Joey Bahary (jbahary2), Srijan Chakraborty (srijanc2)

ECE 428 MP1 (DISTRIBUTED LOG QUERIER) REPORT

The distributed grep MP was created as a tool for debugging all of our future MPs. The design we decided to go with was a client-server model with every machine doubling as a server AND a potential client through different processes. We chose to grep first and then pool results, because we wanted to outline specific results by name of machine as well as the line numbers off each machine. The network links are a severe bottleneck for data transfer, so we chose this to also send less information. Since this is a debugging tool, we used a fixed set of IPs defining our internal nodes. Finally, for fault tolerance, when a server goes down, we catch this phenomenon using Go's errors library, report it as down, and move on to the next VMs to query. For testing we decided to ensure our distributed grep could handle a few different cases. For our first test case we elected to create a script to generate a large amount of random words, and to insert at an interval the VM's hostname. We then created another script to verify that our program could grep and detect the lines that the hostname was inserted in. For our second test we decided to generate another log file, and insert the date at certain intervals. Then we used a regex line to test our code. Finally, below is a graph of our clusters reporting real time time it takes to query 4 VMs with 60MB log files while other VMs are down, using the time shell command. We have 10 trials of data. The orange line is the average and the grey the standard deviation. The latency seems to be pretty standard, with the average time being around 1.897. There was a spike at the second trial; this was a regexp search, and thus was higher than regular. The data overall shows a standard deviation of 0.5, which is not huge but not miniscule; we believe this to vary so much due to the results of the grep varying so much, resulting in the **amount** of content returned over the network causing more slowdown. This is due to how we designed to grep on server and return to client; had we pulled entire file then grepped, we would have a MUCH higher average but a much lower std. deviation.