# MP4 Report

**Gan Yao(ganyao2), Linxi Li(linxili2)**

In this MP we implemented a fault-tolerant Distributed Machine Learning platform that is able to simultaneously handle two jobs at most and being able to dynamically balance the resources by detecting the real-time query rate of the two jobs.

## Design

We chose our models based on two of the most popular topics in Machine Learning: NLP and CV. Our NLP task was handled by a Transformer model, it is able to translate English into Celtic language. This is a relatively heavy model compared to the capability of our VMs. Our vision task was image classification handled by Vision Transformer. We deployed these two models on our VMs by tuning pretrained models and set up the corresponding environments for them.

Our system has a one-coordinator structure, meaning there are one Coordinator and many Workers. The Coordinator is responsible for instructing workers to initialize, start, end tasks, and of course balancing resource across workers. Whenever a client submits a job to Coordinator, it will create and initialize a Tracker object that will be solely responsible for monitoring that job in later phases. When client requests arrives for later phases, e.g. starting training, starting inference, Tracker objects will be in charge for communicating with Workers.

As for fair-time inference, there is another object on Coordinator called Scheduler, which will be created when there are more than one jobs submitted to Coordinator. Before inference phase starts, Scheduler will assign machines to two Jobs based on some prior expectation of corresponding model type. For instance, if ratio of models' prior expected inference speed of job A and job B is 1/2, then the ratio of machines assigned to job A and job B is 2/1. After inference phase starts, Scheduler will constantly monitor the true query rate, number of queries processed averaged over the past ten seconds, of two jobs and dynamically balance the resource assignment. It will always try to make the query rates of two jobs as close as possible.

As always, there is a fault tolerance mechanism built in our system. Whenever a failure of Worker is detected by membership service, Coordinator will route failure messages to Trackers, who then will check whether the failed Worker is a member of its Job. If it is, Tracker will remove that failed machine from its member list and make sure to reassign that Worker's task to other active Workers later. What if Tracker finds a Worker is irresponsive before the membership service? Tracker will ping that Worker again to make sure if it is dead. If that Worker failed to respond again, then Tracker will mark that Worker as 'dead' and wait for membership service's message to remove that Worker.

To handle Coordinator's failure, we have a stand-by Coordinator always ready to take charge. The stand-by Coordinator has exactly same function as Coordinator, except that it will not send out any instruction before it is activated. Before activated, the stand-by Coordinator only listen to all communications happen inside the system, making it always aware of the latest status of the whole system. Whenever the Coordinator fails, it will be activated an function as a regular Coordinator.

**Past MP Use**

Our IDunno system is built upon past MPs. We use MP1 to query distributed log files for debugging. We use MP2 membership service for failure detection and mapping machine names to IP addresses. MP3 SDFS is an incredibly important part of IDunno. We need SDFS to share files among clients, Coordinator, and Workers. The distribution of our jobs requires a lot of writing and reading among VMs, so having a stable and reliable SDFS system is really important for MP4.

**Findings and discussions**

1a) Fair-Time Inference

As mentioned before, whenever there are more than one Jobs submitted to the Coordinator, a Scheduler object will be created and assign resources to jobs according to the prior expectation of inference speed. So if the two jobs are of the same model, which of course have the same expected inference speed, then they will each be assigned half of total resources. And when two Jobs have ratio of expected inference speed of 1/2, the ratio of assigned resources will be 2/1. Note that these expected inference speeds are just rough estimations so we never expect the system to be stable in the first place. Most work of resource balance will be done when the inference phase proceed and Scheduler object can observe the true query rate.

1b) – 3) Other Measurements

| Trials\Field | Non-Coordinator Failure Recover | Job1->2 time to start execute | Recovery of Coordinator |
|---|---|---|---|
| **1** | 3.21s | 33.10s | 3.65s |
| **2** | 2.75s | 30.46s | 2.45s |
| **3** | 3.33s | 27.58s | 2.98s |
| **4** | 3.38s | 31.25s | 3.13s |
| **5** | 3.65s | 30.18s | 4.15s |
| **AVG** | 3.264 | 30.514 | 3.272 |
| **STD** | 0.32936 | 1.99699 | 0.65094 |

In general, it will not usually take more than 4 seconds to recover the failure of non-coordinator and worker to recover. However, since when Job 2 was added to job1, it requires a lot of writing to the SDFS and also the load of the Computer Vision model, it usually take more than 30s to load the model and ready to work. I think the data we got are all within the expectation of us, since we do understand it requires some time to split the large file into smaller ones, write them into SDFS, and actually load the model. And there will be a certain level of information exchange between the workers and the coordinators when there are failure happening, but this process does not require write and read, so this is relatively shorter. We adjusted the parameter of the machine learning models to make the query rate to be as close as possible. However, when job is close to the end, the resources distribution would usually incline more to the visual model, usually the eventual result is often 5:3(Right before the NLP job ends, since after this its VMs will be available and will then be distributed to another job). We think because it is harder for VMs to perform Visual jobs, since it actually requires a lot of calculation between the image matrix data, also since we only provide URLs for visual jobs, it will take some time for VM to download images. This will be significantly impacted by the network conditions.