Joseph Shumway

2/14/2023

CSCE 463

HW 1.3 Report

1. My code is built using different classes that perform their functions on their own. I did my best to modularize my code from the start so that it can easily be modified and rearranged.

Main – creates file reader, sets up URLs vector, creates the Crawler, and tells it to start threads

FileReader – reads input file and fills a vector array with each URL

Crawler – holds shared memory and creates threads for URLProcessor and parser

URLProcessor – does the dirty work of HW1 (parse URL, DNS, Connect, load, etc)

HTMLParser – supplied parser

Utility – helper class to keep main clean. Used with FileReader to supply URLs to Crawler

Lessons learned: going low level in C is not always the optimal approach. Std::vector was much better than I thought it would be and solved a lot of issues. Also, waitForMultipleObjects did not work as I thought and caused a lot of debugging red herrings. I ended up using a for loop with WaitForSingleObject instead.

Trace:

```
Opened URL-input-1M.txt with size 66152005
[  2] 5000 Q986880 E    7106 H  7084 D  5555 I 5555 R    0 C    0 L   0
        *** crawling 0.0 pps @ 0.0 Mbps

[  4] 5000 Q981920 E   12066 H 12001 D 10755 I10755 R  310 C    0 L   0
        *** crawling 0.0 pps @ 3.7 Mbps

[  6] 5000 Q976134 E   17852 H 17707 D 15934 I15934 R  748 C   52 L   4K
        *** crawling 26.0 pps @ 14.3 Mbps

[  8] 5000 Q970214 E   23772 H 23442 D 21089 I21089 R 1233 C  135 L  11K
        *** crawling 41.5 pps @ 18.7 Mbps

[ 10] 5000 Q964202 E   29784 H 29242 D 26346 I26346 R 1622 C  206 L  20K
        *** crawling 35.5 pps @ 21.4 Mbps

[ 12] 5000 Q958008 E   35978 H 35151 D 31705 I31705 R 1966 C  267 L  27K
        *** crawling 30.5 pps @ 16.6 Mbps

[ 14] 5000 Q951949 E   42037 H 40925 D 36944 I36944 R 2351 C  325 L  34K
        *** crawling 29.0 pps @ 16.2 Mbps

[ 16] 5000 Q945741 E   48245 H 46879 D 42293 I42293 R 2684 C  424 L  40K
        *** crawling 49.5 pps @ 21.9 Mbps

[ 18] 5000 Q939606 E   54380 H 52730 D 47610 I47610 R 2971 C  503 L  58K
        *** crawling 39.5 pps @ 25.8 Mbps
```

```
[ 20] 5000 Q933430 E   60556 H 58644 D 52994 I52994 R 3268 C  583 L  72K
       *** crawling 40.0 pps @ 18.5 Mbps

[ 22] 5000 Q927317 E   66669 H 64508 D 58358 I58358 R 3574 C  666 L  82K
       *** crawling 41.5 pps @ 20.0 Mbps

[ 24] 5000 Q921222 E   72764 H 70404 D 63796 I63796 R 3845 C  731 L  87K
       *** crawling 32.5 pps @ 12.1 Mbps

[ 26] 5000 Q915174 E   78812 H 76246 D 69236 I69236 R 4132 C  786 L  98K
       *** crawling 27.5 pps @ 16.1 Mbps

[ 28] 5000 Q909177 E   84809 H 82044 D 74653 I74653 R 4420 C  845 L 100K
       *** crawling 29.5 pps @ 9.1 Mbps

[ 30] 5000 Q903197 E   90789 H 87792 D 80032 I80032 R 4764 C  900 L 107K
       *** crawling 27.5 pps @ 12.8 Mbps

[ 32] 5000 Q898138 E   95848 H 92650 D 84550 I84550 R 5037 C  946 L 117K
       *** crawling 23.0 pps @ 13.9 Mbps

[ 35] 5000 Q832999 E  160987 H152310 D137028 I137028 R 5366 C 1001 L 128K
       *** crawling 18.3 pps @ 12.9 Mbps

[ 37] 5000 Q772319 E  221667 H201278 D180599 I180599 R 5392 C 1001 L 128K
       *** crawling 0.0 pps @ 0.2 Mbps

[ 39] 5000 Q707460 E  286526 H258514 D232642 I232642 R 5426 C 1003 L 128K
       *** crawling 1.0 pps @ 1.4 Mbps

[ 41] 5000 Q664721 E  329265 H296365 D266875 I266875 R 5438 C 1003 L 128K
       *** crawling 0.0 pps @ 0.1 Mbps

[ 43] 5000 Q627516 E  366470 H327793 D295290 I295290 R 5445 C 1003 L 128K
       *** crawling 0.0 pps @ 0.1 Mbps

[ 45] 5000 Q559511 E  434475 H380055 D343293 I343293 R 5446 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 47] 5000 Q529601 E  464385 H405386 D366187 I366187 R 5446 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 49] 5000 Q510538 E  483448 H422536 D381357 I381357 R 5446 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 51] 5000 Q493098 E  500888 H438063 D394580 I394580 R 5446 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 53] 5000 Q481121 E  512865 H447388 D402671 I402671 R 5446 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 56] 5000 Q449389 E  544597 H475616 D427437 I427437 R 5446 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 59] 5000 Q423236 E  570750 H499746 D448974 I448974 R 5446 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 61] 5000 Q403215 E  590771 H518073 D464672 I464672 R 5446 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 63] 5000 Q374283 E  619703 H543729 D486625 I486625 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 66] 5000 Q341282 E  652704 H573635 D510674 I510674 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 68] 5000 Q299066 E  694920 H611458 D540785 I540785 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 70] 5000 Q266155 E  727831 H641506 D563069 I563069 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps
```

```
[ 72] 5000 Q249058 E  744928 H657778 D576010 I576010 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 74] 5000 Q229831 E  764155 H675639 D589325 I589325 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 77] 5000 Q211674 E  782312 H692776 D602075 I602075 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 80] 5000 Q190275 E  803711 H712530 D616628 I616628 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 83] 5000 Q167658 E  826328 H733600 D632565 I632565 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 86] 5000 Q150525 E  843461 H749502 D644563 I644563 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 89] 5000 Q131981 E  862005 H766387 D657433 I657433 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 92] 5000 Q109021 E  884965 H787629 D672891 I672891 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 94] 5000 Q 86057 E  907929 H809005 D689188 I689188 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 97] 5000 Q 68358 E  925628 H825620 D702171 I702171 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[ 99] 5000 Q 50667 E  943319 H842548 D715291 I715291 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[102] 5000 Q 24845 E  969141 H867189 D733824 I733824 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[104] 4971 Q     0 E  993986 H891075 D752542 I752542 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[108] 549 Q     0 E  993986 H891079 D755536 I755536 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[111] 322 Q     0 E  993986 H891079 D755544 I755544 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[113] 222 Q     0 E  993986 H891079 D755545 I755545 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[115] 130 Q     0 E  993986 H891079 D755545 I755545 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[117]  10 Q     0 E  993986 H891079 D755545 I755545 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

[119]   0 Q     0 E  993986 H891079 D755545 I755545 R 5447 C 1003 L 128K
       *** crawling 0.0 pps @ 0.0 Mbps

Extracted 993986 URLs @ 8352/s
Looked up 891079 DNS names @ 7488/s
Attempted 755545 site robots @ 6349/s
Crawled 1003 pages @ 8/s (51.88 MB)
Parsed 128957 links @ 1083/s
HTTP codes: 2xx = 1003, 3xx = 1466, 4xx = 2132, 5xx = 27, other = 1
```

2. 1,003 pages with status code 2xx contained 128,957 links. This means there are on average 129 links per page found by the parser. For Google's web graph, from 1 trillion pages at the same ratio of links per page, you can expect 129 Trillion links/edges. Since each edge is a 64-bit hash, that's a total of 129T x 16 bytes = 2064 Terabytes. For the URL hashes stored at 64 bits each, it would be 1 Trillion x 16 bytes = 16TB. Total size = 2080 Terabytes
3. The average page size for all pages (assuming Google was able to crawl all pages) would be pulled from our successful crawls: 1003 pages / 51.88MB = 20MB. The bandwidth needed for 10 Billion pages/day would have to be: 10 Billion x 20MB = 200 Petabytes per day = 18,519 Gigabits per second.
4. The probability that an input link contains a unique host is: 891,079 / 993,986 = 89%
   The probability that a unique host has a valid DNS is: 755545 / 891079 = 85%
   The percentage of contacted sites with a 4XX robots.txt file is:
   (1003 + 1466 + 2132 + 27 + 1) / 755545 = 0.6%
   (Any sites that had their HTTP codes checked a second time in the output passed robots 4XX)
5. I was unable to get a single page to load that also had a TAMU link in it, most likely since most of these sites have moved or required HTTPS. I did some manual testing as well and sure enough I couldn't find a single page that had TAMU as a link that would also pass robots and page tests, but I was able to count other links such as "pressthink.org". I can however tell you what I did to try and get these values. I brought in the char* buffer from HTMLparser and used strstr() to find the first occurrence of "tamu.edu", then checked that the next character was '/', '\r', or '\n' (this got rid of false positives like "tamu.edu.org." I also checked if the current host was tamu.edu and InterlockedIncremented a counter for that, and a separate one for if it was not. With these I would be able to output at the end of the trace how many TAMU links parsed were from TAMU and how many were from outside sources.

Thanks for reading!