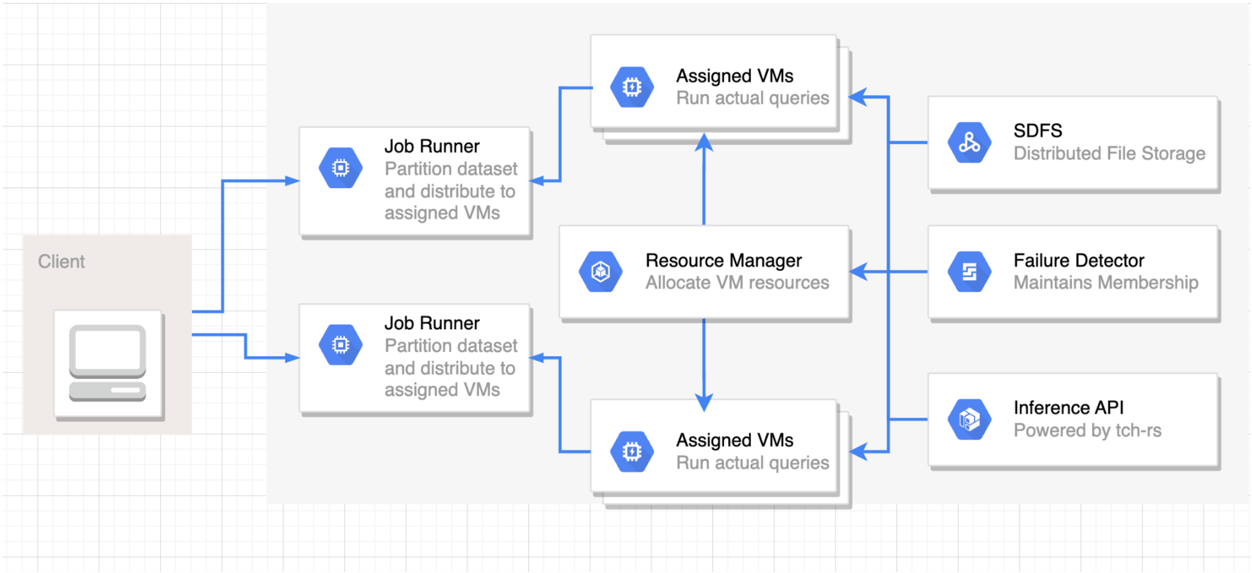
CS425 MP4 Report
By: Tony Chang(yaowenc2), Cay Zhang(cz74)

Design:

We've decided to use resnet18 and alexnet as our models to conduct the inference on the dataset Imagenet 2012. We've utilized tch-rs, the rust bindings for the C++ api of Pytorch to load models and conduct inference.

We have drawn a rough diagram of our system components below:

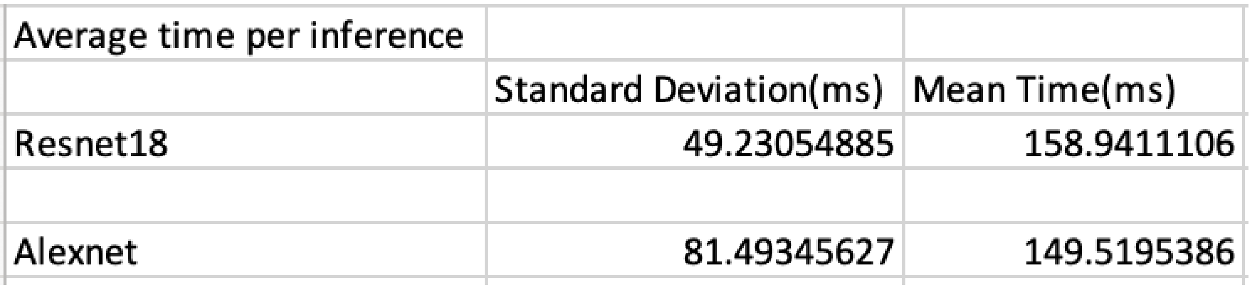


When we use the predict command on any VM, we will spawn the 2 jobs across the VMs. The leader will determine and dynamically adjust hyperparameters such as the query rate. Prediction results are sent back to and displayed at the leader. VMs will be dynamically assigned to the jobs when there are multiple jobs spawning to ensure a steady query rate across different jobs.

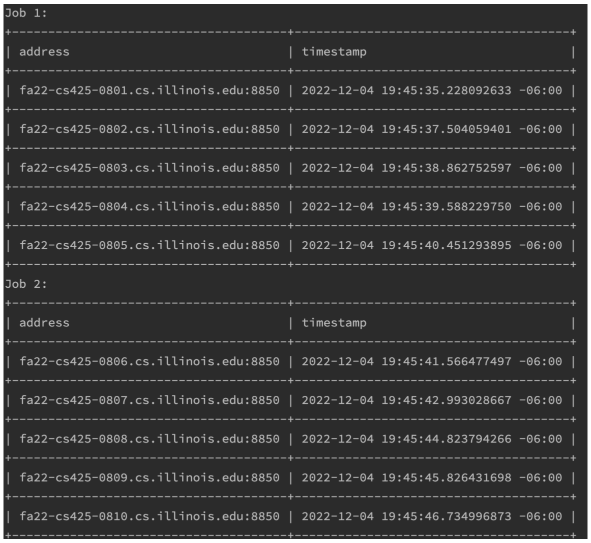
A leader select its successor and constantly updates the successor with its current running job information. When the leader fails, its successor becomes the new leader and updates every VM about its new identity. If there is inference work already running when the former leader fails, the new leader will try to pick up where it left off.

Fair Time Inference:

1a. Ratio of resources(VMs):



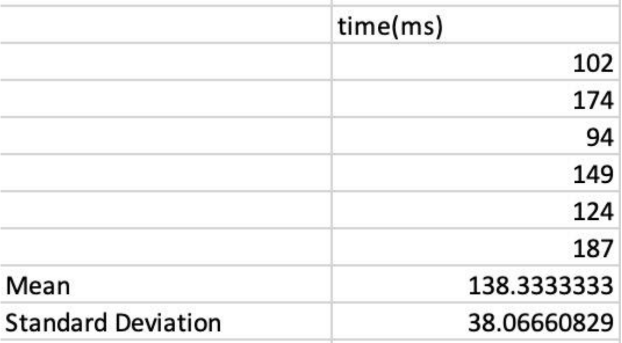
Average time per inference		
	Standard Deviation(ms)	Mean Time(ms)
Resnet18	49.23054885	158.9411106
Alexnet	81.49345627	149.5195386



```
Job 1:
+--------------------------------------+--------------------------------------+
| address                              | timestamp                            |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0801.cs.illinois.edu:8850 | 2022-12-04 19:45:35.228092633 -06:00 |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0802.cs.illinois.edu:8850 | 2022-12-04 19:45:37.504059401 -06:00 |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0803.cs.illinois.edu:8850 | 2022-12-04 19:45:38.862752597 -06:00 |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0804.cs.illinois.edu:8850 | 2022-12-04 19:45:39.588229750 -06:00 |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0805.cs.illinois.edu:8850 | 2022-12-04 19:45:40.451293895 -06:00 |
+--------------------------------------+--------------------------------------+
Job 2:
+--------------------------------------+--------------------------------------+
| address                              | timestamp                            |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0806.cs.illinois.edu:8850 | 2022-12-04 19:45:41.566477497 -06:00 |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0807.cs.illinois.edu:8850 | 2022-12-04 19:45:42.993028667 -06:00 |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0808.cs.illinois.edu:8850 | 2022-12-04 19:45:44.823794266 -06:00 |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0809.cs.illinois.edu:8850 | 2022-12-04 19:45:45.826431698 -06:00 |
+--------------------------------------+--------------------------------------+
| fa22-cs425-0810.cs.illinois.edu:8850 | 2022-12-04 19:45:46.734996873 -06:00 |
+--------------------------------------+--------------------------------------+
```

The algorithm approximately allocates equal amount of resources for both jobs. As we can see in the screenshot on the left, we have 5 VMs for job1 and 5 VMs for job2. This is reasonable since the average time per inference of resent18 is very similar to Alexnet.

1b. Time for 2nd job to start executing queries:

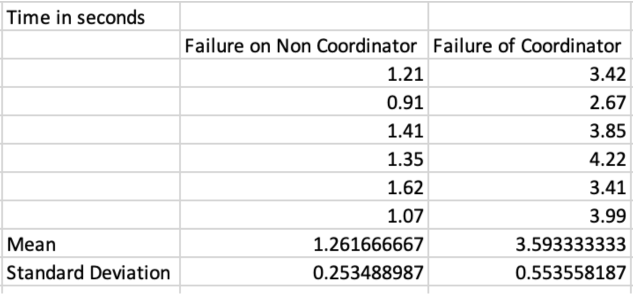


	time(ms)
	102
	174
	94
	149
	124
	187
Mean	138.3333333
Standard Deviation	38.06660829

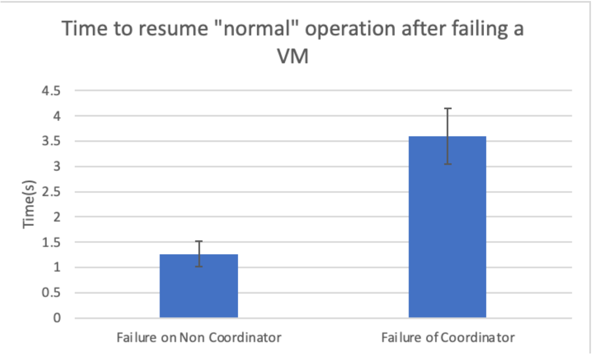
The following trials were done when we turn on all 10 VMS. The second job will start executing queries nearly instantly. This is expected because we just need to spawn another job and VMs will starting allocated and execute the queries.

Failure Detection:

2 & 3. Time to resume “Normal” Operation



Time in seconds		
	Failure on Non Coordinator	Failure of Coordinator
	1.21	3.42
	0.91	2.67
	1.41	3.85
	1.35	4.22
	1.62	3.41
	1.07	3.99
Mean	1.261666667	3.593333333
Standard Deviation	0.253488987	0.553558187



We’ve run 6 trials overall for both failing a non-coordinator or coordinator VM. We see that failure of noncoordinator VM takes shorter time to resume normal operation and has shorter standard deviation. This is reasonable since coordinator failure detection has a longer period and requires more data to be transferred around VMs. Thus, we expect more time needed when a coordinator vm failed.