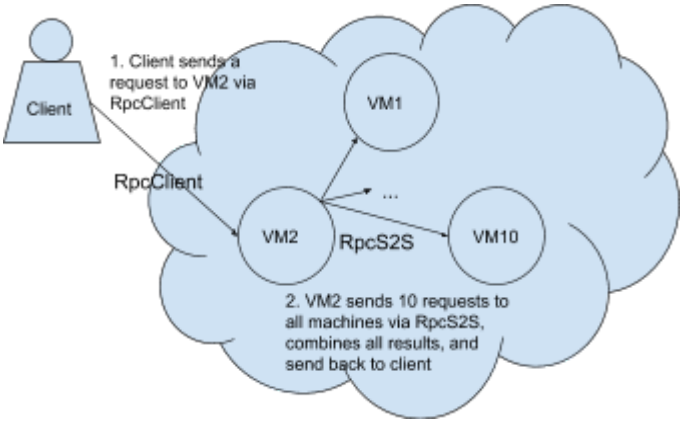
CS425/ECE428 MP1 Report

Yen-Chieh Sung (ycsung2), Chih-Shin Wang (cswang6)

Design

- Language/Communication: Golang with RPC
- Server structure
    - Design principle: we believe that client should only send one request to one machine, and no matter which machine it sends, the client should always get all machines' result. Saying that the client sends a request to VM2, then VM2 should be responsible for calculating its result, and sending/collecting other 9 machines result. Thus, there are 2 types of RPC, RpcClient and RpcS2S.

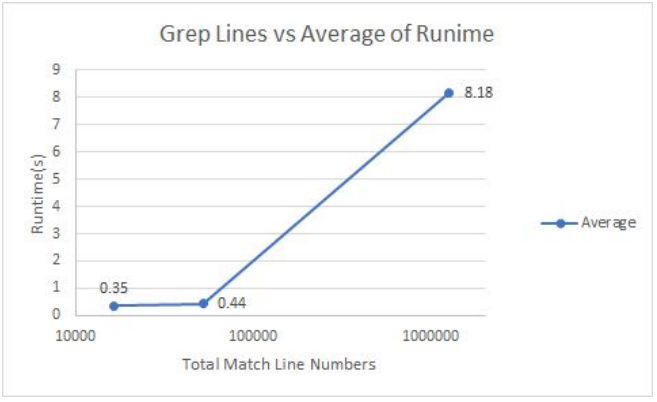


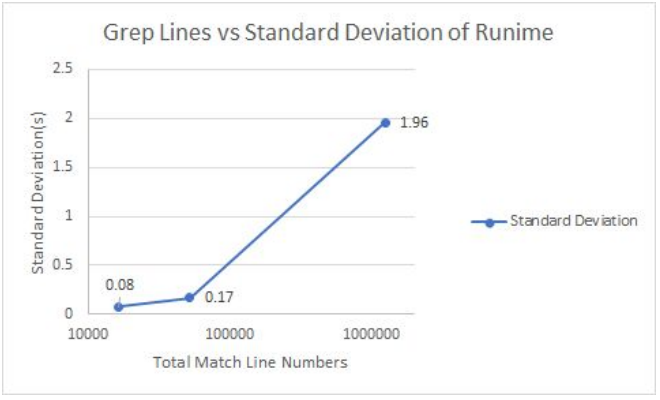
    - RpcClient: it is the API for client. When using this type of APIs, client will get all 10 machines' result by sending only one request to one arbitrary machine.
    - RpcS2S (RPC server to server): it is the RPC for communications between server.

Unit Test

In our unit test, we test RPC server to server (S2S) functionality. The test function first sends 3 hard-coded testing log file to 3 machines respectively, and sends S2S grep request to those machines. Finally the response will be compared with the hard-coded answers. We do not test grep function since it should be tested by the library developer.

Plot & Discussion





We have tried different patterns with different match line numbers (from small to large: ”^[0-9]*[a-z]{5}”, "12:[0-9]{2}:[0-9]{2}", and ".*"). The plot shows that there is a portion of non-parallelizable runtime since the runtime saturated around 0.35s. Also, The runtime fluctuated widely when the transmission data is large. For example, when trasmitting whole logs, the time varys from 5 to 10 seconds.