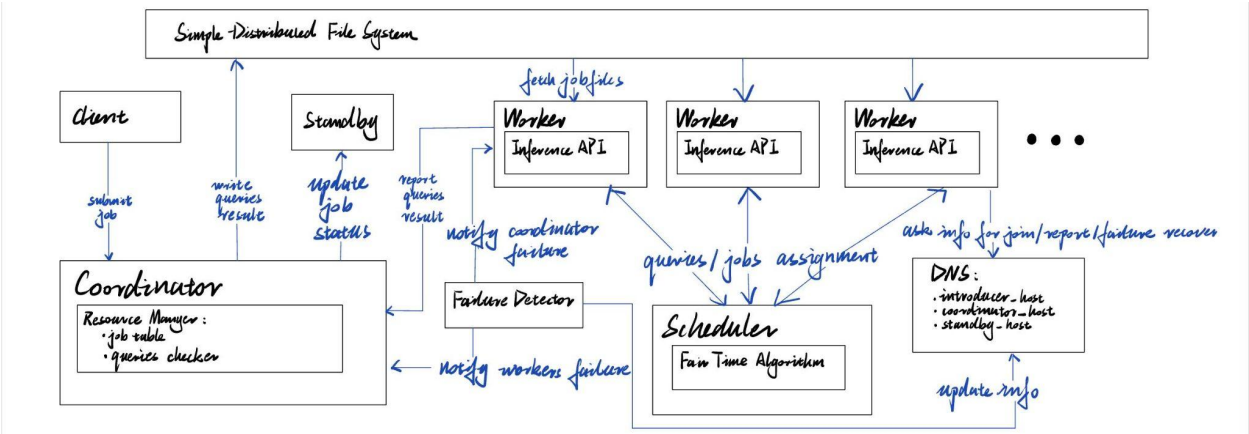
Peiran Wang(peiranw3) & Hanbo Guo(hanbog2)

Architecture and Design



1) Definition:

- Query: the smallest input to a model (api): an image, a line of text, or if api permits, a batch of images. Each query corresponds to one file in SDFS.
    - States: IDLE -> Scheduled -> Completed
- Job: a collection of queries for one model.
- Query rate: number of queries completed per time interval.
- Processing time: time spent on api.

2) Overview:

The IDunno system contains multiple **Nodes**. Each node is either a coordinator or a worker. The coordinator communicates with the client for job updates, schedules queries to workers, handles query result output, and synchronizes query states with other coordinators (only one standby in IDunno). Each worker repeatedly asks the scheduler for new queries to work with and reports back to the coordinator upon completion.

Apart from **Nodes**, the IDunno system assumes a persistent **DNS** server, which tells **Nodes** who is the current coordinator. Every failure in IDunno is reported to DNS, and DNS updates the coordinator information if necessary.

3) Design:

Coordinator: the coordinator bookeeps information of jobs in a **JobTable**. With each (batch of) query(ies) completed, the coordinator writes to a result file in SDFS. With each new job submitted or query completion, the coordinator syncs this information with the standby coordinator. It is important to note that, by design, the record of **completed queries** in the standby coordinator's JobTable is identical to the **result file in SDFS**.

Scheduler: every time a worker requests queries to work on, the coordinator calls **scheduler** to decide which job and which queries to feed the worker.

Worker failure: each worker only has few (batch size) queries, which are marked as "idle" when the coordinator detects the failure. The scheduler would schedule these queries to future requests from other workers.

Coordinator failure: each worker repeatedly **requests jobs** from the coordinator. If the coordinator fails-stop, the worker detects this failure and asks **DNS** for the standby coordinator address. The standby coordinator knows what jobs have been completed.

Data and discussion

1a: ratio of resources across jobs

Since we will have only two jobs at the same time, let t_1, t_2 be the time that job1 and job2 needs to process one images, b_1, b_2 be the batch sizes of job1 and job2. And let x, y be the number of VM we should give to job1 and job2. Assume we have z VM to assign. We have following equations to help us determine the ratio of resources that we should give to different jobs:

$x/(t_1 b_1) - y/(t_2 b_2) <= max(x/(t_1 b_1), y/(t_2 b_2)) * 0.2$

$x/(t_1 b_1) - y/(t_2 b_2) >= max(x/(t_1 b_1), y/(t_2 b_2)) * (-0.2)$

$x + y = z$ (where x and y should be integer)

Using the models(jobs) we choose as examples. Our model1 needs approximately 1 sec to process 1 image, and our model2 needs approximately 0.25 sec to process 1 image. By setting both batch sizes as 1, z=10, and solving the equations we have above, we can have the ratio 8:2 of resources that we need to assign for job1 and job2.

1b: time elapsed between 2nd job starts and 2nd job being scheduled

Setup: 10 machines, job1 average processing time 0.24 (SD 0.12) seconds, job2 average processing time 1.09 (SD 0.20) seconds. Job 1 batch size = 4, job 2 batch size = 1.

Result: 1.31 (SD 0.48) seconds.

Process: start job 1, wait for 10 seconds, start job 2. Calculate the difference between job2 start time and its earliest query schedule time.

Explanation: When the second job is added, it has a query rate = 0. Thus the scheduler would assign the second job to the next available worker (or workers). Thus, the time elapsed before the 2nd job starts execution is smaller than the (batch) query completion time (usually equals to batch size * processing time) of one batch of queries. In our setup, each batch of job1 takes ~1.0 seconds to complete, along with ~0.3 seconds overhead in writing the result to SDFS and syncing with the standby coordinator. This also explains the high standard deviation: sometimes a batch of queries completed right after job2 started.

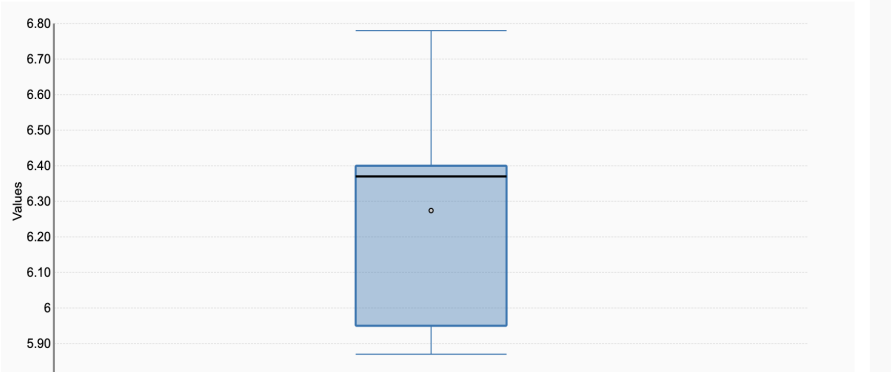
2: how long does the cluster resume “normal” operation with node failure

We define “normal” operation as: after a worker failure, the time difference between two nodes is getting back to 20% of each other and becoming stable again.

Setup: 10 machines, job1 average processing time 0.24 (SD 0.12) seconds, job2 average processing time 1.09 (SD 0.20) seconds. Job 1 batch size = 1, job 2 batch size = 1.

Process: start job 1, wait for 10 seconds, start job 2. Calculate the difference between job2 start time and its earliest query schedule time.

Data: We have 5 trials, and the times we need to back “normal” are: 6.4,5.87,6.37,6.78,5.95 s, the average is 6.2724s and the SD is 0.3706. (plot is in the next page)



Explanation: When a worker fails, it might be scheduled (by the coordinator) with some queries. The FD needs ~ 5 seconds to detect and report this failure to the coordinator, which could restart these queries. Thus, it is expected to take 6 seconds to resume the “normal operation”, which matches the experiment result.

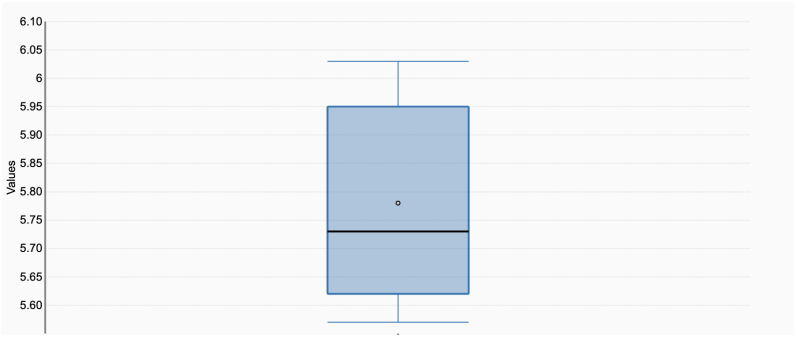
3: how long does the cluster resume “normal” operation with coordinator failure

We define “normal” operation as: the standby coordinator kicks in and resumes from failed coordinator’s states (JobTable).

Setup: 10 machines, job1 average processing time 0.24 (SD 0.12) seconds, job2 average processing time 1.09 (SD 0.20) seconds. Job 1 batch size = 1, job 2 batch size = 1.

Process: start job 1, wait for 10 seconds, start job 2. Calculate the difference between job2 start time and its earliest query schedule time.

Data: We have 5 trials, and the times we need to back “normal” are: 5.73,5.95,6.03,5.57,5.62s, the average is 5.78 and the SD is 0.2022.



Explanation: Upon failure of coordinator, the FD needs ~ 5 seconds to detect and propagate this failure to DNS and worker nodes. When a worker receives this failure, it asks DNS for the new coordinator address, which takes marginal time. So the expected time taken to resume “normal” operation is < 6 seconds, which is consistent with the experiment result.