**TECHNIQUES-TO-IMPLEMENT-WEB-CRAWLERS-USING-MULTI-THREADING**

This project aims to implement multithreaded web crawlers with and without locks. There is a shared resource for multiple web crawlers, which is the URL frontier queue. The URL frontier queue is of fixed length, which stores the URLs that need to be crawled. Each crawler is going to extract a task (URL to crawl) from the frontier, crawl this web page to find further links available on this page, and push the fetched links into the frontier for further crawling. To avoid collision inside the URL frontier, locks must be employed for any push or pop from the frontier queue. This paper distinguishes the optimistic locking system and pessimistic locking system and implements two locking mechanisms: semaphore and monitor. It also analyzes the Coarse-grained locking technique and Fine-grained locking technique for the web crawlers. The performance of the different locking techniques and lock-free approach can be evaluated by the number of threads crawled within a given time slice. Keywords— multithreaded web crawlers, semaphore, monitor, performance

**Background**

It is difficult to profile a webcrawler on external servers as they always have some sort of *api rate limiting* in place to avoid (DoS) denial of service attacks. We hosted a local server using fastapi to *remove the server api-rate-limiting random bias* from different multithreading-based webcrawler runs. This server is based on two of the fastest-growing Python libs

1. [fastapi](https://fastapi.tiangolo.com/) and
2. [uvicorn](https://www.uvicorn.org/).

The local server also has limits During testing, we found, we needed to restart the server on subsequent webcrawler runs. We automated with that with a simple script called driver.py driver.py script run shell commands via python subprocess to restart the local server in consecutive webcrawler runs to maximize throughput.

**How to create a setup for the project**

Note: We recommend a Linux/UNIX environment to execute the project. Although, by installing virtual LINUX environment, will help in generating records. However, graph plot is not supported due to lack of UI

1. Update links to be crawled in file ...TECHNIQUES-TO-IMPLEMENT-WEB-CRAWLERS-USING-MULTI-THREADING\TestFiles\TestURL.txt
2. Make sure you have python3 and pip3 installed.

**Windows**

1. Set up virtual Linux environment - To execute Shell Script File Using Windows Subsystem For Linux, download Ubuntu From The Microsoft Store, Integrate With WSL and initialize the newly installed Linux distro
2. follow steps for Linux in Ubuntu

**Linux/UNIX**

1. In case you are unable to navigate to the code directory, clone the git code: git clone https://github.com/divaamahajan/TECHNIQUES-TO-IMPLEMENT-WEB-CRAWLERS-USING-MULTI-THREADING.git
2. Navigate into TECHNIQUES-TO-IMPLEMENT-WEB-CRAWLERS-USING-MULTI-THREADING/setup\_startup cd TECHNIQUES-TO-IMPLEMENT-WEB-CRAWLERS-USING-MULTI-THREADING/setup\_startup
3. Run quick-setup.sh script bash +x quick-setup.sh.
4. Make sure you have virtual env installed. sudo apt install python3.8-venv
5. Activate the python virtual env with command printed between # lines. for example

########################################################################

Activate dev profile by running following command

source /home/username/.serv-coder/bin/activate

########################################################################

for above output, execute source /home/username/.serv-coder/bin/activate

**How to run the driver program**

1. Navigate to SourceCode directory in your terminal ~/code/TECHNIQUES-TO-IMPLEMENT-WEB-CRAWLERS-USING-MULTI-THREADING/SourceCode
2. Execute python3 driver.py
3. Please make sure to change the output log directory, if you change the set of seed URLs, for accurate results
4. 