CS 425 MP1

Jiayi Luo (jiayil5)
Zhoutong Jiang (zjiang30)

For this MP1, we select Go as the standard language and net/rpc as the key package to use. The reason for using rpc package is its capacity to transmit large file stream without losing information.

1. Design:

1.1 Client

The client can receive the grep command from users and distribute the command to different servers. For each server, a separate file stream thread is built through tcp using net/rpc package. After sending the grep command, the server will receive the command from client and send back the grep results stream to client.

1.2 Server

The server is a threaded TCP server, listening to a predefined port (8080). When a call is received on the server, the server will grep on the local log file and return the matched lines to the client side.

2. Test:

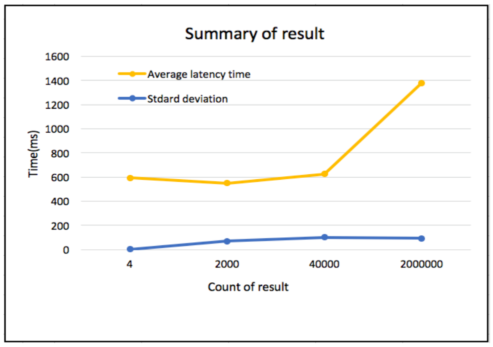
2.1 Log files generation

We have two different log generation patterns:

(1) As a unit test, we predefine one single pattern in each individual log file for each VM, as such, the returned count can be checked to ensure the basic functionality of our system.

(2) In order to generate more heterogenous pattern to test the correctness of our system and evaluate the performance of our system, we further created a second pattern of log files ourselves. In this test, the same log file is used across each VM server. Each log file is roughly 60M as required in the documentation. We generate the log file with 4 different patterns (strings as a key), each with (1, 500, 10000, 500000) frequency. Thus, we can test our system by comparing the returned count frequency and the anticipated value.

3. Performance Evaluation:



After verifying the correctness of our system by Test (1). We conduct experiements on the log file generated as Test (2) mentioned above on 4 VMs as servers, and query from 1 client VM. From the figure, firstly we can see that the count of results return matched our expectation, indicating that no data are lost during the process, which is the advanatage of TCP. The latency time is demonstrated as follows. we could observe that the latency time increase with the number of results returned (or the amount of data need to be transfered), which is intuitive. On the other hand, the performance of our system is relatively stable, which yields a low standard deviation of latency time.