TP4: Running multiple MapReduce applications in *Hadoop*

Shadi Ibrahim
(March 2nd, 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 run under a specific scheduler. We shall examine how to allocate resources under the fair scheduler and how to enable preemption technique.

Exercise 1: Configuring your scheduling policy

Question 1.1

Hadoop has recently been used to run multiple divers MapReduce applications belonging to multiple concurrent users, thanks to its built-in schedulers (i.e., Fifo, Fair and Capacity schedulers). The default scheduler in Hadoop is FiFo. To run under fair scheduling policy, in `conf/mapred-site.xml` add:

```
<configuration>
  <property>
    <name>mapred.jobtracker.taskScheduler</name>
      <value>org.apache.hadoop.mapred.FairScheduler</value>
    </property>
</configuration>
```

Similarly, if you want to run under Capacity scheduler change the value to *org.apache.hadoop.mapred.CapacityTaskScheduler*

Exercise 2: Configuring Fair scheduler

Question 2.1

Fair scheduler uses a two-level scheduling hierarchy. At the top level, cluster slots are allocated across pools using weighted fair sharing i.e. the higher the weight a pool is configured with, the more resources it can acquire. Each user, by default, is assigned one pool. At the second level and within each pool, slots are divided among jobs, using either Fifo with priorities (the same with Fifo scheduler) or a second level of fair sharing. In the second level of fair scheduling, each job is assigned a weight equal to the product of its (user-defined) priority and the number of tasks.

To set the number of pools (users and allocation policy which a pool), change the value in `conf/fair-scheduler.xml`:

```
<allocations>
   <pool name="root">
      <minMaps>0</minMaps>
      <minReduces>0</minReduces>
      <minSharePreemptionTimeout>5</minSharePreemptionTimeout>
      <schedulingMode>fair</schedulingMode>
      </pool>
   <pool name="sibrahim">
      <minMaps>0</minMaps>
      <minReduces>0</minReduces>
      <minSharePreemptionTimeout>5</minSharePreemptionTimeout>
      <schedulingMode>fifo</schedulingMode>
      </pool>
</allocations>
```

In this example, two users (pools) are configured. The first pool are running with Fair allocation within the pool and the second one is running with Fifo allocation within the pool.
Run two RandomWriter (an application which generates 10GB of textual data per node) under both Fifo and Fair (with fair allocation within the pool – you are using pool) and see the execution time and the waiting time of both applications.

Run 6 sort applications (vary the size from 3*2GB, 2*4GB 1*10GB) under both Fifo and Fair (with fair allocation within the pool – you are using pool) and see the execution time, data locality and the waiting time of both applications.

Question 2.1

Preemption has been widely studied and applied for many different use cases in the area of computing. Similarly, Hadoop can also benefit from the preemption technique in several cases (e.g., achieving fairness, better resource utilization or better energy efficiency). However, only kill technique is available in Hadoop which can be used by developers as a preemption technique. This preemption enabled in Hadoop's `mapred-site.xml`:

```
<configuration>
   <property>
      <name>mapred.fairscheduler.preemption</name>
         <value>true</value>
      </property>
</configuration>
```

Run two RandomWriter (an application which generates 10GB of textual data per node) under Fair (with fair allocation within the pool) and enable and disable preemption. See the execution time and the waiting time of both applications.

Run 6 sort applications (vary the size from 3*2GB, 2*4GB 1*10GB) under Fair (with fair allocation within the pool – you are using pool) and enable and disable preemption. See the execution time, data locality and the waiting time of both applications.