TP2: Utilizing and Configuring *Hadoop*: HDFS

Shadi Ibrahim
(February 9th, 2017)

MapReduce has emerged as a leading programming model for data-intensive computing. It was originally proposed by Google to simplify development of web search applications on a large number of machines.

Hadoop is a java open source implementation of MapReduce sponsored by Yahoo! The Hadoop project is a collection of various subprojects for reliable, scalable distributed computing. The two fundamental subprojects are the Hadoop MapReduce framework and the HDFS. HDFS is a distributed file system that provides high throughput access to application data. It is inspired by the GFS. HDFS has master/slave architecture. The master server, called NameNode, splits files into blocks and distributes them across the cluster with replications for fault tolerance. It holds all metadata information about stored files. The HDFS slaves, the actual store of the data blocks called DataNodes, serve read/write requests from clients and propagate replication tasks as directed by the NameNode.

The Hadoop MapReduce is a software framework for distributed processing of large data sets on compute clusters. It runs on the top of the HDFS. Thus data processing is collocated with data storage. It also has master/slave architecture. The master, called Job Tracker (JT), is responsible of : (a) Querying the NameNode for the block locations, (b) considering the information retrieved by the NameNode, JT schedule the tasks on the slaves, called Task Trackers (TT), and (c) monitoring the success and failures of the tasks.

The goal of this TP is to study how to configure the platform to expose different important features of HDFS (e.g., chunk size). We suppose you have an instance of Hadoop running on your local machine and on Grid'5000, as seen in the previous TP.

Exercise 1: Configuring your Hadoop platform

Question 1.1

The HDFS **chunk (block)** size is the smallest unit of data that the file system can store. This value has an influence on the MapReduce jobs execution. HDFS is meant to handle large files. If you have small files, smaller block sizes are better. If you have large files, larger block sizes are better.
Let's focus on the latter case. A large chunk size offers several important advantages. First, it reduces clients' need to interact with the master because reads and writes on the same chunk require only one initial request to the master for chunk location information. The reduction is especially significant for our workloads because applications mostly read and write large files sequentially. Second, since on a large chunk, a client is more likely to perform many operations on a given chunk, it can reduce network overhead by keeping a persistent TCP connection to the chunkserver over an extended period of time. Third, it reduces the size of the metadata stored on the master. This allows to keep the metadata in memory, which in turn brings several advantages. Finally, the chunk size directly impacts on the number of map tasks, as the input data is split into several blocks (of chunk size), each assigned to a map task.

Clearly, the chunk size is a crucial factor for Hadoop performance. Currently, the default block size in Hadoop is set to **128 MB**. To change the block size for **new files**, you can update the `conf/hdfs-site.xml` file:

```
<configuration>
  <property>
    <name>dfs.block.size</name>
      <value>67108864</value> (This value is equal to 64MB)
    </property>
</configuration>
```

From Command line: you can change the Block size of the file when uploading it as:
`bin/hadoop fs -Ddfs.block.size=xx -put source destination`

In order to have a full report on the usage of your HDFS in the command line:
`bin/hadoop dfsadmin -report`

Question 1.2

HDFS supports the fsck command to check for various inconsistencies. It it is designed for reporting problems with various files, for example, missing blocks for a file or under-replicated blocks. Unlike a traditional fsck utility for native file systems, this command does not correct the errors it detects. Normally NameNode automatically corrects most of the recoverable failures. By default fsck ignores open files but provides an option to select all files during reporting. Please check its usage: *https : //hadoop.apache.org/docs/r*1.0.4*/commands_manual.html#fsck*

```
bin/hadoop fsck FILEPATH -files -blocks -locations
```

Question 1.3

HDFS data blocks might not always be placed uniformly across data nodes, meaning that the used space for one or more data nodes can be underutilized. Therefore, HDFS supports rebalancing data blocks using various models. One model might move data blocks from one data node to another automatically if the free space on a data node falls too low. Another model might dynamically create additional replicas and rebalance other data blocks in a cluster if a sudden increase in demand for a given file occurs. HDFS also provides the hadoop balance command for manual rebalancing tasks. One common reason to rebalance is the addition of new data nodes to a cluster. When placing new blocks, name nodes consider various parameters before choosing the data nodes to receive them. Some of the considerations are:

- Block-replica writing policies
- Prevention of data loss due to installation or rack failure
- Reduction of cross-installation network I/O
- Uniform data spread across data nodes in a cluster

Usage

```
bin/hadoop balancer [-threshold <threshold>]
threshold: Percentage of disk capacity
```

Question 1.4

During start up the NameNode loads the file system state from the fsimage and the edits log file. It then waits for DataNodes to report their blocks so that it does not prematurely start replicating the blocks though enough replicas already exist in the cluster. During this time NameNode stays in Safemode. Safemode for the NameNode is essentially a read-only mode for the HDFS cluster, where it does not allow any modifications

to file system or blocks. Normally the NameNode leaves Safemode automatically after the DataNodes have reported that most file system blocks are available.

```
bin/hadoop dfsadmin -safemode get/enter/leave
```