

1. Code architecture and Lessons learned

Architecture: The web crawler designed for this project is a multi-threaded application capable of processing multiple URLs concurrently to efficiently crawl web content. Here's a breakdown of its core components and functionality.

- **URL Management:** URLs are managed through a queue system where each thread can pull tasks independently, ensuring efficient distribution of work.
- **HTTP Communications:** The crawler constructs HTTP requests to fetch pages and handles responses, parsing the headers and content.
- **DNS Resolution:** Incorporates DNS lookup to resolve hostnames into IP addresses, which is crucial for network requests.
- **Concurrency Control:** Utilizes mutexes and atomic operations to manage concurrent access to shared resources like URL lists and counters, preventing race conditions and ensuring data integrity.
- **Error Handling:** Robust error handling mechanisms to manage network errors, invalid responses, and other runtime exceptions.
- **Statistics Tracking:** Real-time tracking of various metrics such as number of pages crawled, number of links found, HTTP status codes encountered and others.

Key Components:

- **Robots.txt Compliance:** Checks each domain for a robots.txt file to ensure the crawler respects site-specific scraping rules.
- **Link Extraction:** Uses an HTML parsing library to extract links from each webpage, which are then processed and added to the queue if they have not been visited.
- **Resource Management:** Handles network connections and system resources efficiently, ensuring that sockets are properly managed and closed after transactions to prevent resource leaks.

Lessons Learned:

- **Multi-threading Efficiency:** Learned the importance of balancing the number of threads to optimize the use of CPU and network resources without overwhelming the web server or causing contention issues.
- **Error Handling:** Reinforced the need for comprehensive error handling in network applications to manage unpredictable network behavior and server responses.
- **Data Integrity:** The critical role of synchronization mechanisms in preventing data corruption and ensuring the accuracy of shared data among threads.
- **Resource Management:** Gained insights into the management of system resources, particularly in network programming where proper handling of sockets and responses is crucial for maintaining crawler stability and performance.

- **Compliance and Ethics:** Understood the importance of adhering to web standards and ethical scraping guidelines, such as respecting robots.txt and managing request rates to avoid negatively impacting the performance of target websites.

The following figures show a complete trace of 1M URLs input.

```

Opened URL-input-1M.txt with size 66152005 bytes
[ 2]5000 Q947934 E 52070 H 8436 D 3212 I 1983 R 322 C 198 L 1K
    *** crawling 98.8 pps @ 8.1 Mbps
[ 4]5000 Q910034 E 89971 H 13654 D 8122 I 5212 R 833 C 697 L 11K
    *** crawling 247.6 pps @ 32.2 Mbps
[ 6]5000 Q871215 E 128790 H 19255 D 12965 I 8473 R 1382 C 1260 L 24K
    *** crawling 279.1 pps @ 35.3 Mbps
[ 8]5000 Q835592 E 164415 H 24639 D 17850 I11737 R 1964 C 1837 L 44K
    *** crawling 286.8 pps @ 42.3 Mbps
[10]5000 Q790821 E 209183 H 30357 D 23078 I15204 R 2522 C 2387 L 57K
    *** crawling 274.3 pps @ 40.8 Mbps
[12]5000 Q755620 E 244435 H 35318 D 27207 I17839 R 3008 C 2898 L 67K
    *** crawling 255.0 pps @ 39.7 Mbps
[14]5000 Q724893 E 275115 H 40515 D 31865 I20904 R 3438 C 3336 L 74K
    *** crawling 216.9 pps @ 28.8 Mbps
[16]5000 Q692582 E 307422 H 46085 D 36841 I24048 R 3944 C 3811 L 84K
    *** crawling 236.4 pps @ 35.9 Mbps
[18]5000 Q665848 E 334163 H 50468 D 40580 I26426 R 4387 C 4279 L 93K
    *** crawling 232.8 pps @ 27.8 Mbps
[20]5000 Q638264 E 361740 H 54804 D 44364 I28806 R 4786 C 4654 L 101K
    *** crawling 185.9 pps @ 20.3 Mbps
[22]5000 Q601391 E 398613 H 59380 D 48286 I31315 R 5206 C 5093 L 110K
    *** crawling 219.3 pps @ 30.1 Mbps
[24]5000 Q554067 E 445940 H 65384 D 53404 I34391 R 5706 C 5573 L 121K
    *** crawling 239.0 pps @ 28.4 Mbps
[26]5000 Q512043 E 487961 H 70664 D 57916 I37137 R 6112 C 5988 L 130K
    *** crawling 206.1 pps @ 37.0 Mbps
[28]5000 Q456151 E 543853 H 76568 D 62858 I40031 R 6580 C 6436 L 143K
    *** crawling 222.2 pps @ 34.6 Mbps
[30]5000 Q414584 E 585579 H 82835 D 68155 I42967 R 7048 C 6908 L 152K
    *** crawling 234.6 pps @ 30.6 Mbps
[32]5000 Q370447 E 629557 H 88868 D 73053 I45133 R 7472 C 7338 L 166K
    *** crawling 213.3 pps @ 27.4 Mbps
[34]5000 Q321604 E 678416 H 95754 D 78672 I47290 R 7826 C 7715 L 179K
    *** crawling 186.0 pps @ 24.2 Mbps
[36]5000 Q274362 E 725680 H102612 D 84203 I49392 R 8215 C 8096 L 188K
    *** crawling 190.1 pps @ 30.6 Mbps
[38]5000 Q233353 E 766660 H108351 D 88790 I51052 R 8516 C 8415 L 197K
    *** crawling 157.0 pps @ 26.6 Mbps
[40]5000 Q193657 E 806347 H114218 D 93569 I52628 R 8773 C 8680 L 203K
    *** crawling 130.3 pps @ 18.4 Mbps
[42]5000 Q143507 E 856497 H121359 D 99368 I54634 R 9126 C 9024 L 212K
    *** crawling 170.5 pps @ 24.2 Mbps
[44]5000 Q 95258 E 904753 H127970 D104507 I56507 R 9441 C 9328 L 219K
    *** crawling 151.6 pps @ 23.1 Mbps
[46]5000 Q 50166 E 949865 H134463 D109528 I58266 R 9737 C 9628 L 225K
    *** crawling 148.6 pps @ 29.4 Mbps
[48]4228 Q      E1000004 H139301 D113691 I59755 R10009 C 9908 L 236K
    *** crawling 139.1 pps @ 22.4 Mbps
[50]4227 Q      E1000004 H139301 D113691 I59755 R10009 C 9908 L 236K
    *** crawling 0.0 pps @ 0.0 Mbps

Extracted 1000004 URLs @ 20687.7/s
Looked up 139301 DNS names @ 2881.8/s
Attempted 59755 robots @ 1236.2/s
Crawled 9908 pages @ 205.0/s (175MB)
Parsed 236220 links @ 4886.8/s
HTTP codes: 2xx = 5095, 3xx = 1300, 4xx = 3394, 5xx =114, other = 6

```

2. Calculate the average number of HTML links per 2xx page

Average numofLinks per 2xx page= $L/2xx \text{ pages} = 236020/5095 = 46.32 \text{ L/Page}$

Edges per Page = Average Links per Page

Total Edges = Total Pages×Edges per Page

Storage per Edge = 64 bits per link=8 bytes(since1byte=8bits)

Total Storage = Total Edges × 8 bytes

- **Average Number of HTML Links per Page with a 2xx Status Code:** The average number of links per page for pages that successfully returned a 2xx HTTP status code is approximately **46.32 links per page**.
- **Estimate of the Size of Google's Webgraph:**
 - **Total Edges in the Webgraph:** Given the average of 46.32 links per page and assuming Google crawls 1 trillion pages, the total number of edges is approximately $46.32 \times 1 \text{ trillion} \approx 46.32 \text{ trillion}$.
 - **Total Storage Required for the Webgraph:** If each edge requires 8 bytes (since each URL/link is represented by a 64-bit hash), the total storage needed is approximately **370.59 petabytes**.

3. Determine the average page size and estimate bandwidth needs

- Crawled 9908 pages
- Total data downloaded: 175 MB.
- Average Page Size: Total data downloaded/ Crawled pages = 0.018 MB/Page
- To estimate the bandwidth needed to crawl 10 billion pages in one day:
 - Bandwidth in Gbps = Total Data in Gigabits/Seconds in a day; and convert Mbps to Gbps
 - Bandwidth in Gbps = $0.018 \text{ MB per Page} * 10 * 10^6 * 8 / (86400 * 1024) = 16.28 \text{ Gbps}$

4. Probability analyses

Probability of a link containing a unique host

- Total number of links in the input file = 1000004
- Total number of unique hosts = 139301
 - probability that a link contains a unique host = $139301/1000004 = 0.139$

Probability that a unique host has a valid DNS record

- Total number of unique hosts = 139301
- Number of unique hosts that successfully resolved a DNS query = 113691
 - probability that a unique host has a valid DNS record = $113691/139301 = 0.82$

Percentage of contacted sites that had a 4xx robots file

- Total number of sites where robots.txt was requested (passed unique IP check) = 59755
- Number of those requests that returned a 4xx status = 10009

- percentage of contacted sites had a 4xx robots file = $10009/59755 = 0.17$

5. Hyperlink analysis to TAMU domain (1M URLs)

- Hyperlinks to TAMU**
 - The number of 2xx pages that contain at least one hyperlink to 'tamU.edu': 3.
 - The number of pages hosted outside the TAMU network: 3.

The following Figure show the output after adding the (Hyperlink Analysis of TAMU domain)

```
Extracted 1000004 URLs @ 15534.0/s
Looked up 139301 DNS names @ 2163.9/s
Attempted 59758 robots @ 928.3/s
Crawled 9907 pages @ 153.9/s (172MB)
Parsed 214786 links @ 3336.5/s
HTTP codes: 2xx = 5039, 3xx = 1313, 4xx = 3436, 5xx = 112, other = 7
Total Pages with valid TAMU links found: 3
Total Pages with valid TAMU links originating from outside TAMU: 3
```

- Regex Matching:** Introduced a regular expression to identify links that point to the Texas A&M University (TAMU) domain (tamU.edu). The regex checks for URLs that strictly belong to the TAMU domain, ensuring it captures the right format and excludes subdomains or similar patterns not exactly matching tamU.edu.
- Checking Origin:** To determine whether a hyperlink originates from outside the TAMU network, a condition was created where the crawler checks the host against the TAMU domain using the same regex. If the host of the page being crawled does not match the TAMU regex, it is counted as an external link.

The following figure illustrates the changes in the code.

```
regex tamuRegex(R"(^http://(?:.*\.)?tamU\.edu\b(?:\/.*)?)");
for (int i = 0; i < nLinksPage; i++) {
    string link(linkBufferPage);
    if (regex_match(link, tamuRegex)) {
        stats.incrementPagesWithValidTAMULinks();
        string origin = "http://" + parts.host;
        if (!regex_match(origin, tamuRegex)) {
            stats.incrementExternalOriginTAMULinks();
        }
        break;
    }
    linkBufferPage += strlen(linkBufferPage) + 1;
}
```