Build and validation of prestodb-presto 0.236

As per the initial analysis of the build and validation performed on RHEL 7.6 ppc64le, there were test failures in 13 packages, namely, presto-main, presto-function-namespace-managers, presto-redis, presto-rcfile, presto-hive, presto-raptor, presto-mysql, presto-postgresql, presto-geospatial, presto-jdbc, presto-verifier, presto-benchmark-runner, presto-spark-base. These packages with respect to type of failures were divided into 5 categories:

1. TimeZoneNotSupported, out of memory (presto-main)
2. mysql-server related failures (presto-function-namespace-managers, presto-mysql, presto-verifier, presto-benchmark-runner, presto-raptor, presto-rcfile)
3. postgresql related failures (presto-postgresql)
4. hadoop, snappy related failures (presto-hive, presto-jdbc, presto-geospatial, presto-spark-base)
5. redis related failures (presto-redis)

Failure #1 above is seen in two tests of presto-main package and those failures are seen on intel as well. So, ignored/skipped those two tests, as being in parity with intel.

Failure #2 related to mysql-server does not necessarily relate to a mysql-server installation on the system. But it is related to unavailability of mysql-Linux-amd64.tar.gz in testing-mysql-server-5-0.6.jar which gets downloaded from maven central repo as a dependency necessary for test execution. This tarball contains mysql binary distribution package which gets extracted to /tmp/testing-mysql-serverxxxx during test execution and the server is executed from there against which the tests are executed. In order to add the ppc64le support to the jar file, the changes for ppc64le needed to be added to the repo, https://github.com/prestodb/testing-mysql-server. And since availability and support of mysql binary tarballs for ppc64le is scarce, we confirmed with the community whether they will accept maridb as an alternative for ppc64le, to which the community agreed. Please refer to the following github issues for more details: https://github.com/prestodb/testing-mysql-server/issues/12, https://github.com/prestodb/presto/issues/14549. A PR was raised to that effect: https://github.com/prestodb/testing-mysql-server/pull/13 and was merged by the community. The tagged release to maven central repo however is not pushed yet. So, update to the presto repo to include a newer jar having ppc64le support is pending. But there has not been any response from the community on the ETA. Followed up for that at https://github.com/prestodb/testing-mysql-server/issues/17.

Failure #3 related to postgresql is similar to the one related to mysql-server above. In this case postgresql-Linux-ppc64le.tar.gz is missing in testing-postgresql-server-9.6.3-4.jar. The changes needed to be added to https://github.com/prestodb/testing-postgresql-server. More details on this can be found on the github issue: https://github.com/prestodb/presto/issues/14550. A PR was raised and merged to that effect: https://github.com/prestodb/testing-postgresql-server/pull/3. The situation of the tagged release and changes thereafter is same as the mysql-server issue above.

Failure #4 related to hadoop, snappy is due to incompatible libhadoop.so, libsnappy.so for ppc64le in hadoop-apache2-2.7.4-7.jar and missing ppc64le library libsnappyjava.so in snappy-java-1.1.7.1.jar. To fix the incompatibility issue, CentOS 7 based ppc64le libraries needed to be added to https://github.com/prestodb/presto-hadoop-apache2. And since binary publishing through IBM firewall is quite an exhaustive process and is not recommended, a travis job was setup to extract the libraries from an rpm (https://github.com/amitsadaphule/HDP-ppc64le-bin-factory) and push to a fork. A PR was raised through this fork: https://github.com/prestodb/presto-hadoop-apache2/pull/45. But the community has not responded to that. More details can be found at https://github.com/prestodb/presto-hadoop-apache2/issues/44. Also, regarding the libsnappyjava.so issue, newer releases (>= 1.1.7.4) of the jar have the support for ppc64le, but they need java 11, while presto still defaults to using java 8. More details are at https://github.com/prestodb/presto/issues/14549#issuecomment-656571077 and the comments post that.

Failure #5 is due to missing ppc64le based redis-server binary in embedded-redis-0.6.jar. The changes for ppc64le needed to be added to https://github.com/kstyrc/embedded-redis. A PR was raised to that effect https://github.com/kstyrc/embedded-redis/pull/117, but no response was received from the community. More details can be found on the github issue: https://github.com/kstyrc/embedded-redis/issues/71.

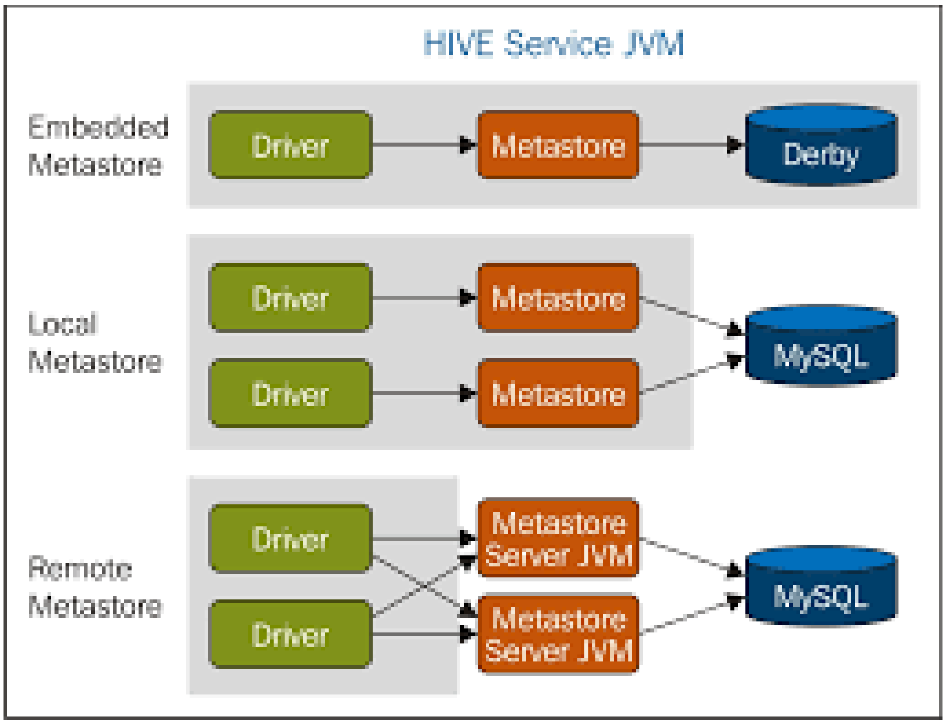
The build script for presto 0.236 is @ https://github.com/ppc64le/build-scripts/tree/master/p/prestodb-presto

This RTC story covers all the tasks related to prestodb-presto build and validation.

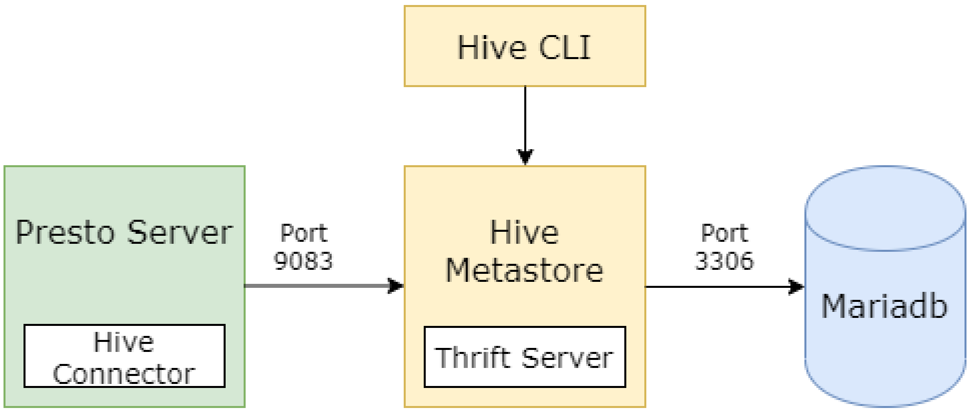
Please find the details about the demo using presto-server, presto-cli with hive metastore in the following pages.

Hive Metastore

As shown in the image below, hive metastore can be configured in 3 different modes:



Out of these, the one that we've set up is a local metastore. In Local mode, the Hive metastore service runs in the same process as the main HiveServer process, but the metastore database runs in a separate process and can be on a separate host. The metastore service communicates with the metastore database over JDBC. In our case, the metastore database is mysql/mariadb. Please refer the diagram below which reflects the current set up:



Presto Server

The build artifacts for presto server are *bin, lib, plugin* directories which need to be copied over to the installation directory on a machine and then bin directory needs to be added to the PATH. A directory *etc* needs to be created in this installation directory with the following configuration files:

config.properties

```
coordinator=true
node-scheduler.include-coordinator=true
http-server.http.port=8080
query.max-memory=5GB
query.max-memory-per-node=1GB
query.max-total-memory-per-node=2GB
discovery-server.enabled=true
discovery.uri=http://localhost:8080
```

jvm.config

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

log.properties

```
com.facebook.presto=INFO
com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory=WARN
com.ning.http.client=DEBUG
com.facebook.presto.server.PluginManager=INFO
com.facebook.airlift.discovery.client=INFO
```

node.properties

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

catalog/hive.properties

connector.name=hive-hadoop2

hive.metastore.uri=thrift://localhost:9083

The command ***launcher run*** is used to execute the presto server. **The *hive.metastore.uri* in hive.properties provides the presto-server with the contact point for the hive metastore.**

Hive Metastore

Since presto is compatible with Hadoop 2.x and not 3.x, Hadoop/hive from HDP 2.x repo is installed as:

cd /etc/yum.repos.d/

wget http://public-repo-1.hortonworks.com/HDP/centos7-ppc/2.x/updates/2.6.1.0/hdp.repo

yum install -y hive_2_6_1_0_129 hive_2_6_1_0_129-metastore hive_2_6_1_0_129-server hive_2_6_1_0_129-server2

mkdir -p /presto-root/hive/warehouse

The only additional configuration after this installation for hive metastore is editing **/etc/hive/conf/hive-site.xml** file to have the following properties in the configuration tag:

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://localhost/metastore</value>
  <description>the URL of the MySQL database</description>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.jdbc.Driver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hive</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>mypassword</value>
</property>

<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>false</value>
</property>

<property>
  <name>datanucleus.fixedDatastore</name>
  <value>true</value>
</property>

<property>
  <name>datanucleus.autoStartMechanism</name>
  <value>SchemaTable</value>
</property>

<property>
  <name>hive.metastore.uris</name>
  <value>thrift://localhost:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore
host</description>
</property>

<property>
  <name>hive.metastore.schema.verification</name>
  <value>true</value>
</property>

<property>
  <name>hive.metastore.warehouse.dir</name>
  <value>/presto-root/hive/warehouse/</value>
</property>
```

As mentioned above, ***hive.metastore.uris*** is the URI exposed by the thrift server which is used by presto-server to connect to. Rest of the configurations are for connecting to the mariadb database and for warehouse directory location.

Mariadb

Installation of mariadb and the jdbc connector for that for hive:

yum install -y mariadb-server

systemctl start mariadb

yum install -y mysql-connector-java

ln -s /usr/share/java/mysql-connector-java.jar /usr/hdp/current/hive-client/lib/mysql-connector-java.jar

And then the database needs to be configured for use with hive metastore:

mysql -u root -p

password: ENTER

MariaDB [(none)]> CREATE USER 'hive'@'localhost' IDENTIFIED BY 'mypassword';

MariaDB [(none)]> CREATE DATABASE metastore;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON metastore. TO 'hive'@'localhost';*

MariaDB [(none)]> FLUSH PRIVILEGES;

MariaDB [(none)]> quit;

/usr/hdp/2.6.1.0-129/hive/bin/schematool -initSchema -dbType mysql

Demo

Terminal 1

hive --service metastore

Terminal 2

hive

```
hive> CREATE SCHEMA tutorials2;

hive> create table tutorials2.author(auth_id int, auth_name varchar(50), topic varchar(100)) STORED AS SEQUENCEFILE;

hive> insert into table tutorials2.author values (1, 'Doug Cutting', 'Hadoop'), (2, 'James Gosling', 'java'),(3, 'Dennis Ritchie', 'C');
```

Terminal 3

```
launcher run -v
```

Terminal 4

```
presto --server localhost:8080 --catalog hive

presto:default> show schemas from hive;

    Schema
--------------------
default
information_schema
tutorials2
(3 rows)

Query 20200525_170843_00003_g3c6k, FINISHED, 1 node
Splits: 19 total, 19 done (100.00%)
0:01 [3 rows, 50B] [4 rows/s, 82B/s]

presto:default> show tables from hive.tutorials2;
Table
--------
author
(1 row)

Query 20200525_170852_00004_g3c6k, FINISHED, 1 node
Splits: 19 total, 19 done (100.00%)
0:01 [1 rows, 26B] [1 rows/s, 40B/s]

presto:default> select * from hive.tutorials2.author;
auth_id |   auth_name    | topic
---------+----------------+--------
    1 | Doug Cutting   | Hadoop
    2 | James Gosling  | java
    3 | Dennis Ritchie | C
(3 rows)

Query 20200525_170919_00005_g3c6k, FINISHED, 1 node
Splits: 17 total, 17 done (100.00%)
0:01 [3 rows, 185B] [2 rows/s, 137B/s]
```

References:

https://www.tutorialspoint.com/apache_presto/apache_presto_hive_connector.htm

https://prestodb.io/docs/current/installation/deployment.html

https://prestodb.io/docs/current/connector/hive.html

http://bdlabs.edureka.co/static/help/topics/cdh_ig_hive_metastore_configure.html

http://hadooptutorial.info/hive-metastore-configuration/

https://mapr.com/docs/61/Hive/Config-MariaDBForHiveMetastore.html