CS 425 – Distributed systems
Distributed Grep

The Distributed Log Query System has two components - Server and Client. Both the components are deployed in all the machines of the distributed system. The Client on any machine can query the logs from all the other machines and display the query results in the terminal. Both the client and the server components are implemented in Java. Sockets are used for the Client-Server communication. A shell script with the name “dgrep” is used to simplify the command line interface for both client and server.

Log Querier Server

The server component has two classes - LogQuerierServer and LogQuerierServerThread. Every machine in system runs an instance of LogQuerierServer. The server can handle simultaneous requests from multiple clients. Each request is handled asynchronously by a separate worker thread. Whenever a request is received, the LogQuerierServer accepts the connection and spawns a thread (an instance of LogQuerierServerThread) to handle the request. This worker thread is responsible for handling the request from the client. It receives the arguments (key and value regexps and other grep options) as a ClientArgs object from the client and performs the grep operation on the file “machine.i.log” where i is the serverId of the server. The result of the grep operation is streamed back to the client over the client socket. The data is streamed and not kept in the memory. Hence the memory usage of the server and the worker threads is minimal.

Log Querier Client

The client component has two classes - LogQuerierClient and LogQuerierClientThread. The LogQuerierClient class takes care of reading list of servers from a property file (boltdb.prop) and spawns a client thread (an instance of LogQuerierClientThread) to talk to each server. Each of the client thread sends the grep arguments to the server, gets the grep output and writes it to a temp file. Once the thread finishes, LogQuerierClient instance will be notified upon which it prints out the file to the console.
Design decision made on the client side : One client thread per server so that once a server finishes, we would see its output on the client side immediately and doesn't depend on the state of execution at any other servers.

Unit Test Design

The LogQuerierTest class is used to perform unit tests on the Distributed Log Query system. It has methods to (i)Generate Log files (ii)Execute Distributed grep on all/some machines (iii)Execute Local grep (iv)Compare the output of distributed and local grep. The log files are generated with frequent (occurring in 70% of the lines), Not so frequent (occurring in 25% of the lines), Rare (occurring in 4% of the lines) and Very rare (occurring in 1% of the lines) patterns. Every line in the log file is appended with “NODE##i” where i is her serverId on which the log resides. The unit tests test the system with {Frequent, NotSoFrequent, Rare, VeryRare} patterns on {One, Some, All} machines in the system.

Performance

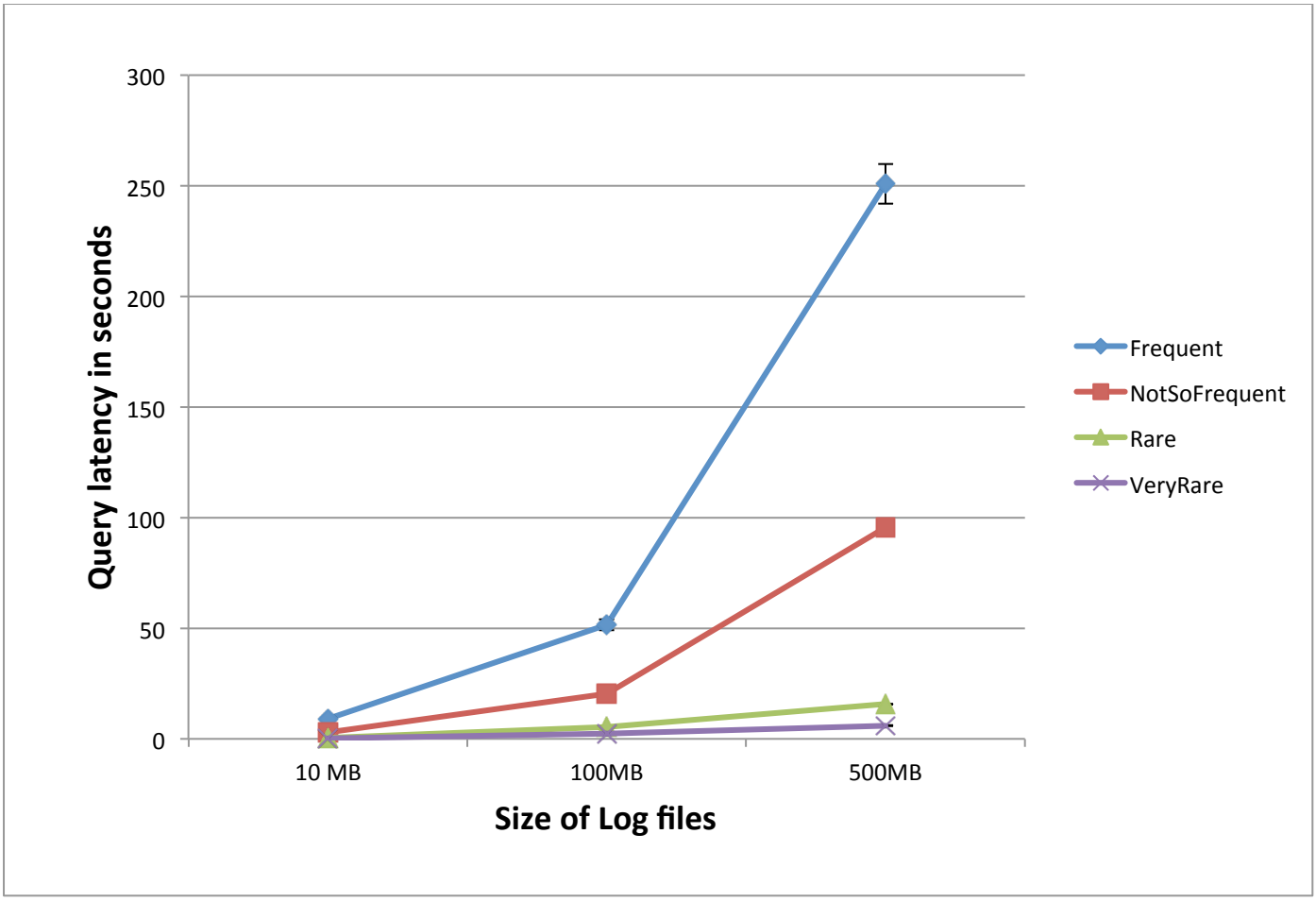
The query latency is calculated from the time the request is made till the results from all the servers are received. Hence a delay at one server increases the overall query latency. The Average Query latency (for 5 readings) for 100MB logs on 4 machines for various patterns are as follows -
Frequent – (54.398s, 48.667s, 52.425s, 49.404s, 53.049s) ⇨ **Avg. Latency = 51.5886 seconds**
Not So Frequent – (19.429s, 19.636s, 20.764s, 20.664s, 21.868s) ⇨ **Avg. Latency = 20.4734s**
Rare – (5.942s, 4.963s, 5.283s, 5.023s, 5.154s) ⇨ **Avg. Latency = 5.273 seconds**
Very Rare – (2.197s, 2.468s, 2.233s, 2.863s, 2.667s) ⇨ **Avg. Latency = 2.4856 seconds**

Performance of the Log Querier system for different log sizes

10 MB
Frequent – (9.078s, 8.846s, 8.936s, 9.126s, 9.278s) ⇨ **Avg. Latency = 9.0528 seconds**
Not So Frequent – (2.887, 2.328, 2.957, 3.028, 2.768) ⇨ **Avg. Latency = 2.7936 seconds**
Rare – (0.443s, 0.391s, 0.390s, 0.533s, 0.392s) ⇨ **Avg. Latency = 0.4298 seconds**
Very Rare – (0.287s, 0.221s, 0.218s, 0.212s, 0.202s) ⇨ **Avg. Latency = 0.228 seconds**

100 MB
Frequent – (54.398s, 48.667s, 52.425s, 49.404s, 53.049s) ⇨ **Avg. Latency = 51.5886 seconds**
Not So Frequent – (19.429s, 19.636s, 20.764s, 20.664s, 21.868s) ⇨ **Avg. Latency = 20.4734s**
Rare – (5.942s, 4.963s, 5.283s, 5.023s, 5.154s) ⇨ **Avg. Latency = 5.273 seconds**
Very Rare – (2.197s, 2.468s, 2.233s, 2.863s, 2.667s) ⇨ **Avg. Latency = 2.4856 seconds**

500 MB
Frequent – (254.06s, 245.97s, 239.52s, 251.63s, 263.32s) ⇨ **Avg. Latency = 250.89 seconds**
Not So Frequent – (96.532s, 92.782s, 98.017s, 94.317s, 95.732s) ⇨ **Avg. Latency = 95.476s**
Rare – (15.557s, 15.855s, 15.768s, 15.636s, 15.938s) ⇨ **Avg. Latency = 15.7508 seconds**
Very Rare – (6.411s, 5.836s, 5.963s, 5.783s, 6.107s) ⇨ **Avg. Latency = 6.02 seconds**



Observations from the graph

- ⇨ Irrespective of the log file size, the query latency is much higher for the frequent patterns. This is because the frequent patterns appear in 70% of the lines in the log files. The servers end up streaming 70% of the lines in the log files thereby increasing the query latency.

- ⇨ The query latency is directly proportional to the size of the log files. As the size of the log files increase, there is a significant increase in the query latency. From our experiments we found that the execution time of the grep command increases only by a few seconds as the files grow larger. Thus, most of the time is spent in transferring the data over the network. Also, on the client side, in order to avoid multiple client threads writing on to the console simultaneously, we made a design decision to make the client threads write the output from the servers to temporary files before displaying them to the user. This brings in disk I/O which adds up to the query latency.

- ⇨ The low standard deviation (so low that it is hardly visible in the graph) for the plotted data points shows that the experiment has fetched consistent results. Since the experiment was performed post-midnight in the EWS labs, the low traffic during the nights has contributed to the consistent results of the experiment.