PrestoDB Administration Fundamentals
Why, What and How?
About me

Ravishankar Nair

Chief Consultant at PassionBytes

Distributed Computing, Big Data, PrestoDB….

LinkedIn

Email: rshankarn@acm.org
1. Why PrestoDB?
Not one DB

As data increases, enterprises are selecting data stores according to the characteristics of use case – Polyglot Persistence (Martin Fowler)
Universal Adapter with Multi join capability

Consider each input as each data source. Output is a SQL interface to outside world. Additionally, you can join multiple inputs!
2. What is PrestoDB?
Driven by Community – Presto Foundation

High performance MPP SQL engine
- Interactive & Batch ANSI SQL queries
- Proven scalability and concurrency
- Distributed

Connector Based Approach
- Many default connectors, extensible
- Plugin Architecture

Community Driven under Linux Foundation (Presto Foundation)
Deploy on prem, cloud or hybrid
PrestoDB Users

https://github.com/prestodb/presto/wiki/Presto-Users
PrestoDB at High Level
3. How to use PrestoDB?
Two ways

Docker, AMI or thanks to Ahana (Presto Foundation Member)

https://ahana.io/getting-started/
- docker pull ahanaio/prestodb
- docker pull ahanaio/prestodb-cli
- AWS AMI, Google Cloud are also available

Manual Installation
UseCases-1

**General Use Cases for Any Industry – Data Virtualization**

An organization can make value by various ways described

**Combine all data sources including new ones**

- Allowing single view of all data

**Create your queries and understand data**

- Drag and Drop Query Builder allows to see all data in single window

**Distribute Processing**

- Use unlimited parallelism for distribution of queries

**Consume**

- Access the unified data from multitude of consumers

---

**Overall Benefits:**

Create a Modern Data Lake, Get All Data at one place, Expand AI/ML capabilities, Start Analyzing Data for better business decisions, Ease the job of managing many resources, tools and infrastructure
UseCases-2

*General Use Cases for Any Industry - Archiving*

For those who are using expensive archival solutions

- **Reduce Extra Licensing costs by moving data at rest to cloud/on prem object storage**
  - Less frequently used data can be archived

- **Push Down Processing**
  - Use push down predicate logic so that processing happens where the data lives

- **Reduce Storage Costs by storing in modern formats like ORC**
  - Columnar formats, flattened data allows easy querying

- **Combine and Consume**
  - Access the unified data from archived and breathing data

---

**Overall Benefits:**

Reduced License costs, No need for tapes for retention of data, available as and when needed. Use horizontal scaling
UseCases-3

Customer Retention

Customers are the biggest assets, and we help retaining them

Unified customer profile
A single profile combining all historical and present touch points

Real and Non-intrusive rewards and spot promotions, Targeted Advertising
By looking at the customer preferences and current requirements via real time feeds, the organization can issue spot promotional offers

Clickstream Analytics
Looking at time spent of applications, improvements can be made to make sure customer reaches fulfillment scenario faster

AI and Machine Learning
Apply algorithms like Markov models and customer segmentation, affinity etc with Python and invoke as SQL

Overall Benefits:
Retain existing customers as well as acquire new with unseen opportunities thus improving the business outcomes
Tuning PrestoDB
Configuration Properties

- Complete list [here](#)
  - broadcast/partitioned/automatic joins
  - Resource Groups

Query Efficiency

- Presto Aria
- Run Presto queries on Spark
Project Aria

https://www.youtube.com/watch?v=aXBOiL5dm2U

https://engineering.fb.com/2019/06/10/data-infrastructure/aria-presto/
Accessing PrestoDB

- From Spark/Java Applications – Prestodb JDBC
- From Python - PyHive
- From R – Rpresto
- Node-Red 😊 - https://flows.nodered.org/node/node-red-contrib-presto-client

Other Features

- Custom Functions
- Custom Connectors
- Event Listeners ….and more
Resources

PrestoDB Official Website

Please join the community and help us grow, together

- **Project Aria** – PrestoDB can now push down entire expressions to the data source for some file formats like ORC. [Blog Design](#)
- **Project Presto Unlimited** – Introduced exchange materialization to create temporary in-memory bucketed tables to use significantly less memory. [PR Blog](#)
- **User Defined Functions** – Support for dynamic SQL functions is now available in experimental mode. [Docs](#)
- Building Modern Data Lakes [Part 1, Part 2](#)

- **Apache Pinot and Druid Connectors** – [Docs](#)
- **RaptorX** – Disaggregates the storage from compute for low latency to provide a unified, cheap, fast, and scalable solution to OLAP and interactive use cases. [Issue](#)
- **Presto-on-Spark** Runs Presto code as a library within Spar executor. [Design Docs](#)
- **Disaggregated Coordinator (a.k.a. Fireball)** – Scale out the coordinator horizontally and revamp the RPC stack. Beta in Q4 2020. [Issues](#)
- **Ultimate Duo in Distributed Computing: Running PrestoDB on Spark**
Join Us

- Slack
- Virtual Meetup Groups
- presto-users google mailing list
- Join the Presto Foundation

“We” are smarter than “me”!
THANK YOU!

PERCONA
LIVEONLINE
MAY 12 - 13th
2021
Appendix

Code Examples
Example 1: Python to PrestoDB through PyHive

```python
import pandas as pd
from pyhive import presto
conn = presto.connect(
    host='localhost',
    port=8080,
    protocol='http',
    catalog='mysql',
    schema='ml',
    username='ravi',
)
cursor = conn.cursor()
query = """select e.first_name, e.last_name, e.phone_number, j.min_salary, j.max_salary
from mysql.ml.employees e inner join oracle.hr.jobs j on e.job_id = j.job_id"""

names = ['First Name', 'Last Name', 'Phone Number', 'MIN SAL', 'MAX SAL']
cursor.execute(query)
demo_df = pd.DataFrame(cursor.fetchall(), columns = names)
demo_df.head()
```
Example 2: R to PrestoDB through RPresoto

```r
con <- dbConnect(
  R Presto::Presto(),
  host = "http://localhost",
  port = 8080,
  user = 'ravi',
  catalog = 'mysql',
  schema = 'ml'
)
res <- dbSendQuery(con, "select e.*, j.min_salary, j.max_salary from mysql.ml.employees
e inner join oracle.hr.jobs j on e.job_id = j.job_id")
print(dbFetch(res, -1))
```
Example 3: Using PrestoDB Aria

Enable at session:

```
SET SESSION pushdown_subfields_enabled=true;
SET SESSION local.pushdown_filter_enabled=true;  //Assuming local is a catalog, example is schema
  //and ratings is the name of table

EXPLAIN (TYPE DISTRIBUTED) select count(*) from local.example.ratings where rating between 3.0 and 5.0;
```

//You will see that the filter condition is pushed to the underlying store
Example 4: Using SQL-ML on PrestoDB

PrestoDB supports some of the ML functions

```
SELECT
classify(features(5.9, 3, 5.1, 1.8), model) AS prediction_perconalive
FROM (  
    SELECT  
    learn_classifier(species, features(sepal_length, sepal_width, petal_length, petal_width)) AS model
    FROM  
    mysql.ml.iris
) t
```
Example 5: Using HyperLogLog Function

create table mysql.logs.sample(flowid varchar, hll varbinary);

INSERT INTO mysql.logs.sample SELECT flowid, cast(approx_set(SourceIP) AS varbinary) FROM mysql.logs.iplog group by flowid;

SELECT cardinality(merge(cast(hll AS HyperLogLog))) AS weekly_unique_users FROM mysql.logs.sample;
Example 6: Running PrestoDB on Spark

Assume that Spark is running on passionbytes:7077, query.sql is containing the query you want to run on Spark. Write your query in a file called query.sql.

```
spark-submit \
--master spark://passionbytes:7077 \
--executor-cores 2 \ 
--conf spark.task.cpus=2 \ 
--class com.facebook.presto.spark.launcher.PrestoSparkLauncher \ 
  presto-spark-launcher-*\jar \ 
--package presto-spark-package-*\tar.gz \ 
--config config.properties \ 
--catalogs presto/etc/catalog/ \ 
--file query.sql
```

--- INSERT ---