**Hadoop Raspberry Pi performance tests and optimization**

For those tests I used the same sample text files as for the single node setup.

[Download sample files](http://www.widriksson.com/wp-content/uploads/2014/10/hadoop_sample_txtfiles.tar.gz)

Those tests are to highlight some of the issues that can occur when you run hadoop the first time and especially in a Raspberry Pi cluster since it is very limited.  The tests will do some things “very wrong” in order to point out the issues that can occur. If you just want to optimize for the Raspberry Pi you can check out the changes that are made in the last test. Also please notice that those test are done for the mediuim.txt sample file provided above and is no “general-purpose” optimizations. If you have used Hadoop before those test are probably of no use for you since you already have figured out what to do

**First run**

Start two three SSH terminal windows – one for each node. Then start a monitoring program in each of them. I used [nmon](http://nmon.sourceforge.net/pmwiki.php" \o "Nmon" \t "_blank) but you could as well go with top or any other monitor of your choice. Now you will be able to watch the load put on your Pi’s by the WordCount MapReduce program.

Go back to your main terminal window (for node1) and upload files to HDFS and run the WordCount program:

hadoop dfs -copyFromLocal mediumfile.txt /mediumfile2.txt

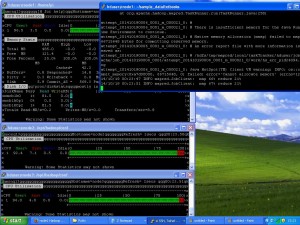
hadoop jar /opt/hadoop/hadoop-examples-1.2.1.jar wordcount /mediumfile2.txt /mediumfile2-out.txt

Then watch the monitors of your nodes. Not much going on on node2 and node3? But node1 is running all of the job? The JobTracker is not distributing the jobs out to our other nodes. This is because as default HDFS is configured for use of really large files and the block-size is set to 64mb. Our file is only 35MB (medium.txt) hence it will only be split into one block and hence only one node can work on it.

**Second run**

**Optimize block size**

In order to tackle the block-size problem above edit the conf/hdfs-site.xml on all your nodes and to the following:

[](http://www.widriksson.com/wp-content/uploads/2014/10/2.jpg)**hdfs-site.xml**

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.block.size</name>

<value>1048576</value>

</property>

</configuration>

The above configuration will set block size to 1mb. Lets make another run and see what happens:

hadoop jar /opt/hadoop/hadoop-examples-1.2.1.jar wordcount /mediumfile2.txt /mediumfile3-out.txt

|  |  |  |
| --- | --- | --- |
| **File** | **Size** | **WordCount execution time (mm:ss)** |
| mediumfile.txt | 35MB | 14:24 |

Still not very impressive, right? It’s even worse than the single node setup… This is due to that when you upload a file to HDFS and you do it locally e.g. from a datanode (which we are doing since node1 is a datanode) it will copy the data local. Hence all our blocks are now on node1. Hadoop also tries to run jobs as close as possible to where the data i stored to avoid network overhead. However some of the blocks might get copied over the node2 and node3 for processing but node1 is moste likely to get the most load. Also node1 is running as NameNode and JobTracker and has additional work to do. Also I noticed in several of the jobs the job failed with out of memory exception as seen in picture to the right. Then 1mb of block-size is might be to small even on the Pi’s depending on our file size. But now will have our file split into 31 blocks where each block will cause a bit of overhead. (The less blocks we need the better – if we still can evenly spread the blocks across our nodes).

**Third run**

**Optimize block size**

Lets make another try. This time we change the block-size to 10mb: (conf/hdfs-site.xml)

**hdfs-site.xml**

<property>

<name>dfs.block.size</name>

<value>1048576</value>

</property>

**Format NameNode**

Node1 got a bit overloaded in the previous scenario we will now remove its role as TaskTracker and DataNode. Before we can remove node1 as DataNode format the namenode (as we otherwise would end up with dataloss since we have the dfs.replication set to 1 our data is not redundant)

On all nodes:

rm -rf /hdfs/tmp/\*

On node1:

hadoop namenode -format

**Configure Node1 to only be master**

Edit conf/slaves and remove node1. Then stop and start the cluster again:

stop-mapred.sh

stop-dfs.sh

start-dfs.sh

start-mapred.sh

Then upload our sample data and start the job again:

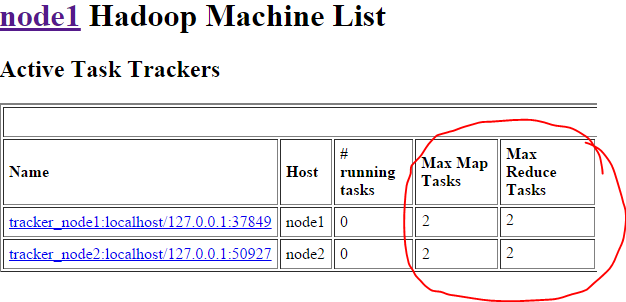
hadoop dfs -copyFromLocal mediumfile.txt /mediumfile.txt

hadoop jar /opt/hadoop/hadoop-examples-1.2.1.jar wordcount /mediumfile.txt /mediumfile-out.txt

|  |  |  |
| --- | --- | --- |
| **File** | **Size** | **WordCount execution time (mm:ss)** |
| mediumfile.txt | 35MB | 6:26 |

So now we actually got a bit of improvement compared to a single node setup. This is due to that when you upload a file to HDFS from a client e.g. not locally on the DataNode Hadoop will try to spread the blocks evenly among the nodes and not as in our previous test. However this is still not optimal since now we are not using node1 to its full processing potential. What we would like to do is to have all nodes as DataNodes and TaskTrackers with the file blocks spread nice and evenly on all of them.

Also if you go to http://node1:50030 and click on number 3 under “nodes” in the table you will see that our nodes are setup to be able to handle 2 map tasks (See picture below). However the Raspberry Pi is a one (and one pretty slow) processor core. It will most likely not perform well of running multiple tasks. So lets set things correct in the last run.

[](http://www.widriksson.com/wp-content/uploads/2014/10/hadoop-web-task-trackers1.png)

**Fourth run**

**Re-format NameNode (again)**

On all nodes:

rm -rf /hdfs/tmp/\*

On node1:

hadoop namenode -format

**Optimize block size**

Lets make the block-size a bit smaller than before. Lower it to 5mb.

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.block.size</name>

<value>5242880</value>

</property>

</configuration>

**Configure TaskTrackers max tasks**

As mentioned in the last text of previous test. If you go to http://node1:50030 and look on your nodes you will se that max map and reducer tasks are set to 2. This is to much for the Raspberry Pi’s. We will change max map and reducer tasks to the amount of CPU cores each device has: 1.

On all your nodes:

**mapred-site.xml**

 <configuration>

<property>

<name>mapred.job.tracker</name>

<value>node1:54311</value>

</property>

<property>

<name>mapred.tasktracker.map.tasks.maximum</name>

<value>1</value>

</property>

<property>

<name>mapred.tasktracker.reduce.tasks.maximum</name>

<value>1</value>

</property>

</configuration>

**Configure Node1 back to act as both slave and master**

Edit conf/slaves and add node1. Then stop and start the cluster again:

stop-mapred.sh

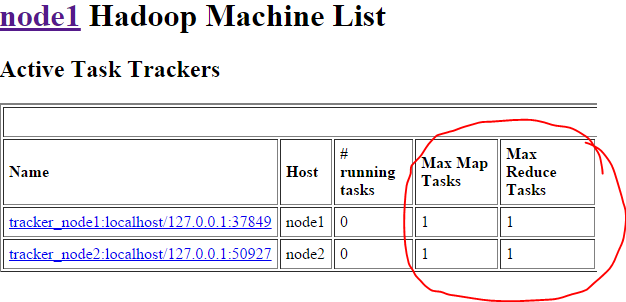
stop-dfs.sh

start-dfs.sh

start-mapred.sh

**Verify Max Map Tasks and Max Reduce Tasks**

Go to http://node1:50030, click your nodes in the cluster summary table and ensure max map and max reduce tasks are set to 1:

[](http://www.widriksson.com/wp-content/uploads/2014/10/hadoop-web-task-trackers.png)

**Upload Sample file (again)**

hadoop dfs -copyFromLocal mediumfile.txt /mediumfile.txt

**Balance HDFS file system**

Of course it is possible to upload data on one node and the distribute it evenly across all nodes. Run the following to see how our mediumfile.txt currently is stored:

hadoop fsck /mediumfile.txt -files -blocks -racks

As you most likely will see all the blocks are stored on node1. In order to spread the blocks evenly on all nodes run the following:

hadoop balancer -threshold 0.1

The threshold parameter is a float value from 0 to 100 (percentage). The lower the more balanced your blocks will be. Since we only have one file and that file is a very small percentage of our total storage we need to set it really small to put the balancer into work. After the balancing is complete very the file blocks again by:

hadoop fsck /mediumfile.txt -files -blocks -racks

**Last run**

hadoop jar /opt/hadoop/hadoop-examples-1.2.1.jar wordcount /mediumfile.txt /mediumfile-out.txt

|  |  |  |
| --- | --- | --- |
| **File** | **Size** | **WordCount execution time (mm:ss)** |
| mediumfile.txt | 35MB | 5:26 |