Pallavi Singh
Software Consultant
Knoldus Software LLP

Geetika Gupta
Software Consultant
Knoldus Software LLP

Agenda:

- Query Engine
- Query Engine vs Database
- Big data Analytics
- Interactive Analytics
- What is Presto?
- Brief History of Presto
- Presto users in Production
- Architecture of Presto
- Connectors
- Presto vs Hive
- Setup
- Setting Up a Cluster
- Demo

knólˣ

Query Engine

- A software component positioned on the top of a data server that is in charge of implementing query functionalities against DB data and providing answers to users/applications.

- Query Engine is responsible for translating user queries into efficient data retrieval and processing operations, as well as executing these operations on single or multiple nodes in order to find query answers.

- The efficiency of query engine is primarily counted on its speed and insight seeing capabilities.

Query Engine v/s Databases

- Databases often bundle the query engine and storage into a single tightly coupled system.
- This coupling provides certain optimizations but hampers the flexibility and agility.
- Query engines decouple the storage and engine. They operate independent of the data storage.

Big data Analytics

- Big data analytics are techniques that are used to analyze large datasets in order to **extract patterns, trends, correlations and summaries**.
- The insights drawn by running big data analytics depend primarily on the capabilities of the underlying query engine.
- Use cases of Big Data Analytics are network monitoring, risk management and fraud detection, transaction cost analysis and pricing analytics, algorithmic trading, intelligence and surveillance

Interactive Analytics

- The ability to run complex queries across complex data landscapes where we have the complexity intelligence to visit thousands of nodes in real time.
- Current tools are able to synthesize multiple streams of rapidly flowing data and perform complex operations on them using the traditional ETL methods. But the missing lead is ability to explore the data by issuing different types of queries and drilling down until a specific insight is uncovered.

  Example :

  Impala and Redshift, considered to be some of the world’s fastest big data compute engines, took an average of 1.5 minutes to refresh the Tableau dashboard for a mere 3 billion rows. Which is far from real-time by any measure, and does not allow analysts to perform interactive exploration of the data.

Presto

- Presto is an open source distributed SQL query engine for running interactive analytic queries against data sources of all sizes ranging from gigabytes to petabytes.
- Developed at Facebook, later contributed to open source
- Hive, Impala, HAWQ, IBM Big SQL, Drill, Tajo, Pig,DryadLINQ, Jaql
- Presto is designed to handle data warehousing and analytics: data analysis, aggregating large amounts of data and producing reports.
- Presto is ANSI SQL Compliant
- Presto has all the major build in core functionalities – like parser, sql functions and monitoring capabilities.

knólx

Key Differentiators

- Support for Cross platform query capabilities
- Support for federated queries
- Performance and Scale
- Used in production at many well known web-scale companies
- Storage independent
- A vast list of pluggable extensions

knólx

What Presto is not !

- Presto is not a database
- Presto is not storage dependent
- Presto understands SQL but does not provide features of standard databases.
- Not a general purpose relational database
- Not a replacement for MySQL, PostgreSQL or Oracle
- It is not designated to handle OLTP

History of Presto

Presto = Performance

- Data stays in memory during execution and is pipelined across nodes MPP-style
- Vectorized columnar processing
- Presto is written in highly tuned Java
  - Efficient in-memory data structures
  - Very careful coding of inner loops
  - Bytecode generation

In Production

knólx

facebook.

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

NETFLIX

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

knólx

twitter

-200+ nodes on-premises
-Parquet nested data

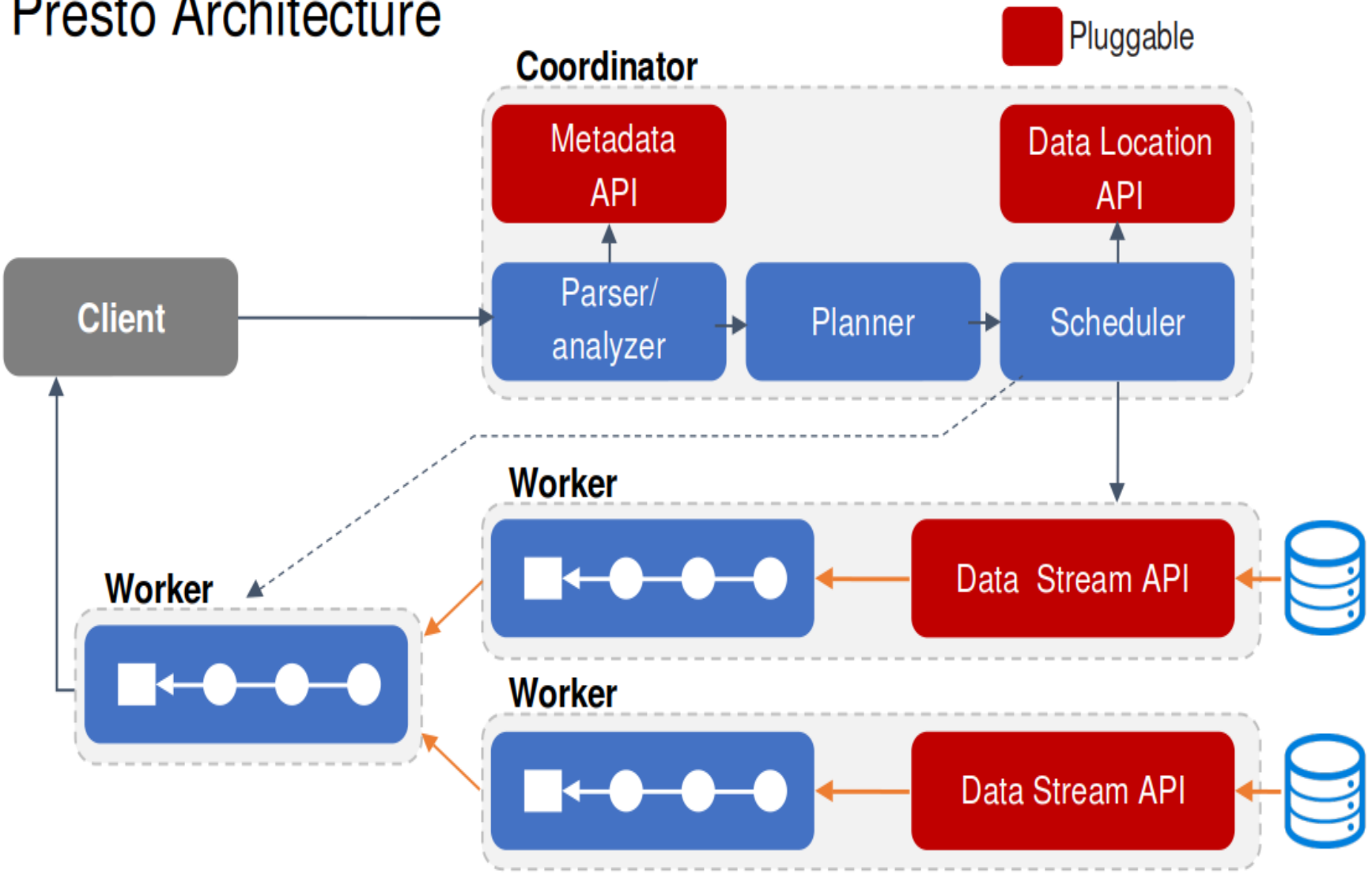
UBER

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

-120+ nodes in AWS
-2PB is S3 and
-200+ users

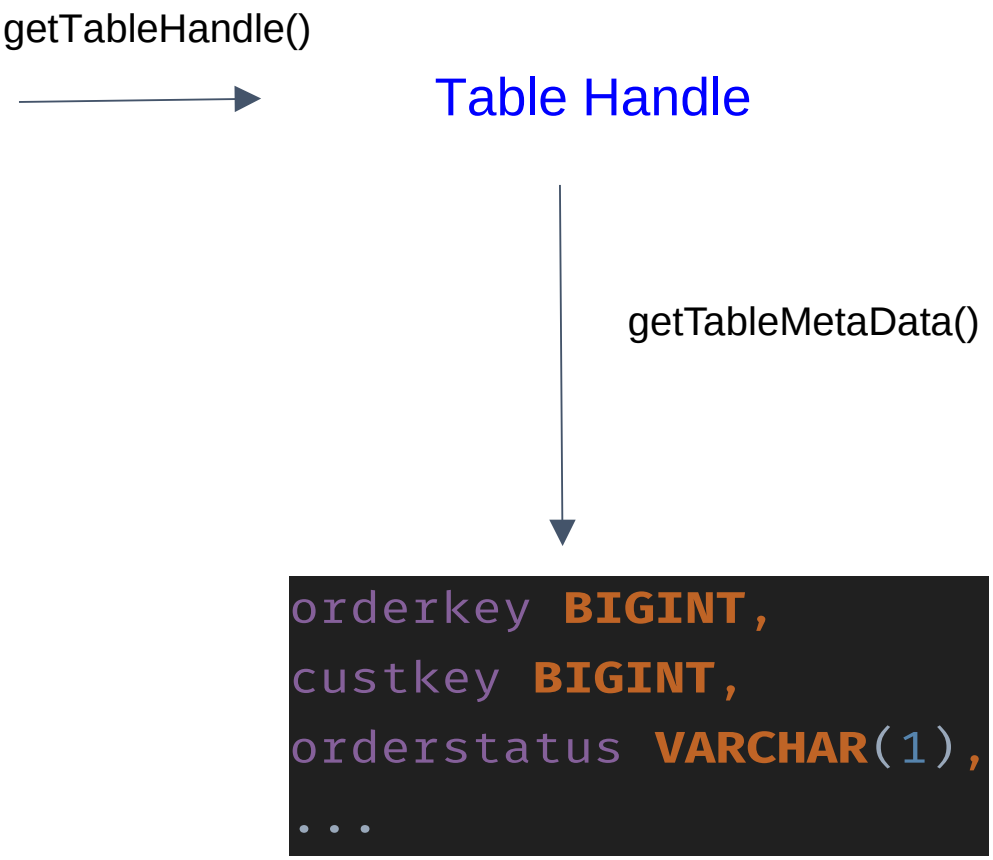
FINRA

Presto Architecture



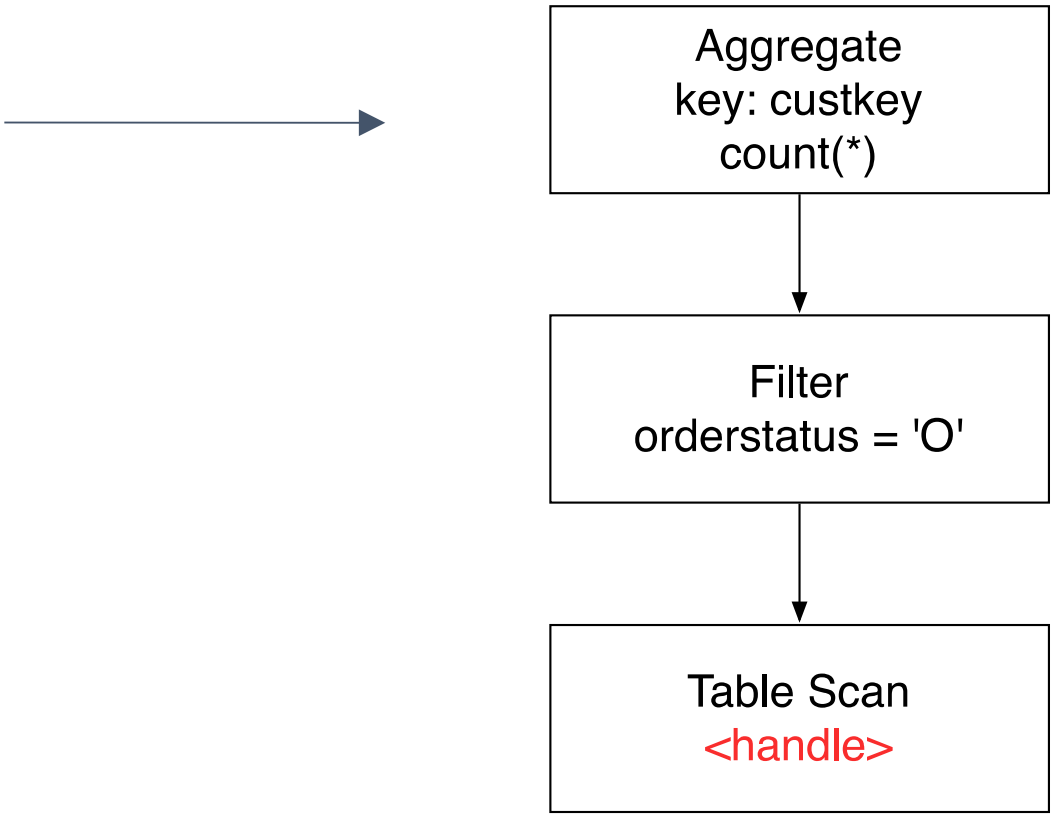
Query Analysis

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

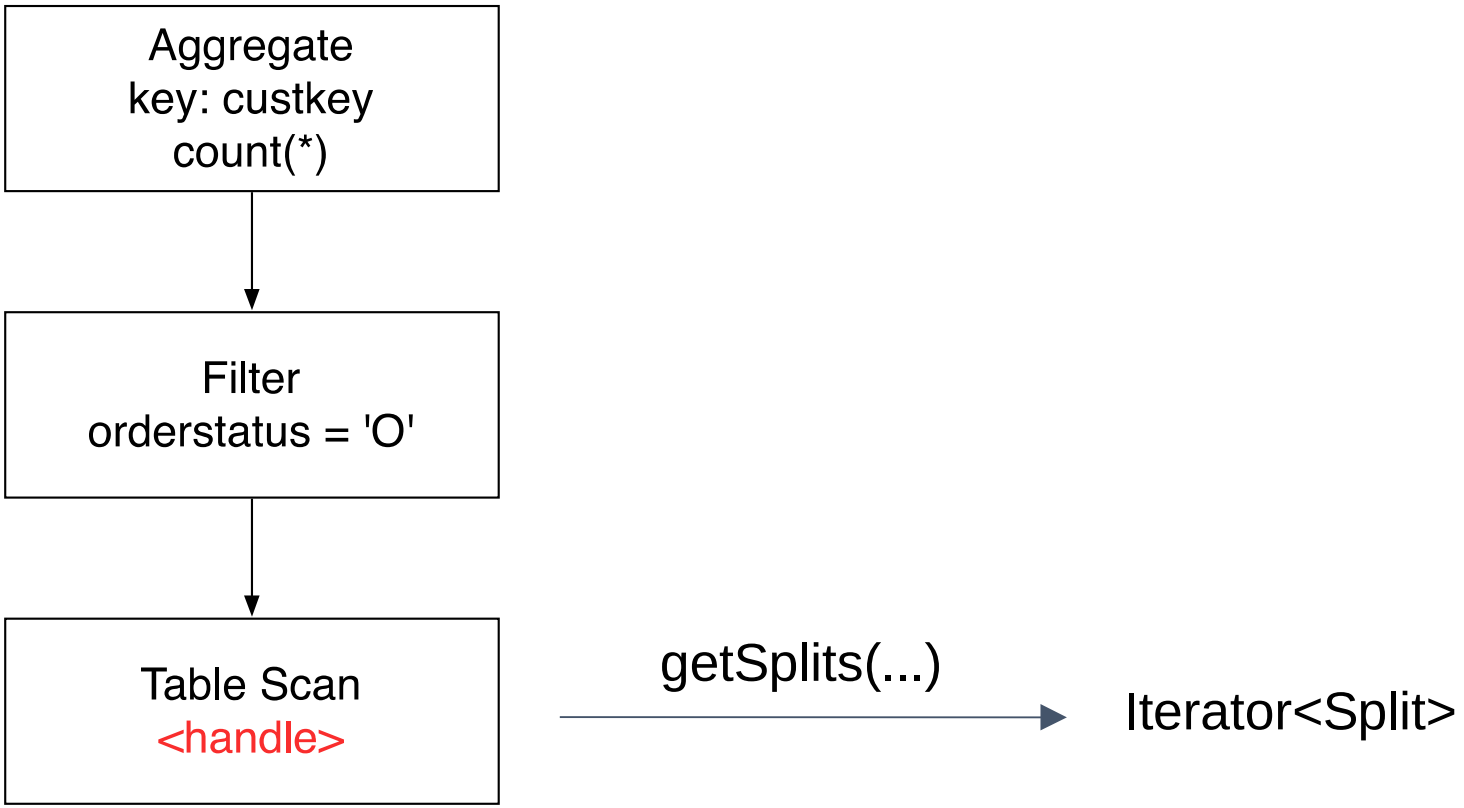


Query Planning

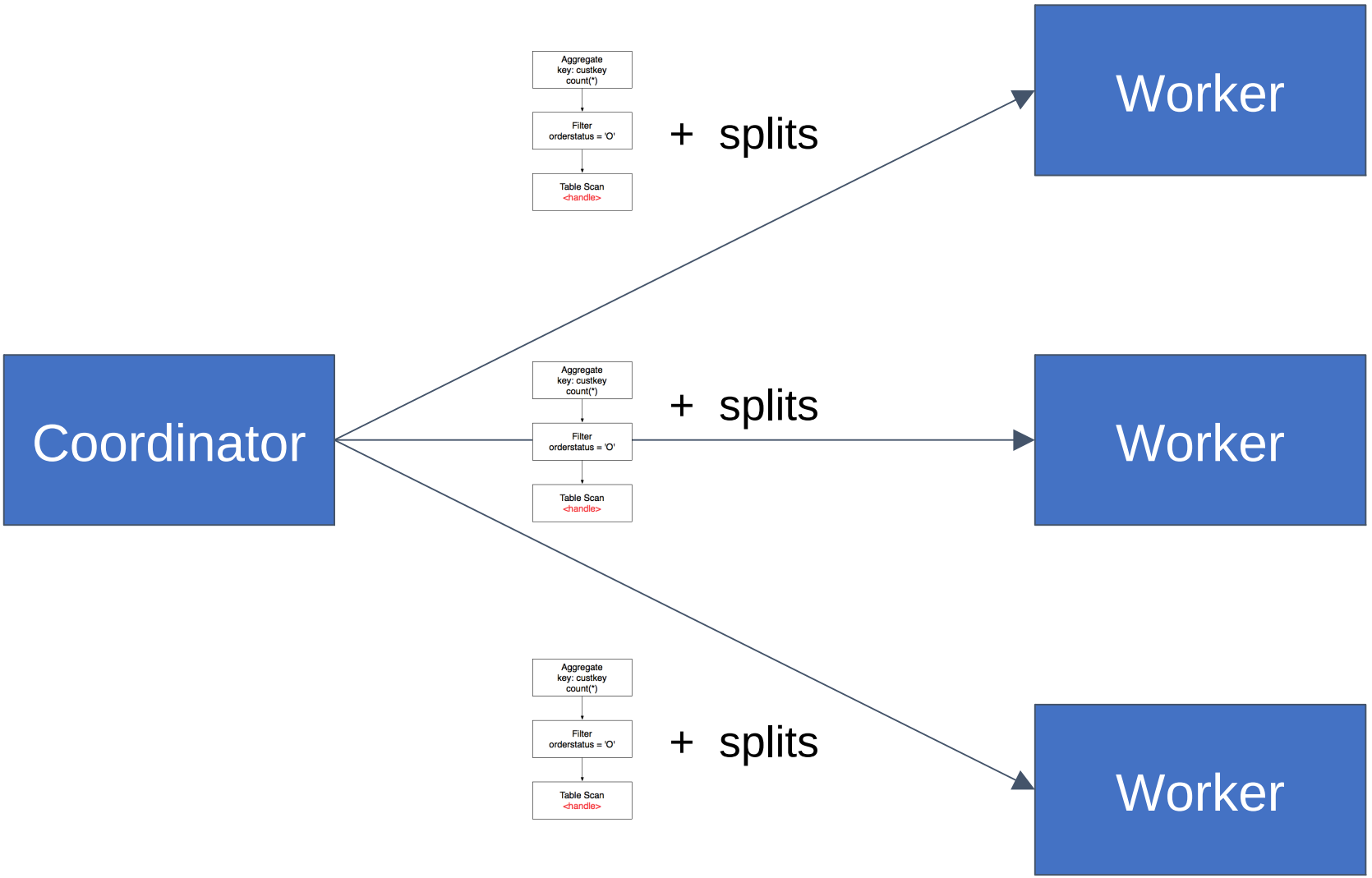
```
SELECT custkey, count(*)
FROM hive.tpch.orders
WHERE orderstatus = 'O'
GROUP BY custkey
```



Query Execution

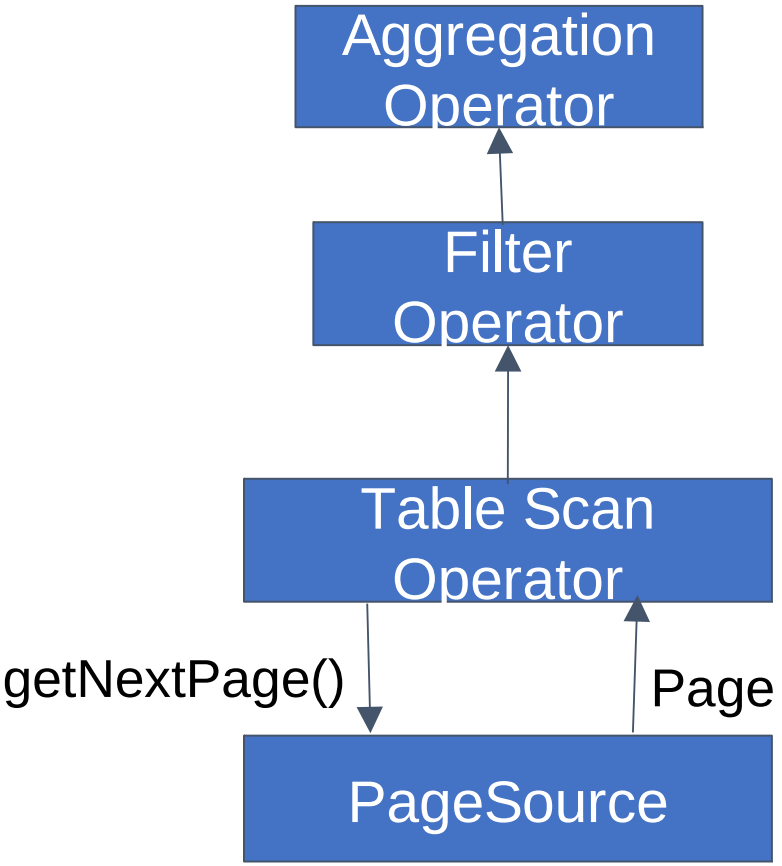


Query Execution



knólx

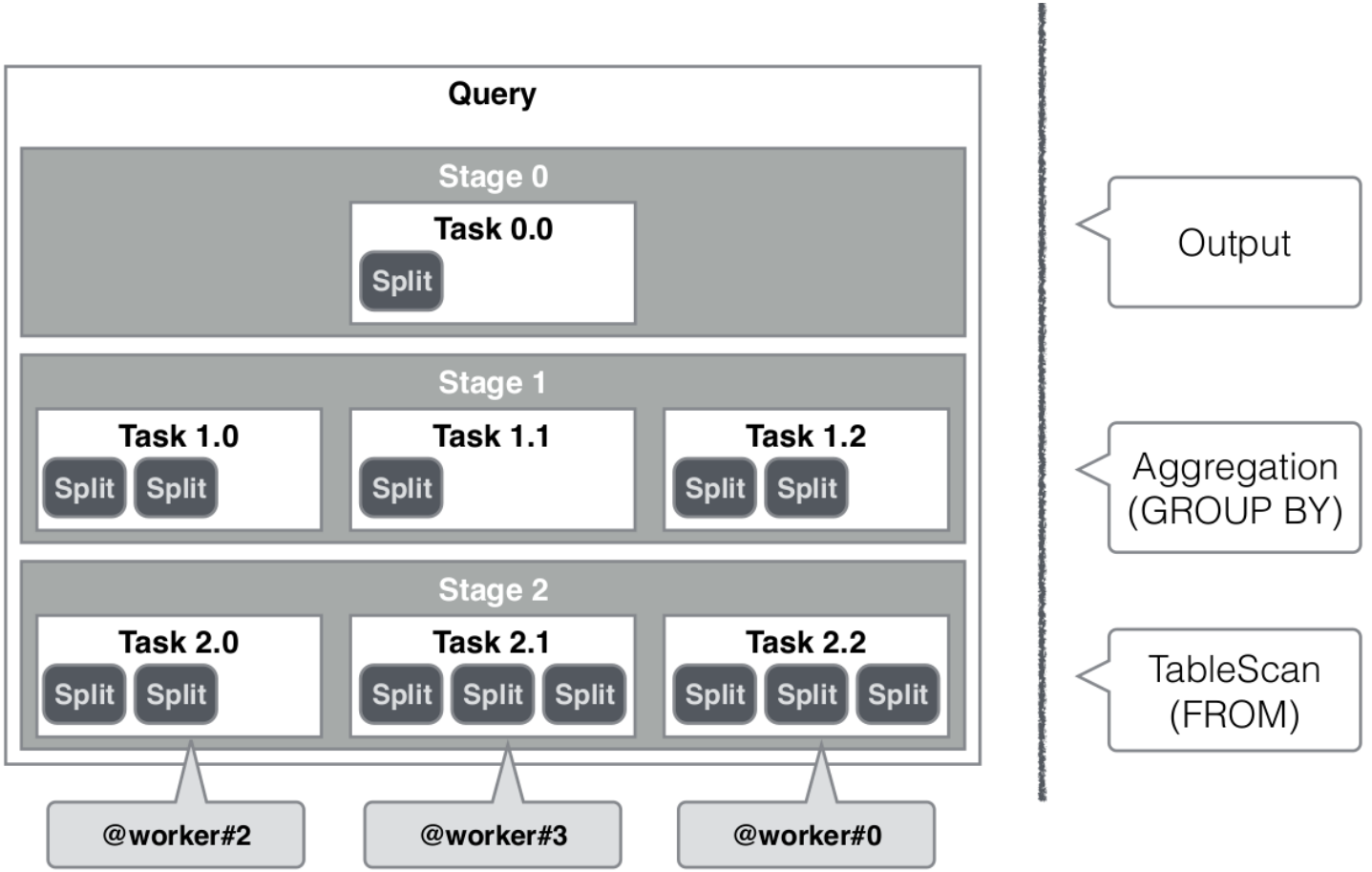
Query Execution



Query Execution Model

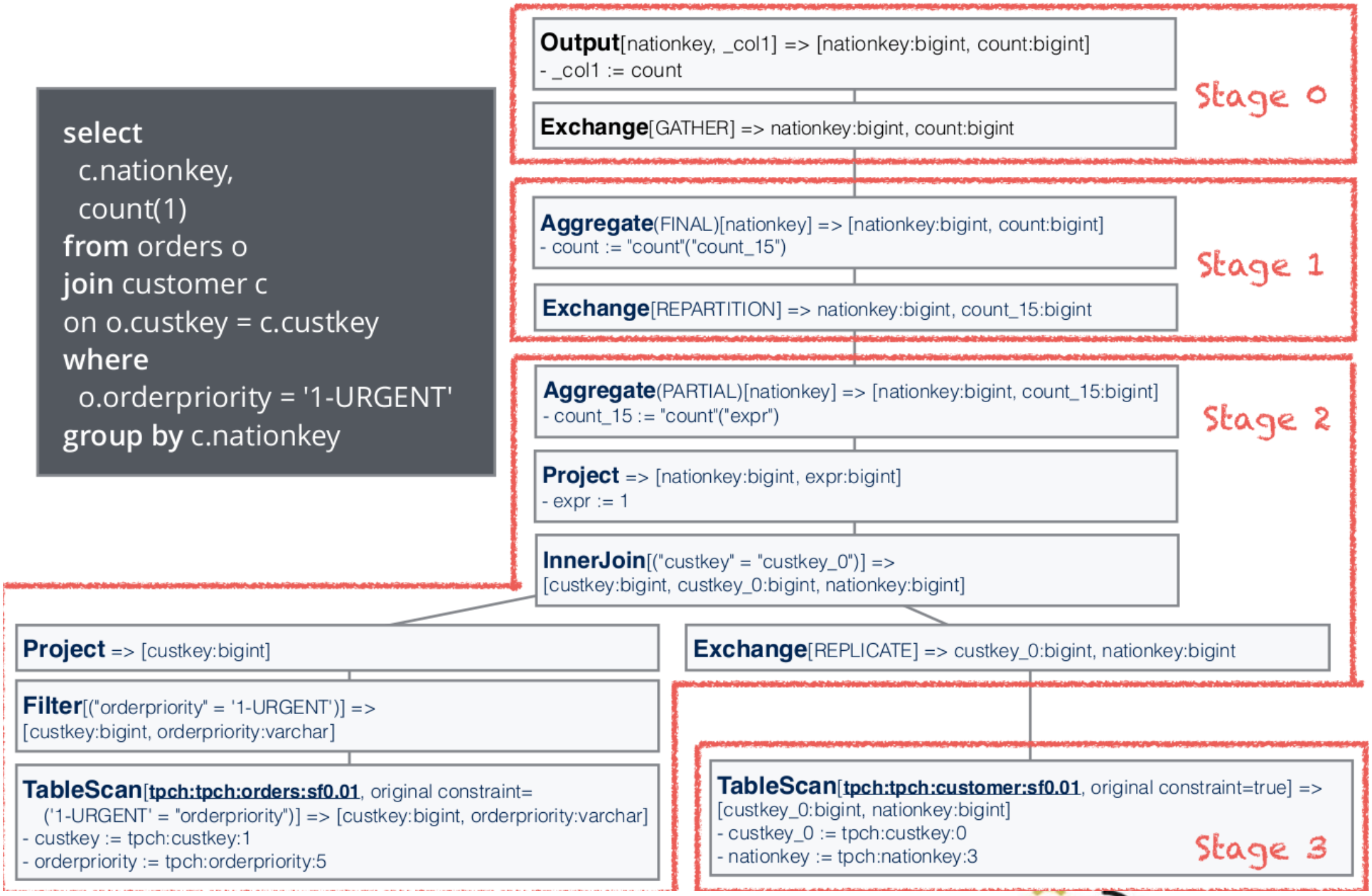
- **Query**: When Presto parses a statement, it converts it into a query and creates a distributed query plan which is then realized as a series of interconnected stages running on Presto workers. A query refers to the configuration and components instantiated to execute sql statement
- **Stage:** When Presto executes a query, it does so by breaking up the execution into a hierarchy of stages.
- **Tasks:** Stage is implemented as a series of tasks distributed over a network of Presto workers. Tasks act upon or process splits.
- **Split:**Tasks operate on splits which are sections of a larger data set. Stages at the lowest level of a distributed query plan retrieve data via splits from connectors.

Query Execution Model



knólx

```
select
  c.nationkey,
  count(1)
from orders o
join customer c
on o.custkey = c.custkey
where
  o.orderpriority = '1-URGENT'
group by c.nationkey
```



Connectors

- Connectors are storage plugins that allow access to different data sources.
- Need to provide interfaces for fetching metadata, getting data locations and accessing the data.
- Presto provides a wide variety of plugins to access various data sources.

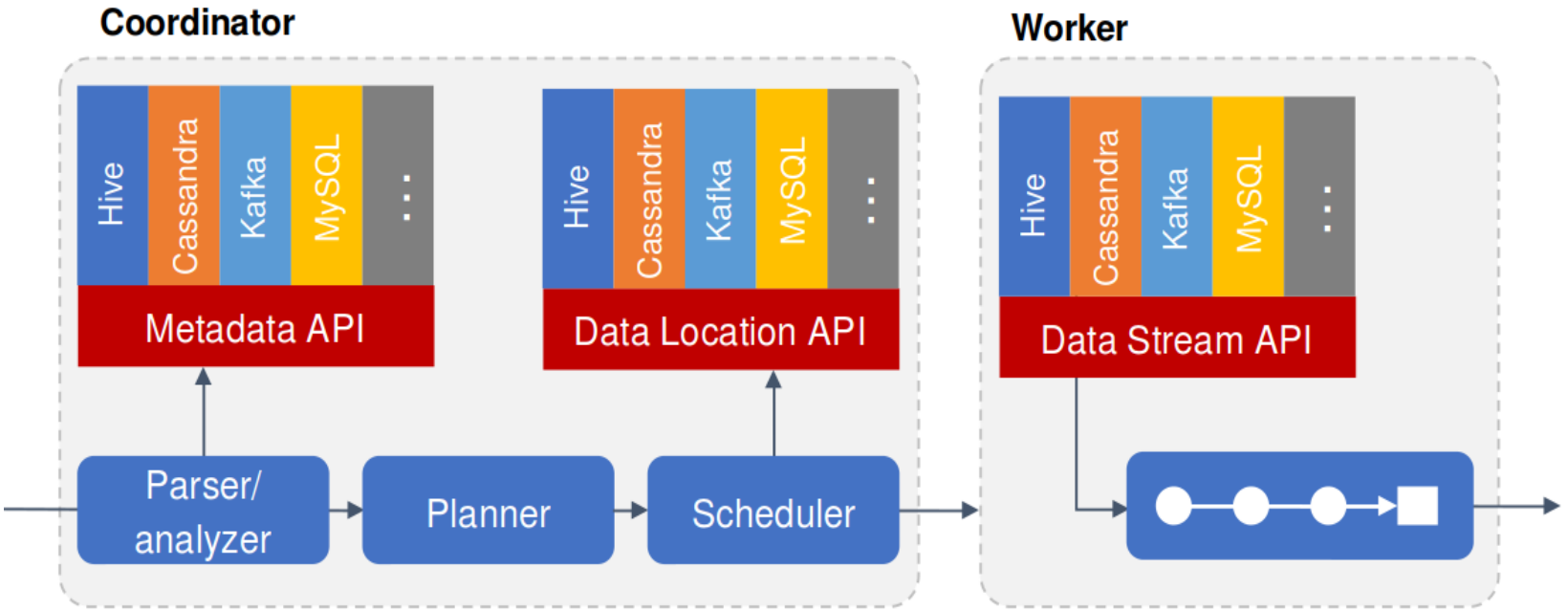
Presto Extensibility – connectors

kafka

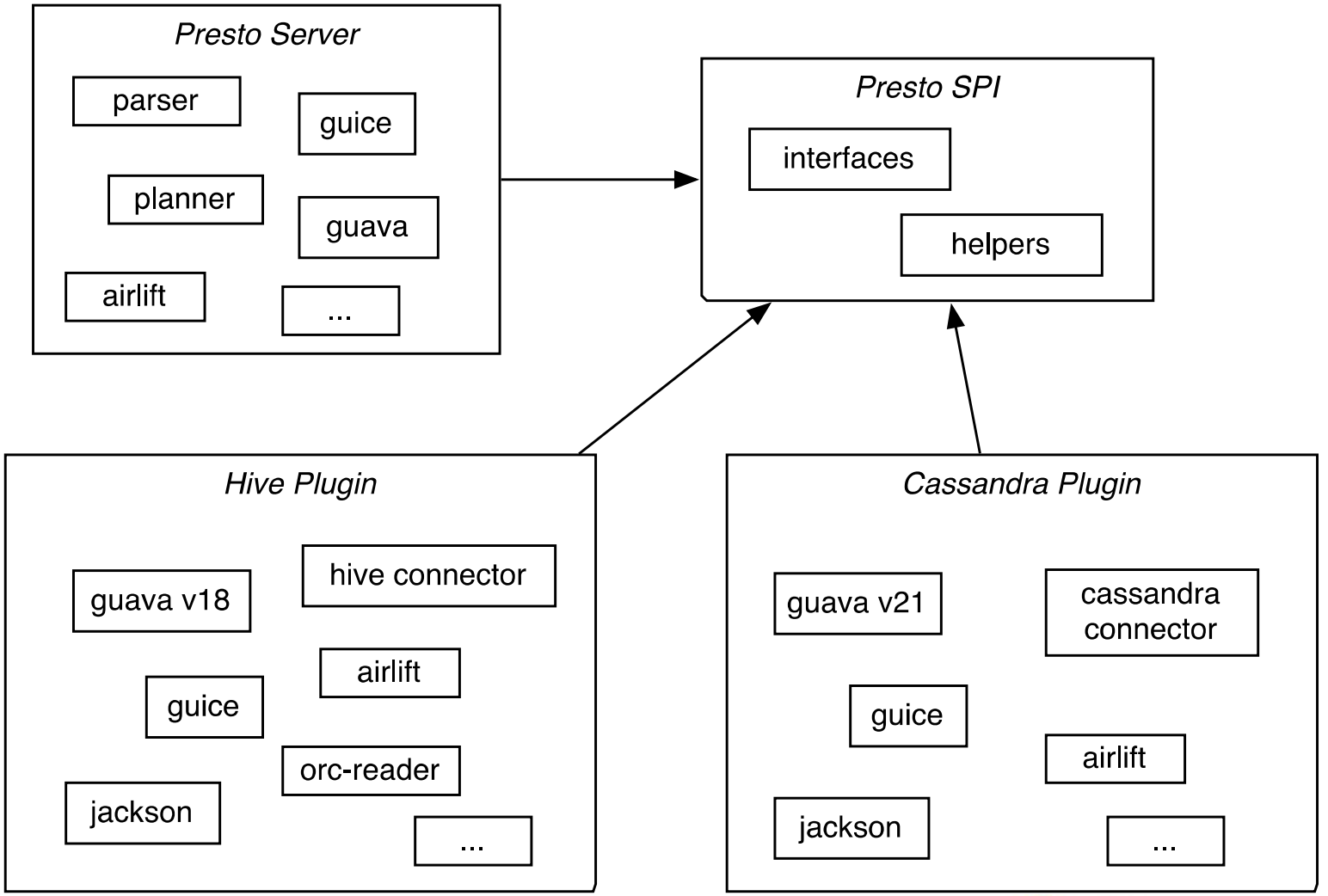
knólˣ

Presto Extensibility – connectors

- MetadataAPI provides information about the tables and the column types
- Data LocationAPI provides information related to physical location of blocks
- Data StreamAPI is responsible for fetching of data

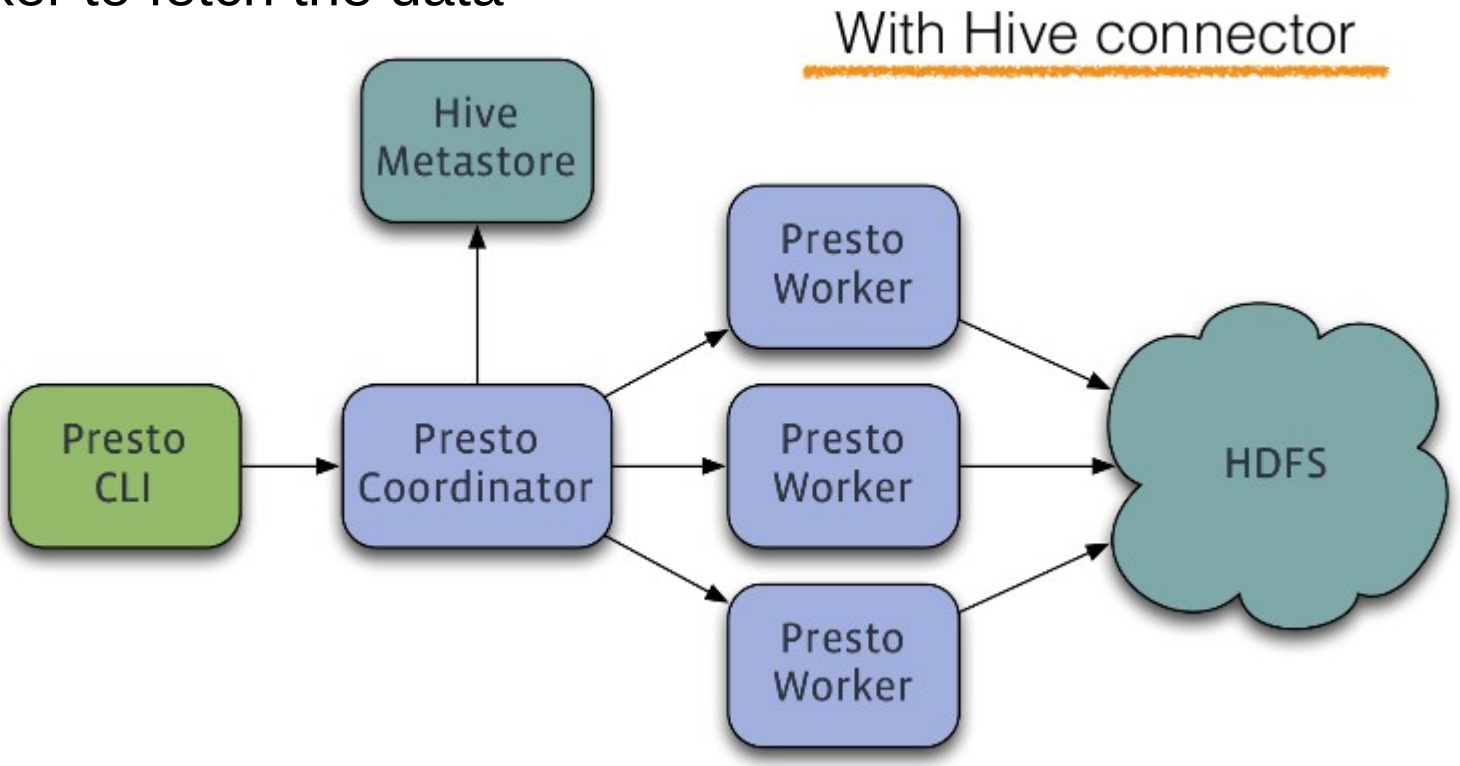


Connectors Architecture



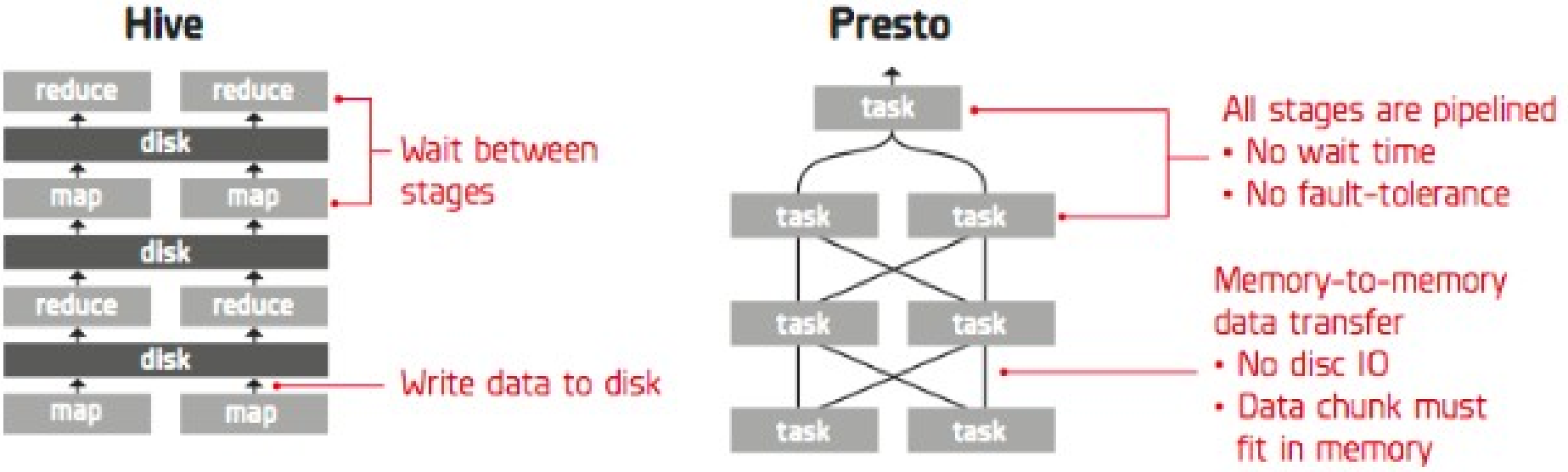
Architectural overview with Hive Connector

- The coordinator interacts with the hive metastore for Metadata information
- Scheduler performs pipeline execution, assigns work to the closest worker to fetch the data



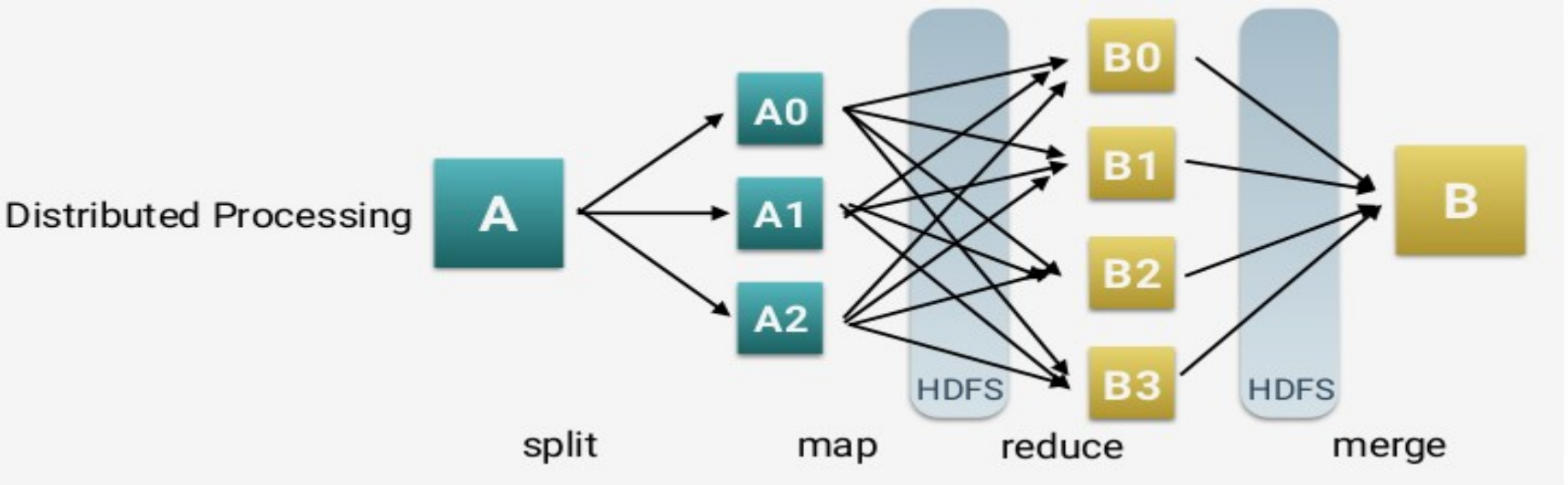
Uses Hadoop MR for execution	Pipelined execution model
Spills intermediate data to file system	Intermediate data in-memory
Can tolerate failures	Does not tolerate failures
Automatic join ordering	User specified join ordering
Can handle join of two large tables	One table needs to fit in memory
More data types	Limited data types
Very good for large Fact-to-Fact joins	Optimized for star schema joins (1 large Fact table and many smaller dimension tables)

Presto vs Hive



Hive

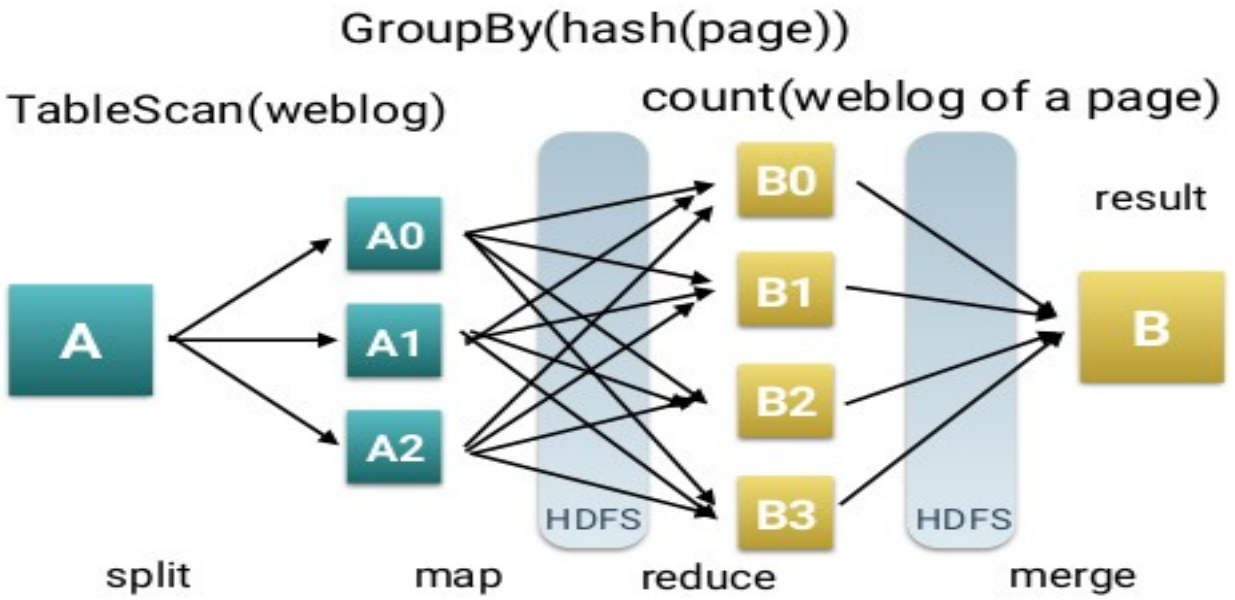
- Translates SQL into MapReduce(Hadoop) programs
- HDFS is used for storing intermediate results
- Provides fault-tolerance, but slow
- All stages run one after another



Hive Execution

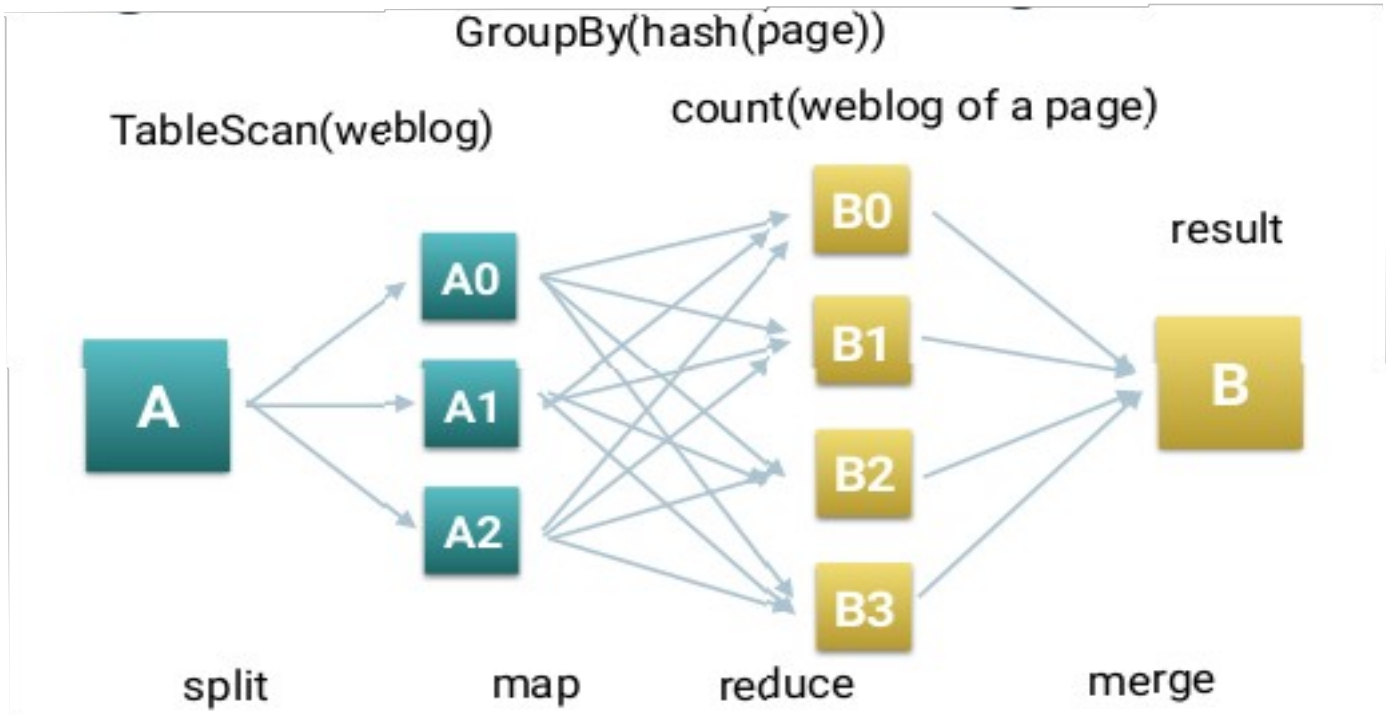
- Mapping SQL stages into MapReduce program

```
SELECT page, count(*) FROM weblog
GROUP BY page
```

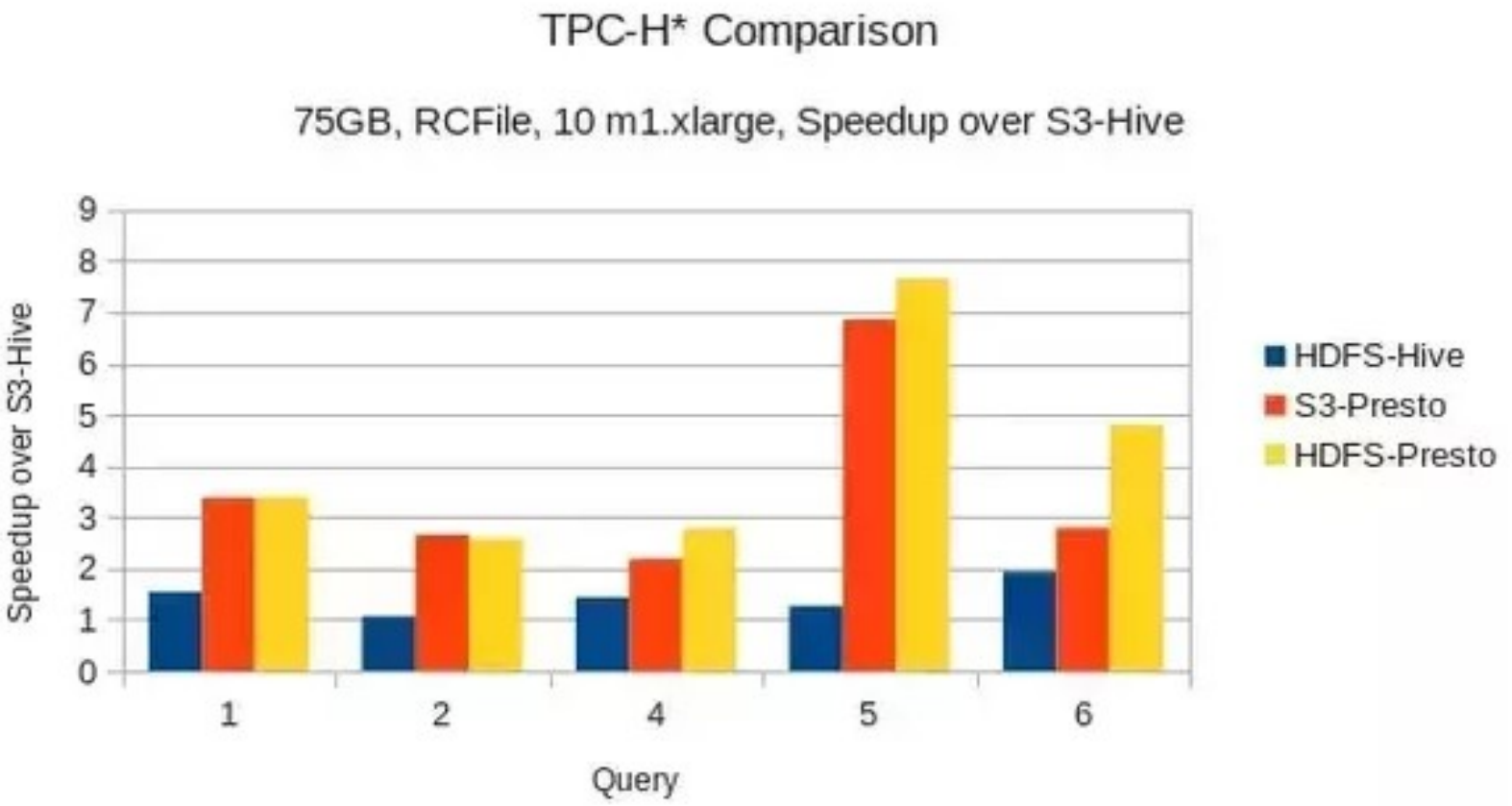


Presto Execution

- Uses HTTP for data transfer
- No intermediate storage like HDFS
- Can start all the stages at once and stream data from one stage to another.



Presto vs Hive



Presto shows a speed up of 2-7.5x over Hive and it is also 4-7x more CPU efficient than hive

knólx

Set up

- Download the Presto server tarball, presto-server-0.183.tar.gz, and unpack it.

Configuring Presto

Create an etc directory inside the installation directory. This will hold the following configuration:

Node Properties: environmental configuration specific to each node

JVM Config: command line options for the Java Virtual Machine

Config Properties: configuration for the Presto server

Catalog Properties: configuration for Connectors (data sources)

Set up

Node Properties : The node properties file, etc/node.properties, contains configuration specific to each node. The following is a minimal etc/node.properties:

```
node.environment=production
node.id=ffffffff-ffff-ffff-ffff-ffffffffffff
node.data-dir=/var/presto/data
```

JVM Config : The JVM config file, etc/jvm.config, contains a list of command line options used for launching the JVM. The following provides a good starting point for creating etc/jvm.config:

```
-server
-Xmx16G
-XX:+UseG1GC
-XX:G1HeapRegionSize=32M
-XX:+UseGCOverheadLimit
-XX:+ExplicitGCInvokesConcurrent
-XX:+HeapDumpOnOutOfMemoryError
-XX:+ExitOnOutOfMemoryError
```

Set up

Config Properties

The config properties file,etc/config.properties, contains the configuration for the Presto server. Every Presto server can function as both a coordinator and a worker. The following is a minimal configuration for a single machine for testing that will function as both a coordinator and worker:

```
coordinator=true
node-scheduler.include-coordinator=true
http-server.http.port=8080
query.max-memory=5GB
query.max-memory-per-node=1GB
discovery-server.enabled=true
discovery.uri=http://example.net:8080
```

Set up

Catalog Properties : Catalogs are registered by creating a catalog properties file in the etc/catalog.

- **For hive catalog** : Create etc/catalog/hive.properties to mount the hive-hadoop2 connector as the hive catalog, replacing example.net:9083 with the correct host and port for your Hive metastore Thrift service:

```
connector.name=hive-hadoop2
hive.metastore.uri=thrift://example.net:9083
```

- **For postgresql catalog** : create a catalog properties file in etc/catalog named postgresql.properties

```
connector.name=postgresql
connection-url=jdbc:postgresql://example.net:5432/database
connection-user=root
connection-password=secret
```

knólx

Set up

Running Presto : To start presto server

```
bin/launcher run
```

Presto CLI : Download presto-cli-0.184-executable.jar, rename it to presto, make it executable with chmod +x, then run it:

```
./presto --server localhost:8080 --catalog hive --schema default
```

Cluster Setup

- In node.properties node.id must be unique for all the nodes.
- In config.properties the coordinator parameter must be true for the master node and false for other nodes
- If you want to use your master node for computation work you can set the property

  node-scheduler.include-coordinator=true

References

- Official Website : https://prestodb.io/
- https://github.com/prestodb/presto
- http://www.teradata.co.uk/products-and-services/Presto/Presto-Download
- https://github.com/Teradata/presto
- Installing and Running Presto
- Getting Introduced with Presto
- Connecting To Presto via JDBC

Thank You