**Introduction**

We have already finished stage 1 and have a solid understanding on how the ds-simulator work. The major task of stage 2 is to design and implement a new scheduling algorithm, and use the ds-sim and various config files to evaluate our algorithm.

**Problem Definition**

Our system is as same as stage 1. But this time, we have to focus on how to schedule servers. There is a great number of servers in this system, their memory size, disk size and CPU cores are in different numbers. Each time our client receives a new job from simulator, we have to determine which server should finish this job. Our target is to design a new scheduling algorithm for this problem.

**Algorithm Description**

We have learned three scheduling algorithms, first fit, best fit and worst fit. They are also the baseline for stage 2. I designed a new scheduling algorithm, named MJF(Minimum Jobs Fit). The idea is really simple, that is, always assign jobs to the server with the least current tasks. Comparing to baseline algorithms, MJF could strongly decrease the average waiting time and turnaround time.

Turnaround is really import in modern operating systems and distributed systems. For example, In a multi-user operating system, the rotation time can seriously affect the user experience.

**Implementation**

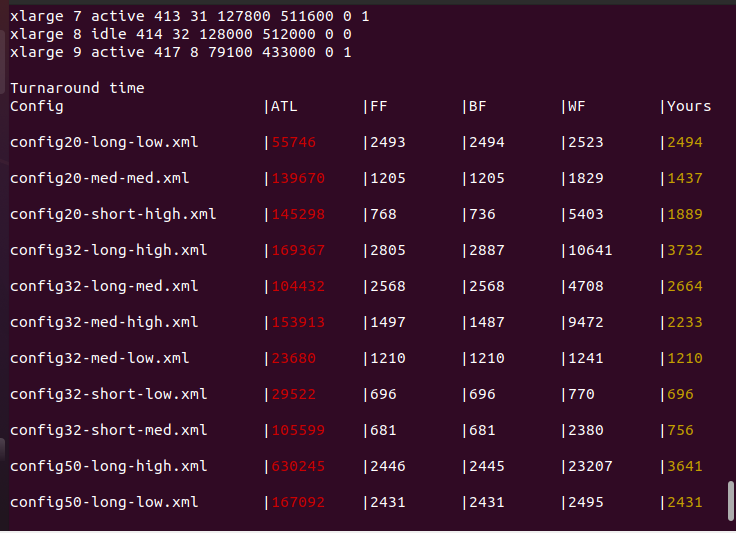
Each time client accepts a new job, it first queries all capable servers. Then the client traverses all the servers, find the one with least jobs, and assign the job to it.

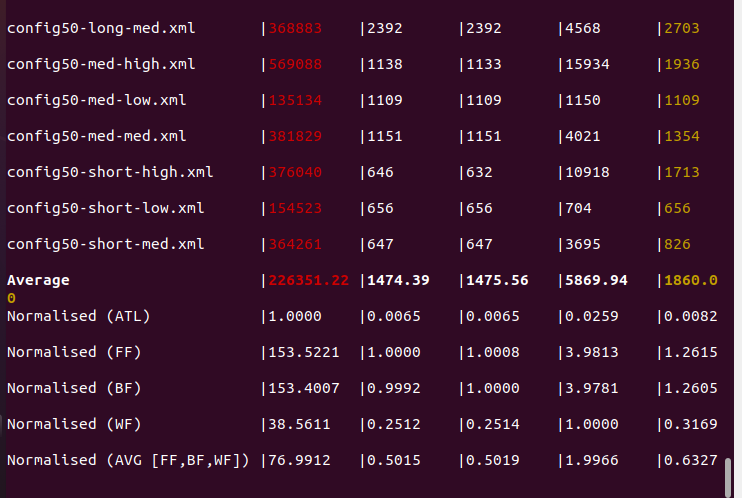
Below is the core code pieces of this algorithm.

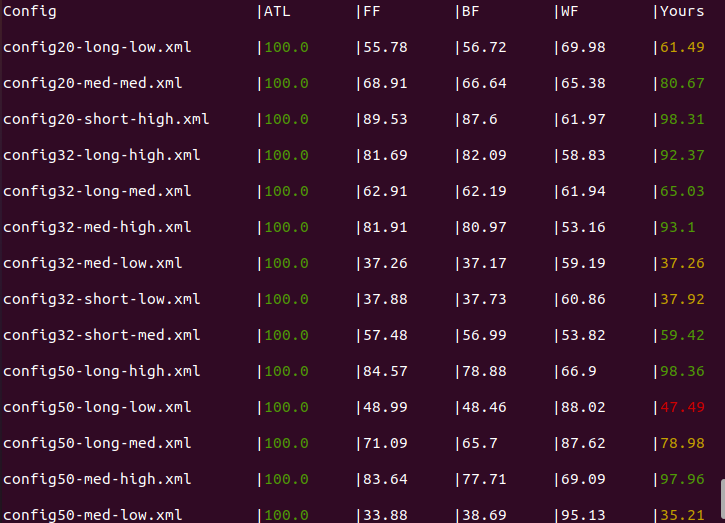
**for** (**int** i = 0; i < **serverList**.size(); i++) {  
 Server s = **serverList**.get(i);  
 **if** (s.**waitingJobs** + s.**runningJobs** < maxJobs) {  
 maxJobs = s.**waitingJobs** + s.**runningJobs**;  
 serverIdx = i;  
 }  
}

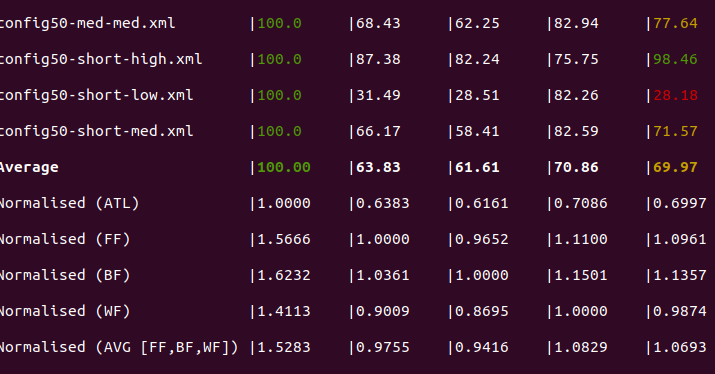
**Evaluation**

As what we expect, this algorithm has a good performance on reducing turnaround time. The following pictures are the test results for my algorithm.

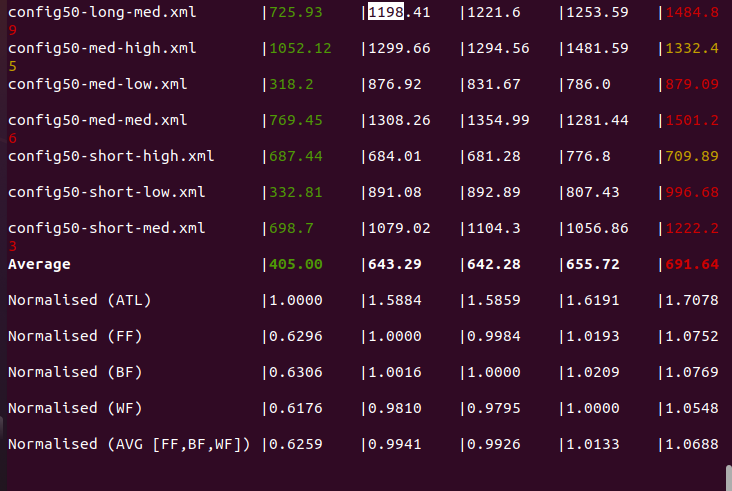












**Conclusion**

As we can see, comparing to three baseline algorithms, our MMF algorithm take an advantage on average waiting time and turnaround time. This algorithm is good for situations that every job should be processed as quick as possible.

Reference