AN INTRODUCTION TO APACHE YARN

Apache Yarn 101

Here we describe Apache Yarn, which is a resource manager built into Hadoop. But it also is a stand-alone programming framework that other applications can use to run those applications across a distributed architecture.

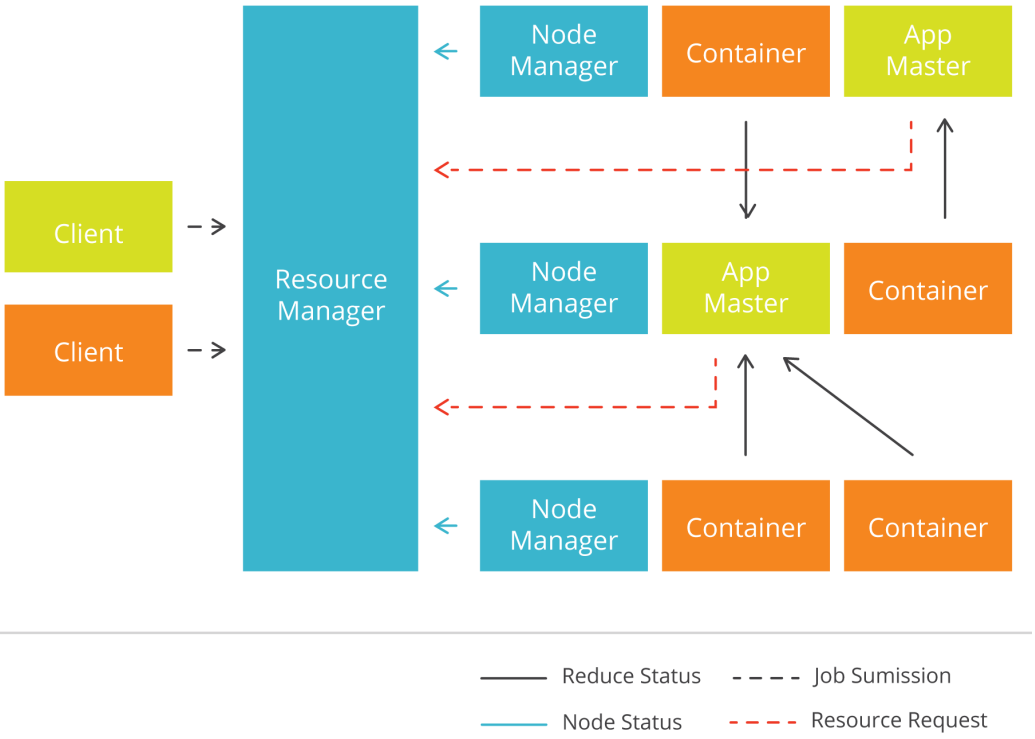
We illustrate Yarn by setting up a Hadoop cluster as Yarn by itself is not much to see. It is not something you work with from the command line except under special circumstances. But getting it to work across a distributed cluster is an exercise meant to illustrate what it does. When you set up Hadoop on a single machine you are not using Yarn as it was meant to be used. This is because Yarn helps colocate services rather than partition them. That means it reduces the number of servers an organization needs by fully utilizing each one instead of setting aside a single server or servers to run a specific task.

Apache Yarn (Yet Another Resource Negotiator) is the result of the rewrite of Hadoop by Yahoo to separate resource management from job scheduling. Not only does this improve Hadoop, it means Yarn is a standalone component that you can use with other software, like Apache Spark, or you can write your own application using Yarn, thus making your application run across a distributed architecture.

Yarn does basically the same thing as Mesos. What it does is keep track of available resources (memory, CPU, storage) across the cluster (meaning the machines where Hadoop is running). Then each application (e.g., Hadoop) asks the resource manager what resources are available and doles those out accordingly. It runs two daemons to do this: Scheduler and ApplicationsManager.

Yarn uses Linux cgroups to control tasks and keep them within their allocation resource. Linux cgroups are similar to Docker and Mesos containers. The goal is to keep the task with its allocated memory, CPU, network, and storage usage, although with Hadoop 2.7 this is limited to limiting CPU.

Yarn is a framework, meaning an API. In terms of the graphic below, we see the Node Manager querying the Resource Manager to execute and then monitor the MapReduce tasks.



To review Hadoop, it is a distributed file system (HDFS) meaning you can use it to copy files there across a distributed architecture. But it also runs MapReduce jobs for analytics.That means it takes an input file and maps it (i.e., run an operation across every data element in the set) and then reduce, which collapses all of that into one single value or (key->value) pairs grouped by key or tuples (a,b,c). So it needs something like Yarn to coordinate all of that.

Yarn-site.xml

In Hadoop, the Yarn config file is located in

`$HADOOP_HOME/etc/hadoop/yarn-site.xml.`

So are the Hadoop config files.
The default Yarn configuration has only this one property which is all you need for a basic test.

```
  yarn.nodemanager.aux-services
    mapreduce_shuffle
```

Whatever you put here overrides the Yarn defaults.

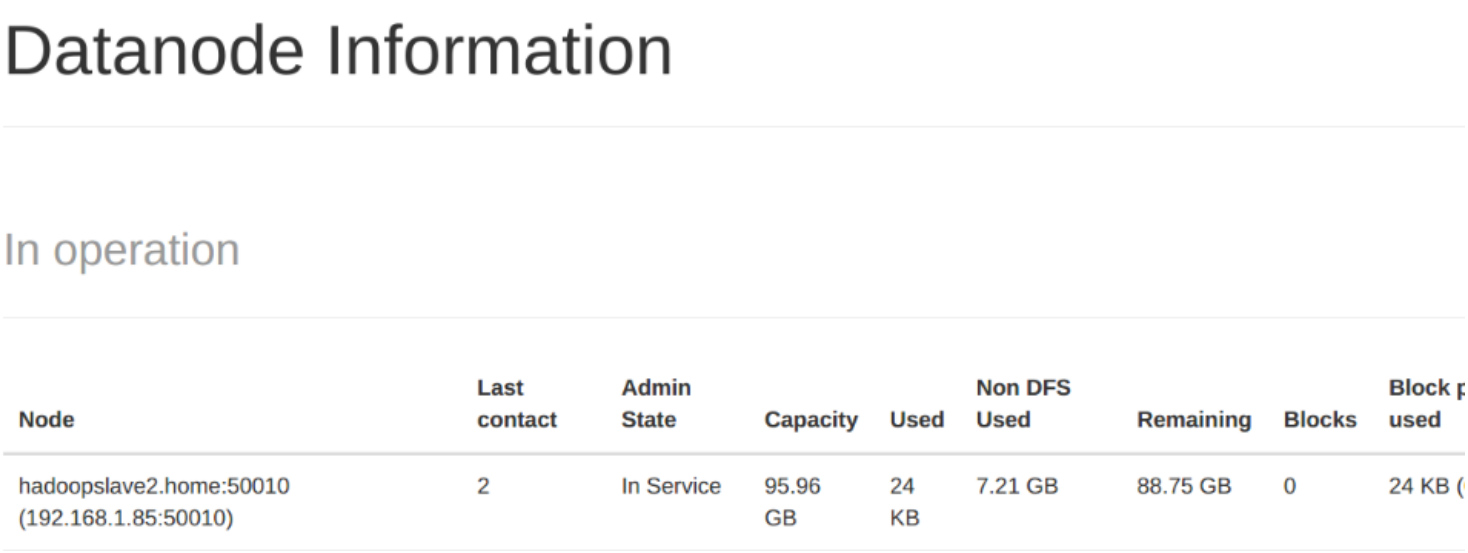
Using Yarn in a Hadoop cluster

If you are studying Apache Yarn and Hadoop then you should set up a Hadoop cluster. Most Hadoop students set up Hadoop on one machine and stop there. But then they do not get to see how Hadoop runs in a cluster and how to work with that. Behind the scenes Yarn is coordinating the MapReduce operations, but there is no visible evidence of that unless you go digging in the logs. You will more likely be digging in the Hadoop logs as configuring core-site.xml and hdfs-site.xml are the complicated parts and the logs is where you look for and correct config errors. As far as Yarn goes there are no configuration changes needed at all.

The steps to configuring Hadoop to run in a cluster are:

1. Spin up at least 2 virtual machines. Three would be better.
2. Create the Hadoop user and do the following Hadoop installation steps and run the Hadoop commands as that user.
3. Put hostnames of all 3 machines and their IPs in /etc/hosts
4. Create a folder on the master to store Hadoop data. The name you put here must correspond with step 7.
5. Install Hadoop on the machine you designate as the master.
6. Install Java on all 3 VMs.
7. Look at Hadoop's instructions for configuring core-site.xml and hdfs.xml. The instructions vary slightly by Hadoop version.
8. Edit $HADOOP_HOME/etc/hadoop/slaves and add the hostnames of the slaves.
9. Add JAVA_HOME to hadoop-env.sh.
10. Do not install Hadoop on the slaves. Rather copy the entire Hadoop installation (i.e., the $HADOOP_HOME directory) to the slaves.
11. Set the .bashrc config file or the Hadoop user as shown below.
12. Run start_dfs.sh to start Hadoop and start_yarn.sh. You do not need to restart Hadoop as you work through installation problems with Hadoop and restart that. (It will probably take a few times to get it correct.)
13. Run jps on the slaves and make sure the dataNode process is running.
14. Run jps on the master and make sure the nameNode process is running.
15. Make sure you can open this URL on the master http://localhost:50070/dfshealth.html#tab-overview
16. Format the Hadoop file system using: hadoop namenode -format.
17. Install Apache Pig and make sure you can run the example shown below.

If everything is working step 15 should look should display this webpage:



Yarn command line

You can run Yarn from the command line to get info. But you do not really need it.

The file $YARN_HOME/etc/hadoop/yarn-env.sh is where you override Yarn options for Yarn command line options.

Add the classpath to your environment so you can use Yarn from the command line.

```
export CLASSPATH=`yarn classpath`:$CLASSPATH
```

The command Yarn classpath returns all the jar files needed to run yarn

.bashrc
Here is what .bashrc should look like. Change your Java and Hadoop directories to match the versions you are using:

```
export JAVA_HOME=/usr/local/jdk-8u121/jdk1.8.0_121
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/usr/local/hadoop/hadoop-2.7.3/
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_CLASSPATH=$HADOOP_HOME/etc/hadoop
```

Run Apache Pig example with Yarn

Here we run a Hadoop job across the cluster by using Apache Pig. Pig is a lot easier to use than writing lengthy Java code. It's what Netflix does.

Download and install Apache Pig on the master. It is an easy install as you just download and unzip it and source the environment. Run these commands as the Hadoop user.

Create this simple text file data.csv. This shows some people and their ages. We will make a directory and copy it to Hadoop:

```
walker,23
steen,45
antonella,35
fernando,24
```

Make the Hadoop directory:

```
hadoop fs -mkdir /data
```

Copy the file data.csv there:

```
hadoop fs -put /home/walker/Downloads/data.csv /data/
```

Now run Pig in mapReduce mode, meaning using the distributed cluster:

```
pig -x mapreduce
```

It will open the grunt console:

Then type these two commands into the grunt shell:

```
ages = LOAD '/data/data.csv' USING PigStorage(',') as
(NAME:chararray,age:int);
dump ages
```

It should display:

```
(walker,23)
(steen,45)
 (antonella,35)
 (fernando,24)
```

Using Yarn with other applications

You can use Yarn with any application that you write to create a multithreaded distributed program. All you need to do is to use the YarnClient object in your program. Then implement the ApplicationMaster. You tell yarn what JAR files and other files the app needs and what command to run. Then it runs your application on a container.

Doing this is quite complex. You could use Apache Twill whose goal is to reduce this complexity. At the time this document was written Apache will was not working.

Using Apache Spark with Yarn

You can also run Apache Spark with Yarn. You need a distribution of Spark that is built with Yarn support. When you get it working you should be able to open the Scala shell using:

```
spark-shell --master yarn
```