HADOOP - STREAMING

http://www.tutorialspoint.com/hadoop/hadoop_streaming.htm Copyright © tutorialspoint.com

Hadoop streaming is a utility that comes with the Hadoop distribution. This utility allows you to create and run Map/Reduce jobs with any executable or script as the mapper and/or the reducer.

Example Using Python

For Hadoop streaming, we are considering the word-count problem. Any job in Hadoop must have two phases: mapper and reducer. We have written codes for the mapper and the reducer in python script to run it under Hadoop. One can also write the same in Perl and Ruby.

Mapper Phase Code

```
!/usr/bin/python
import sys
# Input takes from standard input for myline in sys.stdin:
# Remove whitespace either side myline = myline.strip()
# Break the line into words words = myline.split()
# Iterate the words list for myword in words:
# Write the results to standard output print '%s\t%s' % (myword, 1)
```

Make sure this file has execution permission $chmod + x/home/expert/hadoop - 1.2.1/mapper.py$.

Reducer Phase Code

```
#!/usr/bin/python
from operator import itemgetter
import sys
current_word = ""
current_count = 0
word = ""
# Input takes from standard input for myline in sys.stdin:
# Remove whitespace either side myline = myline.strip()
# Split the input we got from mapper.py word, count = myline.split('\t', 1)
# Convert count variable to integer
   try:
      count = int(count)
except ValueError:
   # Count was not a number, so silently ignore this line continue
if current_word == word:
   current_count += count
else:
   if current_word:
      # Write result to standard output print '%s\t%s' % (current_word, current_count)
   current_count = count
   current_word = word
# Do not forget to output the last word if needed!
if current_word == word:
   print '%s\t%s' % (current_word, current_count)
```

Save the mapper and reducer codes in mapper.py and reducer.py in Hadoop home directory. Make sure these files have execution permission $chmod + xmapper.pyandchmod + xreducer.py$. As python is indentation sensitive so the same code can be download from the below link.

Execution of WordCount Program

```
$ $HADOOP_HOME/bin/hadoop jar contrib/streaming/hadoop-streaming-1.
2.1.jar \
   -input input_dirs \
   -output output_dir \
   -mapper <path/mapper.py \
   -reducer <path/reducer.py
```

Where "\" is used for line continuation for clear readability.

For Example,

```
./bin/hadoop jar contrib/streaming/hadoop-streaming-1.2.1.jar -input myinput -output
myoutput -mapper /home/expert/hadoop-1.2.1/mapper.py -reducer
/home/expert/hadoop-1.2.1/reducer.py
```

How Streaming Works

In the above example, both the mapper and the reducer are python scripts that read the input from standard input and emit the output to standard output. The utility will create a Map/Reduce job, submit the job to an appropriate cluster, and monitor the progress of the job until it completes.

When a script is specified for mappers, each mapper task will launch the script as a separate process when the mapper is initialized. As the mapper task runs, it converts its inputs into lines and feed the lines to the standard input *STDIN* of the process. In the meantime, the mapper collects the line-oriented outputs from the standard output *STDOUT* of the process and converts each line into a key/value pair, which is collected as the output of the mapper. By default, the prefix of a line up to the first tab character is the key and the rest of the line *excludingthetabcharacter* will be the value. If there is no tab character in the line, then the entire line is considered as the key and the value is null. However, this can be customized, as per one need.

When a script is specified for reducers, each reducer task will launch the script as a separate process, then the reducer is initialized. As the reducer task runs, it converts its input key/values pairs into lines and feeds the lines to the standard input *STDIN* of the process. In the meantime, the reducer collects the line-oriented outputs from the standard output *STDOUT* of the process, converts each line into a key/value pair, which is collected as the output of the reducer. By default, the prefix of a line up to the first tab character is the key and the rest of the line *excludingthetabcharacter* is the value. However, this can be customized as per specific requirements.

Important Commands

Parameters	Description
-input directory/file-name	Input location for mapper. *Required*
-output directory-name	Output location for reducer. *Required*
-mapper executable or script or JavaClassName	Mapper executable. *Required*
-reducer executable or script or JavaClassName	Reducer executable. *Required*
-file file-name	Makes the mapper, reducer, or combiner executable available locally on the compute nodes.
-inputformat JavaClassName	Class you supply should return key/value pairs of Text class. If not specified, TextInputFormat is used as the default.
-outputformat JavaClassName	Class you supply should take key/value pairs of Text class. If not specified, TextOutputformat is used as the default.
-partitioner JavaClassName	Class that determines which reduce a key is sent to.
-combiner streamingCommand or JavaClassName	Combiner executable for map output.
-cmdenv name=value	Passes the environment variable to streaming commands.

-inputreader	For backwards-compatibility: specifies a record reader class *insteadofaninputformatclass*.
-verbose	Verbose output.
-lazyOutput	Creates output lazily. For example, if the output format is based on FileOutputFormat, the output file is created only on the first call to output.collect *orContext*. *write*.
-numReduceTasks	Specifies the number of reducers.
-mapdebug	Script to call when map task fails.
-reducedebug	Script to call when reduce task fails.

Loading [MathJax]/jax/output/HTML-CSS/jax.js