Assignment 10: Virtual Machine version

You can work on this assignment in small groups, but write up your submissions individually. Alternate version: with the Raspberry Pi cluster.

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. Ryan will be quite cross if we go into the server room to start unplugging nodes to see what happens.

But we can use virtual machines to create our own cluster…

Virtual Machine Cluster

Requirements for the computer where you do this: a 4-core processor, 8GB memory, 25GB free disk space. The desktops in the computer lab meet these specs, so you can certainly do this assignment there.

Start by cloning the VM cluster project *(https://github.com/gregbaker/vm-cluster)* which contains configuration recipes for the cluster.

```
git clone https://github.com/gregbaker/vm-cluster.git
```

There are a few settings at the top of the `Vagrantfile` that you can tweak if you like, particular the `NUM_NODES` which should be **at least 3** for this exercise, but can be a little higher if you want (and your computer can handle the memory allocation).

Once you're happy with the configuration, start the cluster. This will take a few minutes but should be completely automatic:

```
git submodule update --init --recursive
vagrant up
```

Working with the cluster

The nodes are named `master.local` (NameNode, ResourceManager, JobTracker, login) and `hadoop1.local` to `hadoop3.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 VMs running, you should be able to SSH to the master node:

```
yourcomputer> vagrant ssh master
```

There is a little more setup to do for this assignment:

```
master> dfs-format
master> start-all # wait a minute for the services to start
master> make big-file
```

You can copy code into the project directory and it will be available on the nodes at `~/project`. You can start jobs like on the real cluster with `spark-submit`, or start a Spark shell with `pyspark`.

Web Frontends

- HDFS NameNode: http://localhost:9870/ *(http://localhost:9870/)*
- YARN ResourceManager: http://localhost:8088/ *(http://localhost: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 (`vagrant reload hadoop1`) 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 stopping the Hadoop services and stop one of the nodes:

```
master> stop-all
master> exit
yourcomputer> vagrant halt hadoop1
```

Now start the cluster again:

```
yourcomputer> vagrant ssh master
master> start-all
```

[It will fail to start on the node that's off, but otherwise should be okay.] Give it some time: after a few seconds, you should be able to visit the HDFS namenode and YARN application master (URLs above) and see the n-1 active DataNodes/TaskTrackers.

We are imagining that `hadoop1` 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> vagrant up hadoop1
yourcomputer> vagrant ssh hadoop1
hadoop1> clear-dfs
```

The `clear-dfs` script deletes all of the HDFS data in `/hadoop/*` on that computer. Effectively, `hadoop1` 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 *n*-1 nodes (because the services are already running), but will bring services up on `hadoop1`. 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 (here: two or three, depending on the number of nodes you started). 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://localhost:9870/ *(http://localhost:9870/)*) → 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 `hadoop2` that we're failing.

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

```
yourcomputer> vagrant reload hadoop2
```

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 VM 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> vagrant ssh hadoop2
hadoop2> clear-dfs
```

As before, get things started from the master node:

```
yourcomputer> vagrant ssh master
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.

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(10000000000, numSlices=100).sum()
```

Start the job, and once the executors are start getting something done, look at the Spark frontend (http://localhost:8088/ *(http://localhost:8088/)*). Select the running application → ApplicationMaster. You will likely have to **edit the URL** of the ApplicationMaster/Spark Frontend: replace `master.local` with `localhost`.

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> vagrant reload hadoop3
```

Keep watching the attempt page in the frontend (reloading as necessary), and the output of your application in the terminal. How does YARN/Spark 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 output/part-r-00000
```

What happens if a node containing some of that file disappears for a few minutes (and then rejoins the cluster)? What if the drive it's on “fails”?

Of course, you're free to experiment further with your VMs: you can always `vagrant destroy; vagrant up` if you want to start fresh.

Shutting Down

If you want to shut down gracefully when you're done:

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

When you're done with the VMs (and especially if you're on the lab computers, **please free up the disk space**):

```
yourcomputer> vagrant destroy
```

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 10 in CourSys.

Updated Wed Nov. 14 2018, 13:59 by ggbaker.