CSE5306, Distributed Systems
Fall 2017, Project 3
Due date: 11:59pm, Dec. 10, submission through Blackboard

Please read this:

1) Project assignments are to be completed by teams.
2) Total points possible: 100 pts.
3) Please add the following statement in the beginning of your submission.
   I have neither given or received unauthorized assistance on this work
   Signed: *Date:*

Project submission:

1. A report that briefly describes how did you solve the problems and what you learned? For example, you may want to make discuss the problems you encountered and the solutions to the problems.

2. The programming codes including any scripts that you wrote the run the program.

3. A detailed README file showing how to compile and run your program

Assignment-1 (40pts): Set up a realistic big data processing environment. Deploy Apache YARN in a Linux environment. YARN should be deployed in the pseudo-distributed mode, in which the daemons are run on a single node as separate processes. Download the stable Hadoop release 2.7.2. See the installation guide for reference. Note that you need to configure YARN to specify the replication factor and the block size of HDFS, the location of YARN daemons, and other YARN configurations. Once YARN is up and running, install the Intel Big Data benchmarks HiBench in YARN. 1) Show the successful completion of at least one MapReduce and one Spark job from the HiBench benchmarks; 2) explore to change the number of mappers, reducers for the MapReduce job, and the number of Spark executors for the Spark job, and study the performance of the jobs with different settings. Explain in your report how you changed the job configuration. Plot the trend of job performance with different number of mappers, reducers and executors.

Assignment-2 (20pts): Implement a simple MapReduce job – *IntSum* and execute it in the YARN environment. The job adds all integer numbers of an input file and outputs the sum of these numbers. Learn from the examples included in the YARN distribution, such as wordcount or grep, on how to write a MapReduce job. The source code of the examples can be found at hadoop-2.7.2/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-2.7.2-sources.jar

Use the following steps to test your program:

1. Make the HDFS directories required to execute MapReduce jobs
```
$ bin/hdfs dfs -mkdir /user
$ bin/hdfs dfs -mkdir /user/<username>
```

2. Copy the input files into the distributed filesystem
```
$ bin/hdfs dfs -put PATH_TO_INPUT_FILE input
```

3. Test your newly added program

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar
intsum input output
```

4. Check your results in the output file

```
$ bin/hdfs dfs -get output output
$ cat output/*
```

Assignment-3 (20pts): Re-implement the *IntSum* job in Spark and execute it in the YARN environment. Quantitatively compare the performance of the MapReduce and Spark implementations using the same input file.

Assignment-4 (20pts): Read the papers listed below and other online articles if needed, and answer the following questions based on your own understandings.

1. What are the key differences between Hadoop and Spark, and their respective advantages?
2. Discuss how to recover a failed task in Hadoop and Spark, respectively.

References:

1. Jeffery Dean and Sanjay Ghemawat, MapReduce: simplified data processing on large clusters, *in Proc. of the 6th conference on Symposium on Opearting Systems Design & Implementation (OSDI),* 2014
2. Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica, Spark: Cluster Computing with Working Sets, in Proc. of *the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2010
3. Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica, Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, in Proc. of *the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2012*