HDFS Filesystem Support

Developing for HDFS in C++, Python on Linux

Environment Setup

Developing locally requires some environment setup.

First, you must install a Hadoop release for Linux, which contains the libhdfs RPC client library

```
HADOOP_VERSION=2.6.5
wget -q -O hadoop-$HADOOP_VERSION.tar.gz "https://www.apache.org/dyn/mirrors/mirrors.cgi?
action=download&filename=hadoop/common/hadoop-$HADOOP_VERSION/hadoop-$HADOOP_VERSION.tar.gz" &&
    tar -zxf hadoop-$HADOOP_VERSION.tar.gz


# You can put this wherever you wish
sudo mv hadoop-$HADOOP_VERSION /opt/hadoop-$HADOOP_VERSION
```

After installing Hadoop, you must do some environment setup:

```
# This is where you installed Hadoop
export HADOOP_HOME=/opt/hadoop-$HADOOP_VERSION


# Add Hadoop Java libraries to your CLASSPATH, and
# add native libraries to LD_LIBRARY_PATH
export CLASSPATH=`$HADOOP_HOME/bin/hadoop classpath --glob`
export HADOOP_OPTS="$HADOOP_OPTS -Djava.library.path=$HADOOP_HOME/lib/native"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HADOOP_HOME/lib/native/


# Set ARROW_HOME to the location of your Arrow repository checkout
export CLASSPATH=$ARROW_HOME/integration/hdfs:$CLASSPATH
```

Running local HDFS cluster, running unit tests

With the above environment setup out of the way, we can spin up a local HDFS cluster with docker-compose:

```
$ docker-compose up hdfs-data-node-1
```

This will start the HDFS namenode on localhost with RPC port 9000. The C++ and Python unit tests rely on these environment variables:

```
export ARROW_HDFS_TEST_HOST=localhost
export ARROW_HDFS_TEST_USER=root
export ARROW_HDFS_TEST_PORT=9000
```

Now, you should be able to run the unit tests:

```
# From a C++ source build
$ debug/arrow-io-hdfs-test
```