Presto: SQL-on-Anything
DataWorks Summit, San Jose 2017

Martin Traverso, Facebook

Matt Fuller, Teradata

facebook.

TERADATA

Let's Make Some Noise!

@DataWorksSummit

#DWS17

#prestodb

#facebook

#teradata

What is Presto?

- Open source distributed SQL query engine
  - Originally developed by Facebook
  - ANSI SQL compliant
  - Like Hive, it's not a database

- Key Differentiators
  - Performance & Scale
  - Cross platform query capability, not only SQL on Hadoop
  - Supports federated queries
  - Used in production at many well known web-scale companies

- Distributed under the Apache License, hosted on GitHub

Presto in Action

facebook.

NETFLIX

UBER

twitter

GROUPON

Bloomberg

airbnb

Multiple clusters (1000s of nodes total)
300PB in HDFS, MySQL, and Raptor
1000s users, 10-100s concurrent queries

LinkedIn

Dropbox

WIX

Atlassian

FINRA

Presto in Action

facebook

NETFLIX

UBER

twitter

GROUPON

Bloomberg

airbnb

250+ nodes on AWS
40+ PB stored in S3 (Parquet)
Over 650 users with 6K+ queries daily

LinkedIn

Dropbox

WiX

Atlassian

FINRA

Presto in Action

facebook

NETFLIX

UBER

twitter

GROUPON

Bloomberg

airbnb

300+ nodes (2 dedicated clusters)
100K+ & 20K+ queries daily

LinkedIn

Dropbox

WIX

Atlassian

FINRA

Presto in Action

facebook

NETFLIX

UBER

twitter

GROUPON

Bloomberg

airbnb

200+ nodes on-premises
Parquet nested data

LinkedIn

Dropbox

WiX

Atlassian

FINRA

Presto in Action

facebook   NETFLIX   UBER   twitter

GROUPON

120+ nodes in AWS
2PB is S3 and 200+ users
supported by Teradata

Bloomberg

airbnb

LinkedIn

Dropbox   WiX   Atlassian   finra

Presto Architecture

Pluggable

Coordinator

Metadata API		Data Location API

Client → Parser/analyzer → Planner → Scheduler

Worker

Worker

Data Stream API
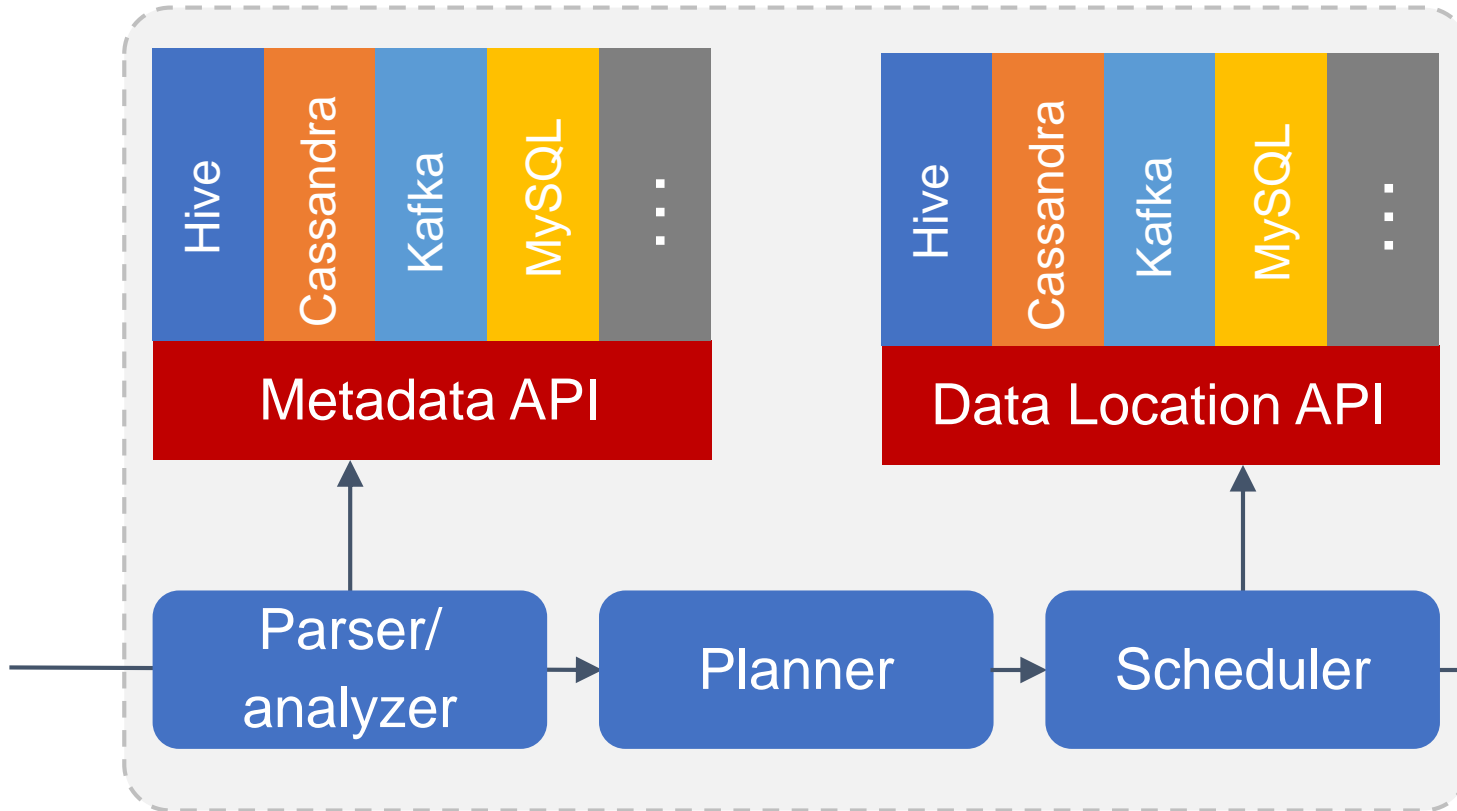
Worker

Data Stream API
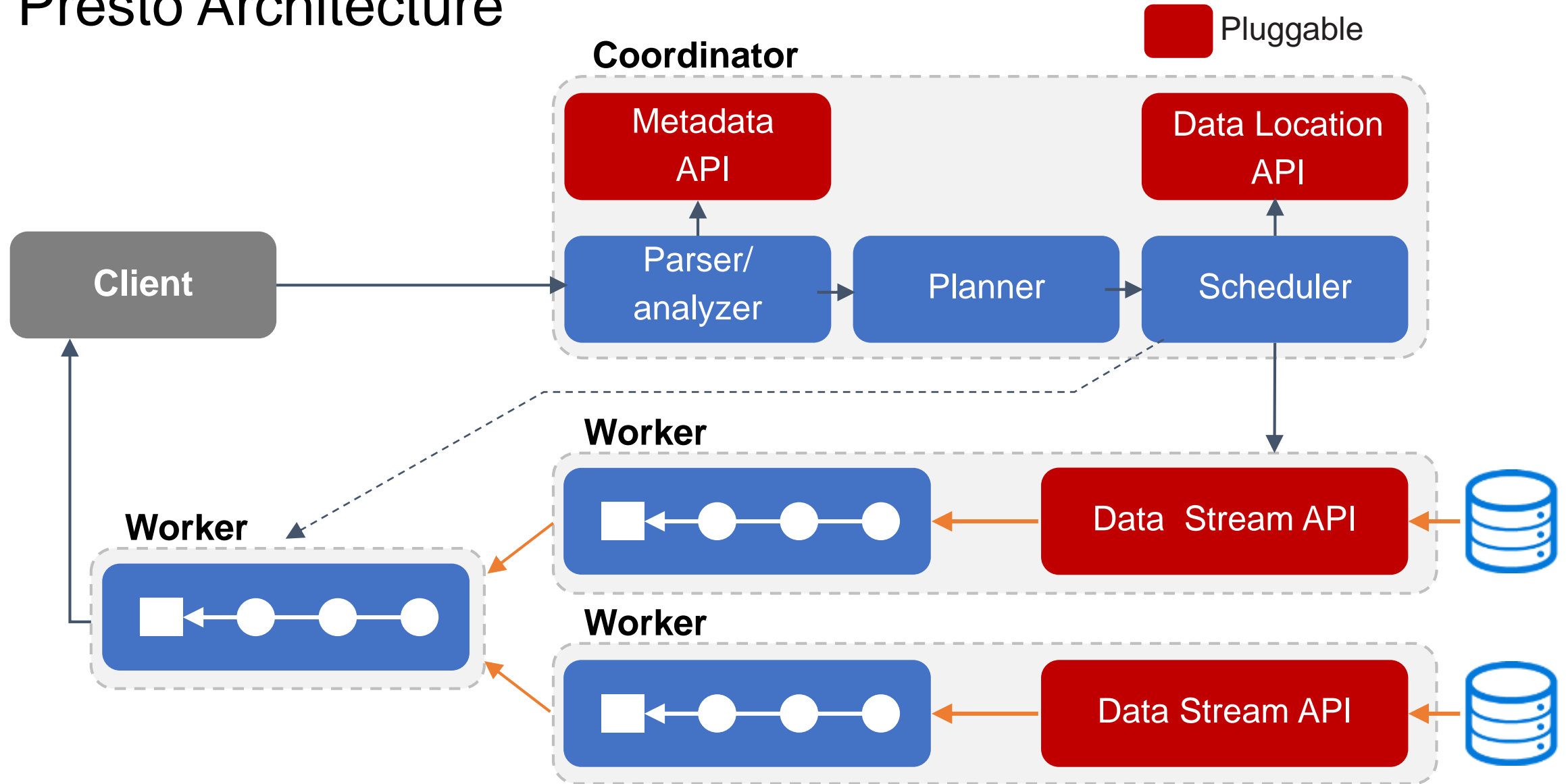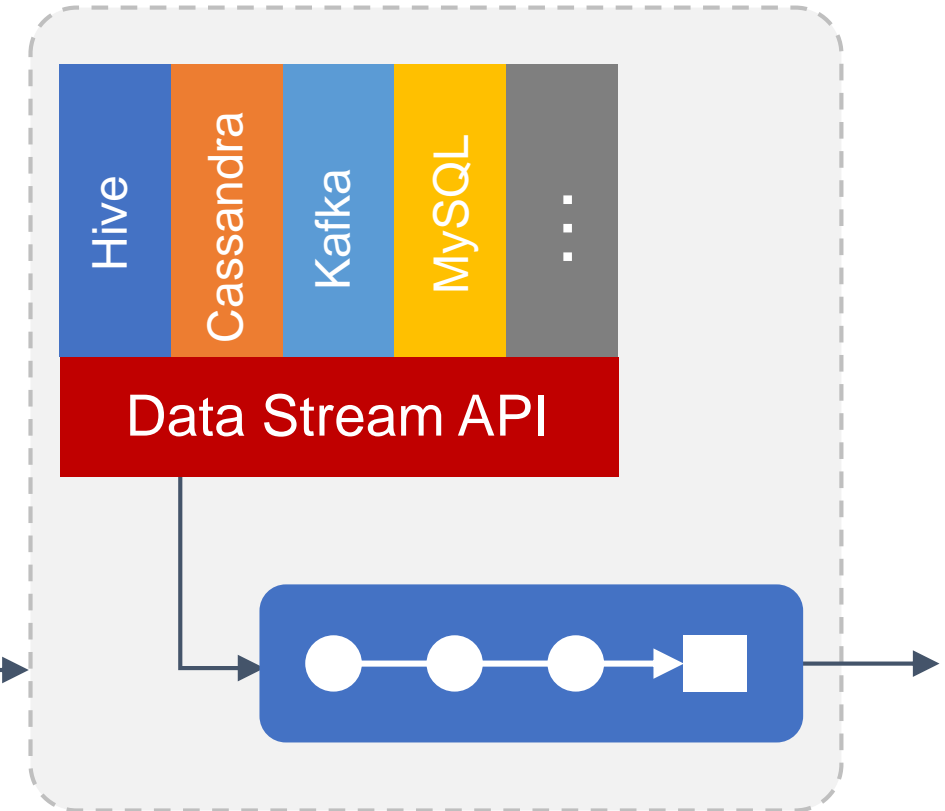
Presto is not a Database!

- Presto is a distributed query execution engine
    - Storage Independent

- Pluggable extensions
    - Connectors
    - Functions
    - Types
    - System access controllers
    - Resource group configuration managers
    - Event listeners
    - ...

- Built-in core functionalities
    - parser, execution, types, sql functions, monitoring

Presto Extensibility - Connector

Presto Connectors
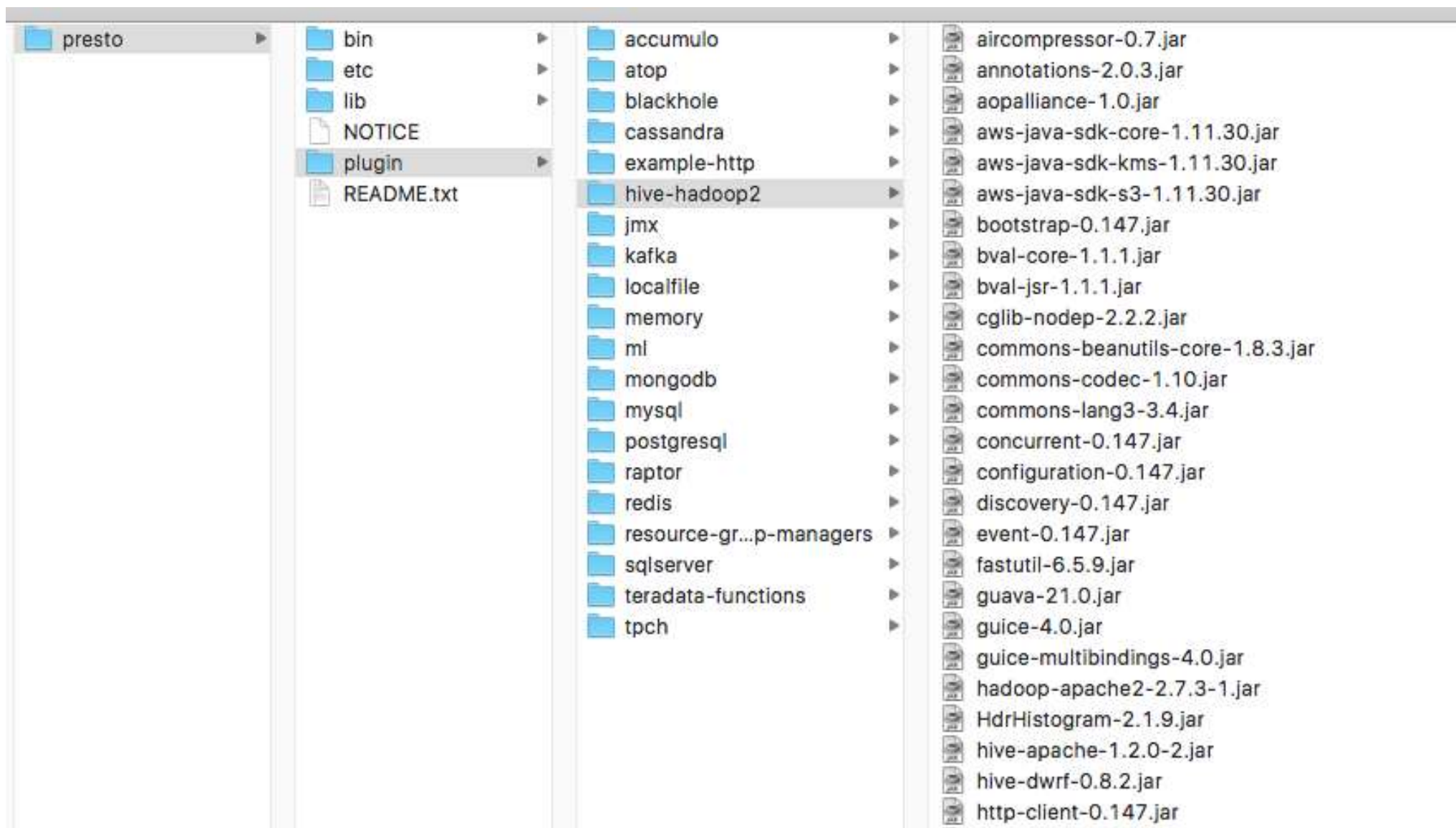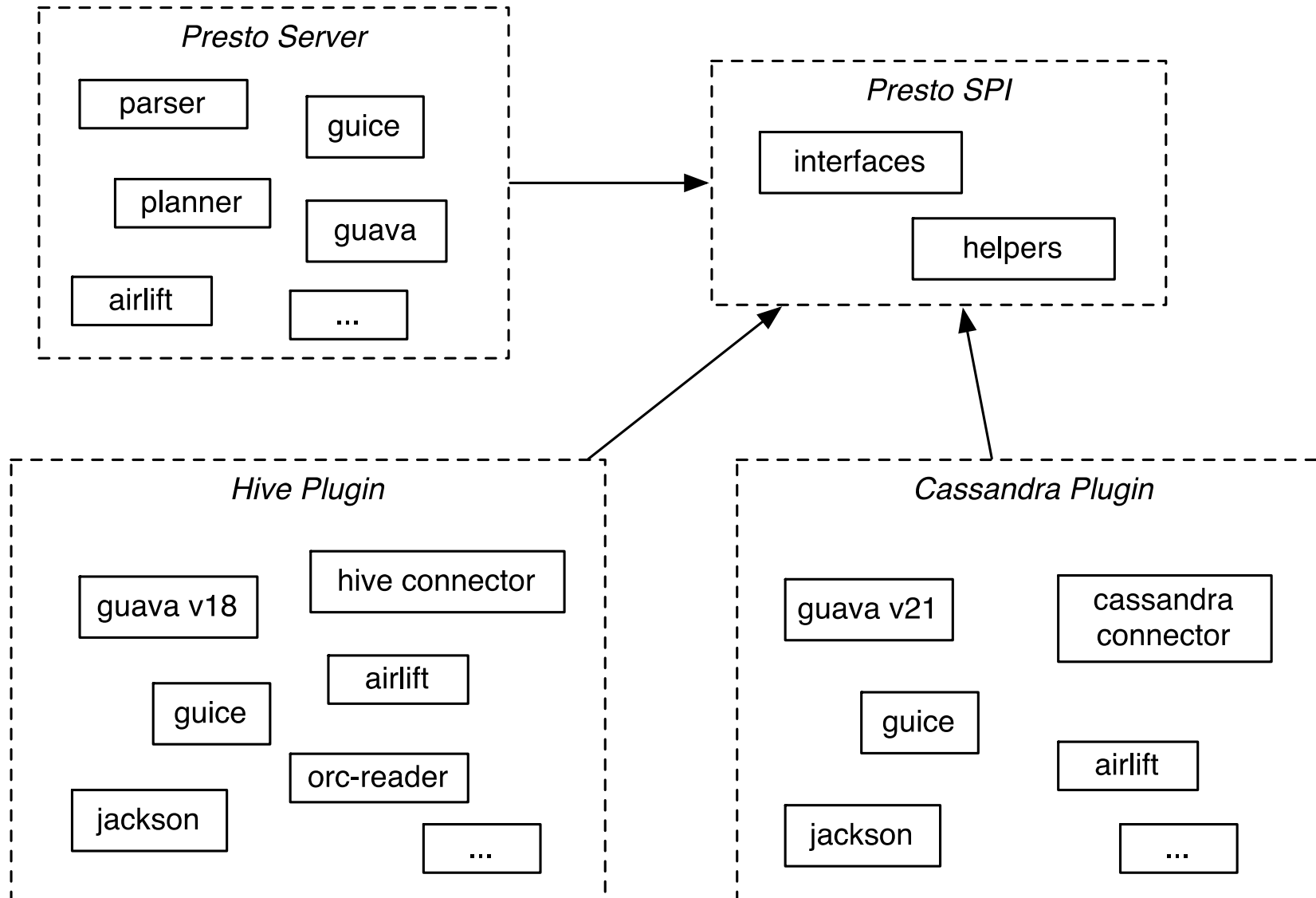
Plugins

Classloader Isolation

Plugin Interface

```java
public interface Plugin
{

  default Iterable<ConnectorFactory> getConnectorFactories() { return emptyList(); }

  default Iterable<Type> getTypes() { return emptyList(); }
  default Set<Class<?>> getFunctions() { return emptySet(); }
  default Iterable<EventListenerFactory> getEventListenerFactories() { return emptyList(); }
  ...
}
```
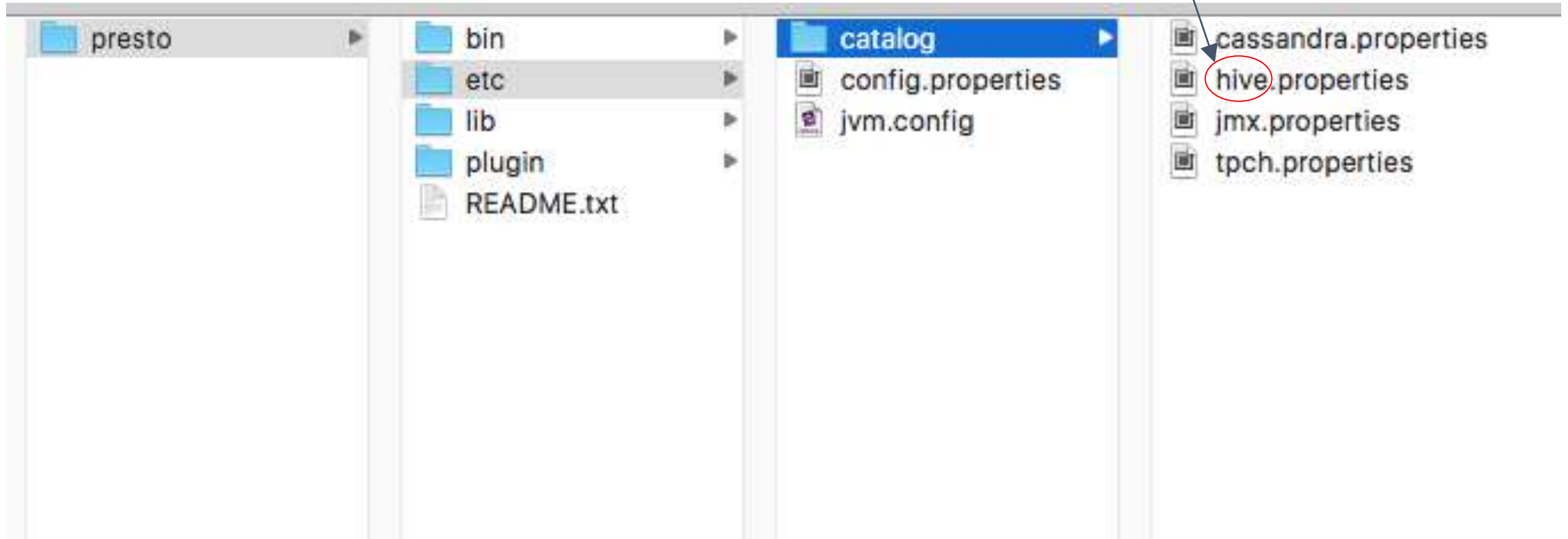
Connector Configuration

- Catalog namespace owned by connector

Catalog Name

Connector Configuration

```
connector.name=hive-hadoop2

hive.metastore.uri=thrift://hadoop-master:9083

hive.metastore.authentication.type=KERBEROS
hive.metastore.service.principal=hive/hadoop-master@EXAMPLE.COM
hive.metastore.client.principal=hive/_HOST@EXAMPLE.COM
hive.metastore.client.keytab=/etc/presto/conf/hive-presto-master.keytab

hive.hdfs.authentication.type=KERBEROS
hive.hdfs.impersonation.enabled=true
hive.hdfs.presto.principal=presto-server/_HOST@EXAMPLE.COM
hive.hdfs.presto.keytab=/etc/presto/conf/presto-server.keytab
```

Query Analysis

```sql
SELECT custkey, count(*)
FROM hive.tpch.orders
WHERE orderstatus = 'O'
GROUP BY custkey
```
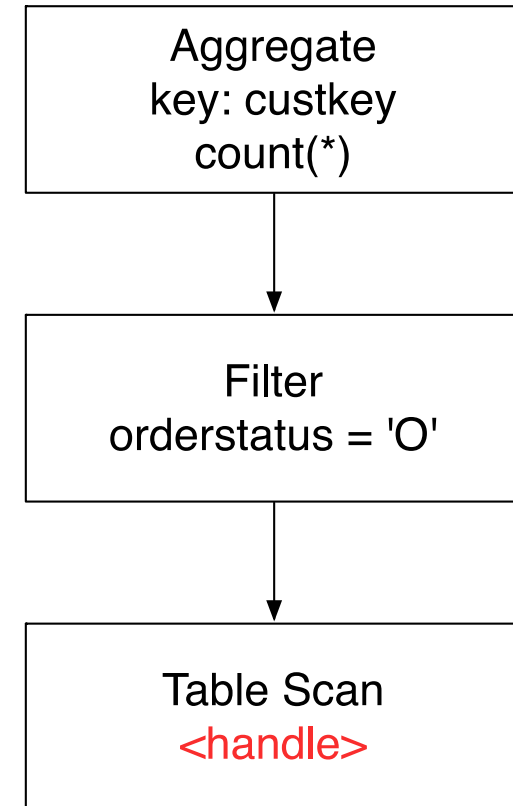
getTableHandle(...)
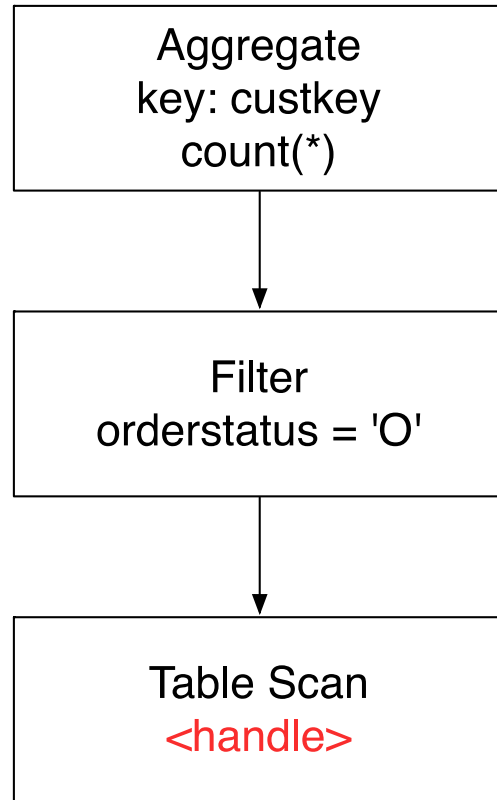
Table Handle

getTableMetadata(...)

```
orderkey BIGINT,
custkey BIGINT,
orderstatus VARCHAR(1),
...
```

Query Planning

```sql
SELECT custkey, count(*)
FROM hive.tpch.orders
WHERE orderstatus = 'O'
GROUP BY custkey
```

→

Aggregate
key: custkey
count(*)

Filter
orderstatus = 'O'

Table Scan

Query Execution

```
┌─────────────────────┐
│     Aggregate       │
│    key: custkey     │
│     count(*)        │
└─────────────────────┘
          │
          ▼
┌─────────────────────┐
│      Filter         │
│  orderstatus = 'O'  │
└─────────────────────┘
          │
          ▼
┌─────────────────────┐
│    Table Scan       │
│     <handle>        │
└─────────────────────┘
```

Split
- Handle to logical chunk of a table
- Attributes
    - Remote access?
    - Location

getSplits(...) ⟶ Iterator<Split>
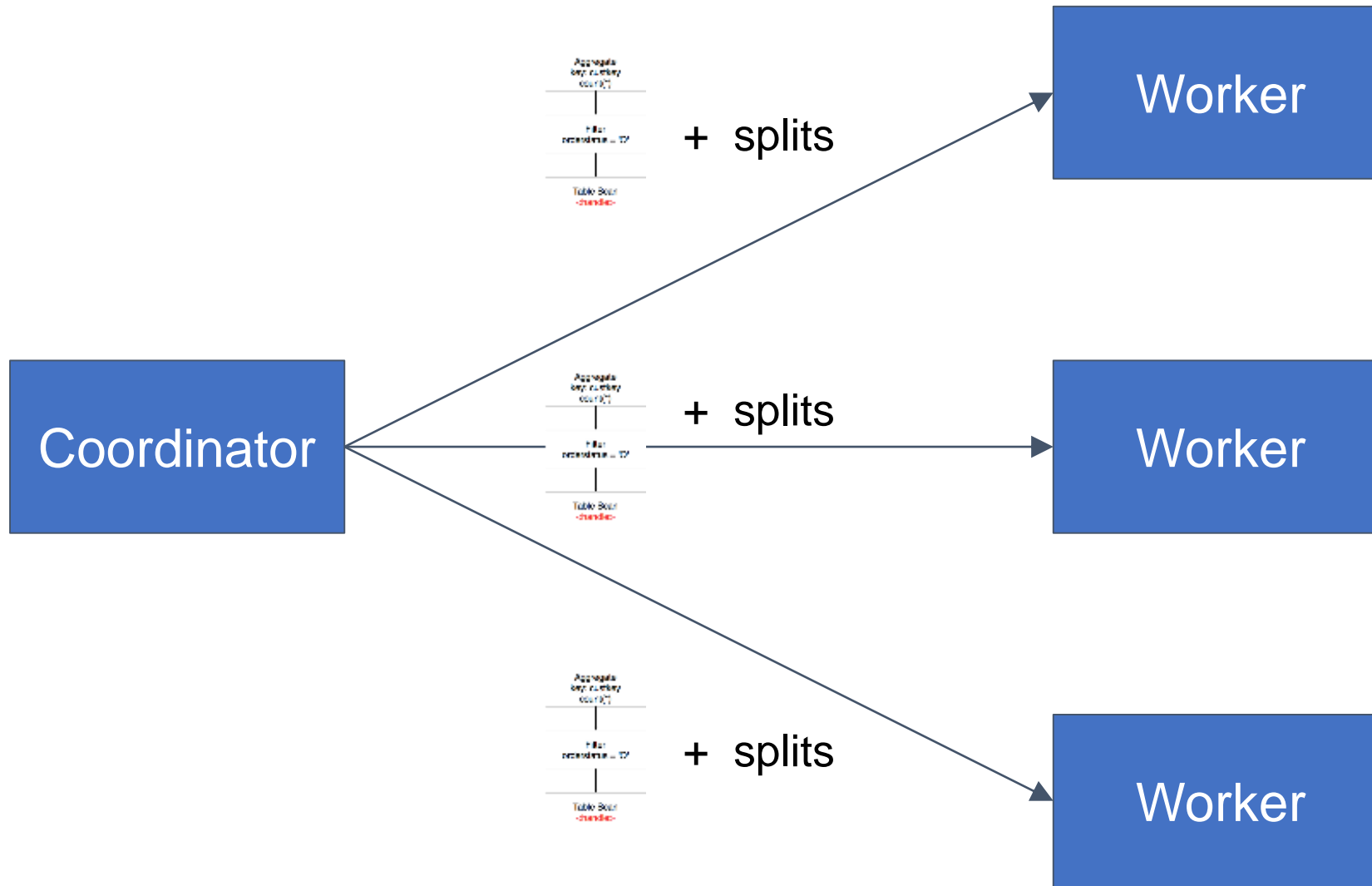
Query Execution

Query Execution
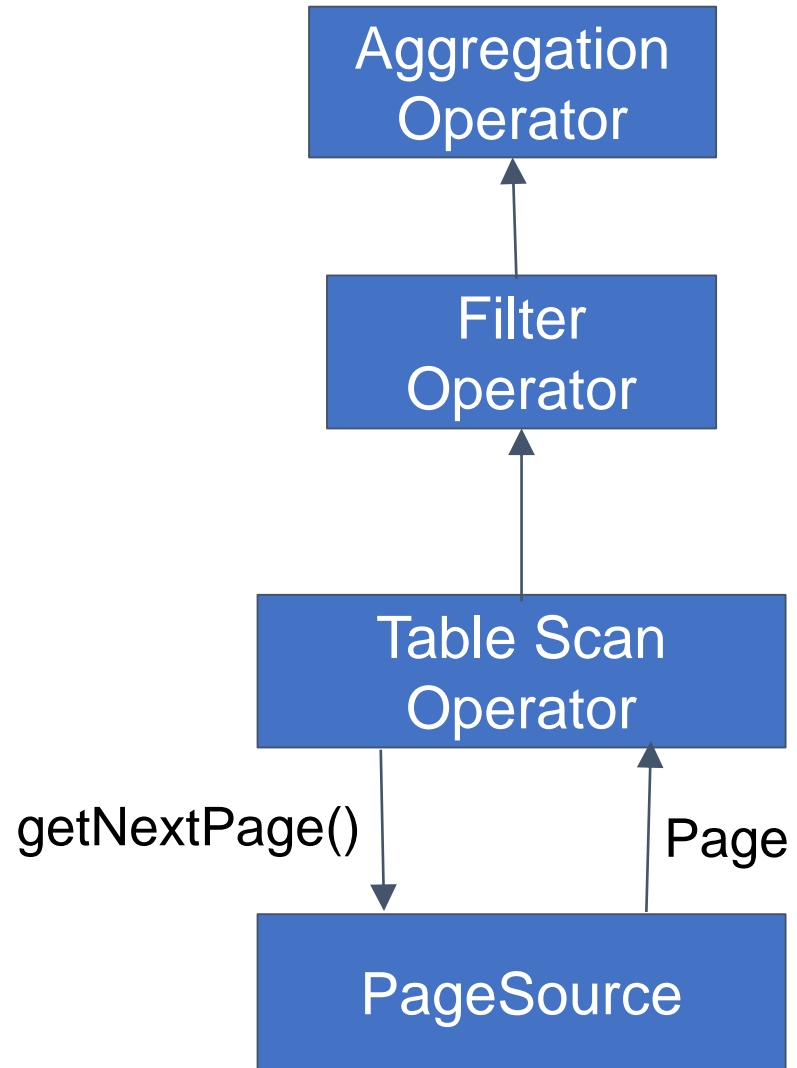
Presto Connectors @Facebook

- Hive connector
  - Warehouse (ad-hoc / batch)

- Raptor connector
  - Dashboards
  - Reporting backend for A/B testing framework

- Sharded MySQL connector
  - Reporting backend for user-facing products

- Other custom connectors for specialized data stores

Presto Connectors @Teradata Customers

- Teradata QueryGrid + Presto
  - Teradata, Hadoop, S3, Cassandra, RESTful

- Customer Use Cases
  - Recent sales data in **Teradata** needs to be joined with archived sales data that resided in **Hadoop**
  - Hadoop user using **Presto** needs to access pre-computed financial record in **Teradata**
  - Existing supplier data that is in Teradata is joined with archived product data that resides in **Amazon S3**

Teradata QueryGrid (powered by Presto)
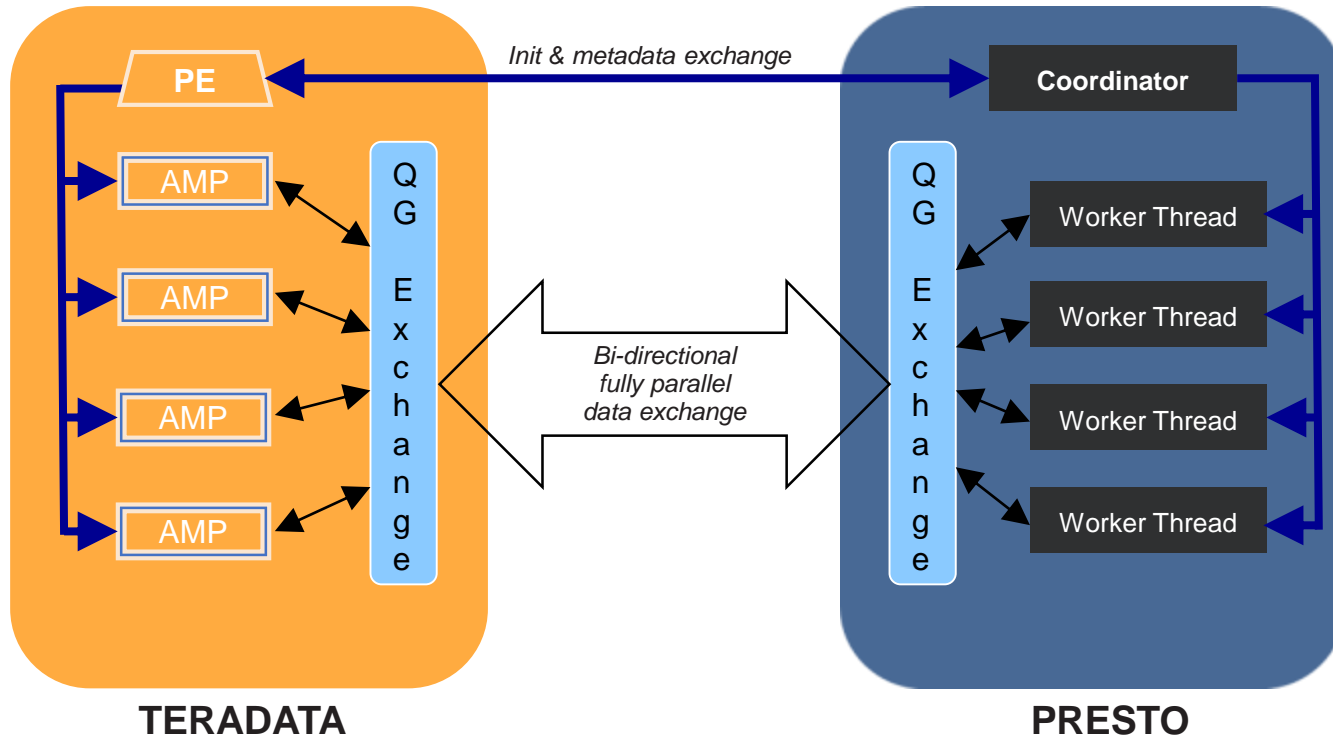
TERADATA

PRESTO

- **Key features:**
  - Low latency
  - High performance
  - Concurrency
  - Pushdown
  - Data conversion
  - Compression
  - Efficient CPU usage

Teradata QueryGrid SQL Examples

Teradata query joining data from Hadoop via Presto:

```
SELECT * FROM websales_current UNION ALL SELECT * FROM
websales_archive@presto;
```

Presto query joining data in Teradata:

```
SELECT * FROM td.sales.websales current UNION ALL SELECT
* FROM hive.sales.websales archive;
```

Conclusions

- Presto Connector API is expressive

- 3rd Party data source is 1st class citizen

- Single ANSI SQL to rule them all
  - Use BI tools on data which is not BI friendly

- Rapid data integration

Write your own connector!

- Issue SQL to GitHub!
    - https://developer.github.com/v3/
    - `SELECT count(*) FROM prestodb.presto.stargazers;`

- Connector Example
    - https://github.com/prestodb/presto/tree/master/presto-example-http

- Documentation
    - https://prestodb.io/docs/current/develop.html

Additional Resources

- Website
  - www.prestodb.io

- Presto Users Groups
  - www.groups.google.com/group/presto-users

- GitHub:
  - www.github.com/prestodb/presto
  - www.github.com/Teradata/presto (Teradata's development "fork")