CS6240

Assignment Report: Distributed EC2 Sorting

Vikas Boddu, Jun Cai, Xi Wang, Xinyuan Wang

April 1, 2016

Introduction

This document describes our implementation of a distributed sorting system using EC2 nodes. The purpose of this application is to find the top 10 temperatures amongst any given data set.

High-Level Design

Our design for distributed EC2 sorting starts of with each EC2 node receiving a list of EC2 IP addresses and S3 files that they will need to read. While reading this these files, we sample temperature data every n lines. After finishing the read of S3 files, we establish communications with the other nodes and send our sample data to each other node in the network. Then each node calculates pivots depending on how many nodes are there i.e. how many buckets for the quicksort to feed into. After calculating the pivots, each node sends the data appropriately to each node. Then each node sorts the data and outputs it to a S3 bucket.

Low-Level Design

We split up our design into multiple components such as: Cluster Management, Data Processing, Communication, and Barrier.

The EC2 nodes are managed through a bash script that allows easy creation and termination of 2 or 8 nodes. We partition the input S3 bucket based on the size of it's files and distribute an equal workload to each node. Finally, we maintain a list of EC2 IP addresses for each node we create. This allows us to tell our nodes which IPs they need to connect to. Also it is useful for SSH'ing into each node.

Data processing is the same for each node. It involves reading in the data from S3 and gathering sampling data from it. It also involves the calculation of pivots. This is done by taking the midpoint in a 2 node cluster. For a 8 node cluster, we take 7 medians. This is done by getting equal 8 intervals. This allows it to be scalable on any number of nodes.

Our communication consists of sending and receiving data between EC2 nodes. This part includes the creation and destruction of sockets. It includes sending temperature data and also ready messages. The ready messages are part of a barrier. The barrier is used as we do not want nodes to be going ahead of this part without all the other nodes also being ready.

When having multiple stages distributed across nodes, we do not want nodes to proceed to the next stage. If this were to happen, then they would be operating on incomplete or maybe even incorrect data. Hence we have a barrier that blocks processes for the node until it has received every other node's ready signals. Ready signals will only be sent when a node receives all the data it needs for the next stage.

Challenges Faced

One of the biggest issues we faced was that sometimes nodes did not receive the 'ready' signal from other nodes. When this signal was not received, nodes would be stuck in the barrier phase. We figured out that the issue might be due to our socket's buffers being overflown with data and that this packet was dropped. We fixed this by having these important barrier communications being done through a separate socket that is not touched by other functions.

One issue that we have not overcome, is the time complexities of 2 node sorting. With our current setup, one of the nodes takes 3x as long as the other node. We think that our design reaches a threshold on this node as it deals with a 0.2GB increment of data than the quicker node.

Performance

When running on EC2 m3.2xlarge instances, our distributed sorting program will take about 218 seconds (processing time only, 356 seconds if includes the data transfer to/from S3) to finish with eight slave nodes. Though in the 2-node case, we see a strange performance variance where one of the node can finish in about 820 seconds (processing time only, 1400 seconds data transfer included) and the other node will take 3500 to 4500 seconds to finish the work. We have confirmed that the workload is fairly balanced where each of the node will output data of the same size (6 GB).

Here is a runtime sample for each node in one of our tests:

(8-node setup)

Node No.	Processing Time(s)	Including Data Transfer(s)
0	201	355
1	186	315
2	198	321
3	159	281
4	186	319
5	167	282
6	191	328
7	211	336

(2-node setup)

Node No.	Processing Time(s)	Including Data Transfer(s)
0	4027	4585
1	821	1400

Results:

Here are the top 10 temperatures from our sorting output:

WBAN#	DATE	TIME	TEMPERATURE
40604	19960915	1550	558.5
26627	19980604	1152	251.6
14847	19960714	1150	245.1
26442	19980818	1250	237.6
40604	19971016	1954	214
40504	19971107	56	212.4
12816	19980124	2147	201
3866	19991215	955	201
14611	19991210	1855	201
14780	19991207	1155	201