MP2 – Distributed Log Querier – CS425 T3 (3 hours)

Kevin Peters – klpeter2
Jack Montag – montag1

Design

For the distributed log system, we decided to stick to the acronym *KISS (Keep It Simple, Stupid!)* and design the system accordingly. Distributed systems can get complicated very quickly; therefore it is best to keep things as simple as possible to avoid design flaws and too much complexity. The design of the log system consists of a single client and server(s), also called host(s). The client prompts a user to enter a pattern along with the host(s) he or she wishes to retrieve log file(s) from. The client then opens a socket using the C++ Boost ASIO library and attempts to connect to each of the host(s). At this point it is important to mention that a server that is to be queried should have the server executable running in order to receive connections, otherwise the client will output a "connection refused" error to the console and not return a log file for that particular server. The server essentially opens a socket when it's started and listens for incoming connections from a client. Upon a successful connection between the server and a client via TCP protocol, the client sends a message to the server which includes the user defined pattern to search for and the server's designated host name (e.g., "machine.1.log" == "linux5.ews.illinois.edu"). The server then creates a log file named with the designated host name and fills it with lines consisting of "TargetL" and "Garbage". The actual size of the log file is specified by the user in the Constants.h file (see implementation below). Upon successful creation of the log file by the server, it then saves it to the ~/tmp/ folder. Afterwards, a call to the system "grep" command is made that searches for all occurrences of the pattern the user specified, and saves those occurrences along with their line numbers to another temporary file in the ~/tmp/ folder. This is done, because the server will be sending these results back to the client; storing them in memory would slow down the program too greatly. After the temporary grep file is created, the server then reads its lines and sends them back to the client that has been listening for the server(s) it successfully made a connection with to respond. The lines are then sent from the server to the client and stored in a temporary file. After the server(s) have finished transferring all of the log data, they delete their temporary files, and the client is now ready to print the log data it has stored in its temporary files to the console. After printing the results to the console, the client then deletes any temporary files it may have created and finishes execution. The server(s) should now be restarted in order to repeat this process again.

Testing

The distributed logging system takes on average 7 seconds to retrieve the results from running grep on a on a 100MB log file with a pattern occurring every 10th line. Due to the limitations of EWS, we only tested our system on 3 machines which took a total of 25 seconds to retrieve. We've also included a hard coded unit test which checks the correctness of the queried log files. The same method used to generate a log file on a server is used by the client to generate a log file in the unit test. The grep command is then used with the pattern the user entered on this log file, with the results being saved to a temporary file. The unit test then checks to see if the temporary file matches the log(s) retrieved from the server(s). The test passes if it confirms that the files are the same. Note that this unit test is enabled by default, but can be disabled if desired in Constants.h.

Implementation and Usage

To use the distributed log system, begin by starting the dgrep-server executable on any machine that is intended to be a server (i.e., generates a log file). When the dgrep-server starts, it will be in an infinite loop listening for incoming connections. To query the server(s) for their log(s), start the dgrep-client executable like so with the following arguments: "./dgrep-client <pattern> <machine.1.log> <machine.2.log> … <machine.n.log>". An example would be: "./dgrep-client TargetL machine.1.log machine.2.log". This would the query the machines for the pattern "TargetL". Note that the only strings generated in the log files are "TargetL" and "Garbage" with "TargetL" appearing every 10th line. After querying the servers, it is important to end each server process after completion. Querying the server(s) again without restarting their process will result in an exception being thrown.

In order to change which machines are mapped to which address (e.g., "linux5.ews.illinois.edu" == machine.1.log), you can go into the mp2-server/src folder and open the Constants.h file. Here you can change the machine mappings, along with the size of the log files to generate (default 10MB), the location of the temporary directory, and whether or not to include unit tests. Note, that when changing the size of the log files or to enable/disable unit tests, the Constants.h file located in the mp2-client/src folder must also be modified. Failure to do so may result in unit test failure. ** When changing settings, make sure to recompile the executable that has the changed source/header files with the associated changes. **

Compilation

The MP2 directory contains 2 subdirectories: mp2-server and mp2-client. The subdirectories each contain a folder named debug. Inside the debug folder is a makefile. Typing "make all" in the command prompt with the debug folder as the working directory will compile the dgrep-server and dgerp-client executables in the mp2-server and mp2-client debug folders respectively. Typing "make clean" will also remove any of the object files the compiler has created.