VE472HW2

吴佳遥 517370910257

1. Preparation

1.1

See `src/ex4/input/csv.py`

try with

```
python3 csv.py [pagination] [students number] [number of generated grades]
```

Will generate a foler named `results` in the same directory as `csv.py` which contains all the results.

1.2

Use command

```
hdfs getconf -confKey dfs.blocksize
```

```
# hadoop @ hadoop-slave2 in ~ [19:21:44]
$ hdfs getconf -confKey dfs.blocksize
134217728
```

The block size is 134217728 bytes, namely 128MB.

2. Filecrush

2.1

Crush consumes directories containing many small files with the same key and value types and creates fewer, larger files containing the same data.

We could use Filecrush to crush such small datasets together into an overall big dataset. And then use the big dataset as the input.

2.2

```
hadoop jar filecrush/target/filecrush-2.2.2-SNAPSHOT.jar
com.m6d.filecrush.crush.Crush -Dfs.block.size=128000000 --input-format text --clone
--output-format sequence --compress gzip /src/results src/outputs/ 20210602115700
```

```
# hadoop @ hadoop-master in ~/Desktop/VE472HW2 [13:03:46] C:1
$ hadoop jar filecrush/target/filecrush-2.2.2-SNAPSHOT.jar com.m6d.filecrush.crush.Crush -Dfs.block.size=128000000 --input-format text --clon
e --output-format sequence --compress gzip /src/results/ /src/outputs/ 20210602115700
2021-06-02 13:04:28,519 INFO Configuration.deprecation: mapred.output.compress is deprecated. Instead, use mapreduce.output.fileoutputformat.
compress
2021-06-02 13:04:28,520 INFO Configuration.deprecation: mapred.output.compression.type is deprecated. Instead, use mapreduce.output.fileoutpu
tformat.compress.type
2021-06-02 13:04:28,520 INFO Configuration.deprecation: mapred.output.compression.codec is deprecated. Instead, use mapreduce.output.fileoutp
utformat.compress.codec
2021-06-02 13:04:28,874 INFO zlib.ZlibFactory: Successfully loaded & initialized native-zlib library
2021-06-02 13:04:28,875 INFO compress.CodecPool: Got brand-new compressor [.deflate]
2021-06-02 13:04:28,880 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
Exception in thread "main" java.io.FileNotFoundException: File hdfs://hadoop-master:9000/user/hadoop/tmp/crush-c6485beb-1635-4a16-bc95-8c1c70
33d3e7/out does not exist.
        at org.apache.hadoop.hdfs.DistributedFileSystem.listStatusInternal(DistributedFileSystem.java:1061)
        at org.apache.hadoop.hdfs.DistributedFileSystem.access$1000(DistributedFileSystem.java:133)
        at org.apache.hadoop.hdfs.DistributedFileSystem$24.doCall(DistributedFileSystem.java:1132)
        at org.apache.hadoop.hdfs.DistributedFileSystem$24.doCall(DistributedFileSystem.java:1129)
        at org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSystemLinkResolver.java:81)
        at org.apache.hadoop.hdfs.DistributedFileSystem.listStatus(DistributedFileSystem.java:1139)
        at org.apache.hadoop.fs.FileSystem.listStatus(FileSystem.java:1902)
        at org.apache.hadoop.fs.FileSystem.listStatus(FileSystem.java:1944)
        at com.m6d.filecrush.crush.Crush.getOutputMappings(Crush.java:795)
        at com.m6d.filecrush.crush.Crush.cloneOutput(Crush.java:751)
        at com.m6d.filecrush.crush.Crush.run(Crush.java:666)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:90)
        at com.m6d.filecrush.crush.Crush.main(Crush.java:1330)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:323)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:236)
```

3. S3DistCp

3.1

--groupBy=PATTERN:

In case we have several small data files which share a same naming pattern, we could use --groupBY to concatenate all of our data files with this special naming pattern. Thus we can efficiently copy large amounts of data (first stored seperately in several small files) from Amazon S3 into HDFS and then compress all of them into a larger data file.

3.2

Skipped

4. Avro

4.1

b) Snappy codec is the implementation of Snappy compression and decompression.

4.2

See `src/ex4/ex4.json`

Output in `src/ex4/src/main/java/ex4/avro/AvroFile.java`

4.3

See `src/ex4/src/main/java/CompactSmallFiles.java`

4.4

See `src/ex4/src/main/java/ExtractSmallFiles.java`

Overall Test:

The whole process is already written in `src/ex4/src/main/java/Main.java`.

Run it. Output is in `src/ex4/output/newResults`

Run the script at `src/ex4/test.sh` to diff all the files between original csv files in `src/ex4/input/results` and processed new csv files in `src/ex4/output/newResults`

Note

All the generated files in Section 1 and Section is deleted.

You could go to `src` and exec `happyAll.sh` for the overall process