SparkLab - Spark Jobs Memory Tuning

Scenario: We are going to generate 100000 records using PySpark code and observes how Spark performs as we change the configuration parameters. We are doing this through an Amazon EMR cluster with 1 m5.xlarge master node and 2 m5.xlarge core nodes. Each m5.xlarge instance has 4 virtual cores (vCPUs) and 16 GB RAM. All these calculations are for the --deploy-mode cluster, which we recommend for production use.

Configuration parameters:

spark.executor.memory – Size of memory to use for each executor that runs the task

spark.executor.cores – Number of virtual cores

num-executors – Number of executors. Set this parameter unless spark.dynamicAllocation.enabled is set to true

1. Create DataGenerator.py file and insert below code inside it

from pyspark.sql.functions import rand, randn

from pyspark.sql import SparkSession

from pyspark import SparkConf, SparkContext, SQLContext

sc = SparkContext.getOrCreate()

sqlContext = SQLContext(sc)

#Create SparkSession

spark = SparkSession.builder.master("yarn").appName("SparkByExamples.com").getOrCreate()

df = sqlContext.range(0, 10000000)

df.count()

df2 = df.select("id", rand(seed=1000).alias("uniform"), randn(seed=555).alias("normal"))

row1 = df2.agg({"id": "max"}).collect()[0]

print row1["max(id)"]

df2.createOrReplaceTempView("df2")

df\_part1 = spark.sql("select \* from df2 where id between 1 and 999999 order by id desc")

row2 = df\_part1.agg({"id": "max"}).collect()[0]

print row2["max(id)"]

df\_part1.write.format("csv").mode("overwrite").save("/home/hadoop/data/output1/")

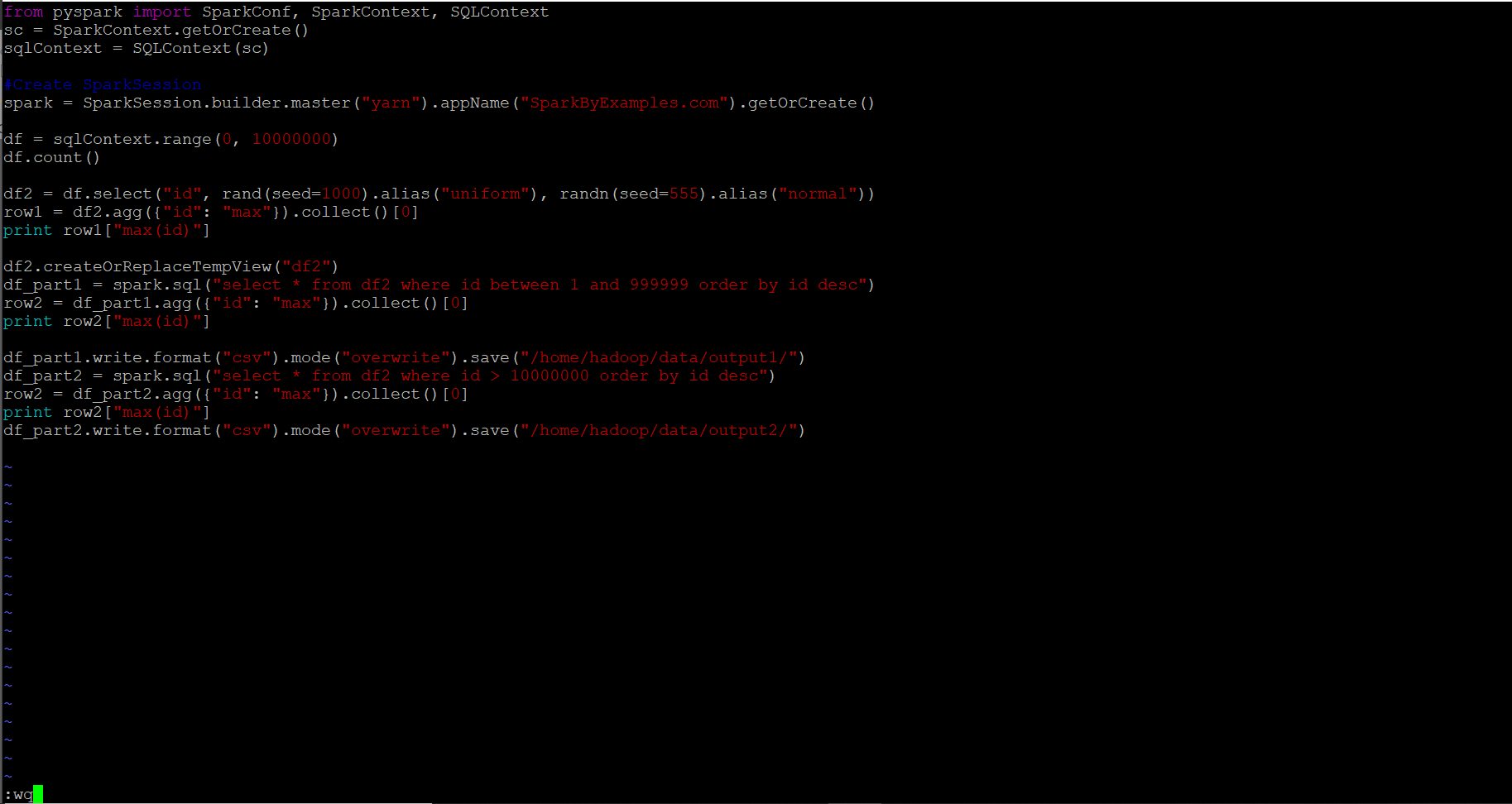
df\_part2 = spark.sql("select \* from df2 where id > 10000000 order by id desc")

row2 = df\_part2.agg({"id": "max"}).collect()[0]

print row2["max(id)"]

df\_part2.write.format("csv").mode("overwrite").save("/home/hadoop/data/output2/")



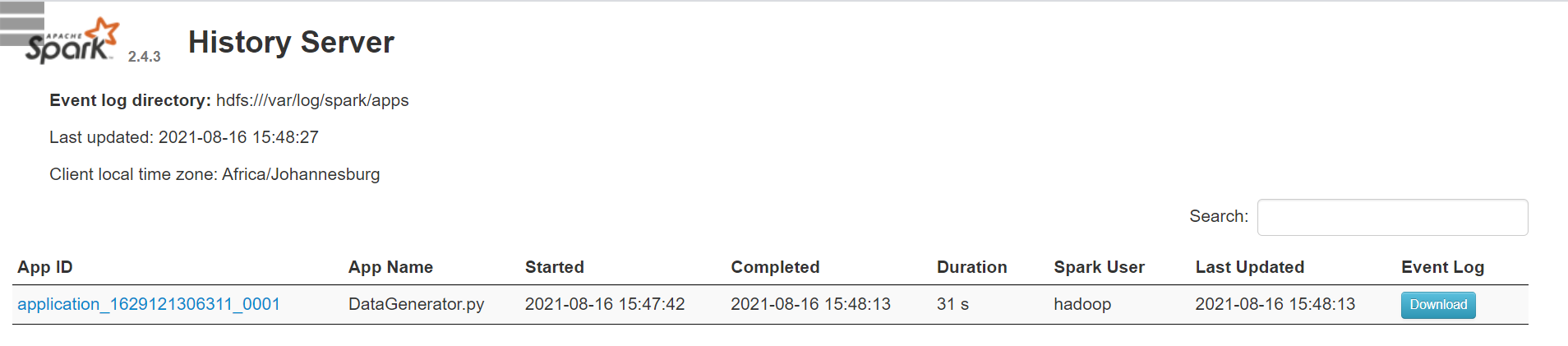


1. Execute below command with default configuration parameters on the EMR master node

spark-submit --master yarn --deploy-mode cluster DataGenerator.py



1. Log onto SparkUI to see how long the application took to run

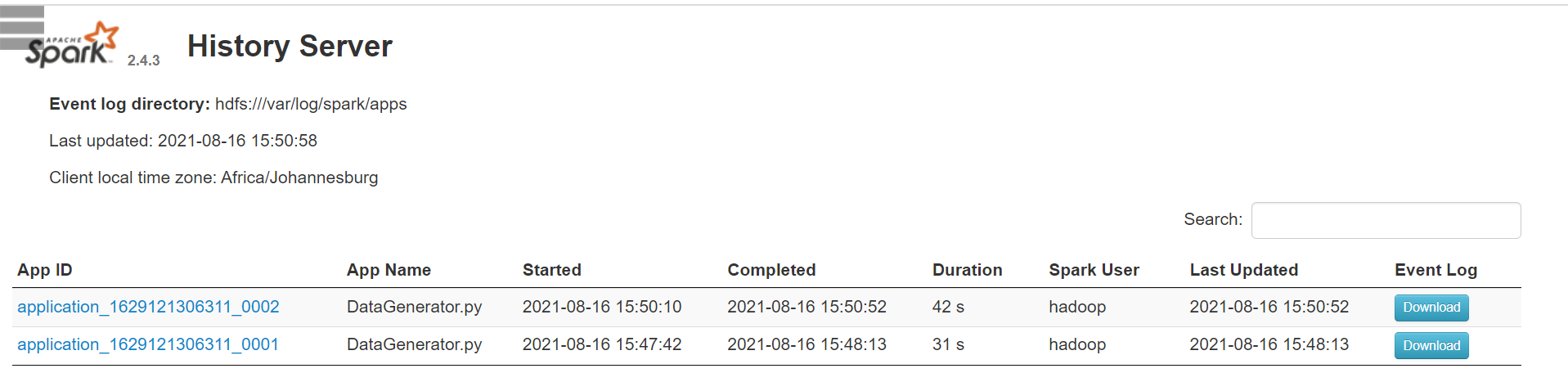


1. Execute below command with tuned configuration parameters on the EMR master node

spark-submit --master yarn --deploy-mode cluster --num-executors 5 --executor-memory 3g --executor-cores 1 DataGenerator.py



1. Log onto SparkUI to see how long the application took to run



1. Execute below command with tuned configuration on the EMR master node

spark-submit --master yarn --deploy-mode cluster --num-executors 1 --executor-memory 10g --executor-cores 3 DataGenerator.py



1. Log onto SparkUI to see how long the application took to run

