MULTI-THREADED WEB CRAWLER

PROJECT REPORT

Submitted by

M.DURGA PRASAD(RA2211003011093)

T.TABISH(RA2211003011087)

M.SAI MUKESH(RA2211003011076)

Under the guidance of

Dr.M.VIJAYA LAKSHMI

(Assistant Professor, Department of Computing Technologies)

21CSC202J - OPERATING SYSTEMS

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

SEP 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR- 603 203

BONAFIDE CERTIFICATE

Certified that this B. Tech project report titled **"MULTI-THREADED WEB CRAWLER"** is the bonafide work of **M.DURGA PRASAD [RA2211003011093],T. TABISH [RA2211003011087]** andM. **SAI MUKESH [RA2211003011076]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation based on which a degree or award was conferred on an earlier occasion for this or any other candidate.

Department of Computing Technologies
Kattankulathur
603 203

M. Pushpalatha

DR. M. VIJAYA LAKSHMI
Assistant Professor
Department of Computing Technologies
SRMIST
KATTANKULATHUR

DR. PUSPALATHA
Head of the Department
Department of Computing Technologies
SRMIST
KATTANKULATHUR

Department of Computing Technologies

SRM Institute of Science and

Technology

Own Work Declaration Form

Degree/Course: B. Tech in Computer Science and Engineering

Student Names: M. Durga Prasad, T. Tabish, M. Sai Mukesh

Registration Number: RA2211003011093, RA2211003011087, RA2211003011076

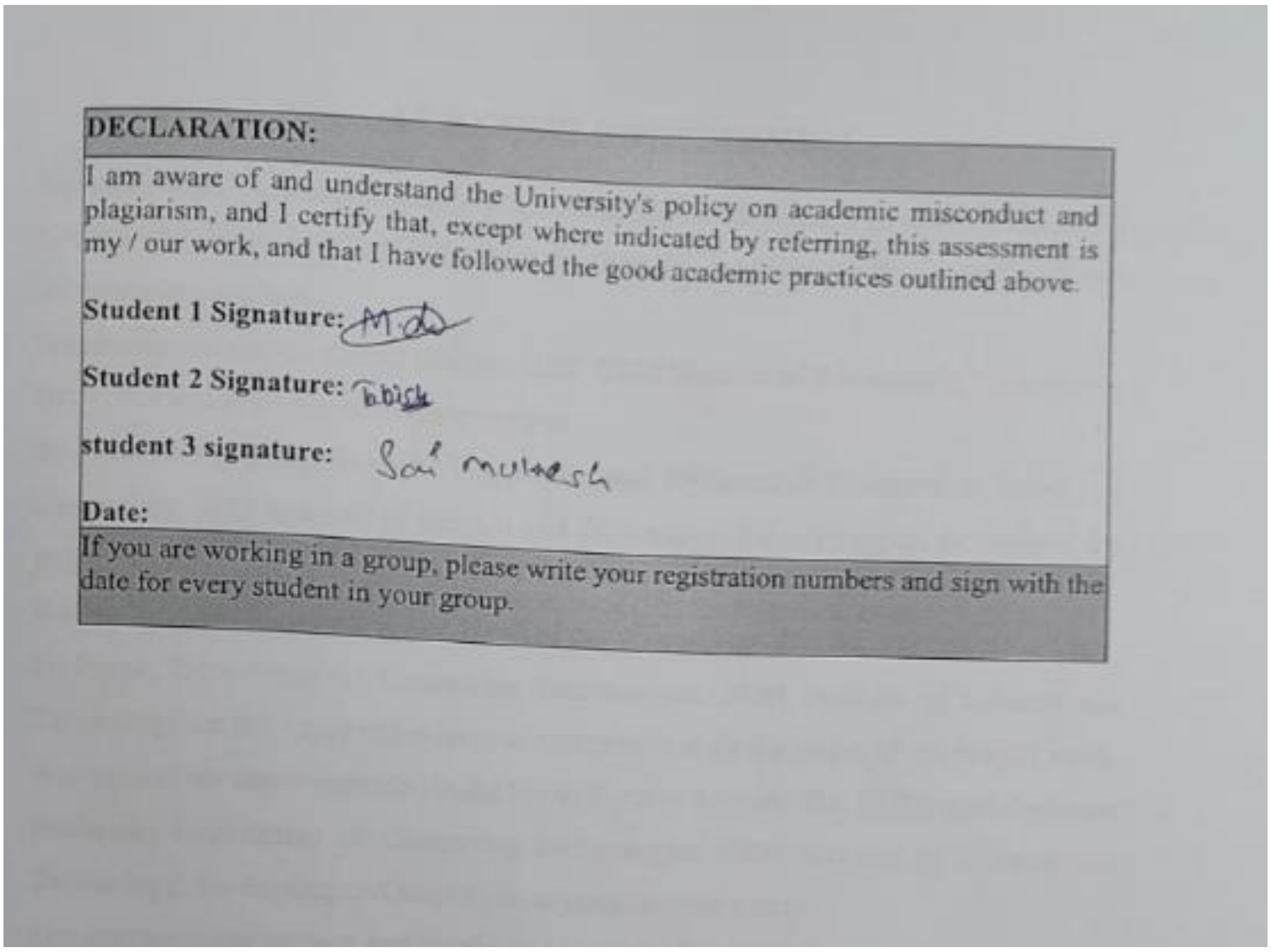
Title of Work: MULTI-THREADED WEB CRAWLERS

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- References/lists all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data, etc. that are not my own.
- Not making any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g., fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the course handbook/ university website.

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.



DECLARATION:

I am aware of and understand the University's policy on academic misconduct and plagiarism, and I certify that, except where indicated by referring, this assessment is my / our work, and that I have followed the good academic practices outlined above.

Student 1 Signature: [illegible]

Student 2 Signature: [illegible]

student 3 signature: Sai mukesh

Date:

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T.V. Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the projectwork.

We are incredibly grateful to our Head of the Department, **Dr. M. PUSHPALATHA,** Professor, Department of Computing Technologies, SRM Institute of Science and Technology,for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor, **Dr.** B. Kanisha Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. M. Vijaya Lakshmi** Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She providedus with the freedom and support to explore the research topics of our interest. Her passion forsolving problems and making a difference in the world has always been inspiring.

We sincerely thank the Computing Technologies Department staff and students, at SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

M.DURGA PRASAD (RA2211003011093)
T. TABISH (RA2211003011087)
M.SAI MUKESH (RA2211003011076)

TABLE OF CONTENTS

ABSTRACT

This project describes mutual exclusion principle for multithreaded web crawlers. The existing web crawlers use data structures to hold frontier set in local address space. This space could be used to run more crawler threads for faster operation. All crawler threads fetch the URL to crawl from the centralized frontier. The mutual exclusion principle is used to provide access to frontier for each crawler thread in synchronized manner to avoid deadlock. The approach to utilize the waiting time on mutual exclusion lock in efficient manner has been discussed in detail. Multithreaded web crawlers have become indispensable tools for collecting vast amounts of data from the World Wide Web. However, the concurrent nature of these crawlers poses significant challenges, particularly when multiple threads attempt to access and manipulate shared resources, such as URL queues, visited page databases, and other critical data structures. To ensure the integrity of data and prevent race conditions, the mutual exclusion principle plays a pivotal role in designing efficient and robust multithreaded web crawlers. This abstract introduces various mutual exclusion techniques such as locks, semaphores, and mutexes, providing a brief overview of their application in web crawler development. The abstract includes references to practical examples and best practices from well-established web crawler frameworks, showcasing how mutual exclusion principles are applied in the field. Understanding and implementing mutual exclusion principles in multithreaded web crawlers is essential for ensuring data consistency, reducing contention, and improving overall crawling efficiency. This abstract serves as a starting point for developers and researchers aiming to address these critical concerns in the context of web crawling, promoting effective and reliable data collection from the web Crawlers.

1.INTRODUCTION

1.1 GENERAL

Web crawlers, often referred to as robots or spiders, play a pivotal role in the functioning of search engines by navigating the vast and ever-expanding World Wide Web. These programs exploit the graph structure of the internet to efficiently gather information. At the heart of any search engine lies an efficient crawler, crucial for providing accurate and timely search results. The rapid growth of the World Wide Web necessitates the adoption of fast and effective crawling processes to meet the demands of modern search engines.

1.2 PURPOSE

The primary purpose of web crawlers is to systematically explore the web, collecting data from various websites. The efficiency and speed of these crawlers are paramount for search engines to deliver high-quality results to users. Also known as spiders or robots, these crawlers operate in a multithreaded manner, initializing data structures like queues to manage the list of URLs to be visited. Load balancing becomes a critical aspect to ensure that resources are utilized efficiently across multiple crawler threads. The ultimate goal is to extract valuable metadata, such as body text, title, and link text, from web pages. This metadata is then used by indexers to rank pages, forming the basis for search engine results.

1.3 SCOPE

In the era of big data and information-driven decision-making, the ability to harness the vast resources of the web is of paramount importance. A solid understanding of mutual exclusion principles in multithreaded web crawling is essential for engineers, developers, and researchers. This knowledge ensures the development of effective and reliable web crawling systems capable of navigating the complexities of the modern web while maintaining data integrity and consistency. This exploration into mutual exclusion principles serves as a foundational step toward building robust web crawling systems, addressing challenges posed by multithreading and incorporating real-world examples and best practices.

2.LITERATURE REVIEW

TITLE	PUBLISHER	ALGORITHM USED	DATA SET USED	INFERENCE
The Anatomy of a Search Engine	" Seventh International World Wide Web Conference (WWW98), Brisbane, Australia"	PageRank	" Web crawling, Indexing, and Ranking"	"This paper introduced the foundational concepts behind Google's search engine, including the PageRank algorithm, which played a pivotal role in revolutionizing web search. It is a seminal work in the field of web search and information retrieval."
A World Wide Web Resource Discovery System	"Budi Yuwono, Savio L. Lam, Jerry H. Ying, Dik L. Lee"	Vector Space Model	Resource discovery on the World Wide Web.	" The early works in the field of web resource discovery. It might discuss techniques for finding and cataloging web resources, which were fundamental aspects of early web crawling and indexing"

3.PROPOSED METHODOLOGY

3.1 Problem Analysis and Specification:

Define the objectives and requirements of the web crawler, including the specific data to be collected and the desired performance metrics. Identify potential challenges related to concurrent access to shared resources.

3.2 Thread Design and Resource Identification:

Determine the number of threads to be used in the web crawler, considering factors like CPU cores and system resources. Identify shared resources, such as URL queues, visited page databases, and data structures that need to be accessed and manipulated by multiple threads.

3.3 Concurrency Control Mechanisms:

Select appropriate concurrency control mechanisms for managing shared resources. Common options include locks, semaphores, mutexes, and reader-writer locks. Design a synchronization strategy that ensures mutual exclusion while minimizing contention among threads.

3.4 Data Structure Selection:

Choose or design data structures that are thread-safe and optimized for concurrent access. This may include concurrent queues, thread-safe dictionaries, and databases designed for multithreaded environments.

3.5 Thread Synchronization:

Implement thread synchronization by encapsulating critical sections of code with appropriate synchronization primitives. Ensure that only one thread can access and modify shared resources at a time to prevent data corruption and race conditions.

3.6 Error Handling and Recovery:

Develop error-handling mechanisms to address exceptions and failures in multithreaded operations, such as handling network errors and server responses gracefully. Implement mechanisms for thread recovery and resilience in case of thread crashes or interruptions.

3.7 Load Balancing:

Implement load balancing strategies to distribute work evenly among threads. Ensure that no single thread is overwhelmed with tasks while others remain idle.

3.8 Performance Optimization:

Continuously monitor and profile the web crawler's performance, identifying bottlenecks and areas for improvement. Implement performance optimizations, such as parallel processing of tasks and asynchronous I/O operations, to enhance the overall efficiency of the web crawler.

3.9 Testing and Validation:

Thoroughly test the multithreaded web crawler using a range of scenarios and datasets to ensure that mutual exclusion is effectively maintained. Conduct stress testing to evaluate the crawler's performance and robustness under heavy workloads.

3.10 Deployment and Scaling:

Deploy the web crawler in a production environment, ensuring that it can scale horizontally to handle large-scale crawling tasks. Monitor the deployed crawler and make necessary adjustments to accommodate changing workloads and web conditions.

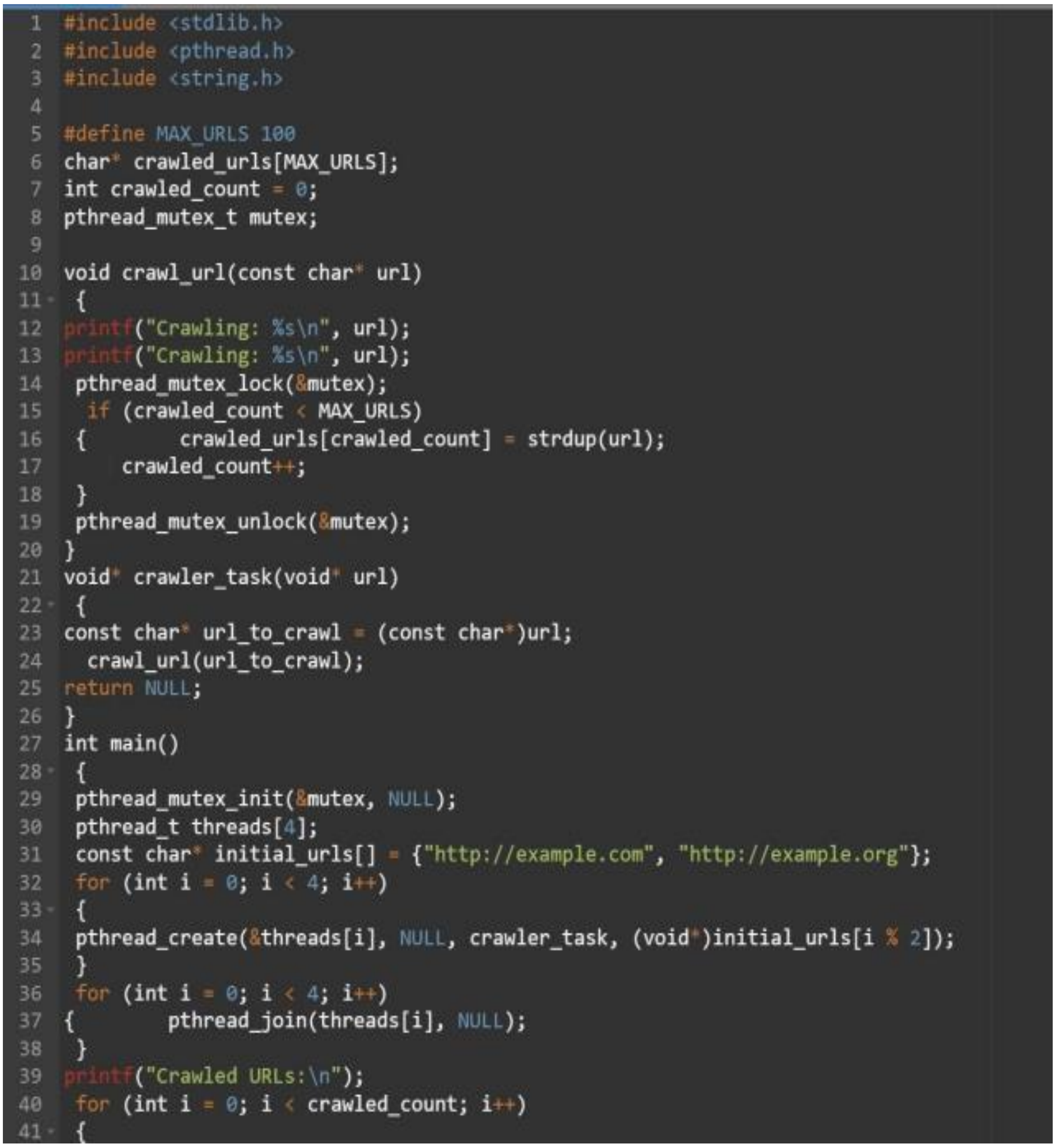
3.11 Documentation and Knowledge Sharing:

Document the implementation, synchronization techniques, and best practices followed in the project. Share knowledge and findings with the development team to promote a deeper understanding of mutual exclusion principles.

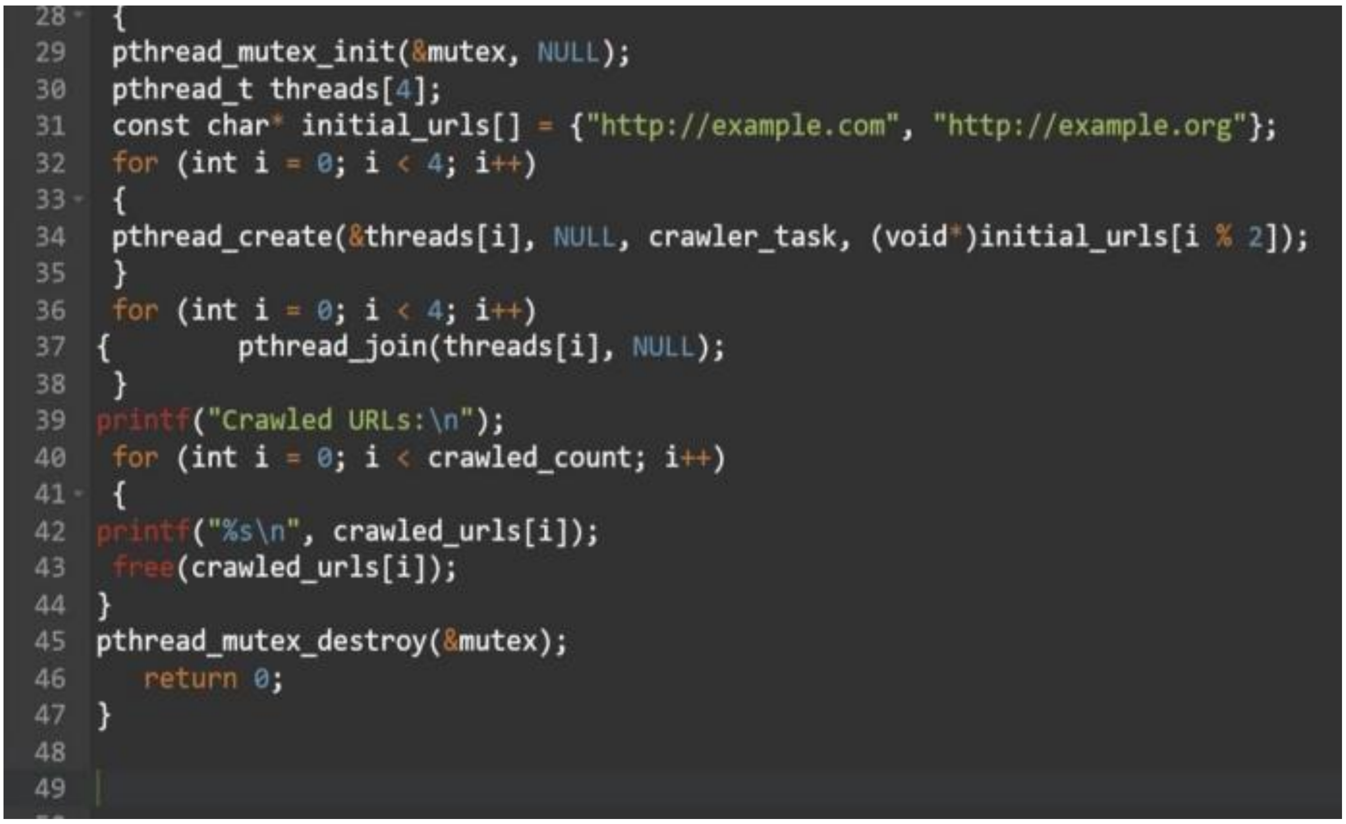
3.12 Ongoing Maintenance and Improvement:

Continue to maintain and improve the web crawler as the web landscape evolves. Address any emerging challenges and adapt to changes in web technologies. By following this proposed methodology, developers can create a multithreaded web crawler that effectively implements mutual exclusion principles, ensuring the reliable and efficient collection of web data while minimizing data integrity risks associated with concurrent access to shared resources.

4.IMPLEMENTATION

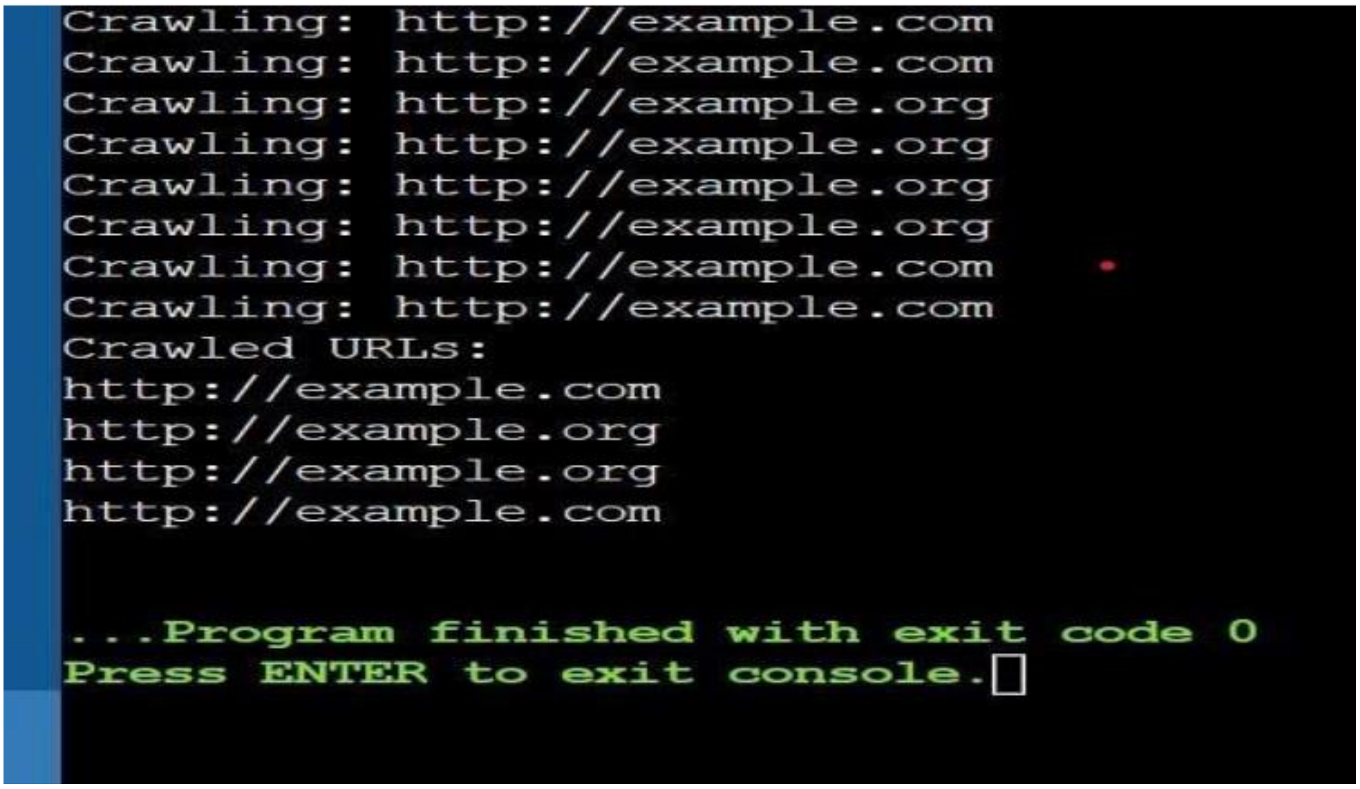


```
#include <stdlib.h>
#include <pthread.h>
#include <string.h>

#define MAX_URLS 100
char* crawled_urls[MAX_URLS];
int crawled_count = 0;
pthread_mutex_t mutex;

void crawl_url(const char* url)
 {
printf("Crawling: %s\n", url);
printf("Crawling: %s\n", url);
 pthread_mutex_lock(&mutex);
  if (crawled_count < MAX_URLS)
 {        crawled_urls[crawled_count] = strdup(url);
     crawled_count++;
 }
 pthread_mutex_unlock(&mutex);
}
void* crawler_task(void* url)
 {
const char* url_to_crawl = (const char*)url;
  crawl_url(url_to_crawl);
return NULL;
}
int main()
 {
 pthread_mutex_init(&mutex, NULL);
 pthread_t threads[4];
 const char* initial_urls[] = {"http://example.com", "http://example.org"};
 for (int i = 0; i < 4; i++)
 {
 pthread_create(&threads[i], NULL, crawler_task, (void*)initial_urls[i % 2]);
 }
 for (int i = 0; i < 4; i++)
{        pthread_join(threads[i], NULL);
 }
printf("Crawled URLs:\n");
 for (int i = 0; i < crawled_count; i++)
 {
```



```
 {
 pthread_mutex_init(&mutex, NULL);
 pthread_t threads[4];
 const char* initial_urls[] = {"http://example.com", "http://example.org"};
 for (int i = 0; i < 4; i++)
 {
 pthread_create(&threads[i], NULL, crawler_task, (void*)initial_urls[i % 2]);
 }
 for (int i = 0; i < 4; i++)
{        pthread_join(threads[i], NULL);
 }
printf("Crawled URLs:\n");
 for (int i = 0; i < crawled_count; i++)
 {
printf("%s\n", crawled_urls[i]);
 free(crawled_urls[i]);
}
pthread_mutex_destroy(&mutex);
   return 0;
}
```

5.RESULTS



```
Crawling: http://example.com
Crawling: http://example.com
Crawling: http://example.org
Crawling: http://example.org
Crawling: http://example.org
Crawling: http://example.org
Crawling: http://example.com
Crawling: http://example.com
Crawled URLs:
http://example.com
http://example.org
http://example.org
http://example.com


...Program finished with exit code 0
Press ENTER to exit console.
```

The result of implementing mutual exclusion principles in a multithreaded web crawler is a system that effectively manages concurrent access to shared resources, ensuring data integrity, and reliable performance. Here are the key outcomes and benefits of applying mutual exclusion in a multithreaded web crawler

Data Integrity: Mutual exclusion principles prevent race conditions, ensuring that shared resources, such as URL queues and visited page databases, are accessed and modified by only one thread at a time. This safeguards against data corruption and maintains data integrity throughout the crawling process.

Concurrency Control: The use of synchronization mechanisms like locks, semaphores, and mutexes allows for orderly and safe access to shared data structures. This control is essential in preventing conflicts and contention among threads, leading to a more predictable and efficient crawler operation.

Improved Performance: While synchronization mechanisms introduce some overhead, a well-designed multithreaded web crawler with mutual exclusion can achieve superior performance compared to a single-threaded crawler. Multiple threads can work concurrently, enhancing throughput and reducing the time required to crawl a large number of web pages.

Scalability: The application of mutual exclusion principles enables the web crawler to scale with the addition of more threads or processing nodes. This scalability is crucial for handling larger workloads and accommodating the ever-expanding web.

Load Balancing: By effectively managing the allocation of tasks among threads, mutual exclusion can lead to balanced work distribution, ensuring that no thread is overwhelmed while others remain idle. This results in a more efficient and fair utilization of resources.

Resilience: A well-implemented web crawler with mutual exclusion can recover gracefully from errors and exceptions, such as network failures or thread crashes. It can continue its crawling tasks without significant data loss or interruption.

Consistency: With mutual exclusion, the web crawler maintains consistent state and ensures that all threads have a shared view of the data. This consistency is crucial when dealing with interrelated data structures in a distributed environment.

Reduced Contention: Through effective mutual exclusion strategies and data structure design, contention and bottlenecks are minimized, contributing to a smoother and more responsive web crawler.

Reliability: A multithreaded web crawler that enforces mutual exclusion is more reliable in handling a wide range of web content and sites. It can adapt to different web environments and data sources while maintaining the integrity of the crawled data.

Knowledge Sharing: Implementing mutual exclusion in a web crawler promotes knowledge sharing and collaboration among development teams. It encourages best practices and continuous improvement in crawler design and operation.

6.CONCLUSION

We can observe that for lower rate values, small increase in rate brings down tpickurl by large amounts. For larger rate values, large increase in rate brings small change in tpickurl. Also, it presents new approach to utilize mutex waiting time for parsing operation. This leads to increased performance of the crawler, parser and efficient utilization of resources.This presents a new approach for implementing multithreaded crawlers using mutual exclusion locks, which results in performance improvement as compared to traditional crawlers. The approach of utilizing mutex waiting time proves efficient if employed for parsing or other useful operations within crawler threads.

7.FUTURE SCOPE

The future work will focus on minimizing the time incurred in acquiring the lock, writing data to database and releasing the lock. This time is represented as grey section in graphs shown in this document. This may be accomplished by interacting with operating systems at a lower level to speed up the locking and releasing the mutex lock. Also, we will cover the aspects that will enhance the performance by providing an efficient synchronization model across crawler and parser threads.

8.REFERENCES

1. Lawrence Page, Sergey Brin. The Anatomy of a search Engine.
2. Submitted to the Seventh International World Wide Web Conference
3. (WWW98). Brisbane, Australia
4. Budi Yuwono, Savio L.Lam, Jerry H. Ying, Dik L. Lee. A World Wide Web Resource Discovery System. The Fourth International WWW Conference Boston, USA, December 11-14, 1995.
5. Gautam Pant, Padmini Srinivasan, Filippo Menczer. Crawling the Web.
6. Allan Heydon, Marc Najork. Mercator: A Scalable, Extensible Web Crawler
7. Muhammad Shoaib, Shazia Arshad. Design and Implementation of web information gathering system
8. Joo Yong Lee, Sang Ho Lee. Scrawler: A Seed by Seed Parallel Web Crawler.
9. Boldi P., Codenotti B., Santini M., and Vigna S. UbiCrawler: a scalable fully distributed web crawler. Software Pract. Exper., 34(8):711–726, 2004
10. S.chakraborti, M.van den Crawling: A new approach to topic-specific web resource discovery”. In the 8th International World Wide Web Conference, 1999