Robert Judka
CS550
Programming Assignment 3

Performance Results

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

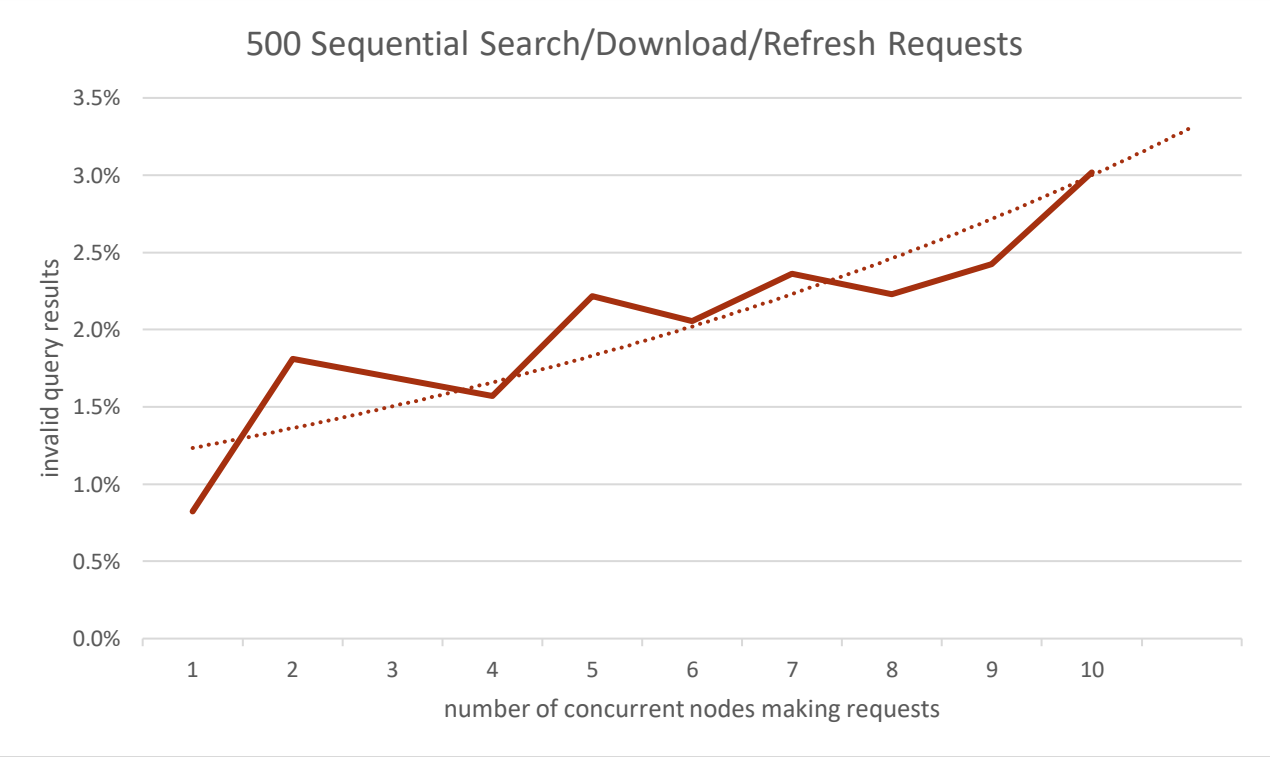
Node Simulation

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 'push' for *PUSH*, 'pull_n' for *PULL FROM NODES*, and 'pull_p' for *PULL FROM PEERS*). The simulation will automatically create 10 peers and 18 nodes (but only have *n* nodes send *search*, *download*, and *refresh* requests). Requests are made randomly as are any modifications to files of nodes not making requests. At the end, all the peer and node processes are killed and the '*_client.log' logs should contain any data necessary for the evaluation.

Evaluation

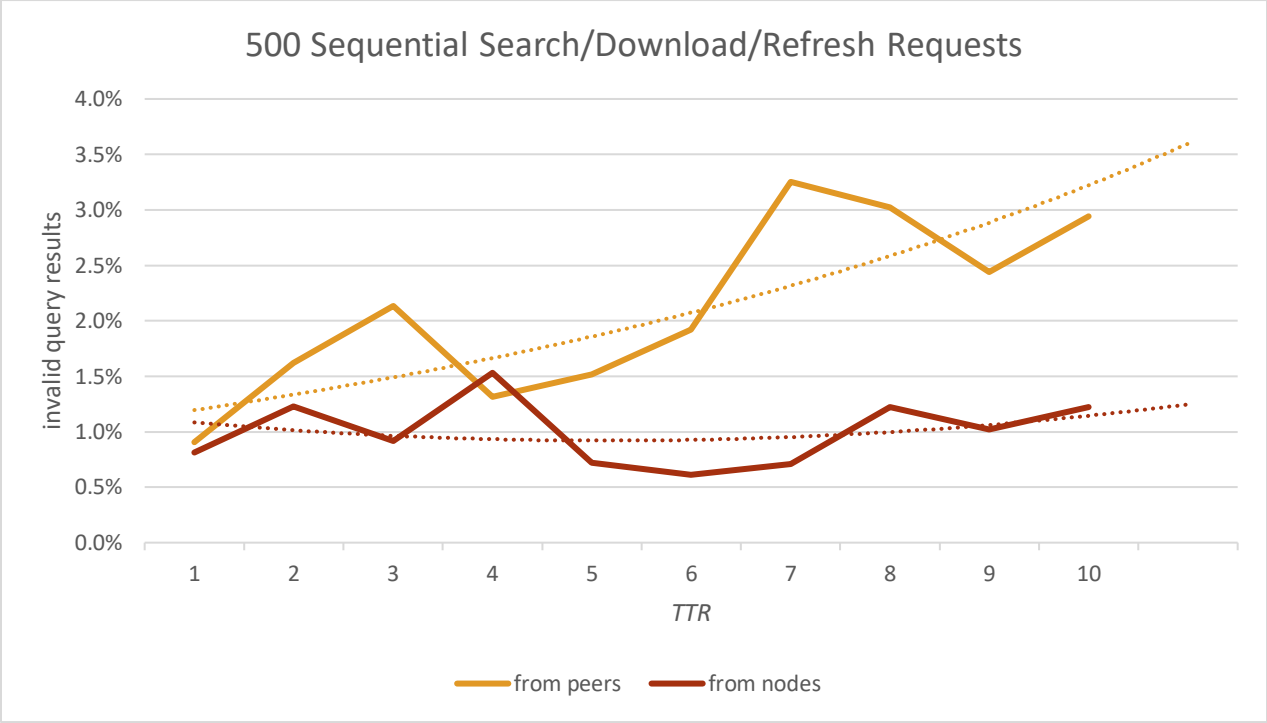
The script 'evaluation.py' takes all the log data generated by the simulations and calculates the average invalid query results. NOTE both these scripts may fail if the directories are not set up correctly.

PUSH Consistency Method



Within our limited-sized network, we can see the *PUSH* method provides favorable consistency results, however scalability could be an issue. This could be due to the increasing traffic on the network as more nodes are making requests and/or by the need for messages to broadcasted to all other servers anytime a file is modified. Both these scenarios would cause communication to be slowed down between each server, affecting how quickly stale files could be invalidated.

PULL Consistency Methods



The instability in both these methods is very apparent, which could mainly be caused by the *lazy* evaluation of *TTR* values. Instead of checking as every *TTR* value expires, we would simply mark a file expired and get around to checking it later. As expected, the greater the *TTR* value is, the more inconsistency there is in the system since we are checking for updates to downloaded files less frequently.

Comparing the two methods, we can see the *PULL FROM PEERS* performs a lot worse as the *TTR* value increases. This could be attested by the fact that the cache of modified files grows larger waiting for *TTR*, and thus causing a flood of invalidation requests when the *TTR* expires. The *PULL FROM NODES* method also has the added benefit of not needing to go through the peers (which can be seen as a bottleneck of sorts) since it can directly query the origin node to compare the version number. The only communication a node has to make with its peer if the cached file gets invalidated.

Discussion

Since the experiments done using the *PUSH* method were slightly different than those done using the *PULL* methods, the graphs don't necessarily show which method "is better". First, the *PUSH* evaluation seems to point to the conclusion that it the least consistent, however the measurements are comparing invalid query percentages as more nodes perform queries, rather than the *PULL* evaluations which are comparing a variable *TTR* value.

Assuming equal amount of traffic in any of the scenarios, the *PUSH* method has the advantage of updating files as soon as an update takes place, thus allowing for greater consistency between the origin file and remote file(s) (but not necessarily in the super peer). This of course comes at the cost of more traffic between the servers, because the nodes are essentially stateless and need to send the invalidation message to all other nodes.

The *PULL* methods, on the other handle, have the advantage of targeting only the nodes which need to be checked for consistency (i.e. the nodes which have downloaded files) because of the additional state that is stored. The tradeoff for them is finding a *TTR* value which does not affect bandwidth but also allows for files to be updated frequently enough to not be a problem.

As mentioned early, the *PULL FROM NODES* method ends up being most consistent since the nodes can directly check their remote files' validity with the respective origin nodes and update their peer accordingly. As long as a reasonable *TTR* value is used, nodes can remain relatively consistent as the validation check does not need to traverse a majority of the system.

Given the advantages and disadvantages mentioned above, we can see why certain services, such as email, use a pull-based approach so the email server can offload the responsibility of managing updates to the clients. While, when real-time information is needed, such as from IoT sensors or any smartphone notification, a push-based approach is more viable. However, as each of these approaches have their weaknesses, a hybrid of the two methods can be used to find a good middle ground, which has been a successful approach in web-based caching.