CS425 Distributed System MP1 report

Design

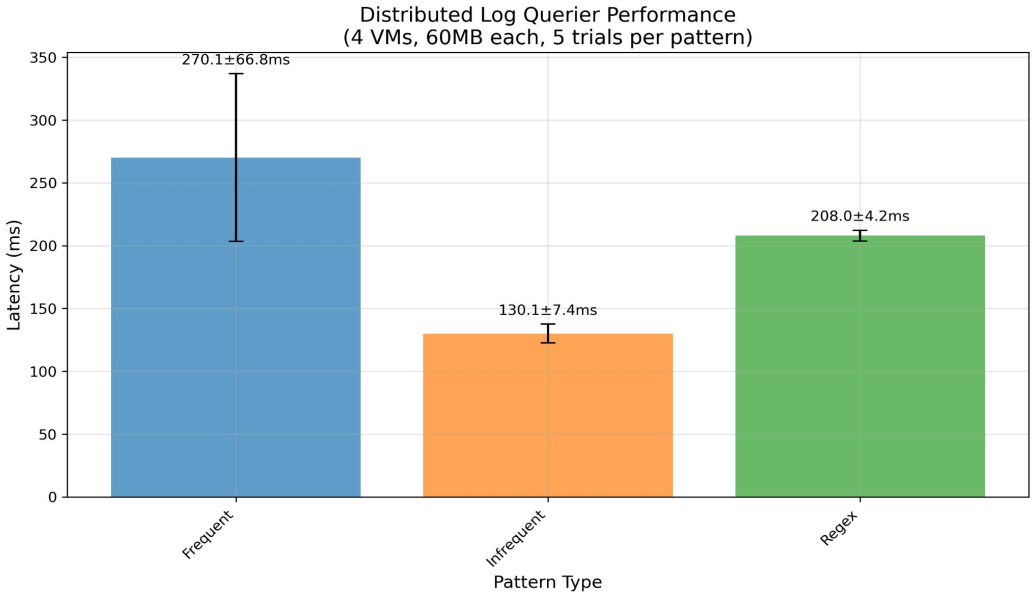
Our distributed log querier implements a parallel RPC-based architecture using Go's built-in RPC framework in a client-server model. The client initiates queries and aggregates results from all servers, while each server runs a local grep process on its assigned log file. Communication occurs through TCP-based RPC calls with configurable timeouts. The client spawns concurrent goroutines to contact all 10 servers simultaneously, and each server executes grep with the given pattern and returns matching lines with filename prefixes. The client then aggregates the results and presents them with line counts per machine.

Several design decisions optimize performance: servers are queried concurrently rather than sequentially to minimize latency, grep runs locally on each server instead of transferring files to save bandwidth, and connection and call timeouts (2s and 10s) provide fault tolerance against server failures. Additionally, the absence of a central coordinator removes a single point of failure and reduces overall system complexity.

Unit Tests

Unit tests covered frequent patterns, infrequent patterns, regex-based queries, and distributed verification across machines. To verify correctness, we generated known log files with predictable patterns on all 10 machines, executed remote grep queries via RPC, and compared the results against local grep runs on the same files. We checked that remote and local outputs matched exactly and calculated both per-machine and total match counts.

Performance



The results demonstrate that our parallel architecture effectively distributes the grep workload, with performance scaling well with different patterns. It also shows that queries involving larger matches or more complex evaluation (frequent keywords, regex) naturally cost more time. Infrequent lookups are cheaper in average latency.