PARALLELIZING WEB SCRAPING TO IMPROVE PERFORMANCE AND SCALABILITY

BY

NA YI CHUN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2023

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: **PARALLELIZING WEB SCRAPING TO IMPROVE PERFORMANCE AND SCALABILITY**

Academic Session: JUNE 2023

I NA YI CHUN

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

[illegible]	Verified by,
	Alex
(Author's signature)	(Supervisor's signature)
Address:	
B-12-7 SANDILANDS	
GAT LEBUH SANDILANDS	____Ooi Boon Yaik_____
GEORGETOWN, 10300	
PENANG	Supervisor's name
Date: 15 SEPTEMBER 2023	**Date**: 15 SEPTEMBER 2023

Universiti Tunku Abdul Rahman			
Form Title : **Sample of Submission Sheet for FYP/Dissertation/Thesis**			
Form Number: **FM-IAD-004**	Rev No.: **0**	Effective Date: **21 JUNE 2011**	Page No.: **1 of 1**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 15 SEPTEMBER 2023

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that **__NA YI CHUN____**(ID No: **_20ACB05770_**) has completed this final year project/ dissertation/ thesis* entitled "**PARALLELIZING WEB SCRAPING TO IMPROVE PERFORMANCE AND SCALABILITY**" under the supervision of **TS DR OOI BOON YAIK** (Supervisor) from the Department of **COMPUTER SCIENCE**, **FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY.**

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

[illegible]

____

NA YI CHUN

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled "**PARALLELIZING WEB SCRAPING TO IMPROVE PERFORMANCE AND SCALABILITY**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : [illegible]

Name : NA YI CHUN

Date : 15 SEPTEMBER 2023

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my supervisor, Ts Dr Ooi Boon Yaik and my academic advisor, Dr Wong Chee Siang for their motivation and guidance through my research journey for this project. I would like to express my gratitude especially to my supervisor for their insight, expertise, and encouragement, which has helped me greatly in the completion of this project.

Secondly, I would like to thank my family and friends for their unconditional love and support. Their constant encouragement, patience, and understanding have been a source of motivation for me. They have been there for me through the ups and downs of this research journey, and I could not have done it without them.

ABSTRACT

Web scraping is the process of extracting data from websites, usually for analysis or other purposes. Web scraping is increasingly important in e-commerce because it enables businesses to extract valuable data from competitors' websites, especially product prices. This data can be used to gain insights into market trends, optimize pricing strategies, and improve product offerings. However, web scraping can be a challenging task due to some reasons such as the sheer volume of data available, network bandwidth, etc., when monitoring product prices across numerous e-commerce platforms. The main purpose of this project is to help whoever is suffering from the long waiting time to scrape desired information much faster than usual regardless of whether the data is small-scale or large-scale from the internet through the integration of web scraping technologies and distributed computer system. The proposed solution requires user interaction to be configured and initialized. It employs a message queuing algorithm to divide scraping tasks into smaller units and utilizes multiple worker nodes for concurrent web data extraction. In the scraping process, Selenium Web Driver interacts with specified web elements based on user-defined selectors or XPATH, allowing asynchronous HTTP requests and responses. Performance metrics such as response times, bandwidth data, and task status will be monitored for benchmarking and error handling. After finishing the scraping process, the scraped results are stored in a CSV file for further analysis.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CPN	Colored Petri Net
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CSV	Comma Separated Values
GIL	Global Interpreter Lock
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
URL	Uniform Resource Locator
XML	Extensible Markup Language

Chapter 1 - Introduction

1.1 Problem Statement and Motivation

The majority of online content is displayed in the form of HTML documents, which are designed for human consumption rather than machine learning and are considered unstructured data. This unstructured data is growing rapidly on the internet each year. Due to its lack of adherence to any specific data format, extracting and storing large volumes of internet data for future use poses a significant challenge [1]. Our primary focus is to extract and store substantial amounts of internet data efficiently for future analysis, a task that is achievable through utilizing a specialized technique known as Web Scraping. Web scraping is the process of extracting data from websites, usually for analysis or other purposes. Web scraping is increasingly important in e-commerce because it enables businesses to extract valuable data from competitors' websites, especially product prices. This data can be used to gain insights into market trends, optimize pricing strategies, and improve product offerings. However, web scraping can be a challenging task due to some reasons such as the sheer volume of data available, network bandwidth, etc., when monitoring product prices across numerous e-commerce platforms. On top of that, web scraping can be resource-intensive such as demand for high computational processing power. As a result, web scraping can be a complex and challenging task that requires careful planning and execution.

Problem Statement 1: Default web scraping configuration using a single thread.

Using the default web scraping configuration, which employs a single thread, sets a significant constraint on the speed and effectiveness of web data extraction. This method retrieves data in a sequential manner, leading to sluggish performance and impracticality when dealing with extensive web scraping tasks. For instance, when attempting to acquire information from a thousand web pages, employing a single-threaded approach means that data retrieval occurs one page at a time, resulting in a time-consuming operation.

Problem Statement 2: Dynamic nature of many websites.

Dynamic content refers to website content that changes in response to user input, server-side scripts, or other factors such as location or time of day. Handling changes in web documents over time is one of the toughest issues in web data extraction as the challenge with dynamic content is that it can make it difficult for web scraping scripts to capture data accurately. The development of JavaScript frameworks in web development has caused significant changes to how data is embedded and how web pages are rendered, thus affecting the structure of web documents [2]. For example, if a script is designed to scrape a specific element from a webpage, but that element is only loaded after the page has finished loading, the script may not be able to find it. Hence, the proposed solution aims to handle dynamic content more effectively.

1.2 Objectives

This application aims to further enhance the processing performance and scalability of web scraping and data extraction processes by applying suitable scraping techniques and tools to coordinate the process effectively.

Hence, the objectives of this project including:

1. To build a scalable web scraping solution that can handle dynamic content and be customized to suit specific scraping requirements.
2. To design and develop distributed systems to scrape web data.

1.3 Project Scope and Direction

The scope of this project is to design and develop a custom library tailored for web scraping tasks that incorporate various optimization techniques to streamline the data extraction process. This library would be developed using Python programming language while Visual Studio Code would serve as the development environment for testing and debugging the library. In terms of web scraping, this library implemented Selenium, a powerful web automation tool, to enable browser automation. This allows the library to mimic human behaviour to interact with web pages so that data can be extracted from websites effectively without being detected by the antibot system if there is any. Furthermore, the backend infrastructure of the library is developed to facilitate task management and communication using Flask, a lightweight and versatile Python web framework. On top of that, algorithms such as message queuing is implemented to handle task management efficiently as it enables asynchronous and parallel processing of web scraping requests. It means that multiple scraping tasks can be queued and executed concurrently without blocking or delaying other processes, making the library suitable for large-scale data extraction tasks while ensuring it is much faster. In the end, there would be test cases and performance benchmarks to validate the library's effectiveness and speed improvements.

1.4 Contributions

This project makes several significant contributions. First, it introduces a custom Python library that enhances the efficiency and speed of web scraping operations by leveraging the capabilities of Selenium to be able to perform web scraping on a wide range of websites. Secondly, this project also provides a more flexible and scalable architecture to monitor scraping tasks. Moreover, this project is able to handle a high volume of scraping requests without delays by implementing message queuing that allows tasks to be executed asynchronously in parallel. Ultimately, these contributions address the real-world challenges of web scraping by providing a comprehensive solution that enhances performance and scalability in this domain.

1.5 Report Organization

There are seven chapters in this report. Chapter 1 introduces the project and outlines the key concepts that guided the project's development. In Chapter 2, relevant technologies and existing research relevant to the solution and algorithm employed in the development of this project will be reviewed. In Chapter 3, we will go through the theories underlying the methods employed in this project and give a general description of the system architecture. In Chapter 4, attention is given to the system design and functionality. In Chapter 5, we will discuss the system implementation such as hardware setup, software setup, how the system operates, implementation issues and challenges as well as concluding remarks. In Chapter 6, the system will be evaluated, and the performance of the system will be discussed. Lastly, there will be a conclusion and recommendation for future work in Chapter 7.

Chapter 2 - Literature Review

This chapter will discuss some popular existing web scraping tools and techniques, which are related to the ones utilized in this proposed web scraping solution. Other papers related to improving the performance of web scraping will also be reviewed.

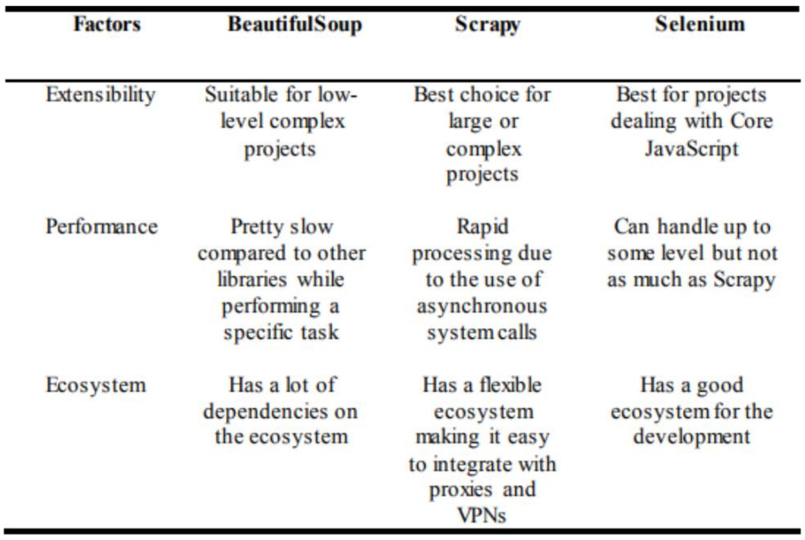
2.1 Review of the Technologies

2.1.1 Web Scraping using Python

Python is one of the most popular programming languages nowadays. Although some users may not be familiar with it, but it is quite simple to learn even if users have never used it or have no prior programming expertise. Since Python is a widely used programming language, if there is any problem regarding the Python code, it is always recommended to find the solution through the large online community where there are high chances that someone else has faced a similar issue and solved it. In addition, Python code is relatively readable and simple to understand, hence, the library of code is highly reusable and easy to maintain as well compared to Java and C language [3].

2.1.2 Comparison between BeautifulSoup, Selenium, and Scrapy

The following are some examples of web scraping tools that use Python. First, Scrapy is a framework that helps developers create and scale large web crawling projects. It also includes a web-based shell to simulate a human user's browsing behaviour. A graphical interface is provided in the web-based crawler to simplify the use of the web scraping tool, making it easier for non-programmers to collect web content. Beautiful Soup is a tool for scraping HTML and other XML documents, which is how data gets extracted. It offers simple Pythonic functions for navigating, finding, and changing a parse tree as well as a toolkit for disintegrating HTML files and extracting needed data using lxml or html5lib. Selenium refers to a web page rendering tool that allows the process of surfing a website to be automated [4]. [5] uses Python web scraping tool to extract information from more than 3000 documents to automate the process of collecting and storing the data. When the authors manually search for data in each document manually, they manage to complete 25 tasks every hour. In contrast, after they use a web scraping tool to automate the process, more than 1000 tasks can be completed per hour effortlessly.



Factors	BeautifulSoup	Scrapy	Selenium
Extensibility	Suitable for low-level complex projects	Best choice for large or complex projects	Best for projects dealing with Core JavaScript
Performance	Pretty slow compared to other libraries while performing a specific task	Rapid processing due to the use of asynchronous system calls	Can handle up to some level but not as much as Scrapy
Ecosystem	Has a lot of dependencies on the ecosystem	Has a flexible ecosystem making it easy to integrate with proxies and VPNs	Has a good ecosystem for the development

Table 2.1.2.1 – Comparison between Python Web Scraping Libraries and Frameworks

2.1.3 Testing Using Selenium Web Driver

[6] offers a comprehensive overview of Selenium Web Driver after using it for software testing. The author emphasizes that the execution speed of Selenium Web Driver is considered the fastest among all components of the Selenium toolkit as starting a server is unnecessary because it can directly interact with the browser for test case execution. On top of that, it also offers better support for testing dynamic web pages. One example of Selenium Web Driver that can behave like a real user is that it will not fill a disabled text box. The author also suggests that Selenium Web Driver supports integration with other tools to enhance usability and accuracy.

2.1.4 Flask and Message Queue

A message queuing system acts as a buffer for asynchronous data communication between different components or systems. It stores messages and allows producers and consumers to connect, publish, or consume messages from queues or topics. This decouples the sender and receiver, promoting independent operation and improving system flexibility and efficiency. Message queuing system is able to handle increased in demand. For example, the system can easily accommodate higher processing requirements by adjusting factors such as message volume, partitions, and producers or consumers. Hence, it is scalable [7]. [8] suggests an architecture that users can access via a web application to monitor conditions with machine learning. To build the web interface that connects CoreSys-Cloud's MQTT broker and a

MySQL database, Flask is used. Flask is chosen for its lightweight and flexible nature, making it ideal for creating user-friendly web interfaces. This choice indicates the author's intention to provide an accessible and efficient method of accessing and managing data from the cloud-based system. Besides that, message queuing (MQTT) is utilized to enable communication between the OPC-UA/MQTT gateway and CoreSys-Cloud. MQTT is a lightweight protocol that excels in transmitting data between devices and servers, making it an ideal option for IoT and automation systems.

2.1.5 Summary of the Technologies Review

In conclusion, Python is the most suitable choice among other programming language. Even though Scrapy has more advantages compared to Selenium, but Scrapy lacks built-in support for scraping dynamic websites that heavily rely on client-side rendering through JavaScript, especially e-commerce websites while Selenium can automate a web browser to load the web page, render and interact with JavaScript to extract desired data. Hence, Selenium is preferred rather than Scrapy. In order to improve performance and scalability of web scraping processes, Selenium Web Driver, Flask, and Message Queue have to be integrated into the system as well.

2.2 Review of the Existing Systems/Applications

2.2.1 Design and Implementation of Scalable, Fully Distributed Web Crawler for a Web Search Engine

In this paper [5], DCrawler, a scalable and completely distributed web crawler is proposed and put into practice. The primary features include platform independence, workload decentralization, as it can assign the tasks efficiently for dividing the domain to crawl, and fault tolerance. The proposed crawler is made up of numerous agents that coordinate independently to ensure each agent scans a portion of the web. Each agent has multiple threads assigned to it, and each thread is dedicated to visiting a single host. The threads visit each host using a breadth-first search approach. Additionally, multiple techniques are implemented to ensure that various threads are visiting different hosts concurrently to avoid overloading any particular host with an excessive number of requests. The web crawler is capable of downloading multiple million pages per day, making it efficient for purposes like web search and web characterization.

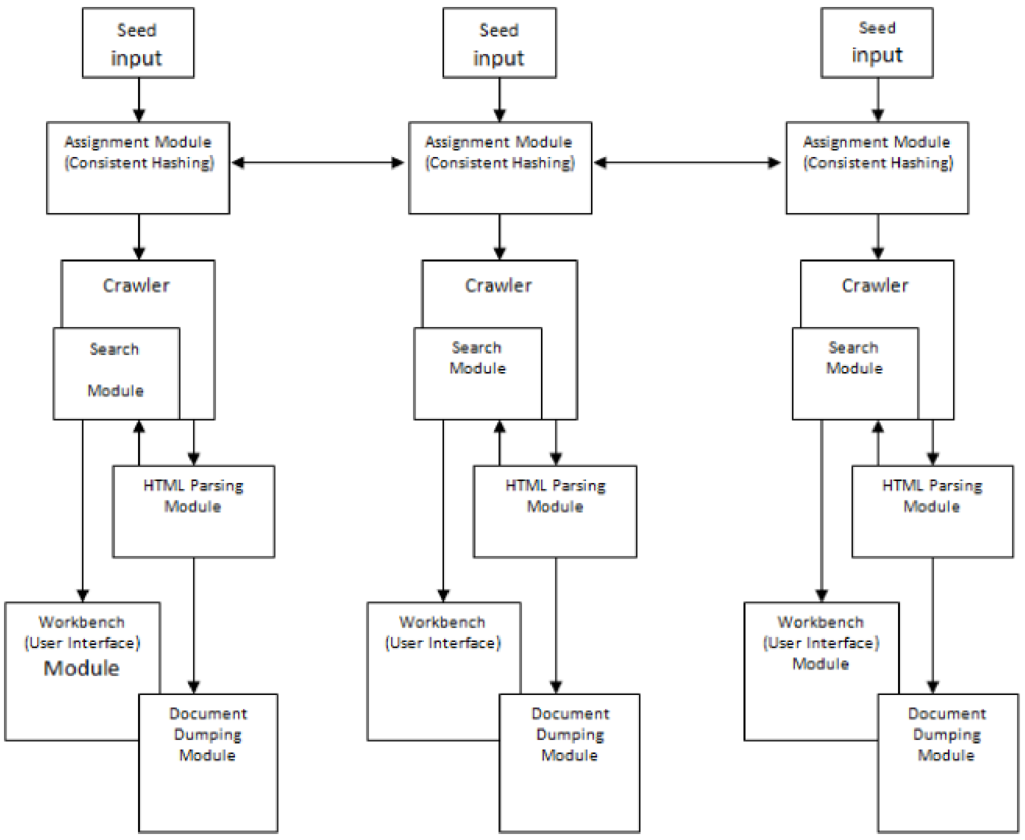


Figure 2.2.1.1 – Cooperation between multiple agents in DCrawler

Based on the figure above, the crawling architecture shown in the figure aims to divide the crawling jobs into separate segments to be executed by particular modules. The flowchart can be comprehended as follows:

The paragraph discusses the collaboration between three distinct crawlers and their corresponding logic, as illustrated in Figure 2.2.1.1. The seed represents the initial list of URLs provided to the crawler, which then visits them. The assignment module computes which crawler is responsible for a specific host using an identifier-based consistent hashing technique. The core crawling module follows a basic crawling algorithm and fetches the web page associated with the URL, which is then passed to the HTML parsing module. The HTML parsing module extracts various components of the web page and returns them to the crawler, and the extracted links are passed back to the assignment module for further processing. The HTML parsing module also enables focused crawling based on constraints. The workbench is a user interface module that provides real-time control over the crawling process and displays graphical representations and statistics. Finally, the document dumping module is used to store the fetched pages in a local machine or network server, and the available capacity is also considered as a factor that affects the weight of the crawler. In summary, the paragraph describes the process of crawling, parsing, and storing web pages, as well as the user interface that enables real-time control over the crawling process.

2.2.2 Synchronizing Distributed Scraping

[9] presents a synchronized and distributed web scraping system for comparing multiple flight ticket-selling outlets. The system compares at least two outlets from a common search query to improve the efficiency of web scraping for extracting ticket prices. Workload can be distributed across multiple machines, and scraping bots can be synchronized for data collection. Each machine in the proposed configuration uses one scraping thread and several machines are used to enable simultaneous comparison of multiple outlets due to limited resources.

The benefits of the proposed solution include that the system ought to remain active constantly. For instance, the implementation of the price comparison system, which included 12 different bots, was tested for over two months. The system experienced occasional crashes of individual bots, but overall, it remained stable and unaffected for comparisons with other bots. The distributed design of the solution offers the flexibility for many machines to use many bots on to conduct comparisons across a variety of outlets and extract desired data from each outlet concurrently.

The proposed solution for synchronized scraping aims to avoid deadlocks through the use of synchronization barriers, including synchronization at the start, synchronization during search, and timeouts. The Colored Petri Net (CPN) modeling technique is used to explain the solution. CPN is an extension of Petri Nets and uses a high-level programming language to create timed models. The CPN model consists of places and transitions that are connected through synchronization barriers to ensure efficient and synchronized data extraction.

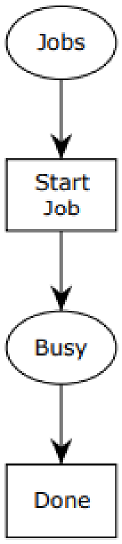


Figure 2.2.2.1 – An example of Petri Net

Figure 2.2.2.1 shows a "Jobs" place that is initialized with three tokens representing jobs. Only the "Start Job" transition is enabled at first, but when it is fired, the "Jobs" place is left with two tokens and the "Busy" place has one token. Eventually, the CPN model described in the paper reaches its final state when the "Done" place gathers all three tokens and none in the "Jobs" place.

2.2.3 Research on Scrapy-Based Distributed Crawler System for Crawling Semi-structure Information at High Speed

[10] introduces a distributed crawler system to crawl video websites based on Scrapy for high-speed crawling of semi-structured information. The proposed system is also designed to overcome the low utilization rate and efficiency of single machine crawlers in extracting complex semi-structured information on website pages with video contents. This paper proposes an improved approach to distributed web crawling by integrating Redis Database and Scrapy-Redis into the Scrapy framework. The result is a faster and more efficient system capable of crawling semi-structured data. Additionally, the integration allows for a standardized storage strategy. According to the authors, the traditional single machine crawler is no longer suitable for large-scale data acquisition requirements, but the use of a distributed crawler can fulfill the growing need for web page data. The use of multiple machines to cooperate in information crawling can lead to improved efficiency and robustness of the crawler. Video websites such as Tencent, Youku, SOHU, and iQIYI are used for testing purpose.

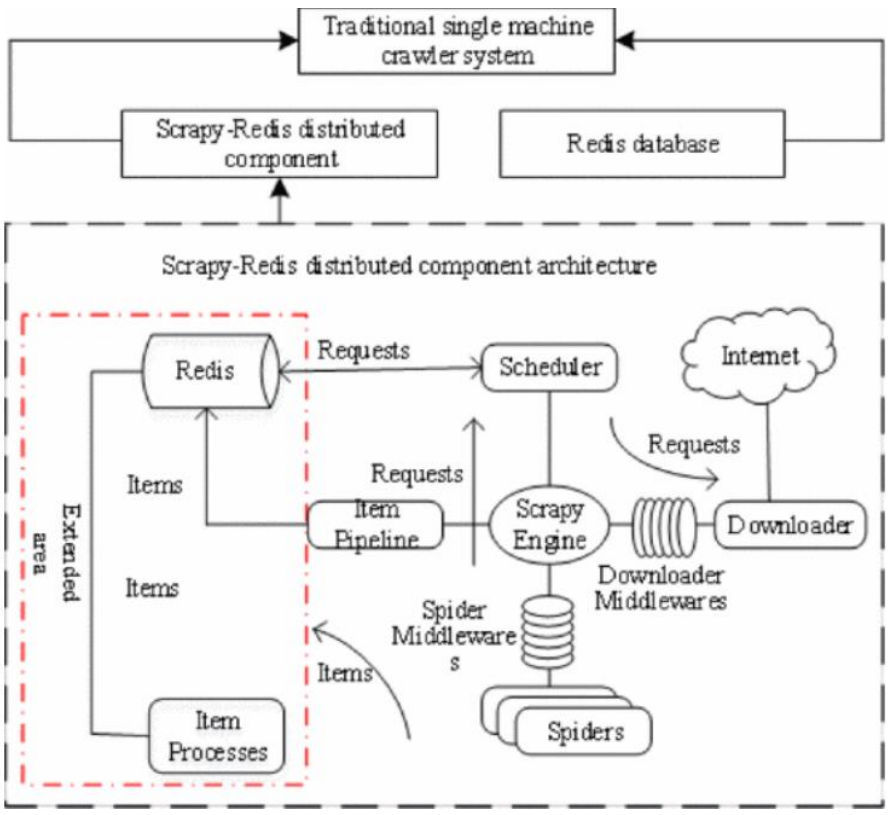
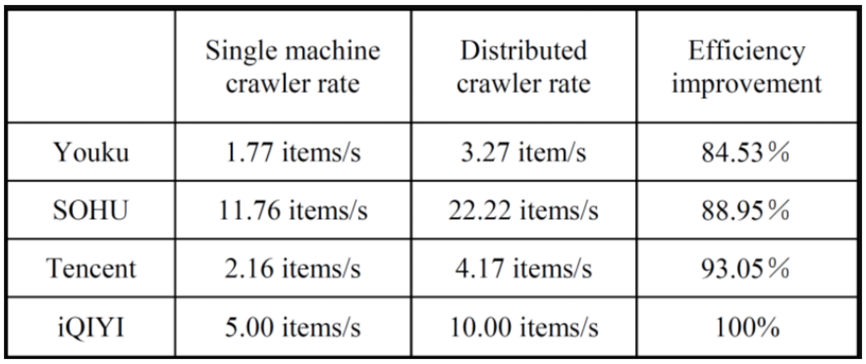


Figure 2.2.3.1 – Enhanced scheme of distributed extension for standard single crawler

The goal of extending the single crawler system to a distributed system is to have specialized modules execute distinct segments of the crawling task. This is done through the use of Scrapy-Redis distributed components and Redis databases, which can mark crawled links in the crawling queue to prevent repeated crawling issues. By adding Redis databases to the traditional Scrapy framework, the workload among multiple crawlers can be coordinated in a distributed system. Through it, each crawler can access the same Redis database, enabling the scheduling and duplicate management of the Redis database as the Redis database stores information about which URLs have already been crawled, which URLs are waiting to be crawled, and which URLs are being crawled by which machine. This allows multiple machines to work together efficiently to crawl vast number of URLs in parallel, without duplicating effort or stepping on each other's toes. Eventually, this improves the single crawler system, resulting in a distributed high-speed information crawling system.



	Single machine crawler rate	Distributed crawler rate	Efficiency improvement
Youku	1.77 items/s	3.27 item/s	84.53%
SOHU	11.76 items/s	22.22 items/s	88.95%
Tencent	2.16 items/s	4.17 items/s	93.05%
iQIYI	5.00 items/s	10.00 items/s	100%

Table 2.2.3.1 – Comparison between single and distributed crawler

Based on Table 2.2.3.1, the distributed crawler performs better compared to a single machine crawler as it shows efficiency gains of 84.53%, 88.95%, 93.05%, and 100% for four video sites.

2.2.4 Summary of the Existing Systems

In [5], although the web crawler implements novel parallel crawling to improve the performance, the proposed multiprocessing crawlers are only limited to using one machine to execute.

On the other hand, the solution proposed in [9] has a distributed design that enables the usage of unlimited machines and bots to carry out the price comparison of multiple outlets simultaneously, extracting the required data from each outlet. However, due to the limited resources available to each machine in [9], each machine only runs one scraping thread. When using the Scrapy-Redis distributed component to improve the performance of a standard machine crawler in [10], multiple machines can be used in a distributed architecture.

The distributed nature of the system in [9] and [10] allows the workload to be divided among these different machines, and each machine can work independently to perform its assigned tasks. This can help to increase the speed and efficiency of the crawling process, as well as to improve the scalability and robustness of the system. However, the use of multiple hosts may also increase the cost of maintaining and managing the system.

Chapter 3 - System Methodology/Approach

3.1 Methodologies and General Work Procedures

The following summarizes the library's general methodology in the proposed solution. First, to initiate the Python script and configure its settings, the user needs to input specific parameters including the website link to be scraped, web page elements, the desired number of items to scrape, and the parallel processing setup such as the number of threads. After that, the system will import libraries and set up configurations to perform initialization and prepare the necessary data structures. Next, the system will implement the message queuing algorithm by dividing the scraping tasks into smaller units and adding them to a task queue where each task represents an item to be scrapped. Meanwhile, the system will create a specific number of worker nodes, which are equivalent to the number of threads specified by the user when calling the function. These workers collaborate to fetch and scrape data from the web concurrently without blocking or delaying one another. Now, the system will browse the website link provided earlier and look for web page elements specified by the user to scrape. In this session, each worker navigates web pages through Selenium Web Driver to interact with specified web page elements based on CSS selectors or XPATH to extract data while message queuing allows the HTTP requests and responses to be captured asynchronously. Throughout the scraping process, the system monitors and logs the response time, bandwidth data, and status of each task for benchmarking and to ensure that exceptions can be handled smoothly without interrupting the process. Once all tasks are completed, the system will calculate the total response time for the scraping process and store the scraped results in a CSV file for further analysis which is not within this project scope. Visualizations such as bar charts will be generated as well using Matplotlib library to provide insights into the performance of different configurations of the system. Finally, the user can review the scraped data and performance analysis and rerun the scraping process by adjusting the configuration of the aforementioned parameters if necessary.

3.2 Use Case Diagram and Description

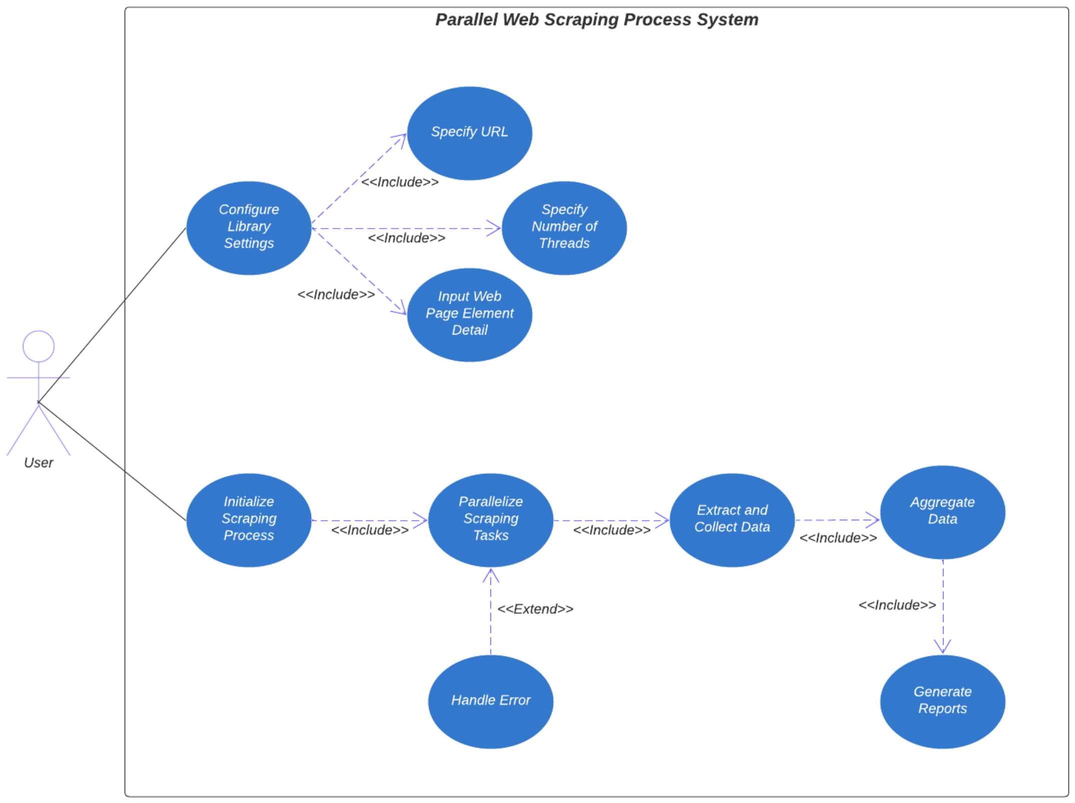
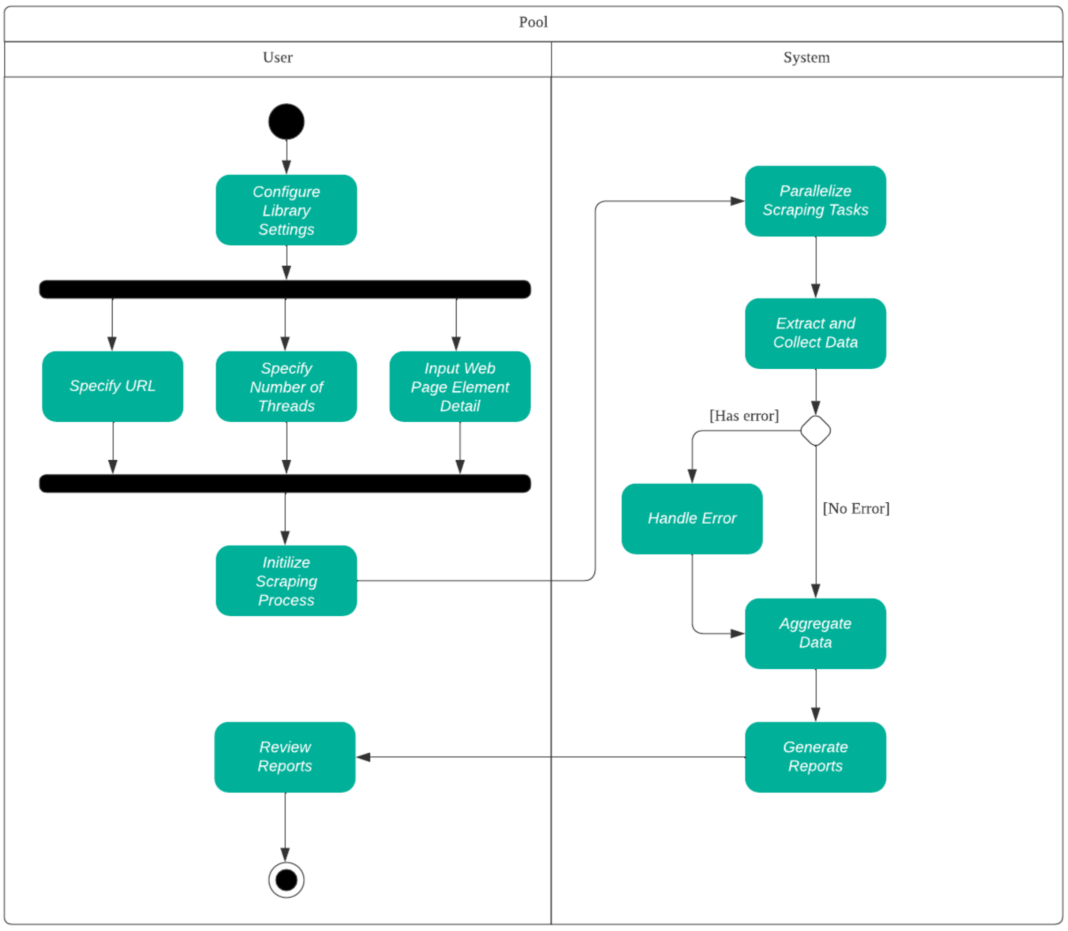

Parallel Web Scraping Process System
User
Configure Library Settings
Specify URL
Specify Number of Threads
Input Web Page Element Detail
<<Include>>
<<Include>>
<<Include>>
Initialize Scraping Process
Parallelize Scraping Tasks
Extract and Collect Data
Aggregate Data
Handle Error
Generate Reports
<<Include>>
<<Include>>
<<Include>>
<<Extend>>
<<Include>>

Figure 3.2.1 – Use Case Diagram

Action	Done by	Description
Configure library settings	User	Provide parameters needed by the library function.
Specify URL	User	Provide a website link to be scrapped.
Specify number of threads	User	Define the number of threads per process.
Input web page element detail	User	For the scraper to locate the specific element on the web page.
Initialize scraping process	User	Run the process.
Parallelize Scraping Tasks	System	Divide the scraping tasks into smaller units and assign them to multiple workers or threads. It manages the concurrent execution of tasks.
Extract and Collect Data	System	Collect data from the target web page using Selenium asynchronously. HTTP requests and responses are captured to extract information.
Handle Error	System	Manage exceptions that may occur during data acquisition. It logs errors and continues scraping.
Aggregate Data	System	Store collected data such as scraped information, response time, and network bandwidth for further analysis.
Generate Reports	System	Create reports and visualizations to provide insights to the user.

Table 3.2.1 – Description of Use Case Diagram

3.3 Activity Diagram



Pool
User
System
Configure Library Settings
Specify URL
Specify Number of Threads
Input Web Page Element Detail
Initilize Scraping Process
Review Reports
Parallelize Scraping Tasks
Extract and Collect Data
[Has error]
Handle Error
[No Error]
Aggregate Data
Generate Reports

Figure 3.3.1 – Activity Diagram

Chapter 4 - System Design

4.1 System Block Diagram

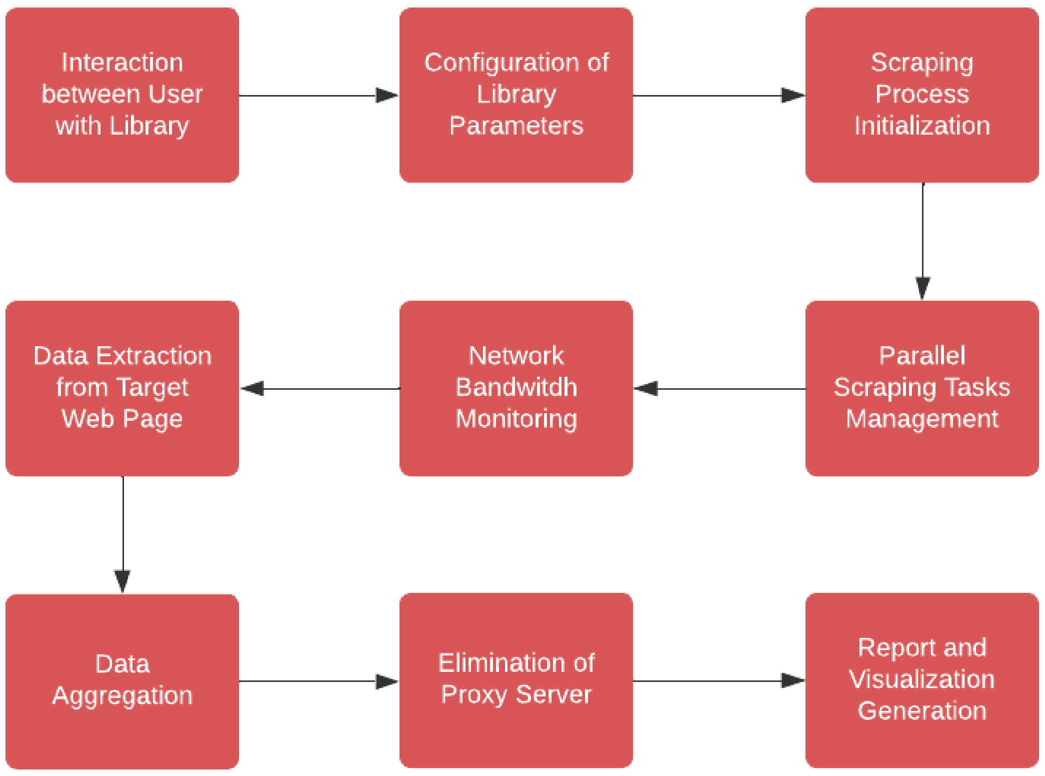

Interaction between User with Library
Configuration of Library Parameters
Scraping Process Initialization
Parallel Scraping Tasks Management
Network Bandwitdh Monitoring
Data Extraction from Target Web Page
Data Aggregation
Elimination of Proxy Server
Report and Visualization Generation

Figure 4.1.1 – System Block Diagram

The flowchart begins with "Interaction between User with Library" and "Configuration of Library Parameters" where the user needs to configure the web scraping parameters. For example, the user has to provide the website URL to scrape, the web page elements detail such as CSS Selector or XPATH per item they aim to scrape and define the number of threads per process to enable multithreading. Upon completion of the configuration, the user may initiate the web scraping process.

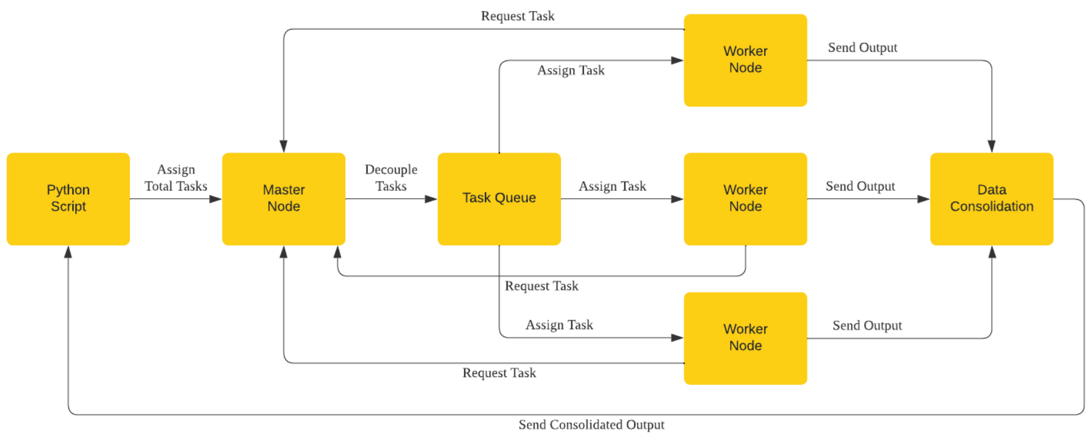


Figure 4.1.2 – Message Queue Algorithm

Once the system has been activated, it will elect a master node and generate multiple worker nodes according to the number of threads set by the user earlier. The master node will manage the parallelization of the scraping tasks by dividing them into smaller units and store in the task queue. Initially, every worker node will be assigned a task to scrape particular web page elements. Instead of assigning an equal number of tasks to every worker node in the first place, the message queue algorithm is designed to assign tasks to any worker node that is not executing any task at the moment. Another way round, the worker node will send a request to the master node for a new task when it has finished all assigned tasks. The advantage of designing the algorithm in this way instead is to prevent any delay caused by any worker node. For example, there are 4 worker nodes, and each worker node is assigned 10 tasks. If worker node A has finished the 10 tasks much earlier than worker nodes B, C, and D, there will be a waste of resources while waiting for other worker nodes that are slow due to latency or other issues. Hence, the overall performance and efficiency will be affected negatively as well.

In “Network Bandwidth Monitoring”, BrowserMob Proxy, an open-source tool that captures and records HTTP traffic between the Selenium Web Driver browser in this case, and a target web server that acts as a proxy server, is utilized to monitor network activity. This includes the bandwidth usage and data exchanged during web scraping. Once an instance of the proxy server is created, it is automatically configured to listen on a specified port (port 8090). Before initializing a scraping task, HTTP Archive (HAR) data is captured for that particular task. Once the web scraping process begins, the proxy server sits between the Selenium Web Driver browser and the target web server to intercept all HTTP requests and responses including headers, URLs, response times, and data size. After each web scraping task is completed, the captured HAR data will be saved and stored in a directory. This step is valuable as it helps to identify the potential bottlenecks and diagnose any issues related to network communication as well as for performance analysis and optimization purposes.

In "Data Extraction from Target Web Page", the scraping process captures HTTP requests and responses to extract information asynchronously using Selenium. Meanwhile, if there is any error detected during data extraction such as the target web page is failed to be accessed, it is caught using a try-except block and the corresponding exception message will be printed to indicate the occurrence of the error. After printing the error message, the code continues to the next task without terminating the entire scraping process in order to maximize data extraction and minimize disruptions caused by individual task errors. This allows the system to log errors and it will affect the smoothness of the scraping process.

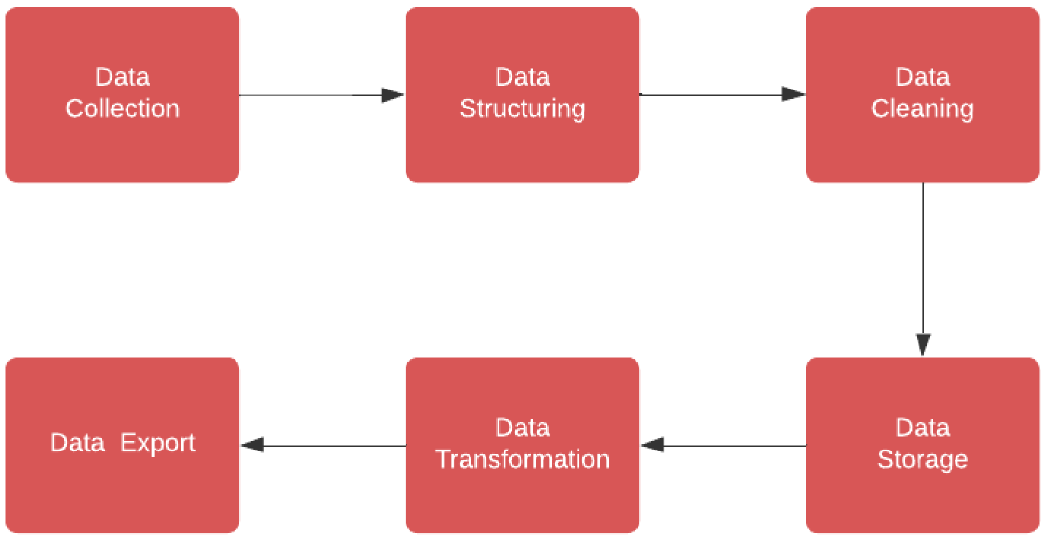


Figure 4.1.3 – Data Aggregation

The "Data Aggregation" module represents the process of collecting and organizing the data extracted from the scraping process into a structured format for user review and reporting. Before data aggregation can take place, the system performs web scraping to extract data, mostly unstructured, from the target web pages such as product names and product prices. After collecting the data, the data is organized into a structured format so that the data can be stored in data structures such as tables, lists, or dictionaries. Next, data needs to be cleaned to avoid formatting issues and data redundancy. Once the data is structured and cleaned, it will be stored in CSV file for easy data retrieval and visualization. Data transformation is needed to standardize the time by converting the time unit into seconds. Finally, the aggregated data is ready to be exported to external tools or formats for analysis and visualization.

After finishing running all the scraping tasks, the access to the proxy server will be eliminated and reports and visualization such as network activity, scraped results, total response times, etc., will be generated for user review.

Chapter 5 - System Implementation

5.1 Hardware Setup

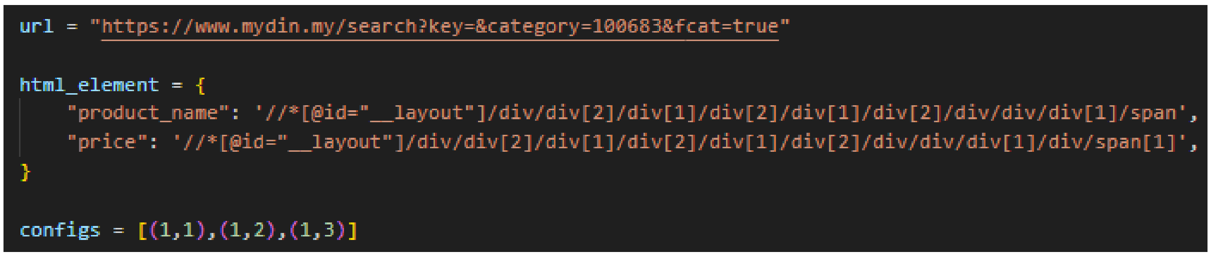
Description	Specifications
Model	Acer Nitro AN515-57
Processor	Intel Core i7-11800H
Operating System	Windows 10
Graphic	NVIDIA GeForce RTX 3060 Laptop GPU
Memory	16GB DDR4 RAM
Storage	512GB SATA HDD, 240GB SSD

Table 5.1.1 – Specifications of Laptop

5.2 Software Setup

1. Flask – Version 2.3.2
2. Microsoft Visual Studio Code – Version 1.77.3
3. Python – Version 3.11.3
4. Selenium – Version 4.8.3
5. Selenium Web Driver (Chrome Driver) – Version 116.0.5845.96

5.3 System Operation

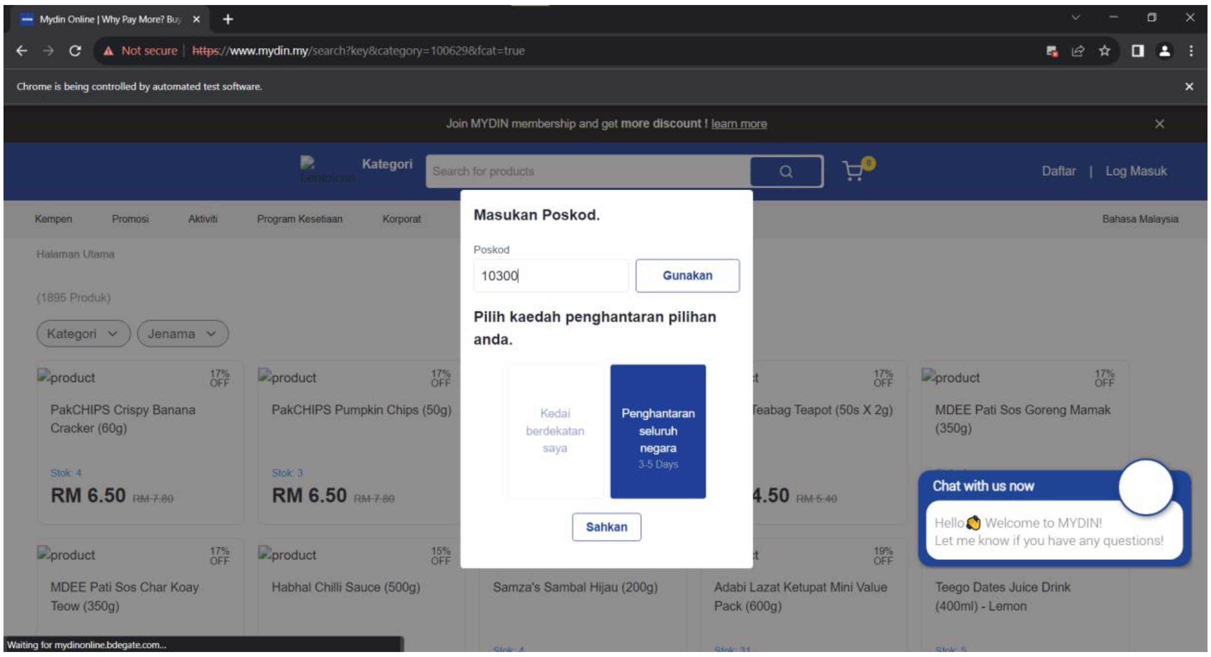


```
url = "https://www.mydin.my/search?key=&category=100683&fcat=true"

html_element = {
    "product_name": '//*[@id="__layout"]/div/div[2]/div[1]/div[2]/div[1]/div[2]/div/div/div[1]/span',
    "price": '//*[@id="__layout"]/div/div[2]/div[1]/div[2]/div[1]/div[2]/div/div/div[1]/div/span[1]',
}

configs = [(1,1),(1,2),(1,3)]
```

Figure 5.3.1 – Configuration

The solution is a library function that will prompt inputs from the user,

1. Target web page URL to be scrapped.
2. HTML structure of the desired web page element(s).
3. Select the number of threads.
    - (1,1) represents (number of processes, number of threads).
    - allows scaling the web scraping operations horizontally by adding more threads.



Chrome is being controlled by automated test software.

Join MYDIN membership and get more discount ! learn more

Masukan Poskod.

Poskod: 10300

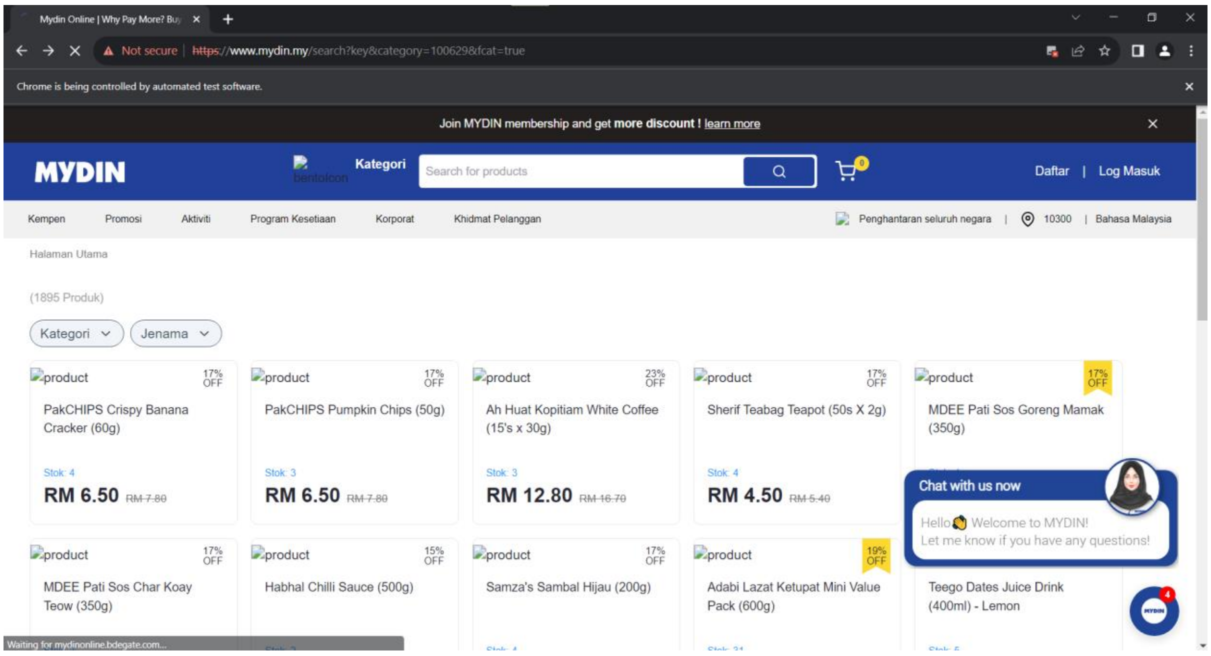
Pilih kaedah penghantaran pilihan anda.

Hello Welcome to MYDIN! Let me know if you have any questions!

Mydin Online | Why Pay More? Buy
Not secure
https://www.mydin.my/search?key&category=100629&fcat=true
Kategori
Search for products
Daftar | Log Masuk
Kempen
Promosi
Aktiviti
Program Kesetiaan
Korporat
Bahasa Malaysia
Halaman Utama
(1895 Produk)
Kategori
Jenama
Gunakan
Kedai berdekatan saya
Penghantaran seluruh negara
3-5 Days
Sahkan
PakCHIPS Crispy Banana Cracker (60g)
PakCHIPS Pumpkin Chips (50g)
Teabag Teapot (50s X 2g)
MDEE Pati Sos Goreng Mamak (350g)
RM 6.50
RM 6.50
Chat with us now
MDEE Pati Sos Char Koay Teow (350g)
Habhal Chilli Sauce (500g)
Samza's Sambal Hijau (200g)
Adabi Lazat Ketupat Mini Value Pack (600g)
Teego Dates Juice Drink (400ml) - Lemon
Waiting for mydinonline.bdegate.com...

Figure 5.3.2 – Pop Up Window

The pop-up window will be dismissed using Selenium to interact with it like a real user.


Mydin Online | Why Pay More? Bu

Not secure | https://www.mydin.my/search?key&category=100629&fcat=true

Chrome is being controlled by automated test software.

Join MYDIN membership and get **more discount** ! learn more

MYDIN | Kategori | Search for products | Daftar | Log Masuk

Kempen | Promosi | Aktiviti | Program Kesetiaan | Korporat | Khidmat Pelanggan | Penghantaran seluruh negara | 10300 | Bahasa Malaysia

Halaman Utama

(1895 Produk)

Kategori | Jenama

Product	Discount	Stok	Price	Original price
PakCHIPS Crispy Banana Cracker (60g)	17% OFF	Stok: 4	RM 6.50	~~RM 7.80~~
PakCHIPS Pumpkin Chips (50g)	17% OFF	Stok: 3	RM 6.50	~~RM 7.80~~
Ah Huat Kopitiam White Coffee (15's x 30g)	23% OFF	Stok: 3	RM 12.80	~~RM 16.70~~
Sherif Teabag Teapot (50s X 2g)	17% OFF	Stok: 4	RM 4.50	~~RM 5.40~~
MDEE Pati Sos Goreng Mamak (350g)	17% OFF			
MDEE Pati Sos Char Koay Teow (350g)	17% OFF			
Habhal Chilli Sauce (500g)	15% OFF			
Samza's Sambal Hijau (200g)	17% OFF			
Adabi Lazat Ketupat Mini Value Pack (600g)	19% OFF			
Teego Dates Juice Drink (400ml) - Lemon				

Chat with us now

Hello 👋 Welcome to MYDIN!
Let me know if you have any questions!

Waiting for mydinonline.bdegate.com...

Figure 5.3.3 – Access Target Web Page

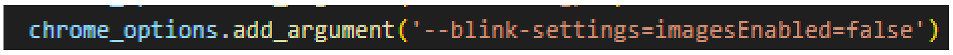

```
chrome_options.add_argument('--blink-settings=imagesEnabled=false')
```

Figure 5.3.4 – Configuration of Chromedriver

The argument *“--blink-settings=imagesEnabled=false”* is used in the Chrome WebDriver configuration to optimize the web scraping process by disabling the loading of images when the web page is rendered in the browser.

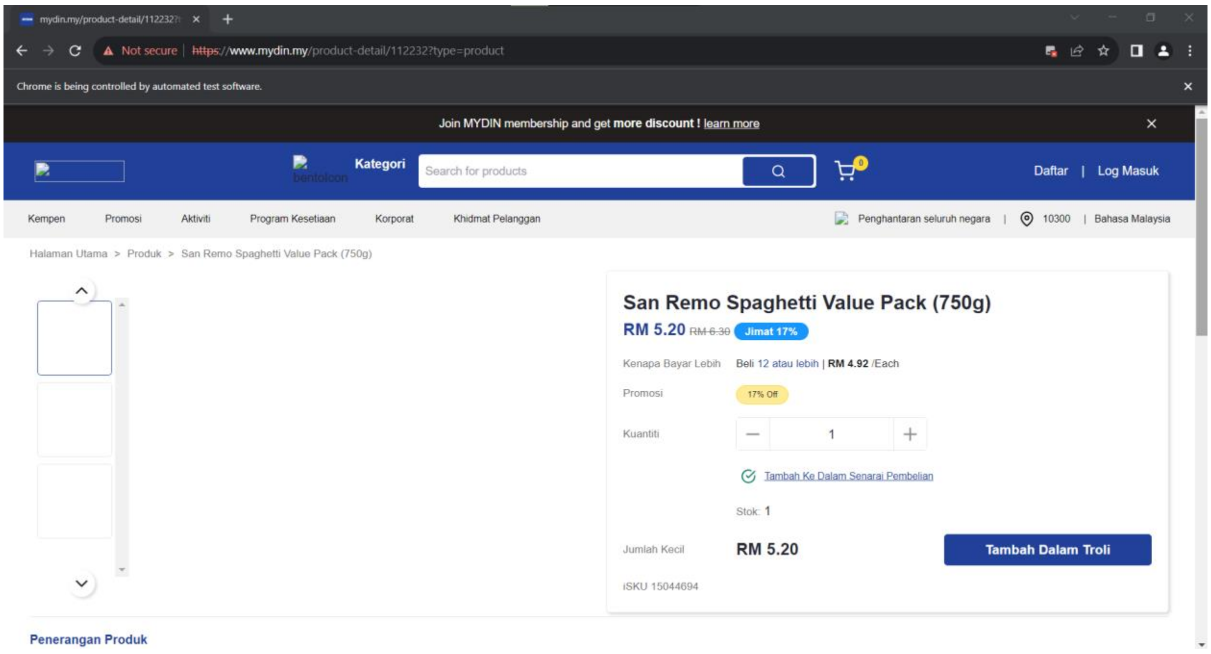

mydin.my/product-detail/112232?
Not secure
https://www.mydin.my/product-detail/112232?type=product
Chrome is being controlled by automated test software.
Join MYDIN membership and get more discount ! learn more
Kategori
Search for products
Daftar | Log Masuk
Kempen
Promosi
Aktiviti
Program Kesetiaan
Korporat
Khidmat Pelanggan
Penghantaran seluruh negara | 10300 | Bahasa Malaysia
Halaman Utama > Produk > San Remo Spaghetti Value Pack (750g)
San Remo Spaghetti Value Pack (750g)
RM 5.20 ~~RM 6.30~~ Jimat 17%
Kenapa Bayar Lebih
Beli 12 atau lebih | RM 4.92 /Each
Promosi
17% Off
Kuantiti
1
Tambah Ke Dalam Senarai Pembelian
Stok: 1
Jumlah Kecil
RM 5.20
Tambah Dalam Troli
iSKU 15044694
Penerangan Produk

Figure 5.3.5 – Product Detail Page

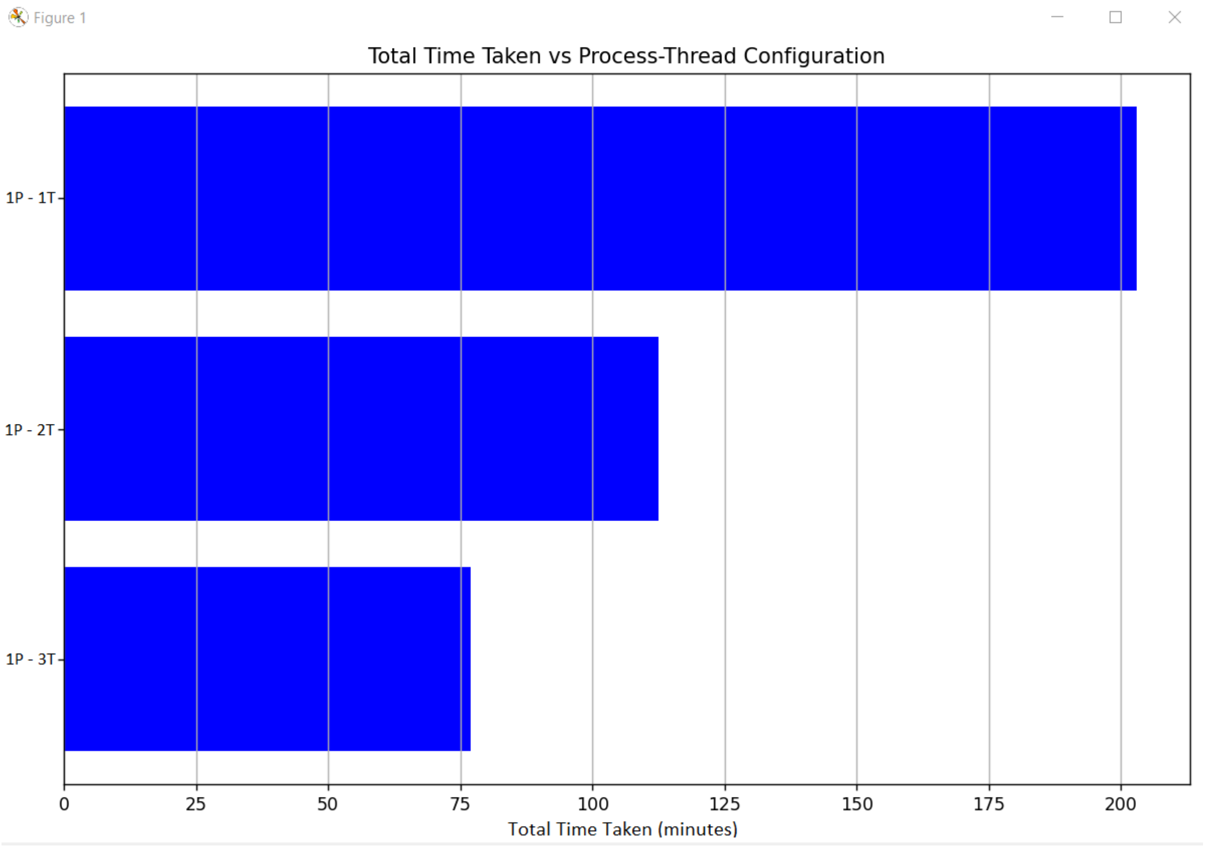
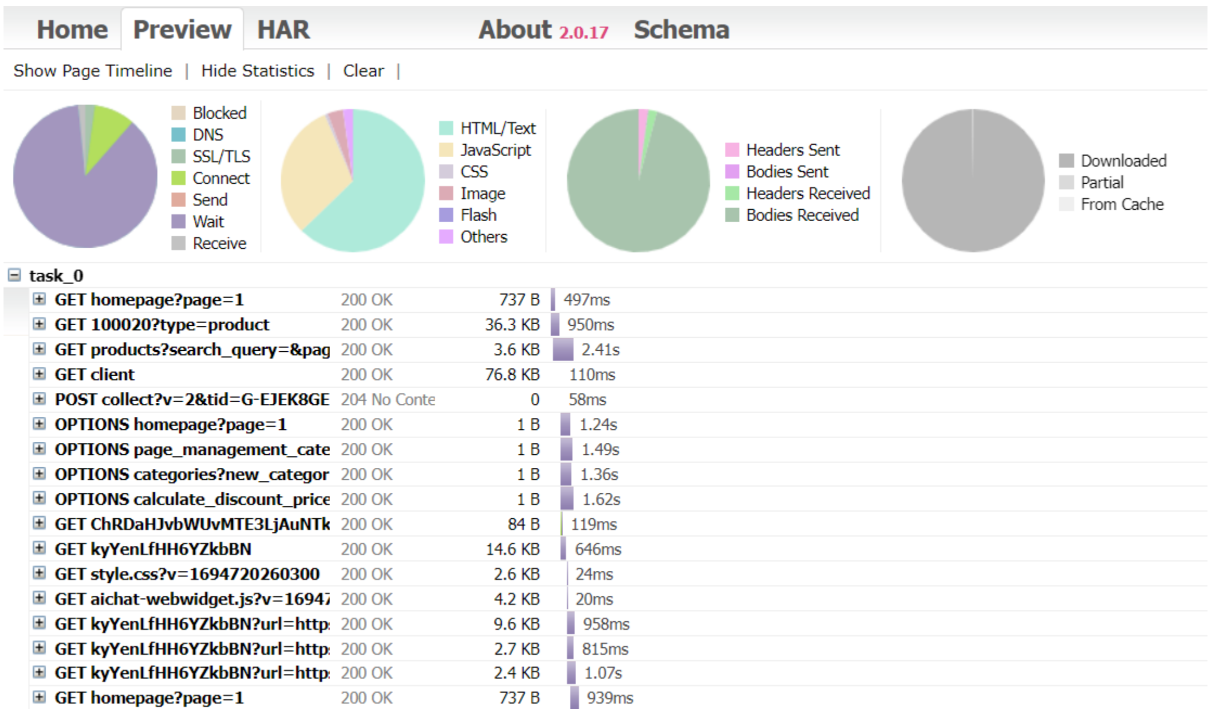


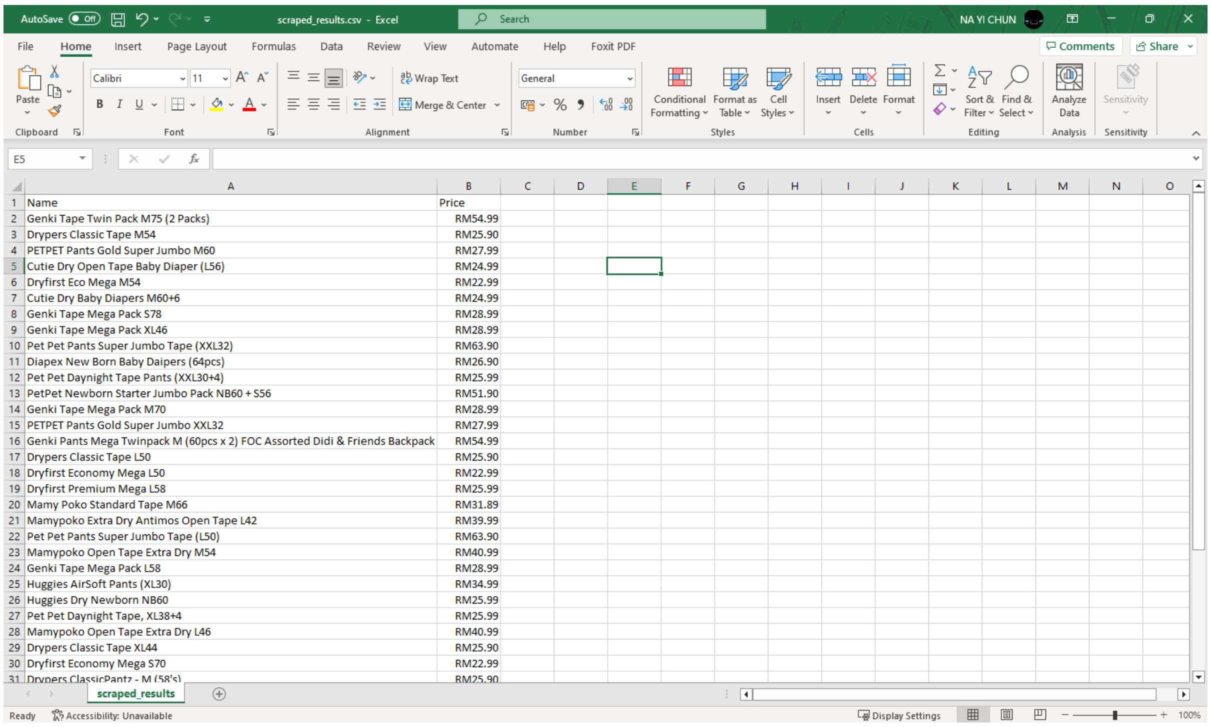
Figure 5.3.6 – Response Time



task_0

GET homepage?page=1	200 OK	737 B	497ms
GET 100020?type=product	200 OK	36.3 KB	950ms
GET products?search_query=&pag	200 OK	3.6 KB	2.41s
GET client	200 OK	76.8 KB	110ms
POST collect?v=2&tid=G-EJEK8GE	204 No Conte	0	58ms
OPTIONS homepage?page=1	200 OK	1 B	1.24s
OPTIONS page_management_cate	200 OK	1 B	1.49s
OPTIONS categories?new_categor	200 OK	1 B	1.36s
OPTIONS calculate_discount_price	200 OK	1 B	1.62s
GET ChRDaHJvbWUvMTE3LjAuNTk	200 OK	84 B	119ms
GET kyYenLfHH6YZkbBN	200 OK	14.6 KB	646ms
GET style.css?v=1694720260300	200 OK	2.6 KB	24ms
GET aichat-webwidget.js?v=16947	200 OK	4.2 KB	20ms
GET kyYenLfHH6YZkbBN?url=http	200 OK	9.6 KB	958ms
GET kyYenLfHH6YZkbBN?url=http	200 OK	2.7 KB	815ms
GET kyYenLfHH6YZkbBN?url=http	200 OK	2.4 KB	1.07s
GET homepage?page=1	200 OK	737 B	939ms

Figure 5.3.7 – Network Bandwidth Data


scraped_results.csv - Excel

	A	B
1	Name	Price
2	Genki Tape Twin Pack M75 (2 Packs)	RM54.99
3	Drypers Classic Tape M54	RM25.90
4	PETPET Pants Gold Super Jumbo M60	RM27.99
5	Cutie Dry Open Tape Baby Diaper (L56)	RM24.99
6	Dryfirst Eco Mega M54	RM22.99
7	Cutie Dry Baby Diapers M60+6	RM24.99
8	Genki Tape Mega Pack S78	RM28.99
9	Genki Tape Mega Pack XL46	RM28.99
10	Pet Pet Pants Super Jumbo Tape (XXL32)	RM63.90
11	Diapex New Born Baby Daipers (64pcs)	RM26.90
12	Pet Pet Daynight Tape Pants (XXL30+4)	RM25.99
13	PetPet Newborn Starter Jumbo Pack NB60 + S56	RM51.90
14	Genki Tape Mega Pack M70	RM28.99
15	PETPET Pants Gold Super Jumbo XXL32	RM27.99
16	Genki Pants Mega Twinpack M (60pcs x 2) FOC Assorted Didi & Friends Backpack	RM54.99
17	Drypers Classic Tape L50	RM25.90
18	Dryfirst Economy Mega L50	RM22.99
19	Dryfirst Premium Mega L58	RM25.99
20	Mamy Poko Standard Tape M66	RM31.89
21	Mamypoko Extra Dry Antimos Open Tape L42	RM39.99
22	Pet Pet Pants Super Jumbo Tape (L50)	RM63.90
23	Mamypoko Open Tape Extra Dry M54	RM40.99
24	Genki Tape Mega Pack L58	RM28.99
25	Huggies AirSoft Pants (XL30)	RM34.99
26	Huggies Dry Newborn NB60	RM25.99
27	Pet Pet Daynight Tape, XL38+4	RM25.99
28	Mamypoko Open Tape Extra Dry L46	RM40.99
29	Drypers Classic Tape XL44	RM25.90
30	Dryfirst Economy Mega S70	RM22.99
31	Drypers ClassicPantz - M (58's)	RM25.90

scraped_results

Figure 5.3.8 – Scraped Result

5.4 Implementation Issues and Challenges

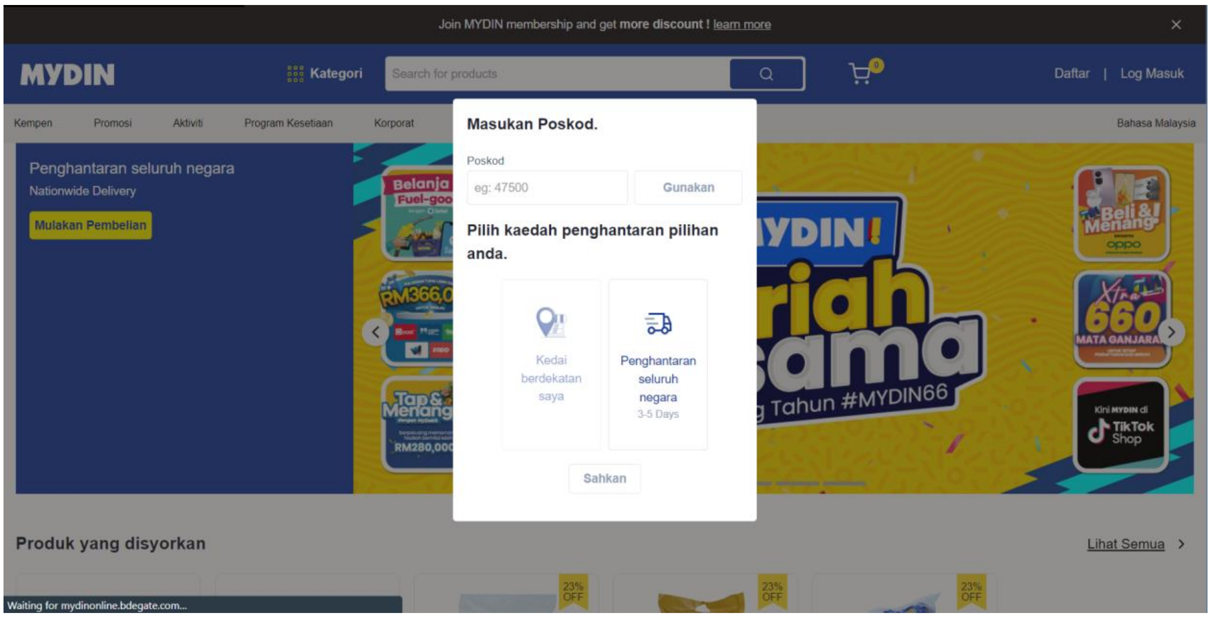


Figure 5.4.1 – Pop Up Window (Mydin)

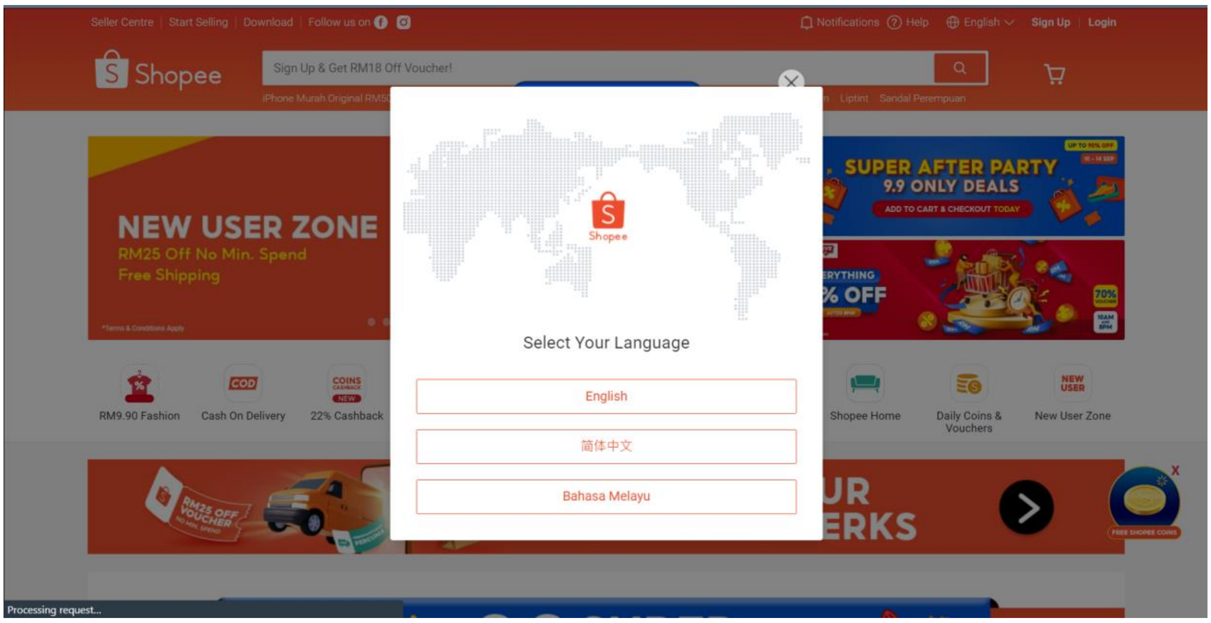


Figure 5.4.2 – Pop Up Window (Shopee)

Even though this project has been designed to adapt to different kinds of web page URLs and different HTML structures of the web pages and Selenium provides methods to interact with the pop-up elements to dismiss them, the library still cannot foresee the structure of pop-up windows when user prompt new or unseen web pages. This pop-up may block access to the underlying page until they are closed or interacted with. This can prevent the scraper from accessing the desired data prompted by the user earlier.

Another issue such that some Shopee URLs do not want their data to be scraped by bots, because they may lose their competitive advantage, violate their terms of service, or expose their customers' privacy. Therefore, they have a strong anti-bot system to prevent or limit web crawling and web scraping activities on their websites. This may happen to other websites as well, especially popular platforms such as Amazon, Lazada, etc. Hence, it restricts the practicality of this project.

5.5 Concluding Remark

Although there were limitations in the project's implementation as mentioned earlier, it can still be considered a success as it has produced a functional system that achieves its main goal. However, it is important to acknowledge that the system still has to be applicable to a wide range of users to prove its usefulness and relevance to a diverse user base. Therefore, it is essential to come up with creative solutions to address these issues.

Chapter 6 - System Evaluation and Discussion

6.1 System Testing and Performance Metrics

Type of Testing	Context
Performance Testing	The system's performance should be measured using attributes such as response time and network bandwidth data. The performance should be compared among scraping processes that use different numbers of threads.
Scalability Testing	The system should be tested to ensure that it can handle an increasing number of requests and maintain its performance and response time.

Table 6.1.1 – System Testing

6.2 Testing Result

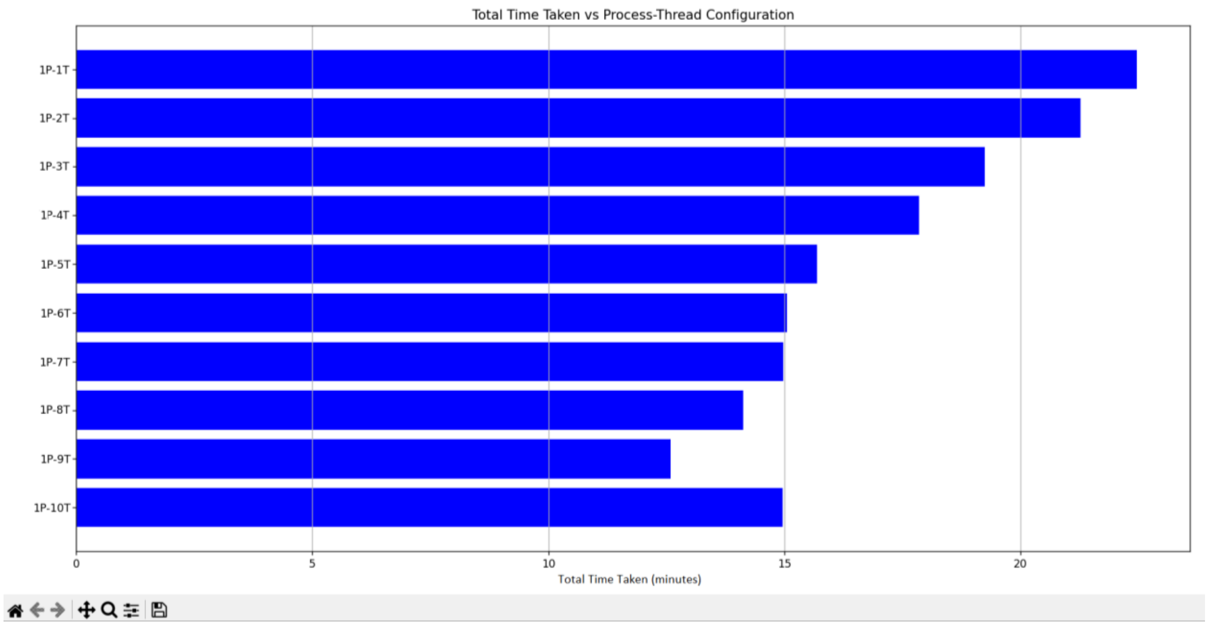


Figure 6.2.1 – Response Time for Task 1

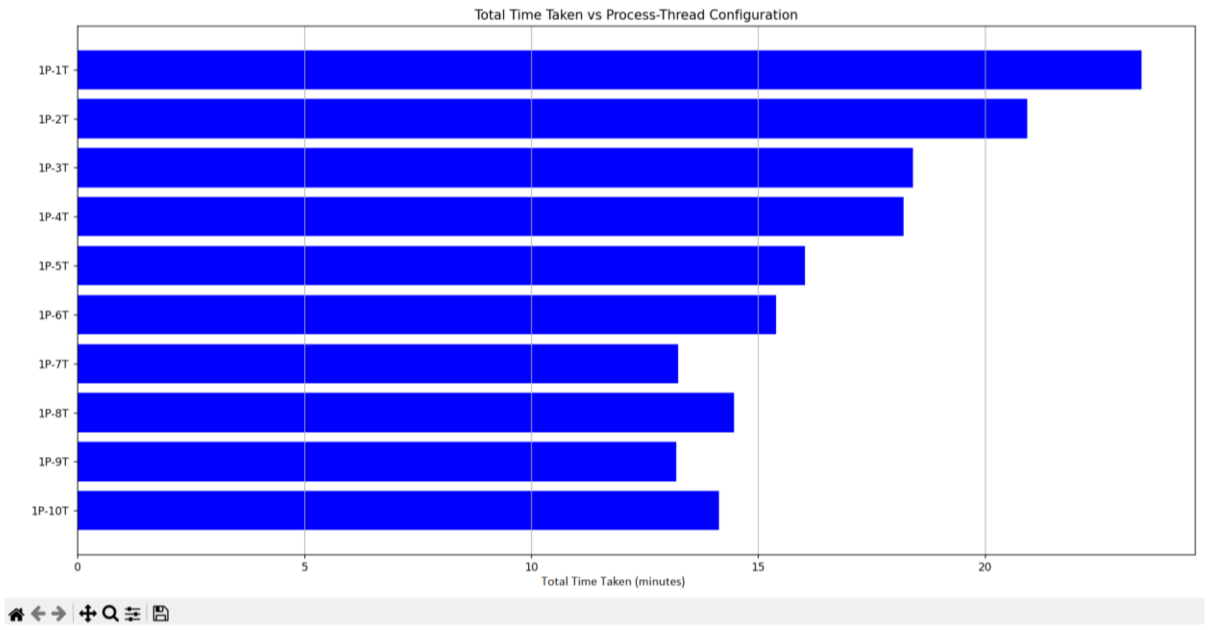


Figure 6.2.2 – Response Time for Task 2

Task 1 is performed by scraping 100 products from an e-commerce website while Task 2 scrapes 200 products from the same e-commerce website. In terms of performance testing, based on Figure 6.2.1, the initial linear scaling of response time from 1 thread to 9 threads suggests that the web scraping solution is efficiently utilizing additional threads to handle increased workloads. This efficiency indicates that increasing the number of threads results in faster processing times and improved concurrency, which is a positive output. However, the rebound in response time at 10 threads indicates that there might be resource limitations or bottlenecks that start to impact performance when too many threads are used concurrently. On top of that, if there are more threads than available CPU cores, the system may still execute tasks concurrently, but the threads have to switch between one another more frequently, which may potentially lead to overhead and increased response times. On the other hand, the pattern in Figure 6.2.2 is somehow similar to the pattern in Figure 6.2.1, which is considered a good result for scalability testing as the response time and performance is almost the same.

6.3 Project Challenges

For concurrency issues such as the presence of Global Interpreter Lock (GIL) in Python, which ensures that only one thread can execute Python code at a time within a single process. This means that Python threads may not fully utilize the CPU resources for CPU-bound tasks. However, when implementing multithreading for parallel tasks, generating and switching between threads still experiences some overhead despite having the GIL disabled to eliminate the restriction of using only one thread to execute Python code.

6.4 Objectives Evaluation and Concluding Remark

It is hard to determine whether the project's objectives have been completely fulfilled. However, the proposed library can scrape the target data successfully faster than using default Selenium to do so by only prompting the user to perform simple configurations such as providing a target URL, desired web page elements to be scraped, and determining the number of threads to initiate the asynchronous scraping tasks. The target website can be an e-commerce website with dynamic content, which consists of heavy client-side rendering using JavaScript frameworks. However, the system still requires the user to provide the updated HTML structure such as CSS Selector or XPATH to relocate the target elements on the web page. For objective 2, the system will be improved to use multiple machines to generate more threads to further enhance the performance and scalability of the scraping process.

Chapter 7 - Conclusion and Recommendation

7.1 Conclusion

The main purpose of this project is to help whoever is suffering from the long waiting time to scrape desired information much faster than usual regardless of whether the data is small-scale or large-scale from the internet through the integration of web scraping technologies and distributed computer system. While the suggested approach may not exhibit remarkable performance in terms of multi-computing, it undeniably excels in terms of the performance metrics outlined in Chapter 6. Hopefully, the aim can be accomplished with the web scraping solution proposed in this project.

7.2 Recommendations

To enhance the performance of the proposed solution further, a highly effective approach would be exploring distributed scraping solutions, wherein multiple machines or instances collaborate to scrape websites concurrently. Tools such as Scrapy Cluster and distributed task queues can be considered as well in distributing the workload. By embracing this strategy, web scraping operations can scale horizontally so that they can accommodate increased data volumes and more sophisticated scraping tasks as user demands expand.

REFERENCES

[1] R. S. Chaulagain, S. Pandey, S. R. Basnet and S. Shakya, "Cloud Based Web Scraping for Big Data Applications," in *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, New York,, IEEE, 2017, pp. 138-143.

[2] R. Lawson, Web Scraping with Python, Packt Publishing, 2015.

[3] B. Zhao, "Web Scraping," in *Encyclopedia of Big Data*, Corvallis, Springer International Publishing AG, 2017, pp. 1-3.

[4] G. Richards, N. DeVito and P. Inglesby, "How We Learnt to Stop Worrying and Love Web Scraping," *Nature*, vol. 585, no. 7826, pp. 621-622, September 2020.

[5] M. Kumar and P. Neelima, "Design and Implementation of Scalable, Fully Distributed Web Crawler for a Web Search Engine," *International Journal of Computer Applications*, vol. 15, no. 7, pp. 8- 13, February 2011.

[6] P. Ramya, V. Sindhura and P. V. Sagar, "Testing using selenium web driver," in *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, IEEE, 2017, pp. 1-7.

[7] G. Fu, Y. Zhang and G. Yu, "A Fair Comparison of Message Queuing Systems," *IEEE Access*, vol. 9, pp. 421-432, 2021.

[8] F. Arévalo, M. R. Diprasetya and A. Schwung, "A Cloud-based Architecture for Condition Monitoring based on Machine Learning," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, Porto, IEEE, 2018, pp. 163-168.

[9] G. Meesters, "Synchronising Distributed Scraping," Open University of the Netherlands, Heerlen, 2021.

[10] F. Yin, X. He and Z. Liu, "Research on Scrapy-Based Distributed Crawler System for Crawling Semi-structure Information at High Speed," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, Chengdu, 2018.

APPENDIX

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 4**
Student Name & ID: Na Yi Chun – 20ACB05770	
Supervisor: Ts Dr Ooi Boon Yaik	
Project Title: Parallelizing Web Scraping to Improve Performance and Scalability	

1. WORK DONE [Please write the details of the work done in the last fortnight.] Performed literature review on message queue algorithm and Flask.
2. WORK TO BE DONE Find a suitable website to scrape for a large volume of data.
3. PROBLEMS ENCOUNTERED Scraped product details from Shopee during Project I but failed to scrape now due to its strong antibot system.
4. SELF EVALUATION OF THE PROGRESS Still following the timeline.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 6**
Student Name & ID: Na Yi Chun – 20ACB05770	
Supervisor: Ts Dr Ooi Boon Yaik	
Project Title: Parallelizing Web Scraping to Improve Performance and Scalability	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

Successfully scraped the movie details of IMDB Top 250 TV Shows.

2. WORK TO BE DONE

Modify the scraping code to enable multiprocessing and multithreading.

3. PROBLEMS ENCOUNTERED

The process of scraping all 250 movie details is very slow.

4. SELF EVALUATION OF THE PROGRESS

Still following the timeline.

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 10**
Student Name & ID: Na Yi Chun – 20ACB05770	
Supervisor: Ts Dr Ooi Boon Yaik	
Project Title: Parallelizing Web Scraping to Improve Performance and Scalability	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

Have modified code that includes multiprocessing and multithreading capability.

2. WORK TO BE DONE

Implement the message queue algorithm to further improve the solution.
Find out the problem of why the result does not work as what theory says.

3. PROBLEMS ENCOUNTERED

The performance should improve further as the number of threads increases, but the scraping process failed to improve further when 5 threads and above were used.

4. SELF EVALUATION OF THE PROGRESS

Still following the timeline.

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 12**
Student Name & ID: Na Yi Chun – 20ACB05770	
Supervisor: Ts Dr Ooi Boon Yaik	
Project Title: Parallelizing Web Scraping to Improve Performance and Scalability	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Have integrated the message queue algorithm into the existing code.

2. WORK TO BE DONE

Find out the way to disable GIL in Python to enable concurrency.
Find a new use case to support the usefulness of this proposed solution.

3. PROBLEMS ENCOUNTERED

IMDB was not a suitable use case for this project.

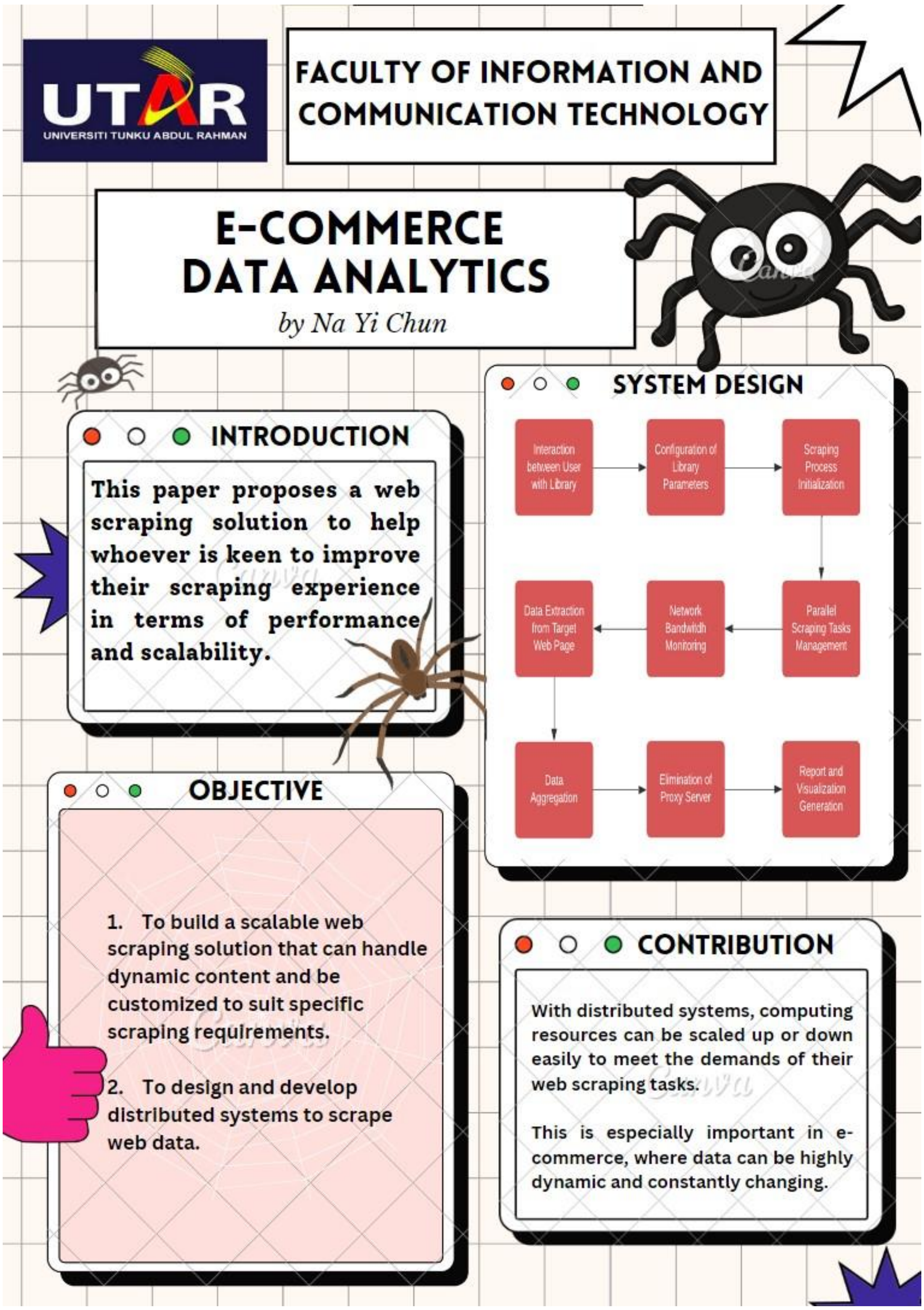
4. SELF EVALUATION OF THE PROGRESS

Still following the timeline.

[illegible]
Supervisor's signature

[illegible]
Student's signature

POSTER



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

E-COMMERCE DATA ANALYTICS

by Na Yi Chun

INTRODUCTION

This paper proposes a web scraping solution to help whoever is keen to improve their scraping experience in terms of performance and scalability.

SYSTEM DESIGN

Interaction between User with Library
Configuration of Library Parameters
Scraping Process Initialization
Parallel Scraping Tasks Management
Network Bandwidth Monitoring
Data Extraction from Target Web Page
Data Aggregation
Elimination of Proxy Server
Report and Visualization Generation

OBJECTIVE

1. To build a scalable web scraping solution that can handle dynamic content and be customized to suit specific scraping requirements.

2. To design and develop distributed systems to scrape web data.

CONTRIBUTION

With distributed systems, computing resources can be scaled up or down easily to meet the demands of their web scraping tasks.

This is especially important in e-commerce, where data can be highly dynamic and constantly changing.

PLAGIARISM CHECK RESULT

Turnitin Originality Report
20ACB05770_FYP2_Turnitin by Na Yi Chun
From FYP (FYP)

- Processed on 15-Sep-2023 09:26 +08
- ID: 2166473097
- Word Count: 5554

Similarity Index
6%
Similarity by Source

Internet Sources:
4%
Publications:
1%
Student Papers:
4%

sources:

1 1% match (Internet from 19-May-2016)
http://www.ijcaonline.org/volume15/number7/pxc3872629.pdf

2 1% match (Fulian Yin, Xiating He, Zhixin Liu. "Research on Scrapy-Based Distributed Crawler System for Crawling Semi-structure Information at High Speed", 2018 IEEE 4th International Conference on Computer and Communications (ICCC), 2018)
Fulian Yin, Xiating He, Zhixin Liu. "Research on Scrapy-Based Distributed Crawler System for Crawling Semi-structure Information at High Speed", 2018 IEEE 4th International Conference on Computer and Communications (ICCC), 2018

3 < 1% match (student papers from 24-Apr-2023)
Submitted to Universiti Tunku Abdul Rahman on 2023-04-24

4 < 1% match (student papers from 28-Apr-2023)
Submitted to Universiti Tunku Abdul Rahman on 2023-04-28

5 < 1% match (student papers from 17-Apr-2023)
Submitted to Universiti Tunku Abdul Rahman on 2023-04-17

6 < 1% match (student papers from 04-Sep-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-09-04

7 < 1% match (student papers from 15-Apr-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-04-15

8 < 1% match (Internet from 30-Mar-2023)
http://eprints.utar.edu.my/4755/1/fyp_IB_2022_OWK.pdf

9 < 1% match (Internet from 30-Mar-2023)
http://eprints.utar.edu.my/4732/1/fyp_IB_2022_CJS.pdf

10 < 1% match (Internet from 11-Oct-2022)
http://eprints.utar.edu.my/4262/1/17ACB01592_FYP.pdf

11 < 1% match (student papers from 07-Dec-2022)
Submitted to Liverpool John Moores University on 2022-12-07

12 < 1% match (Internet from 03-Oct-2022)
https://link.springer.com/chapter/10.1007/978-3-030-90885-0_1?code=9c961edd-07a4-4668-a19e-e8bc8ca7afa4&error=cookies_not_supported

13 < 1% match (Internet from 05-Aug-2023)
http://sandbox.ijcaonline.org/archives/volume15/number7/1963-2629?format=pdf

14 < 1% match (student papers from 14-Jun-2023)
Submitted to International Islamic University Malaysia on 2023-06-14

15 < 1% match (student papers from 26-Jun-2023)
Submitted to University of Maryland, Global Campus on 2023-06-26

16 < 1% match (Internet from 30-Aug-2023)
https://vocal.media/education/the-key-to-unlocking-a-6-figure-income-create-your-own-online-community

17 < 1% match (student papers from 02-Aug-2023)
Submitted to Asia e University on 2023-08-02

18 < 1% match (Fernando Arevalo, Mochammad Rizky Diprasetya, Andreas Schwung. "A Cloud-based Architecture for Condition Monitoring based on Machine Learning", 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), 2018)
Fernando Arevalo, Mochammad Rizky Diprasetya, Andreas Schwung. "A Cloud-based Architecture for Condition Monitoring based on Machine Learning", 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), 2018

19 < 1% match (Internet from 02-May-2020)
https://docplayer.net/12147155-Customer-education-on-mobile-banking.html

20 < 1% match (Internet from 30-Sep-2022)
https://publikationen.sulb.uni-saarland.de/bitstream/20.500.11880/32108/1/phdthesis.pdf

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Na Yi Chun
ID Number(s)	20ACB05770
Programme / Course	Computer Science
Title of Final Year Project	Parallelizing Web Scraping to Improve Performance and Scalability

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 6 % **Similarity by source** Internet Sources: 4 % Publications: 1 % Student Papers: 4 %	
Number of individual sources listed of more than 3% similarity: 0	
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Alex

______________________________ Signature of Supervisor

Name: Ooi Boon Yaik

Date: 15 SEPTEMBER 2023

______________________________ Signature of Co-Supervisor

Name: ______________________

Date: ______________________

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB05770
Student Name	Na Yi Chun
Supervisor Name	Ts Dr Ooi Boon Yaik

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

[illegible]

____

(Signature of Student)
Date: 15/9/2023