Assignment 5A

[You can work on this assignment in small groups, but write up your submissions individually. Groups will need to take turns with the hardware: no sympathy will be available if you all wait until the last minute. Alternate version: with virtual machines.]

The actual Cluster that we have for this course is nice, but it has one notable limitation: it actually works.

Part of the selling point of the Hadoop technologies is that they are fault tolerant. It's one thing to read that in the docs, but another to see it actually play out. James will be quite cross if we go into the server room to start unplugging nodes to see what happens.

So, I built an alternate cluster to meet this need (and let's be honest, because it was fun)…

Raspberry Pi Cluster

The cluster consists of seven Raspberry Pi 2 model B *(https://www.raspberrypi.org/products/raspberry-pi-2-model-b/)* s, each with an 8GB microSD card (boot, OS, home) and 32GB USB flash drive (HDFS). That gives a total of 28 cores, and 280GB of storage. Full parts list and configuration details *(https://github.com/gregbaker/raspberry-pi-cluster)* are on GitHub, if you're interested.

Working with the cluster

The computer you're working on needs to be on the internal network for the cluster. Connect a laptop to one of the LAN (yellow) ethernet ports on the router.

TODO: or connect to the wifi? or use wifi for bridge to outside internet?

That should be all the configuration necessary: plug in the cluster's power bar and turn it on.

The nodes are named `master.local` (NameNode, ResourceManager, JobTracker, login) and `hadoop1.local` to `hadoop6.local` (DataNodes and TaskTrackers). [I will adopt the convention of putting a `computername>` prompt on commands for this assignment: there are too many to keep track of otherwise.]

Once you have the Pis running, you should be able to SSH to the master node:

```
yourcomputer> ssh pi@master.local
```

The password for the `pi` account is on a note in the case.

You need to start the services on the nodes (this starts HDFS and YARN):

```
master> start-all
```

You can copy code by SCP to `master.local` and then start jobs like on the real cluster with `spark-submit`, or start a Spark shell with `pyspark`.

Web Frontends

- HDFS NameNode: http://master.local:50070/ *(http://master.local:50070/)*
- YARN ResourceManager: http://master.local:8088/ *(http://master.local:8088/)*

Notes on the Simulation

Rather than actually destroying disks and nodes, the instructions below do a couple of things to simulate failure. These are:

- Restarting a node (`sudo reboot`) stops all of the Hadoop processes and effectively makes the node disappear from the cluster. It is as if the node failed completely (until we restart the Hadoop processes).
- The script `clear-dfs` deletes all of the HDFS files on the node. Doing this (while HDFS is not running) is effectively the same as a disk failing and being replaced.

Expanding the cluster

Start by shutting down everything:

```
master> halt-all
```

Unplug the power from node 6: we will be starting with a 6 node cluster (5 + master) and seeing how easy it is to add nodes to Hadoop.

Power up the cluster and start everything (minus node 6):

```
yourcomputer> ssh pi@master.local
master> start-all
```

[It will fail to start on `hadoop6`, but otherwise should be okay.] Give it some time: these are not fast machines. After 30 seconds or so, you should be able to visit the HDFS namenode and YARN application master (URLs above) and see 5 active DataNodes/TaskTrackers.

We are imagining that `hadoop6` is a brand new node being added to expand capacity on the cluster. Plug it in and wait for it to boot. You should soon be able to connect to it. Clear any existing HDFS data on it:

```
yourcomputer> ssh pi@hadoop6.local
hadoop6> clear-dfs
```

The `clear-dfs` script deletes all of the HDFS data in `/hadoop/*` on that computer. Effectively, `hadoop6` is now a new node with a newly-formatted hard drive.

To add it to the cluster, on `master` simply:

```
master> start-all
```

This will fail on nodes 1–5 (because the services are already running), but will bring services up on `hadoop6`. Check in the web frontends that it has appeared as a DataNodes and TaskTrackers. The overall capacity (storage, memory, cores) should have increased appropriately as well.

Drive Failure

HDFS stores redundant copies (by default three copies of each block). There are two reasons: (1) so the content is available on multiple nodes as input to jobs, and (2) to provide protection against disk failure.

So, what actually happens when a disk fails?

Find a file in the HDFS filesystem: go to the web frontend (http://master.local:50070/ *(http://master.local:50070/)*) → Utilities → Browse the filesystem. When you click a filename, a popup window gives you some metadata, including the list of nodes where that file is stored.

Pick one of the nodes where that file is stored. We're going to simulate failure of that disk. For the commands below, I'm going to assume it's `hadoop5` that we're failing.

SSH to the sacrificial node. Restart it: this will stop the HDFS processes and probably would have been necessary to replace the drive anyway:

```
yourcomputer> ssh hadoop5.local
hadoop5> sudo reboot
```

Have a look at the list of Datanodes in the web frontend. (Keep reloading the page to get updated statuses.) What happens?

After HDFS gives up on the “failed” node, have a look at the number of “Under-Replicated Blocks” on the NameNode front page. What happens to the file you were watching? Was the replication maintained? [❓]

Note: the dead-node timeouts have been made very short on the Pi cluster. They are typically much longer so something like a network switch reboot doesn't declare everything behind it dead.

Replacement Drive

Now we will wipe the HDFS data from the disk, as we would see on a newly-provisioned drive:

```
yourcomputer> ssh hadoop5.local
hadoop5> clear-dfs
```

As before, get things started from the master node:

```
yourcomputer> ssh master.local
master> start-all
```

In the web frontend, you should see the node with the “replaced” drive back in the cluster. It should have more free space than the other nodes. (If that really bothers you, you can run `hdfs balancer -threshold 2` to rebalance.)

Aside: You might have noticed that the “add a new node” and “replace a drive” procedures are quite similar. To me that makes it feel like what's in a Hadoop cluster is very ephemeral: the cluster consists of whatever resources happen to be there just at the moment. The YARN/HDFS machinery takes care of the rest.

Computation Failure

Now we will simulate failure of a node *during* a job's execution.

Make sure the cluster is up and running. Start a job that will take long enough to give you some time: this can even by the `pyspark` shell with:

```
sc.range(1000000000, numSlices=100).sum()
```

Start the job, and once the executors are start getting something done, look at the Spark frontend (http://master.local:8088/ *(http://master.local:8088/)*). Select the running application → ApplicationMaster.

Find a node that's doing some work (i.e. one listed in the “Executors” tab with tasks running). Log into that node and restart it: this will rudely stop the NodeManager and leave any work unfinished. If node 3 is the one being killed:

```
yourcomputer> ssh hadoop3.local
hadoop3> sudo reboot
```

Keep watching the attempt page in the frontend (reloading as necessary), and the output of your application in the terminal. How does YARN respond to the node failing? [❓]

Other Things To Try

There are many things our little cluster can do (if slowly). These aren't required for the assignment, but may be interesting.

Pick a (small) file and decrease its replication factor to one:

```
master> hdfs dfs -setrep -w 1 /user/pi/output/part-r-00000
```

What happens if the drive it's on "fails"? Or if you bring up the cluster without that node?

More experiments

If you'd like to experiment more, feel free. I'll just ask if you play with any config files, that you keep backups and restore when you're done. Please also leave the `/user/pi/wordcount` and `/user/pi/euler-input` HDFS directories in place for the next group.

Shutting Down

Please shut the system down gracefully when you're done:

```
master> stop-all # somewhat optional, but nice
master> halt-all
```

After a new seconds for the machines to shut down, you can power off/unplug.

Questions

In a text file `answers.txt`, answer these questions:

1. What happened to the HDFS file when one of the nodes it was stored on failed?
2. How did YARN/MapReduce behave when one of the compute nodes disappeared?
3. Were there more things you'd like to try with this cluster, or anything you did try that you think should have been in this assignment?

Submitting

Submit your `answers.txt` to Assignment 5A in CourSys.

Updated Sun Nov. 05 2017, 22:57 by ggbaker.