CSCI4061 - Project 3 - Group 46

Pierre Abillama
Derek Frank
Jack Dong

Performance Analysis Report:

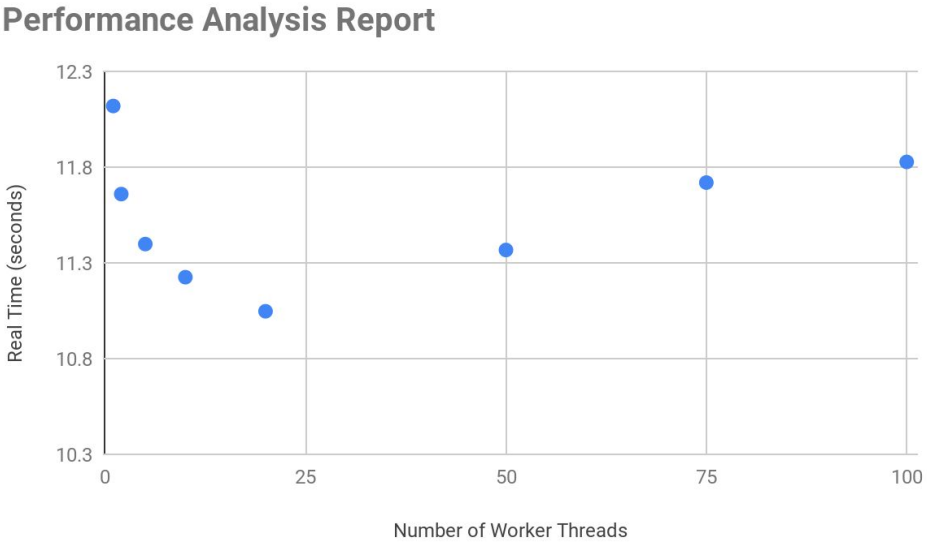
In order to investigate the effect of the number of worker threads on the time it takes to service requests, the following procedure was executed:

1. After navigating to the project directory and running ***make***, run the following command in one terminal window where <# workers> is the independent variable:
   `> ./web_server 9000 ./testing 100 <# workers> 0 100 100`

2. After navigating to the tests folder under the project directory, run the following command in another terminal window:
   `> time cat ../testing/bigurls | xargs -n 1 -P 8 wget`

The results obtained are summarized in the table below.

Worker Threads	Real Time (s)
1	12.069
2	11.609
5	11.348
10	11.175
20	10.997
50	11.317
75	11.669
100	11.777



Interpretation of results:

The results obtained highlight the presence of a “sweet-spot” for the number of worker threads servicing requests. When the number of threads is too low, the time spent servicing all requests is primarily due to the latency of servicing a request. As the number of threads increase, the server’s throughput increases despite the newly introduced overhead due to context switching, time slicing, and synchronization. This means the real time is expected to decrease as shown in the graph above. After the “sweet-spot” of 20 worker threads, the overhead now outweighs the benefits of threading via concurrency and improved throughput. Therefore, we expect the real time to slowly increase as shown in the graph above.

Caching Mechanism:

As discussed in the README, we implemented an LFU caching mechanism. In order to ensure that our caching mechanism works properly, we ran the following commands:

Note: T<#> denotes the terminal window number, and that the cache size is 5 entries

```
(T1) > ./web_server 9000 ./testing 100 20 0 100 5
(T2) > wget http://127.0.0.1:9000/image/jpg/29.jpg
(T2) > wget http://127.0.0.1:9000/text/html/30.jpg
(T2) > wget http://127.0.0.1:9000/text/html/31.jpg
(T2) > wget http://127.0.0.1:9000/text/html/32.jpg
(T2) > wget http://127.0.0.1:9000/text/html/33.jpg
(T2) > wget http://127.0.0.1:9000/text/html/34.jpg
(T2) > wget http://127.0.0.1:9000/text/html/32.jpg
(T2) > wget http://127.0.0.1:9000/text/html/30.jpg
(T2) > wget http://127.0.0.1:9000/text/html/31.jpg
(T2) > wget http://127.0.0.1:9000/text/html/33.jpg
(T2) > wget http://127.0.0.1:9000/text/html/36.jpg
(T2) > wget http://127.0.0.1:9000/text/html/34.jpg
(T2) > wget http://127.0.0.1:9000/text/html/31.jpg
(T2) > wget http://127.0.0.1:9000/text/html/32.jpg
(T2) > wget http://127.0.0.1:9000/text/html/33.jpg
(T2) > wget http://127.0.0.1:9000/text/html/30.jpg
(T2) > wget http://127.0.0.1:9000/text/html/34.jpg
```

The output obtained at terminal window T1 was:

```
> [0][1][5][/image/jpg/29.jpg][17772][2874micros][MISS]
> [1][1][4][/image/jpg/30.jpg][17772][2802micros][MISS]
> [2][1][5][/image/jpg/31.jpg][17772][3205micros][MISS]
> [3][1][4][/image/jpg/32.jpg][17772][2106micros][MISS]
```

```
> [4][1][5][/image/jpg/33.jpg][17772][2560micros][MISS]
> [5][1][4][/image/jpg/34.jpg][17772][4053micros][MISS]
> [6][1][5][/image/jpg/32.jpg][17772][19micros][HIT]
> [7][1][4][/image/jpg/30.jpg][17772][21micros][HIT]
> [8][1][5][/image/jpg/31.jpg][17772][30micros][HIT]
> [9][1][4][/image/jpg/33.jpg][17772][20micros][HIT]
> [10][1][5][/image/jpg/36.jpg][17772][7046micros][MISS]
> [11][1][4][/image/jpg/34.jpg][17772][1335micros][MISS]
> [12][1][5][/image/jpg/31.jpg][17772][17micros][HIT]
> [13][1][4][/image/jpg/32.jpg][17772][33micros][HIT]
> [14][1][5][/image/jpg/33.jpg][17772][23micros][HIT]
> [15][1][4][/image/jpg/30.jpg][17772][33micros][HIT]
> [16][1][5][/image/jpg/34.jpg][17772][23micros][HIT]
```

This is consistent with our understanding of how caching can improve performance. When a cache MISS occurs, the worker thread has to read the file from the disk, which is slow and blocking operation. After the file has been read, it is added to the cache so that a subsequent request can utilize the content of the previous request. As expected, the HIT time is much smaller than the MISS time. In addition our LFU is shown to work properly. After the 6th `wget` command, we are expected to have `30.jpg, 31.jpg, 32.jpg, 33.jpg,` and `34.jpg` in the cache. Then after the 10th `wget` command, 34.jpg is the entry with the lowest usage frequency. It will be evicted at the 11th `wget` command, and re-entered at the 12th `wget` command (which explains why it's a MISS).