Big Data Technologies

Agenda

- MR execution with single reducer
- Combiner
- MR execution with multiple reducer
- Partitioner
- MR execution flow
- Hadoop Performance
- MR execution on YARN
- Map-only job
- Reduce-only job
- Hadoop streaming

MR execution with single reducer

- Refer slides

Combiner

- Refer slides
- terminal> hadoop jar mr4-jobtotalsal-0.0.1-SNAPSHOT.jar /user/nilesh/deptemp/input /user/nilesh/deptemp/output4
- terminal> hadoop fs -head /user/nilesh/deptemp/output4/part-r-00000

Multiple reducers

- By default, there is only one reducer per job.
- If the huge data to be aggregated by the reducer, it may fail with out of memory error. In such case, we should increase number of reducers.
- To change number of reducer, do one of the following.

- Java code:

```java
job.setNumReduceTasks(2);
```

- Configuration (xml):

```xml
<property>
    <name>mapreduce.job.reduces</name>
    <value>2</value>
</property>
```

- In few projects, number of mapper to reducer ratio is maintained as 10:1 or 20:1, depending on amount of data sent from mapper to reducer.
- If cluster node configuration is higher (e.g. 32/64 GB RAM, 8 cores), the amount of data that can be handled by one reducer is upto 10 GB.
- terminal> hadoop jar mr4-jobtotalsal-0.0.1-SNAPSHOT.jar /user/nilesh/deptemp/input /user/nilesh/deptemp/output5
- terminal> hadoop fs -head /user/nilesh/deptemp/output5/part-r-00000
- terminal> hadoop fs -head /user/nilesh/deptemp/output5/part-r-00001

Partitioner

- Refer slides
- terminal> hadoop jar mr4-jobtotalsal-0.0.1-SNAPSHOT.jar /user/nilesh/deptemp/input /user/nilesh/deptemp/output6
- terminal> hadoop fs -head /user/nilesh/deptemp/output6/part-r-00000
- terminal> hadoop fs -head /user/nilesh/deptemp/output6/part-r-00001

MR execution flow

- HDFS --> InputFormat & RecordReader --> Mapper --> Partitioner --> Sort/Group --> Combiner --> Spill on disk (Mapper) --> Network --> Spill on disk (Reducer) --> Merge/Group --> Reducer --> OutputFormat & RecordWriter --> HDFS.

Hadoop Performance

- Refer slides
- Spill:
  - Map-side 1, ~~2~~, Reduce-side 3
  - To avoid Map side spill: io.sort.mb, ...
  - To avoid Reduce side spill: input.buffer.percent, ...
- Combiner:
  - To reduce network traffic
  - To reduce reducer data load
- Number of reducers:
  - Mapper to Reducer ratio
  - Max data handled by reducer
  - Set Partitioner
- Compression:
  - Compress Mapper output to reduce network traffic

MR execution on YARN

- ResourceManager
- NodeManager
- MRAppMaster
- YarnChild -- Map task/Reduce task
- Refer slides

Map-only job

- A MR job can have zero reducers and then the job is referred as Map-only job.

```
job.setNumReduceTasks(0);
```

- Map only job does processing on each record; But doesn't perform any aggregation operations.
- Applications
  - Data cleansing
  - Data filtering
  - Data pre-processing (e.g. handle null values, ...)
  - Data ingestion/transfer
- Sqoop
  - Sqoop = SQL for Hadoop
  - Hadoop eco-system to transfer from RDBMS to Hadoop and vice-versa.
  - Sqoop commands e.g. "import", "export" internally produces map-only jobs to transfer data to/from RDBMS which internally uses DbInputFormat/DbOutputFormat.

Reduce-only job

- Technically reduce-only job is not possible. Hadoop is designed in such a way that reduce takes its input from mapper output only.
- But reduce only job can be emulated by using IdentityMapper (org.apache.hadoop.mapred.lib.IdentityMapper). IdentityMapper send input data as it is to the output.

```
job.setMapperClass(IdentityMapper.class);
```

Hadoop streaming

- Hadoop streaming enable us to implement Hadoop MR job in any programming language which has capability to input from terminal and output to terminal.
- This feature internally based on IO redirection.
- HDFS --> InputFormat --> Streaming Mapper <--> my-mapper.py
  - Predefined streaming mapper class of Hadoop
    - Launch a process to execute user defined mapper (e.g. python process to run my-mapper.py)
    - Input received from InputFormat is sent to user-defined mapper using Input redirection (on its stdin).
  - The user-defined mapper class should get data record by record from stdin, process it and send output to stdout.
  - Predefined streaming mapper class of Hadoop

        - Read output data of the user-defined mapper class using Output redirection (from its stdout).
        - Send it for further map-side processing i.e. Partitioner, Sorting, Combiner, Spill.
- Network --> Streaming Reducer <--> my-reducer.py
    - Execution is similar to mapper.
- Since IPC is involved (JVM and streaming process like Python), performace is always less than Java MR job.
- terminal> hadoop fs -mkdir -p /user/nilesh/words/input
- terminal> hadoop fs -put $HADOOP_HOME/LICENSE.txt /user/nilesh/words/input
- terminal> hadoop fs -ls /user/nilesh/words/input
- terminal> hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.2.jar -files mapper.py,reducer.py -input /user/nilesh/words/input -output /user/nilesh/words/output3 -mapper mapper.py -reducer reducer.py
- terminal> hadoop fs -head /user/nilesh/words/output3/part-00000

Assignment

- From statewise.csv file collect State Name, Confirmed, Recovered, Deceased and Active cases. Also remove bad records.