Team05 - Yuankai (yuankai4) and Boyou (boyouh2)

MP4 Report: IDunno, a Distributed Learning Cluster Team05

Design: We chose Alexnet and Resnet as models. The system will run python scripts to handle the training and the inferences. In the training stage, we will load the model and save it locally. The VM will use the models to do the inference. We use all VMs as workers and one of them will also do the coordinator work. Any VM can be the client. When the user adds a job, the coordinator will process the input by reading the list of image filenames and separating them into batch sizes per query. We use a zip file to make it easier to add the test input files to the system. The coordinator will know the list of available workers and assign the query to them. For each worker, it will assign one query at a time. After the query is done, the worker will send the result back to the coordinator and become available again. The coordinator can then assign a new query to it. The coordinator will store the result locally first. When the jobs are done, it will submit the result to SDFS. The coordinator is also responsible for storing all the query results and time data and calculating the job statistics. All above operations on the job queries will be reflected to a backup coordinator that is selected during setup.

When the coordinator fails, we will use the backup coordinator. It will find a new backup coordinator and then broadcast the information into the system. We will redo the work that is in progress with other workers because they might have already submitted the result to the failed coordinator before the system detects its failure.

When a worker fails, we will remove the worker from the available workers so that the coordinator won't assign new queries to it. The coordinator keeps a record of all the tasks that are in progress. So it can move the tasks assigned to those workers back to the to-do list and it will find some other workers to do the queries later. If it happens to be the backup coordinator, the coordinator will find a new backup and save a backup of the job data there.

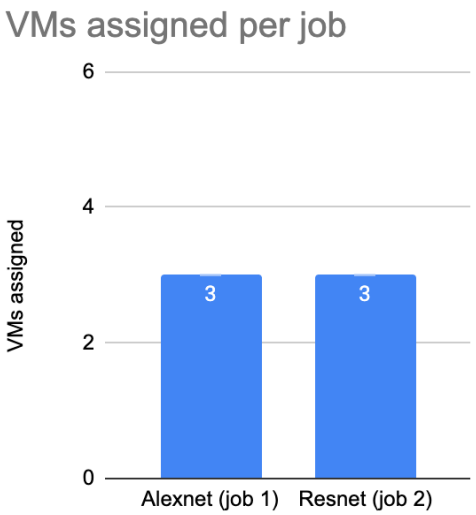
Measurements:

1) Fair-Time Inference:

1a) When a job is added to the cluster (1 -> 2 jobs), what is the ratio of resources that IDunno decides on across jobs to meet fair-time inference?

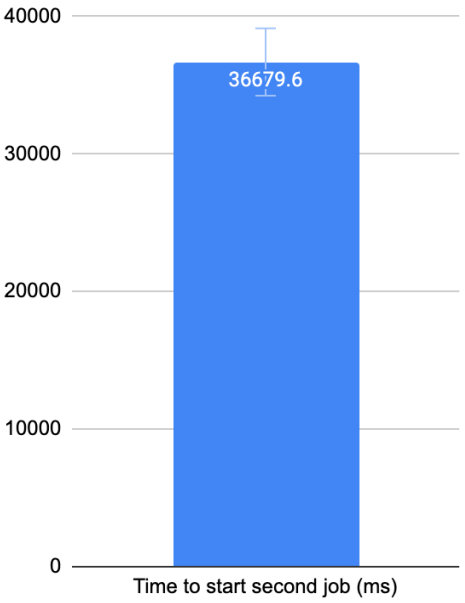
The coordinator will try to balance the query rate of the two jobs. Therefore, when it assigns a query to an available worker, it will prioritize the query that has a lower query rate. Since we only assign one query to a worker at a time, the worker can easily pick the expected model to run the query and return the result back to the coordinator. This design can handle any unexpected latency or failed worker and can handle different runtime of the models. Because the runtime of using Alexnet and Resnet is about the same, we would probably see an even split of resources.

We use 6 VMs to do the testing. When the second job is just added, the coordinator assigns 0 VMs to run the first job and 6 VMs to run the second job. When the query rate of the second job increases and become close to that of the first job, the coordinator assigns 3 VMs to run the first job and 3 VMs to run the second job. Standard deviation is 0 because we get the same result in all five measurements. The result is expected because it will achieve a similar query rate on the two jobs by working on the job that has a lower query rate first and then split the resource evenly.



1b) When a job is added to the cluster (1 -> 2 jobs), how much time does the cluster take to start executing queries of the second job?

The coordinator will need to get the file from SDFS, process the input job, and separate it into queries. Then the system will need to wait for a VM to finish its current job and after that, the coordinator can assign it to the new job. We do the testing using a job of 28 queries. The result is expected because it needs to get the file from SDFS and read the file to create queries. Both are time-consuming operations. We also need to add the wait time before a VM becomes available. It is also expected that the standard deviation is relatively small because our system doesn't have too much uncertainty during this operation.

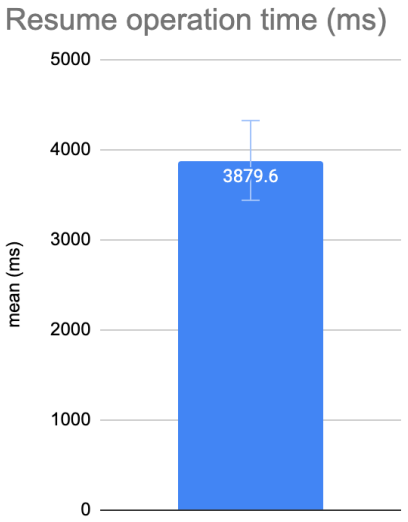


Times to start the second job (ms)	mean (ms)	std
38701, 33370, 37326, 39677, 34324	36679.6	2449.1

2) After failure of one (non-coordinator) VM, how long does the cluster take to resume "normal" operation (for inference case)?

It will first detect the failure and then update the membership list. The coordinator will know the work leaves and re-add the query that the leaving node is working on back to the to-do list. This process does not affect the normal operation of other machines. The result is expected because most of the time will be spent on detecting the failure. The variation in time also comes from the interval between the last ping message and the failure.

Resume operation time (ms)	mean (ms)	std
3590, 4760, 3687, 3638, 3723	3879.6	442.5



3) After failure of the Coordinator, how long does the cluster take to resume "normal" operation (for inference case)?

Similarly, it will first detect the failure and then update the membership list. The backup node will become the coordinator and share this with other nodes. It also needs to reset the in-progress work and reorganize the query data after the original coordinator's leaving. Then the new coordinator will start to assign queries to the workers. The result is expected because most of the time is still on failure detection. The backup coordinator already has all the data needed to assign the queries.

Resume operation time (ms)	mean (ms)	std
3625, 3817, 3228, 3616, 3542	3565.6	191.7