BuildingHadoopFromSVN

Building Hadoop 1.x

This documents some of the tricks needed to make building/running Hadoop 1.x easier

Set Ant up

- In `ANT_OPTS`, set up any proxy you need. For example

```
export ANT_OPTS=-Dhttp.proxyHost=web-proxy -Dhttp.proxyPort=8088 -Dhttps.proxyHost=web-proxy -Dhttps.
proxyPort=8088
```

- set `ANT_ARGS` to `-logger org.apache.tools.ant.listener.BigProjectLogger` if you want the "big project logger", which is handy when you build across different modules.

Check out the Hadoop modules

Create a root directory, `~/Hadoop`

Check out from SVN Avro (just for the source when debugging), and the Hadoop modules

```
svn co https://svn.apache.org/repos/asf/hadoop/avro/trunk avro
svn co https://svn.apache.org/repos/asf/hadoop/common/trunk hadoop-common
svn co https://svn.apache.org/repos/asf/hadoop/hdfs/trunk hadoop-hdfs
svn co https://svn.apache.org/repos/asf/hadoop/mapreduce/trunk hadoop-mapreduce
```

patch the `build.xml` files of hadoop* to make the project names all lower case otherwise they get published with the wrong name in Ivy

sharing build.properties

The build files all load in {`${user.home}/build.properties`} , so you can set settings there for every project.

They also load in {`${basedir}/build.properties`}

You can share a `build.properties` file

1. Make the file: `touch ~/Hadoop/build.properties`
1. Symlink it

```
   ln -s  ~/Hadoop/build.properties  ~/Hadoop/hadoop-common
   ln -s  ~/Hadoop/build.properties  ~/Hadoop/hadoop-hdfs
   ln -s  ~/Hadoop/build.properties  ~/Hadoop/hadoop-mapreduce
```

Now changes propagate around automatically.

Create a new Hadoop version

In `build.properties`:

```
version=0.21.0-alpha-15
hadoop.version=${version}
hadoop-core.version=${version}
hadoop-mr.version=0.21.0-dev
```

Why the different versions? I check versions in to SVN for hudson et al, bumping up my version count every release. For stuff I build locally, we use a version counter one notch higher. This ensures that local builds get the latest versions, while the other ones get releases.

Now, we are going to have fun by symlinking from to the specific artifacts we create. This stops us having to care about publishing things. Any local build is automatically picked up -until that version number is changed.

Create the core JARs

1. Change to `hadoop-common` : `cd ~/Hadoop/hadoop-common`
1. `ant clean jar jar-test ivy-publish-local`
1. If you have made any changes to the source, run `ant test` until the tests pass.
1. run `ls -l build/*.jar` -expect to see two Jars

```
-rw-r--r-- 1 hadoop users 1059526 2009-08-18 16:23 build/hadoop-common-0.21.0-alpha-15.jar
-rw-r--r-- 1 hadoop users  408716 2009-08-18 16:23 build/hadoop-common-test-0.21.0-alpha-15.jar
```

1. link everything up:

```
ln -s ~/Hadoop/hadoop-common/build/hadoop-common-0.21.0-alpha-15.jar ../hadoop-hdfs/lib
ln -s ~/Hadoop/hadoop-common/build/hadoop-common-test-0.21.0-alpha-15.jar ../hadoop-hdfs/lib
ln -s ~/Hadoop/hadoop-common/build/hadoop-common-0.21.0-alpha-15.jar ../hadoop-mapreduce/lib
ln -s ~/Hadoop/hadoop-common/build/hadoop-common-test-0.21.0-alpha-15.jar ../hadoop-mapreduce/lib
```

This has now propagated the JAR to the different directories, and put it in the Ivy repository for other programs to pick up.

The JAR is not copied, just linked to the `hadoop-common/build/` version. This means

- Whenever a change is made to the common project, and it is rebuilt, the other projects get the change immediately.
- If you change the version number in common, all the links break
- If you do an `ant clean` in common/ the other projects will not build until you have gone `ant jar jar-test` again. You could change the links to point to the ivy artifacts under `~/.ivy2/local/org.apache.hadoop/`. This would give access to the cached versions, and is a half-way house between full Ivy integration for Hadoop JAR retrieval, and integration of local builds via symbolic links. But it does require `ivy-publish` to work everywhere.

Build the HDFS JARS

1. Change to `hadoop-hdfs` : `cd ~/Hadoop/hadoop-hdfs`
1. `ant clean jar jar-test ivy-publish-local`
1. If you have made any changes to the source, run `ant test` until the tests pass. It's OK at this point for the `run-test-hdfs-with-mr` tests to fail.
1. do an `ls -l build` to check the JARs are there and versioned right
1. Symlink the hdfs JARS into mapreduce

```
  ln -s ~/Hadoop/hadoop-hdfs/build/hadoop-hdfs-0.21.0-alpha-15.jar ../hadoop-mapreduce/lib
  ln -s ~/Hadoop/hadoop-hdfs/build/hadoop-hdfs-test-0.21.0-alpha-15.jar ../hadoop-mapreduce/lib
```

Build the MapReduce JARS

1. Change to `hadoop-mapreduce` : `cd ~/Hadoop/hadoop-mapreduce`
1. `ant clean jar jar-test`
1. Symlink these jars back into hadoop-hdfs (ooh, a loop!)

```
  ln -s ~/Hadoop/hadoop-mapreduce/build/hadoop-mapred-0.21.0-alpha-15.jar ../hadoop-hdfs/lib
  ln -s ~/Hadoop/hadoop-mapreduce/build/hadoop-mapred-test-0.21.0-alpha-15.jar ../hadoop-hdfs/lib
```

1. Patch build.properties to pick up the new version

```
hadoop-mr.version=${version}
```

1. Run the mapreduce tests
1. Go back to hadoop-hdfs and run the tests and make sure that the tests under `run-test-hdfs-with-mr` pass