5 Useful Tips to Make Your Hadoop Deployment Robust

Hadoop will help you prevent fraud, enable you to understand your customers better, and drive product innovation. These are just a few of its many benefits. Hadoop is an incredibly flexible and easy-to-use platform for storing and running your big data operations. That said, adopting Hadoop as a framework isn't a cakewalk.
When working with Hadoop for BI or other production purposes, the most important component is obviously the data layer. The following sections contain simple tips that you can implement to better manage and protect your data:

HDFS Snapshots

One of the most important aspects of any data-based system is its backup. As many know, it's not a question of if you'll need it, but rather a question of when you'll need it.
There are two types of scenarios in which one would use backup. The first is hardware failure, disk malfunctions, corrupted file systems, and events of that nature. For these cases, Hadoop's complete replication gives us excellent protection if implemented correctly.
The second scenario is human or application error, such as accidentally deleting a file and code, which corrupts the data. In this scenario, Hadoop's replication won't be of help since it will replicate the corrupted data. In addition, Hadoop does provide "Trash", similar to the one that we have in Windows, which protects us from accidentally deleting files. However, it has its faults:

- It only saves deleted files for 24 hours
- It does not protect us from bugs which corrupt our data
- It could be inactive if someone chooses the "Skip Trash" reference

In order to protect your data, Hadoop supports a mechanism called HDFS Snapshots, which works very similarly to Snapshot in terms of its common enterprise grade storage.
When we "snapshot" a folder or file, Hadoop creates another link to the file, very similar to the "ln" command in Unix. So, even if we delete or change the original file, we still have a copy.

for example, say we have a ***/user/bi/reports/2016*** folder in our HDFS, which contains all of our BI reports for 2016, with this listing:
/user/bi/reports/2016/00001.part
/user/bi/reports/2016/00002.part
/user/bi/reports/2016/00003.part

We will begin by running a command with HDFS admin which permits it to create snapshots of this folder:

```
hdfs dfsadmin -allowSnapshot /user/bi/reports/2016
```

Now, let's say that we want to do a daily snapshot of this folder, so we will run the snapshot:

```
hdfs dfs -createSnapshot /user/bi/reports/2016
```

Hadoop will now create a sub-directory which contains a link to all of the original files:

/user/bi/reports/2016/.snapshot/s20160507-100551.31

This folder will contain a backup of the original files:
/user/bi/reports/2016/.snapshot/s20160507-100551.31/00001.part
/user/bi/reports/2016/.snapshot/s20160507-100551.31/00002.part
/user/bi/reports/2016/.snapshot

Hadoop will not allow us to delete the folder **/user/bi/reports/2016/.snapshot/s20160507-100551.31/** or any of its content. To restore the files, we simple need to copy them back to our original folder.

Offsite Backup to AWS S3

The next level in protecting our data is implementing a remote backup, which can support your disaster recovery processes. Even though it's less likely, there are still cases in which a disaster can strike our data center, such as massive hardware failures and hackers deleting our files. For that purpose, the public cloud can serve as a perfect secondary backup site.
A quick and simple solution is to back up all of our data offsite on S3, which is a dynamic storage solution offered by Amazon cloud.
Hadoop has a useful feature called distcp, which will allow you to copy data across data centers. Since Hadoop is designed to easily work with S3, one simple command can automatically copy all of our important data from Hadoop clusters to S3.
Obviously, this solution requires a high upload bandwidth from our data centers to AWS. In some cases, using a smart incremental strategy with a moderate bandwidth can also give us a good solution.

Once you've set up your AWS account and created a bucket on S3, all you have to do is run the following command from your Hadoop cluster:

```
hadoop distcp -update -delete -m 3 {hadoop-folder-to-copy} s3a://{AWS-Key}:{AWS-Key-Secret}@{aws-bucket}/remote-
folder


#Here is an example:
hadoop distcp -update -delete -m 3 /user/bi/reports/2016 s3a://AKIAJAEEFFESAW2KJA:
S3HcWsNf4FADEZxSGEtTByFESFSsuVe7AF@company-bi-reports-2016/bi/reports/2016
```

Parameters:

- **update** – Overwrite the destination if its size is different from the source
- **delete** – Delete files that are no longer in the source folder to ensure we have an identical copy of our current source folder
- **m** – Define how many mappers you want to use. If you are copying a lot of data, you may want to restrict the number of mappers used so that the process won't hog all the resources on your Hadoop cluster

this process does require customized adjustments when transferring large amounts of data. You may need to run the command several times before you get the first mirror.
Usually the following parameters need more tweaking: ***fs.s3.maxRetries, fs.s3.sleepTimeSeconds, fs.s3n.multipart.uploads.enabled, fs.s3a.buffer.dir, fs.s3a.attempts.maximum.***

Learn more about these parameters

Archive Files

Anyone who has worked with HDFS long enough knows there are many issues when handling multiple small files. Hadoop can only support a limited number and there are several strategies for handling them. The simplest and quickest method is to use the HAR files. HAR is a built-in archive format that can contain large amounts of files with minimal overhead to the NameNode.
The benefit of an archive file over a zip file is that it's built inside Hadoop and you can directly access each file individually. However, unlike zip files, archive files do not compress the data.

Here is the command I use to create an archive file:

```
hadoop archive -archiveName {archive-name}.har -p {relative-folder} {folder-list} {archive-folder-placement}
```

For example, let's say we have an HDFS folder "/user/archive/bi/2015/csv" with 100,000 csv files. All we have to do in order to create the archive file is run the following command:

```
hadoop archive -archiveName 2015_csv.har -p /user/bi/reports/2015 csv /user/archive/reports
```

This command will create a new file /user/archive/reports/2015_csv.har that contains the same files that reside in /user/archive/bi/2015/csv.
Hadoop will create the archive file as a new copy of the data, but won't delete the old folder. In result, a part of the archiving routine should be to delete the old files once we've archived them.

In order to access the contents of the files inside of the archive, we need to use a special schema "har://" for Hadoop, so for example, to list all the files we will run:

```
hadoop fs -ls har:////user/archive/reports/2015_csv.har
```

In order to view the contents of one of the files, we can use the regular command:

```
hadoop fs -text har:////user/archive/reports/2015_csv.har/sales.csv
```

Similar to an archive folder, the contents of an archive file can be used as input for MapReduce jobs, or directly accessed. Archive files cannot be modified, therefore once it's created, no additional files can be added or deleted.

Replication Strategies

The main purpose of the HDFS replication strategy is to make sure that we don't loose our data in the case of our DataNodes malfunctioning.
There are some scenarios in which we can apply different replication strategies. For example, let's say we are working on a development cluster and a replication of 2 is good enough. Or, perhaps some of our data is just a clone of other systems and we can always clone it again. Why waste storage space?

On the other hand, there are cases that we want a higher data locality for highly used data, so raising the replication to 4 or even 5 to multiple folders may give us better performance.
With Hadoop, it's quite simple to change a folder replication setting.

All you have to do is run:

```
hadoop dfs -setrep -R -w {new-replication-factor} {folder}


#For example, let's change the replication factor of the folder '/user/bi/reports/management' to 4:
hadoop dfs -setrep -R -w 4 /user/bi/reports/management

#Hadoop will automatically change the replication.
```

Working with Different Clusters

Hadoop loads our cluster configurations from its configurations folder (e.g. /etc/hadoop). However, what if we wanted to work with several different clusters from the same machine?
One of the easiest ways to accomplish this is to command Hadoop to work with different configuration folders by setting the HADOOP_CONF_DIR environment variable.
For example, we can create several folders with configurations for different Hadoop clusters:
~/hadoop/production_east_cluster
~/hadoop/development_cluster
~/hadoop/production_west_cluster

Now every time we want to work with a specific cluster, we just set the HADOOP_CONF_DIR to point to the directory containing that cluster's configuration files:

```
export HADOOP_CONFI_DIR=~/hadoop/production_west_cluster
```

From now on, whenever we use the 'hadoop' command for our client, it will use the configuration files residing under the folder "~/hadoop /production_west_cluster". We can also shorten the procedure by using our own custom alias or script to make the transition easier.

Bonus Tip

To easily find the largest consumer of disk space, I use the following command:

```
hadoop fs -du -h /user | grep T
```

This will only display users that occupy more than 1TB of data. Of course, you can play with the parent folder that you listed and look within other sub folders.