CS 425 MP4 Report: IDunno, a Distributed Learning Cluster

Group32 Dingsen Shi(dingsen2) Ruipeng Min(rmin3)

Design:

Based on the former MPs, we build a distributed Machine Learning (ML) platform called IDunno. In our system, we load two pre-trained model (resnet & lenet) in SDFS and classify test images from cat/dog & MNIST Dataset by reading queries from SDFS and output results to it.

For the fair-time inference phase, coordinator process fetches the query of job type (Job1/Job2) and batch size (=10) from client-side to calculate query rate every 10 seconds. Specifically, our system will count the total processed queries number of each job during a fixed period as its query processing rate. After comparing the query rate of two jobs, adding the worker VM numbers for the job with low rate and reducing the workers for the job with high rate to keep query processing rates of different jobs within 20% or 10% of each other.

For the fault-tolerance phase, our system can detect the failed processes based on MP2 and evenly allocate the queries haven't processed on the failed processes to other alive workers processing the same jobs. Since the query rate and query count of the two jobs will change significantly after a failure/leave/rejoin, our system will immediately check the query rate and reassign workers for two jobs. Our system also has a standby coordinator to deal with the fault of coordinator process. Same as coordinator, the standby coordinator stores all the commands from client process, metadata information of SDFS, results of inference phase, and so on. However, it will not output result data to client-side unless the coordinator fails. Before the coordinator being replaced, the connection between client and coordinator will be locked to make sure no more new jobs added. Since the query rate and query count of the two jobs will also change significantly after the failure of coordinator, our system will immediately check the query rate and reassign workers for two jobs to keep balance.

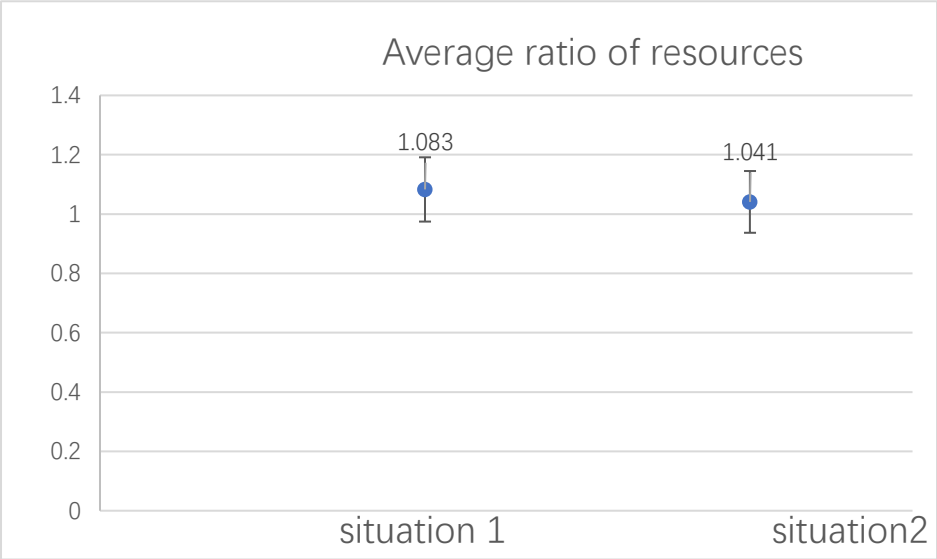
Measurements：

1. Fair-Time Inference:

1.1 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?

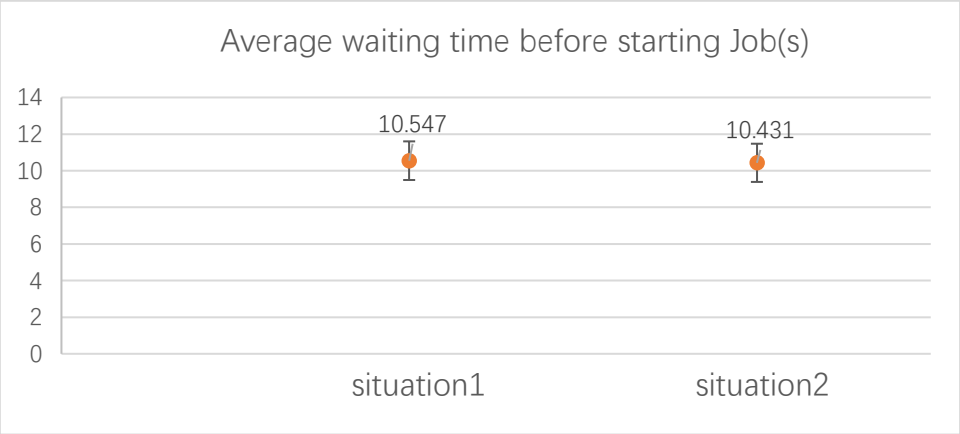
Assumption: Job1 is cat/dog dataset classified by Resnet and Job2 is MNIST dataset processed by Lenet. The ratio of resources is calculated as the average $\frac{\text{workers for Job1}}{\text{workers for Job2}}$ after rebalance for 10 times.

	Average workers for Job1(number)	Average workers for Job2(number)	Average ratio of resources	Standard deviation
Added Job1 while Job2 processing	5.2	4.8	1.083	0.051
Added Job2 while Job1 processing	5.1	4.9	1.041	0.026



1.2 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?

	Average waiting time before starting Job1(s)	Average waiting time before starting Job2(s)	Standard deviation
Added Job1 while Job2 processing	10.547	-	0.523
Added Job2 while Job1 processing	-	10.431	0.389



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

	Average rebalance time(s)	Standard deviation
one non-coordinator VM fails	22.571	2.289

3. After failure of the coordinator, how long does the cluster take to resume “normal” operation (for inference case)? Unless otherwise mentioned all experiments are with multiple jobs in the cluster.

	Average rebalance time(s)	Standard deviation
coordinator VM fails	26.832	3.731

Discussion:

1. Fair-Time Inference:

1.1 As we expected, the end result of two situations when a job is added to the cluster is almost the same. That is, no matter adding Job1 when Job2 is processing or adding Job2 when Job1 is processing, our system can rebalance equally to meet fair-time inference.

1.2 In our IDunno, query rate for different job is normally calculated and checked every 10 seconds. When a new job joins, our system can start comparing and adjusting the two query rates immediately after the next check period (10s). Thus, normally it takes around 10 seconds to start the new job after joining.

2. In the system, any failure of VM will immediately leads the coordinator to reassign queries after one check period (10s) and detect the query rate difference after one more check period (10s). Thus, it makes sense that the average rebalance time is a little bit longer than 20 seconds.

3. Based on our design, the standby coordinator and the original coordinator are running simultaneously and independently. The original coordinator fails will only cause the standby coordinator to start sending result messages to the client. So, the two query rates will be output to client-side after the next check period (10s). Then the standby coordinator will reassign queries and show the balance query rate at the following check period (10s). Thus, it makes sense that the average rebalance time is a little bit longer than 20 seconds.