CS 425 MP4 Report

Karan Sodhi (ksodhi2) & Nashwaan Ahmad (nahmad31)

Design:

For our IDunno implementation, we built off our previous MP's SDFS system. Like previously, every node could join a group and each node held the full membership list. The first node that started the group is the original coordinator. The coordinator manages scheduling jobs on the system alongside its other functions from the last MP.

In order to run a job, IDunno must go through a training phase. Currently, our implementation supports the Tensorflow InceptionV3 model to label images and the Tensorflow Multibox model to detect where people are located in an image. Any node can train the model by inserting a local copy of the model and test images into SDFS.

Next, a node can run a job by sending a TCP request to the coordinator containing what type of job to run and the batch size. The coordinator then starts scheduling the job by simultaneously sending TCP requests to the group containing what type of job to run and names of the batch of test files to run the model on. The nodes then run inference on the query and send back the results to the coordinator. The coordinator then writes the results to a file, saves it to SDFS, and schedules a new job on the machine.

If multiple jobs are run on the system at the same time, the coordinator must decide which job to schedule on a free machine. It does this by keeping track of how many queries of each job were able to be processed in the last 10 seconds. The coordinator will choose to schedule the same job the machine ran previously unless the difference between the query rates of the jobs is more than 20% which in that case the coordinator will choose to schedule the job with the lower query rate on the machine.

The coordinator also keeps track of what query every machine is currently running. If the coordinator detects that a worker went down it will reschedule the query it was running to a different node. Query rates between jobs will eventually converge within 20% since when the coordinator detects that the query rate for the job that ran on the failed node has slowed down, it will allocate additional resources to the job.

If the coordinator fails, then the next node on the membership list goes through an election process as described in the previous MP report where it tries to rebuild the system into a good state by replicating files. Additionally, it repairs jobs that were running in the system by sending a TCP request to every node asking what query it was on in the last Inception and Multibox iteration and the results of the query that it was not able to send to the original coordinator. The new coordinator writes the results into SDFS, calculates the next queries to run, and resumes scheduling the jobs.

Past MP Use:
As stated above, MP3 was used to maintain the SDFS where the models, training data, and results were stored. MP2 was used to build the distributed membership list which helped to elect the coordinator and to know what nodes to contact. MP2 was also used to detect failures which helped to trigger the replication of nodes and restarting of jobs. Additionally, whenever a job was received, a worker task scheduled or completed, or when a job was completed, that information is logged to a file. That file was able to be queried by MP1 which helped us a lot when debugging.

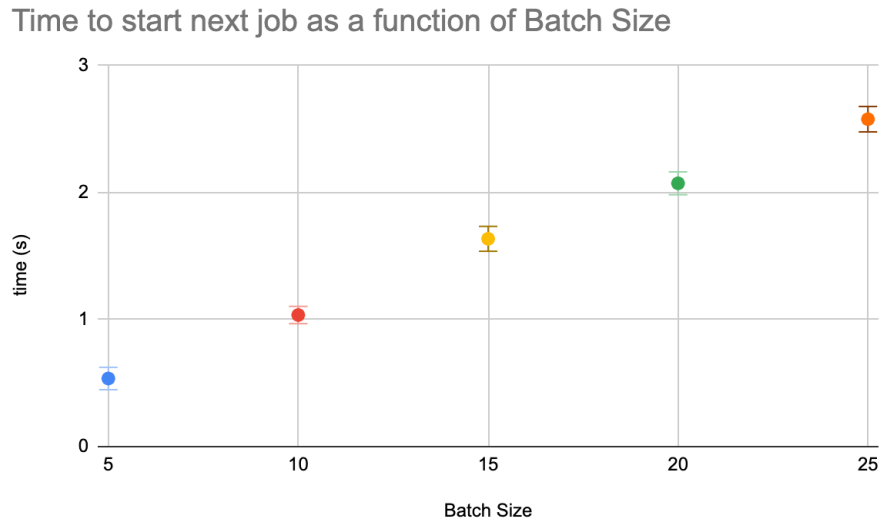
Measurements:
1a. When a job is added to the cluster, IDunno dynamically allocates resources to meet fair-time inference. It decides on the ratio of resources to allocate depending on the speed of a query. For example, if Job 1's query took 3 seconds and Job 2's query took 1 second, then 75% of resources would be allocated to Job 1 and 25% to Job 2 in order for them to achieve the same average number of queries per second.

1b. When a job is added to the cluster it takes on average the cluster 0.536 seconds with a standard deviation of .088 seconds to start executing queries of the second job. This makes sense since as soon as a node is free it will be assigned the new job.

2. After the failure of one (non-coordinator) VM, on average it takes the cluster 15.66 seconds with a standard deviation of 1.31 seconds to resume "normal" operation. Part of the reason it takes so long is because the coordinator has to also re-replicate the files the failed nodes was holding, delaying the rescheduling and processing of queries.

3. After the failure of the coordinator VM, on average it takes the cluster 19.498 seconds with a standard deviation of .733 seconds to resume "normal" operation. In order to resume normal operation the new coordinator has to go through the election process, re-replicate files, and re-calculate the jobs it needs to run, which is why it takes so long.

Differences between batch size primarily affect 1b) as shown in this graph



The time it took to start the next job increased linearly as batch size increased which makes sense because the bigger the batch size the more time it will take for a node to be free and request the new job.