Tejus Kurdukar (tkurdu2), Sahan Kumarasinghe (sahansk2)

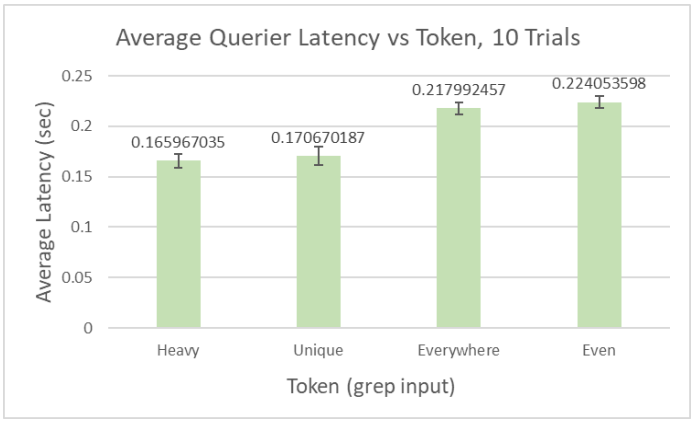
Design and Overview

Our distributed log querier is based on a client-server model. Each VM contains a continuously running logger binary. One "listener goroutine" in the logger listens for incoming TCP connections, and spawns a "worker goroutine" to handle requests. Worker goroutines are responsible for executing the command received from the socket. The worker goroutine writes command output locally into `/tmp`. Finally, the worker goroutine streams the file back with metadata to the querier. The querier binary dispatches remote commands to each logger concurrently by spawning one "query goroutine" per logger, allowing each logger to execute the command asynchronously. A mutex is used to protect stdout; only one goroutine can direct output from a logger to stdout at a time. To keep output readable, a query goroutine holds the mutex until all output from the logger is transferred back. See `logger.go, querier.go`.

Unit Testing

First, we generate one sample log per VM. These sample logs consist of random bytes and "tokens" - small signatures, appearing at different, parameterized frequencies across VMs. We validate querier output for four tokens - an "everywhere" token that appears in all VM logs on 50% with lines, a "unique" token that appears in VM 1, a "even" token that appears even-numbered VMs, and a "heavy" token that appears frequently (50% of lines) on VM1 and rarely (5% of lines) on others. See `generate_test_data.py, run_unit_tests.py`.

Performance Profiling



We generate logs of size 60 MB each with the mentioned tokens. We run the querier on VM5 to query VMs 1-4, comparing querying latency between the four tokens. We use the `date` utility to measure the querier execution time and average across ten trials. The graph plots mean and marks one stdev above and below.
We noticed that "heavy" and "unique" tokens have similar performance. Given that both tokens appear frequently on VM 1, this is expected.
However, we expected slightly more latency from heavy token, since it appears on other VMs. The internal implementation of grep may contribute to increased latency for unique token.

We noticed that Everywhere and Even token both have similar performance, but greater latency than Heavy/Rare. This is expected because the Everywhere and Even tokens appear more across all VMs. Surprisingly, the Even token has greater latency than the Everywhere token. This is surprising because even token is on fewer VMs. Something expected was that Heavy and Everywhere have significantly different performance. This is expected because the raw number of bytes written to disk and sent over the network for each VM is less for Heavy (Heavy sent back on average 5.25 bytes per VM, Everywhere sent back 6 bytes per VM).