CS 425 – MP4 Report

Group 69 – mgoel7 and bachina3

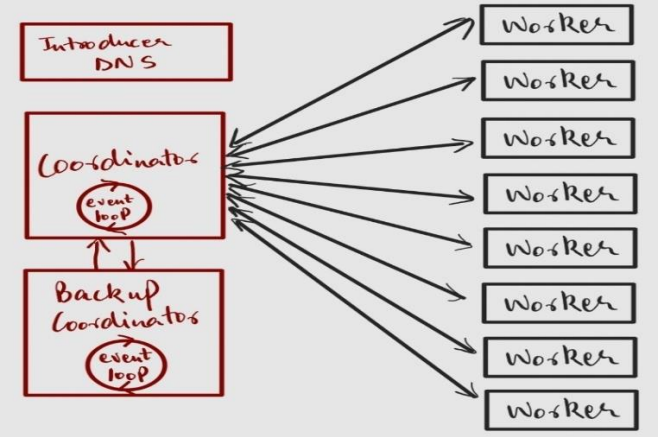
DESIGN AND ALGORITHM

Our IDunno scheduler is built on top of an asynchronous design paradigm where each server handles every request in a single thread. This significantly reduces performance overheads usually incurred with threads while ensuring high level of concurrency. We have implemented the system using Python.

In our implementation, we have designated one node as the coordinator and one node as a backup coordinator. These two nodes are not part of the workers nodes who process the jobs. The rest of the nodes double as worker nodes and clients. The coordinator node is responsible for keeping track of the running jobs and pre-empting them if the resources need to be rebalanced. The coordinator is also responsible for relaying its state to the backup coordinator for checkpointing. To make sure that a new node knows who the current leader/introducer in the network is, we have implemented a separate process to be dedicated as a DNS server which lets the new node know of the introducer's host and port. This DNS server is not part of the network and we assume that this process is always running. Furthermore, we have used two models for inference, namely InceptionV3 and ResNet50. We have also created 2 applications which work image classification jobs.

The operations in the system are run in the following order:

1. **SUBMIT_JOB_REQUEST** – The client node sends the submit job request to the leader node. The client specifies the model it wants to run this job on and also images it wants to run the job on.
2. **Leader Job Processing** – When the leader receives the submit job request, it processes the job into fixed batch sizes for each VM. This batch size can be set before running the models. It then adds all the batches to the model's queue. The leader also relays this information to the backup coordinator to update the checkpoint. The leader then checks if any job is already running. If no job is running, it assigns all free workers to 1 batch each. If a job is already running, the leader calculates an optimal splitting of the worker nodes for each job. It then pre-empts tasks from the worker nodes and schedules the job the of the other model to run on it by sending the WORKER_TASK_REQUEST command. It then pops the sent batch from the pending queue to the in-progress queue.
3. **WORKER_TASK_REQUEST**: Whenever, a worker receives this command, it pre-empts the task it was already running. It then fetches the images from the SDFS required for this batch of queries. After fetching, the worker loads the inference model and performs the inference task. After completion, the worker puts the generated output in the SDFS and notifies the leader node using the WORKER_TASK_REQUEST_ACK command.
4. **WORKER_TASK_REQUEST_ACK –** When the leader node receives this command from the worker, it updates it state and removes the batch from the in-progress queue. It also calculates and update various job metrics like query rate, query processing time and the total queries processing so far. Furthermore, it updates the backup coordinator to update its state and metrics like total queries processed so far.



FAILURE TOLERANCE

As soon as a failure is detected by the leader, the leader pre-empts the job that was being run by the failed node and puts it back to the front of the pending queue. This batch will then be scheduled as soon as any other worker node gets free or if the balance of the resources needs to be changed at that moment. If a leader node itself fails, then the backup coordinator is elected as the new leader and it starts running jobs from its last updated checkpoint. Any job that was currently running on the worker nodes gets pre-empted and rerun in this case.

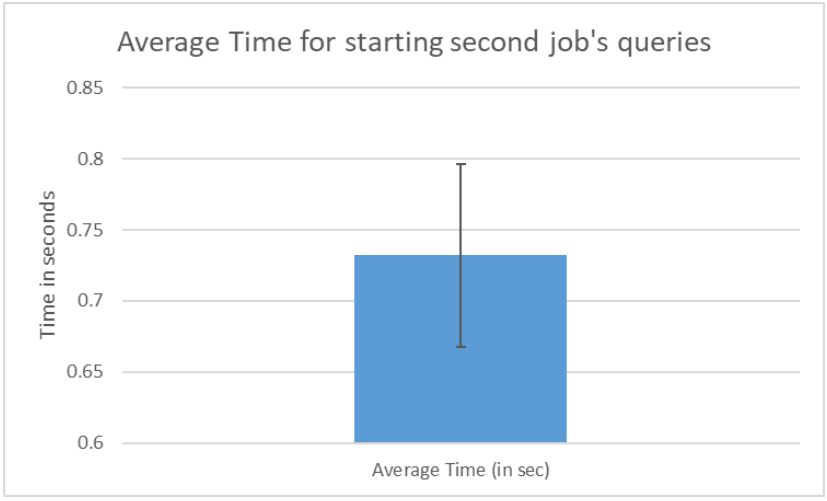
PERFORMANCE

1. **Ratio of resources allocated to each job**

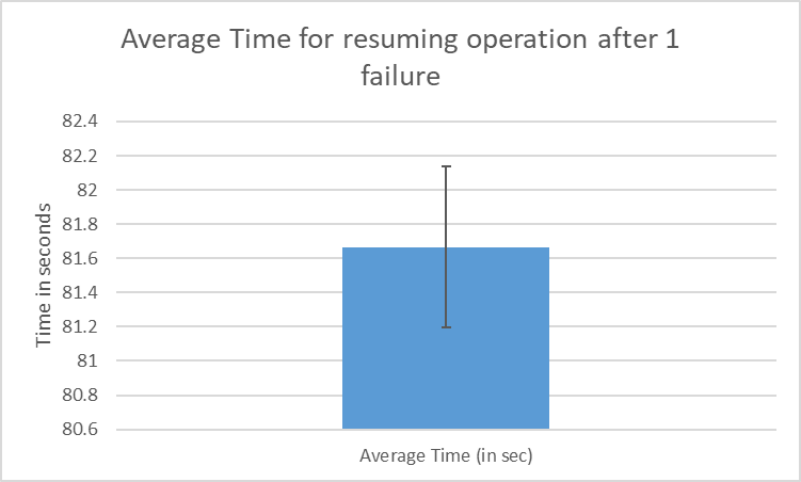
   Based on our testing, our system is correctly allocating the resources to pass the fair resource allocation based on the batch sizes for each model.

	Batch sizes for each model			
Model	10 queries of InceptionV3 and 20 queries of ResNet50	10 queries of InceptionV3 and 5 queries of ResNet50	10 queries of InceptionV3 and 10 queries of ResNet50	2 queries of InceptionV3 and 10 queries of ResNet50
InceptionV3	5 VMs	4 VMs	4 VMs	6 VMs
ResNet50	3 VMs	4VMs	4 VMs	2 VMs

2. **Time taken by the cluster to start executing queries of the second job**
   We have implemented our system in such a way that if the second job's query runs while the first job is already running, the system will pre-empt at least one of the VMs and assign it to the second job. Therefore, the time taken by the cluster to start executing the queries of the second job will be very less. We have plotted the following graph based on our testing. As evident, the time taken is miniscule.



3. **Time taken by the system to resume normal operation after one failure**
   We have calculated the time taken by our system to resume normal operations after one failure in the following way: We calculate the time it takes for the system to detect the failure and remove the failed node from everyone's membership list. We then calculate the time it takes from this point to schedule another batch on non-failed VMs. The total time is the time it takes for the system to rebalance and resume normal operations. The plot for this metric is given below.



We have set our PING_DURATION to be 12 seconds and CLEANUP TIME to be 30 seconds. Furthermore, we wait for 4 pings to suspect a node as a failure. Therefore, most of the time in this metric is the failure detection time and not the time taken to schedule the job.

4. **Time taken by the system to resume normal operation after coordinator failure.**
We have calculated the time taken by our system to resume normal operations after one failure in the following way: We calculate the time it takes for the system to detect the failure and remove the coordinator from everyone's membership list. We then calculate the time it takes for election to take place which involves sending the election and coordinate messages along with updating the introducer DNS. We then calculate the time it takes from this point to reschedule the jobs in the second coordinator's queue in all the non-failed VMs. The total time is the time it takes for the system to rebalance and resume normal operations. The plot for this metric is given below.