Chương 4

Cơ sở dữ liệu phi quan hệ NoSQL - phần 3

Xử lý truy vấn SQL cho dữ liệu lớn

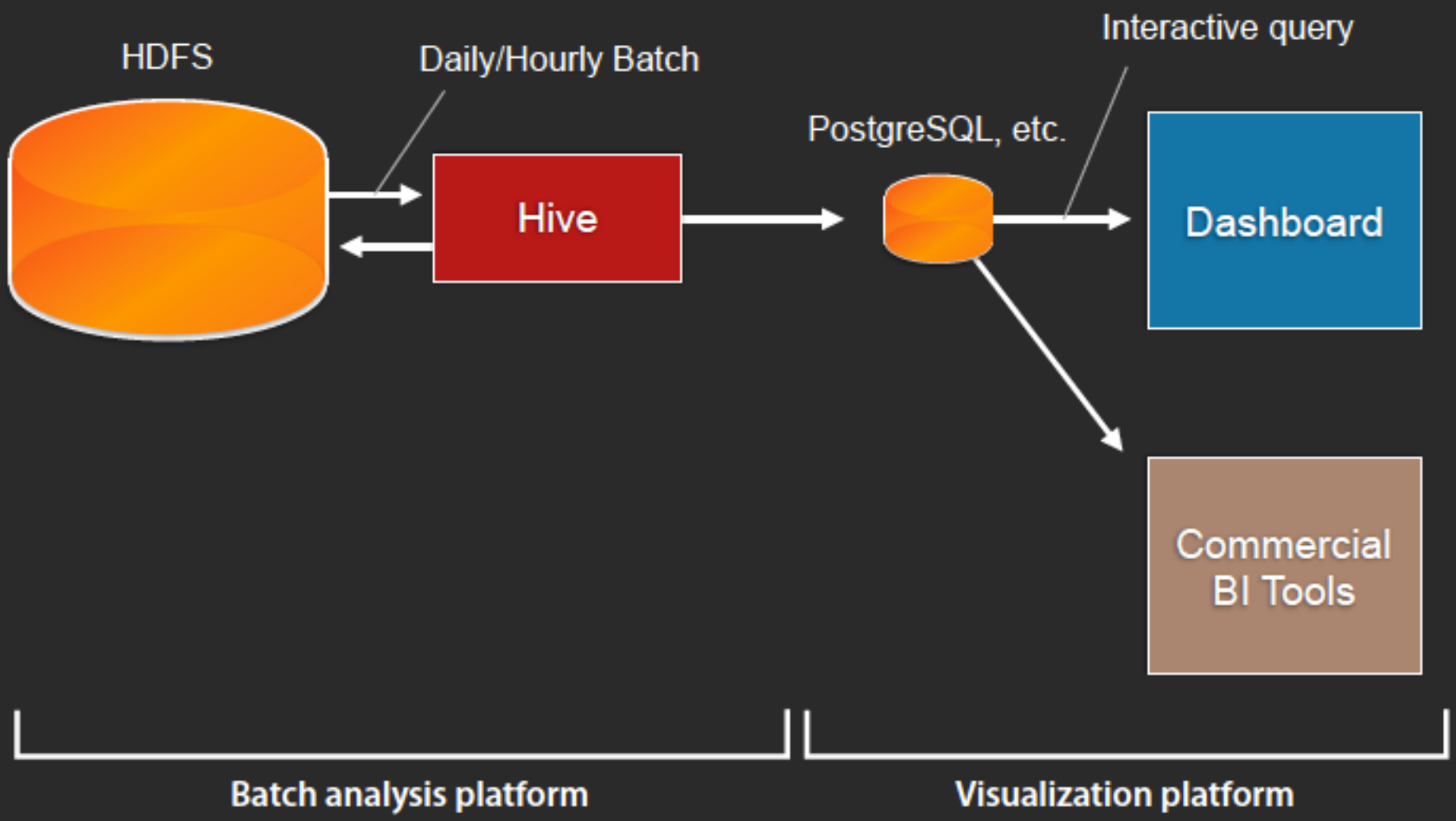
History

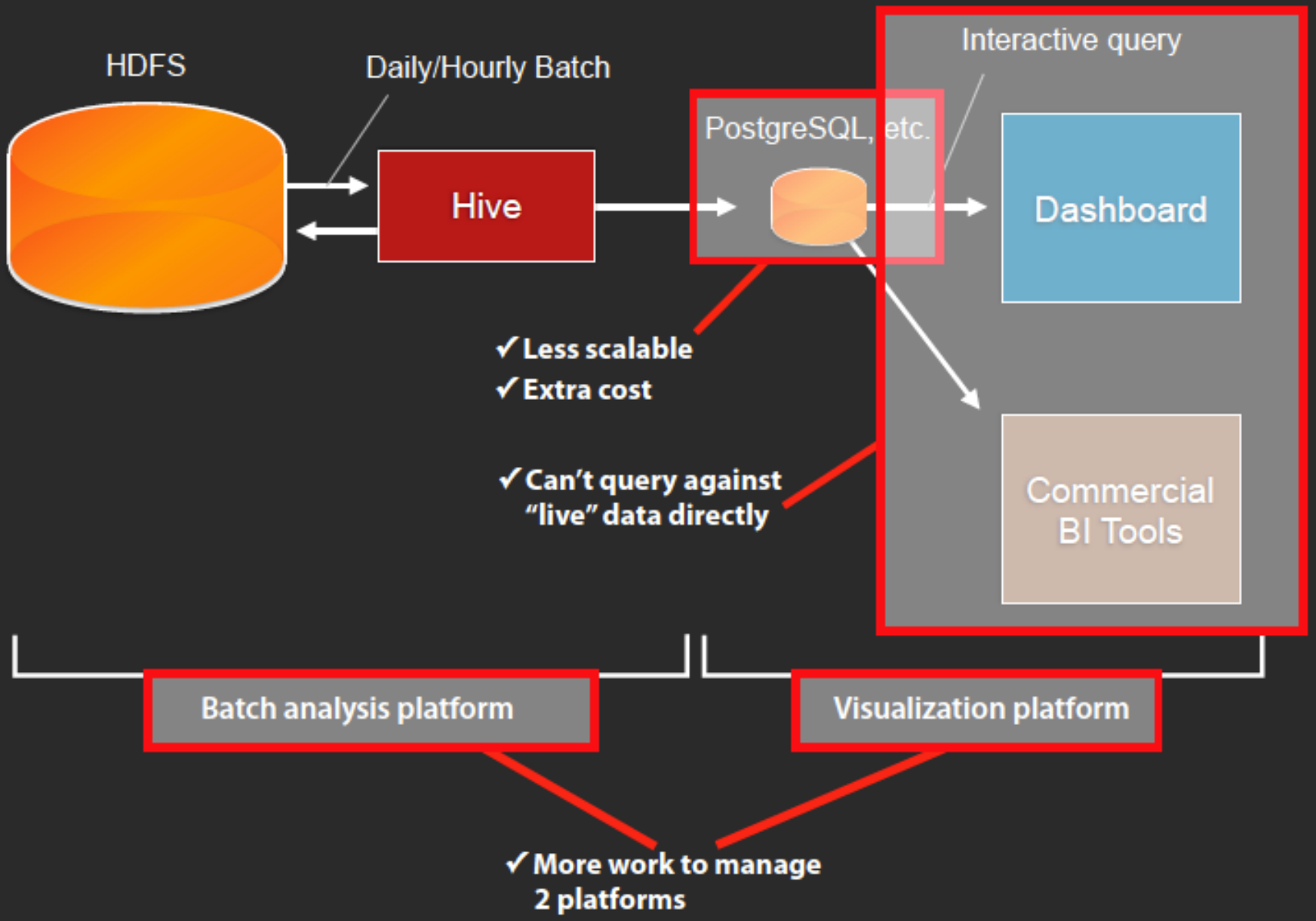
- 2012 Fall: Project started at Facebook
  - Designed for interactive query
  - with speed of commercial data warehouse
  - and scalability to the size of Facebook
- 2013 Winter: Open sourced
  - 30+ contributes in 6 months
  - including people from outside of Facebook
- 2019: 300+ contributors

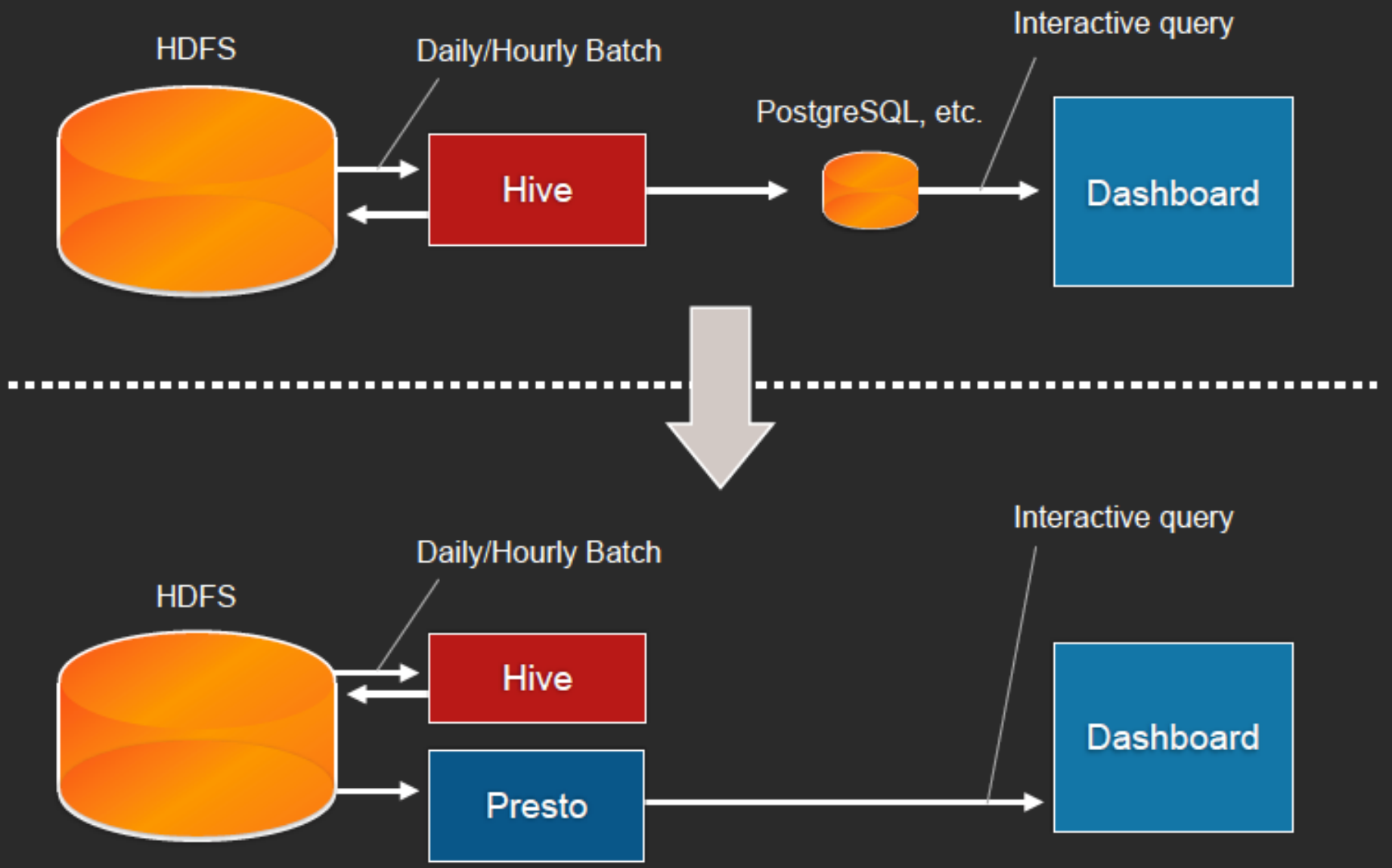
Motivation

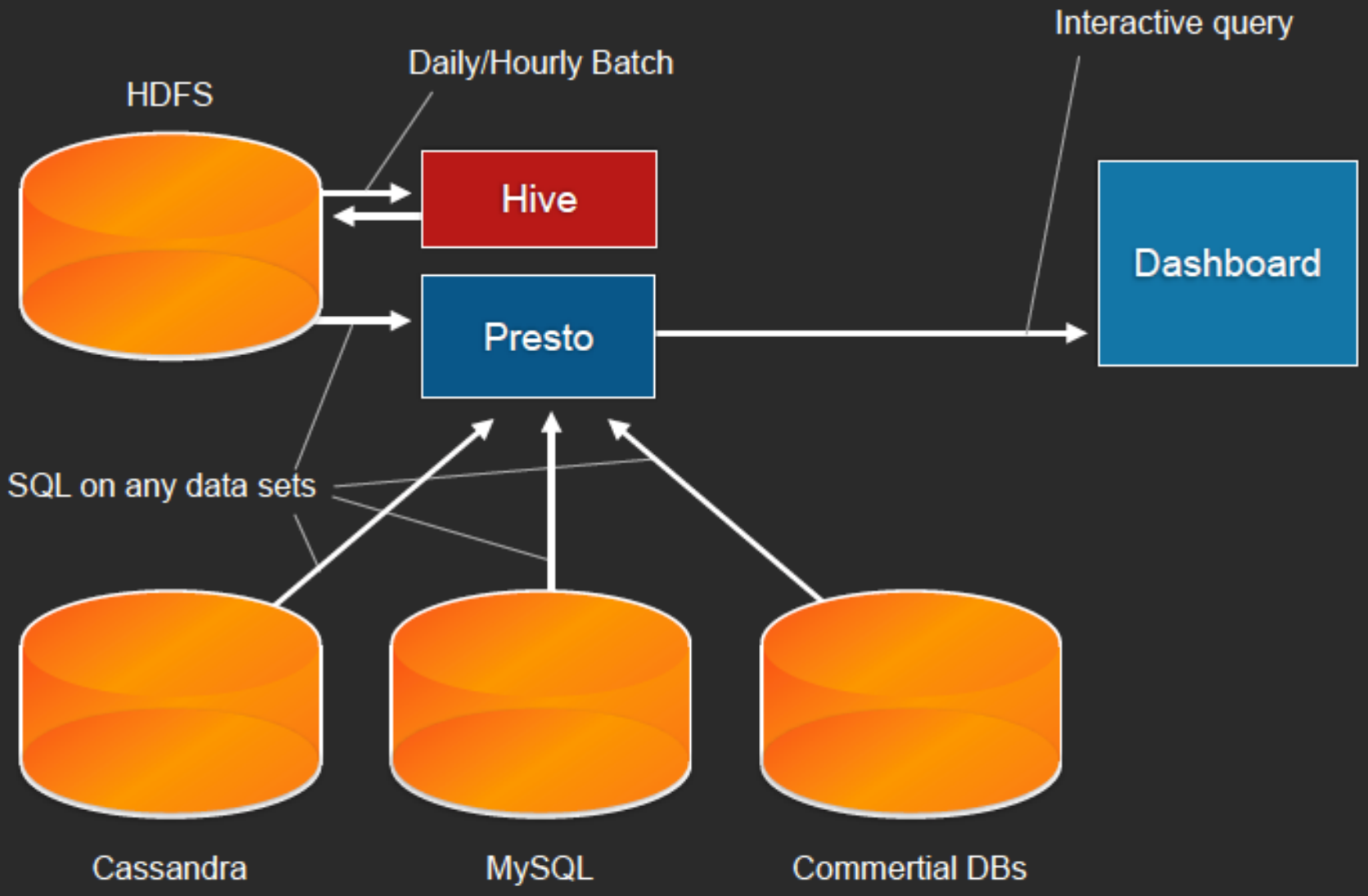
- We couldn’t visualize data in HDFS directly using dashboards or BI tools
  - because Hive is too slow (not interactive)
  - or ODBC connectivity is unavailable/unstable
- We needed to store daily-batch results to an interactive DB for quick response (PostgreSQL, Redshift, etc.)
  - Interactive DB costs more and less scalable by far
- Some data are not stored in HDFS
  - We need to copy the data into HDFS to analyze

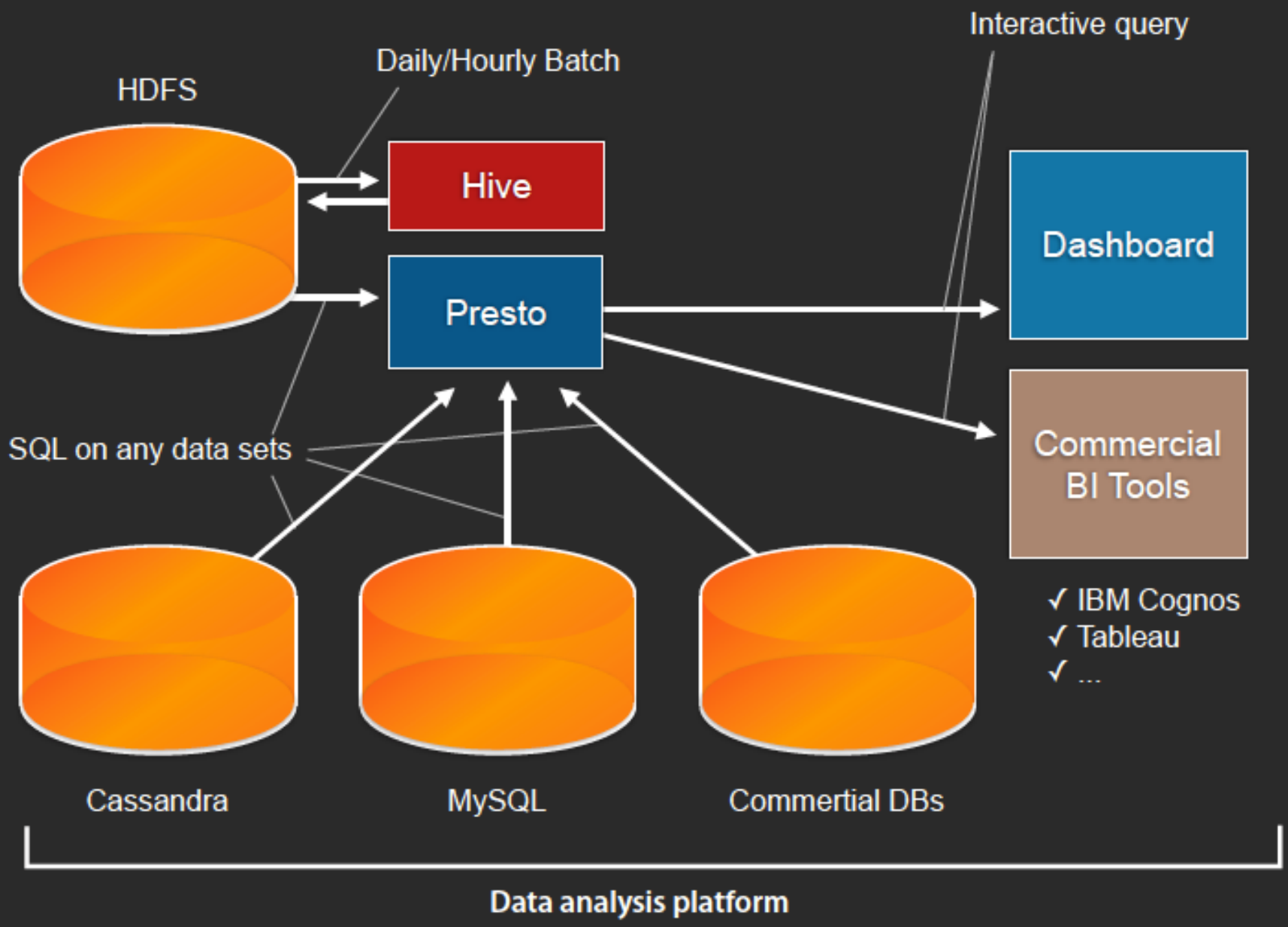
ability to quickly and easily extract insights from large amounts of data


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
✓Less scalable
✓Extra cost
✓Can't query against "live" data directly
Batch analysis platform
Visualization platform
✓More work to manage 2 platforms


HDFS
Daily/Hourly Batch
Hive
PostgreSQL, etc.
Interactive query
Dashboard
HDFS
Daily/Hourly Batch
Hive
Presto
Interactive query
Dashboard


Interactive query
Daily/Hourly Batch
HDFS
Hive
Dashboard
Presto
SQL on any data sets
Cassandra
MySQL
Commertial DBs


Interactive query
Daily/Hourly Batch
HDFS
Hive
Presto
Dashboard
Commercial
BI Tools
SQL on any data sets
✓ IBM Cognos
✓ Tableau
✓ ...
Cassandra
MySQL
Commertial DBs
Data analysis platform

What can Presto do?

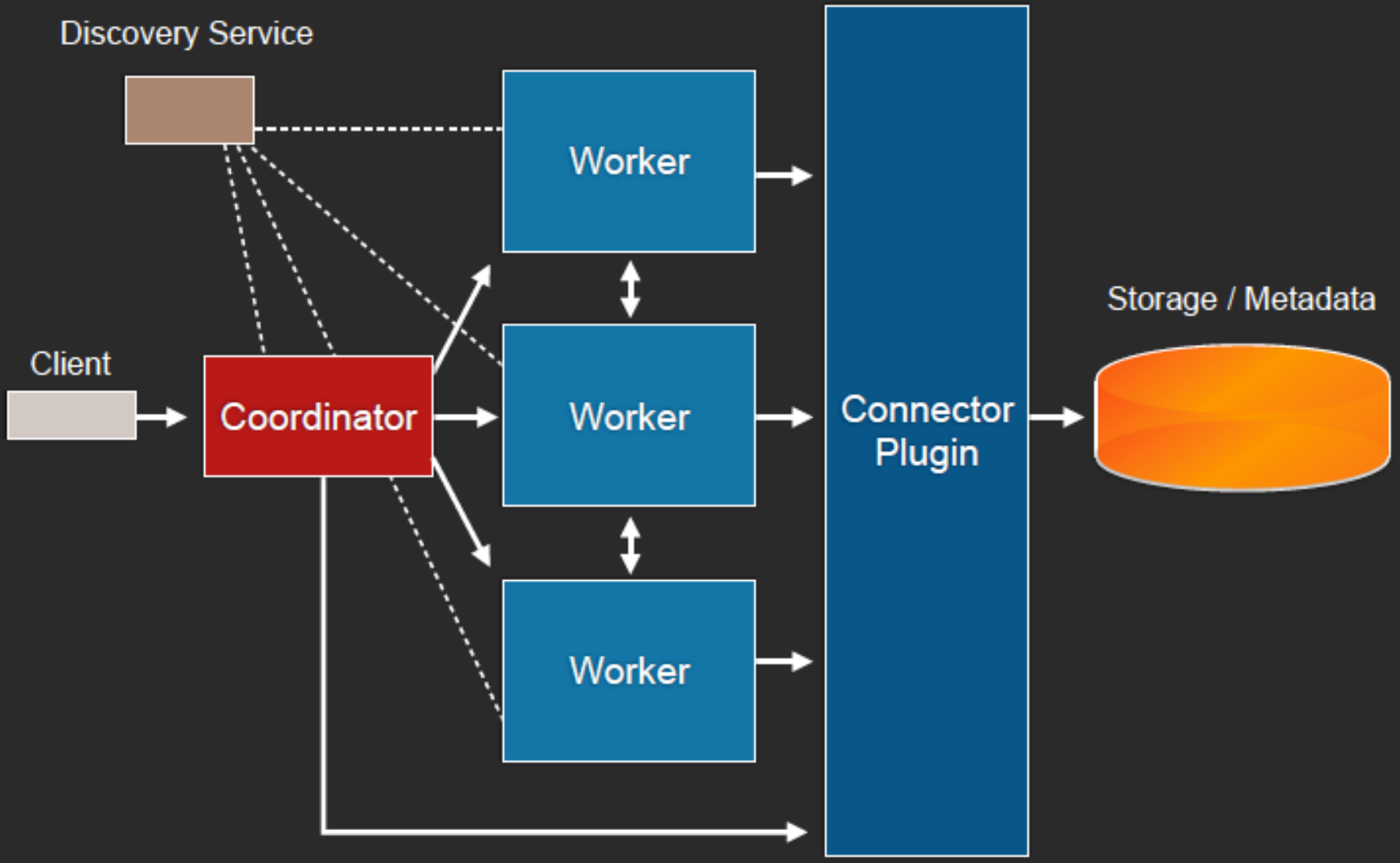
- Open-source distributed SQL query engine that has run in production at Facebook since 2013
  - ANSI SQL interface
- Query interactively (in milli-seconds to minues)
  - MapReduce and Hive are still necessary for ETL
- Query using commercial BI tools or dashboards
  - Reliable ODBC/JDBC connectivity
- Query across multiple data sources such as Hive, HBase, Cassandra, or even commertial DBs
  - Plugin mechanism
- Integrate batch analisys + visualization into a single data analysis platform

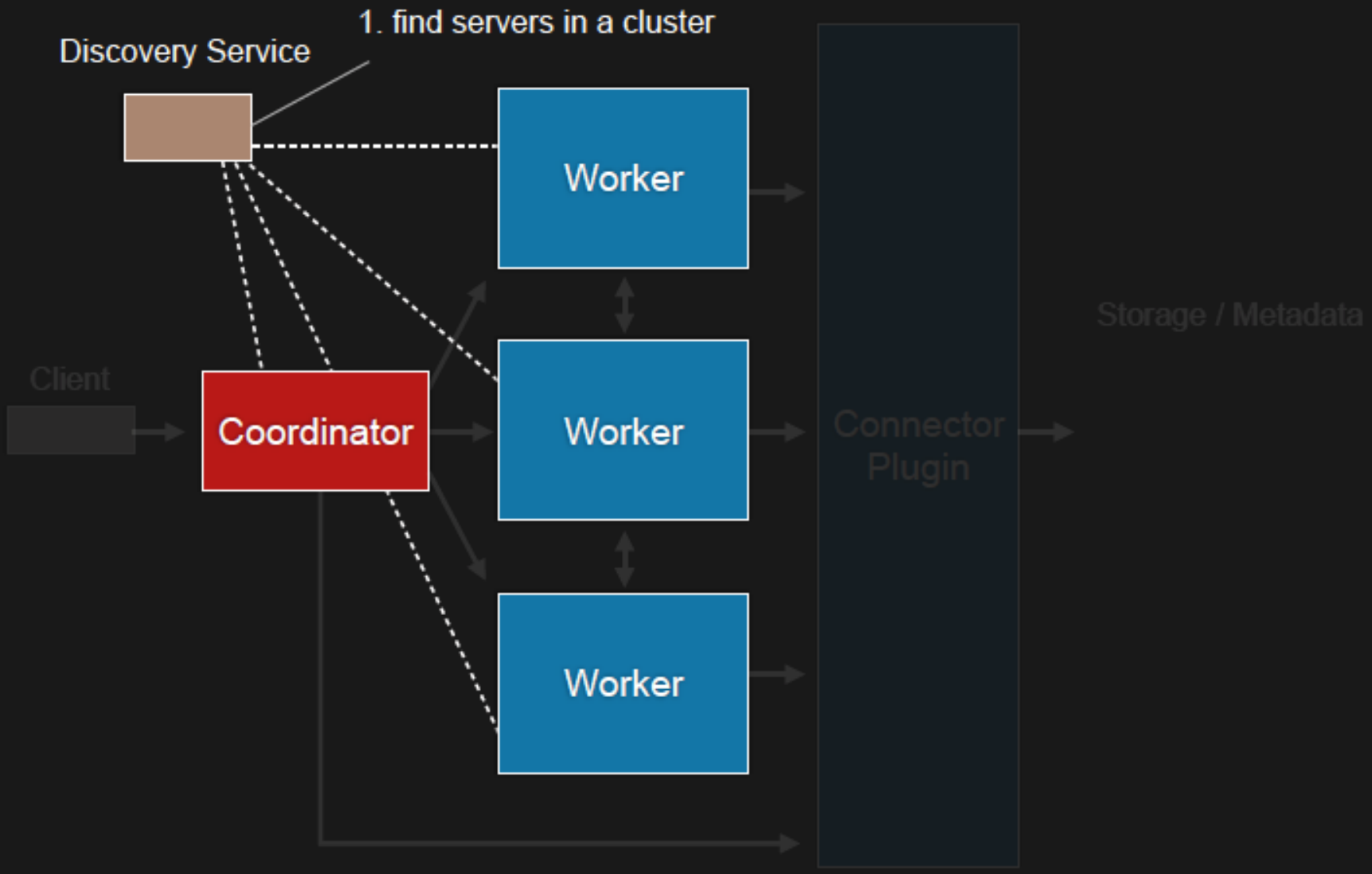
Presto deployment

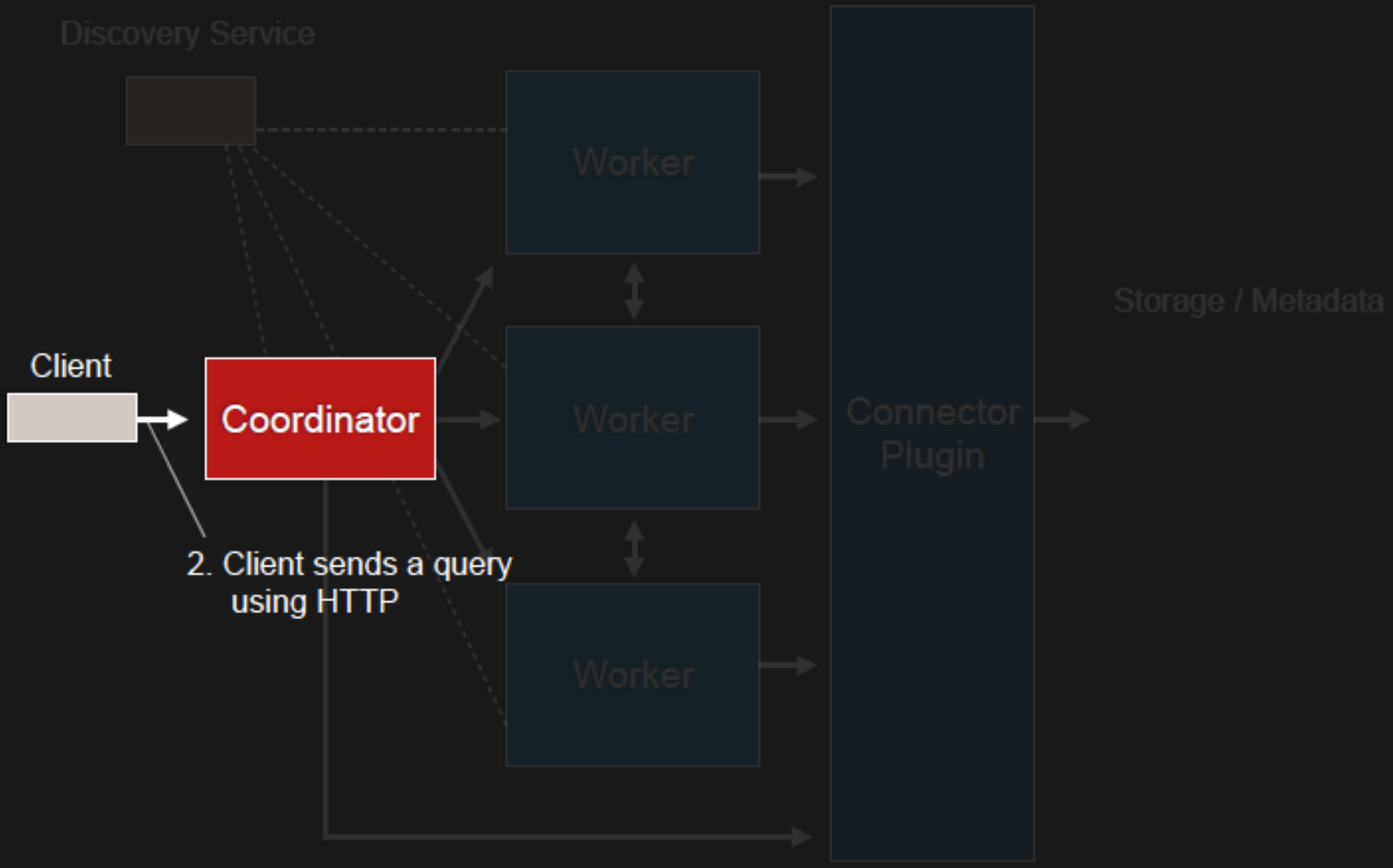
- Facebook (2013)
  - Multiple geographical regions
  - Scaled to 1,000 nodes
  - Actively used by 1,000+ employees who run 30,000+ queries every day
  - Processing 1PB/day

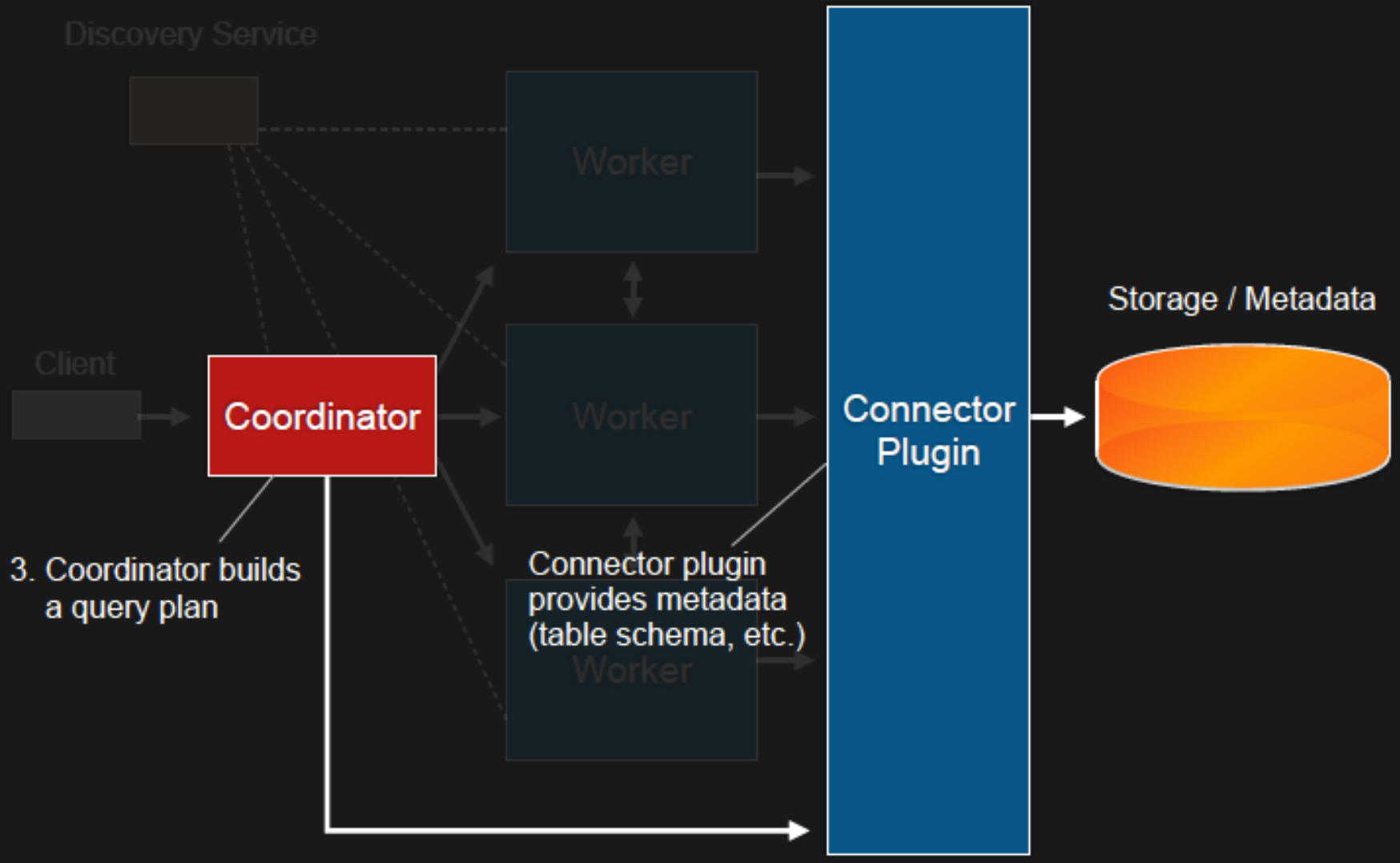
zuora
YAHOO! JAPAN
JD.COM 京东
FreeWheel
amazon
Marin SOFTWARE
mercado Libre.com
LinkedIn
jampp
looker
Comcast
WIX.com
TREASURE DATA
airbnb
Walmart
Alibaba Group
UBER
slack
Bloomberg
GROUPON
GREE
cogo labs
FINRA
twitter
trulia
shopify
Atlassian
AdRoll
SHAZAM
Pinterest
openspan
NASDAQ
Dropbox
CUEBIQ
NETFLIX

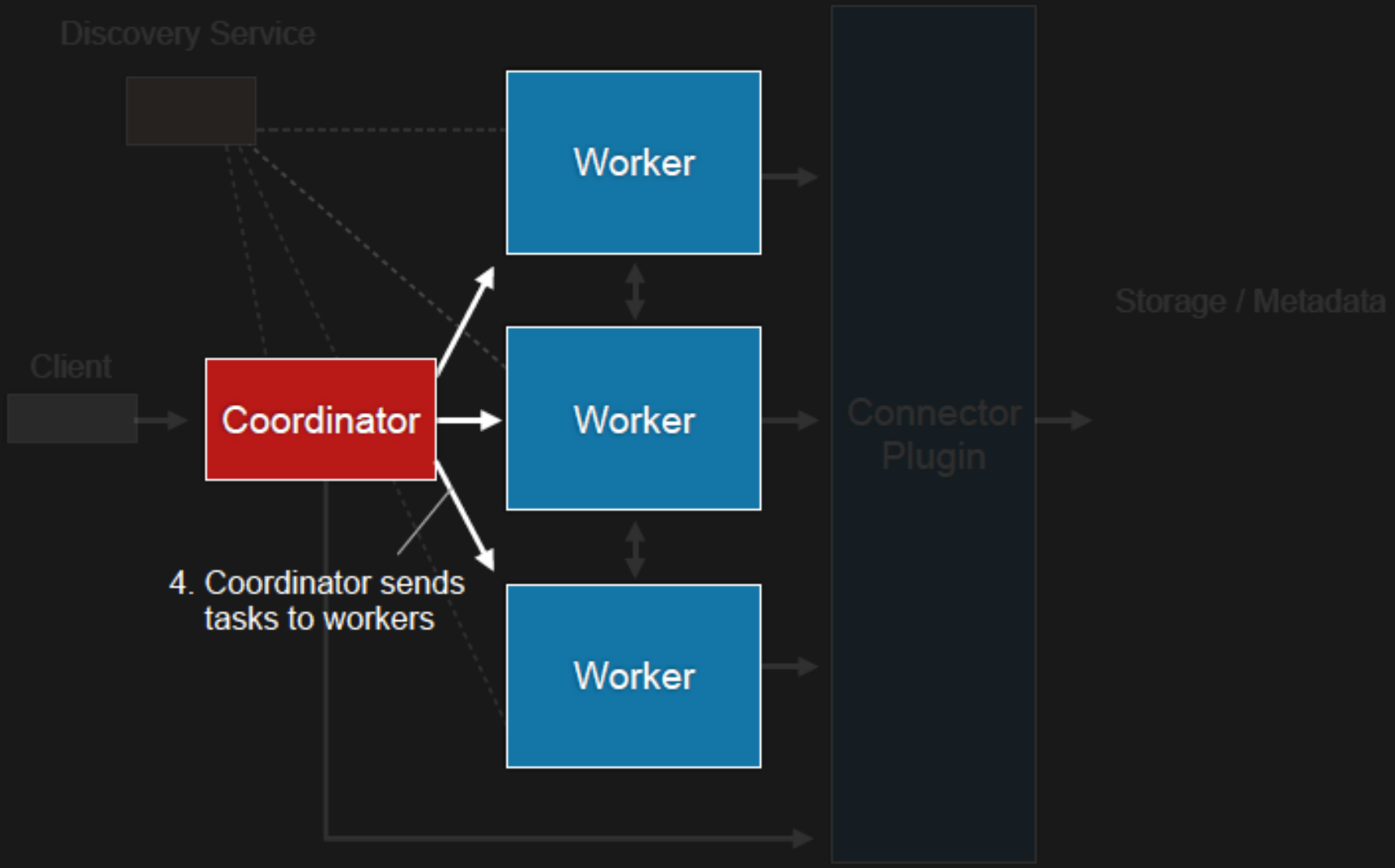
Presto architecture

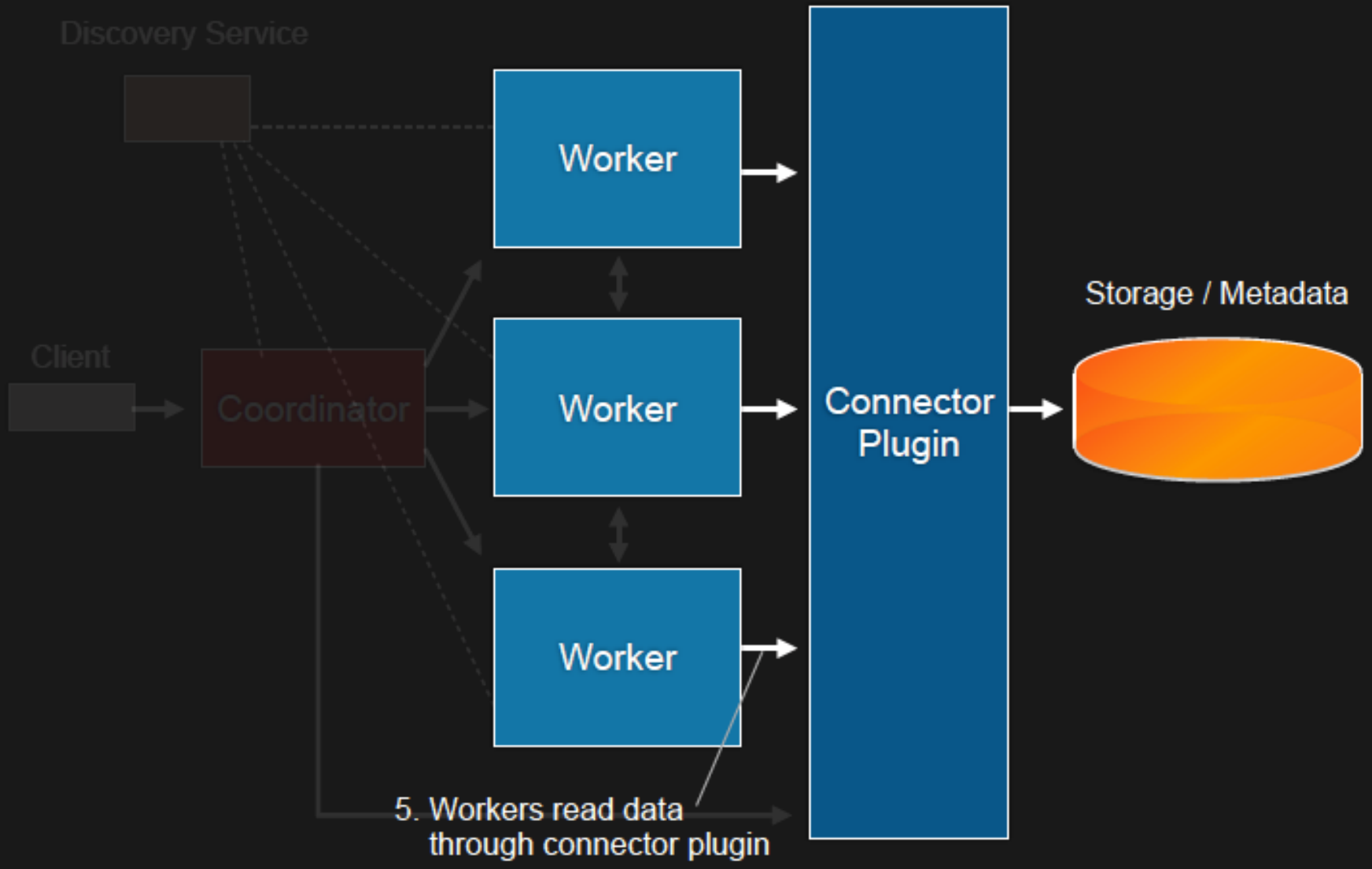


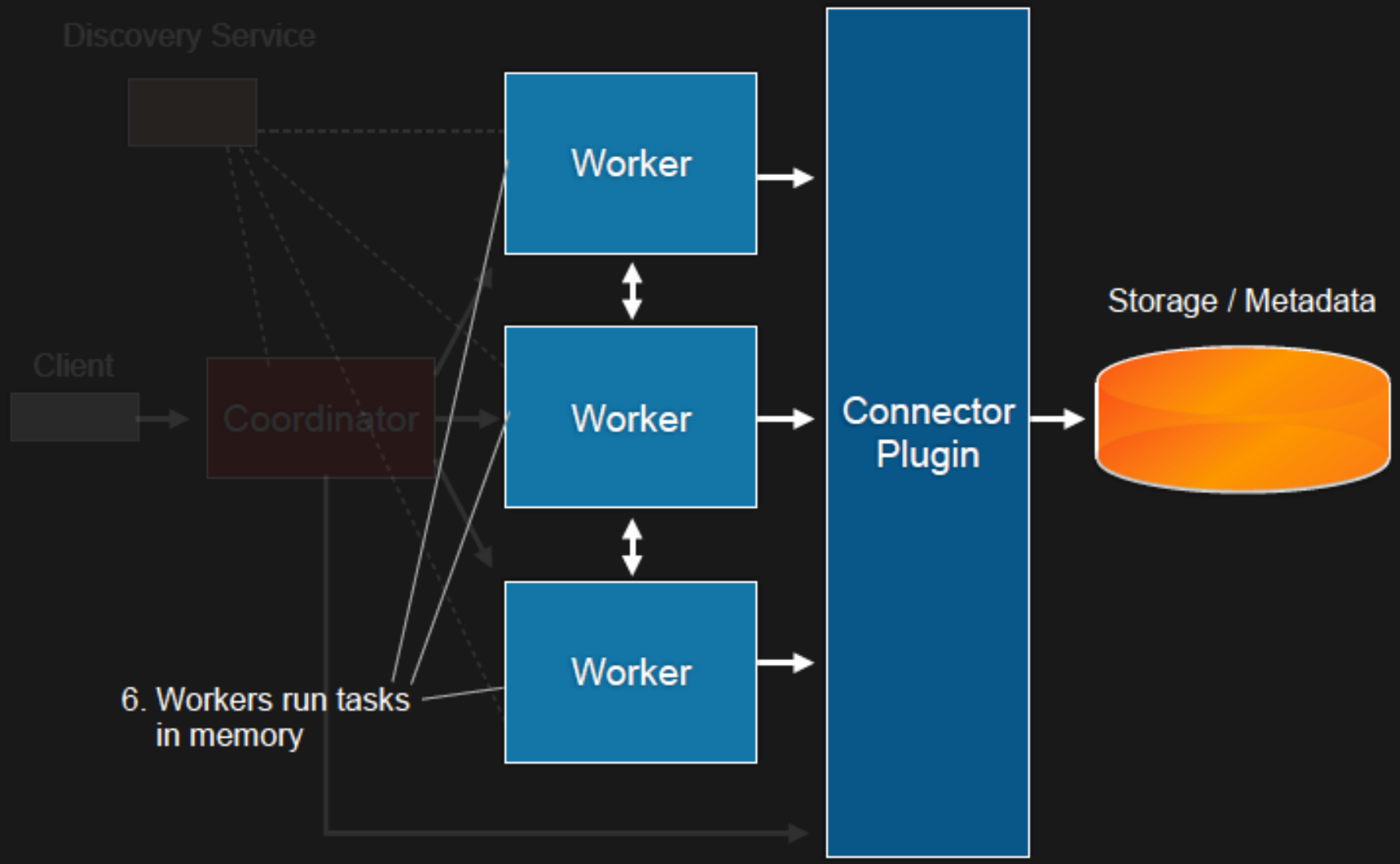

1. find servers in a cluster
Discovery Service
Worker
Storage / Metadata
Client
Coordinator
Worker
Connector
Plugin
Worker

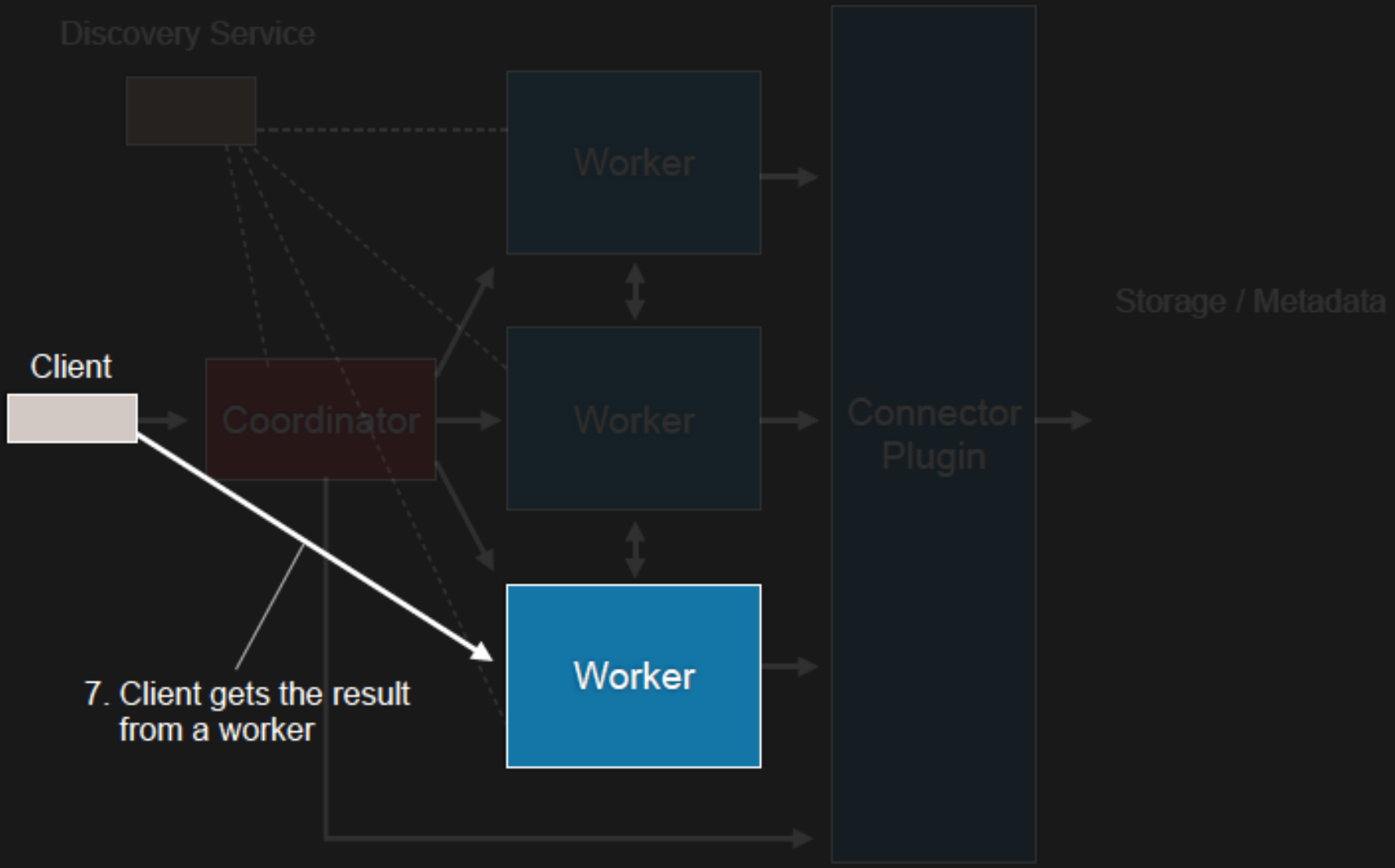

Discovery Service
Worker
Storage / Metadata
Client
Coordinator
Worker
Connector Plugin
2. Client sends a query using HTTP
Worker

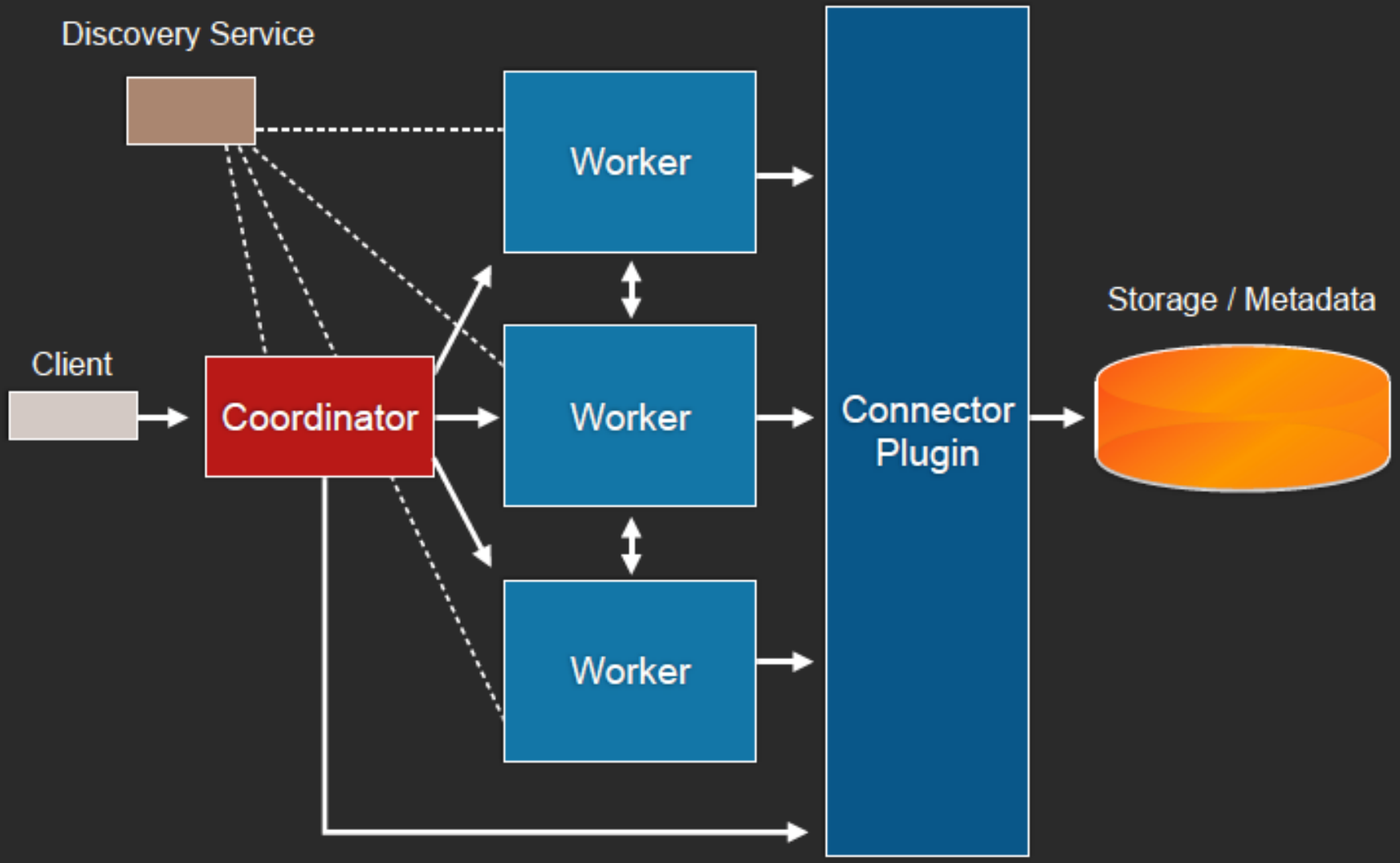

Discovery Service
Worker
Worker
Worker
Client
Coordinator
Connector Plugin
Storage / Metadata
3. Coordinator builds a query plan
Connector plugin provides metadata (table schema, etc.)


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

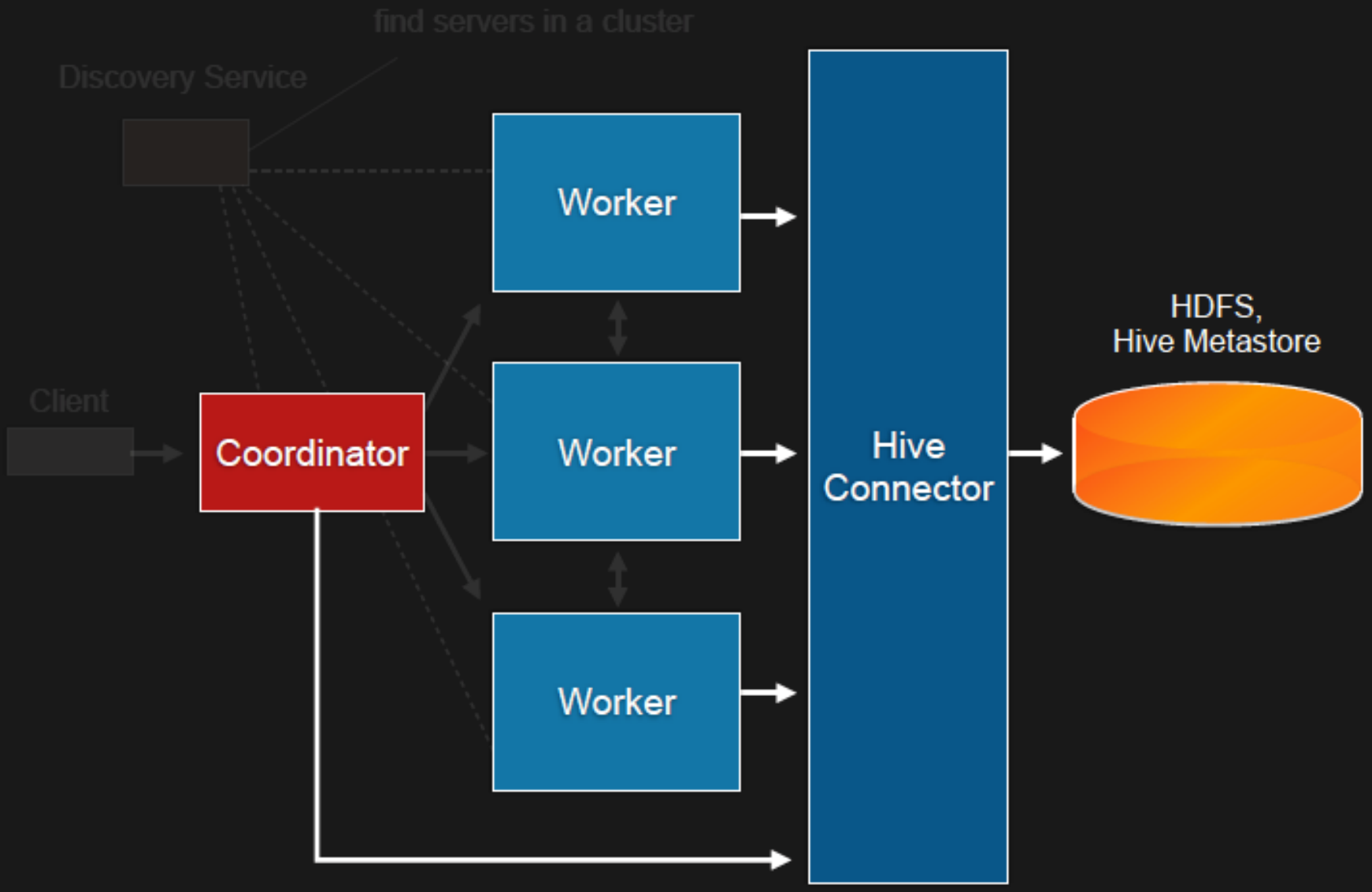

Discovery Service
Worker
Storage / Metadata
Client
Coordinator
Worker
Connector Plugin
Worker
7. Client gets the result from a worker


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

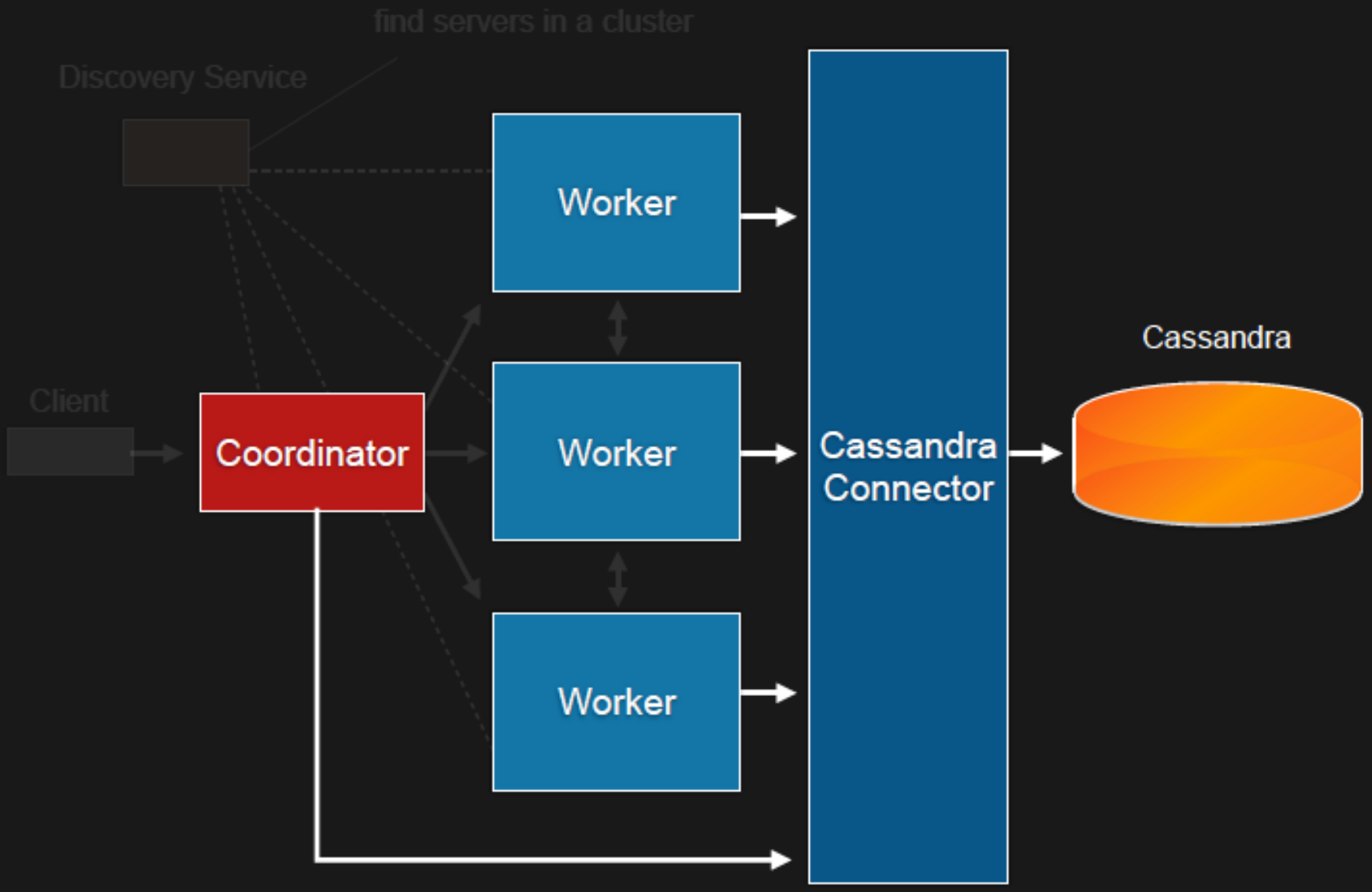
Presto connectors

- Connectors are plugins to Presto
  - written in Java
- Access to storage and metadata
  - provide table schema to coordinators
  - provide table rows to workers
- Implementations
  - Hive connector
  - Cassandra connector
  - MySQL through JDBC connector (prerelease)
  - Or your own connector

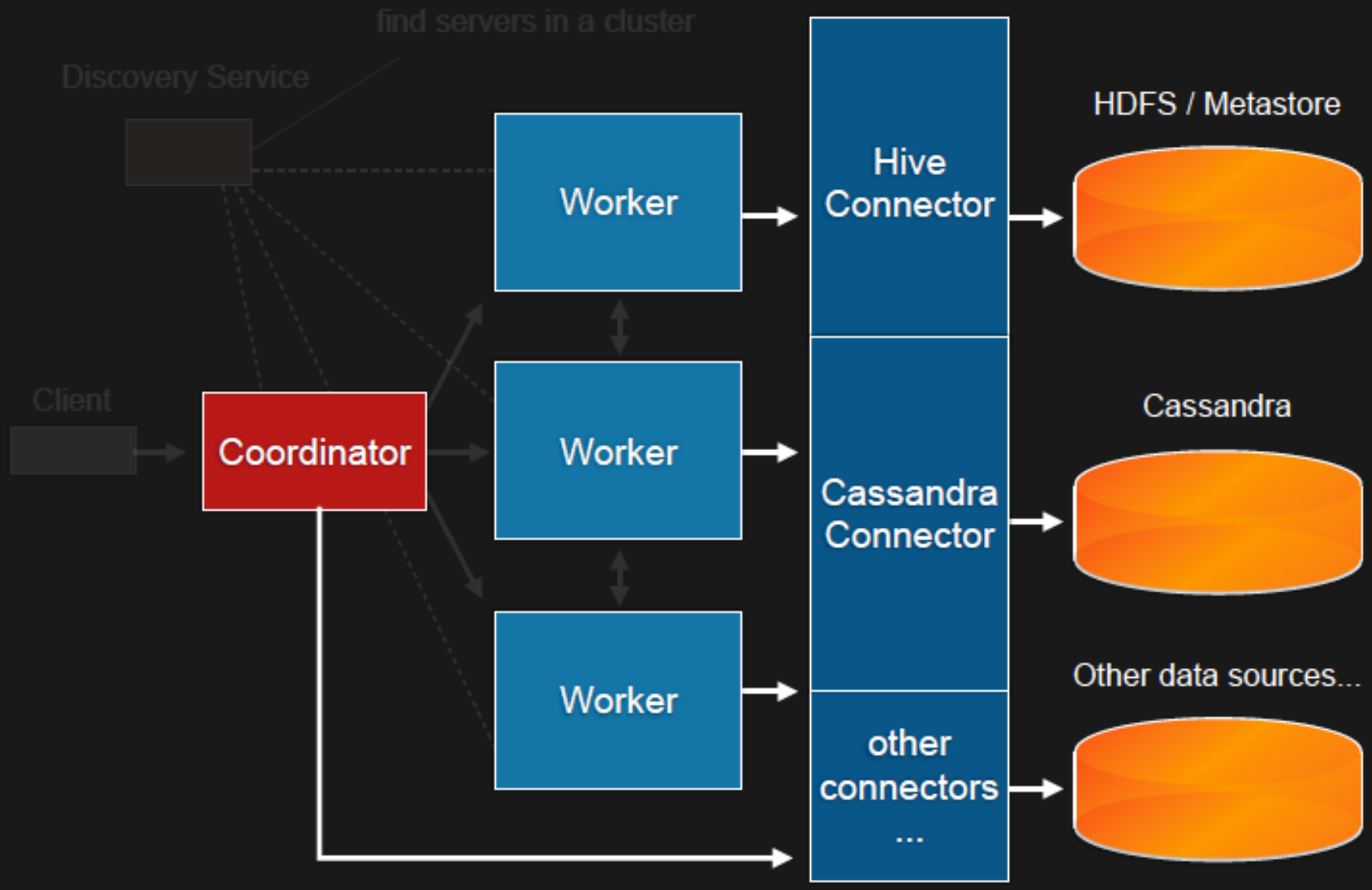
Hive connector



Cassandra connector



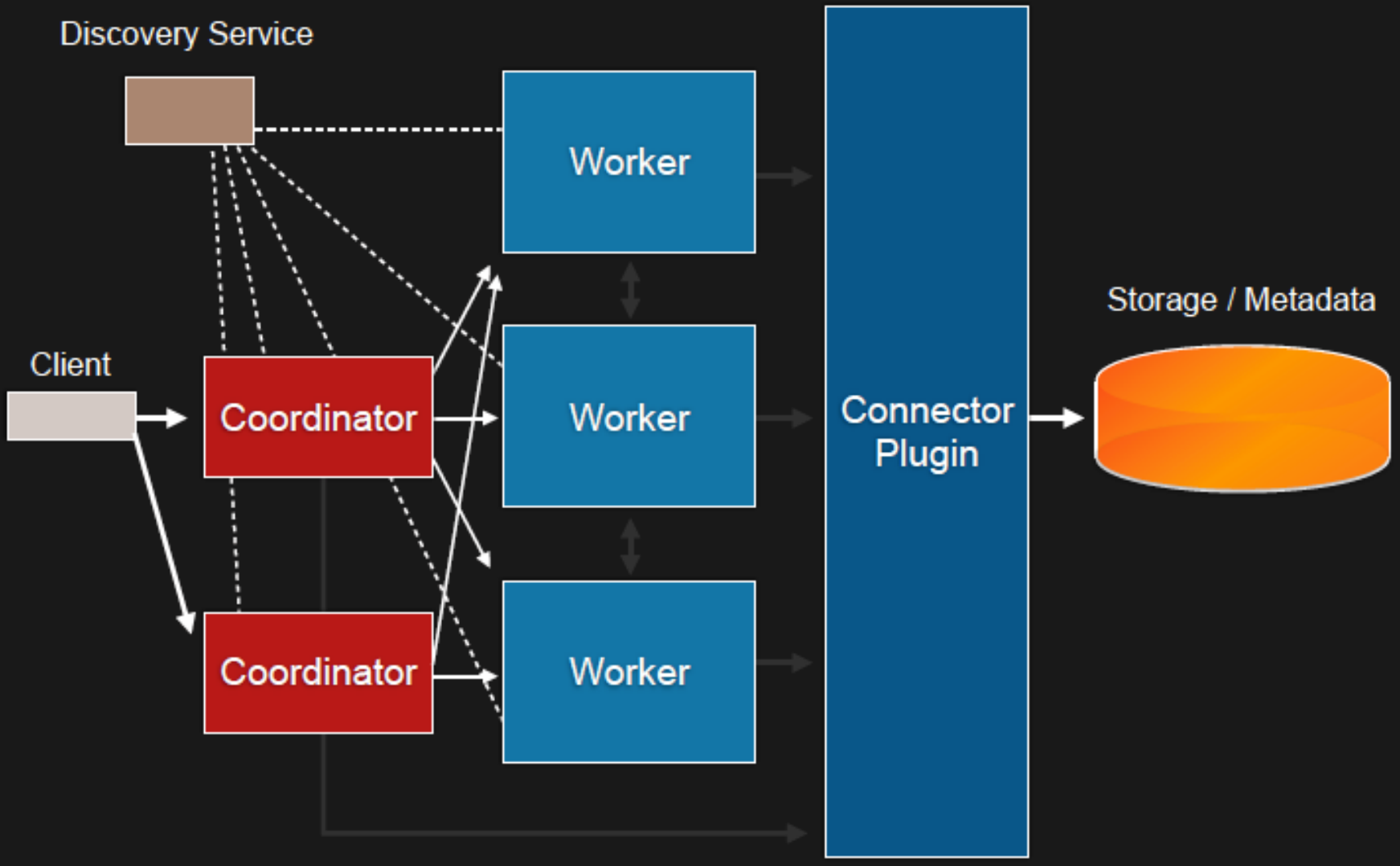
Multiple connectors in a query



Distributed architecture

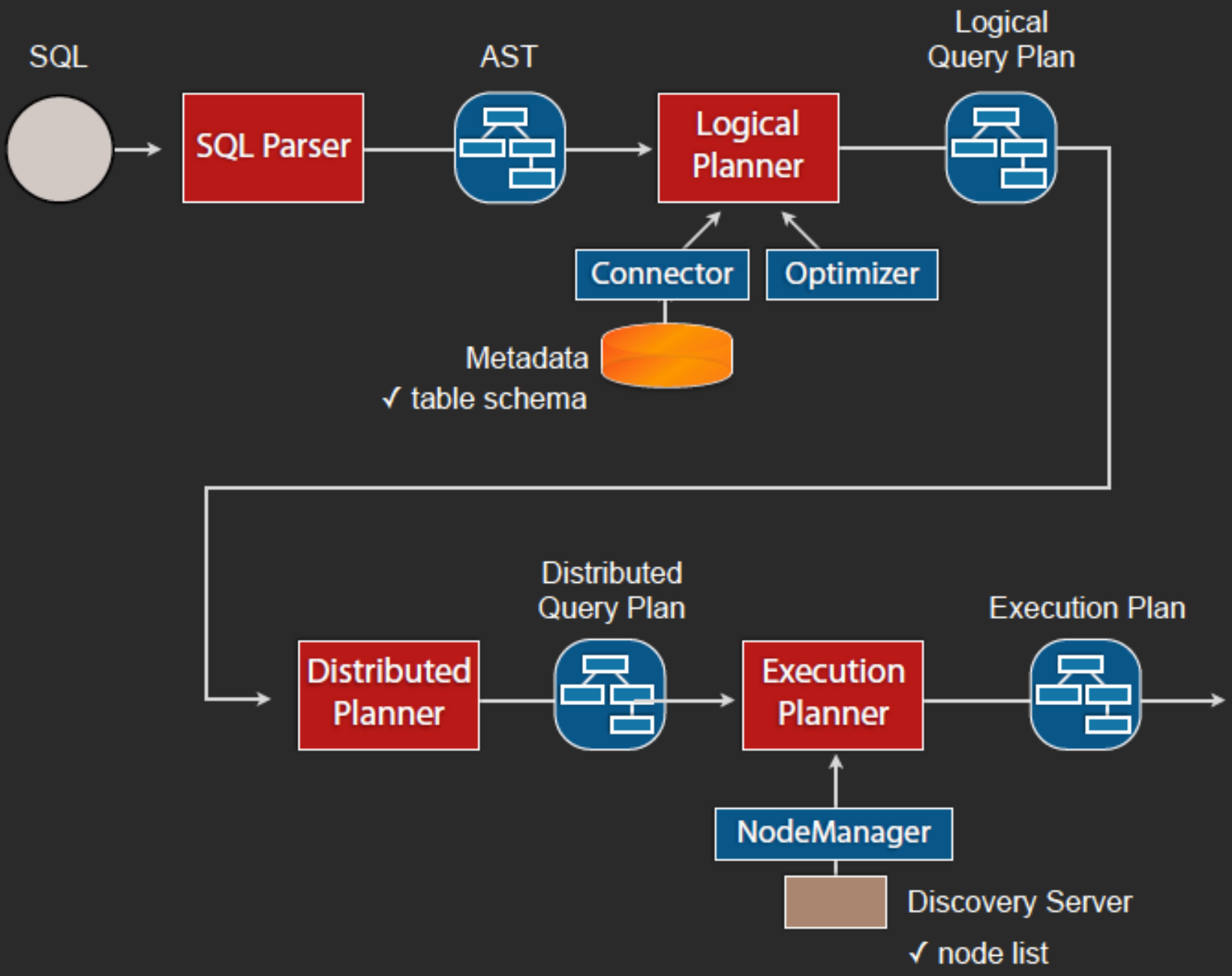
- 3 type of servers:
  - Coordinator, worker, discovery service
- Get data/metadata through connector plugins.
  - Presto is NOT a database
  - Presto provides SQL to existent data stores
- Client protocol is HTTP + JSON
  - Language bindings: Ruby, Python, PHP, Java (JDBC), R, Node.JS...

Coordinator HA



Presto execution model

- Presto is NOT MapReduce
- Presto's query plan is based on DAG
  - more like Apache Tez or traditional MPP databases
- How query runs?
  - Coordinator
    - SQL Parser
    - Query Planner
    - Execution planner
  - Workers
    - Task execution scheduler


SQL
AST
Logical Query Plan
SQL Parser
Logical Planner
Connector
Optimizer
Metadata
✓ table schema
Distributed Query Plan
Execution Plan
Distributed Planner
Execution Planner
NodeManager
Discovery Server
✓ node list

Query Planner

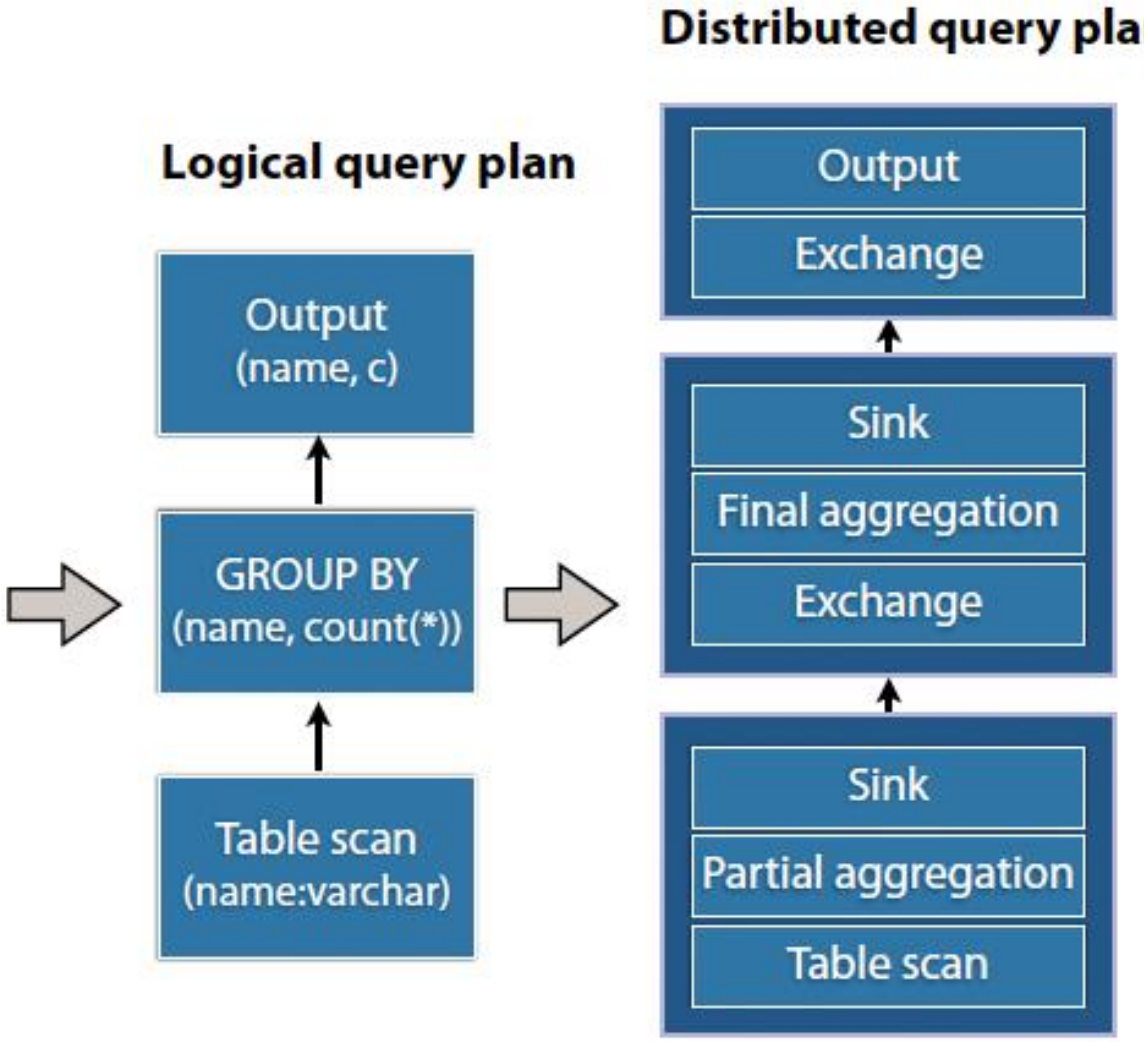
SQL

```
SELECT
  name,
  count(*) AS c
FROM impressions
GROUP BY name
```

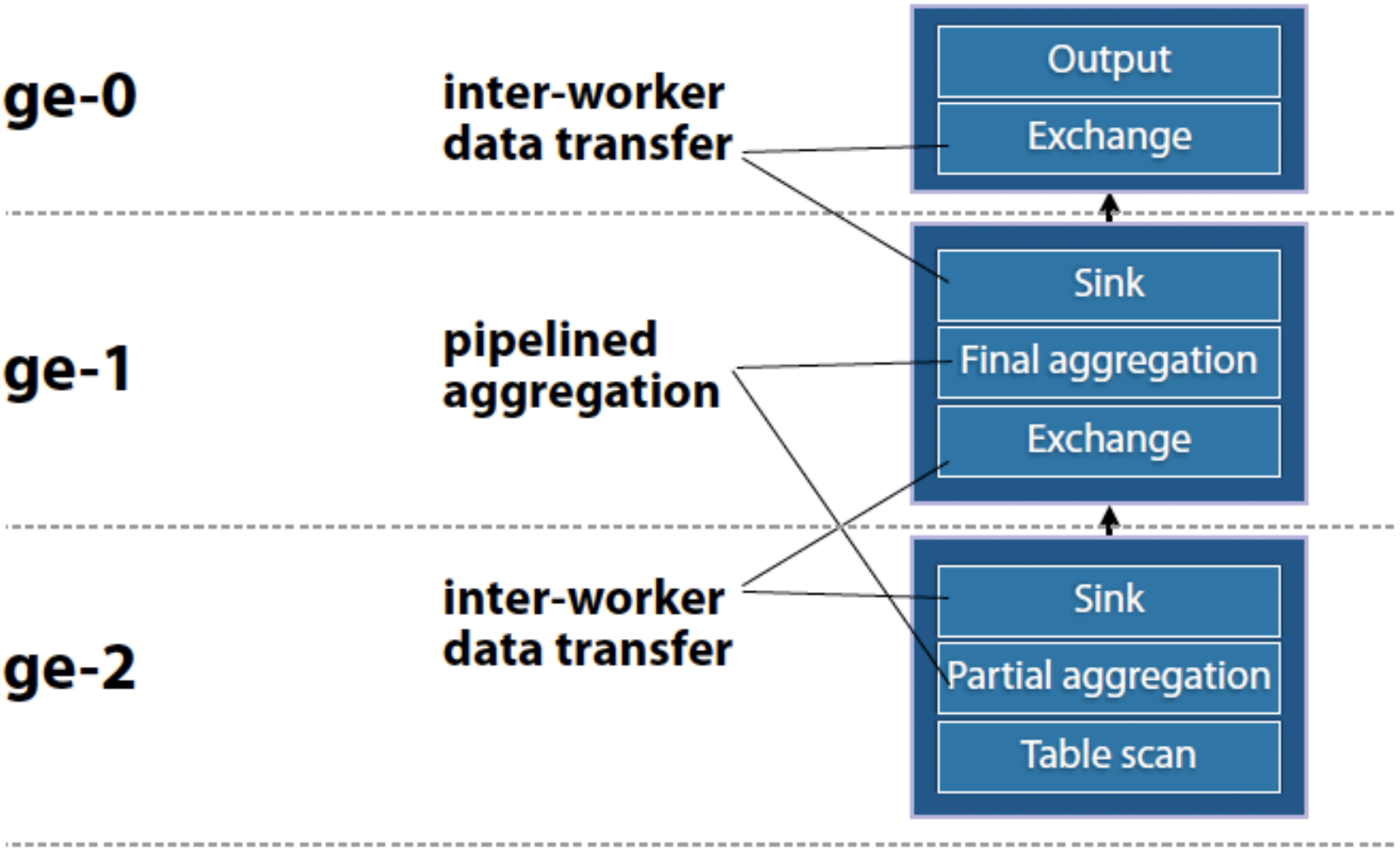
+

Table schema

```
impressions (
  name varchar
  time bigint
)
```



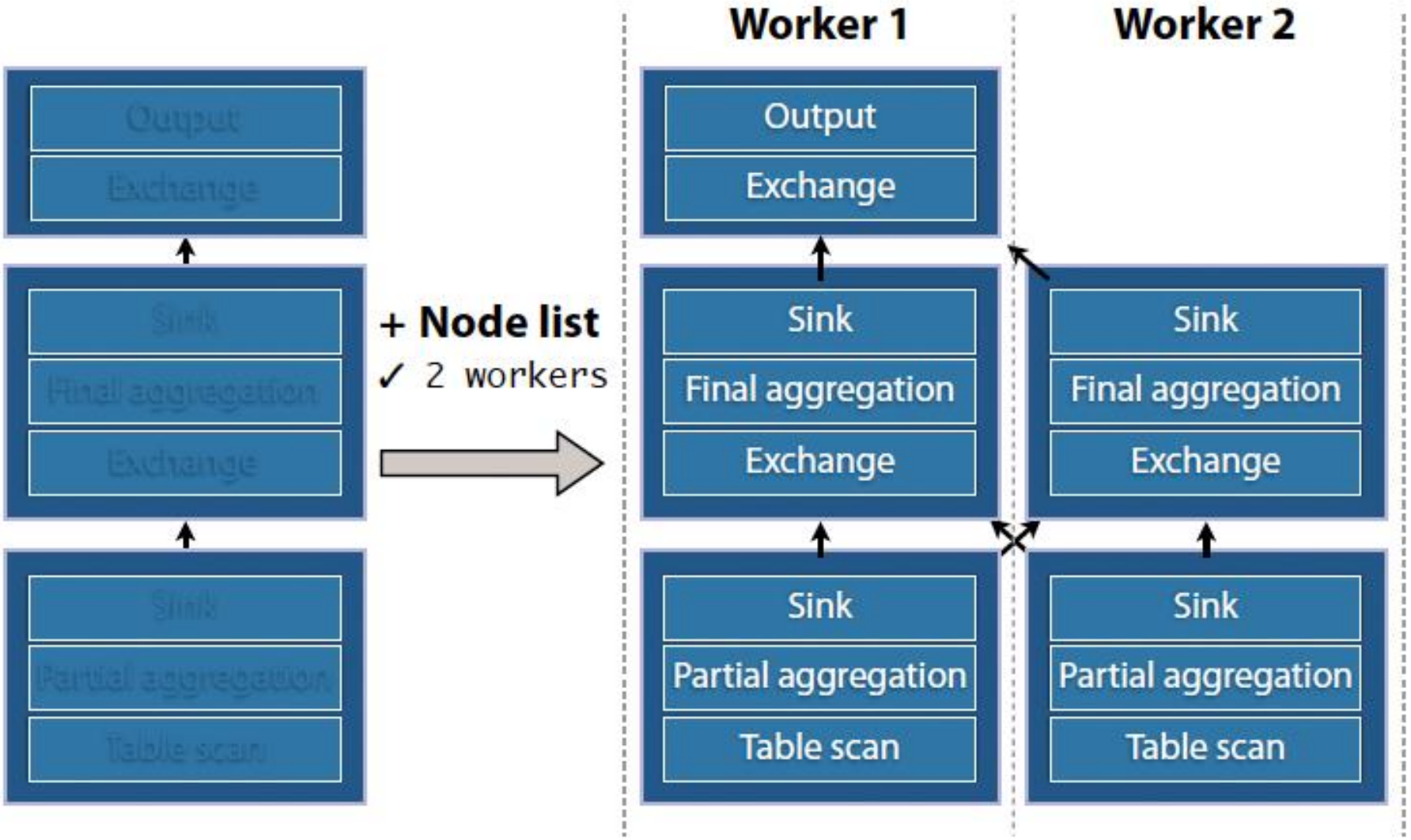
Query Planner - Stages



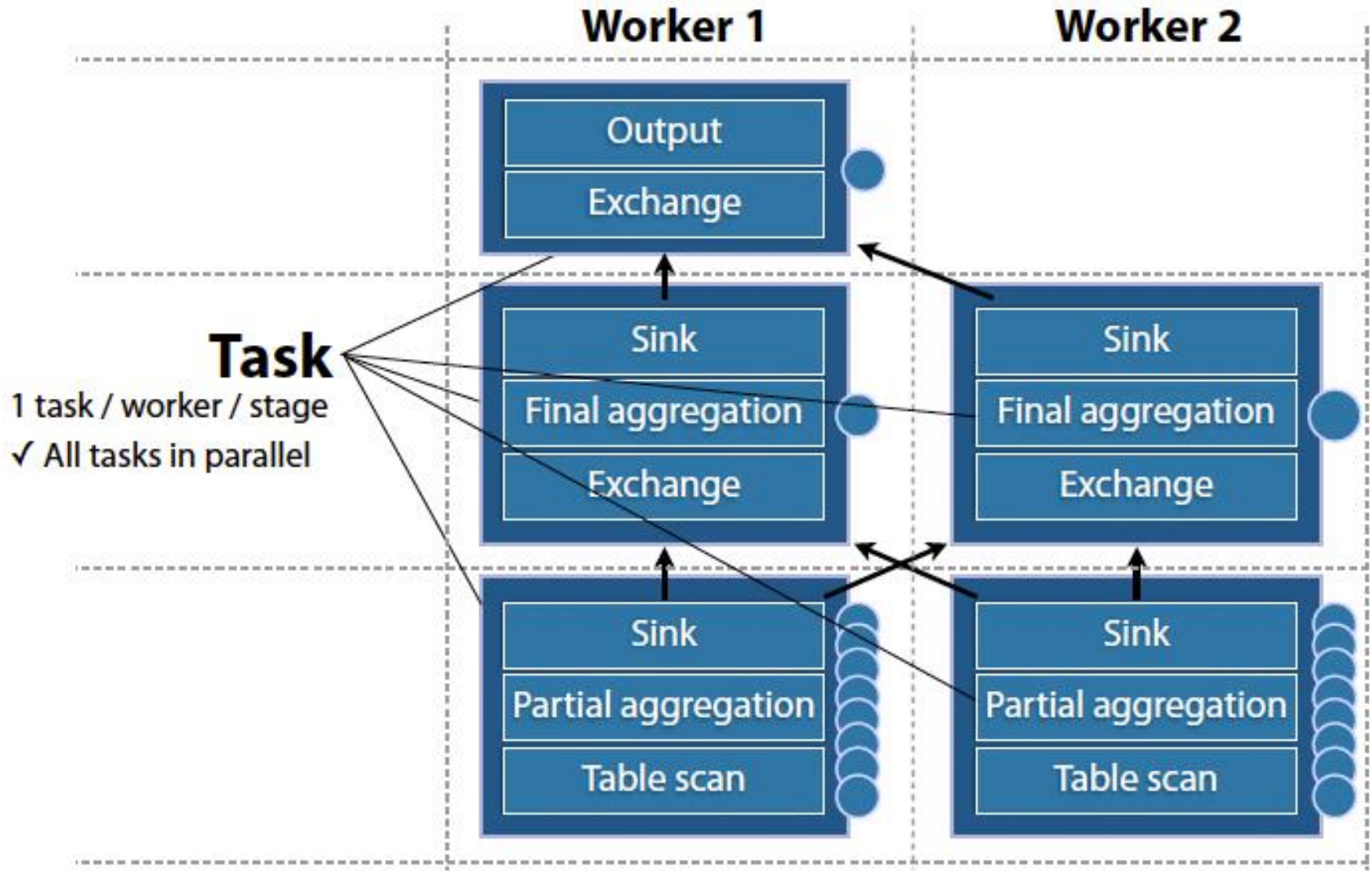
Stages

- A stage is a part of the plan that can be executed in parallel across workers
  - Workers execute same computation on different sets of input data
  - Buffered in-memory data transfers (shuffles) between stages to enable data exchange
- Shuffles add latency, use up buffer memory, and have high CPU overhead

Execution Planner

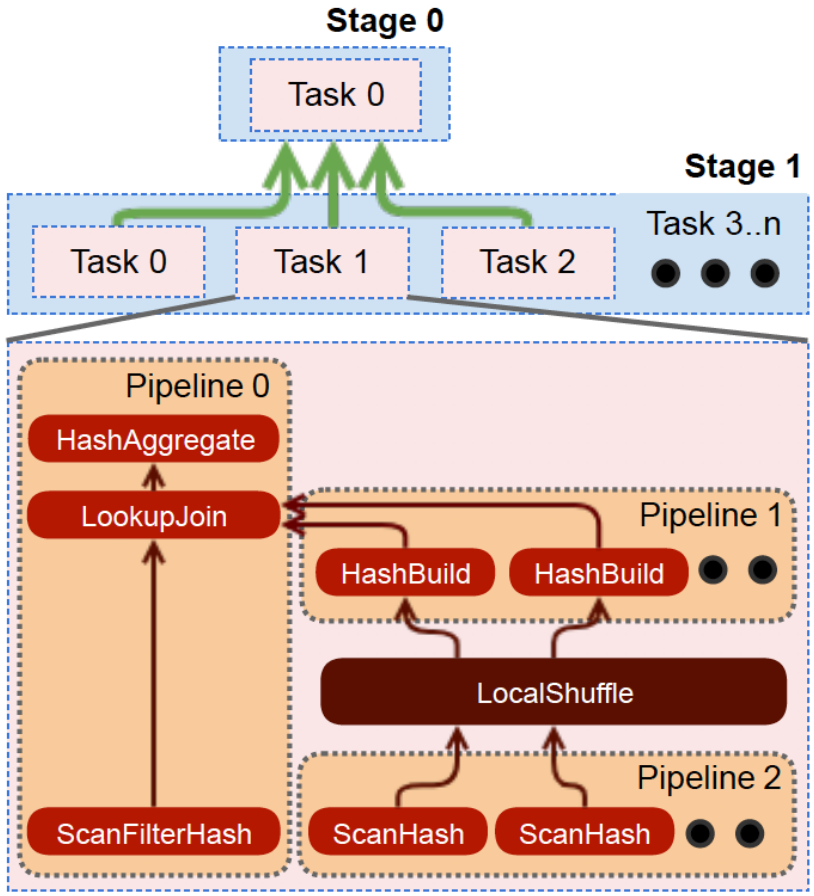


Execution Planner - Tasks



Tasks

- The coordinator distributes plan stages to workers in the form of executable tasks
- 1 task / 1 worker / 1 stage
- A task may have multiple pipelines
- A pipeline consists of a chain of operators



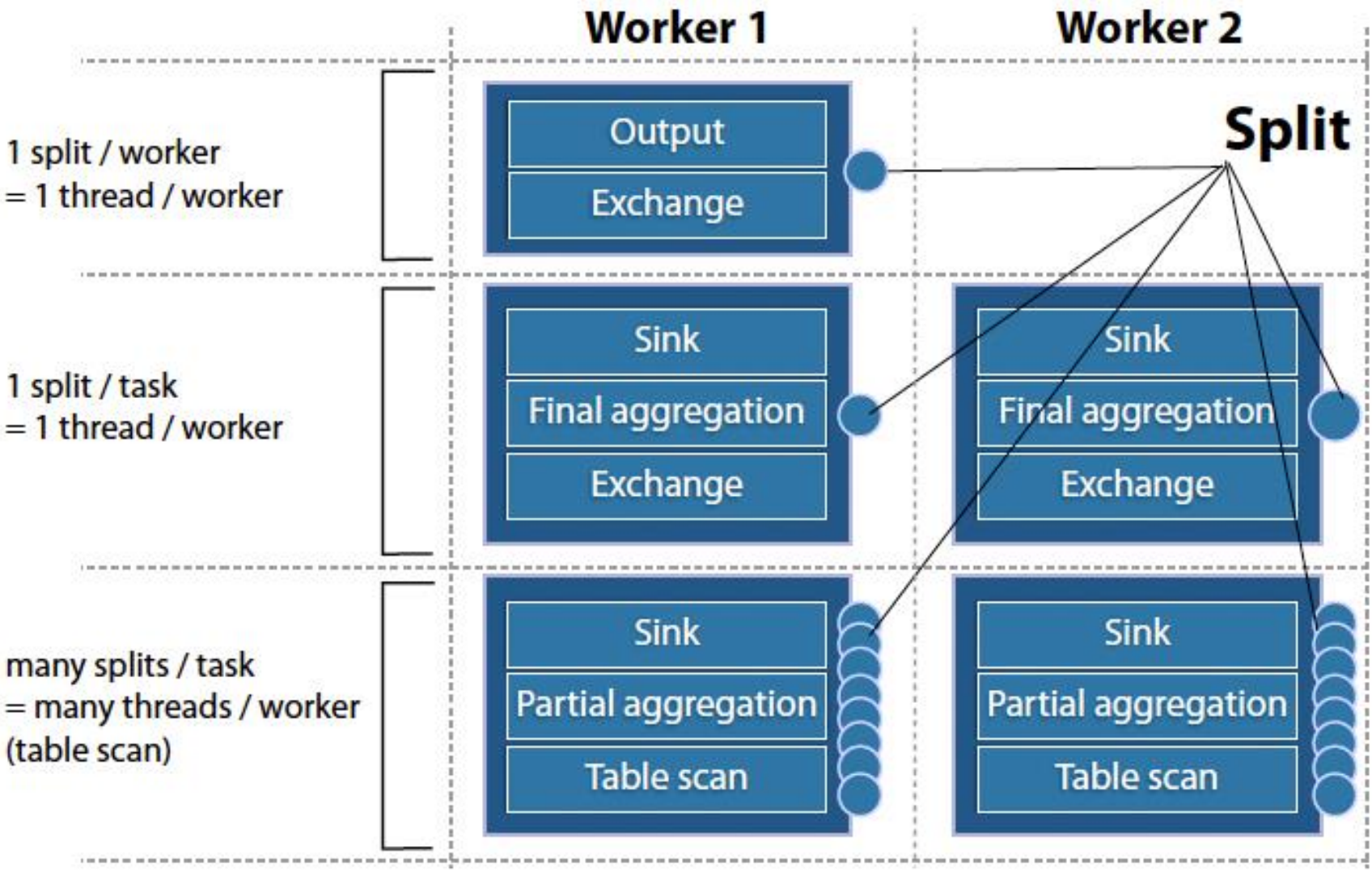
Scheduling

- To execute a query, Scheduler makes two sets of scheduling decisions
- Stage Scheduling
  - all-at-once: minimizes wall clock time by scheduling all stages of execution concurrently
    - data is processed as soon as it is available
    - benefits latency-sensitive use cases such as Interactive Analytics, Developer/Advertiser Analytics, and A/B Testing
  - Phased:
    - Upon a stage should be scheduled according to the policy, it begins to assign tasks for that stage to worker nodes
    - improves memory efficiency for the Batch Analytics use case

Task scheduling

- Stages are classified into leaf and intermediate stages
- Leaf Stages
  - task scheduler takes into account the constraints imposed by the network and connectors when assigning tasks to worker nodes
    - co-located workers and storage nodes
    - Connector Data Layout constraints
- Intermediate Stages
  - Tasks for intermediate stages can be placed on any worker node
  - The engine still needs to decide how many tasks should be scheduled for each stage

Execution Planner - Split



Split scheduling

- Splits are opaque handles to an addressable chunk of data
  - in an external storage system
  - or intermediate results produced by other workers
  - Eg. Reading from HDFS
    - A split is a file path and offsets to a region of the file
- Split Assignment
  - Splits are assigned to each task lazily
    - Splits are enumerated as the query executes, not up front
    - Queries that can start producing results without processing all the data
    - Splits are assigned to worker with shortest queue
    - Reduces metadata memory usage on coordinator

Query optimization: Data layouts

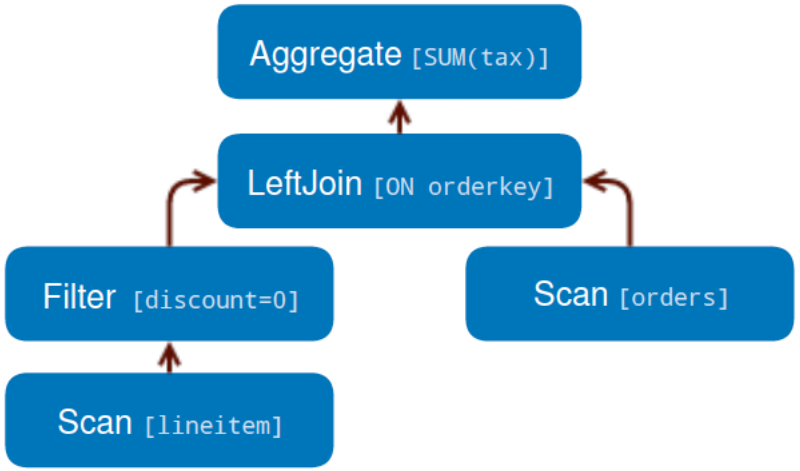
- Optimizer takes advantage of physical layout of data
  - Properties: partitioning, sorting, grouping, indexes
- Tables can have multiple layouts with different properties
  - Layouts can have a subset of columns or data
- Optimizer chooses best layout for query
- Tune queries by adding new physical layouts

Query optimization: Predicate pushdown

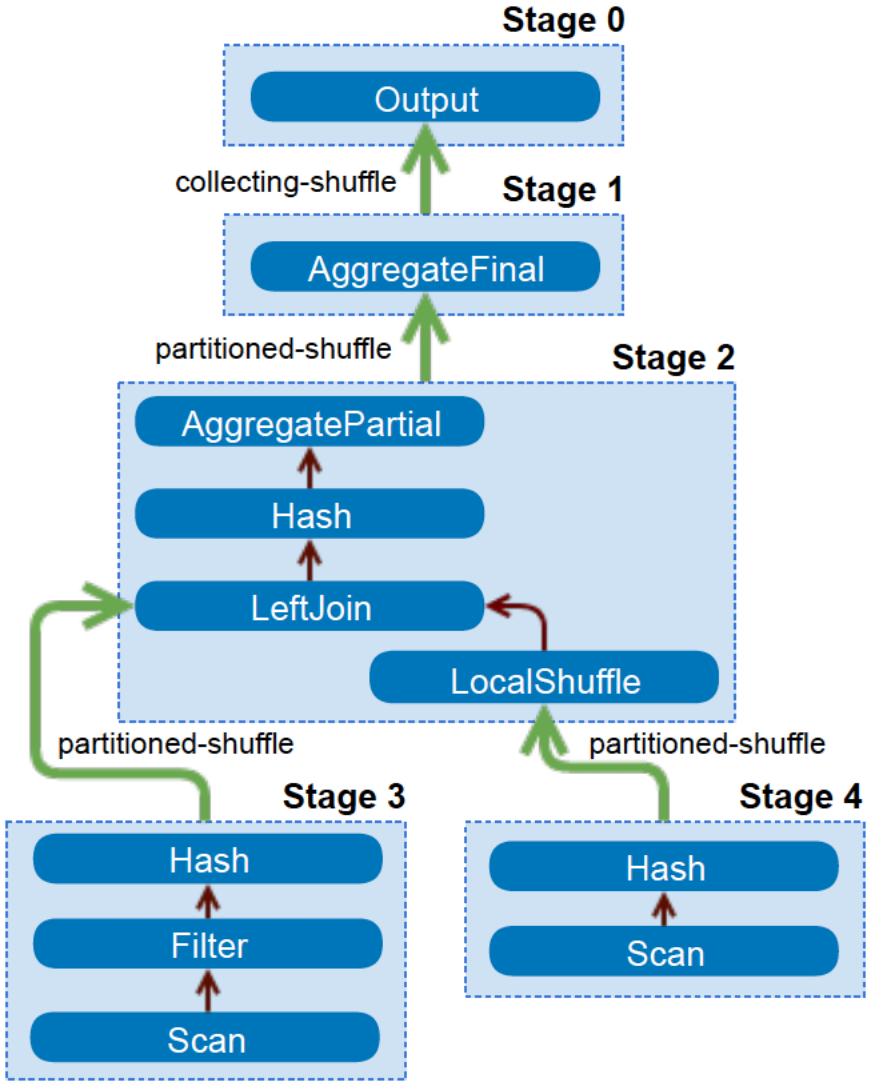
- The optimizer can push range and equality predicates down through the connector improves filtering efficiency
- Engine provides connectors with a two part constraint:
  - Domain of values: ranges and nullability
  - “Black box” predicate for filtering

Eg.

```
SELECT
  orders.orderkey, SUM(tax)
FROM orders
LEFT JOIN lineitem
  ON orders.orderkey = lineitem.orde
WHERE discount = 0
GROUP BY orders.orderkey
```



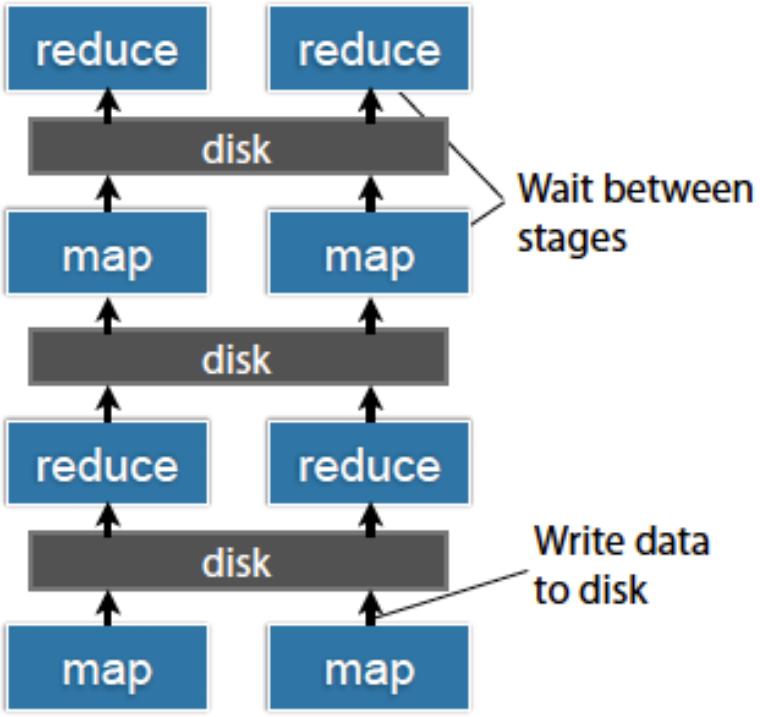
Logical plan



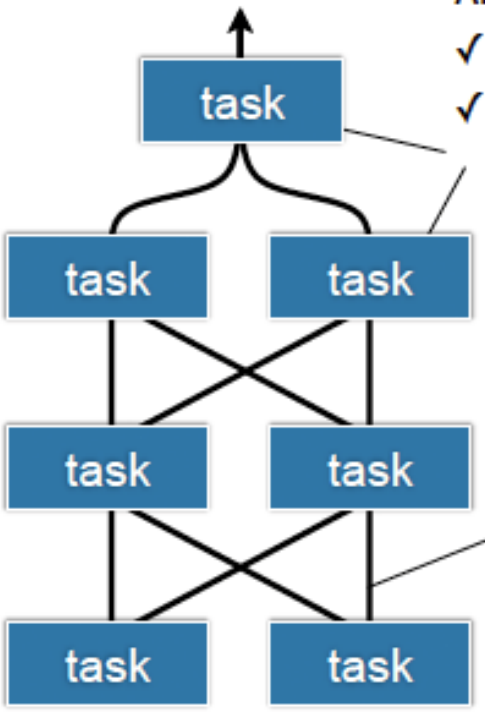
Distributed plan

MapReduce vs. Presto

MapReduce



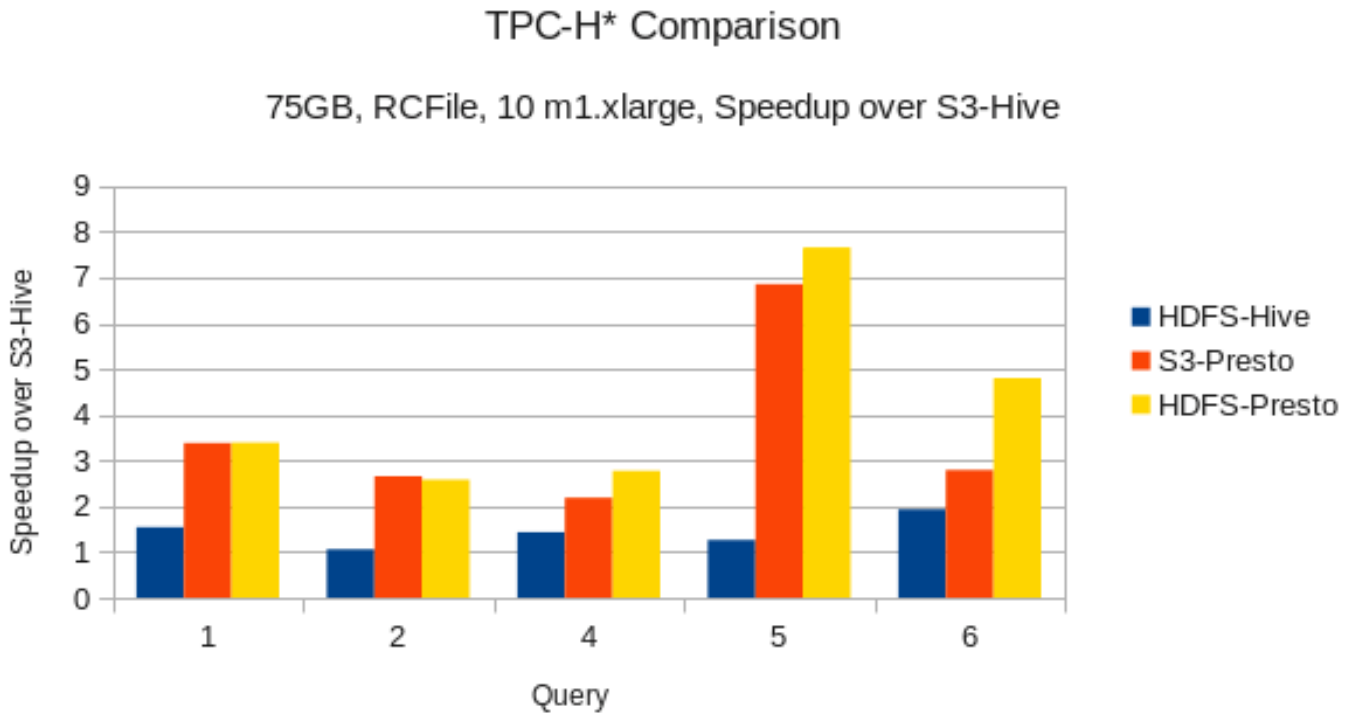
Presto



Query Execution

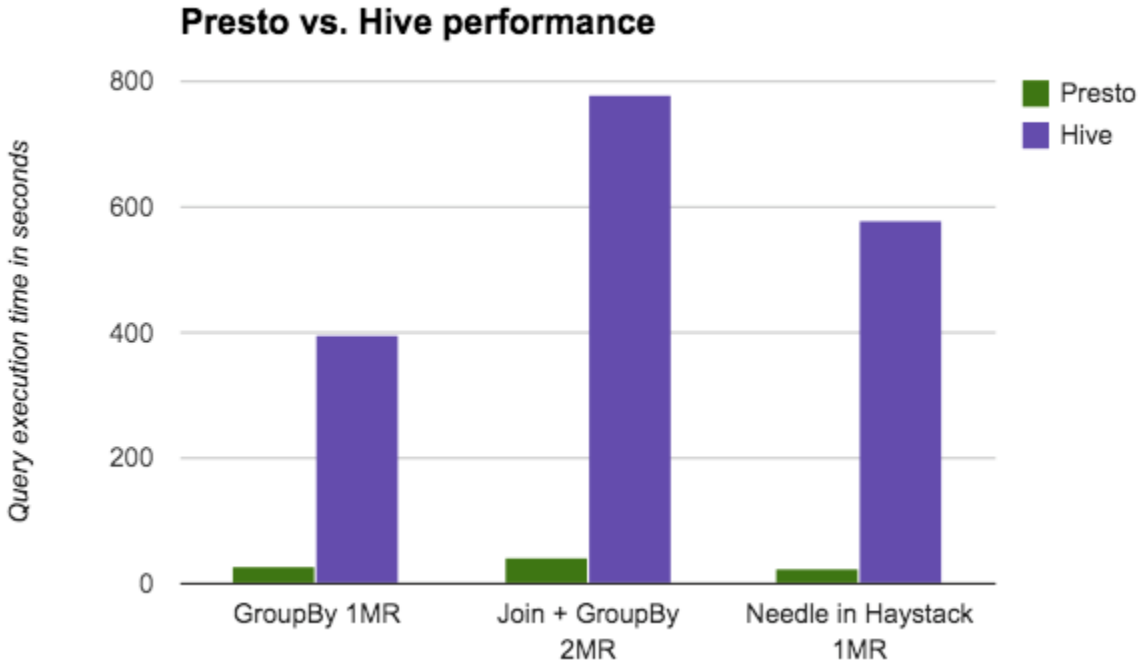
- SQL is converted into stages, tasks and splits
- All tasks run in parallel
  - No wait time between stages (pipelined)
  - If one task fails, all tasks fail at once (query fails)
- Memory-to-memory data transfer
  - No disk IO
  - If aggregated data doesn’t fit in memory, query fails
  - Note: query dies but worker doesn’t die. Memory consumption of all queries is fully managed

Qubole Presto benchmark

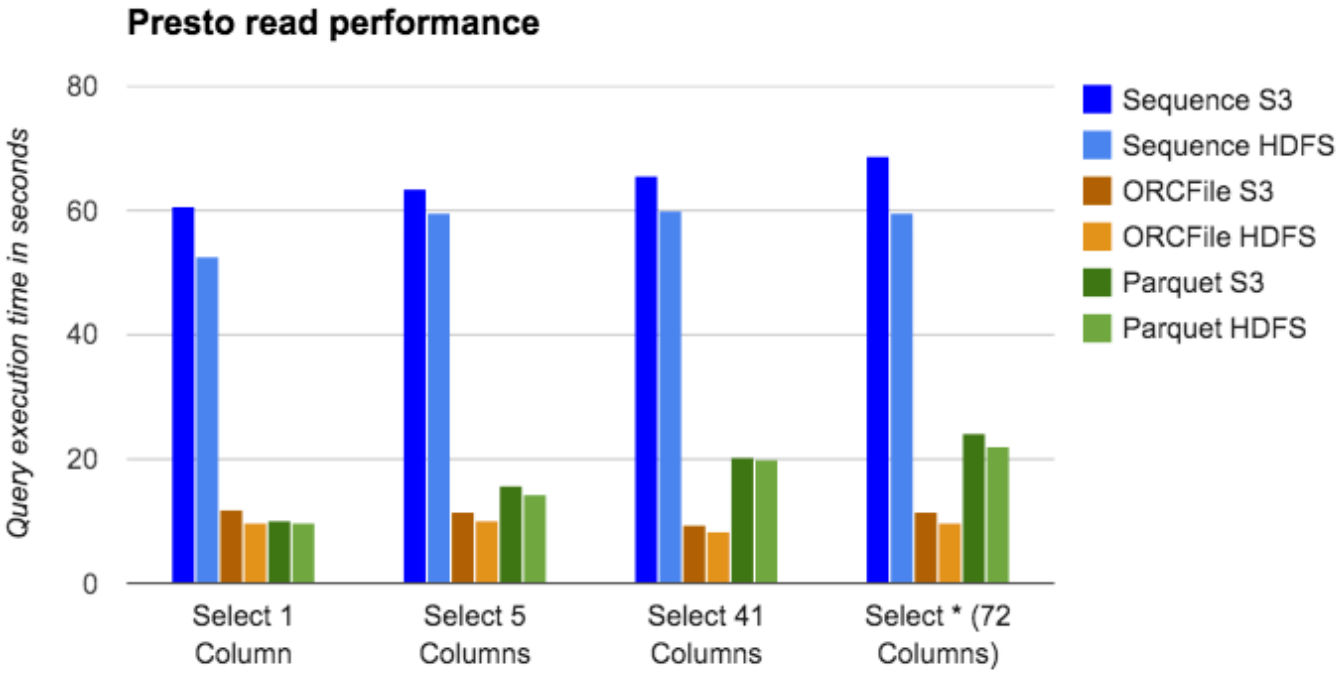


Netflix Presto benchmark (1)

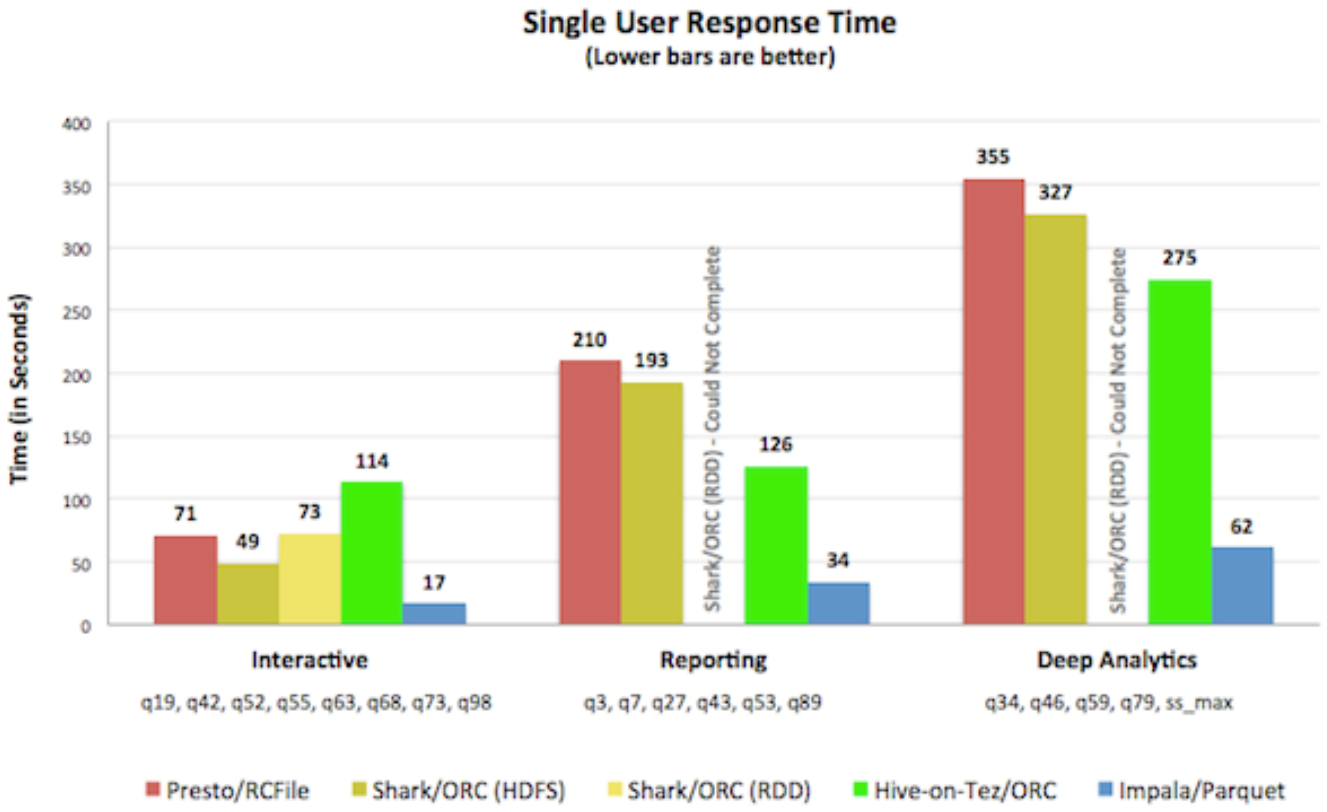
- a group-by query, a join plus a group-by query, and a needle-in-a-haystack (table scan)
- Parquet input files on S3/140GB to 210GB file size
- 40 nodes m2.4xlarge



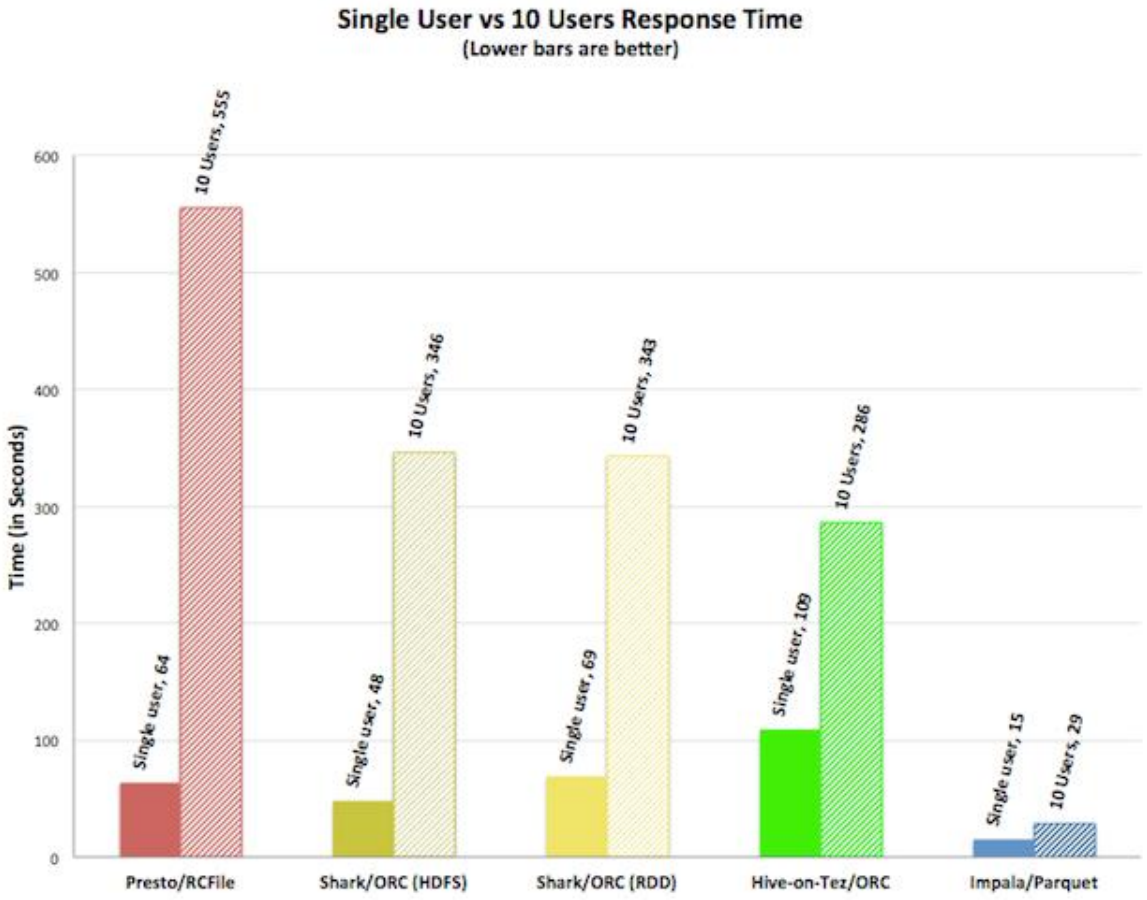
Netflix Presto benchmark (2)



Cloudera benchmark (1)

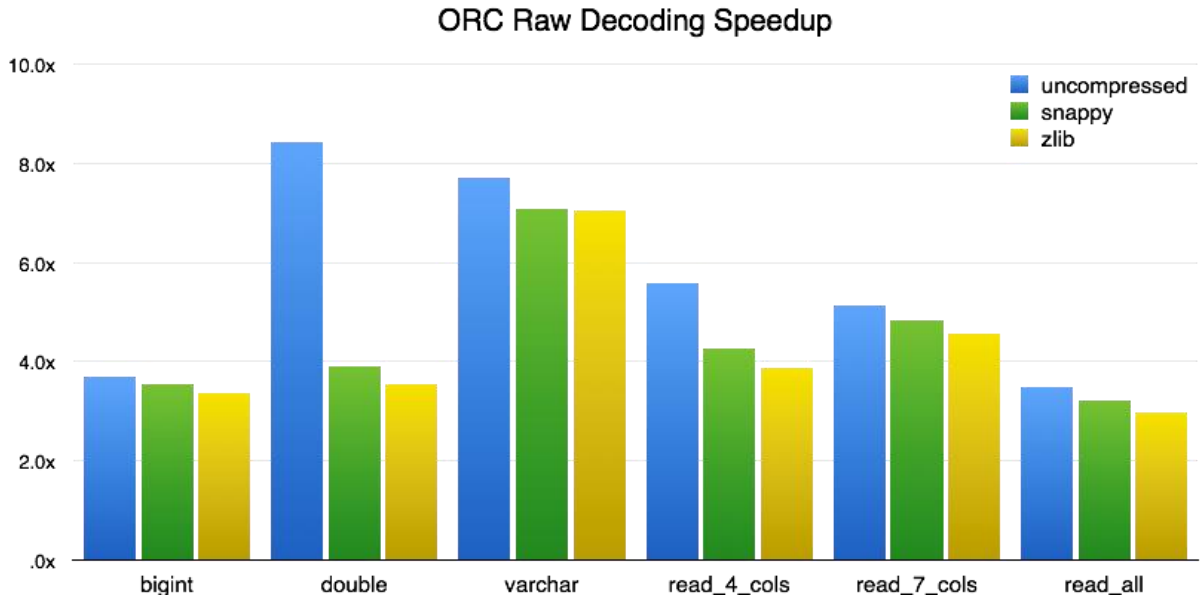


Cloudera benchmark (2)



Even faster: Data at the speed of Presto ORC

- 6 million rows using the TPC-H
- Read various sets of columns
- Old Hive-based ORC reader vs. the new Presto ORC reader



References

- https://code.facebook.com/posts/370832626374903/even-faster-data-at-the-speed-of-presto-orc/
- https://www.facebook.com/notes/facebook-engineering/presto-interacting-with-petabytes-of-data-at-facebook/10151786197628920/
- Traverso, Martin. "Presto: Interacting with petabytes of data at Facebook." 2014.
- https://www.slideshare.net/frsyuki/presto-hadoop-conference-japan-2014