VE472 — Methods and tools for big data

Homework 2

Manuel — UM-JI (Summer 2021)

Reminders

- Write in a neat and legible handwriting or use $\LaTeX$
- Clearly explain the reasoning process
- Write in a complete style (subject, verb, and object)
- Be critical on your results

As discussed in the lectures HDFS was designed to deal with large files. Unfortunately big data does not always come as large files. In this homework we will study various approaches to mitigate this problem.

Ex. 1 — *Preparation*

1. Adjust your program from lab 2 exercise 2 to generate many small `csv` files.
2. Find the block size on your Hadoop installation, and explain how you did it.
   Hint: the cluster block size is defined in the variable `fs.block.size`.

Ex. 2 — *Filecrush*

Filecrush perfectly illustrates the complexity of the hadoop ecosystem. The project (`https://github.com/edwardcapriolo/filecrush`) has not been touched since 2014. When searching online some sources assume it works with recent Hadoop versions while others warn about the utility being only compatible with Hadoop 1.

1. Read the documentation and briefly explain how we would need to use it in our case.
2. Try to run it on Hadoop with the following options. Explain each the used options.
   - `-Dfs.block.size=128000000`
   - `--clone`
   - `--compress gzip`
   - `--input-format text`
   - `--output-format sequence`
3. Depending on the success or failure, explain what happened.

Ex. 3 — *S3DistCp*

An alternative and more recent solution consists in using S3DistCp developed by Amazon.

1. Read the documentation at `https://docs.aws.amazon.com/emr/latest/ReleaseGuide/UsingEMR_s3distcp.html` and explain how the `--groupBy` option can be used to solve our problem. What other options do we need to add?
2. Install S3DistCp and aggregate all the small files into a larger one.

Ex. 4 — *Avro*

While the two previous solutions work well for text files it may happen that the data to process is composed of many small binary files. In that case one of the simplest solution consists in packaging them into a single Avro file.

1. Tools.
   a) Read about Avro at `https://avro.apache.org/` and install it.

b) What is the Snappy codec and when is it best used?

2. Define a json schema composed of three fields: `filename`, a string representing the name of the file; `filecontent` of type bytes, which stores the actual binary content of the file; and `checksum`, a string containing the SHA-1 checksum of the file.

3. Write a Java class `CompactSmallFiles`, to read a directory containing small files and compacting them inside a single Avro file. Set the Avro writer to use the Snappy codec.
   Hint: use the `writer.setCodec` method to enable Snappy compression.

4. Write a Java class `ExtractSmallFiles`, to read an Avro file generated with `CompactSmallFiles` and extract the small files it contains. Upon extracting a small file, the checksum must be verified and an error displayed if it does not match the original one.
   Hint: as Avro embeds the schema into its files there is not need to redefine it.

Various other approaches could have been used to overcome the problem of small files in HDFS. Most simple ones such as zipping files together suffer from a lack of flexibility. For instance it would require to write a custom input format, and it would not be possible to split the zip file into pieces.

Besides when storing files it is important to use a format which can take advantage of data locality to balance the load over the cluster. For example in MapReduce, if the mapper input is a text file containing information on an archive of the data stored on some other nodes, then the work would be run of the mapper's node, i.e. far from the actual data.

Another simple solution which works well but suffers the limitation of not providing compression is the use of a Hadoop Archive file (HAR).

Alternative distributed filesystems such as MapR natively support both large and small files. Therefore before setting up a Hadoop cluster it is important to know what will be its main usage and decide what filesystem to use accordingly.