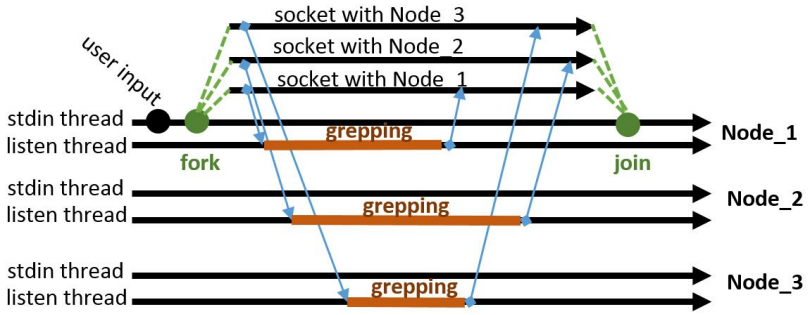
Luis Remis - remis2
Xuewei Zhang - xzhan160

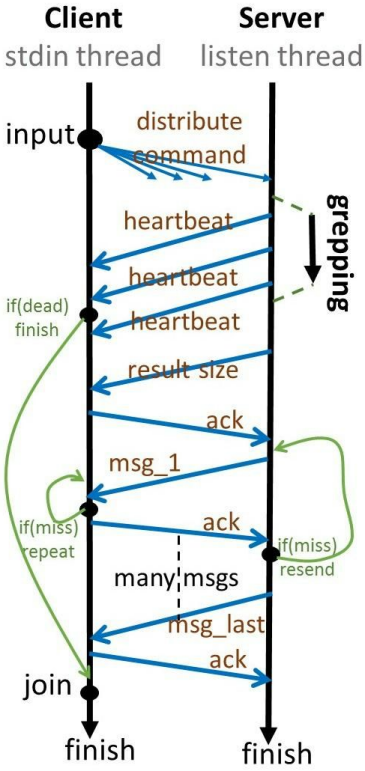
Design and Implementation

Our grepping application was written in C++ to in order to achieve good performance.

We wanted to have the same exact code in every machine under the same workflow. Our application launches 2 threads when it is run: one thread listens to socket connections, and the other thread waits for user to input a command.

Once receive a user input, this node acts as Client: NUMBER_OF_NODES threads fired to connect to each node (including itself), request the data, receive the data and print it (with mutual exclusion to prevent race condition during printing). After all the threads are done, connections are closed, and the User threads stays idle waiting for a new grep command. In a similar way, after a server replies a query, it remains idle waiting for new connections to come.





Robustness mechanism

We implemented the following robustness mechanism in case of server failure:

1. If any server is not accepting connection, the client thread allocated for that server will exit with an error message, while leaving the rest of the threads work as usual.
2. If, after connection, a server crashes during grepping local files, Client will display an error message and continue to accept data from other servers. We implement this mechanism by heartbeat message.
3. We have a robustWrite&Read method to deal with message latency or missing caused by any reason.
4. We also have spiltWrite&Read method to deal with transfer error with large files.

Testing mechanism

We implemented a Testing Application that will generate the logs to be tested in each node, wait for all the nodes to be ready (by receiving a specific message), and finally issue a query and check if the output is the expected. We use a similar client-server approach as we used for the grepping app itself. We can choose between two different queries to be processed.

Performance

Our application proved to be very good performing. Testing on vm.log files:
when grepping with a non-existing pattern, the query is completed after, on average, 50ms;
when grepping with a frequent pattern (“grep apple”), the query is completed after, on average, 70ms;
when grepping with a very frequent pattern (“grep a”), the query is completed after, on average, 10 s.
This is, transfering almost 100MB of logs and printing them in the screen.