Network Assignment 2

Wang Chenyu & Cai Xuemeng

Job Schedule Policy Implementation

In order to make good job schedule decision, the following things are important:

1. What is the capacity of each server?
2. What is the current workload of each server?

Because the capacity is hidden, in order to infer it we must try it first and use the JCT of the first job that are sent to the server to calculate it. When a server S is handling a single job with file size K, then after it is completed we can use the following formula to infer the capacity C: $C = K/S$. Therefore in the beginning the policy should be:

I: Send one job to each server.

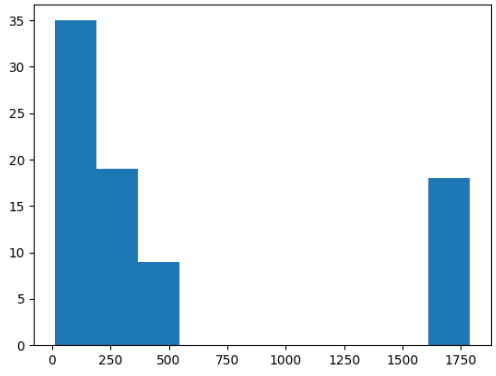
If there are still some jobs remaining after we assigned each server a job and we haven't get any finished job from server, I came up with 2 ideas for the following operations:

1. Repeat <I>.
2. Store these jobs in a queue until there are some jobs finished, then we can calculate the corresponding server capacity and then assign it to the server with the maximum capacity.

Solution 1 can promise that after job scheduler received a job from the client, the job will be forwarded to client without any delay, having no delay will for sure decrease the JCT of the job, however, it have the following drawbacks:

1. Slow down the job that is being processed by the server.
2. Influence the accuracy of the capacity calculation.
3. According to file config_server, we can find that the capacity of servers may have very huge differences, eg: server 8 and 9 only have capacity 1 while server 0 have capacity 50. During stage <I>, we have no choice but to send one job to each of the server, including 'Super slow' servers like server 8 and 9, if we just blindly(because we haven't infer there capacity yet) assign jobs again, we cannot avoid sending another file to server 8 and 9 which will have a huge influence of 90% JCT.

The drawback is very obvious if we use config_server, config_client1:



Because the job size is too big (Then it will take a quite long time to figure out the capacity), we send nearly 20 files to 'Slow servers' before we figure out their capacity.

Therefore I decided that the policy for the second phase should be:

II: Store these jobs in a queue until there are some jobs finished until some server finished the first job.

Then, after the all of the first round's files are finished, we can infer the capacity of all of the servers (If the file size is unknown, we assign the default file size to it.) Then we have stats to schedule job properly. Now there are also two ideas:

1. Calculate 'What server have the least amount of workload unfinished compared to his capacity', what is, who have the smallest $UnfinishedWorkload/Capacity$, that means that if no other job comes in, that server should finish his work fastest, and assign the job to it. (Here we allow parallel run)
2. Only assign job to server that is not working on any jobs. Find all spare servers and assign the work to the one with highest capacity.

1 is the initial idea that I came out with. I thought that allow multiple run will eliminate the delay of the job, however, I found that if I do so then the drawbacks are as follows:

1. "Fast servers" will run many many work at the same time, this will cause some jobs that entered early cannot leave early, thus increase JCT significantly.
2. Sometimes the standard $UnfinishedWorkload/Capacity$ is not perfect, cannot evaluate the server status well

Therefore I decided that the policy for the Third phase should be:

III: Only assign job to server that is not working on any jobs. Find all spare servers and assign the work to the one with highest capacity. If all servers are busy, put the job in the queue.

It turns out that this works quite good and the whole schedule policy's blue print is complete.

Possible Improvements:

Under the current design, I set a DEFAULT_FILESIZE variable that will be used to calculate the capacity of the server if the file size is unknown. However if the value of it can be dynamic, eg: the average file size of all files with known file sizes that we received, then it will be quite flexible.