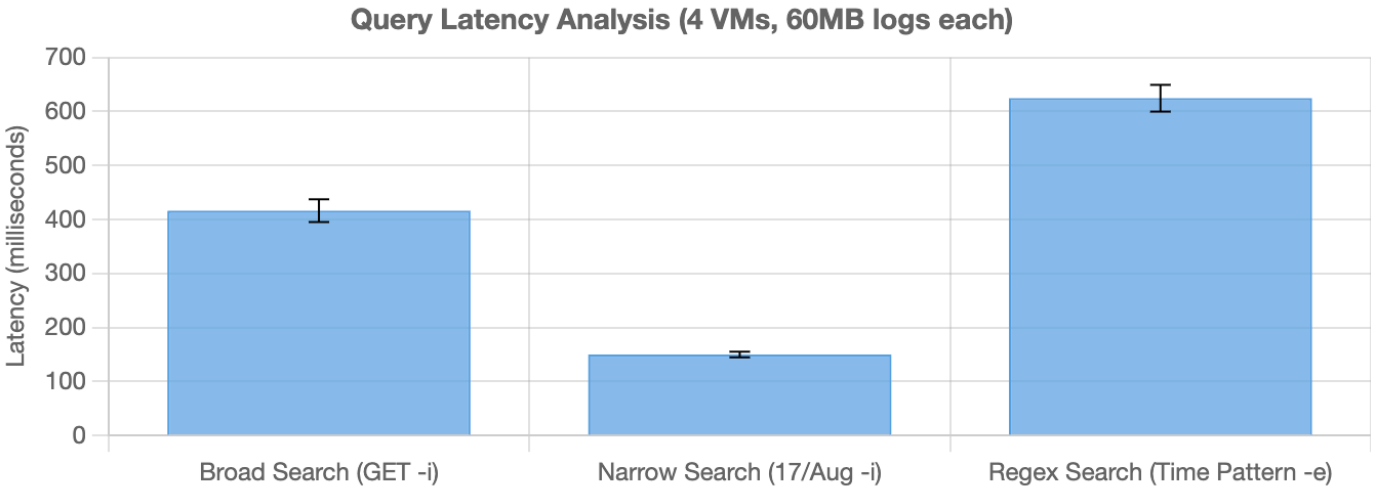
CS 425 Distributed Systems MP1 Report
Aryaman Ramchandran, Aditi Sadwelkar

Our distributed grep system implements a client-server architecture using socket programming adapted from Beej's guide. The client employs Python's ThreadPoolExecutor to query multiple VMs concurrently, sending JSON-formatted grep requests containing patterns and options. Each server executes grep locally on its designated log file (machine.X.log) and returns results.

Unit tests covered three query patterns across 4 VMs, each storing 60MB log files:
(1) Broad case-insensitive search for "GET" representing high-match scenarios.
(2) Narrow date-specific search for "17/Aug" testing selective queries.
(3) Complex regex pattern "[0-9]{2}:[0-9]{2}:[0-9]{2}" evaluating computational overhead.
Each test executed 10 trials to ensure statistical significance.



The performance analysis reveals distinct latency characteristics based on query complexity. The narrow search achieved the lowest average latency (149.9ms, σ=5.5ms) due to fewer matching lines reducing network transfer overhead.

The broad search showed moderate latency (416.3ms, σ=23.1ms) as expected from processing numerous matches.

The regex query exhibited the highest latency (624.4ms, σ=27.9ms), demonstrating the computational cost of pattern matching.

The low standard deviations (1.3-4.5% of means) indicate consistent system performance. The linear relationship between match volume and latency confirms that network I/O dominates execution time. Concurrent querying via ThreadPoolExecutor effectively masks individual server processing delays, achieving sub-second response times even for complex patterns across distributed logs.