Summary:

Overall, this program is designed to perform distributed log query across multiple machines (10) using RPC in Golang. The user can input grep commands, and the grogram will query the logs on all connected machines and aggregate results:

Design:

We implemented the distributed log querier using an RPC server and client pair running concurrently. The server sets up an RPC server that listens for incoming connections. When a connection is accepted, it spawns a new goroutine to handle the RPC connection. The client continuously reads grep commands from the user, connects to every client (once) and queries connected machines for log entries that match the grep patterns. The client then aggregates and prints the results from all servers.

Function Overview:

GetLogQueryResult(): Applies grep on a machine's log file and returns results.

listener(): Sets up an RPC server to listen for and handle incoming connections.

readGrepCommand(): Reads and validates the grep command input from the user.

closeConnections(): Closes all active RPC client connections.

parser(): Extracts and sums numeric values from a given input string.

sender(): Acts as an RPC client, reading grep commands and querying connected machines.

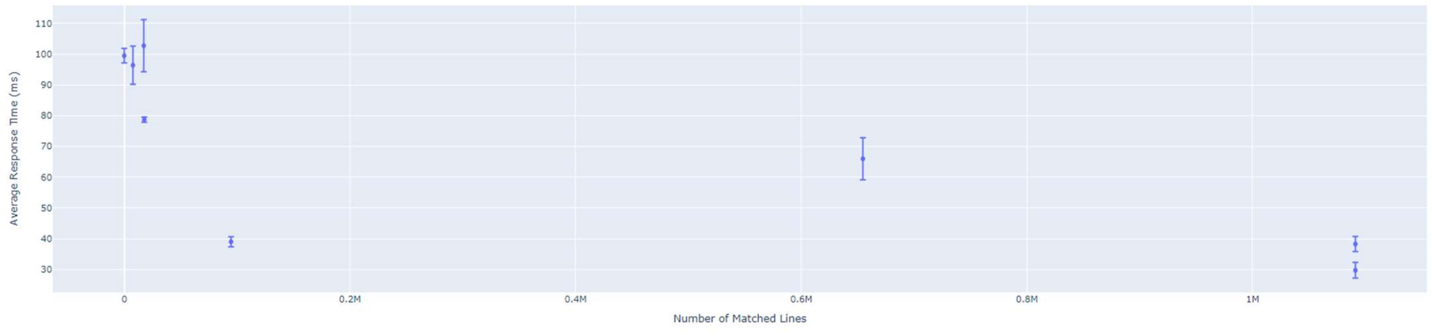
main(): Starts the RPC server and client functionalities.

Unit Tests:

1. Generate distributed grep results with all active servers
2. Generate local grep result of log files corresponding to all active servers
3. Given a pattern, compare grep output between distributed and local

Runtime Analysis:

Each datapoint is tested using 4 machines storing ~60 MB log files with at least 5 trials per data point. The average and standard deviation are as follows:


Average Response Time (ms)
0.2M
0.4M
0.6M
0.8M
1M
Number of Matched Lines