Write Up Assignment 3

For assignment 3 the unit testing I did as follows:

For all cases ran httpserver with localhost and port number of 8080

./httpserver localhost 8080 -l logfile.txt -c

To actually test I used curl as my client

1. Running curl with valid filenames
PUT (first time):

curl -T test.txt http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab -v

Client receives response header:

HTTP/1.1 200 OK
Content-Length: 0

logfile.txt content:

PUT ABCDEFarqdeXYZxyzf012345-ab length 112 [was not in cache]
0000000 54 68 69 73 20 69 73 20 74 68 65 20 64 61 74 61 20 74 68 61
0000020 74 20 69 73 20 69 6E 20 74 68 65 20 74 65 73 74 20 66 69 6C
0000040 65 0A 49 74 20 69 73 20 6F 6E 6C 79 20 61 20 66 65 77 20 6C
0000060 69 6E 65 73 20 6F 66 20 64 61 74 61 0A 54 6F 20 6D 61 6B 65
0000080 20 73 75 72 65 20 74 68 61 74 20 69 74 20 73 65 6E 64 73 0A
0000100 0A 54 45 53 54 20 46 49 4C 45 20 30
========

After trying it a second time

logfile.txt content:

PUT ABCDEFarqdeXYZxyzf012345-ab length 112 [was in cache]
0000000 54 68 69 73 20 69 73 20 74 68 65 20 64 61 74 61 20 74 68 61
0000020 74 20 69 73 20 69 6E 20 74 68 65 20 74 65 73 74 20 66 69 6C
0000040 65 0A 49 74 20 69 73 20 6F 6E 6C 79 20 61 20 66 65 77 20 6C
0000060 69 6E 65 73 20 6F 66 20 64 61 74 61 0A 54 6F 20 6D 61 6B 65
0000080 20 73 75 72 65 20 74 68 61 74 20 69 74 20 73 65 6E 64 73 0A
0000100 0A 54 45 53 54 20 46 49 4C 45 20 30
========

GET:

curl http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab -v

logfile.txt content:

GET ABCDEFarqdeXYZxyzf012345-ab length 0 [was not in cache]

========

If command is ran after running
curl -T test.txt http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab -v

logfile.txt content:

GET ABCDEFarqdeXYZxyzf012345-ab length 0 [was in cache]

Client receives response header:

HTTP/1.1 200 OK
Content-Length: 99

This is the data that is in the test file
It is only a few lines of data
To make sure that it sends

PUT(second time):

curl -T test.txt http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab -v

Client receives response header:

HTTP/1.1 200 Created
Content-Length: 0

logfile.txt content:

PUT ABCDEFarqdeXYZxyzf012345-ab length 112 [was not in cache]
0000000 54 68 69 73 20 69 73 20 74 68 65 20 64 61 74 61 20 74 68 61
0000020 74 20 69 73 20 69 6E 20 74 68 65 20 74 65 73 74 20 66 69 6C
0000040 65 0A 49 74 20 69 73 20 6F 6E 6C 79 20 61 20 66 65 77 20 6C
0000060 69 6E 65 73 20 6F 66 20 64 61 74 61 0A 54 6F 20 6D 61 6B 65
0000080 20 73 75 72 65 20 74 68 61 74 20 69 74 20 73 65 6E 64 73 0A
0000100 0A 54 45 53 54 20 46 49 4C 45 20 30
========

2. Running curl with invalid filenames (short)
PUT:

curl -T test.txt http://localhost:8080 --request-target ABCDEF -v

logfile.txt content:

FAIL: PUT ABCDEF HTTP/1.1 --- response 400

========

GET:

curl http://localhost:8080 --request-target ABCDEF -v

logfile.txt content:

FAIL: GET ABCDEF HTTP/1.1 --- response 400

========

Client receives response header:

HTTP/1.1 400 Bad Request

Content-Length: 0

3. Running curl with invalid filenames (long)

PUT:

curl -T test.txt http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab99999999 -v

logfile.txt content:

FAIL: PUT ABCDEFarqdeXYZxyzf012345-ab99999999 HTTP/1.1 --- response 400

========

GET:

curl http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab99999999 -v

logfile.txt content:

FAIL: GET ABCDEFarqdeXYZxyzf012345-ab99999999 HTTP/1.1 --- response 400

========

Client receives response header:

HTTP/1.1 400 Bad Request

Content-Length: 0

3. Running curl with invalid filenames (invalid characters)

PUT:

curl -T test.txt http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-@! -v

logfile.txt content:

FAIL: PUT ABCDEFarqdeXYZxyzf012345-@! HTTP/1.1 --- response 400

========

GET:

curl http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-@! -v

logfile.txt content:

FAIL: GET ABCDEFarqdeXYZxyzf012345-@! HTTP/1.1 --- response 400
========

Client receives response header:

HTTP/1.1 400 Bad Request
Content-Length: 0

4. Running curl with POST (method other than PUT or GET):

POST:

curl -X POST test.txt http://localhost:8080 --request-target
ABCDEFarqdeXYZxyzf012345-ab -v

logfile.txt content:

FAIL: POST ABCDEFarqdeXYZxyzf012345-ab HTTP/1.1 --- response 500
========

Client receives response header:

HTTP/1.1 500 Internal Server Error
Content-Length: 0

5. Running curl with missing file:

GET:

curl http://localhost:8080 --request-target NOTREALrqdeXYZxyzf012345-ab -v

logfile.txt content:

FAIL: GET NOTREALrqdeXYZxyzf012345-ab HTTP/1.1 --- response 404
========

Clients receives response header:

HTTP/1.1 404 Not Found
Content-Length: 0

6. Running curl with private file:

Make file private:

chmod a-r ABCDEFarqdeXYZxyzf012345-ab

GET:

curl http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab99999999 -v

logfile.txt content:

FAIL: GET ABCDEFarqdeXYZxyzf012345-ab99999999 HTTP/1.1 --- response 403
========

Client receives response header:

HTTP/1.1 403 Forbidden
Content-Length: 0

PUT:

curl http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab -v

logfile.txt content:

FAIL: GET ABCDEFarqdeXYZxyzf012345-ab99999999 HTTP/1.1 --- response 400
========

Client receives response header:

HTTP/1.1 403 Forbidden
Content-Length: 0

Questions:

Using your new httpserver with caching, perform an experiment to demonstrate caching can improve performance (latency and/or throughput) Do a test with caching turned on and compare it with the same test but with caching turned off.

The test I performed was running 4 PUT commands with 4 different files
command to PUT requests
curl -T test1.txt http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab -v &
curl -T test2.txt http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012346-ab -v &
curl -T test3.txt http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012347-ab -v &
curl -T test4.txt http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012348-ab -v

command to GET requests
time curl http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012345-ab -v &
curl http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012346-ab -v &
curl http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012347-ab -v &
curl http://localhost:8080 --request-target ABCDEFarqdeXYZxyzf012348-ab -v

with caching
real 0m0.061s
user 0m0.008s
sys 0m0.011s

without caching
real 0m0.120s
user 0m0.010s
sys 0m0.012s

The results show that httpserver ran with caching provides a 2x speedup, and that overall the performance is faster than without running caching.