baohez2(Baohe Zhang) and kechenl3(Kechen Lu)

CS425 MP1 Report

Part I. System Design

We implement a distributed log query system, which can query distributed log files on multiple machines from any one of those machines. The system follows the client-server architecture. To query distributed log files, the client concurrently sends requests to all machines' servers. Once received the requests, the servers serve client's request by executing the "grep" command locally and responding the query results of their local log files to the client simultaneously. Finally, the client receives these query results and print them to the console with a human-readable format, shown as Fig. 1 below. The system is implemented in Go. The system relies on Go's net package to implement data transmission. Go's concurrency feature enables the system to implement concurrent log query easily. We apply Go's sync package to ensure each server's response can be printed synchronously. We create a config file to store all machine's network addresses and machine-dependent information. We also provide some shell scripts to automatically start or terminate the servers.

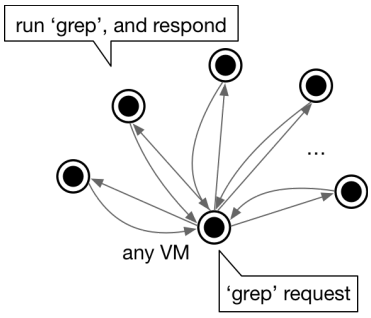

Fig.1 distributed grep client/server design

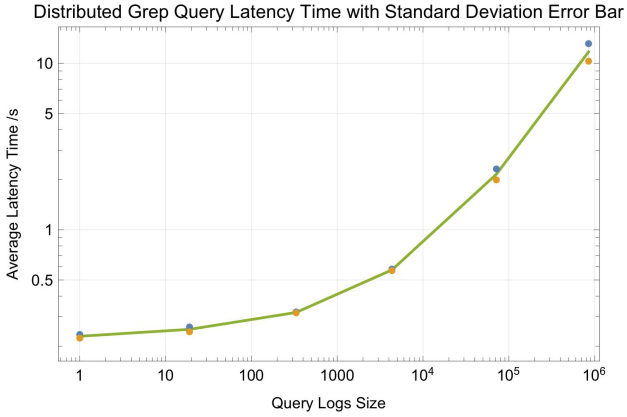

Fig.2 Grep Query Latency in Different Frequency Patterns

Part II. Unit Tests

We applied unit testing to verify the system's correctness, testing different types of pattern by using our distributed grep with log files distributed on each machine, and comparing results of running a grep command locally with all log files stored on the local machine. We verify the identity of the query results of both methods by using the "diff" command in Unix/Linux. If they are identical, then the test passes.

Part III. Performance

The query latency tests were conducted in 4 VMs with around 100MB logs on each machine.We tested our distributed grep client on one machine with different frequency patterns, from only one match to 848895 matches, with log-log scaling plotted as Fig.2, with standard deviation error bar on each point. As the plot shows, if querying infrequent or some normal pattern(query logs size under 10^4 entries), the latency increases slowly(within a second). But if we have frequent pattern query, the latency surges. Meanwhile, we can figure out standard deviation of query latency increase as the matched logs become larger. The results are reasonable because the bottleneck may reside in the network bandwidth and latency with terminal standard output(buffer limit).