CSCI4061 F2018 Project 3

Date: 12/12/18

Name: Cyro Chun Fai Chak, Kin Nathan Chan, Isaac Aruldas

id: 5330106- chanx393, 5312343 - chakx011, 5139488 - aruld002

Part A.

For the assignment, we have tried a few commands to observe the behavior: 1:1, 50:50, and 100:100 numbers of dispatcher and worker thread.

For 1 dispatch and 1 worker, we observe there were a particular behavior during the process.

From the urls file, we can see that each file has 3 requests on the line, and with the 1 worker and dispatcher, it used 0.2s to downloads all 400 files, and we observe that for the first request of each file/URL, it is guarantee that it will be missed, since it cannot be found in cache. After that, the second and the third one will be hit since it is able to locate in the cache. If it is a hit stage in the cache, it can be done in 1 us to finish the request. However, if it is missing stage in the cache, it generally takes up to the range from 30us to 100us. When we increase our size of dispatcher and worker to 50, we can see a difference that it takes 0.08s to download all 400 files. From the request log, we can observe that there was multiple request take extremely long time to process. As for our last approach, we tested with our maximum number of threads pool which is 100 workers and 100 dispatchers. We observe that the total wall clock time is taken longer as the number of threads is added. However, the download process for this trial has reduced to 0.06s. As the rate, we can assume that when we increase the size of the workers and dispatcher, we will be expecting a shorter time of downloader, but more time on searching it in the cache.

Part B.

In this section, we will run our server with 100 dispatcher threads with different numbers of worker threads, such as 1,2,5,10,20,50 75,100. First, we ran the xargs test with the bigurls file with this following command: ./webserver testing 9000 100 worker 0 100 100.

On the client side, we will run the xargs test, as cat bigurls | xargs -n 1 -P 8 wget. This means that we will be running wget 8times at the same time with 1 argument per time. Here is the following results for different numbers of workers: 1, 2, 5, 10, 20, 50, 75, 100.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 5 | 10 | 20 | 50 | 75 | 100 |
| Total Time | 4203 | 6135 | 7428 | 12133 | 40927 | 51017 | 79782 | 96101 |

In general case, we will be expected the more workers we have, the shorter time, the requests will be finished. However, in the big urls, all 144 requests are requesting the same file. In this case, the benefits of multithreads will not be beneficial anymore. If we have multiple different requests are requesting unique files, the time of finishing the request will be shorten since we are running 8 wget at the same time. But for our case, since all 144 requests are requesting the same file, the rest except the first request will take nearly no time (0/1 microsecond) to complete as it will be found in the cache. Therefore, even though we increase the numbers of workers thread, it will not help the situation, and it might even take more time to complete the whole process.