Log Extractor: Project Description

[Note: SSHFS needs to be installed to mount the network file system locally.]

Since data is present on a network file system, the most important aspect to be considered while solving this problem is the total number of round-trips required. Files present on an NFS will be read in blocks of data. So, if the data requested is not in the current block, the program will make a round-trip to fetch the respective block and this would take time.

The code has now been written to minimise the total number of round-trips as much as possible. Also, we want the program to start returning query logs as quickly as possible, in order to minimise the engineer's wait time. To accomplish these goals, binary search is implemented on 2 levels: (1) Across log files (based on file names, since log files are created in chronological order) and (2) Inside start-file for rows (start-file is the log file containing the start timestamp of the query).

First, binary search is used on log file names. Comparison is done between the timestamp corresponding to first log line in the file with start, end timestamps of the range. If the timestamp belongs in the range, we get a hit and the program starts printing the current and subsequent log files concurrently. Binary search is called recursively on the preceding log files and the above idea is repeated to output other log files in the query range.

Log file containing the start timestamp of the queried range will be obtained using the above explained binary search method. For this log file containing the start timestamp, similar ideology of binary search is used to find start row corresponding to the start timestamp. If a row with timestamp greater than or equal to the start timestamp is found, we get a hit and the program starts printing the log lines. Binary search is called recursively on the preceding log lines and the above idea is repeated to output other log lines belonging to the query range.

Using the above implementation, the program will not print logs in chronological order, but since that is not demanded in the problem statement, we don't need to worry about it.

Performance Analysis

This implementation starts outputting the query logs within 1-2 seconds, much faster than my previous implementation where I used multi-level indexing since we are not processing the entire data now to create index tables. Now, we are printing the logs as soon as we get a hit, thus minimising the number of round-trips.