# Parallel Programming Project: Web Scraper

Angelo Miguel Gregorio
College of Computer Studies (CCS)
De La Salle University
Manila, Philippines
angelo gregorio@dlsu.edu.ph

Daniel Baguio
College of Computer Studies (CCS)
De La Salle University
Manila, Philippines
daniel\_baguio@dlsu.edu.ph

#### I. Introduction

This web scraper program gets the emails and names of staff from the *staff-directory* part of the De La Salle University (DLSU) website. Using parallel programming techniques, multiple threads can be instantiated, allowing for quicker retrieval of data from the website.

## II. PROGRAM IMPLEMENTATION

Python was used as this program's main programming language. The program asks for 2 inputs: the maximum scraping time the program can run for (in minutes), or *nTime*, and the total number of threads to be used by the program, or *nThread*. To fulfill the parallel programming requirement of this project, Python's *multiprocessing* library, specifically its *Semaphore* feature, is used, with nThread being used to initialize how many Semaphores will be used by the program through the *multiprocessing*. *Semaphore()* function. In order to access the website, both *Selenium* and *urllib.request* was used.

As the website used AJAX scripts in loading the staff data, Selenium was used to simulate loading said scripts before retrieving staff data from the website. Due to the volatility of the website—it did not load completely all the time due to the Cloudflare servers being inconsistent—only the first 24 staff members from the website are used for this project. Furthermore, a static wait time of 30 seconds is implemented in order to make sure that all AJAX data have successfully loaded before being retrieved by the program.



For Hiring

Accountancy

For each staff entry, only the *value* variable from the button class (the class of the image shown above) is taken as this value is used to access the staff member's page on the website. These value variables are part of the URL for each staff member's page, and these are compiled into a list which will be used to individually access each staff member's web page later on.



Once the variables have been retrieved and compiled, the program takes note of the current time right after the values have been compiled. A thread that individually accesses each staff member's web page is then created for each value retrieved from the website. All threads are then started and joined at the same time, but only a set number of threads denoted by nThread can access their respective web pages at any given time due to the total number of Semaphores only being limited. Aside from this limitation, the threads are only allowed to run within the time denoted by nTime; if the threads are still running when the current runtime exceeds nTime, they are forcefully stopped.

As mentioned earlier, each thread is only able to access one web page at a given time. A list called *email* and *name* are created beforehand which will act as a storage for each retrieved email and name entry from the web pages to be accessed. To retrieve the web page, urllib.request() is used, and to parse the contents of the page, *BeautifulSoup* is used. Only the email and name of the staff member is retrieved from the web page; once these data have been retrieved and appended to the *emails* and *names* lists, the thread will finish executing.

Once all emails and names have been retrieved from their respective web pages, the data are then exported to a CSV file using Python's built-in *csv* library. A TXT file is also exported by the program which contains the main URL accessed by the program, the total number of pages scraped, and the total number of emails retrieved.

## III. RESULTS

The program is able to retrieve data from the website inconsistently, which is mostly due to the website being unresponsive from time to time. Since the website does not load completely each run, it is challenging to fully test the effectiveness of the program. It could be that the website server does not allow for multiple successive queries; however, on the times that the program does run correctly, it is able to export the CSV and TXT files successfully without much issue. It is also apparent that the program is able to finish faster when more threads are used, but since there are more queries being done to the website, the web pages sometimes do not load correctly.

### IV. CONCLUSION

The use of parallel programming techniques expedited the process of retrieving data from the DLSU website, although having too many threads active at a time is not recommended, at least for the website used, as Cloudflare has precautions set in place that prevents requests from passing through whenever suspected *botting* activities, or multiple repeated queries within a short amount of time, are detected. This is to ensure that the integrity of the website is retained and access to it unimpeded. As such, it might be best to only use a small number of threads in order to reduce server strain.

## REFERENCES

- [1] Selenium API
- [2] BeautifulSoup API
- [3] Python API