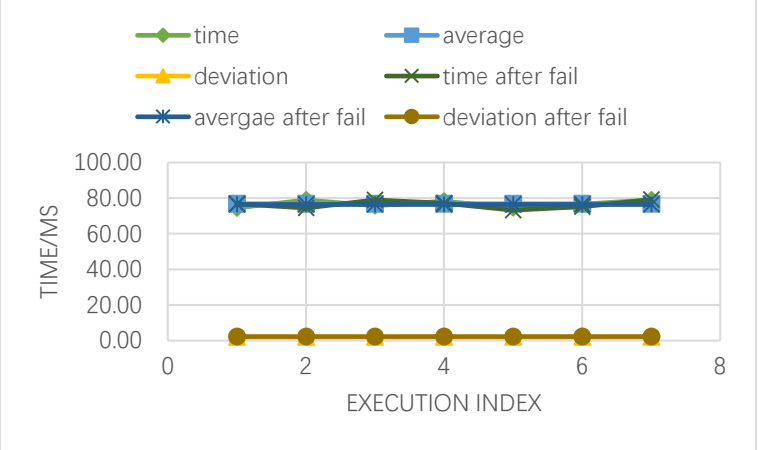
Our design mainly contains 2 parts: server/client and test.

The server/client is for running grep on log files and get results. It uses Golang language. The server listens to port 3333 and tries to connect to client using TCP connection which is more reliable which we think is important in this situation. On the server, it has a service called GrepService which is for grepping on local log files. The GrepService uses the command grep in Linux. It can search for specific pattern and the regular expression. The result is sent back then. The server is always on and stuck in a infinite loop, so it can connects to multiple clients at the same time. The client part has the IP addresses of all VMs. When we run the client part, the main function runs ten different go routines which includes making connection to all servers including itself and using the RPC function of Go language to request for service on the server side. This design could make the program faster because we can have all the servers running grep at the same time, like parallel programming. It is fault-tolerant to TCP connection by catching the connection failure and time out failure by using a timer and a channel of notification (although we have not met this situation).

The test unit is for generating a log file and testing the result against the correct answer we already know. It uses Python to generate random lines based on a given regular expression and some specific lines then write to test.log file. The server part has the test service which calls the Python function and then runs grep on the test log files. The client_test file utilizes Commtest function in client to test the functionality of the whole system automatically and compare the test result with the true one.

The following two figures shows the consuming time when we execute pattern and regular expression. The consumption time for executing pattern is much shorter than that of a regular expression because the grep execution for regular expression need more time than execution for pattern. The standard deviations of them are very small which means each value is closed to the average. Then we fails six machines and four servers still remain working. The execution time doesn't change a lot and standard deviation increases a little which shows some scalability.

Execute grep -c Mozi Average is 76.82678ms Standard deviation is 1.967242 Average after failing six machines is 76.42163ms Standard deviation after failing six machines is 2.249907	
Execute grep -c -E '^[0-9]*[a-z]{5}' Average is 368.1924ms Standard deviation is 4.489011 Average after failing six machines is 363.523044ms Standard deviation after failing six machines is 10.17819