HadoopStreaming

Hadoop Streaming is a utility which allows users to create and run jobs with any executables (e.g. shell utilities) as the mapper and/or the reducer.

```
Usage: $HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/mapred/contrib/streaming/hadoop-streaming.jar [options]
Options:
  -input    <path>                  DFS input file(s) for the Map step
  -output   <path>                  DFS output directory for the Reduce step
  -mapper   <cmd|JavaClassName>     The streaming command to run
  -combiner <JavaClassName>         Combiner has to be a Java class
  -reducer  <cmd|JavaClassName>     The streaming command to run
  -file     <file>                  File/dir to be shipped in the Job jar file
  -dfs    <h:p>|local               Optional. Override DFS configuration
  -jt     <h:p>|local               Optional. Override JobTracker configuration
  -additionalconfspec specfile      Optional.
  -inputformat TextInputFormat(default)|SequenceFileAsTextInputFormat|JavaClassName Optional.
  -outputformat TextOutputFormat(default)|JavaClassName  Optional.
  -partitioner JavaClassName        Optional.
  -numReduceTasks <num>             Optional.
  -inputreader <spec>               Optional.
  -jobconf  <n>=<v>                 Optional. Add or override a JobConf property
  -cmdenv   <n>=<v>                 Optional. Pass env.var to streaming commands
  -cacheFile fileNameURI
  -cacheArchive fileNameURI
  -verbose

In -input: globbing on <path> is supported and can have multiple -input
Default Map input format: a line is a record in UTF-8.  Every line must end
  with an 'end of line' delimiter.  The key part ends at first TAB, the rest
  of the line is the value
Custom Map input format: -inputreader package.MyRecordReader,n=v,n=v
  comma-separated name-values can be specified to configure the InputFormat
  Ex: -inputreader 'StreamXmlRecordReader,begin=<doc>,end=</doc>'
Map output format, reduce input/output format:
  Format defined by what mapper command outputs. Line-oriented

Use -cluster <name> to switch between "local" Hadoop and one or more remote
  Hadoop clusters.
  The default is to use the normal hadoop-default.xml and hadoop-site.xml
  Else configuration will use $HADOOP_HOME/conf/hadoop-<name>.xml

To set the number of reduce tasks (num. of output files):
  -jobconf mapred.reduce.tasks=10
To change the local temp directory:
  -jobconf dfs.data.dir=/tmp
Additional local temp directories with -cluster local:
  -jobconf mapred.local.dir=/tmp/local
  -jobconf mapred.system.dir=/tmp/system
  -jobconf mapred.temp.dir=/tmp/temp
For more details about jobconf parameters see:
  http://wiki.apache.org/hadoop/JobConfFile
To set an environement variable in a streaming command:
   -cmdenv EXAMPLE_DIR=/home/example/dictionaries/

Shortcut to run from any directory:
   setenv HSTREAMING "$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/mapred/contrib/streaming/hadoop-streaming.jar"

Example: $HSTREAMING -mapper "/usr/local/bin/perl5 filter.pl"
           -file /local/filter.pl -input "/logs/0604*/*" [...]
  Ships a script, invokes the non-shipped perl interpreter
  Shipped files go to the working directory so filter.pl is found by perl
  Input files are all the daily logs for days in month 2006-04
```

Practical Help

Using the streaming system you can develop working hadoop jobs with *extremely* limited knowldge of Java. At it's simplest your development task is to write two shell scripts that work well together, let's call them **shellMapper.sh** and **shellReducer.sh**. On a machine that doesn't even have hadoop installed you can get first drafts of these working by writing them to work in this way:

```
cat someInputFile | shellMapper.sh | shellReducer.sh > someOutputFile
```

With streaming, Hadoop basically becomes a system for making pipes from shell-scripting work (with some fudging) on a cluster. There's a strong logical correspondence between the unix shell scripting environment and hadoop streaming jobs. The above example with Hadoop has somewhat less elegant syntax, but this is what it looks like:

```
stream -input /dfsInputDir/someInputData -file shellMapper.sh -mapper "shellMapper.sh" -file shellReducer.sh  -
reducer "shellReducer.sh" -output /dfsOutputDir/myResults
```

The real place the logical correspondence breaks down is that in a one machine scripting environment shellMapper.sh and shellReducer.sh will each run as a single process and data will flow directly from one process to the other. With Hadoop the shellMapper.sh file will be sent to every machine on the cluster that has data chunks and each such machine will run it's own chunk through the shellMapper.sh process on each machine. The output from those scripts *doesn't* run a reduce on each of those machines. Instead the output is sorted so that different lines from various mapping jobs are streamed across the network to different machines (Hadoop defaults to four machines) where the reduce(s) can be performed.

Here are practical tips for getting things working well:

- **Use shell scripts rather than commands** - The "-file shellMapper.sh" part isn't entirely necessary. You can simply use a clause like "-mapper 'sed | grep | awk'" or some such but complicated quoting is can introduce bugs. Wrapping the job in a shell script eliminates some of these issues.
- **Don't expect shebangs to work** - If you're going to run other scripts from inside your shell script, don't expect a line like #!/bin/python to work. To be certain that things will work, run the script directly like "grep somethingInteresting | ***perl*** *perlScript* | sort | uniq -c"

See Also

- HowToDebugMapReducePrograms
- HadoopStreaming AlternativeInterfaces
- Hadoop Streaming