Setting up Phoenix storage for a timeline v2 end-to-end test

Working branch: YARN-2928

Step 1: Pick and deploy the right version of HBase

Phoenix 4.3.x works with HBase 0.98.1 and later versions. However, there is no successful report on running Phoenix 4.3.x on HBase 1. Our current Phoenix writer implementation is tested with Phoenix 4.3.0 with HBase 0.98.11.

Step 2: Setup HBase cluster for testing

Phoenix works on top of HBase. We need to set up an HBase cluster before using Phoenix. As a demo, we may want to set up a standalone HBase cluster for an end-to-end test. To set up a standalone HBase cluster, see: http://hbase.apache.org/book.html#quickstart

**After the test standalone HBase cluster is launched, one may check its status page at port 60010.*

Step 3: Download and Deploy Phoenix

Follow the instructions on http://phoenix.apache.org/installation.html to install Phoenix to HBase. Note that we don't need to follow the last step: "Add the phoenix-[version]-client.jar to the classpath of any Phoenix client."

Step 3.1: Verify if Phoenix is working

Under <Phoenix_dir>/bin, run sqlline with a pointer to the HBase's zookeeper quorum and zookeeper base path. For example:

```
./sqlline.py localhost:2181:/hbase
```

Once connected, try some sqlline commands, like

```
!tables
```

You can expect the console prints out all available Phoenix metadata tables.

Step 4: Apply the Phoenix writer patch

Apply the latest patch on YARN-3134 to YARN-2928 branch. We're continuously updating the patch under this JIRA. After applying the patch, rebuild Hadoop.

Step 5: Change timeline related configurations

Make sure the following settings are changed accordingly.

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>...,timeline_collector</value>
</property>

<property>
    <name>yarn.nodemanager.aux-services.timeline_collector.class</name>
    <value>
org.apache.hadoop.yarn.server.timelineservice.collector.PerNodeTimelineCollecto
rsAuxService
    </value>
</property>

<property>
    <name>yarn.system-metrics-publisher.enabled</name>
    <value>true</value>
</property>

<property>
    <name>yarn.timeline-service.writer.class</name>
    <value>
org.apache.hadoop.yarn.server.timelineservice.storage.PhoenixTimelineWriterImpl
    </value>
</property>
```

Step 6: Launch HDFS and YARN

Step 7: Run a test mapreduce application

Run with mapreduce.job.emit-timeline-data and mapreduce.job.new-timeline-service.enabled both set to true. For example:

```
bin/hadoop jar
share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0-SNAPSHOT.jar
pi -Dmapreduce.job.emit-timeline-data=true
-Dmapreduce.job.new-timeline-service.enabled=true 16 10000
```

Step 8: Check from sqlline

You can run queries in sqlline for the HBase storage. For example:

```
SELECT * FROM TIMELINE_ENTITY;
```

Return all available entities in the table. sqlline only shows the first few columns. To check what exactly is inside, try:

```
SELECT ENTITYID FROM TIMELINE_ENTITY;
```

Or, you may try something more complex. For example, the following statement returns all mappers ordered by their finish time, in descending order:

```
SELECT ENTITYID FROM TIMELINE_EVENT WHERE
EVENTID='MAP_ATTEMPT_FINISHED' ORDER BY TIMESTAMP DESC;
```