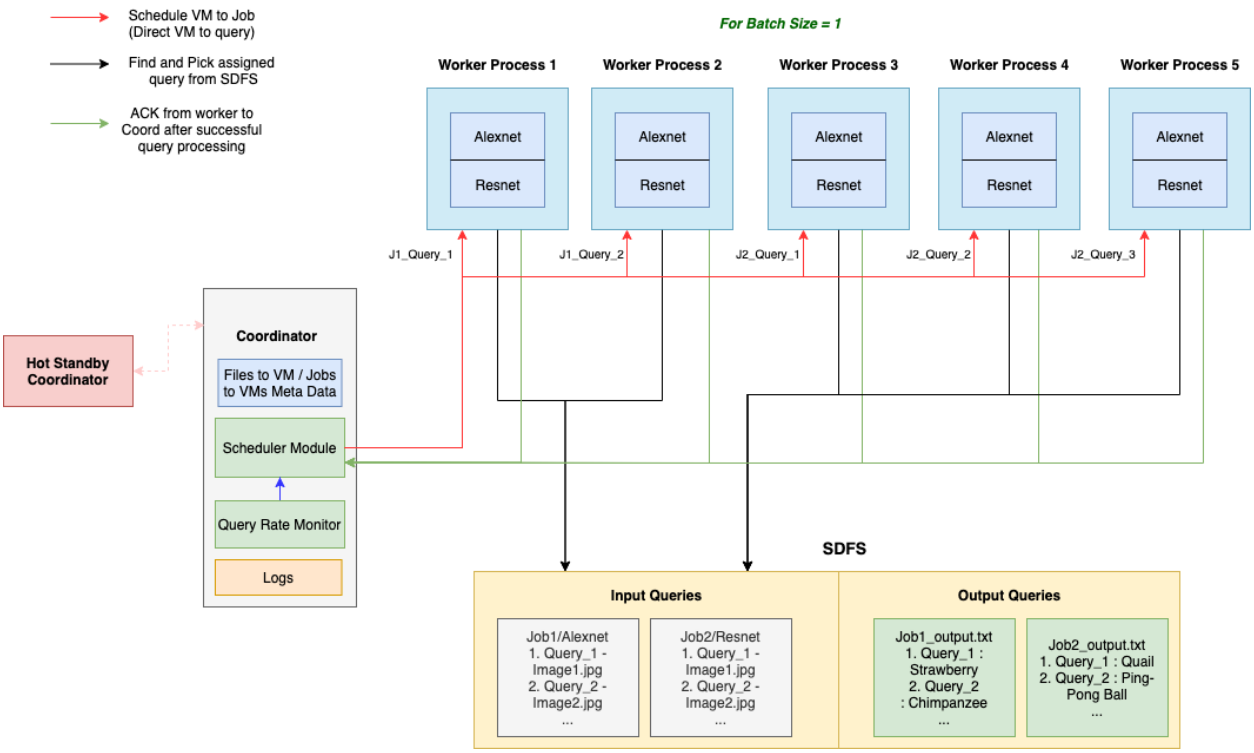
CS425: Machine Problem 4: IDunno

Group 65 : Aruna Parameswaran (netid: aruna2); Sanjit Kumar (netid: sanjitk3)

Design:

The IDunno system is a distributed ML inference query processing system that processes 2 kinds of ML inference jobs asynchronously while ensuring that both jobs get fairly equal amount of resources (VMs) - such that the difference in their query processing rate is within 20%.

The system is designed as described below.



We have opted to use Alexnet and Resnet to pre-train and use them in the worker processes as our two models.

Fair Time Inference Working:

1. The jobs are created and placed in the SDFS system. The data is a part of the test data split from the Imagenet dataset used to train the Alexnet and Resnet datasets for image classification.
2. The Coordinator contains metadata that includes which queries (files/images) are located in which VMs (from MP3 design).
3. The Jobs are first scheduled in the VMs based on the number of queries (or number of queries/batch_size)

4. The Coordinator informs the workers which query(s) to execute by sharing details of which job and which file from the SDFS needs to be fetched
5. The worker process fetches the required file from SDFS, executes the ML job, and returns the prediction value along with an ACK to the coordinator, which writes the result to a JSON file in SDFS
6. The Coordinator's query rate monitor collects ACKs from workers and periodically calculates the query rate of each job. It also calculates the time remaining to complete the overall job
7. If the difference in query rate is greater than 20% it dynamically calculates the desired redistribution of load in the following way,
    1. *fraction of VMs for Job 1 = total time(each query/batch for Job 1)/(total time(each query/batch for Job 1) + total time(each query/batch for Job 2)*
    2. *fraction of VMs for Job 1 = total time(each query/batch for Job 1)/(total time(each query/batch for Job 1) + total time(each query/batch for Job 2)*

For e.g., J1's batch/query is estimated (during inference phase) to take 10ms and J2's batch/query will take 20ms. The Allocation for J1 = 10/30*available VMs and Allocation for J2 = 20/30*available VMs. This means that if there are 9 VMs available in the system, J1 (usually) gets 3 VMs, and J2 is allocated 6 VMs. Intuitively this behavior is expected as J1 takes half the time to complete as J2.

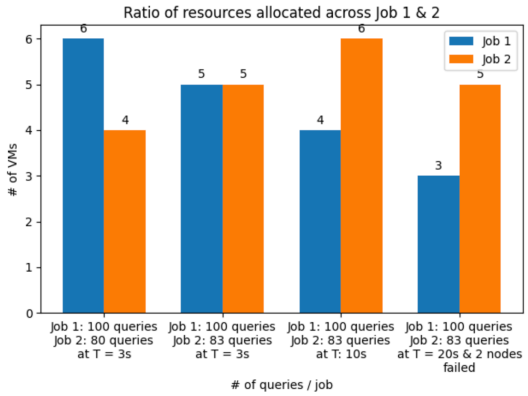
Fault Tolerance:

Fault tolerance covers both failure of a worker node and failure of the coordinator. Since the system is build on top MP2, we leverage the failure detector built back then to detect the failure of any node in the system. The failure of a worker node is detected first by one of its neighboring nodes, and the information is disseminated via the ring-based failure detector. The coordinator, when informed of the worker failure, removes all queries assigned to the VM that have not been yet been completed, and marks the state of each query as 'pending'. The scheduler module then picks up each pending query and assigns it to a new VM. The overall behavior is that despite worker nodes failing while processing queries, the system converges fairly quickly.

The coordinator has a dedicated backup node, which acts as a hot standby coordinator upon failure detection. The coordinator and its standby exchange packets regularly, including ping-acks, and backup of the stored metadata. If the standby coordinator does not receive any new packet in 3 seconds, it marks the coordinator as failed and takes over the system. The worker nodes begin receiving instructions from the backup coordinator. They mark the old coordinator as failed and instead begin communicating with the new coordinator. Immediately upon failure, the standby coordinator performs a query diff and re-executes any queries that were not updated properly.

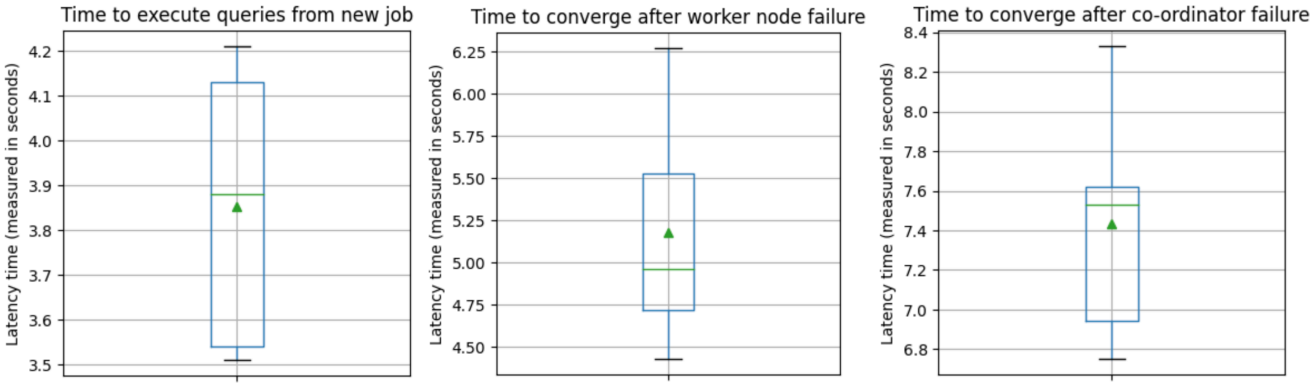
Metrics:

1a) The bar graph depicts the different VM assignments splits across Job1 & Job2. As explained previously, the dynamic calculation helps to ensure that the overall query time across jobs is



within 20%. The value is not always exact as the ratio is rounded down (as seen in data point 2). Another thing to note is that the system initially assumes that both Job 1 and Job 2 are processed at the same rate. Over time, the system adjusts this assignment based on the calculated query rate of the system (as seen in the data point 3). This ratio is also adjusted upon node failure.

1b) The average time required to begin executing queries when a new job is added is approximately 3.9s. Initially when there is only 1 job in the system, the query rate monitor (which also acts as a Load Balancer) assigns all the available VMs to the job. When a new job is added, it recomputes the VM allocation, taking into account the current query processing rate. This re-computation has been configured to be executed once every 3s in the system, which is likely why we see the value just slightly above 3s. When a VM previously assigned to Job 1 is moved to Job 2's pool, the coordinator first waits for any current queries to finish before assigning it new ones, which is also a likely contributing factor here.



2) When a worker node fails, the co-ordinator is informed through the failure detection dissemination process from MP2. The coordinator is responsible for executing MP3's SDFS file repair replication strategy, while simultaneously ensuring that the system is not in a partial state. The system is able to converge fairly quickly due to the scheduling logic implemented; when a worker fails, the coordinator does not immediately reassign or trigger the failed query, but leaves it to the scheduler to pick up at a later point in time.

3) Mitigating the co-ordinator's failure takes up more overhead, especially when the backup co-ordinator kicks in. While the failure can be detected quickly (the backup coordinator periodically sends pings), the system takes more time to converge. Particularly, there may be a number of queries that have already finished executing on the worker nodes but have not been marked by the coordinator. In the worst case, the backup coordinator has to re-trigger these queries to execute once more, which is likely why the time taken is the highest out of all three.