HDFS PRACTICAL

MANAGING BIG DATA (MBD)
2021

DOINA BUCUR

---

1. HDFS OPERATIONS

There are **two file systems** on a computing cluster for big-data. The Network File System (NFS)[1] is a traditional file system for small files. The NFS is often referred to as the "local" file system. The Hadoop File System (HDFS, see Lecture 1) is a modern one for big files. They reside on different disks, but are available to you at the same command line.

Figure out how to do the following data-moving tasks:

(1) Download[2] a data file from the Internet[3] and place it into your NFS `/home/snumber` folder. The example file in the footnote is good, because it is still small, so will download fast. Check that the file is there. Rename the file into something shorter if you like.

(2) Create a new folder on your own HDFS `/user/snumber` folder. Call it, say, "test". Check that the folder is there.

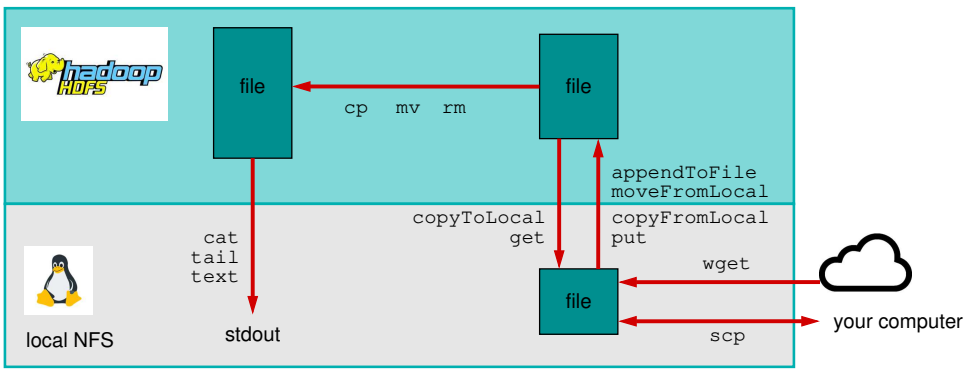
(3) Copy the new data file from point (1) to your new HDFS `/user/snumber/test` folder from point (2). Check that the file is there. You have now successfully completed a download process.

(4) Delete the data file from your NFS home. Check that it's gone. You still have it on the HDFS. All your big (and almost big) datasets should reside on the HDFS, and should never be left on the NFS, because there's not much space there.

(5) Try to read in plain text (on the `stdout`, meaning standard output) the first few lines of the file from the HDFS, using only the command line.

If you don't know how to start: step (1) is just a matter of remembering some basics about the Unix command-line interface, or just following the hints in the footnotes here. For the other points, first go through the demo in Lecture 1, on your CTIT worker machine. The commands in the demo are similar. You don't need to remember these commands: just keep the reference page for the HDFS shell open[4], until you get the hang of it.

Below is a summary of how to move files between NFS and HDFS. The cloud on the right means the Internet.



[1] NFS on Wikipedia

[2] `wget` is installed, so you can do `wget URL`.

[3] This URL from the archive.org domain contains one minute of tweets from early 2021. Get the URL from this pdf with, probably, a right mouse click.

[4] Apache Hadoop 3.0.0 FileSystem Shell

For example, to copy a file from HDFS back to NFS (which you should do only if it's small data!), you would use `get` in this way:

```
hdfs dfs -get /user/snumber/test/file
```

Note on how to write paths on these filesystems: a complete HDFS path has the form

```
hdfs://master-node/path/to/file/or/folder
```

(the master node for HDFS is `ctit048`), while a complete NFS path has the form

```
file:///path/to/file/or/folder
```

You will sometimes need to write the path in full, to specify which filesystem you mean. This will look like:

```
hdfs dfs -ls -h hdfs://ctit048.ewi.utwente.nl/data   # which is the same as:
hdfs dfs -ls -h /data
```

2. Understand which read/write operations are efficient on a DFS

Take these (approximate) latency numbers[5]:

DRAM reference:	100 ns
Send 2 KB over a 1 Gbps network:	20,000 ns
Read from/write to DRAM, 1 MB sequentially:	250,000 ns
Round trip within data center:	500,000 ns
Disk seek:	10,000,000 ns
Read from/write to disk 1 MB sequentially:	20,000,000 ns
Send 2 KB across the Atlantic and back:	150,000,000 ns

Solve this puzzle:

You have a file of 1 TB in size, which contains very many 100-byte records. It is stored on one large disk. You need to read 1% of the records. Which way is faster, (1) or (2) below? You may be asked on Discord by the teacher what numbers you reached exactly.

(1) Seek to each of the records that are needed; read.[6]

(2) Read the entire data set sequentially, and from this, keep only the records you need.

Once you computed the numbers and obtained an answer, you will understand this general, practical rule: *avoid random access on big data*! Essentially, when you deal with big data, the new data "atom" is not the byte or the word (the low-level atom we're used to), but the chunk or block of data.

When *reading* data on a DFS in practice, reads come in two types:

- large contiguous reads (1 MB+), and
- small random reads (1 KB).

Random reads should be avoided: performance-conscious applications batch and sort their small reads to advance linearly through the file. Random writes are practically non-existent: they are supported, but are inefficient and will slow down the system a lot.

When *writing* data on a DFS (the reading material R1 for this week insists on this):

- Once written, the files are mostly read, and mostly sequentially.
- Most files are mutated by appending new data rather than overwriting data.

[5] From Google: Designs, Lessons and Advice from Building Large Distributed Systems

[6] Note: You will find that the reading takes negligible time compared to the seeking, so you can ignore it in the calculation, to get a nice round number.