Recap

- Hadoop installation
- Running JAR file on cluster

Agenda for today

- HDFS API
- Performance tuning in MapReduce jobs
- Ad-hoc analysis with Impala
- Hive/Impala as Query processing tool
- MapReduce job chaining

Performance tuning

- Cluster configuration
- Use compression technique
- Tuning # mappers and reducers
- Use combiner
- Appropriate data type
- Reuse objects
- Profiling

https://blog.cloudera.com/blog/2009/12/7-tips-for-improving-mapreduce-performance/

Modify HDFS Block size

- CLI

```
hadoop fs -D dfs.blocksize=268435456 -copyFromLocal <source> <target>
```

- API

```
OutputStream out = fs.create(new Path(dst), overwrite, bufferSize,
replication, blockSize, new Progressable() {
public void progress() {
System.out.print(".");
}})
```

HDFS REST API

- Allows web access to HDFS
- https://hadoop.apache.org/docs/r1.0.4/webhdfs.html#Document+Conventions

MapReduce Job chaining

- Two separate jobs

- Multiple mappers/reducers within same job

MapReduce Job chaining

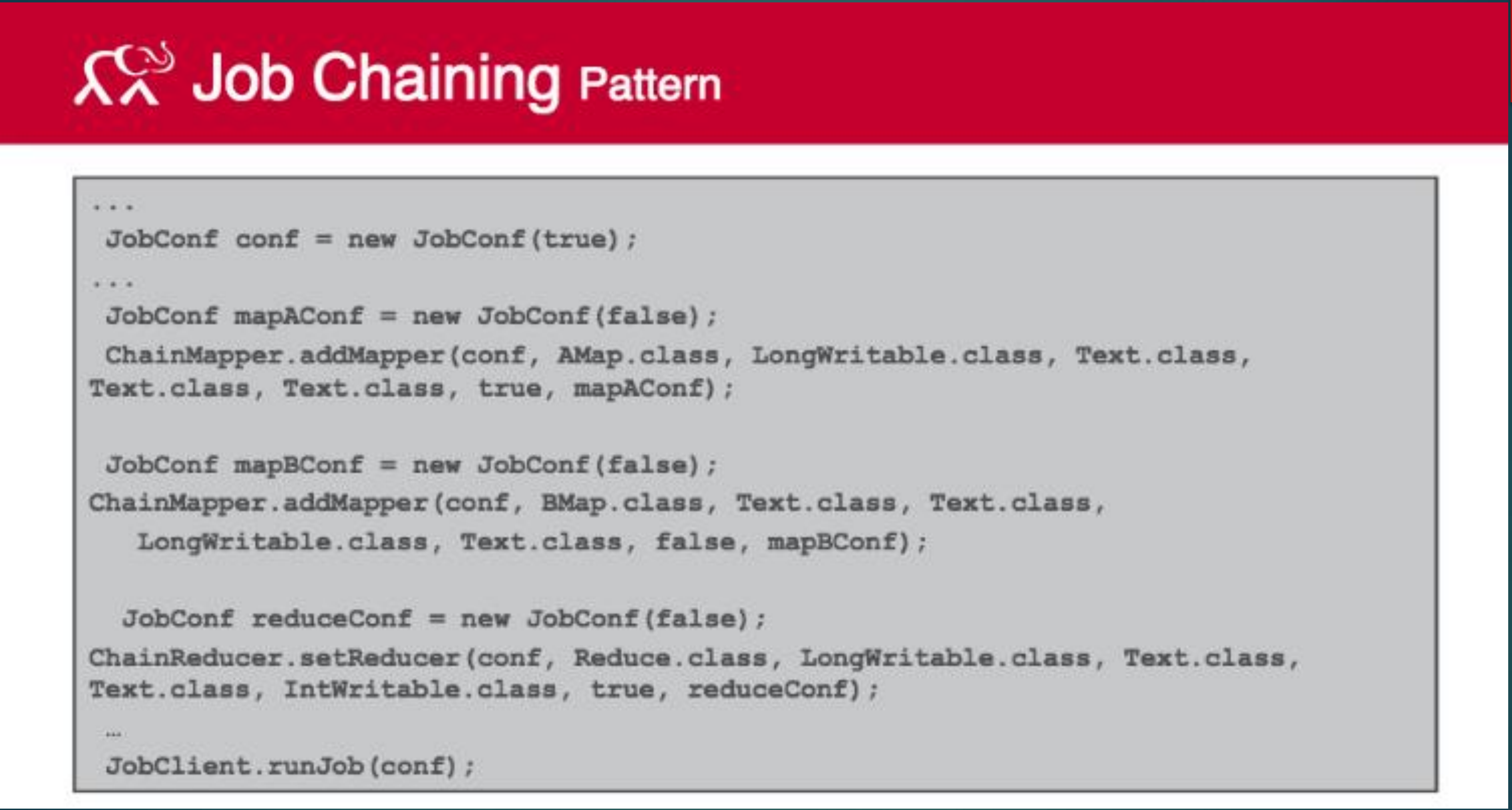
- Two separate jobs

1. Configure first job object and run it.
2. Configure second job object and run it

MapReduce Job chaining

- Multiple mappers/reducers within same job

https://mapr.com/blog/how-to-launching-mapreduce-jobs/



Job Chaining Pattern

```
...
 JobConf conf = new JobConf(true);
...
 JobConf mapAConf = new JobConf(false);
 ChainMapper.addMapper(conf, AMap.class, LongWritable.class, Text.class,
Text.class, Text.class, true, mapAConf);

 JobConf mapBConf = new JobConf(false);
ChainMapper.addMapper(conf, BMap.class, Text.class, Text.class,
   LongWritable.class, Text.class, false, mapBConf);

  JobConf reduceConf = new JobConf(false);
ChainReducer.setReducer(conf, Reduce.class, LongWritable.class, Text.class,
Text.class, IntWritable.class, true, reduceConf);
 …
 JobClient.runJob(conf);
```

HQL

- Create/Drop/Update table
- Insert/Update/Delete data into table
- Partitioning
- Modifying data directly in HDFS
- REFRESH table
- External vs Internal table
- Profiling and Optimization