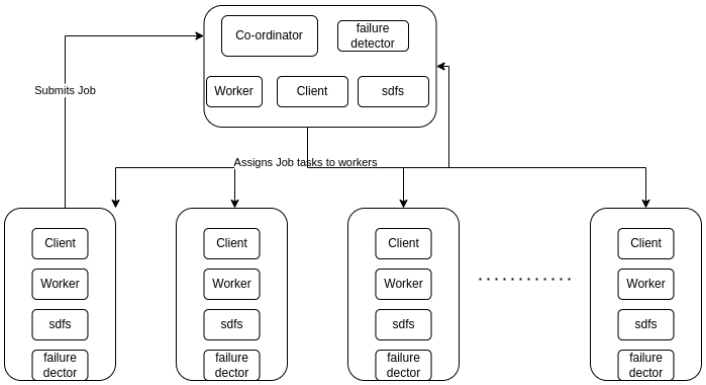


Design:
Our system uses the models LeNet with MNIST data set and Resnet-50 with flowers dataset. We directly took the trained models and uploaded(put) them to SDFS(MP3), when required the worker VM will download the required model to do the inference during the starting of the Job.

We made sure that the training and test data doesnt have anything in common. All the testing files(images) for each model are uploaded to VMs using the MP3 SDFS, each file is replicated on 4 VMs. Job here refers to the Model which is being used to infer. Task refers to the processing a batch size of queries related to a specific job.

Co-ordinator: When an user submits the request from any VM to start the Job along with the batch size, this request will reach the Co-ordinator and Co-ordinator assigns a Job_Id to the Job and then allocate the VMs for this Job. These allocated VMs will be stored as allocated_vms list for each job and these VMs are decided by the Co-ordinator based on the active VMs and ongoing Jobs in the cluster at that moment. To VMs present in allocated_vms list, all the tasks of a Job will be distributed in a round robin fashion. After completion of every inference task, respective VM will send an ACK to the Co-ordinator. Only then the Co-ordinator assigns a next task to that VM. In this way, tasks will never queue up near any VM and at any point of time only one task will be executed at a VM. Co-ordinator maintains the information like allocated_vms list, and the remaining tasks list for each Job. Immediately after every update to this information, Co-ordinator shares this information to Standby Co-ordinator

Standby Co-ordinator: Stand by Co-ordinator will have the same information as Co-ordinator as Co-ordinator sends the message whenever it updates its data of allocated_vms and remaining tasks list. When the Co-ordinator fails, Standby Co-ordinator will get to know about this failure through failure detector MP2 and then Standby Co-ordinator will become Co-ordinator and will let the system know that it became Co-ordinator and then continues the normal operations with the information it has. It also sends the information to the next Standby Co-ordinator.

Worker VM(Non Co-ordinator): Whenever a VM is allocated to Job, then VM loads the required model, when an inference task is assigned to a VM, then it downloads the files(batch of testing files) according to the task. VM fetches those files using get command(MP3) to its local directory and then infer the testing file and update the output(inference result) into a file named as Job_Id.csv and sends back an ACK to Co-ordinator. This Job_Id.csv will be an SDFS file and is updated in the same way as in MP3.

Dynamic Resource Allocation:

Initial allocation of VMs to a Job: Whenever a Job inference request is submitted by a client, The Co-ordinator then takes the count of active VMs in the system and Jobs are that are inprogress in the system and then equally distributes VMs across the jobs.

First Job is requested: Let's assume there are 10 active VMs (VM1-VM10) in the system. As this is the first Job and no Job running currently on the system. Co-ordinator

allocates all the VMs VM1-VM10 to Job1. This strategy maximmises the usage of resources. So now allocated_vms list of Job1 will be {VM1, VM2, …., VM10}. Now all the tasks of Job1 will be assigned to the these VMs in round-robin fashion till all the tasks of Job1 complete.

Second Job is requested: Assume Job 1 is not completed yet and 10 VMs are active. Initially these 10 VMs are allocated equally among the two Jobs i.e now the allocated_vms list of Job 1 will become {VM1, VM2, VM3, VM4, VM5} and allocated_vms list of Job2 will be {VM6 ,VM7 ,VM8 ,VM9 , VM10}. Though the allocated_vms list of Job1 is changed, we dont preempt any running tasks on the VM6-VM10. Only when the task of Job1 is completed then tasks of Job2 will be assigned these VMs in round-robin fashion till all the tasks complete.

Balancing the Query rate: Co-ordinator constantly monitors the query rates for the jobs in progress. If the gap is more than 10% in the query rates then it updates the allocated_vms of the Jobs inorder to make sure the gap is less than 10%. Lets take the scenario where Job1 has higher query rate than Job2 with query rate gap morethan 10%, then the Co-ordinator picks a VM from Job1 and allocates it to Job2. This balancing will continue to happen till the gap is less than 10% in the query rate of both the Jobs. This can be scaled to any number of VMs.

1)Fair-Time Inference:

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

Initially when a Job is requested, VM's are allocated equally across all the Jobs, like as mentioned above in the Dynamic Resources Allocation section. So if there is only one Job, then all 10 VMs will be allocated to it. If there are two Jobs then initially both the Jobs are allocated 5 VMs each so ratio initially will be 1:1, but as the Co-ordinator constantly try to reduce the gap between query rates of Jobs as each Model(Job) has different inference time for a query, VMs might get allocated to task of different Job on the fly as mentioned in the above Balancing the Query rate section. If a Job has high inference time,then more VMs will be allocated to maintain the query rate gap to be less than 10%. So VMs are allocated to the Jobs fairly.

b) When a job is added to the cluster, how much time does the cluster take to start executing queries of the second job?

As we dont preempt the running tasks, Once the VM completes a task then only the tasks of second job will be allocated to this VM. So in the worst case scenario, tasks of Job2 wait till atleast one task of Job1 is completed on a VM that is allocated to Job2.

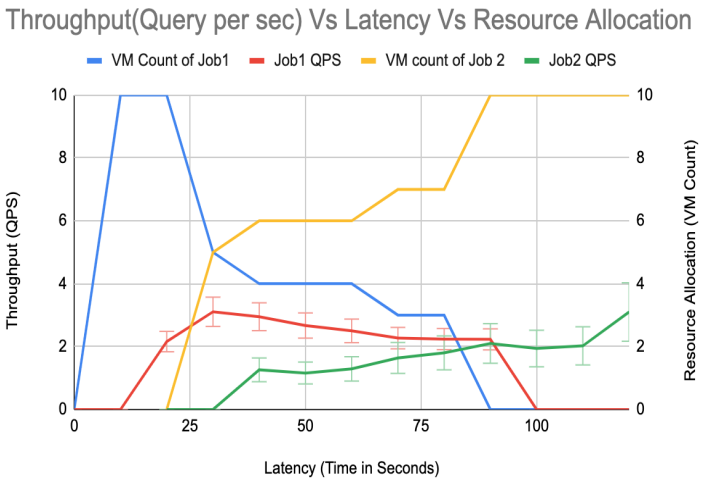
Measurements: We took the measurements five times for each data point and then took the mean and standard deviation. Batch size of both Jobs is taken as 2 for the following datapoints.

	Mean (sec)	Standard deviation (sec)
Time taken for Job 2 to start executing.	4.17846	0.7034
Time taken for cluster to become normal (After 1 Non-Worker failure)	12.433	3.238
Time taken for cluster to become normal (After Co-ordinator failure)	12.892	3.821

2)After failure of one (non-coordinator) VM, how long does the cluster take to resume "normal" operation (for inference case)?,
If any worker VM is failed, through the failure detector(Mp2) this failure will be known to entire system in 4 seconds(worstcase). When the co-ordinator get to know about this failure then the task that was running on failed VM will be re-assigned to another active VM. All the other inference tasks run parallely on all other active VMs without any interruption and balancing of the Query rate also run in parallel to this, only extra overhead is to re-run the failed task again.

3)After failure of the Coordinator, how long does the cluster take to resume "normal" operation (for inference case)?
Co-ordinator constantly updates the standby co-ordinator with the information which it has about the tasks completed and the next tasks for each job. If the co-ordinator fails then through the failure detector(MP2), standby co-ordinator will know that Co-ordinator failed in 4 seconds(worst case). It immediately becomes the coordinator. But in that 4 seconds, VMs continue processing the tasks normally but when the VM completes the task during this time, It tries to send the ACK to co-ordinator but couldn't and No task will be assigned to this VM till Co-ordinator is back. So VM continues to retry sending the ACK to coordinator until it reaches the co-ordinator. After 4 seconds, when the Co-ordinator is back then ACK will be sent successfully and VM will get its next task. Balancing of the query rate will continue once the Co-ordinator is up.

Usage of MP1, MP2 & MP3: As stated above, MP4 leverages MP3 SDFS to fetch the trained models, input files to infer and also to update the results and MP2 failure detector, inorder to update the metadata which contains information about task. Membership list designed in MP2 will help to maintain the information about allocated_vms list and metadata uptodate. We used the same marshaled message format from MP2. We logged the errors of MP4 programs into a file. We then used the distributed grep of MP1 to debug which saved us a lot of time.



Discussion: The plot shows that when the first job is submitted all of the VMs are allocated for Job1. Once the second job is also added(at 20s), the no.of VMs allocated for Job1 dropped along with its query rate and VMs got allocated to Job2 as well.The query rates started converging after a few seconds as the no.of VMs allocated changed to balance the query rate and after a while the query rates of Job1 and Job2 were within 10% of each others query rate.

Once the Job1 was completed all VMs got allocated to Job2 and the query rate of Job2 started increasing. This measurements were take for batch size 5, and from our experiment we found that the larger the batch size the harder it is to maintain the 10% query rate difference and as the batch size gets smaller the query rates converge faster to be within 10% difference(shown in the below table) and get less volatile.

Batch size	2	4	6	8	10
Mean Time to converge (sec)	6.43	9.8	11.17	16.8	22.56