Writing an Hadoop MapReduce Program in Python

Table of Contents

In this tutorial I will describe how to write a simple MapReduce program for Hadoop in the Python programming language.

Motivation

Even though the Hadoop framework is written in Java, programs for Hadoop need not to be coded in Java but can also be developed in other languages like Python or C++ (the latter since version 0.14.1). However, Hadoop's documentation and the most prominent Python example on the Hadoop website could make you think that you *must* translate your Python code using Jython into a Java jar file. Obviously, this is not very convenient and can even be problematic if you depend on Python features not provided by Jython. Another issue of the Jython approach is the overhead of writing your Python program in such a way that it can interact with Hadoop – just have a look at the example in `$HADOOP_HOME/src/examples/python/WordCount.py` and you see what I mean.

That said, the ground is now prepared for the purpose of this tutorial: writing a Hadoop MapReduce program in a more Pythonic way, i.e. in a way you should be familiar with.

What we want to do

We will write a simple MapReduce program (see also the MapReduce article on Wikipedia) for Hadoop in Python but *without* using Jython to translate our code to Java jar files.

Our program will mimick the WordCount, i.e. it reads text files and counts how often words occur. The input is text files and the output is text files, each line of which contains a word and the count of how often it occured, separated by a tab.

Note: You can also use programming languages other than Python such as Perl or Ruby with the "technique" described in this tutorial.

Prerequisites

You should have an Hadoop cluster up and running because we will get our hands dirty. If you don't have a cluster yet, my following tutorials might help you to build one. The tutorials are tailored to Ubuntu Linux but the information does also apply to other Linux/Unix variants.

- Running Hadoop On Ubuntu Linux (Single-Node Cluster) – How to set up a *pseudo-distributed*, *single-node* Hadoop cluster backed by the Hadoop Distributed File System (HDFS)
- Running Hadoop On Ubuntu Linux (Multi-Node Cluster) – How to set up a *distributed*, *multi-node* Hadoop cluster backed by the Hadoop Distributed File System (HDFS)

Python MapReduce Code

The "trick" behind the following Python code is that we will use the Hadoop Streaming API (see also the corresponding wiki entry) for helping us passing data between our Map and Reduce code via `STDIN` (standard input) and `STDOUT` (standard output). We will simply use Python's `sys.stdin` to read input data and print our own output to `sys.stdout`. That's all we need to do because Hadoop Streaming will take care of everything else!

Map step: mapper.py

Save the following code in the file `/home/hduser/mapper.py`. It will read data from `STDIN`, split it into words and output a list of lines mapping words to their (intermediate) counts to `STDOUT`. The Map script will not compute an (intermediate) sum of a word's occurrences though. Instead, it will output `<word> 1` tuples immediately – even though a specific word might occur multiple times in the input. In our case we let the subsequent Reduce step do the final sum count. Of course, you can change this behavior in your own scripts as you please, but we will keep it like that in this tutorial because of didactic reasons. :-)

Make sure the file has execution permission (`chmod +x /home/hduser/mapper.py` should do the trick) or you will run into problems.

mapper.py

```
#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

Reduce step: reducer.py

Save the following code in the file `/home/hduser/reducer.py`. It will read the results of `mapper.py` from `STDIN` (so the output format of `mapper.py` and the expected input format of `reducer.py` must match) and sum the occurrences of each word to a final count, and then output its results to `STDOUT`.

Make sure the file has execution permission (`chmod +x /home/hduser/reducer.py` should do the trick) or you will run into problems.

reducer.py

```
#!/usr/bin/env python

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            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)
```

Test your code (cat data | map | sort | reduce)

I recommend to test your `mapper.py` and `reducer.py` scripts locally before using them in a MapReduce job. Otherwise your jobs might successfully complete but there will be no job result data at all or not the results you would have expected. If that happens, most likely it was you (or me) who screwed up.

Here are some ideas on how to test the functionality of the Map and Reduce scripts.

Test mapper.py and reducer.py locally first

```
# very basic test
hduser@ubuntu:~$ echo "foo foo quux labs foo bar quux" | /home/hduser/mapper.py
foo     1
foo     1
quux    1
labs    1
foo     1
bar     1
quux    1

hduser@ubuntu:~$ echo "foo foo quux labs foo bar quux" | /home/hduser/mapper.py | sort -k1,1 | /home/hduser/reducer.py
bar     1
foo     3
labs    1
quux    2

# using one of the ebooks as example input
# (see below on where to get the ebooks)
hduser@ubuntu:~$ cat /tmp/gutenberg/20417-8.txt | /home/hduser/mapper.py
 The     1
 Project 1
 Gutenberg       1
 EBook   1
 of      1
 [...]
 (you get the idea)
```

Running the Python Code on Hadoop

Download example input data

We will use three ebooks from Project Gutenberg for this example:

- The Outline of Science, Vol. 1 (of 4) by J. Arthur Thomson
- The Notebooks of Leonardo Da Vinci
- Ulysses by James Joyce

Download each ebook as text files in `Plain Text UTF-8` encoding and store the files in a local temporary directory of choice, for example `/tmp/gutenberg`.

```
hduser@ubuntu:~$ ls -l /tmp/gutenberg/
total 3604
-rw-r--r-- 1 hduser hadoop  674566 Feb  3 10:17 pg20417.txt
-rw-r--r-- 1 hduser hadoop 1573112 Feb  3 10:18 pg4300.txt
-rw-r--r-- 1 hduser hadoop 1423801 Feb  3 10:18 pg5000.txt
hduser@ubuntu:~$
```

Copy local example data to HDFS

Before we run the actual MapReduce job, we must first copy the files from our local file system to Hadoop's HDFS.

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -copyFromLocal /tmp/gutenberg /user/hduser/gutenberg
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls
Found 1 items
drwxr-xr-x   - hduser supergroup          0 2010-05-08 17:40 /user/hduser/gutenberg
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /user/hduser/gutenberg
Found 3 items
-rw-r--r--   3 hduser supergroup     674566 2011-03-10 11:38 /user/hduser/gutenberg/pg20417.txt
-rw-r--r--   3 hduser supergroup    1573112 2011-03-10 11:38 /user/hduser/gutenberg/pg4300.txt
-rw-r--r--   3 hduser supergroup    1423801 2011-03-10 11:38 /user/hduser/gutenberg/pg5000.txt
hduser@ubuntu:/usr/local/hadoop$
```

Run the MapReduce job

Now that everything is prepared, we can finally run our Python MapReduce job on the Hadoop cluster. As I said above, we leverage the Hadoop Streaming API for helping us passing data between our Map and Reduce code via `STDIN` and `STDOUT`.

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar contrib/streaming/hadoop-*streaming*.jar \
-file /home/hduser/mapper.py    -mapper /home/hduser/mapper.py \
-file /home/hduser/reducer.py   -reducer /home/hduser/reducer.py \
-input /user/hduser/gutenberg/* -output /user/hduser/gutenberg-output
```

If you want to modify some Hadoop settings on the fly like increasing the number of Reduce tasks, you can use the `-D` option:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar contrib/streaming/hadoop-*streaming*.jar -D mapred.reduce.tasks=16 ...
```

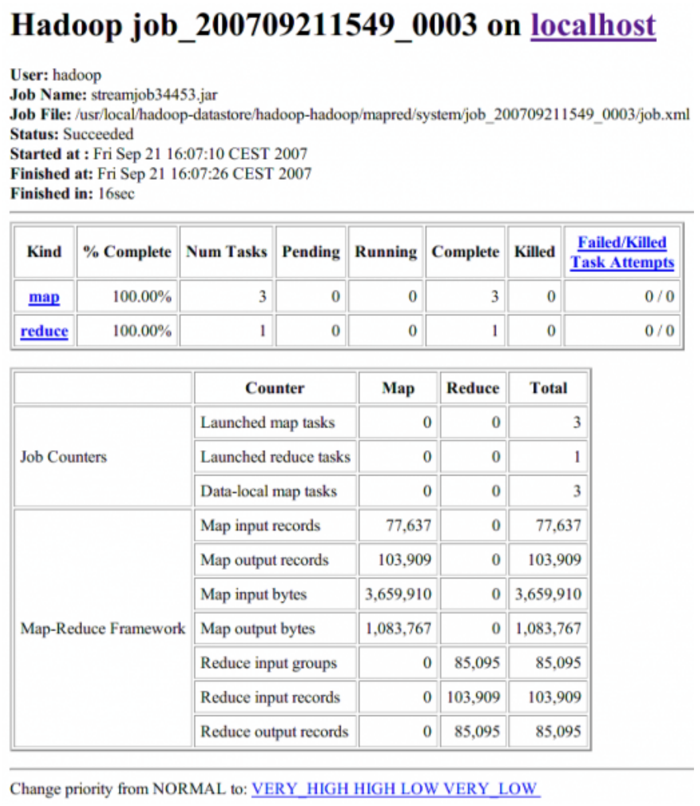
Note about `mapred.map.tasks`: Hadoop does not honor mapred.map.tasks beyond considering it a hint. But it accepts the user specified `mapred.reduce.tasks` and doesn't manipulate that. You cannot force `mapred.map.tasks` but can specify `mapred.reduce.tasks`.

The job will read all the files in the HDFS directory `/user/hduser/gutenberg`, process it, and store the results in the HDFS directory `/user/hduser/gutenberg-output`. In general Hadoop will create one output file per reducer; in our case however it will only create a single file because the input files are very small.

Example output of the previous command in the console:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar contrib/streaming/hadoop-*streaming*.jar -mapper /home/hduser/mapper.py -reducer /home
 additionalConfSpec_:null
 null=@@@userJobConfProps_.get(stream.shipped.hadoopstreaming
 packageJobJar: [/app/hadoop/tmp/hadoop-unjar54543/]
 [] /tmp/streamjob54544.jar tmpDir=null
 [...] INFO mapred.FileInputFormat: Total input paths to process : 7
 [...] INFO streaming.StreamJob: getLocalDirs(): [/app/hadoop/tmp/mapred/local]
 [...] INFO streaming.StreamJob: Running job: job_200803031615_0021
 [...]
 [...] INFO streaming.StreamJob:  map 0%  reduce 0%
 [...] INFO streaming.StreamJob:  map 43%  reduce 0%
 [...] INFO streaming.StreamJob:  map 86%  reduce 0%
 [...] INFO streaming.StreamJob:  map 100%  reduce 0%
 [...] INFO streaming.StreamJob:  map 100%  reduce 33%
 [...] INFO streaming.StreamJob:  map 100%  reduce 70%
 [...] INFO streaming.StreamJob:  map 100%  reduce 77%
 [...] INFO streaming.StreamJob:  map 100%  reduce 100%
 [...] INFO streaming.StreamJob: Job complete: job_200803031615_0021
 [...] INFO streaming.StreamJob: Output: /user/hduser/gutenberg-output
hduser@ubuntu:/usr/local/hadoop$
```

As you can see in the output above, Hadoop also provides a basic web interface for statistics and information. When the Hadoop cluster is running, open http://localhost:50030/ in a browser and have a look around. Here's a screenshot of the Hadoop web interface for the job we just ran.



Hadoop job_200709211549_0003 on localhost

User: hadoop
Job Name: streamjob34453.jar
Job File: /usr/local/hadoop-datastore/hadoop-hadoop/mapred/system/job_200709211549_0003/job.xml
Status: Succeeded
Started at : Fri Sep 21 16:07:10 CEST 2007
Finished at: Fri Sep 21 16:07:26 CEST 2007
Finished in: 16sec

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	3	0	0	3	0	0 / 0
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Launched map tasks	0	0	3
	Launched reduce tasks	0	0	1
	Data-local map tasks	0	0	3
Map-Reduce Framework	Map input records	77,637	0	77,637
	Map output records	103,909	0	103,909
	Map input bytes	3,659,910	0	3,659,910
	Map output bytes	1,083,767	0	1,083,767
	Reduce input groups	0	85,095	85,095
	Reduce input records	0	103,909	103,909
	Reduce output records	0	85,095	85,095

Change priority from NORMAL to: VERY_HIGH HIGH LOW VERY_LOW

Figure 1: A screenshot of Hadoop's JobTracker web interface, showing the details of the MapReduce job we just ran

Check if the result is successfully stored in HDFS directory `/user/hduser/gutenberg-output`:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /user/hduser/gutenberg-output
Found 1 items
/user/hduser/gutenberg-output/part-00000     &lt;r 1&gt;   903193  2007-09-21 13:00
hduser@ubuntu:/usr/local/hadoop$
```

You can then inspect the contents of the file with the `dfs -cat` command:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -cat /user/hduser/gutenberg-output/part-00000
"(Lo)cra"       1
"1490   1
"1498," 1
"35"    1
"40,"   1
"A      2
"AS-IS".        2
"A_     1
"Absoluti       1
[...]
hduser@ubuntu:/usr/local/hadoop$
```

Note that in this specific output above the quote signs (`"`) enclosing the words have not been inserted by Hadoop. They are the result of how our Python code splits words, and in this case it matched the beginning of a quote in the ebook texts. Just inspect the `part-00000` file further to see it for yourself.

Improved Mapper and Reducer code: using Python iterators and generators

The Mapper and Reducer examples above should have given you an idea of how to create your first MapReduce application. The focus was code simplicity and ease of understanding, particularly for beginners of the Python programming language. In a real-world application however, you might want to optimize your code by using Python iterators and generators (an even better introduction in PDF).

Generally speaking, iterators and generators (functions that create iterators, for example with Python's `yield` statement) have the advantage that an element of a sequence is not produced until you actually need it. This can help a lot in terms of computational expensiveness or memory consumption depending on the task at hand.

Note: The following Map and Reduce scripts will only work "correctly" when being run in the Hadoop context, i.e. as Mapper and Reducer in a MapReduce job. This means that running the naive test command "cat DATA | ./mapper.py | sort -k1,1 | ./reducer.py" will not work correctly anymore because some functionality is intentionally outsourced to Hadoop.

Precisely, we compute the sum of a word's occurrences, e.g. `("foo", 4)`, only if by chance the same word (`foo`) appears multiple times in succession. In the majority of cases, however, we let the Hadoop group the (key, value) pairs between the Map and the Reduce step because Hadoop is more efficient in this regard than our simple Python scripts.

mapper.py

mapper.py (improved)

```
#!/usr/bin/env python
"""A more advanced Mapper, using Python iterators and generators."""

import sys

def read_input(file):
    for line in file:
        # split the line into words
        yield line.split()

def main(separator='\t'):
    # input comes from STDIN (standard input)
    data = read_input(sys.stdin)
    for words in data:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        for word in words:
            print '%s%s%d' % (word, separator, 1)

if __name__ == "__main__":
    main()
```

reducer.py

reducer.py (improved)

```
#!/usr/bin/env python
"""A more advanced Reducer, using Python iterators and generators."""

from itertools import groupby
from operator import itemgetter
import sys

def read_mapper_output(file, separator='\t'):
    for line in file:
        yield line.rstrip().split(separator, 1)

def main(separator='\t'):
    # input comes from STDIN (standard input)
    data = read_mapper_output(sys.stdin, separator=separator)
    # groupby groups multiple word-count pairs by word,
    # and creates an iterator that returns consecutive keys and their group:
    #   current_word - string containing a word (the key)
    #   group - iterator yielding all ["&lt;current_word&gt;", "&lt;count&gt;"] items
    for current_word, group in groupby(data, itemgetter(0)):
        try:
            total_count = sum(int(count) for current_word, count in group)
            print "%s%s%d" % (current_word, separator, total_count)
        except ValueError:
            # count was not a number, so silently discard this item
            pass

if __name__ == "__main__":
    main()
```

Related Links

From yours truly:

- Running Hadoop On Ubuntu Linux (Single-Node Cluster)
- Running Hadoop On Ubuntu Linux (Multi-Node Cluster)

From others:

- Chinese translation of 'Writing an Hadoop Mapreduce Program in Python', by Jun Tian

Tweet

Comments

165 Comments **Michael G. Noll** 1 **Login**

Recommend 14 Share Sort by Best

Join the discussion…

Sanjay Gupta • 5 years ago
Hi Michael,
Great tutorial ...

One question, as you mentioned that hadoop does the file sorting and splitting. In the example it the split of the map output file is done across the same word then the reduce will have two entries of this word. Does hadoop takes care of such detail when he does the split of final map?

example

w 1
w 1
------- (if the file split is done here, them in the final reduced output file will have "w 2" followed by "w 3")
w 1
w 1
w 1
x 1
x 1
....
51 • Reply • Share ›

janardhan • 5 years ago
i got a problem in map reduce on python code... error is shown below...

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar contrib/streaming/hadoop-streaming-1.0.0.jar -file /home/hduser/hadoop/mapper.py -mapper
/home/hduser/hadoop/mapper.py -file /home/hduser/hadoop/reducer.py -reducer /home/hduser/hadoop/reducer.py -input /home/hduser/gutenberg/* -
output /home/hduser/output3Warning: $HADOOP_HOME is deprecated.

packageJobJar: [/home/hduser/hadoop/mapper.py, /home/hduser/hadoop/reducer.py, /app/hadoop/tmp/hadoop-unjar2090300167280691382/] []
/tmp/streamjob2369339998637272450.jar tmpDir=null
12/04/09 13:58:30 INFO mapred.FileInputFormat: Total input paths to process : 2
12/04/09 13:58:30 INFO streaming.StreamJob: getLocalDirs(): [/app/hadoop/tmp/mapred/local]
[...snipp...]
12/04/09 13:59:09 ERROR streaming.StreamJob: Job not successful. Error: # of failed Map Tasks exceeded allowed limit. FailedCount: 1.
LastFailedTask: task_201204091339_0004_m_000000
12/04/09 13:59:09 INFO streaming.StreamJob: killJob...
Streaming Job Failed!
```
38 • Reply • Share ›

Anuj • 5 years ago
Hi,

I am getting the following Error.. Any suggestion would be highly helpful..

Edited by Michael G. Noll: I have moved your long logging output to https://gist.github.com/158....

```
lrmraxm:hadoop-0.20.2-cdh3u2 anuj.maurice$ bin/hadoop jar contrib/streaming/hadoop-streaming-0.20.2-cdh3u2.jar -file
/Users/anuj.maurice/Downloads/hadoop-0.20.2-cdh3u2/python/mapper.py -mapper mapper.py -file /Users/anuj.maurice/Downloads/hadoop-0.20.2-
cdh3u2/python/reducer.py -reducer reducer.py -input /oos.txt -output /oos_new
packageJobJar: [/Users/anuj.maurice/Downloads/hadoop-0.20.2-cdh3u2/python/mapper.py, /Users/anuj.maurice/Downloads/hadoop-0.20.2-
cdh3u2/python/reducer.py, /tmp/hadoop-anuj.maurice/hadoop-unjar2426556812178658809/] [] /var/folders/Yu/YuXibLtIHOuWcHsjWu8zM-Ccvdo/-
Tmp-/streamjob4679204253733026415.jar tmpDir=null
```

```
[...snip...]

12/01/04 12:03:04 INFO streaming.StreamJob: map 100% reduce 100%
12/01/04 12:03:04 INFO streaming.StreamJob: To kill this job, run:
12/01/04 12:03:04 INFO streaming.StreamJob: /Users/anuj.maurice/Downloads/hadoop-0.20.2-cdh3u2/bin/../bin/hadoop job -
Dmapred.job.tracker=localhost:9001 -kill job_201201041122_0004
12/01/04 12:03:04 INFO streaming.StreamJob: Tracking URL: http://localhost:50030/jobdetails.jsp?jobid=job_201201041122_0004
12/01/04 12:03:04 ERROR streaming.StreamJob: Job not successful. Error: NA
12/01/04 12:03:04 INFO streaming.StreamJob: killJob...
Streaming Command Failed!
```

15 · Reply · Share ›

Jayaprasad • 4 years ago

Hi Michael,

I was trying to execute this streaming job example. I am getting the following error while i am running this program.

```
hduser@ip-xxx-xxx-xxx-xxx:/usr/local/hadoop/conf$ /usr/local/hadoop/bin/hadoop jar /usr/local/hadoop/contrib/streaming/hadoop-*streaming*.jar
[...]
13/04/17 06:48:16 ERROR streaming.StreamJob: Job not successful. Error: # of failed Map Tasks exceeded allowed limit. FailedCount: 1. LastFail
13/04/17 06:48:16 INFO streaming.StreamJob: killJob...
Streaming Command Failed!
```

Can you please suggest any solution?

13 · Reply · Share ›

Chandrakant • 4 years ago

Michael,
I cant thank you enough for your single-cluster tutorial and this one. I am a complete newcomer to Hadoop and I was able to get running in a few hours, thanks to you!
Also, minor nitpick - just wanted to point out the map reduce programs could be shorter if you used the collections.Counter object provided by the python standard library. Here's a working solution that I used:
mapper.py

```
import sys

def run_map(f):
for line in f:
data = line.rstrip().split()
for word in data:
print(word)

if __name__ == '__main__':
run_map(sys.stdin)
```

see more

13 · Reply · Share ›

be_fair • 4 years ago

Great job Michael. I am a java developer and have never worked on python before. New to hadoop as well. I have gained a lot through your tutorials. I did whatever you suggested to do and it worked like a charm. I then thought of applying the code to a tab delimited file. I changed your mapper to the following.

```
#!/usr/bin/env python
import sys
# input comes from STDIN (standard input)
for line in sys.stdin:
# remove leading and trailing whitespace
line = line.strip()
# split the line into words
words = line.split('\t')
# increase counters
for word in words:
# write the results to STDOUT (standard output);
# what we output here will be the input for the
# Reduce step, i.e. the input for reducer.py
```

see more

9 ^ | ∨ • Reply • Share ›

Krishna • 6 years ago

There is a comment in the reducer program that the output of the mapper is sorted by key. Is this really relevant, because isnt the reducer supposed to get all the key value pairs with the same key or am I missing something here?
Why can't we simply sum up the values of all the key value pairs that come into a single reducer?

9 ^ | ∨ • Reply • Share ›

tej • 5 years ago

hey, i had a doubt

```
hduser@ubuntu:~$ cat /tmp/gutenberg/20417-8.txt | /home/hduser/mapper.py
```

or

```
bin/hadoop jar contrib/streaming/hadoop-*streaming*.jar -mapper /home/hduser/mapper.py -reducer /home/hduser/reducer.py -input
/user/hduser/gutenberg/* -output /user/hduser/gutenberg-output
```

can u tell me how i can get the input file name "20417-8.txt: in my mapper.py program..i am tryin to write inverted index program
i searched the Internet and ppl hav suggested to use os.environ["map_input_file"] but it doesn seem to work..

i am using hadoop-0.20.2 and python 2.6.6

plz help

8 ^ | ∨ • Reply • Share ›

Mehdi Boussarhane • 4 years ago

thank you so much, very interesting blog.
i have one question, can we Writing an Hadoop MapReduce Program in OpenCV ?
or, i have an openCV program, i would like to use it in hadoop, is possible or not ?

5 ^ | ∨ • Reply • Share ›

Anurag Prajapat • 4 years ago

great tutorial....gud for newbees who want to get their hand into hadoop!

5 ^ | ∨ • Reply • Share ›

Dipesh • 5 years ago

Thank you for such a thorough tutorial.

I setup a small cluster using multiple virtual machines in my computer. When I run the map-reduce command, the map task is completed, but reduce task gets stuck. I checked and rechecked the python code. There seem not be any problem. Any suggestion why this might be happening?

5 ^ | ∨ • Reply • Share ›

Maria • 4 years ago

Hi Noll,

How to parse and categorize system application log files in hadoop single node cluster.Is there any mapreduce coding???

3 ^ | ∨ • Reply • Share ›

Piyush Kansal • 5 years ago

Dear Michael,

- I am writing my code in Python. So, can you please suggest how can we introduce cascading using Hadoop Streaming without actually using "Cascading" package
- Do I need to save intermediate files in this case

I tried searching this on internet but could not come up with a definite answer

I have the following scenario: Map1->Red1->Map2->Red2

3 ^ | ∨ • Reply • Share ›

Praveen • 5 years ago

>>The job will read all the files in the HDFS directory /user/hduser/gutenberg, process it, and store the results in a single result file in the HDFS directory /user/hduser/gutenberg-output.

Shouldn't it one file per reducer in the o/p?

3 ^ | ∨ • Reply • Share ›

Michael G. Noll → Praveen • 5 years ago

@Praveen: Yes, in general it will be one file per reducer. In this example however the input files are so small so that it will be just a single file. But I'll

clarify the relevant section.
3 ^ | ⌄ • Reply • Share ›

jo • 4 years ago
i love much thi

i like much this tutorial
a good job Michael
thank you
2 ^ | ⌄ • Reply • Share ›

Pily • 4 years ago
Nithesh you can sort it afterwards.

```
bash# sort -k 2 -n -r part-00000|less
```
2 ^ | ⌄ • Reply • Share ›

Athar Noor ➦ Pily • 4 years ago
Can you please elaborate ? I get error on this line. How to use the command you suggested ? I want to sort the data on the descending order of count. Can you explain what k, 2, n, r, etc does ?
3 ^ | ⌄ • Reply • Share ›

Rob Guilfoyle • 5 years ago
This is hands down the best content I have come across yet for Map Reduce in the Hadoop environment. Very detailed and well written. Thanks so much for this!
2 ^ | ⌄ • Reply • Share ›

Hadoop map • 5 years ago
Awesome post dude. This tutoril regarding hadoop is very helpful to me. Thanks a lot
2 ^ | ⌄ • Reply • Share ›

Nikhil • 5 years ago
Thanks for the great set of tutorials on Hadoop, Michael.

I had a question. This is in the context of a distributed setup involving many nodes, and several large files stored on them with some replication (say three duplicate blocks per block). Now, when I run a standard hadoop streaming task like this one, and I **don't** specify values for the number of map and reduce tasks through mapred.*.tasks, what is the default behaviour like? Does it create some parallelism on its own or does it end up spawning a single task to get the job done?

Thanks again for the great articles.
2 ^ | ⌄ • Reply • Share ›

Thyag • 5 years ago
It was a "wow" moment when I checked my part-00000 file!! Thanks for the nice tutorial
2 ^ | ⌄ • Reply • Share ›

jeff • 5 years ago
I am getting, as another poster (Anuj) was, the Streaming Command Failed! message. However mine happens apparently much sooner: I have not gotten beyond the following:

```
packageJobJar: [/home/jmiller/wordcount/mapper.py, /home/jmiller/wordcount/reducer.py, /tmp/hadoop-jmiller/hadoop-unjar8760597989207755800/] []
/tmp/streamjob1624565313346212981.jar tmpDir=null
Streaming Command Failed!
```

It seems to me that the absence of the input directory or insufficient permissions might cause this failure, but the directory does exist in HDFS, the permission is rwx for everyone on the directory and its contents. Same thing with the output directory.

Could the input file be in the wrong format? Is there a place where more error info would be displayed?

Thanks,

Jeff
2 ^ | ⌄ • Reply • Share ›

X.J ZHOU • 4 years ago
very useful tutorial
1 ^ | ⌄ • Reply • Share ›

Pavel Odintsov • 4 years ago

Hello,

Please fix link http://hadoop.apache.org/co... to http://hadoop.apache.org/do... because first link is broken.

1 ^ | v • Reply • Share ›

Michael G. Noll Owner ➦ Pavel Odintsov • 4 years ago

Thanks, odintsov. Fixed.

1 ^ | v • Reply • Share ›

Chris Hayes • 4 years ago

Python has worker threading with a Pool (http://docs.python.org/2/li... object that has a 'map' function that can work for both the map and reduce functionality.

1 ^ | v • Reply • Share ›

john • 4 years ago

Hello, Mike,

I was trying your code for the first easy mapper.py and reducer.py above.

For some reasons, when I do
echo "foo foo quux labs foo bar quux" | python /home/hduser/mapper.py | sort -k1, 1 | python reducer.py

no results come out. No error messages, either. I do not know what's wrong.

I am using hadoop 1.0.4.tar.gz on my unbunto 10.4.

Would you please advise me of how to fix this problem?

Thank you

1 ^ | v • Reply • Share ›

Dakota Reier ➦ john • 4 years ago

try to remove "python" from your code and just pipe into the script.py

2 ^ | v • Reply • Share ›

Chuchun Kang ➦ Dakota Reier • 3 years ago

I change some code and run it smoothly! Maybe it will help you.

^ | v • Reply • Share ›

shiva krishna • 4 years ago

I am using hadoop to process an xml file,so i had written mapper file , reducer file in python.

suppose the input need to process is **test.xml**.

mapper.py file

```
import sys
import cStringIO
import xml.etree.ElementTree as xml

if __name__ == '__main__':
buff = None
intext = False
for line in sys.stdin:
line = line.strip()
if line.find("<row") != -1:
.............
```

see more

1 ^ | v • Reply • Share ›

Fred Mailhot • 4 years ago

In regards to my previous comment: it was the basic mapper.py and reducer.py, not the ones making use of iterators/generators...

FM.

1 · Reply · Share ›

Fred Mailhot • 4 years ago

In re: timing...I ran the wordcount example on the 3 Gutenberg texts:

- using straight Hadoop (Java mapper & reducer, no streaming): 38s
- using mapper.py and reducer.py from above with Hadoop streaming: 44s

Not a very big timing hit at all.

FM.

1 · Reply · Share ›

shabeera • 4 years ago

very good tutorial

1 · Reply · Share ›

Fabio Pedrazzoli • 4 years ago

Thank you very much for your time for this tutorial Michael, it's super neat and very well explained, it went smooth as silk on Ubuntu 12.04.1 ;)

Cheers
Fabio

1 · Reply · Share ›

Siamac • 5 years ago

Excellent tutorial. Thank you v.m. Michael.

1 · Reply · Share ›

Jack Coughlin • 5 years ago

Fantastic tutorial, thanks so much! What is your sense of the performance impact of using Hadoop Streaming versus a custom jar, over and above the impact of using an interpreted language like Python?

1 · Reply · Share ›

Michael G. Noll ➜ Jack Coughlin • 5 years ago

@Jack Coughlin: Normally Hadoop Streaming is a little bit slower than native (Java) MapReduce jobs.

1 · Reply · Share ›

Word Counter • 5 years ago

Thanks for the tutorial mate. I used this to help me make some awesome applications.

1 · Reply · Share ›

Darshan Hegde • 5 years ago

I'm a newbee to hadoop, was trying out this example. Some how I'm getting the following error:

```
[root@localhost src]# hadoop jar /usr/lib/hadoop-0.20/contrib/streaming/hadoop-*streaming*.jar -file /data/hduser/src/mapper.py -mapper
/data/hduser/src/mapper.py -file /data/hduser/src/reducer.py -reducer /data/hduser/src/reducer.py -input /data/hduser/gutenberg/* -output
/data/hduser/gutenberg-output/
File: /data/hduser/src/mapper.py does not exist, or is not readable.
Streaming Command Failed!
[root@localhost src]#
[root@localhost src]# hadoop dfs -ls /data/hduser/src
Found 2 items
-rw-r--r-- 1 cloudera supergroup 591 2012-07-07 15:27 /data/hduser/src/mapper.py
-rw-r--r-- 1 cloudera supergroup 1129 2012-07-07 15:27 /data/hduser/src/reducer.py
```

But the path is correct. Can anybody please help ?

1 · Reply · Share ›

Michael G. Noll ➜ Darshan Hegde • 5 years ago

@Darshan Hegde: The script file you specify with the <tt>-file</tt> and <tt>-reducer</tt> options must be a local file, not a file in HDFS.

1 · Reply · Share ›

rohan • 5 years ago

Thanks a lot for such a brilliant tutorial .

1 · Reply · Share ›

Hamish Drewry • 5 years ago

A brilliant tutorial - perhaps one of the best I have come across.

1 · Reply · Share ›

Nithesh • 5 years ago
Awesome Tutorial, Very well explained.

Left with a single question, How would I sort this output file in descending order of count? (word with the highest count appears first)

Any help is much appreciated.
1 • Reply • Share ›

Anil Kumar • 5 years ago
The best and simple Hadoop tutorial . I had a very successful run on the first try itself on Ubuntu 11.04 server VMs running in OpenNebula. I automated most of configuration part through contextualization and it started looking much faster to setup and run.
Thanks a lot once again for the tutorial.
1 • Reply • Share ›

Agila • 5 years ago
hi any one help me...

i am working on a project,where i am searching a word in collection of file which is a simple python,i need to convert it into a parallel process using mapreduce (hadoop),,,,,,
1 • Reply • Share ›

jeff • 5 years ago
disregard earlier posts by me -- works fine under the correct user.

thanks very much for this article!
1 • Reply • Share ›

jeff • 5 years ago
Regarding my earlier question, could it simply be that such an early failure is due to Python not being installed on all of the nodes of the Hadoop cluster?
1 • Reply • Share ›

Sachin • 5 years ago
I want to write hadoop streaming job for map reduce, however I am not aware how to get filename where I am getting input line. How I can do that?
1 • Reply • Share ›

Michael G. Noll → Sachin • 5 years ago
@Sachin: Hadoop sets job configuration parameters as environment variables when streaming is used. For instance, <tt>os.environ["mapred_job_id"]</tt> gives you the <tt>mapred.job.id</tt> configuration property. Off the top of my head the name of the input file will be in <tt>os.environ["map_input_file"]</tt>. Note that Hadopo replaces non-alphanumeric characters such as dots "." with underscores.

See Tom White's book *Hadoop: The Definitive Guide* (2nd ed.), page 187 for more information.
5 • Reply • Share ›

Load more comments

ALSO ON **MICHAEL G. NOLL**

Wirbelsturm: 1-Click Deployments of Storm and Kafka clusters with Vagrant and Puppet
6 comments • 3 years ago•
Michael G. Noll — Thibaud -- yes, I have been successfully using Kafka 0.8 and Storm 0.9 with Wurstmeister's spout. I am also about to …

Understanding HDFS quotas and Hadoop fs and fsck tools
1 comment • 5 years ago•
Eric — Would you like to have some of your blog's content republished on DZone.com? We're trying to expand our readership of advanced developers …

Using Avro in MapReduce jobs with Hadoop, Pig, Hive
3 comments • 4 years ago•
super wesman — I'm trying to use your Java example here, but it refers to the deprecated JobConf class. The "official" ColorCount tutorial on the Avro site …

Sending Metrics from Storm to Graphite
10 comments • 3 years ago•
Ishwara Varnasi — Hello, thank you for very informative blog. I have a question about collecting metrics for a flow that covers multiple spout/bolts. I …

About Me

I am a software engineer turned product manager based in Switzerland, Europe. In my day job I am working on products at Confluent, the US startup founded by the creators of Apache Kafka. Read more »

Contact

michael@michael-noll.com

Follow Me

Twitter
Blog RSS
GitHub
Slideshare

Recent Posts

- Integrating Kafka and Spark Streaming: Code Examples and State of the Game
- Apache Storm 0.9 training deck and tutorial
- Apache Kafka 0.8 training deck and tutorial
- Integrating Kafka and Storm: Code Examples and State of the Game
- Wirbelsturm: 1-Click Deployments of Storm and Kafka clusters with Vagrant and Puppet