Final Project - Build a Map-Reduce - PROJECT REPORT

Team Members:

1. Karishma Raj
2. Nephi Calvin
3. Nikhil Sudireddy
4. Raghuveer Ramesh

Contents:

INTRODUCTION

This report is a summary of the activities that we've done to develop a Hadoop-like Framework. We started off by having designing discussions. Having those ready, we constructed a prototype implementation by using a Hadoop Program (Alice Wordcount implementation) as a base, and then commented out all 'org.apache*' classes and wrote Classes with Empty Methods to get a compilation working.

We then proceeded to fill out the Classes' method implementations. This was followed by a Pseudo-Mode that uses Threads to run the framework. We also developed an additional Pseudo mode to mimic the Distributed environment. We used a master-slave approach where there's one Master and multiple Slaves. (One Master to rule them all) The master allocates work to the Slave and each Slave waits indefinitely to receive work instructions. Note that the Slaves do not communicate with each other. This Pseudo –Distributed Mode was achieved by having multiple Slaves and One Master (each running on its own Terminal) communicate with each other using TCP – Socket connections.

We used ANT Scripts to compile and package our framework.

After having tested the Pseudo-Distributed-Mode we then proceeded to Deploy these Binaries onto Amazon EC2 (t1.Micro) instances to test the framework. To deploy, we made use of a few scripts that to start a Cluster of EC2 instances and securely copy the Framework, the application and preload any dependent JARs and then issue Start Commands. The Slaves are all started first and only then the Master Starts.

We then noted the performed tests to run our Framework on different number of parallel instances and have noted performance readings in the following sections.

FUNCTIONALITY AND STRATEGY

The Design of our Framework has these key Phases:

- Pre-Map Phase
- Map Phase
- Shuffle and Sort Phase
- Reduce Phase
- Combine output

PRE-MAP PHASE

The Master Sends out Host, Port and Bucket information across to each of the Slaves so that each Slave can respond to the Master. The Master then reads the input folder, computes the size of the input data, and then decides on the Total number of Mappers that need to be started. It also decides the number of mappers per slave.

MAP-PHASE

Once the Master has decided how the mappers are to be distributed, it sends out instructions to slaves such that, the input files can be mapped in parallel. The output of each mapper is first stored onto Disk and the Transferred to an intermediate bucket. The Map-Phase is set to be completed as soon as every Mapper has written its output to Disk and uploaded to the intermediate-bucket.

SHUFFLE and SORT Phase

The Master waits for each mapper to complete and then starts this phase. In this case, all the intermediate data Is in one bucket as a result of every mapper uploading its intermediate results.

REDUCE PHASE

Once the Shuffle and sort phase is complete, the Master knows the number of Keys that have been computed and then decides that number of Reducers. It also decides on the number of reducers per Machine. It then sends out Reduce instructions to the Slaves. Each Slave performs reduce in parallel and writes its output to both its Local Disk as well and S3-output Bucket.

COMBINE OUTPUT

After the reduce-phase has been completed, the result of every Reducer is placed onto the output-bucket. We then combine these into one file or leave them as multiple files accordingly.

Test Case performance

For each test program:

Did this test program require any new functionality to get to work?

- No

How did performance compare to Hadoop?

- Refer Table below

Did you produce the expected output?

- Yes

Assignment	HADOOP-PSEUDO	MY-HADOOP-PSEUDO (2 Threads)	MY-HADOOP-EC2
Alice-Word count	11 Seconds	4 seconds	10 minutes – (2 instances)
Alice-Median	14.7 Seconds	2 Seconds	31 Seconds – (2 instances)
Cluster Analysis	5.25 Minutes	5 Minutes	15 Minutes - (2 instances)
Missed Connections	52 Minutes	28.5 Minutes	33 Minutes – (4 instances)

CHALLENGES

- Packing of the JARs while having a package structure was tricky. We used ANT scripts to resolve this. We ended up with a bloated framework file of 65~MB (which included all the dependent libs) We then reduced this to about 20KB by having the EC2 machines download the dependencies and used appropriate path files in the Manifest file before running .
- Synchronization issues and Socket issues. We often encountered Synchronization issues and we made of 'Synchronized' keyword for methods that access shared resources to resolve this.
  We got several closed issues in the final combine phase (which took quite a lot of time in Alice word count and Alice Median) due to the number of keys being stored on the intermediate S3 bucket.

CONCLUSION

By building this framework, we've got a deeper understanding of how a combination of good parallelism, efficient data processing and minimal network overhead can lead to overall increased performance. Based on the performance tests we can conclude that our Framework is faster compared to Hadoop especially in Pseudo Mode. We can attribute this to the fact that Hadoop has a lot of overhead under the hood which our framework doesn't have to deal with. We ran all our EC2 distributed tests on the free tier T1.micro instances (single threaded low RAM machines). Using better machines (with more threads and RAM) might have resulted in better overall performance.

There are a few a future enhancements that we can think of. In our framework we noticed that as the number of Keys in the increases, the number of output files increases and this leads to reduced performance while trying to make a large number of s3: bucket puts and gets. Hadoop has several fail safe mechanisms in place in case of a dead/unresponsive machine in a cluster and this could be something to enhance in the future. We've used TCP in all our network communication messages. UDP might have made it faster.

Team member Contribution

Karishma - Report, StartClusterScript, StopCluster, Alice Testing
Nephi - my-mapreduce-script, Slave, Ant Build
Nikhil - Master , Context, MissedConnections Testing
Raghu - Pseudo, ClusterAnalysis Testing, README