CS425 MP4 Report

Group16: Wen-Hsing Huang(whhuang4), Zong-Yun Li(zyl2)

Design:

Overview:

We construct our Distributed Learning Cluster, IDunno, in an HDFS-style architecture and efficient job scheduling, allowing our members in the system remotely manage jobs to complete scalable machine learning tasks.
In our system, there are normal nodes, introducer, and coordinator (master).
Every machine is the normal node, the introducer is set up manually or by default, and the coordinator is selected by the election mechanism.

System Mechanisms:

- **Log Query**: user can request logs from all machines, it is useful for the user to debug.
- **Membership**: user can join or leave the system by sending a request to the introducer, adopting a ring-based linked list to implement membership, hence its insert and delete operations are efficient.
- **Failure Detection**: the system can detect node failure and update membership quickly, adopting ring-backbone SWIM-style failure detection.
- **Election**: when the system initiates or detects the coordinator down, it invokes a ring-based election to select the new coordinator.
- **Simple Distributed File System**: user can put or get the versioned file via our SDFS, it adopts SCP to directly transfer files between machines.
    - get addresses list from coordinator, and then transfer data one by one like a chain.
- **Job Scheduling**: contains two phases, and supports multiple jobs models at the same time and queue jobs (not limited to two models). Scheduler would balance resources to make the query rate similar.
- **Recovery**: the system support introducer and coordinator recovery. For the introducer, the user could use “update-intro” command to set a new introducer that allows machines to join and leave. For the coordinator, the system periodically stores the metadata of the coordinator in SDFS, this stored files would be used to restore the coordinator.
- **Scalable Machine Learning module**: we design an easy-to-scale module for extending supported tasks of the system. There is a hub function to easily switch models to train and inference.

Job Scheduling Design:

Our design has three parts:

1. **Job Scheduler**: handles each model has one queue containing jobs. Also, allocate resources to balance the query rate.
2. **Job dispatcher**: for each job, the coordinator would read file query by query and send it to resources processing.
3. **Job Reply Receiver**: this function collected processed result from assigned machines, organize data and conduct statistical analysis.

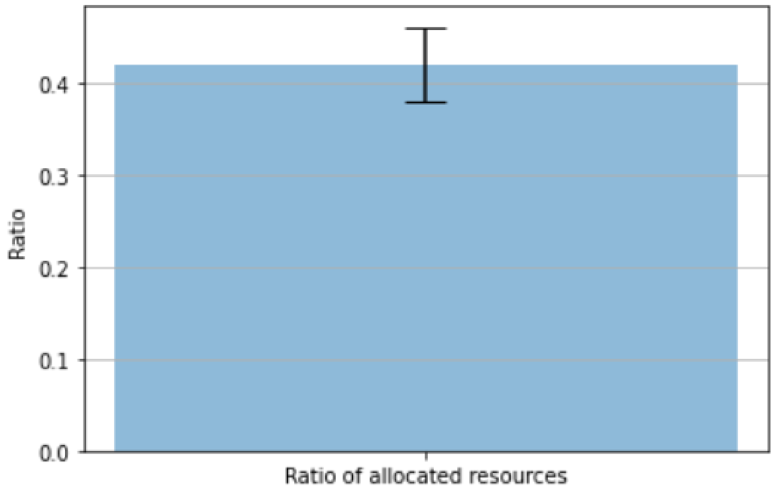
Measurement:

(1) Fair-Time Inference:

(1a) When a job is added to the cluster (1 to 2 jobs), what is the ratio of resources that IDunno decides on across jobs to meet fair-time inference?

Data points	1	2	3	4	5
Resources ratio allocated to added job	40%	40%	50%	40%	40%

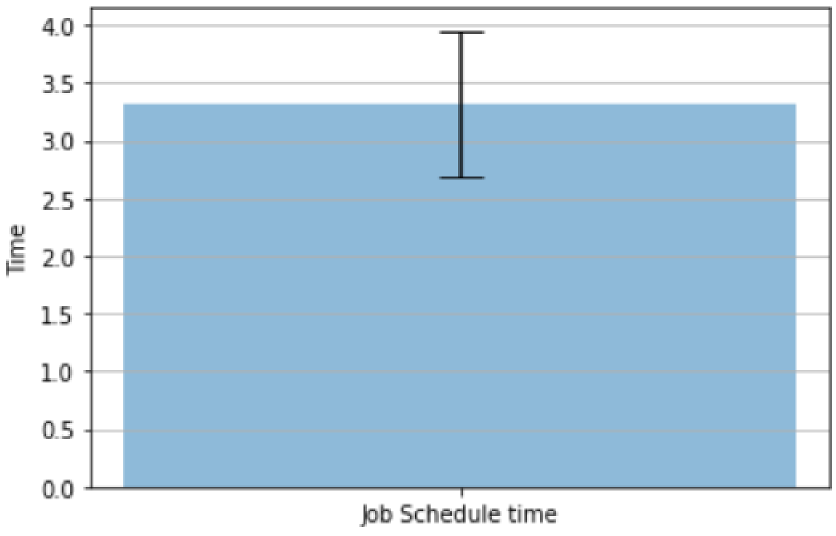
Ratio	5 data points
Average	42%
standard deviations	0.039



(1b) When a job is added to the cluster (1 to 2 jobs), how much time does the cluster take to start executing queries of the second job?

Data points	1	2	3	4	5
Time (s)	3.2902	3.2555	4.5030	2.7853	2.7647

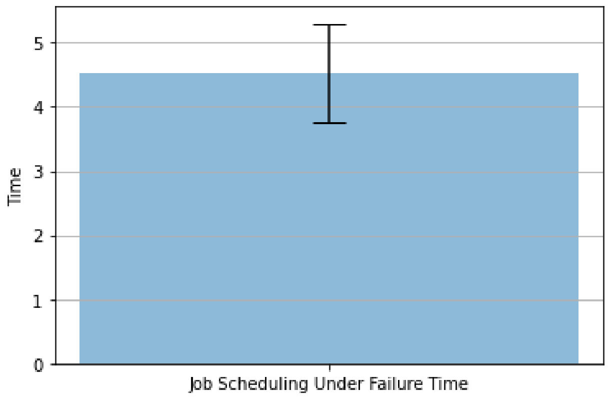
Time (s)	5 data points
Average	3.3197
standard deviations	0.6322



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

Data points	1	2	3	4	5
Time (s)	4.2615	5.9801	4.3597	3.6465	4.3583

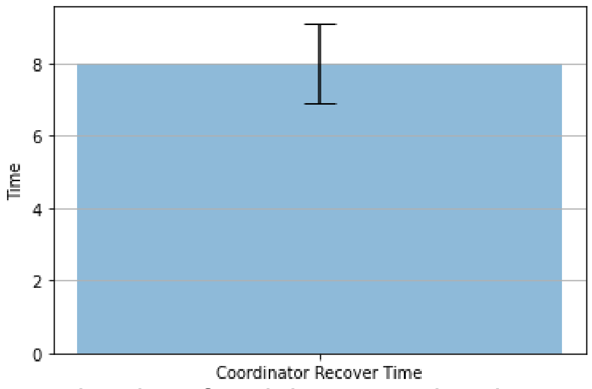
Time (s)	5 data points
Average	4.5212
standard deviations	0.7763



(3) After failure of the Coordinator, how long does the cluster take to 8 resume "normal" operation (for inference case)?

Data points	1	2	3	4	5
Time (s)	8.4048	7.4321	9.5772	8.3205	6.2734

Time (s)	5 data points
Average	8.0016
standard deviations	1.1007



Discussion:
(1a) Two of our models are image processing tasks, therefore it is expectable that processing time of two jobs is similar. As a result, when our system reallocates resources to balance query time, the resources ratio of the two jobs is similar.
(1b) This data is related to reschedule period of job_scheduler. When the job is added to our system and the timer becomes 0, our scheduler would allocate resources to it and create a job_dispatch thread that starts executing queries.
(2) In our implementation, this data is related to our failure_detection and job_scheduler. Our scheduler uses membership to know how many resources can use. Therefore, the system must have to detect failure to restore "normal". Besides, I think failure detection affects more because its is required more time than rescheduling, hence the results are higher than (1b).
(3) In this case, we can observe that it took the most time. I think it is reasonable because the recovery of coordinator requires failure detection, ring-base election, and metadata file I/O.