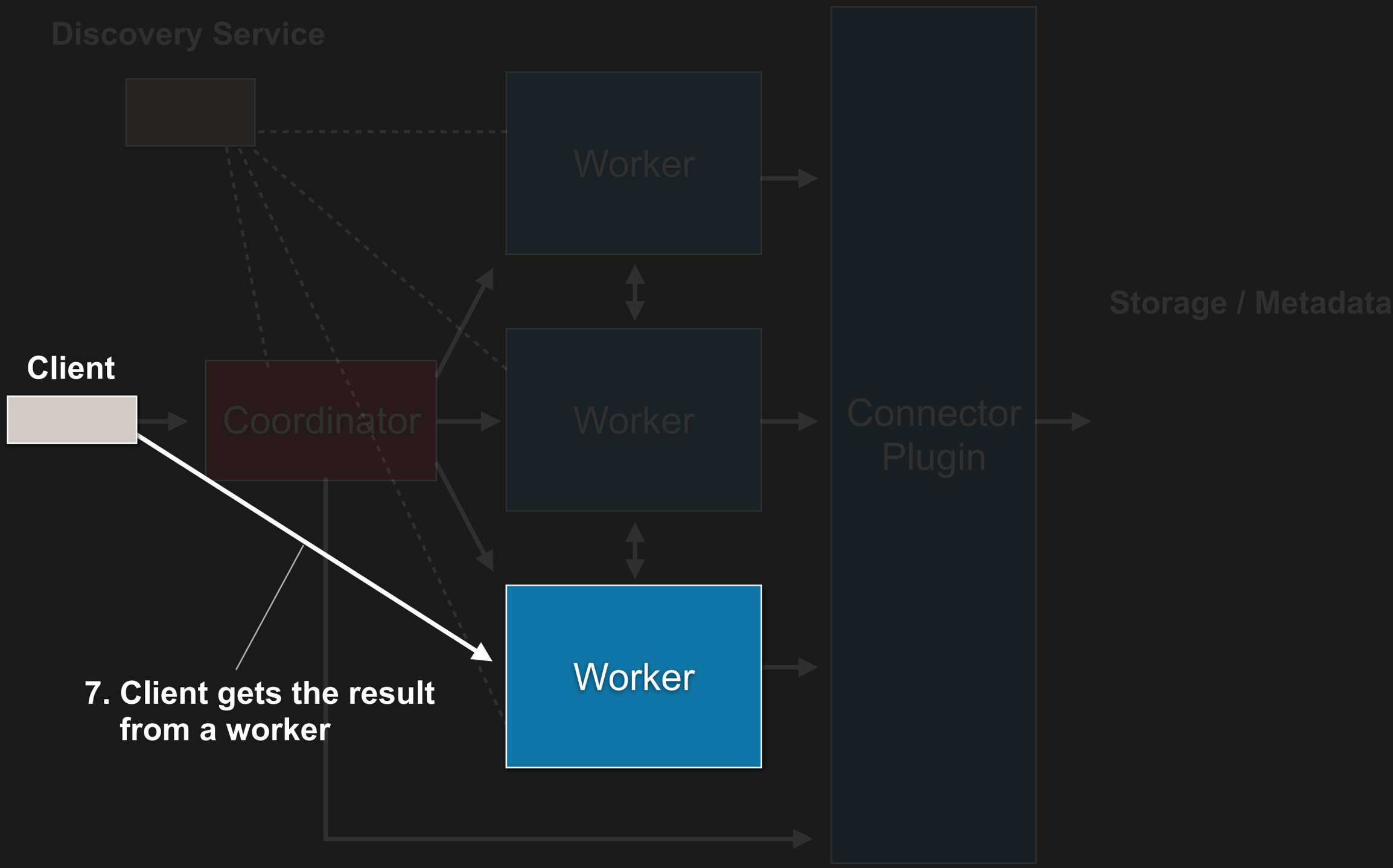
Worker

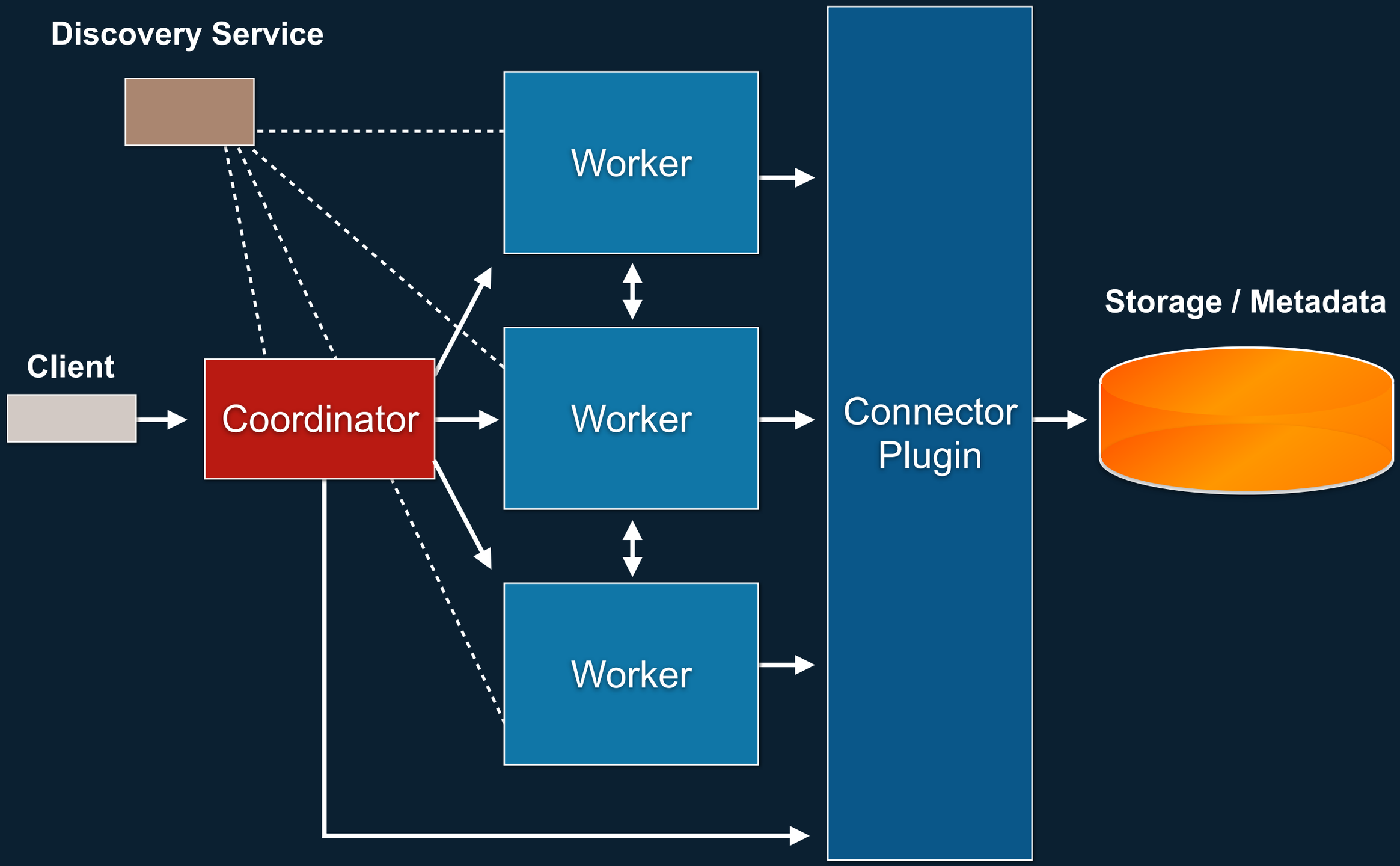
Worker

Connector Plugin

Storage / Metadata

6. Workers run tasks in memory

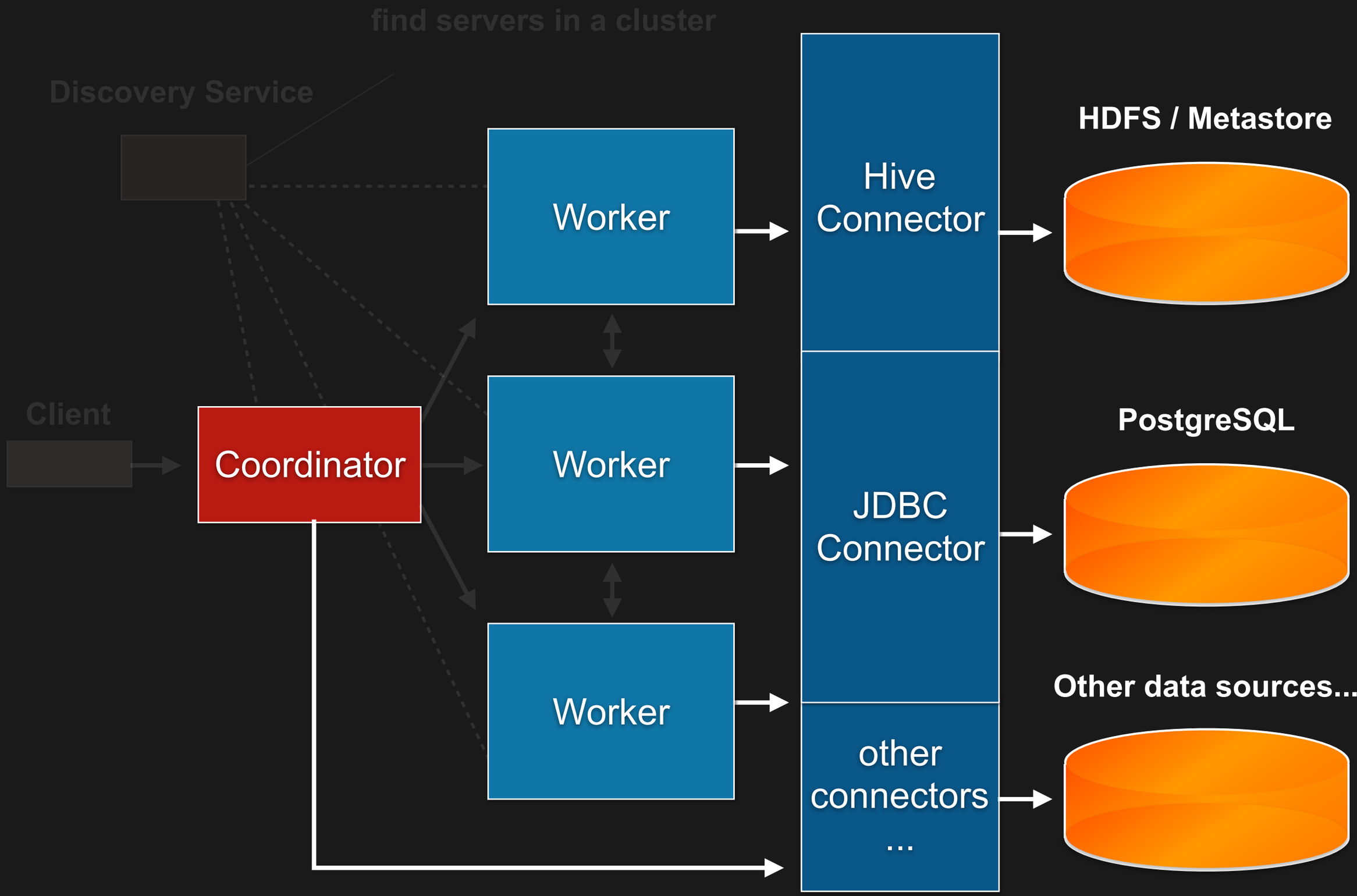

Discovery Service
Worker
Storage / Metadata
Client
Coordinator
Worker
Connector Plugin
Worker
7. Client gets the result from a worker


Discovery Service
Client
Coordinator
Worker
Worker
Worker
Connector Plugin
Storage / Metadata

What's Connectors?

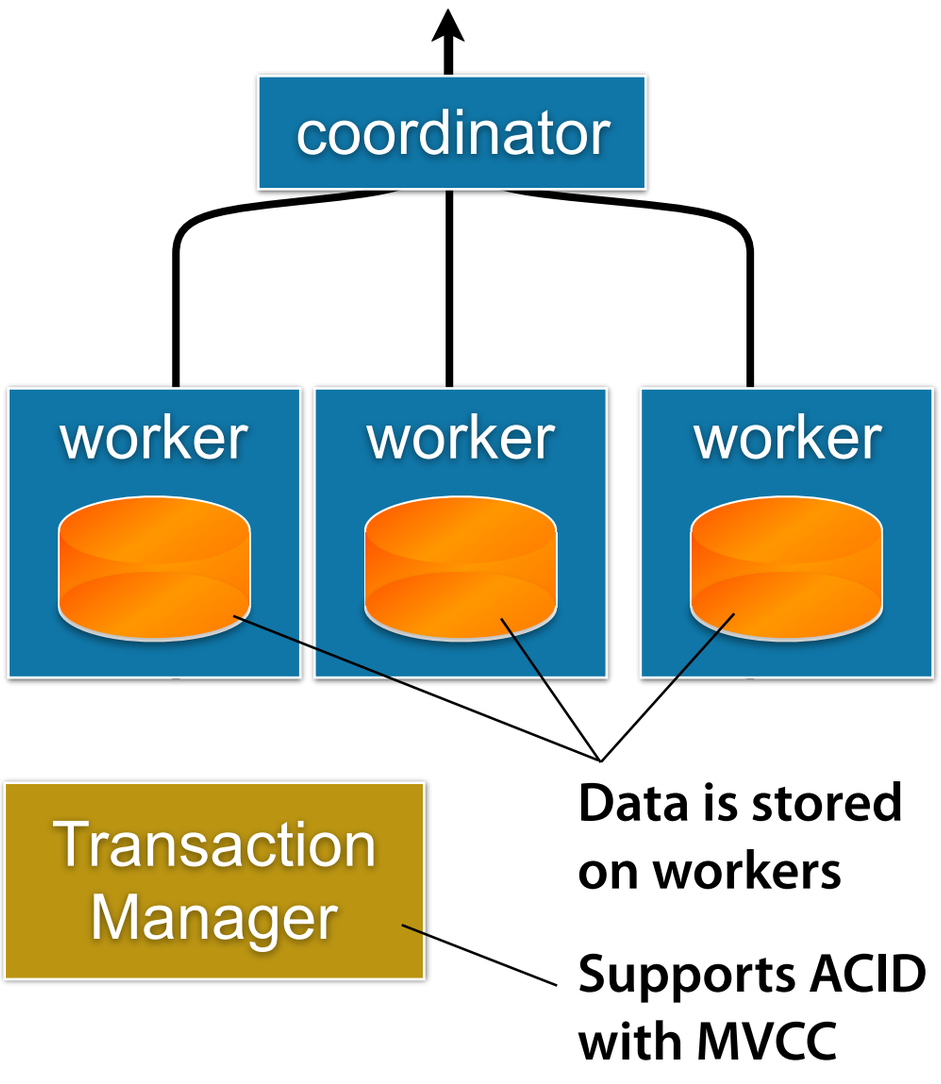
- **Connectors are plugins of Presto**
- **Connectors provide metadata and data to Presto**
  - provide table schema to coordinators
  - provide table rows to workers
- **Implementations:**
  - Hive connector
  - Cassandra connector
  - JDBC connector (scans from RDBMS)
  - Kafka connector, etc.

Multiple connectors in a query

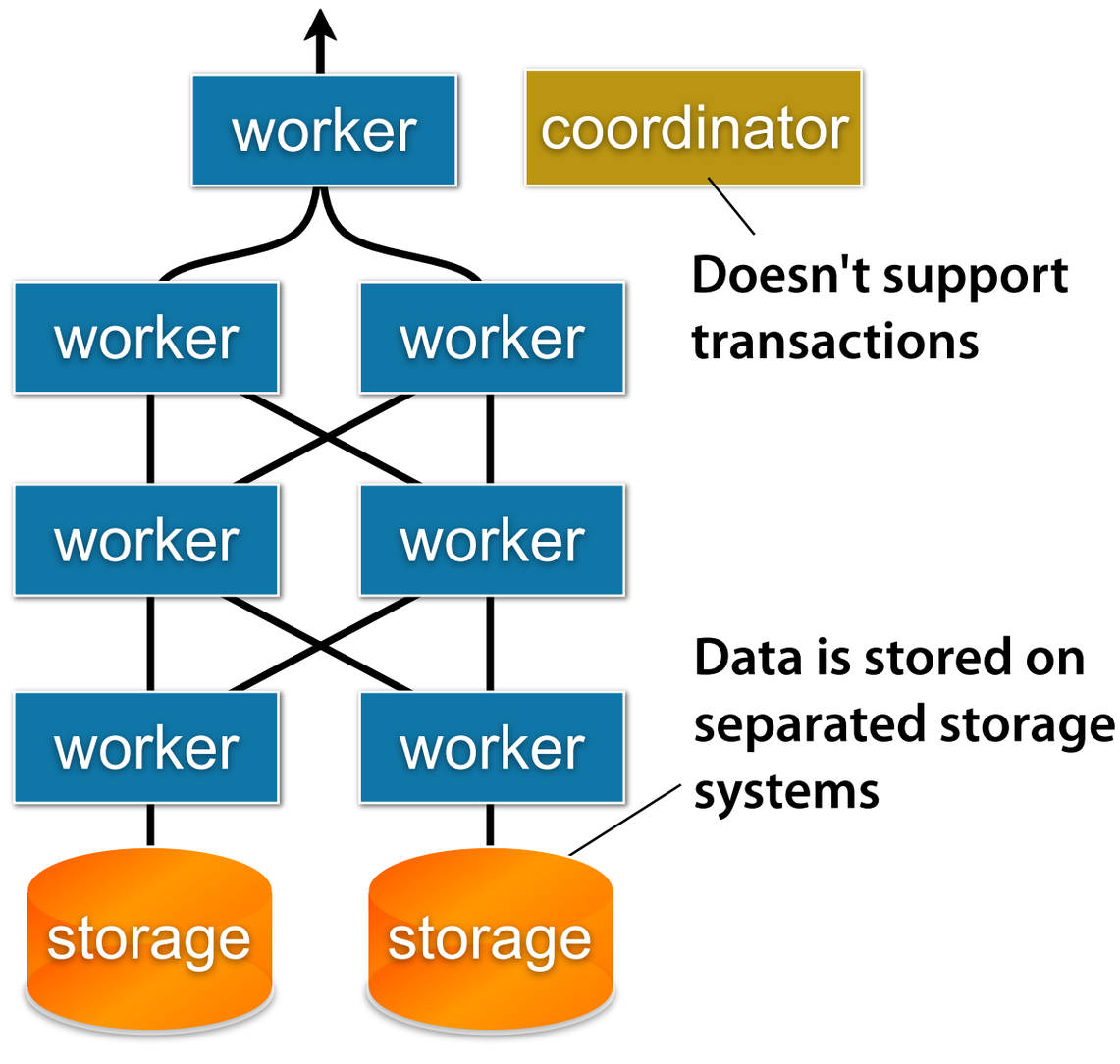


Postgres-XL vs. Presto

Postgres-XL

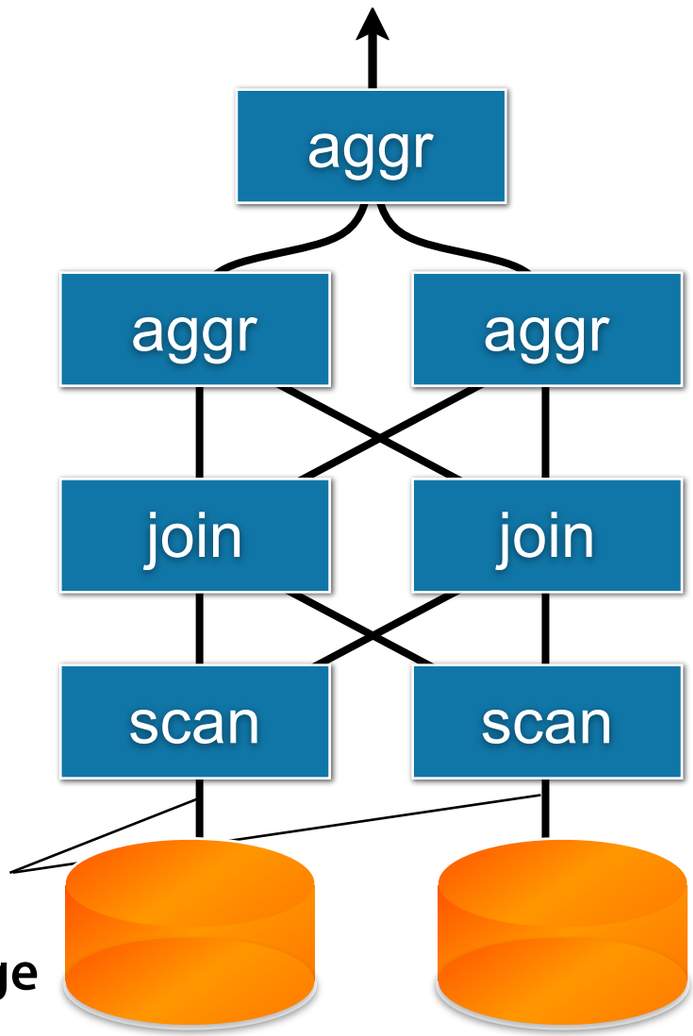


Presto



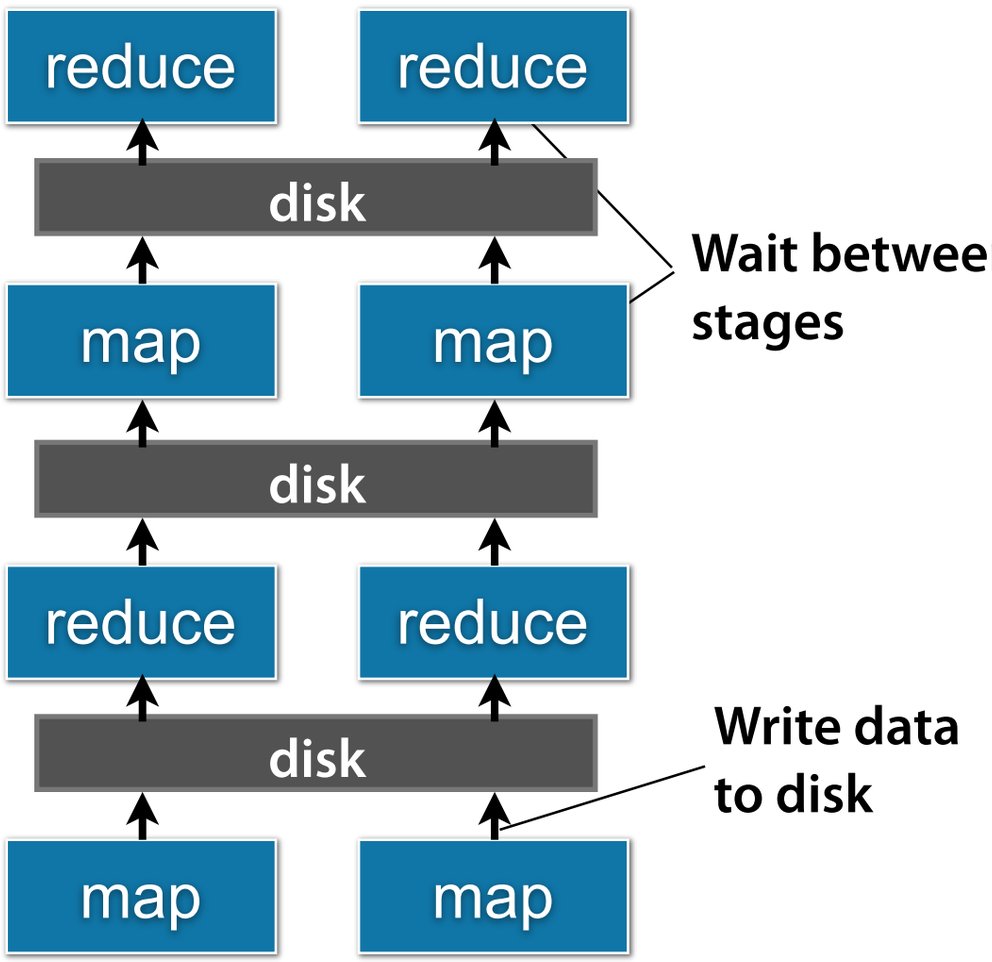
Q. Why do you choose Presto over other databases?

- **A. Because Presto is elastic.**
  - Computation performance is isolated from storage management.
    - Adding a server improves performance instantly. (No data re-distribution when we add a server)
    - Removing server is also done instantly.
  - That's good for cloud-based infrastracture.
    - Scale performance when we need.
    - JOIN across multiple data sources (RDB, S3, etc.) without moving big data.

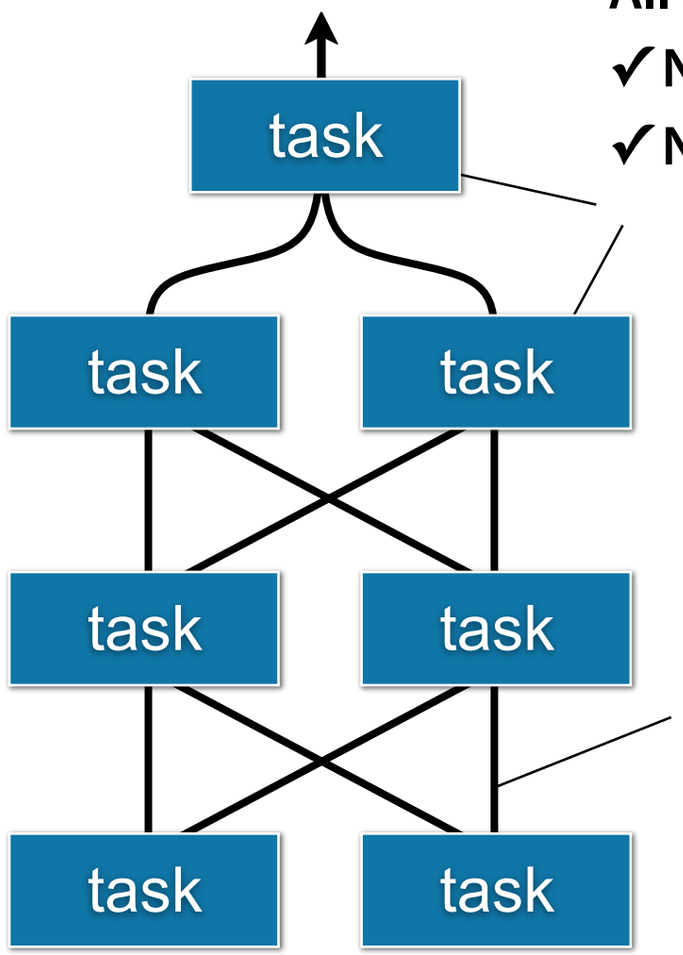


Hadoop MapReduce vs. Presto

MapReduce



Presto



Today’s talk

0. Overview of Presto & data analytics platform

1. Why Presto? - Presto's architecture

2. Prestogres design

3. Prestogres implementation

4. Prestogres hacks

5. Presto internals

2. Prestogres design

PostgreSQL protocol gateway

The problems to solve

- **BI tools need ODBC or JDBC connectivity.**
  - Tableau, IBM Cognos, QlickView, Chart.IO, ...
  - JasperSoft, Pentaho, MotionBoard, ...
- **ODBC/JDBC is VERY COMPLICATED.**
  - psqlODBC: 58,000 lines
  - postgresql-jdbc: 62,000 lines
  - mysql-connctor-odbc: 27,000 lines
  - mysql-connector-j: 101,000 lines
- **Open-source implementation will take long time.**

A solution

> Creates a **PostgreSQL protocol gateway server**

> Reuses PostgreSQL's **stable** ODBC / JDBC driver

Prestogres

PostgreSQL protocol gateway for Presto

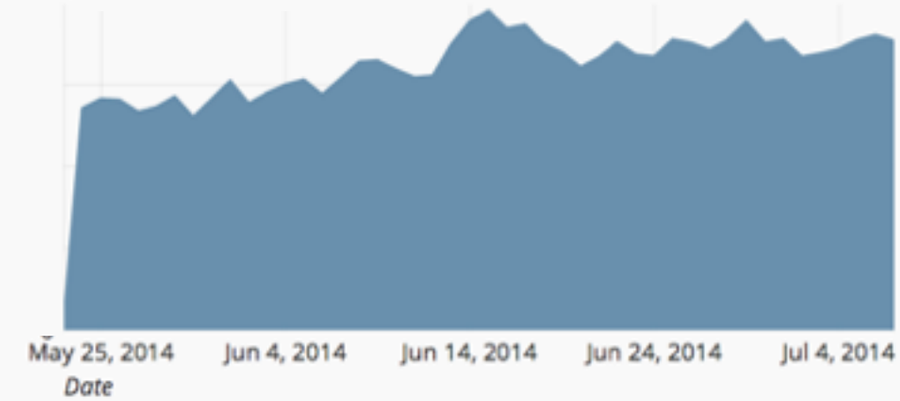
Treasure Data

Dashboards Settings Help

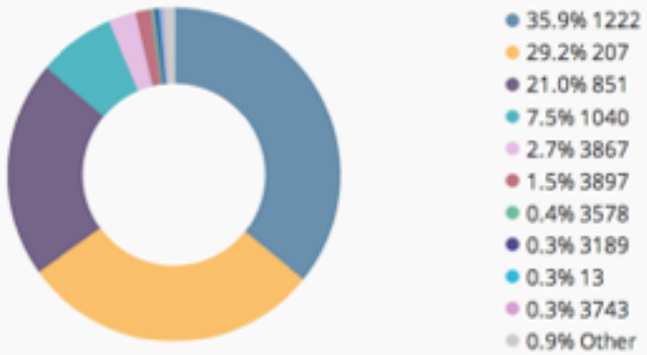
Example Presto Dashboard

Add Element Arrange Presen

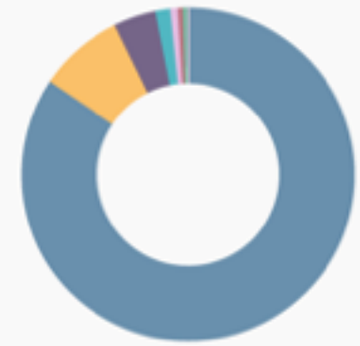
Daily



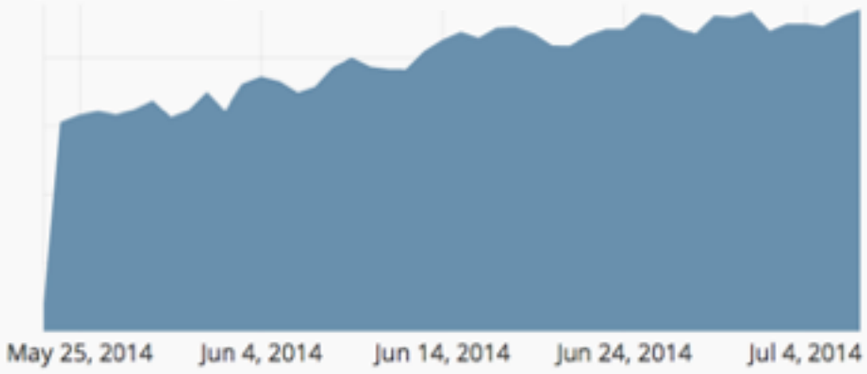
Query Ratio



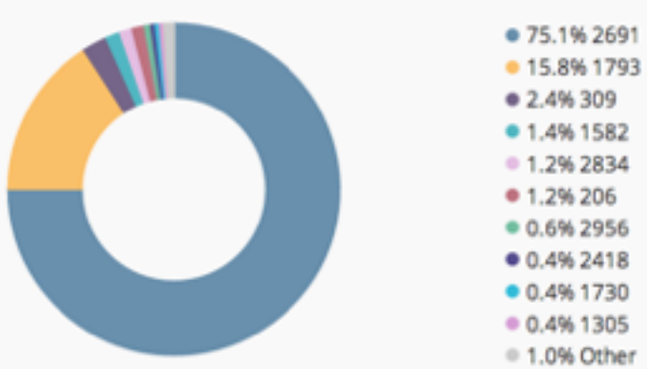
Query Ratio



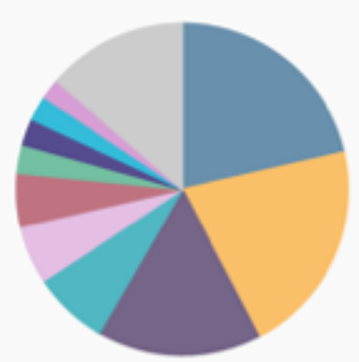
Daily Streaming



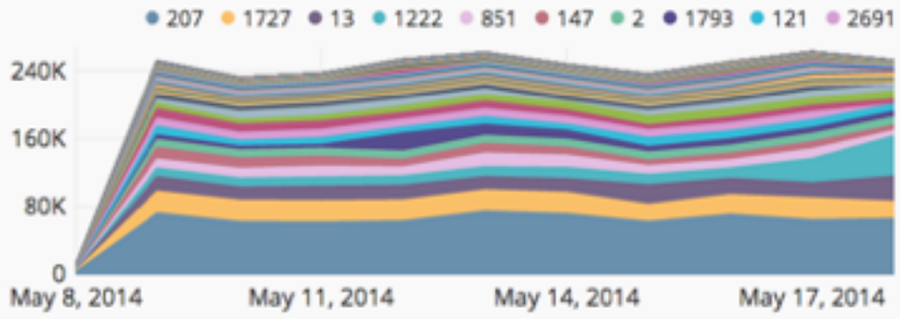
Job



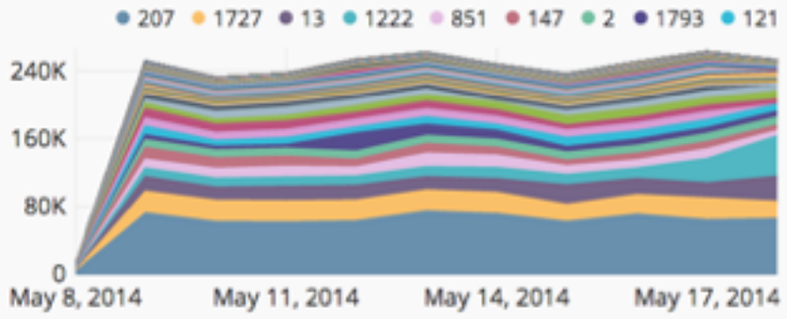
Import Counts



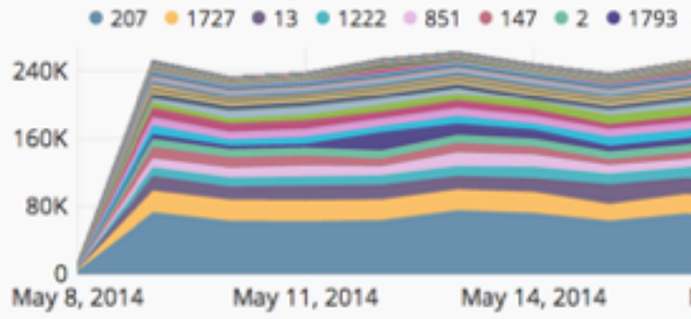
Hive



Hive Task



Hive Task Attempts



dashboard on chart.io: **https://chartio.com/**

Other possible designs were...

a) MySQL protocol + libdrizzle:

> Drizzle includes a well-designed library to implement **MySQL protocol server**.

> Proof-of-concept worked well:

- **trd-gateway** - MySQL protocol gateway server for "Hive"

> Difficulties: clients assumes the server is MySQL but,

- syntax is not ANSI standard: MySQL uses `…`, while Presto uses "…"
- function mismatches: DAYOFMONTH(…) vs EXTRACT(day…)

Other possible designs were...

b) PostgreSQL + Foreign Data Wrapper (FDW):

- JOIN and aggregation pushdown is not available (yet?)

Difficulties to implement PG protocol

- **Emulating system catalogs**
  - pg_class, pg_namespace, pg_proc, …
- **Rewriting transactions (BEGIN, COMMIT)**
  - Presto doesn’t support transactions

Prestogres design

pgpool-II + PostgreSQL + PL/Python

- pgpool-II is a PostgreSQL protocol middleware for replication, failover, load-balancing, etc.
- pgpool-II already implements useful utility functions (parsing SQL, rewriting SQL, hacking system catalogs, …)
- Basic idea:
  - Rewrite queries at pgpool-II and run Presto queries using PL/Python

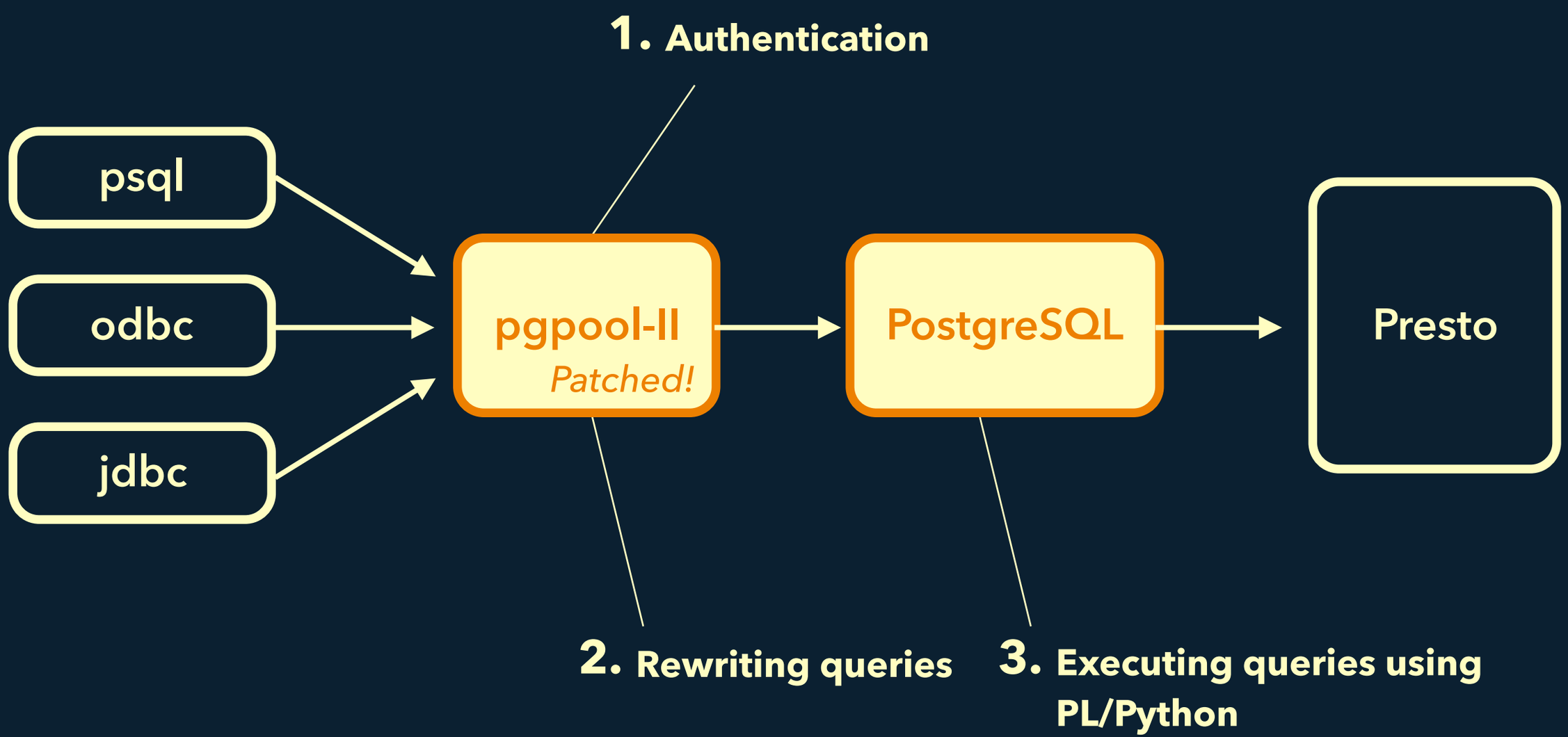
select count(*)
from access

rewrite! →

select * from
python_func('select count(*) from access')

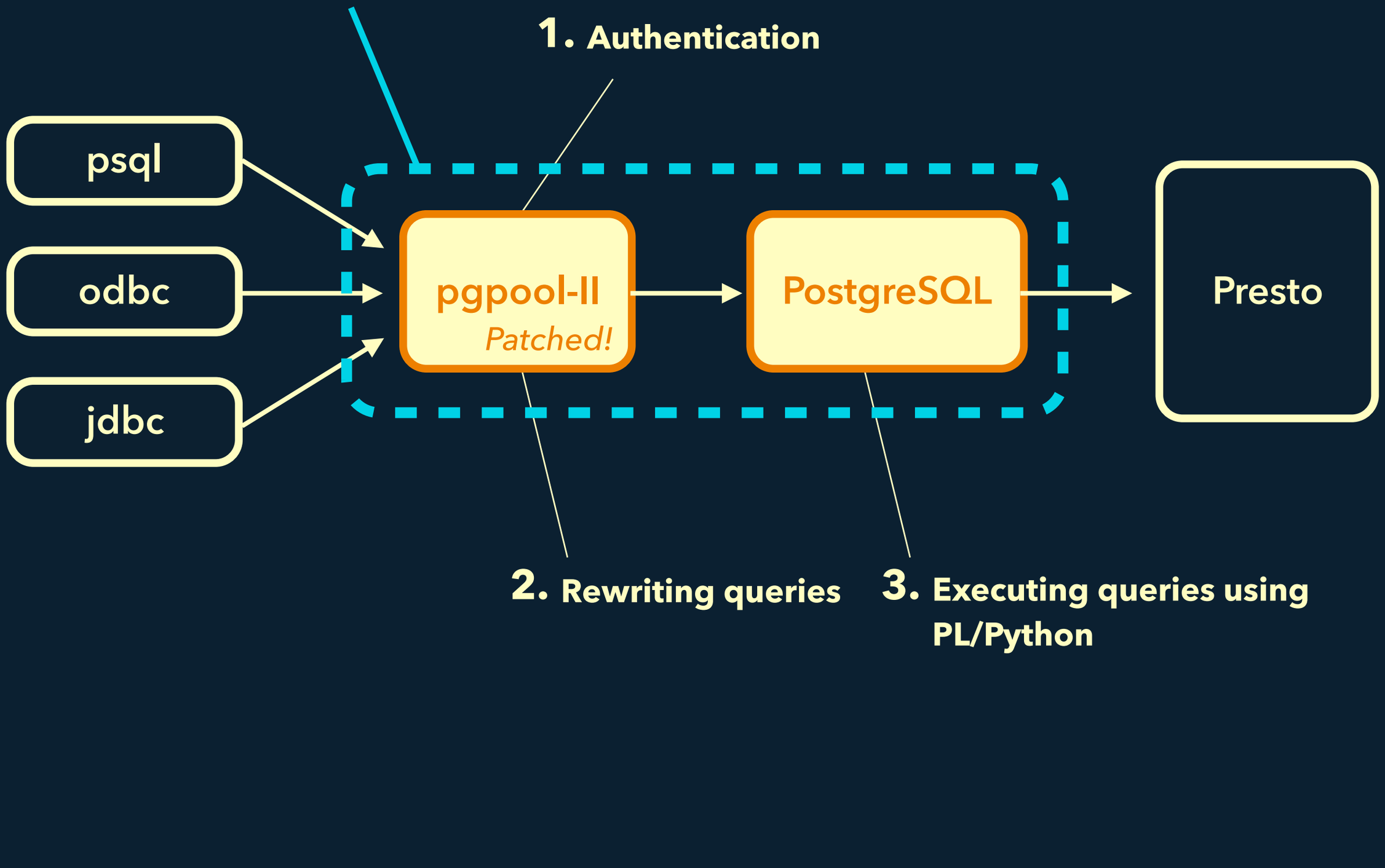
3. Prestogres implementation

Overview

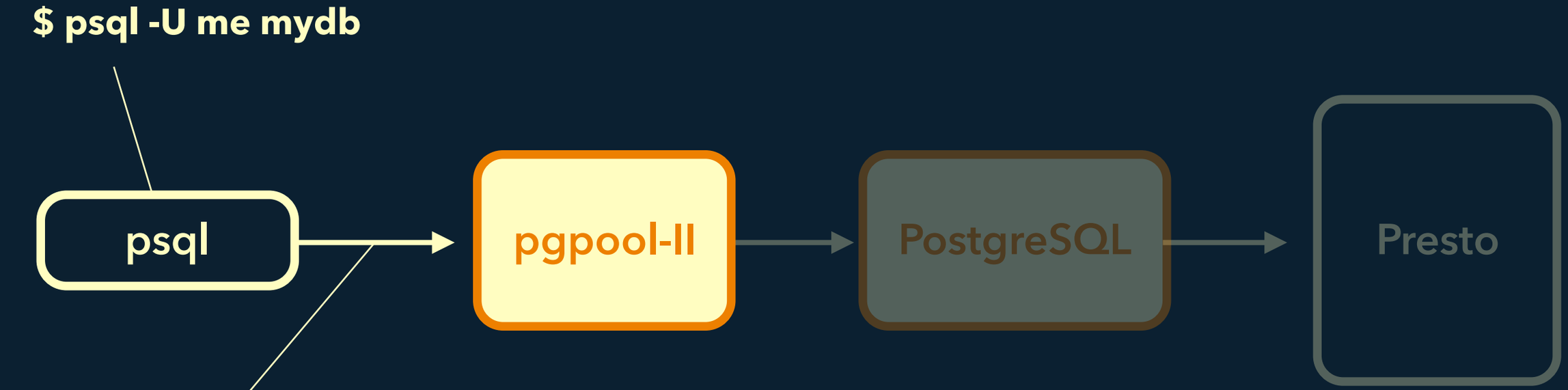

1. Authentication
psql
odbc
jdbc
pgpool-II
Patched!
PostgreSQL
Presto
2. Rewriting queries
3. Executing queries using PL/Python

Overview

Prestogres



Connection



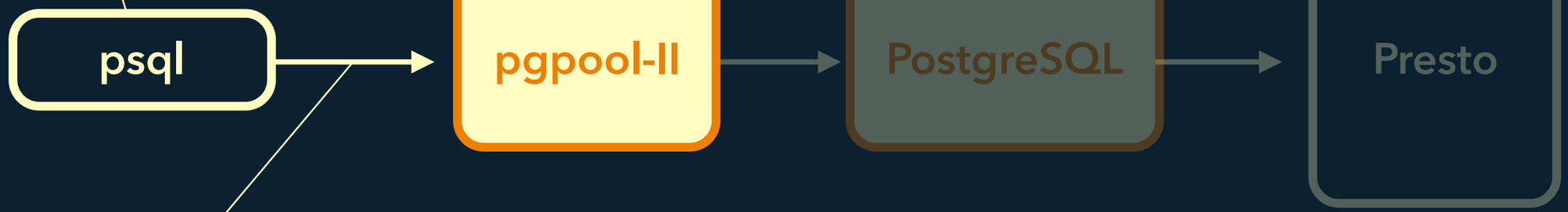

```
StartupPacket  {
                 database = “mydb”,
                 user = “me”,
                 …
               }
```

Connection

prestogres_hba.conf

```
host mydb me 0.0.0.0/0 trust
  presto_server presto.local:8080,
  presto_catalog hive,
  pg_database hive
```

```
$ psql -U me mydb
```



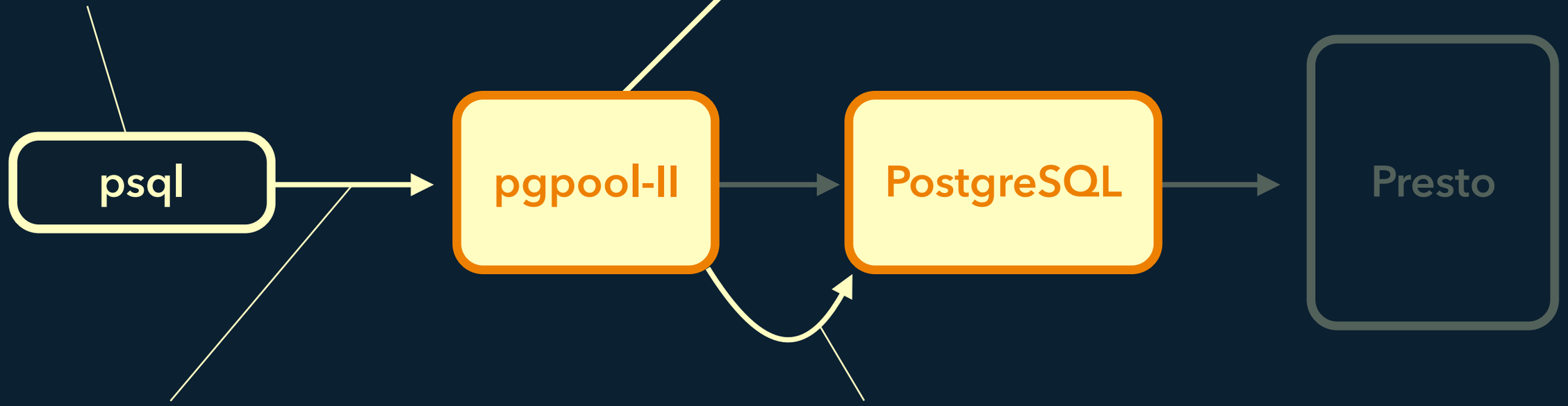

```
StartupPacket  {
                 database = “mydb”,
                 user = “me”,
                 …
               }
```

Connection

prestogres_hba.conf

```
host mydb me 0.0.0.0/0 trust
  presto_server presto.local:8080,
  presto_catalog hive,
  pg_database hive
```

```
$ psql -U me mydb
```




```
StartupPacket  {
                 database = “mydb”,
                 user = “me”,
                 …
               }
```

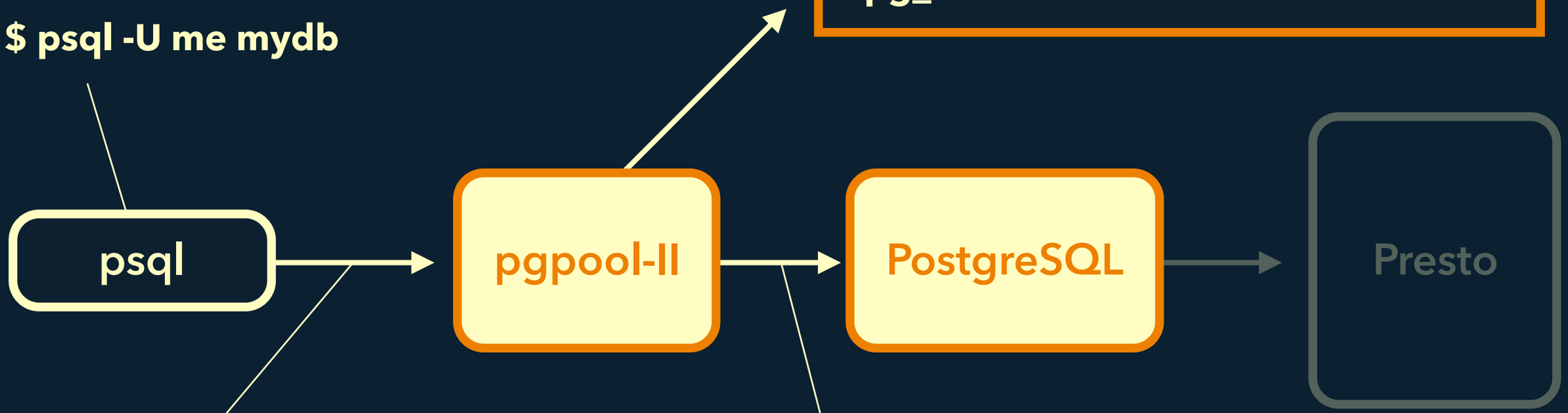
```
libpq host=‘localhost’, dbname=‘postgres’,
       user=‘prestogres’
```

```
> CREATE DATABASE hive;
> CREATE ROLE me;
> CREATE FUNCTION setup_system_catalog;
> CREATE FUNCTION start_presto_query;
```

Connection

prestogres_hba.conf

```
host mydb me 0.0.0.0/0 trust
  presto_server presto.local:8080,
  presto_catalog hive,
  pg_database hive
```



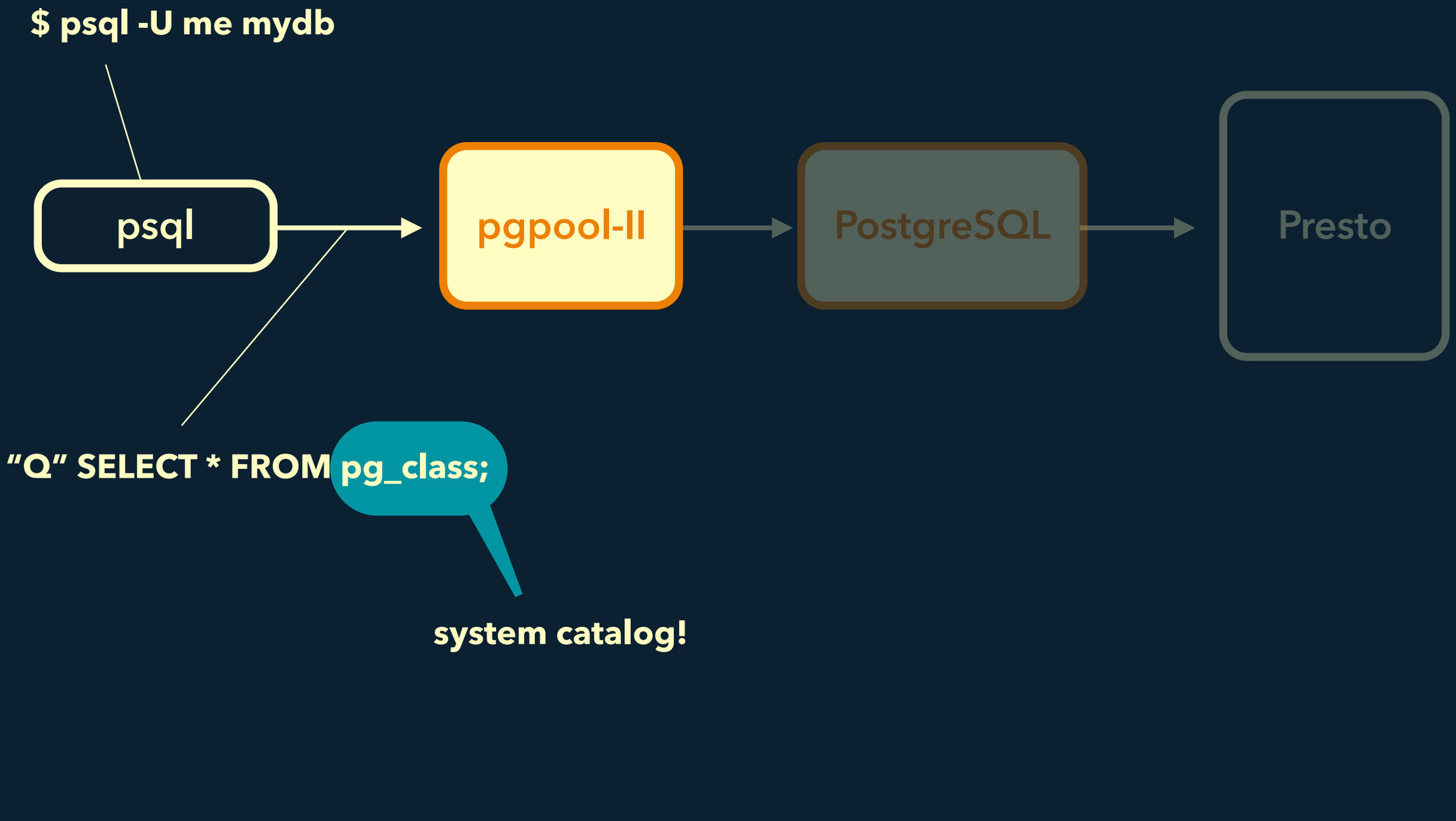

```
StartupPacket  {
                 database = “mydb”,
                 user = “me”,
                 …
               }
```

```
StartupPacket  {
                 database = “hive”,
                 user = “me”,
                 …
               }
```

uses the database and user
which were created right now!

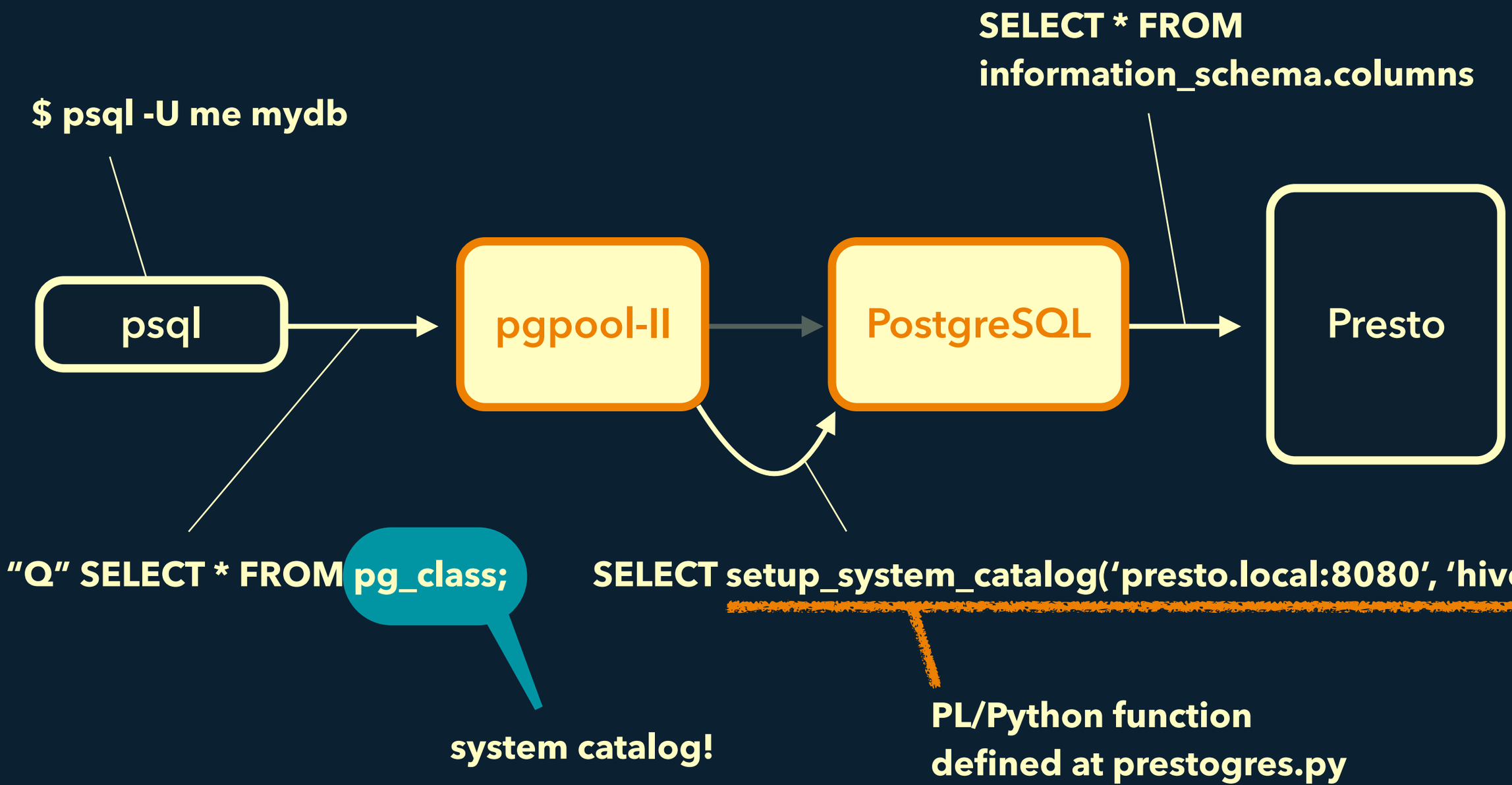
Meta-query

"Query against a system catalog!”



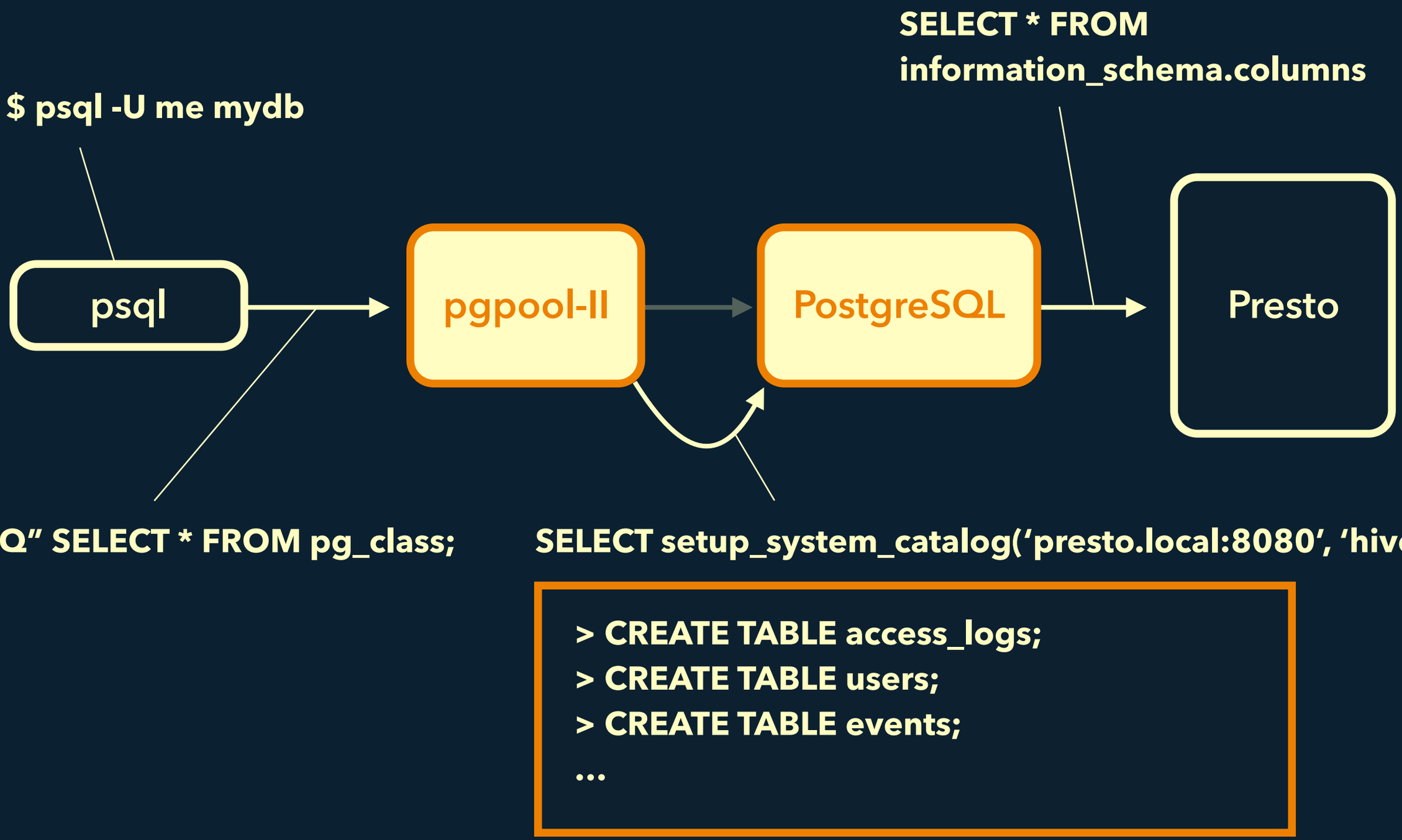
Meta-query

"Query against a system catalog!”



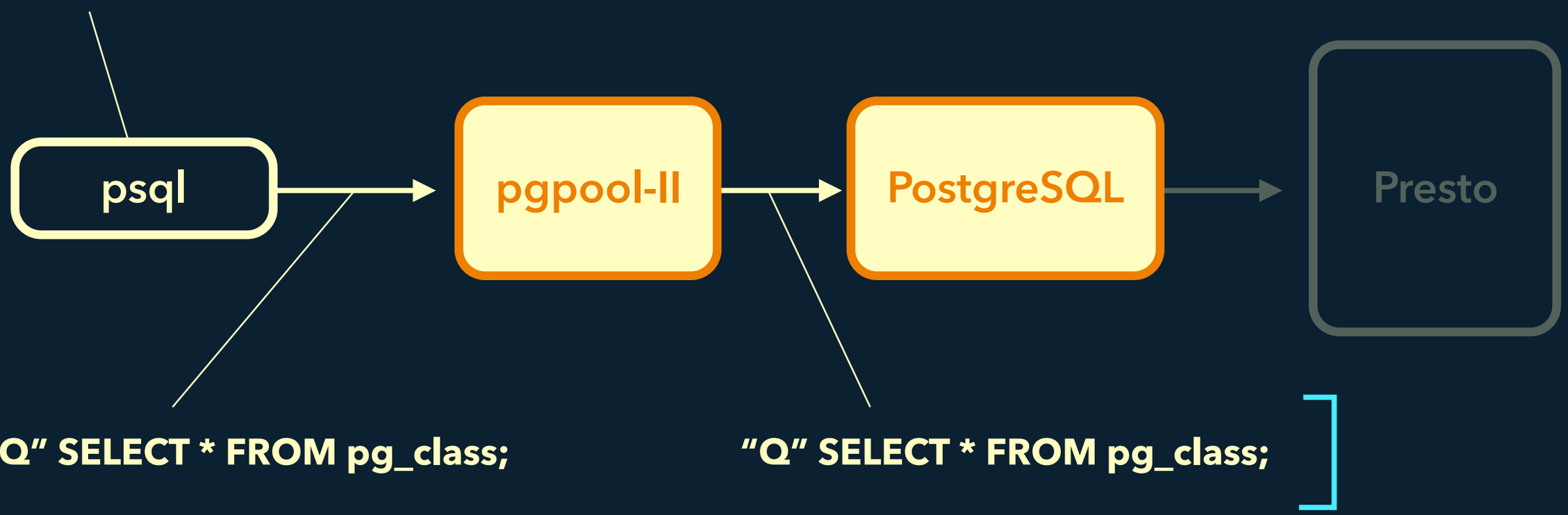
Meta-query

"Query against a system catalog!”



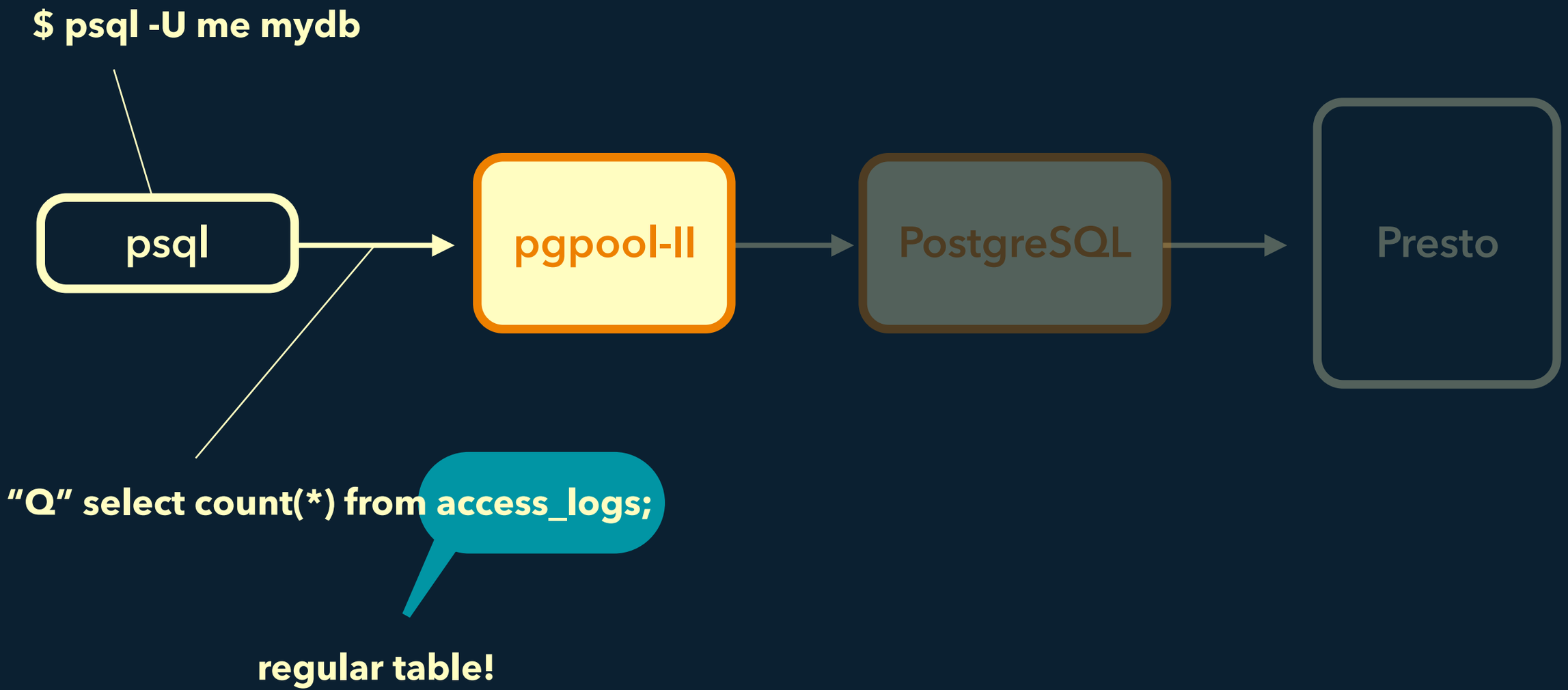
Meta-query

"Query against a system catalog!”



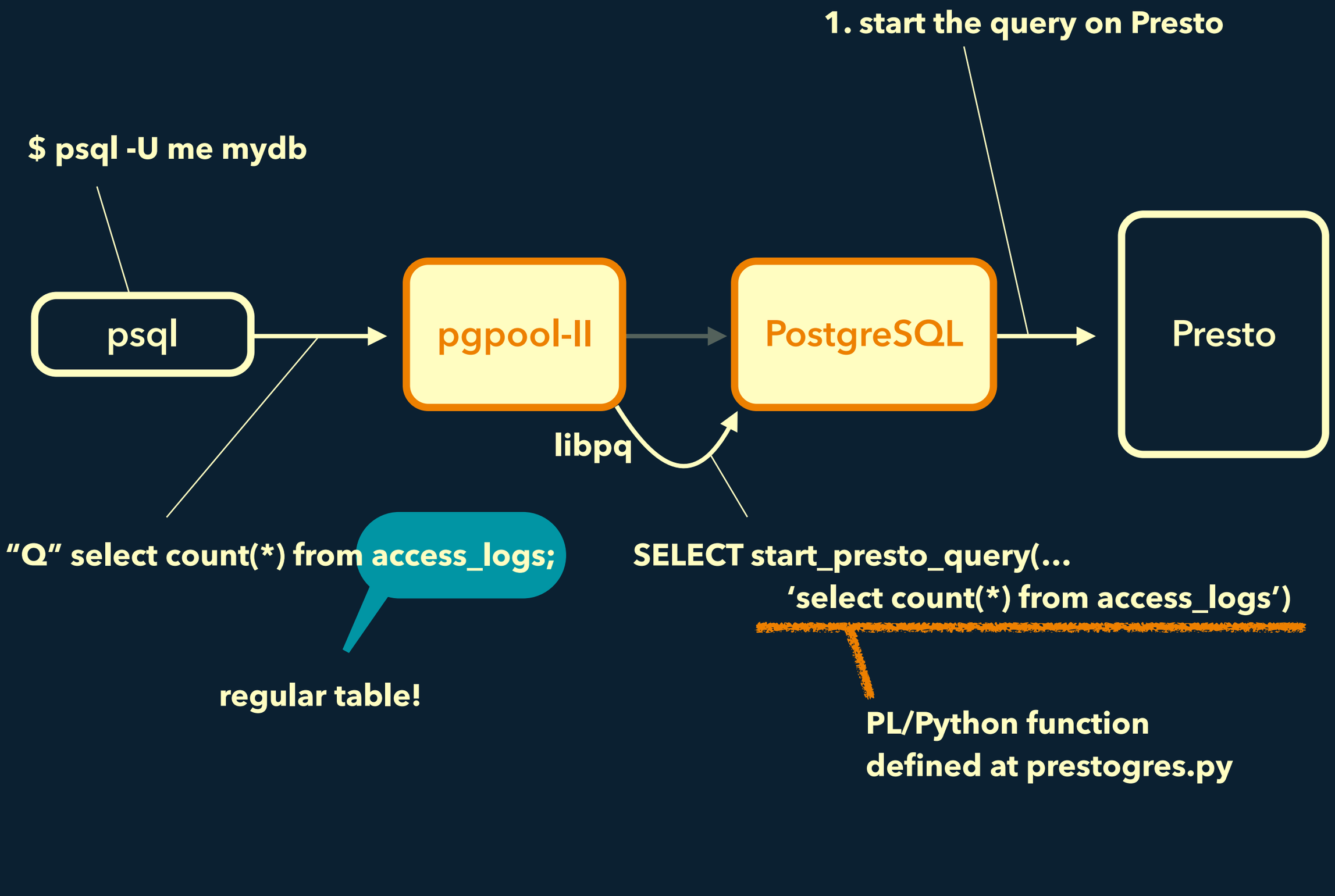
Presto Query

"Query against a regular table!"



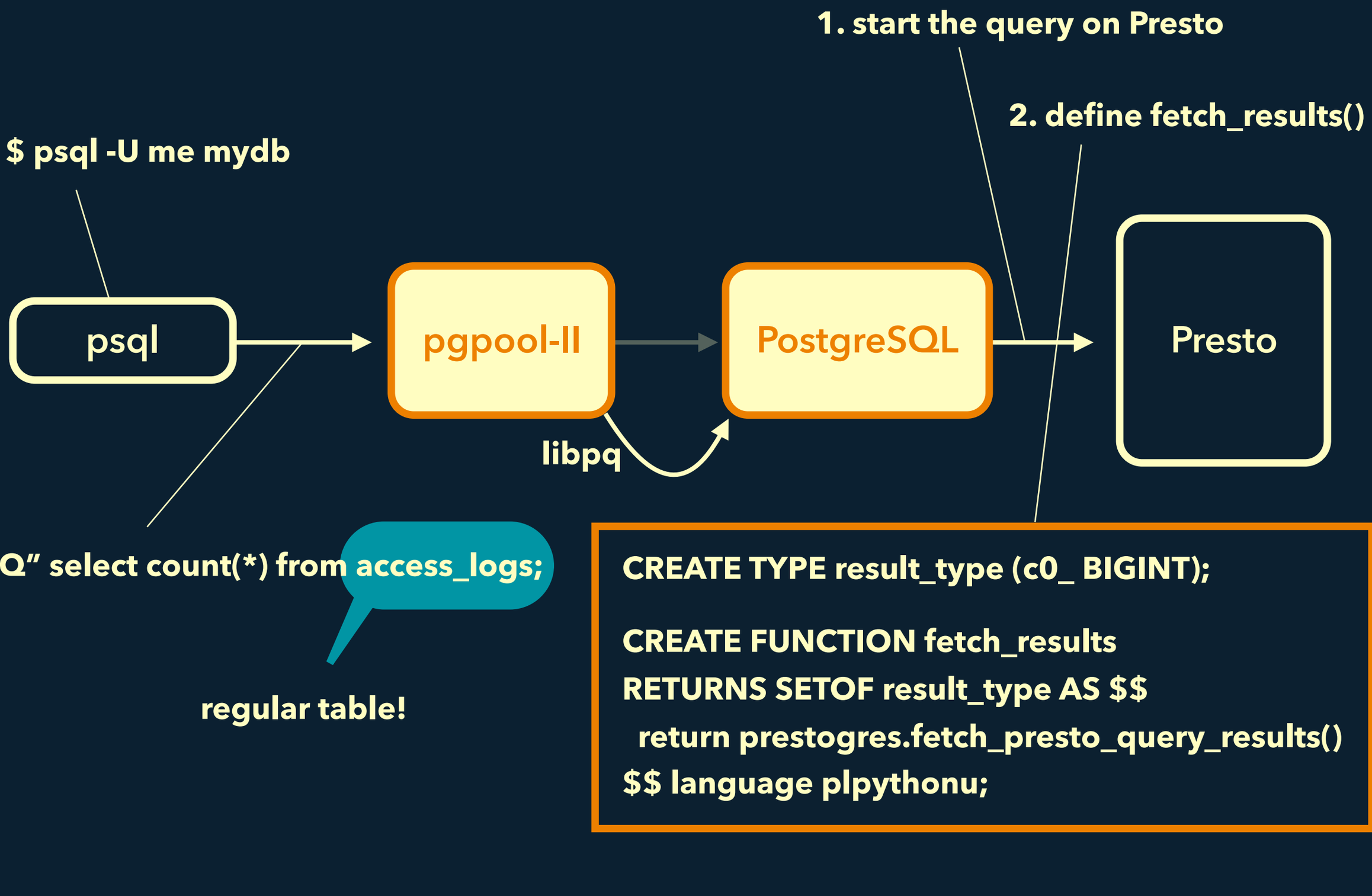
Presto Query

"Query against a regular table!"



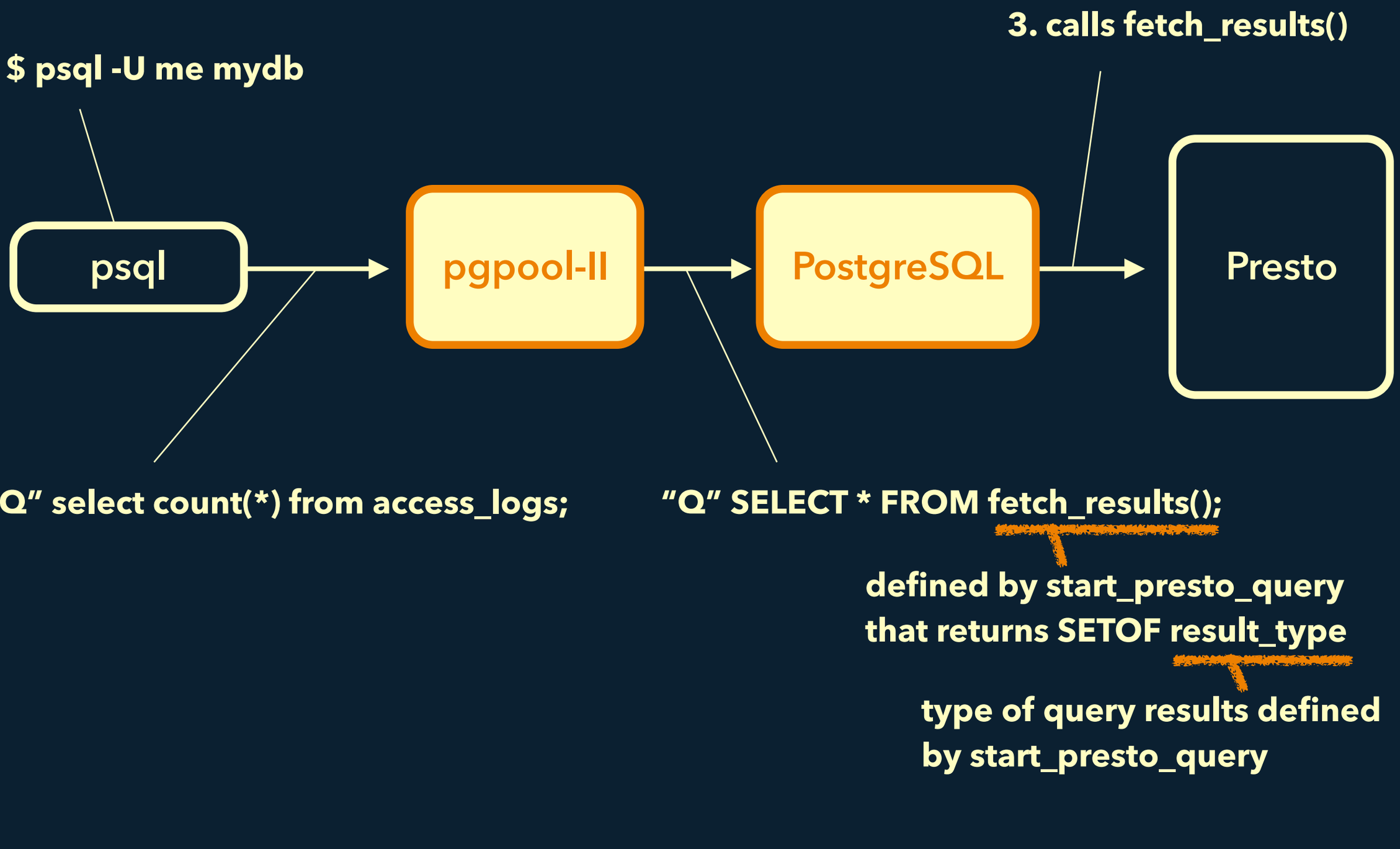
Presto Query

"Query against a regular table!"



Presto Query

"Query against a regular table!"



Examples

- **select * from pg_class**
  - In another connection, pgpool-II runs setup_system_catalog()
  - Then forwards query: select * from pg_class
- **select count(*) from access**
  - In another connection, pgpool-II runs start_presto_query('select count(*) from access', …)
  - Then forwards query: select * from fetch_query_result()
- **BEGIN**
  - Forwards query: BEGIN (no rewrite)

Demo

4. Prestogres hacks

Multi-statement queries

- **BEGIN; select count(*) from access; COMMIT**
  - Parse query in pgpool-II
  - In anothe connection, call start_presto_query('select …')
  - Rewrite query partially:

    BEGIN; select * from fetch_query_result(); COMMIT
- **select count(*) from access; select count(*) from access**
  - not supported :(

Supporting Cursors

- **DECLARE CURSOR xyz FOR select …; FETCH**
  - Parse query in pgpool-II
  - In anothe connection, call start_presto_query('select …')
  - Rewrite query partially:

    DECLARE CURSOR xyz FOR
    select * from fetch_query_result(); FETCH

Error handling

- **select xyz(*) from abc**

```
> do $$
   RAISE EXCEPTION ‘%’, ’Function xyz is not defined’
   USING errcode='42601'
$$
end language plpgsql
```

Faked current_database()

```
DELETE FROM pg_catalog.pg_proc
 WHERE proname='current_database';
CREATE FUNCTION pg_catalog.current_database()
RETURNS name AS $$
begin
   return 'faked_name'::name;
end
$$ language plpgsql stable strict;
```

5. Future works

Future works

Rewriting CAST syntax

Extended query

CREATE TEMP TABLE

Thank you!

Prestogres

PostgreSQL protocol gateway for Presto

https://github.com/treasure-data/prestogres
licensed under Apache License.

Sadayuki Furuhashi

Treasure Data, Inc.
Founder & Software Architect