Dig Data Management Assignment 2

Task2 - Word Count Task on a Serie of Novels Based on Hadoop MapReduce

In this task, we count the words in a series of Novels applying Hadoop MapReduce framework.

How We Implement

In this task, we use Python to go through it. Hadoop provides a streaming method to make it possible to run MapReduce tasks for all programming languages.
Here shows the simplistic codes of mapper,

```
#!/usr/bin/python3.5
# mapper.py
import sys
import jieba

for line in sys.stdin:
    words = jieba.cut(line.strip(), cut_all=False, HMM=False)

    for word in words:
        if len(word) > 1:
            print ('%s\t%s' % (word, 1))
```

It's quite simple and straightforward. We can design any parsing pattern to build up our key-value pairs. We use package 'Jieba' to tokenize Chinese sentences.

```
#!/usr/bin/python3.5
# reducer.py
import sys

current_word, current_count = None, 0

for line in sys.stdin:
    word, count = line.strip().split('\t', 1)
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_count:
            print('%s\t%s' % (current_word, current_count))
        current_count, current_word = count, word
if current_count:
    print('%s\t%s' % (current_word, current_count))
```

We need to treat the key-value pairs as string and do customized parsing here, and then we do a simple counting task. Note that the standard input key-value pairs are sorted by key before passed into reducers, it helps simplify the logic of counting.

Result of Word Count

Top 10

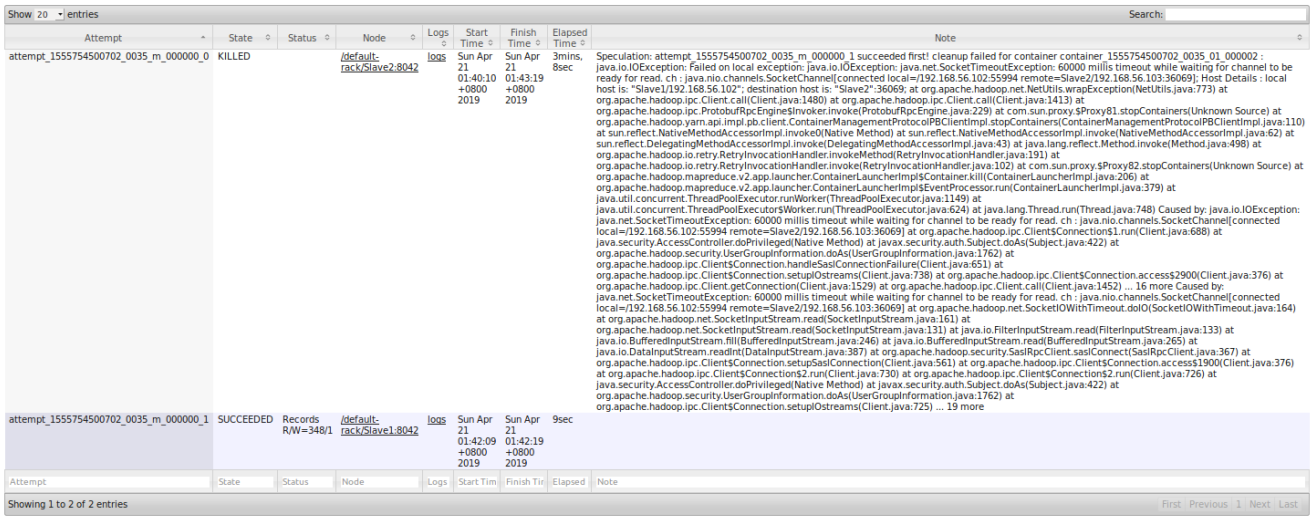
```
1  说道    13714
2  什么    12738
3  自己    10841
4  韦小宝  9920
5  一个    9361
6  咱们    7017
7  武功    6804
8  一声    6520
9  不是    6125
10 师父    6071
```

Last 10 (Chinese only)

```
1  一不怕苦      1
2  一下脸        1
3  一下手        1
4  一下子把      1
5  一万间        1
6  一万户        1
7  一万多        1
8  一万六千多    1
9  一万余        1
10 一万九千      1
```

Observations

1. Kiled/Failed Tasks would be transfered to another nodes to run



Show 20 entries — Search:

Attempt	State	Status	Node	Logs	Start Time	Finish Time	Elapsed Time	Note
attempt_1555754500702_0035_m_000000_0	KILLED		/default-rack/Slave2:8042	logs	Sun Apr 21 01:40:10 +0800 2019	Sun Apr 21 01:43:19 +0800 2019	3mins, 8sec	Speculation: attempt_1555754500702_0035_m_000000_1 succeeded first! cleanup failed for container container_1555754500702_0035_01_000002 : java.io.IOException: Failed on local exception: java.io.IOException: java.net.SocketTimeoutException: 60000 millis timeout while waiting for channel to be ready for read. ch : java.nio.channels.SocketChannel[connected local=/192.168.56.102:55994 remote=Slave2/192.168.56.103:36069]; Host Details : local host is: "Slave1/192.168.56.102"; destination host is: "Slave2":36069; at org.apache.hadoop.net.NetUtils.wrapException(NetUtils.java:773) at org.apache.hadoop.ipc.Client.call(Client.java:1480) at org.apache.hadoop.ipc.Client.call(Client.java:1413) at org.apache.hadoop.ipc.ProtobufRpcEngine$Invoker.invoke(ProtobufRpcEngine.java:229) at com.sun.proxy.$Proxy81.stopContainers(Unknown Source) at org.apache.hadoop.yarn.api.impl.pb.client.ContainerManagementProtocolPBClientImpl.stopContainers(ContainerManagementProtocolPBClientImpl.java:110) at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.lang.reflect.Method.invoke(Method.java:498) at org.apache.hadoop.io.retry.RetryInvocationHandler.invokeMethod(RetryInvocationHandler.java:191) at org.apache.hadoop.io.retry.RetryInvocationHandler.invoke(RetryInvocationHandler.java:102) at com.sun.proxy.$Proxy82.stopContainers(Unknown Source) at org.apache.hadoop.mapreduce.v2.app.launcher.ContainerLauncherImpl$Container.kill(ContainerLauncherImpl.java:206) at org.apache.hadoop.mapreduce.v2.app.launcher.ContainerLauncherImpl$EventProcessor.run(ContainerLauncherImpl.java:379) at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) at java.lang.Thread.run(Thread.java:748) Caused by: java.io.IOException: java.net.SocketTimeoutException: 60000 millis timeout while waiting for channel to be ready for read. ch : java.nio.channels.SocketChannel[connected local=/192.168.56.102:55994 remote=Slave2/192.168.56.103:36069] at org.apache.hadoop.ipc.Client$Connection$1.run(Client.java:688) at java.security.AccessController.doPrivileged(Native Method) at javax.security.auth.Subject.doAs(Subject.java:422) at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1762) at org.apache.hadoop.ipc.Client$Connection.handleSaslConnectionFailure(Client.java:651) at org.apache.hadoop.ipc.Client$Connection.setupIOstreams(Client.java:738) at org.apache.hadoop.ipc.Client$Connection.access$2900(Client.java:376) at org.apache.hadoop.ipc.Client.getConnection(Client.java:1529) at org.apache.hadoop.ipc.Client.call(Client.java:1452) ... 16 more Caused by: java.net.SocketTimeoutException: 60000 millis timeout while waiting for channel to be ready for read. ch : java.nio.channels.SocketChannel[connected local=/192.168.56.102:55994 remote=Slave2/192.168.56.103:36069] at org.apache.hadoop.net.SocketIOWithTimeout.doIO(SocketIOWithTimeout.java:164) at org.apache.hadoop.net.SocketInputStream.read(SocketInputStream.java:161) at org.apache.hadoop.net.SocketInputStream.read(SocketInputStream.java:131) at java.io.FilterInputStream.read(FilterInputStream.java:133) at java.io.BufferedInputStream.fill(BufferedInputStream.java:246) at java.io.BufferedInputStream.read(BufferedInputStream.java:265) at java.io.DataInputStream.readInt(DataInputStream.java:387) at org.apache.hadoop.security.SaslRpcClient.saslConnect(SaslRpcClient.java:367) at org.apache.hadoop.ipc.Client$Connection.setupSaslConnection(Client.java:561) at org.apache.hadoop.ipc.Client$Connection.access$1900(Client.java:376) at org.apache.hadoop.ipc.Client$Connection$2.run(Client.java:730) at org.apache.hadoop.ipc.Client$Connection$2.run(Client.java:726) at java.security.AccessController.doPrivileged(Native Method) at javax.security.auth.Subject.doAs(Subject.java:422) at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1762) at org.apache.hadoop.ipc.Client$Connection.setupIOstreams(Client.java:725) ... 19 more
attempt_1555754500702_0035_m_000000_1	SUCCEEDED	Records R/W=348/1	/default-rack/Slave1:8042	logs	Sun Apr 21 01:42:09 +0800 2019	Sun Apr 21 01:42:19 +0800 2019	9sec	

Showing 1 to 2 of 2 entries — First Previous 1 Next Last

2. Error caused from lack of RAM (exit code=137)
   We need to set `yarn-site.xml` appropriately to solve the problem over lack of RAM.
   `yarn.nodemanager.resource.memory-mb` is set to be `1600` to solve the error this time.
3. Some files and packages(libraries) are required to be allocated and installed on every single nodes. In this task, bugs did torture us for a while for we didn't install package 'Jieba' for Python on Slave1 and Slave2 nodes.
4. The command format matters and diverses in different verions when you use streaming method to run MapReduce tasks. The final legal format turned out to be something like

```
hadoop jar /usr/local/hadoop/hadoop-2.7.7/share/hadoop/tools/lib/hadoop-streaming-2.7.7.jar
-D mapreduce.job.name="novel mapreduce" \
-mapper word_count_mapper.py -file ./word_count_mapper.py \
-reducer word_count_reducer.py -file ./word_count_reducer.py \
-input novel/ -output novel_output
```

Reflections

1. This time, we're rather more familiar with the relation and involved items of 'NameNode', 'DataNode', 'SourceManager' and 'NodeManager.'
2. Trying to reset configuration such as `yarn-site.xml` to solve bugs help us be more into the structure of Hadoop.