Robert Judka
CS550
Programming Assignment 2

Performance Results

The Python scripts for running the evaluation and the data generated by running multiple simulations is found in the 'evaluation/' directory.

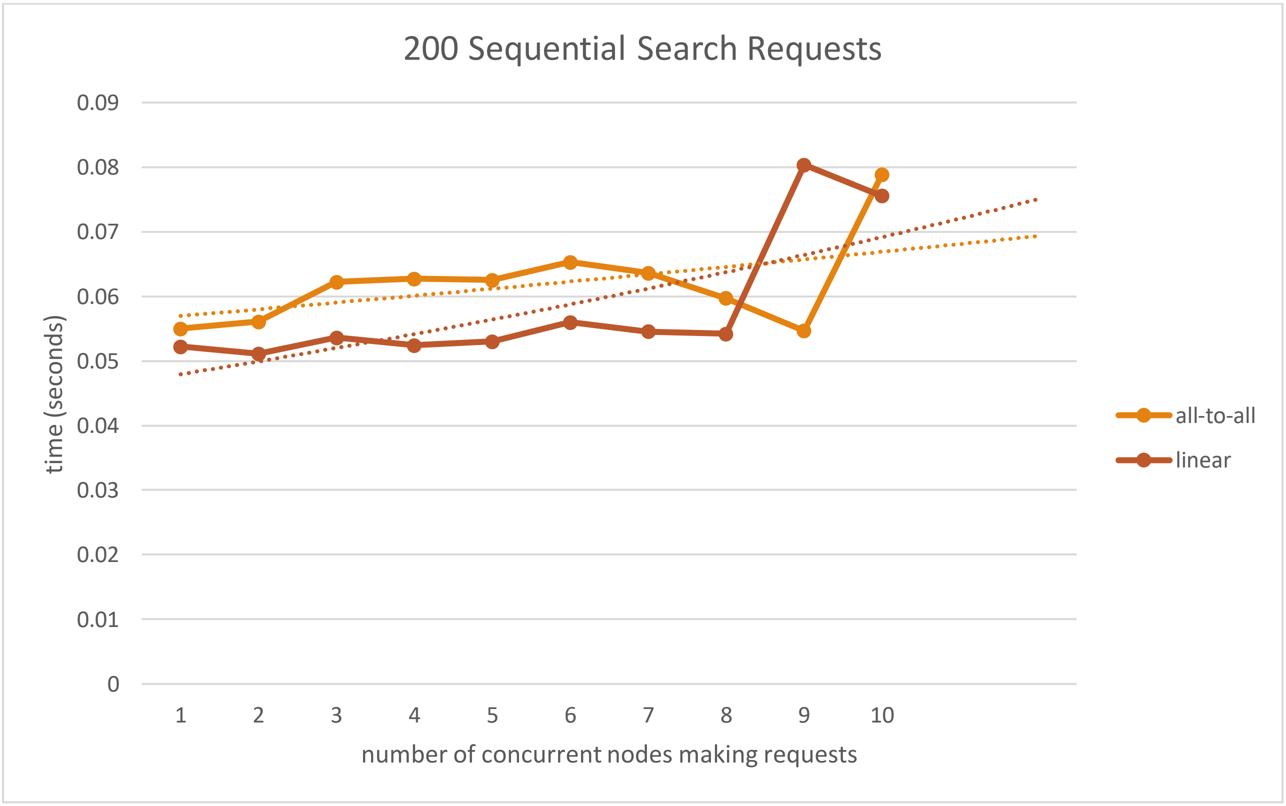
Node Simulation

The script 'node_simulation.py' takes an integer parameter *n* for how many concurrent peers to run, and the topology type (in our case either 'ata' for *all-to-all* or 'l' for *linear*). The simulation will automatically create 10 peers and 18 nodes (but only have *n* nodes send requests) and kill all those processes after the simulation is complete.

Evaluation

The script 'evaluation.py', given a parameter for the topology type to evaluate, takes all the log data generated by the simulation and calculates the average request time for a given number of nodes. NOTE both these scripts may fail if the directories are not set up correctly.

Results



Obviously, something looks a bit off here, *linear* was expected to be slower than *all-to-all*.

Examining the forecast trendlines, we can see that things start to look more “normal”, but it still does not explain the inconsistencies from the experiment conducted. Overall, however, we can see that performance generally decreases as the number of concurrent nodes making requests increases, which is expected in this type of system.

A possible explanation for why the *linear* topology performed better than the *all-to-all* topology could be because of the way peer communication is handled. Working off the assumption that communication between peers would be more intermittent, we opted for only opening connections between two peers when a request was made, instead of having a continuous connection. This worked fine for manual tests during development (as I could not send 200 requests in a matter of seconds), but it has shown to be a suboptimal approach for the evaluation.

Essentially, if my hypothesis is correct, in a *linear* topology there were a lot less connections that needed to be established as a single peer would only ever connect to at most 2 other peers, whereas in the *all-to-all* topology, a peer would be opening and closing some *n* connections for every request (and even more if it had to forward the message), where *n* is the number of peers in the topology. Thus, this problem repeated 200 times sequentially, would drastically increase the connection clashing in the *all-to-all* topology.

Discussion

Normally, in the real world for this hierarchical architecture, a *linear* topology is not used. There a few reasons for this. As each peer is only connected to at most 2 peers, a message must travel through more peers to reach all peers (if possible). This also means that if any peer fails, then some remaining *n* peers become unreachable. Assuming all peers are good, an appropriate *TTL* value would also need to be used to ensure the message can reach all the peers before timing out.

Peers also could become bottlenecks in the system, since there could be a point when one peer must handle multiple messages from many different clients. As there is only one path for each message, they will all need to wait for the single peer to handle them.