PARALLEL WEB CRAWLER FOR IMAGE DOWNLOADS

A PROJECT REPORT

Submitted by

Moukhik Misra – 19BCE2190

Mohit Suhasaria – 19BCE2167

CSE4001 Parallel and Distributed Computing

Slot – L25+L26 (B2)

Computer Science and Engineering

DECEMBER 2021

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CONTENTS

Chapter	Page No.
ABSTRACT	4
Chapter 1: INTRODUCTION	4
Chapter 2: LITERATURE REVIEW	6
2.1 An Image Crawler for Content Based Image Retrieval System	6
2.2 Parallel Crawler Architecture and Web Page Change Detection	7
2.3 RCrawler: An R package for parallel web crawling and scraping	7
2.4 A Survey on Crawlers used in developing Search Engine	8
Chapter 3: PROBLEM FORMULATION	9
3.1 Objectives	9
Chapter 4: METHODOLOGY	10
4.1 Running the Scripts That Will Get the Element on the Web Page using Selenium, Python and Chrome Driver	10
4.1.1 Importing the Necessary Packages	10
4.1.2 Collecting Links	10
4.2 Multithreaded Download of the Images from the Obtained Links	12
4.2.1 Importing the Necessary Packages	12
4.2.2 Implementing the Functionality of The Downloader	13
Chapter 5: RESULTS AND DISCUSSIONS	16
5.1 Full Resolution Image Downloads	16

5.2 Full Resolution Downloaded Images in Folders	20
5.3 Thumbnail Image Downloads and Execution Time Comparison with Respect to 1, 2, 4 and 8 Threads for 10 Image Downloads	21
5.3.1 1 Thread and Limit 10	21
5.3.2 2 Threads and Limit 10	24
5.3.3 4 Threads and Limit 10	26
5.3.4 8 Threads and Limit 10	28
5.3.5 Comparison of Execution Times	30
5.4 GUI of Web Crawler	31
5.5 Downloaded Thumbnail Images	33
Chapter 6: CONCLUSION	36
Chapter 7: REFERENCES	37

ABSTRACT

In this project, we design a parallel web crawler by implementing multithreading concept. Web crawlers are used by search engines for indexing of webpages. A web crawler bot is like someone who goes through all the books in a disorganized library and puts together a card catalogue so that anyone who visits the library can quickly and easily find the information they need. To help categorize and sort the library's books by topic, the organizer will read the title, summary, and some of the internal text of each book to figure out what it's about. The goal of our proposed bot is to crawl websites for images of a particular keyword and automating the download process for matching images. Multi-threading is used to speed up the crawling as the internet has a vast scope and it would be inefficient to use a single process to crawl the internet. The project will help in downloading images of keywords that can be given to the program and the web crawler will scrape google images for those particular keywords and download the images and place them in separate folders based on keywords. The downloaded images maybe used by the user as per their choice.

1. INTRODUCTION

A web crawler is considered to be a search engine bot or a spider that downloads and indexes content from all over the Internet. The goal of such a bot is to learn what (almost) every webpage on the web is about, so that the information can be retrieved when it's needed. They're called web crawlers because crawling is the technical term for automatically accessing a website and obtaining data via a software program. These bots are almost always operated by search engines. By applying a search algorithm to the data collected by web crawlers, search engines can provide relevant links in response to user search queries, generating the list of webpages that show up after a

user types a search into Google or Bing (or another search engine). A web crawler bot is like someone who goes through all the books in a disorganized library and puts together a card catalog so that anyone who visits the library can quickly and easily find the information they need. To help categorize and sort the library's books by topic, the organizer will read the title, summary, and some of the internal text of each book to figure out what it's about. A crawler is a program that downloads and stores Web pages, often for a Web search engine. Roughly, a crawler starts off by placing an initial set of URLs, in a queue, where all URLs to be retrieved are kept and prioritized.
From this queue, the crawler gets a URL (in some order), downloads the page, extracts any URLs in the downloaded page, and puts the new URLs in the queue. This process is repeated until the crawler decides to stop. Collected pages are later used for other applications, such as a Web search engine or a Web cache. As the size of the Web grows, it becomes more difficult to retrieve the whole or a significant portion of the Web using a single process. Therefore, many search engines often run multiple processes in parallel to perform the above task, so that download rate is maximized. We refer to this type of crawler as a parallel crawler

A parallel crawler is a crawler that runs multiple processes in parallel. The goal is to maximize the download rate while minimizing the overhead from parallelization and to avoid repeated downloads of the same page. To avoid downloading the same page more than once, the crawling system requires a policy for assigning the new URLs discovered during the crawling process, as the same URL can be found by two different crawling processes. In this project we have created a web crawler that automates searching of the keywords that is, it scrapes the web for the keywords that have been provided by the user on Google Images and provides a multi-threading solution for the download of the images that have been scraped. Parallel web crawlers can be beneficial as compared to normal web crawlers in many scenarios such as: collecting data from multiple sources (multiple remote servers) instead of just a single source, performing long/complex operations on the collected data (such as doing image analysis or OCR) that could be done in parallel with fetching the data and collecting data from a large web service where you are paying for each query, or where creating multiple connections to the service is within the bounds of your usage agreement among other benefits. Parallel Web Crawlers also have great benefits when it comes to scalability, handling network loads and others. Parallelization is an extremely efficient optimization that can be implemented to a web crawler and that is the objective of our application.

2. LITERATURE REVIEW

2.1. An Image Crawler for Content Based Image Retrieval System.

The above paper designs an Image Crawler tool that collects and indexes group of web images available on the internet. This tool collects the keyword or phrase from the user to retrieve the images from the web. Then these collected keyword is applied to the different general search tools like Google, Yahoo, Bing etc, The collected web page information is stored in the temporary file till 200KB file size from the server. Then this file content is scanned and extract the image URL's and it is compared the URL which is present in the database to avoid the duplicate downloads. The extracted URL's images are downloaded and finally stores unique image and corresponding metadata like filename, url, size etc. in the database. The paper designs an image extraction framework which is quite flexible. Finally, the resulting images are used in the Content Based Image Retrieval (CBIR) system for extracting the relevant images need by the client using the content of the images rather than the text-based information.

2.2. Parallel Crawler Architecture And Web Page Change Detection

This paper puts forward a technique for parallel crawling of the web. The paper puts forward an architecture built on the lines of client server architecture. It discuses a fresh approach for parallel crawling the web using multiple machines and integrates the trivial issues of crawling also. A major part of the web is dynamic and hence, a need arises to constantly update the changed web pages. The authors have used a three-step algorithm for page refreshment. This checks for whether the structure of a web page has been changed or not, the text content has been altered or whether an image is changed. For The server we have discussed a unique method for distribution of URLs to client machines after determination of their priority index. Also, a minor variation to the method of prioritizing URLs on the basis of forward link count has been discussed to accommodate the purpose of frequency of update. The authors manage to design a highly scalable architecture. The issue of overlap of downloads by client crawlers are addressed, as the main server does not give the same URL to different clients who themselves check whether it is already existing or not.

2.3. RCrawler: An R package for parallel web crawling and scraping

In this paper, an R package was created to crawl, parse, store pages, extract contents, and produce data that can be directly employed for web content mining applications. The crawler has a highly optimized system, and can download a large number of pages per second while being robust against certain crashes and spider traps. In this paper, they describe the design and functionality of RCrawler, and report on their experience of implementing it in an R environment, including different optimizations. The crawler begins from a given website URL, provided by the user, and progressively fetches this and extracts new URLs (out- links). These in turn are added to the frontier list to be processed. The crawling process stops when all URLs in the frontier are pro- cessed. The process of a crawling operation is performed by several concurrent processes or nodes in parallel. Essentially, each running node is assigned to a physical control thread provided by the operating system, and works independently. After loading the RCrawler package, they test run the crawler under various configurations, varying the number of cores between one and eight and the number of connections (simultaneous requests) between one and 16. By observing the average number of crawled pages per minute for each scenario. results, it is evident that the crawler speed increases for more cores and connections.

2.4. A Survey on Crawlers used in developing Search Engine

With the increasing size of World Wide Web (WWW), obtaining meaningful information from the web is becoming a tedious task. Search Engine is developed for extracting information from the web. It works as an interface between the web and the user. The three important components of a search engine are: Crawler, Indexer and Page Ranking. Crawler is a component of search engine responsible for traversing webpages and fetching relevant links from the web. This represents huge dependency of any search engine on the crawlers. So, a detailed study about all the available crawlers for understanding the drawbacks and the insights about the working methodology undertaken is necessary before proceeding to develop a smart crawler. For developing a smart crawler which is future scope, a comparative analysis of widely used crawlers like Focused Crawler, Inference-based Crawler, Incremental Crawler, Parallel Crawler and Distributed Crawler is done by the authors. It was found that To obtain quality information, the search engine is developed using a crawler that is capable of bifurcating and accessing the good quality data from the web. Performance analysis of various crawlers is done in detail. Parallel crawlers work by running multiple instances of generic crawlers and distributes the load on the network thereby improving the efficiency of search engines.

3. PROBLEM FORMULATION

3.1. Objectives

The objectives of our project are listed below:

- To build a parallel web crawler that can download images based on keywords.
- To use Selenium to automate the process of Web Crawling
- To utilize multithreading provided by Selenium and Chrome Driver to crawl Google images for the image links.
- To develop a multithreaded downloader for the images that have been scraped by using the web crawler using the multiprocessing library in Python.
- To segregate the downloaded images in different folder based on their search keywords.
- To optimize the process of web crawling by parallelizing the crawler and the downloader leading to a web crawler with enhanced performance.
- To have different features for downloads such as full resolution downloads of images, no gui download of images for the provided keywords.
- To support multiple formats of images for downloading such as jpeg, gif, png.
- To verify the downloads of all the images by checking for any imbalance in the downloads once the process has finished.
- To obtain and compare the execution time of the application for 1, 2, 4 and 8 threads.

4. METHODOLOGY

4.1. Running The Scripts That Will Get The Element On The Web Page Using Selenium, Python And Chrome Driver

4.1.1 Importing the Necessary Packages

The first step involved in creating the parallel web crawler is importing the necessary libraries in python that are essential for creating the web crawler. These packages are time, selenium packages and web driver. We use multiple functions of selenium such as the Selenium web driver, Keys in the selenium.webdriver.common.keys and the By package form selenium.webdriver.common.by as well as the necessary exception handlers such as ElementNotVisibleException and StaleElementReferenceException from selenium.common.exceptions package. Next we need to import the platfrom package in Python followed by WebDriverWait form selenium.webdriver.support.ui. The next packages that need to be imported are the expected_conditions package as EC and the Options package from selenium.webdriver.chrome.options. The next package that needs to be imported is the ChromeDriverManager from webdriver_manager.chrome. Lastly the os.path library has to be imported.

4.1.2 Collecting Links

A function is defined to collect the links of the images that are to be downloaded. This function is defined as __init__ with the parameters of self, no_gui , proxy and executable. The first step is to check the platform that the application is running on. This is achieved by using the platform.system() function to detect the OS. This is necessary step so as to have the correct version of the chromedriver which is essential for automating the tasks and for handling the downloads of images. The application can run on Windows, Linux and Mac OSX. If the application tries running on any other operating system, we throw an OSError to the user. This detection of OS help us set the executable directory of the chromedriver whose functionality is mentioned above. Next it is important to check whether the executable file exists in the necessary directory. If not then we can raise a FileNotFoundError.

Next we have to set the options for the chromedriver, The settings can be toggled by the user. We can have the arguments for no sandbox, to disable dev shm usage, to have no gui, to have proxy servers and then ewe can install the chromedriver with the help of the Chrome Driver Manager with the provided options. Next we can pass parameters such as browser version, chromedriver version and major_version_different and set these values that are as required and that have been detected. The compatibility of the driver and the browser is essential for the functioning of the application. Next, we display the obtained information to the user. We print information such as the current web browser version, the current chrome driver version and we can give a warning if there is a difference in the major versions of the above two.

In order to automate the scraping of images and to provide an immersive GUI, we can automate the scrolling of the page by defining a script to do the same. We can also have a wait and click function which uses the WebDriverWait to wait until an element is clickable. This is done by means of xpath. In Selenium automation, if the elements are not found by the general locators like id, class, name, etc. then XPath is used to find an element on the web page. Next we can have a highlight function to execute a script to set the style as per requirements. For a parallel web crawler, it is important to handle the efficiency of the crawling and to ensure that there is no duplicate downloads that occur in the images that are going to be downloaded by the application. We also define the link to google images which is the site from which the image links and the images are going to be downloaded from. Elements can be found on a page by means of their html tags, or by the body or by their xpath. Once we can obtain a box or element on a page that contains an image, we can check by tag name of img and we can obtain their source by using the get_attribute function. Now goolge images preloades 20 images as base64 and we can also add then to list of source links. Once the link collection is complete, the user can be notified of this fact and it can displayed on the console that the links have been successfully collected. If an error occurs while obtaining the links, we can throw the necessary exceptions.

For scraping google images, it is necessary to have the proper query for the links to google images. This means that we have to set the link with variable search queries in the form of a formattable sting with enterable keywords and url. While passing the query, we can add a timer for the auto scroll functionality and then the crawler can get to work on scraping the necessary images on the page. The images are searched by their xpath and then once the xpath has been determined, we can find the images in the page by use of their tags. if any errors are caught then we can throw the necessary exception statements. Lastly we can collect all the links and display the broad statistics to the user.

4.2. Multithreaded Download of The Images from The Obtained Links

4.2.1 Importing the Necessary Packages

Now that the required links have been collected by using Selenium and the Chrome Web Driver, in parallel, we can now go ahead and create the functionality for the downloads of the images from Chrome. To create this multithreaded downloader, we require a few packages in Python. Firstly, we need to import the os library, next we need to import the requests package followed by the shutil package. The most important package that needs to be imported is the Python multiprocessing package which will form the basis of our multithreaded downloader. The next package that we need to import is the argparse package followed by the collect_links package and then the imghdr package, the base 64 package and the pathlib package and finally the random package.

4.2.2 Implementing the Functionality of The Downloader

Once the packages have been successfully implemented, we can move ahead to implement the functionality of the multithreaded downloader. We can have various features such as full resolution downloads, no gui downloads for better efficiency and skipping already existing downloads, limiting the number of downloads that are to be made. Firstly we create a parent directory called downloads where the downloaded images will be stored. Next we can implement a functionality where the images can get downloaded to separate folders based on the keyword being searched such as images of dogs being stored in the dog folder and so on. Next we need to create an acceptable list of downloadable image extensions. The image extensions supported by this program are jpg, gif and png. To make the new directories, we obtain the current working directory and create a folder named downloads. We can define a function to validate the extension of the image that is to be downloaded and check whether it falls within the acceptable formats of the images that have been defined in the application.

Next, for the keywords, we can create a text file that contains the keywords of all the necessary searches that the user wants to perform that is the subjects of the images that the user wants. Next for the application obtain the necessary keywords, we can read the file using the open() function in python and open the file known as keywords.txt. Next we can read the text line by line using the read() function. Next we can sort these keywords by alphabet and re save them in the same file by using the write() function. Both of these actions use utf-8 encoding for the character set.

After this we can define a function which can save an object to a file. We pass the object as a parameter to the function and use the write() function in python to write the object to the file if it is in base 64 encoding. If the image is not in base 64 encoding we can use the shutil.copyfileobj() function to save the raw data of the object to the file. Next we can define a function that converts base 64 to object. For this we can use the base64.decodebytes() function provided by python and save it as an object with utf-8 encoding and then return the data.

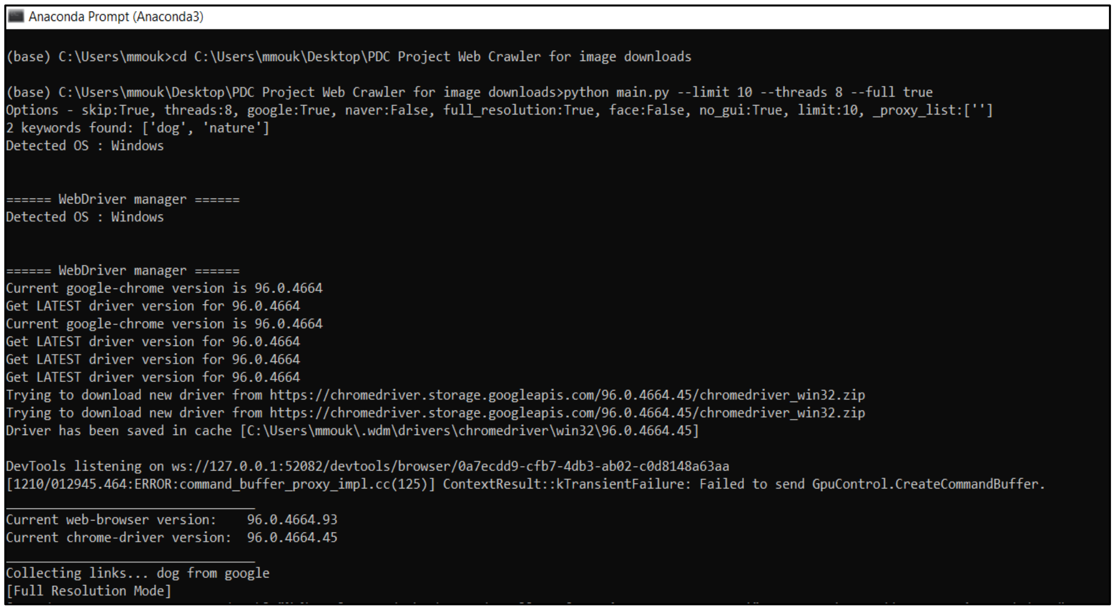
Next important step is to define the function which will download the images. We obtain the links from the web crawler link and check its length to determine the number of runs needed. We keep count of the number of successful downloads that have occurred and the max count of downloads specified by the user. Next we can determine if the image is in base_64 format or not depending on whether it has been preloaded y google images. If the image has data:image/jpeg;base64 then we can determne that the extension is.jpg and that is it in base 64. We can do the same for the other extesions such as png and gif. Otherwise, we can use the requests.get() function to get the response of the link and we can determine the value of the extension by using a created function and set the base 64 boolean to false. Next we save object to file and increment the success count and delete the previous response.

Next, we pass the extension through a validation function to check whether the file is readable or unreadable and if the file is readable and not in the list of supported extensions, we manually change the extension to one which is present in the list of acceptable extensions. If all these methods fail then we can pass an Exception which prints out Download Failed. Next we define the function which invokes the download images function and this function is called the download_from_site() function. This function get the text from the site as well as the face url form the site. If a proxy has been set then it gives a random choice of proxy list of self. Next the chrome driver is initialized and the links are collected from google images and the way the application runs depends on the settings specified by the user. Next, the download of the data is performed by invoking the created functions.

An over arching crawling function exists which defines the multithreading of the application. For multithreading, we have used the multiprocessing package in python. We use the Pool() function to define the number of threads that are going to be used in the program for scraping as well as downloading. We can then define the tasks that will be performed by the threads. We use a non-blocking mapping methods by means of map_async() to map the tasks to thread and it is automatically handles by the Python interpreter. We can then close the parallel part of the program once the crawling and the downloads have completed. Lastly we run a check for data imbalance which means that we check if all the files are properly downloaded as per the limit set by the user. If the number of downloads is less than 50% of the count specified by the user, then it prompts the user to search the specified keyword again and to restart the download process. If the user approves it removes the directory of the keyword where the imbalance has been detected and prompts the user to re-run the program to re-download the removed files. Next we have the main function where we can set the default parameters and settings for the running of the program and we run the manager functions to ensure the successful running of the application.

5. RESULTS AND DISCUSSIONS

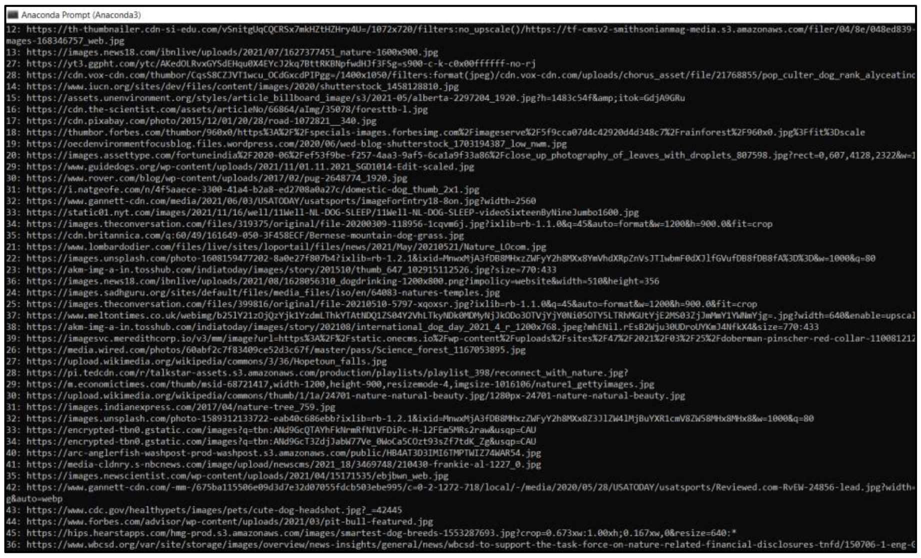
5.1. Full Resolution Image Downloads



```
Anaconda Prompt (Anaconda3)

(base) C:\Users\mmouk>cd C:\Users\mmouk\Desktop\PDC Project Web Crawler for image downloads

(base) C:\Users\mmouk\Desktop\PDC Project Web Crawler for image downloads>python main.py --limit 10 --threads 8 --full true
Options - skip:True, threads:8, google:True, naver:False, full_resolution:True, face:False, no_gui:True, limit:10, _proxy_list:['']
2 keywords found: ['dog', 'nature']
Detected OS : Windows


====== WebDriver manager ======
Detected OS : Windows


====== WebDriver manager ======
Current google-chrome version is 96.0.4664
Get LATEST driver version for 96.0.4664
Current google-chrome version is 96.0.4664
Get LATEST driver version for 96.0.4664
Get LATEST driver version for 96.0.4664
Get LATEST driver version for 96.0.4664
Trying to download new driver from https://chromedriver.storage.googleapis.com/96.0.4664.45/chromedriver_win32.zip
Trying to download new driver from https://chromedriver.storage.googleapis.com/96.0.4664.45/chromedriver_win32.zip
Driver has been saved in cache [C:\Users\mmouk\.wdm\drivers\chromedriver\win32\96.0.4664.45]

DevTools listening on ws://127.0.0.1:52082/devtools/browser/0a7ecdd9-cfb7-4db3-ab02-c0d8148a63aa
[1210/012945.464:ERROR:command_buffer_proxy_impl.cc(125)] ContextResult::kTransientFailure: Failed to send GpuControl.CreateCommandBuffer.
_________________________________
Current web-browser version:    96.0.4664.93
Current chrome-driver version:  96.0.4664.45
_________________________________
Collecting links... dog from google
[Full Resolution Mode]
```

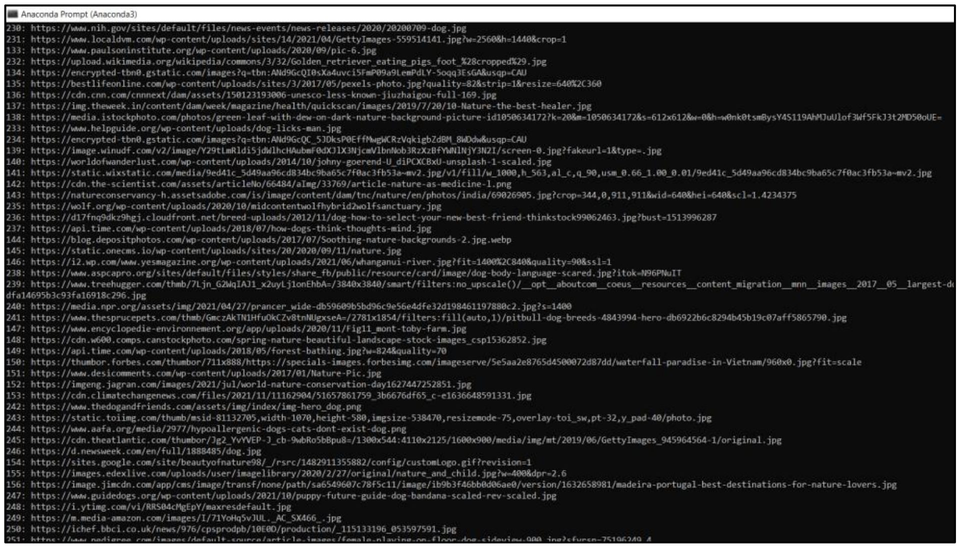
Fig. 1: Anaconda command prompt displaying the start of fullr esolution image scraping and downloading

The above image shows the console output for a run of the application where we have given the settings of limit = 10 which implies that 10 images will be downloaded from the scraped images. Next setting that we have given is threads = 8 which means that the number of threads used for scraping and downloading purposes is equal to 8. The next setting that has been set is full = true. This setting denotes that we are going to download the original resolution of the images from google images by downloading it straight from the original site once the source link of the image has been scraped. Now the application will go the scraped link apply the same methodology to download the original image in full resolution. This process requires time as well as necessitates that the target website allows for scraping to take place on its site and it does not block scraping requests by Selenium and Chrome Driver. The keywords that are being searched in this scenario are dog and nature. Hence 10 full resolution images of dogs and nature are downloaded.



```
12: https://th-thumbnailer.cdn-si-edu.com/vSnitgUqCQCRSx7mkHZtHZHry4U=/1072x720/filters:no_upscale()/https://tf-cmsv2-smithsonianmag-media.s3.amazonaws.com/filer/04/8e/048ed839-
mages-168346757_web.jpg
13: https://images.news18.com/ibnlive/uploads/2021/07/1627377451_nature-1600x900.jpg
27: https://yt3.ggpht.com/ytc/AKedOLRvxGYSdEHqu0X4EYcJ2kq7BttRKBNpfwdHJf3F5g=s900-c-k-c0x00ffffff-no-rj
28: https://cdn.vox-cdn.com/thumbor/CqsS8CZJVT1wcu_OCdGxcdPIPgg=/1400x1050/filters:format(jpeg)/cdn.vox-cdn.com/uploads/chorus_asset/file/21768855/pop_culter_dog_rank_alyceatinc
14: https://www.iucn.org/sites/dev/files/content/images/2020/shutterstock_1458128810.jpg
15: https://assets.unenvironment.org/styles/article_billboard_image/s3/2021-05/alberta-2297204_1920.jpg?h=1483c54f&amp;itok=GdjA9GRu
16: https://cdn.the-scientist.com/assets/articleNo/66864/aImg/35078/foresttb-l.jpg
17: https://cdn.pixabay.com/photo/2015/12/01/20/28/road-1072821__340.jpg
18: https://thumbor.forbes.com/thumbor/960x0/https%3A%2F%2Fspecials-images.forbesimg.com%2Fimageserve%2F5f9cca07d4c42920d4d348c7%2Frainforest%2F960x0.jpg%3Ffit%3Dscale
19: https://oecdenvironmentfocusblog.files.wordpress.com/2020/06/wed-blog-shutterstock_1703194387_low_nwm.jpg
20: https://images.assettype.com/fortuneindia%2F2020-06%2Fef53f9be-f257-4aa3-9af5-6ca1a9f33a86%2Fclose_up_photography_of_leaves_with_droplets_807598.jpg?rect=0,607,4128,2322&w=1
29: https://www.guidedogs.org/wp-content/uploads/2021/11/01.11.2021_SGD1014-Edit-scaled.jpg
30: https://www.rover.com/blog/wp-content/uploads/2017/02/pug-2648774_1920.jpg
31: https://i.natgeofe.com/n/4f5aaece-3300-41a4-b2a8-ed2708a0a27c/domestic-dog_thumb_2x1.jpg
32: https://www.gannett-cdn.com/media/2021/06/03/USATODAY/usatsports/imageForEntry18-8on.jpg?width=2560
33: https://static01.nyt.com/images/2021/11/16/well/11Well-NL-DOG-SLEEP/11Well-NL-DOG-SLEEP-videoSixteenByNineJumbo1600.jpg
34: https://images.theconversation.com/files/319375/original/file-20200309-118956-1cqvm6j.jpg?ixlib=rb-1.1.0&q=45&auto=format&w=1200&h=900.0&fit=crop
35: https://cdn.britannica.com/q:60/49/161649-050-3F458ECF/Bernese-mountain-dog-grass.jpg
21: https://www.lombardodier.com/files/live/sites/loportail/files/news/2021/May/20210521/Nature_LOcom.jpg
22: https://images.unsplash.com/photo-1608159477202-8a0e27f807b4?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxzZWFyY2h8MXx8YmVhdXRpZnVsJTIwbmF0dXJlfGVufDB8fDB8fA%3D%3D&w=1000&q=80
23: https://akm-img-a-in.tosshub.com/indiatoday/images/story/201510/thumb_647_102915112526.jpg?size=770:433
36: https://images.news18.com/ibnlive/uploads/2021/08/1628056310_dogdrinking-1200x800.png?impolicy=website&width=510&height=356
24: https://images.sadhguru.org/sites/default/files/media_files/iso/en/64083-natures-temples.jpg
25: https://images.theconversation.com/files/399816/original/file-20210510-5797-xqoxsr.jpg?ixlib=rb-1.1.0&q=45&auto=format&w=1200&h=900.0&fit=crop
37: https://www.meltontimes.co.uk/webimg/b25lY21zOjQzYjk1YzdmLThkYTAtNDQ1ZS04Y2VhLTkyNDk0MDMyNjJkODo3OTVjYjY0Ni05OTY5LTRhMGUtYjE2MS03ZjJmMmY1YWNmYjg=.jpg?width=640&enable=upscal
38: https://akm-img-a-in.tosshub.com/indiatoday/images/story/202108/international_dog_day_2021_4_r_1200x768.jpeg?mhENil.rEsB2Wju30UDroUYKmJ4NfkX4&size=770:433
39: https://imagesvc.meredithcorp.io/v3/mm/image?url=https%3A%2F%2Fstatic.onecms.io%2Fwp-content%2Fuploads%2Fsites%2F47%2F2021%2F03%2F25%2Fdoberman-pinscher-red-collar-110081212
26: https://media.wired.com/photos/60abf2c7f83409ce52d3c67f/master/pass/Science_forest_1167053895.jpg
27: https://upload.wikimedia.org/wikipedia/commons/3/36/Hopetoun_falls.jpg
28: https://pi.tedcdn.com/r/talkstar-assets.s3.amazonaws.com/production/playlists/playlist_398/reconnect_with_nature.jpg?
29: https://m.economictimes.com/thumb/msid-68721417,width-1200,height-900,resizemode-4,imgsize-1016106/nature1_gettyimages.jpg
30: https://upload.wikimedia.org/wikipedia/commons/thumb/1/1a/24701-nature-natural-beauty.jpg/1280px-24701-nature-natural-beauty.jpg
31: https://images.indianexpress.com/2017/04/nature-tree_759.jpg
32: https://images.unsplash.com/photo-1589312133722-eab40c686ebb?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxzZWFyY2h8MXx8Z3JlZW4lMjBuYXR1cmV8ZW58MHx8MHx8&w=1000&q=80
33: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQTAYhFkNrmRfN1VFDiPc-H-12FEm5MRs2raw&usqp=CAU
34: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT3ZdjJabW77Ve_0WoCa5COzt93sZf7tdK_Zg&usqp=CAU
40: https://arc-anglerfish-washpost-prod-washpost.s3.amazonaws.com/public/HB4AT3D3IMI6TMPTWIZ74WAR54.jpg
41: https://media-cldnry.s-nbcnews.com/image/upload/newscms/2021_18/3469748/210430-frankie-al-1227_0.jpg
35: https://images.newscientist.com/wp-content/uploads/2021/04/15171535/ebjbwn_web.jpg
42: https://www.gannett-cdn.com/-mm-/675ba115506e09d3d7e32d07055fdcb503ebe995/c=0-2-1272-718/local/-/media/2020/05/28/USATODAY/usatsports/Reviewed.com-RvEW-24856-lead.jpg?width=
g&auto=webp
43: https://www.cdc.gov/healthypets/images/pets/cute-dog-headshot.jpg?_=42445
44: https://www.forbes.com/advisor/wp-content/uploads/2021/03/pit-bull-featured.jpg
45: https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/smartest-dog-breeds-1553287693.jpg?crop=0.673xw:1.00xh;0.167xw,0&resize=640:*
36: https://www.wbcsd.org/var/site/storage/images/overview/news-insights/general/news/wbcsd-to-support-the-task-force-on-nature-related-financial-disclosures-tnfd/150706-1-eng-G
```

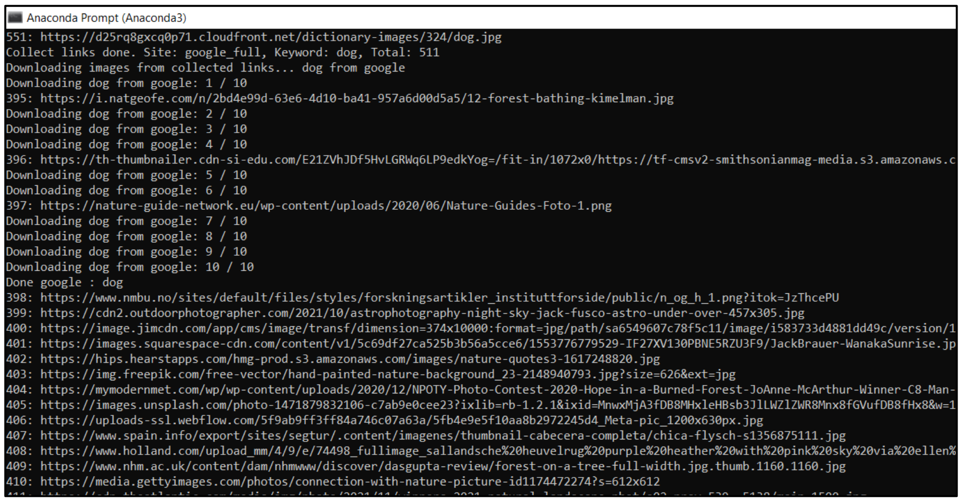
Fig. 2: Links of dog and nature being scraped



```
230: https://www.nih.gov/sites/default/files/news-events/news-releases/2020/20200709-dog.jpg
231: https://www.localdvm.com/wp-content/uploads/sites/14/2021/04/GettyImages-559514141.jpg?w=2560&h=1440&crop=1
133: https://www.paulsoninstitute.org/wp-content/uploads/2020/09/pic-6.jpg
232: https://upload.wikimedia.org/wikipedia/commons/3/32/Golden_retriever_eating_pigs_foot_%28cropped%29.jpg
134: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQI0sXa4uvci5FmP09a9LemPdLY-5oqq3EsGA&usqp=CAU
135: https://bestlifeonline.com/wp-content/uploads/sites/3/2017/05/pexels-photo.jpg?quality=82&strip=1&resize=640%2C360
136: https://cdn.cnn.com/cnnnext/dam/assets/150123193006-unesco-less-known-jiuzhaigou-full-169.jpg
137: https://img.theweek.in/content/dam/week/magazine/health/quickscan/images/2019/7/20/10-Nature-the-best-healer.jpg
138: https://media.istockphoto.com/photos/green-leaf-with-dew-on-dark-nature-background-picture-id1050634172?k=20&m=1050634172&s=612x612&w=0&h=w0nk0tsmBysY4S119AhMJuUlof3Wf5FkJ3t2MD50oUE=
233: https://www.helpguide.org/wp-content/uploads/dog-licks-man.jpg
234: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQC_5JDksP0EffMwgWCRzVqkigbZdBM_8WDdw&usqp=CAU
139: https://image.winudf.com/v2/image/Y29tLmRldi5jdWlhcHAubmF0dXJlX3NjcmVlbnNob3RzXzBfYWNlNjY3N2I/screen-0.jpg?fakeurl=1&type=.jpg
140: https://worldofwanderlust.com/wp-content/uploads/2014/10/johny-goerend-U_diPCXCBxU-unsplash-1-scaled.jpg
141: https://static.wixstatic.com/media/9ed41c_5d49aa96cd834bc9ba65c7f0ac3fb53a~mv2.jpg/v1/fill/w_1000,h_563,al_c,q_90,usm_0.66_1.00_0.01/9ed41c_5d49aa96cd834bc9ba65c7f0ac3fb53a~mv2.jpg
142: https://cdn.the-scientist.com/assets/articleNo/66484/aImg/33769/article-nature-as-medicine-l.png
143: https://natureconservancy-h.assetsadobe.com/is/image/content/dam/tnc/nature/en/photos/india/69026905.jpg?crop=344,0,911,911&wid=640&hei=640&scl=1.4234375
235: https://wolf.org/wp-content/uploads/2020/10/midcontentwolfhybrid2wolfsanctuary.jpg
236: https://d17fnq9dkz9hgj.cloudfront.net/breed-uploads/2012/11/dog-how-to-select-your-new-best-friend-thinkstock99062463.jpg?bust=1513996287
237: https://api.time.com/wp-content/uploads/2018/07/how-dogs-think-thoughts-mind.jpg
144: https://blog.depositphotos.com/wp-content/uploads/2017/07/Soothing-nature-backgrounds-2.jpg.webp
145: https://static.onecms.io/wp-content/uploads/sites/20/2020/09/11/nature.jpg
146: https://i2.wp.com/www.yesmagazine.org/wp-content/uploads/2021/06/whanganui-river.jpg?fit=1400%2C840&quality=90&ssl=1
238: https://www.aspcapro.org/sites/default/files/styles/share_fb/public/resource/card/image/dog-body-language-scared.jpg?itok=N96PNuIT
239: https://www.treehugger.com/thmb/7Ljn_G2WqIAJ1_x2uyLj1onEhbA=/3840x3840/smart/filters:no_upscale()/__opt__aboutcom__coeus__resources__content_migration__mnn__images__2017__05__largest-d
dfa14695b3c93fa16918c296.jpg
240: https://media.npr.org/assets/img/2021/04/27/prancer_wide-db59609b5bd96c9e56e4dfe32d198461197880c2.jpg?s=1400
241: https://www.thesprucepets.com/thmb/GmczAkTN1HfuOkCZv8tnNUgxseA=/2781x1854/filters:fill(auto,1)/pitbull-dog-breeds-4843994-hero-db6922b6c8294b45b19c07aff5865790.jpg
147: https://www.encyclopedie-environnement.org/app/uploads/2020/11/Fig11_mont-toby-farm.jpg
148: https://cdn.w600.comps.canstockphoto.com/spring-nature-beautiful-landscape-stock-images_csp15362852.jpg
149: https://api.time.com/wp-content/uploads/2018/05/forest-bathing.jpg?w=824&quality=70
150: https://thumbor.forbes.com/thumbor/711x888/https://specials-images.forbesimg.com/imageserve/5e5aa2e8765d4500072d87dd/waterfall-paradise-in-Vietnam/960x0.jpg?fit=scale
151: https://www.desicomments.com/wp-content/uploads/2017/01/Nature-Pic.jpg
152: https://imgeng.jagran.com/images/2021/jul/world-nature-conservation-day1627447252851.jpg
153: https://cdn.climatechangenews.com/files/2021/11/11162904/51657861759_3b6676df65_c-e1636648591331.jpg
242: https://www.thedogandfriends.com/assets/img/index/img-hero_dog.png
243: https://static.toiimg.com/thumb/msid-81132705,width-1070,height-580,imgsize-538470,resizemode-75,overlay-toi_sw,pt-32,y_pad-40/photo.jpg
244: https://www.aafa.org/media/2977/hypoallergenic-dogs-cats-dont-exist-dog.png
245: https://cdn.theatlantic.com/thumbor/Jg2_YvYVEP-J_cb-9wbRo5bBpu8=/1300x544:4110x2125/1600x900/media/img/mt/2019/06/GettyImages_945964564-1/original.jpg
246: https://d.newsweek.com/en/full/1888485/dog.jpg
154: https://sites.google.com/site/beautyofnature98/_/rsrc/1482911355882/config/customLogo.gif?revision=1
155: https://images.edexlive.com/uploads/user/imagelibrary/2020/2/27/original/nature_and_child.jpg?w=400&dpr=2.6
156: https://image.jimcdn.com/app/cms/image/transf/none/path/sa6549607c78f5c11/image/ib9b3f46bb0d06ae0/version/1632658981/madeira-portugal-best-destinations-for-nature-lovers.jpg
247: https://www.guidedogs.org/wp-content/uploads/2021/10/puppy-future-guide-dog-bandana-scaled-rev-scaled.jpg
248: https://i.ytimg.com/vi/RRS04cMgEpY/maxresdefault.jpg
249: https://m.media-amazon.com/images/I/71YoHq5vJUL._AC_SX466_.jpg
250: https://ichef.bbci.co.uk/news/976/cpsprodpb/10E0D/production/_115133196_053597591.jpg
251: https://www.pedigree.com/images/default-source/article-images/female-playing-on-floor-dog-sideview-900.jpg?sfvrsn=75196249_4
```

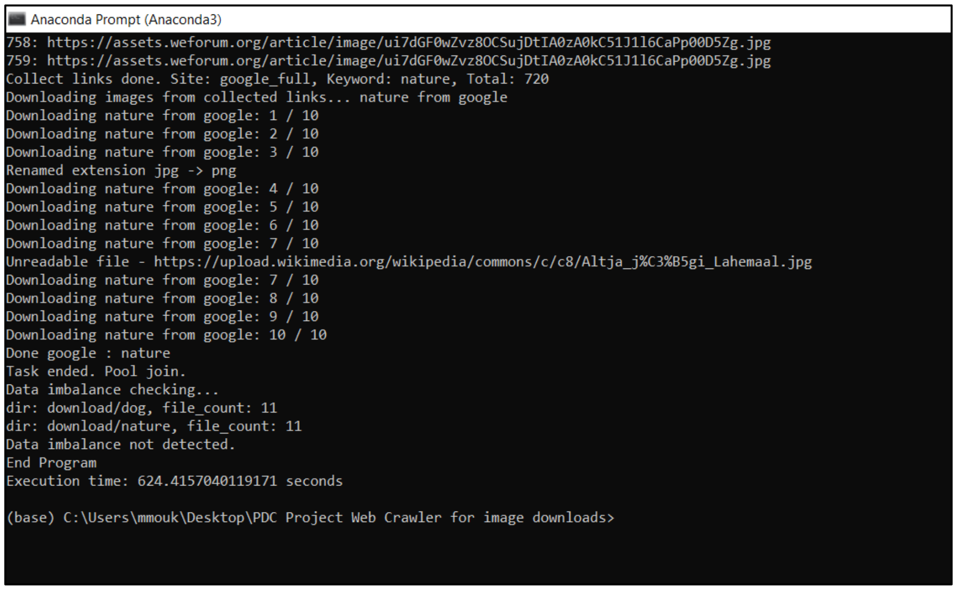
Fig. 3: Links of dog and nature being scraped continued

The above two images show the links of the images that are being scraped by our web scraper for the keywords of dog and nature. Since this is a full resolution image downloads, the scraper needs to go to the source link of the image page and download that particular image.



```
551: https://d25rq8gxcq0p71.cloudfront.net/dictionary-images/324/dog.jpg
Collect links done. Site: google_full, Keyword: dog, Total: 511
Downloading images from collected links... dog from google
Downloading dog from google: 1 / 10
395: https://i.natgeofe.com/n/2bd4e99d-63e6-4d10-ba41-957a6d00d5a5/12-forest-bathing-kimelman.jpg
Downloading dog from google: 2 / 10
Downloading dog from google: 3 / 10
Downloading dog from google: 4 / 10
396: https://th-thumbnailer.cdn-si-edu.com/E21ZVhJDf5HvLGRWq6LP9edkYog=/fit-in/1072x0/https://tf-cmsv2-smithsonianmag-media.s3.amazonaws.c
Downloading dog from google: 5 / 10
Downloading dog from google: 6 / 10
397: https://nature-guide-network.eu/wp-content/uploads/2020/06/Nature-Guides-Foto-1.png
Downloading dog from google: 7 / 10
Downloading dog from google: 8 / 10
Downloading dog from google: 9 / 10
Downloading dog from google: 10 / 10
Done google : dog
398: https://www.nmbu.no/sites/default/files/styles/forskningsartikler_instituttforside/public/n_og_h_1.png?itok=JzThcePU
399: https://cdn2.outdoorphotographer.com/2021/10/astrophotography-night-sky-jack-fusco-astro-under-over-457x305.jpg
400: https://image.jimcdn.com/app/cms/image/transf/dimension=374x10000:format=jpg/path/sa6549607c78f5c11/image/i583733d4881dd49c/version/1
401: https://images.squarespace-cdn.com/content/v1/5c69df27ca525b3b56a5cce6/1553776779529-IF27XV130PBNE5RZU3F9/JackBrauer-WanakaSunrise.jp
402: https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/nature-quotes3-1617248820.jpg
403: https://img.freepik.com/free-vector/hand-painted-nature-background_23-2148940793.jpg?size=626&ext=jpg
404: https://mymodernmet.com/wp/wp-content/uploads/2020/12/NPOTY-Photo-Contest-2020-Hope-in-a-Burned-Forest-JoAnne-McArthur-Winner-C8-Man-
405: https://images.unsplash.com/photo-1471879832106-c7ab9e0cee23?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxleHBsb3JlLWZlZWR8Mnx8fGVufDB8fHx8&w=1
406: https://uploads-ssl.webflow.com/5f9ab9ff3ff84a746c07a63a/5fb4e9e5f10aa8b2972245d4_Meta-pic_1200x630px.jpg
407: https://www.spain.info/export/sites/segtur/.content/imagenes/thumbnail-cabecera-completa/chica-flysch-s1356875111.jpg
408: https://www.holland.com/upload_mm/4/9/e/74498_fullimage_sallandsche%20heuvelrug%20purple%20heather%20with%20pink%20sky%20via%20ellen%
409: https://www.nhm.ac.uk/content/dam/nhmwww/discover/dasgupta-review/forest-on-a-tree-full-width.jpg.thumb.1160.1160.jpg
410: https://media.gettyimages.com/photos/connection-with-nature-picture-id1174472274?s=612x612
```

Fig. 4: End of full resolution download of dog



```
758: https://assets.weforum.org/article/image/ui7dGF0wZvz8OCSujDtIA0zA0kC51J1l6CaPp00D5Zg.jpg
759: https://assets.weforum.org/article/image/ui7dGF0wZvz8OCSujDtIA0zA0kC51J1l6CaPp00D5Zg.jpg
Collect links done. Site: google_full, Keyword: nature, Total: 720
Downloading images from collected links... nature from google
Downloading nature from google: 1 / 10
Downloading nature from google: 2 / 10
Downloading nature from google: 3 / 10
Renamed extension jpg -> png
Downloading nature from google: 4 / 10
Downloading nature from google: 5 / 10
Downloading nature from google: 6 / 10
Downloading nature from google: 7 / 10
Unreadable file - https://upload.wikimedia.org/wikipedia/commons/c/c8/Altja_j%C3%B5gi_Lahemaal.jpg
Downloading nature from google: 7 / 10
Downloading nature from google: 8 / 10
Downloading nature from google: 9 / 10
Downloading nature from google: 10 / 10
Done google : nature
Task ended. Pool join.
Data imbalance checking...
dir: download/dog, file_count: 11
dir: download/nature, file_count: 11
Data imbalance not detected.
End Program
Execution time: 624.4157040119171 seconds

(base) C:\Users\mmouk\Desktop\PDC Project Web Crawler for image downloads>
```

Fig. 5: End of full resolution download of nature and download verification and execution time

From the above two images we can see that we have collected 551 links of the keyword dog out of which there are 511 unique images or links. Once all the links of the keyword dog have been scraped, 10 full resolution images are then downloaded from the scraped links as a limit of 10 has been set while running the application. The scraping of keyword nature continues in parallel. The second image shows that 759 links of keyword nature have been scraped and out of these, we get 720 unique links. Following the scraping, 10 full resolution images of nature keyword are downloaded. Following the completion of the scraping and downloading of the images the Pool is closed which means closing of the parallel portion of the program. Then we check if the correct number of images have been downloaded for all keywords which in this case is 10 and we display the necessary message. There is always one more file than the limit in the download folder as one is the response file which is created when the download is done. We can also see in the output that the execution time for full resolution downloads with 8 threads comes out to 624.415 seconds. Full resolution images take a long time to execute even in parallel because the crawler has to go to each individual site and get the full resolution image element and apply the discussed methods and download a full resolution image file which is larger than thumbnail images.

5.2. Full Resolution Downloaded Images in Folders

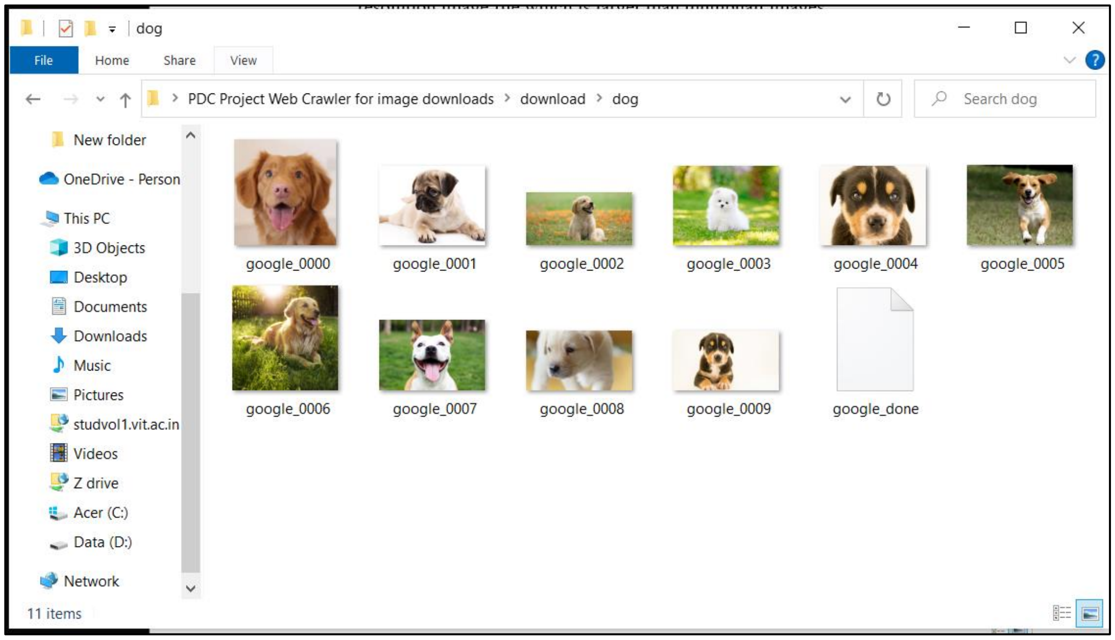


Fig. 6: Full resolution images of dogs

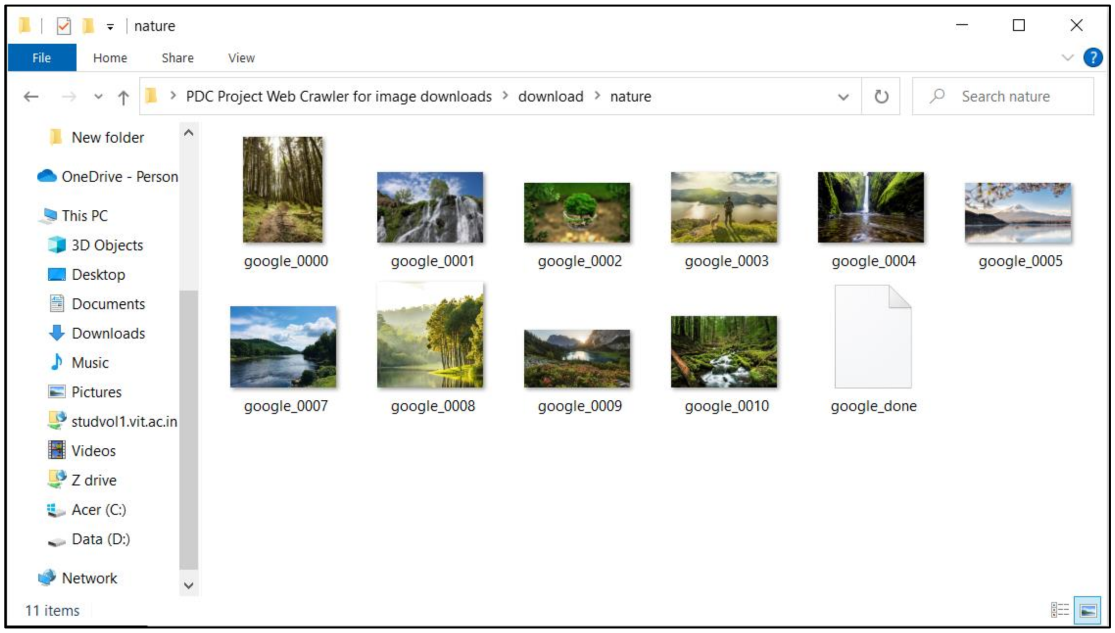
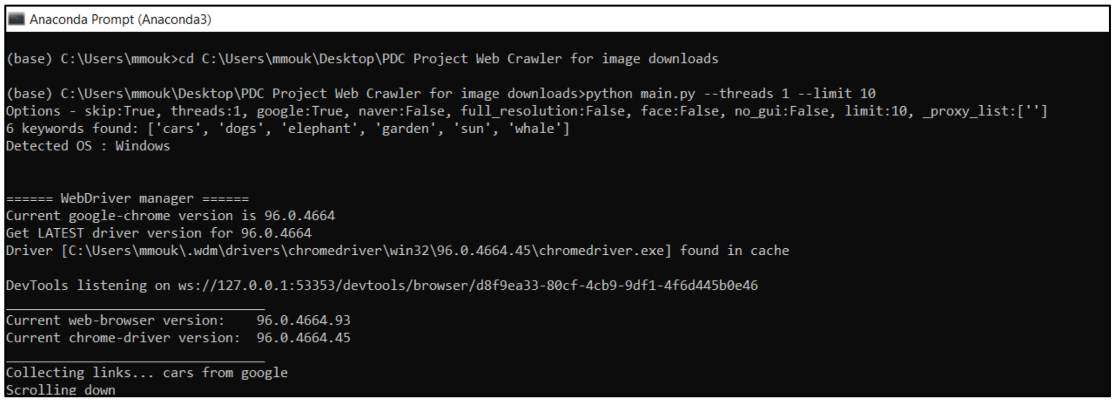


Fig. 7: Full resolution images of nature

The above two images show the downloaded full resolution images of dog and nature in their respective folders. These images are in their original resolution as found on the source website. The limit was set as 10 in the beginning of the run, and we can observe that 10 images have been downloaded for both dog and nature in full resolution mode. The correct number of images have also been downloaded and there is no imbalance in the data.

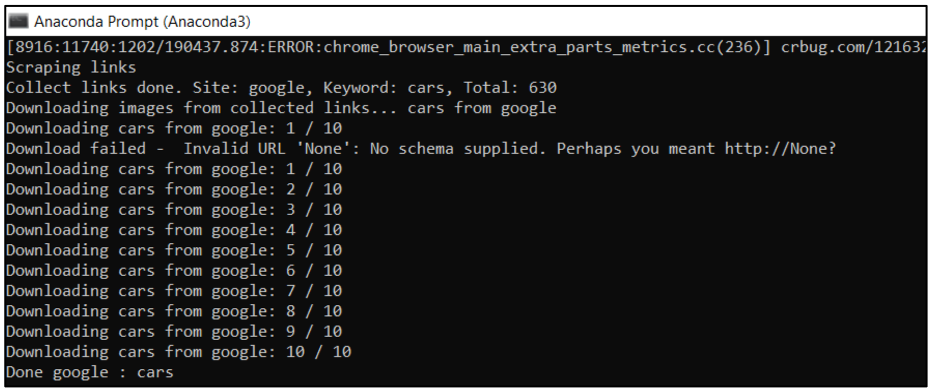
5.3. Thumbnail Image Downloads and Execution Time Comparison with Respect to 1, 2, 4 and 8 Threads for 10 Image Downloads

5.3.1) 1 Thread and Limit 10



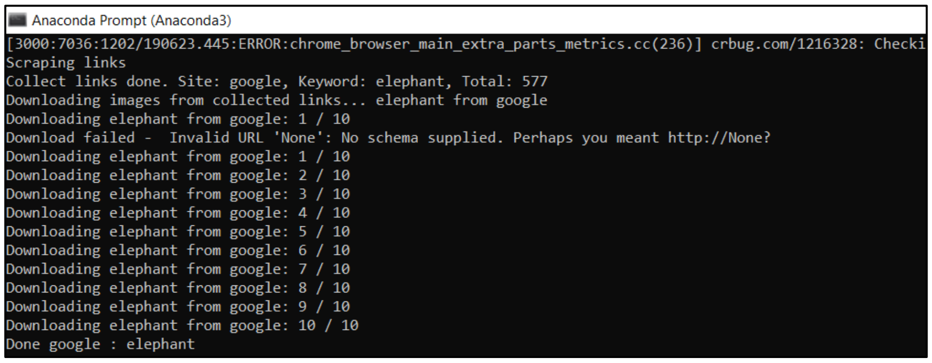
```
Anaconda Prompt (Anaconda3)

(base) C:\Users\mmouk>cd C:\Users\mmouk\Desktop\PDC Project Web Crawler for image downloads

(base) C:\Users\mmouk\Desktop\PDC Project Web Crawler for image downloads>python main.py --threads 1 --limit 10
Options - skip:True, threads:1, google:True, naver:False, full_resolution:False, face:False, no_gui:False, limit:10, _proxy_list:['']
6 keywords found: ['cars', 'dogs', 'elephant', 'garden', 'sun', 'whale']
Detected OS : Windows


====== WebDriver manager ======
Current google-chrome version is 96.0.4664
Get LATEST driver version for 96.0.4664
Driver [C:\Users\mmouk\.wdm\drivers\chromedriver\win32\96.0.4664.45\chromedriver.exe] found in cache

DevTools listening on ws://127.0.0.1:53353/devtools/browser/d8f9ea33-80cf-4cb9-9df1-4f6d445b0e46
_________________________________
Current web-browser version:    96.0.4664.93
Current chrome-driver version:  96.0.4664.45
_________________________________
Collecting links... cars from google
Scrolling down
```

Fig. 8: Thumbnail image scraping and downloading with 1 thread



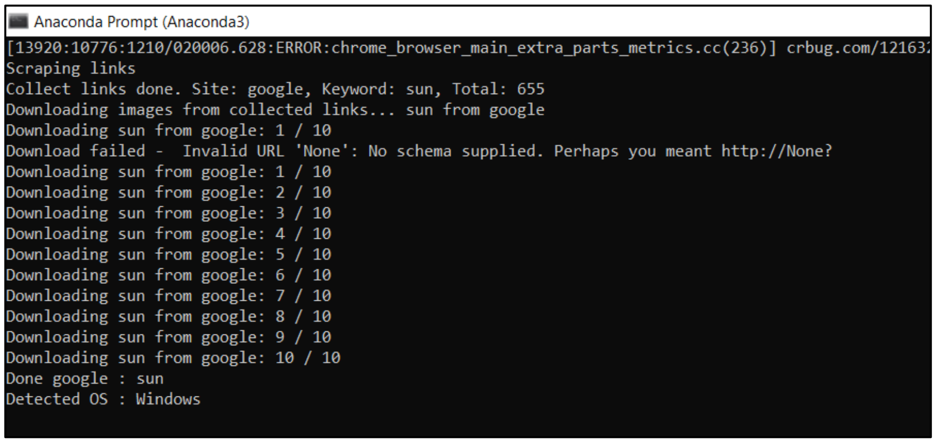
```
[8916:11740:1202/190437.874:ERROR:chrome_browser_main_extra_parts_metrics.cc(236)] crbug.com/121632
Scraping links
Collect links done. Site: google, Keyword: cars, Total: 630
Downloading images from collected links... cars from google
Downloading cars from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading cars from google: 1 / 10
Downloading cars from google: 2 / 10
Downloading cars from google: 3 / 10
Downloading cars from google: 4 / 10
Downloading cars from google: 5 / 10
Downloading cars from google: 6 / 10
Downloading cars from google: 7 / 10
Downloading cars from google: 8 / 10
Downloading cars from google: 9 / 10
Downloading cars from google: 10 / 10
Done google : cars
```

Fig. 9: End of cars images download



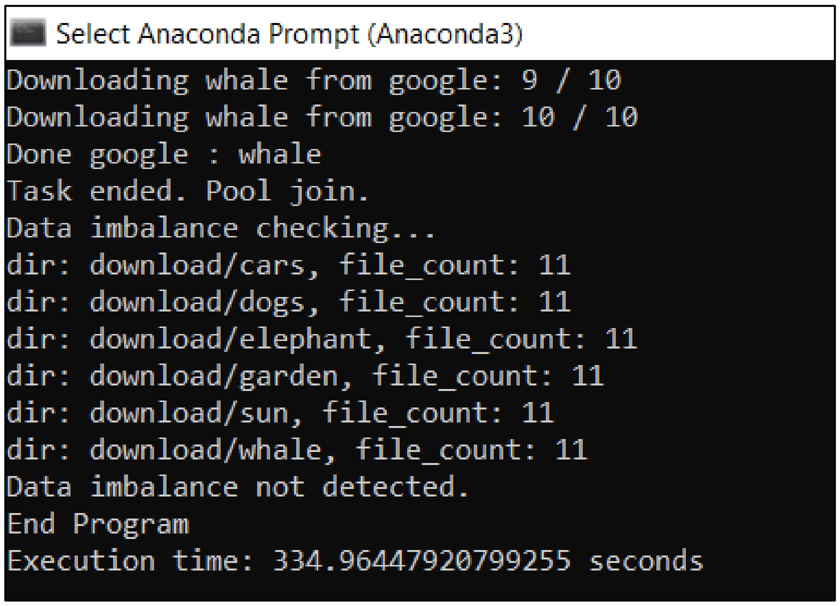
```
[3000:7036:1202/190623.445:ERROR:chrome_browser_main_extra_parts_metrics.cc(236)] crbug.com/1216328: Checki
Scraping links
Collect links done. Site: google, Keyword: elephant, Total: 577
Downloading images from collected links... elephant from google
Downloading elephant from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading elephant from google: 1 / 10
Downloading elephant from google: 2 / 10
Downloading elephant from google: 3 / 10
Downloading elephant from google: 4 / 10
Downloading elephant from google: 5 / 10
Downloading elephant from google: 6 / 10
Downloading elephant from google: 7 / 10
Downloading elephant from google: 8 / 10
Downloading elephant from google: 9 / 10
Downloading elephant from google: 10 / 10
Done google : elephant
```

Fig. 10: End of elephant images download



```
[13920:10776:1210/020006.628:ERROR:chrome_browser_main_extra_parts_metrics.cc(236)] crbug.com/121632
Scraping links
Collect links done. Site: google, Keyword: sun, Total: 655
Downloading images from collected links... sun from google
Downloading sun from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading sun from google: 1 / 10
Downloading sun from google: 2 / 10
Downloading sun from google: 3 / 10
Downloading sun from google: 4 / 10
Downloading sun from google: 5 / 10
Downloading sun from google: 6 / 10
Downloading sun from google: 7 / 10
Downloading sun from google: 8 / 10
Downloading sun from google: 9 / 10
Downloading sun from google: 10 / 10
Done google : sun
Detected OS : Windows
```

Fig. 11: End of sun images download

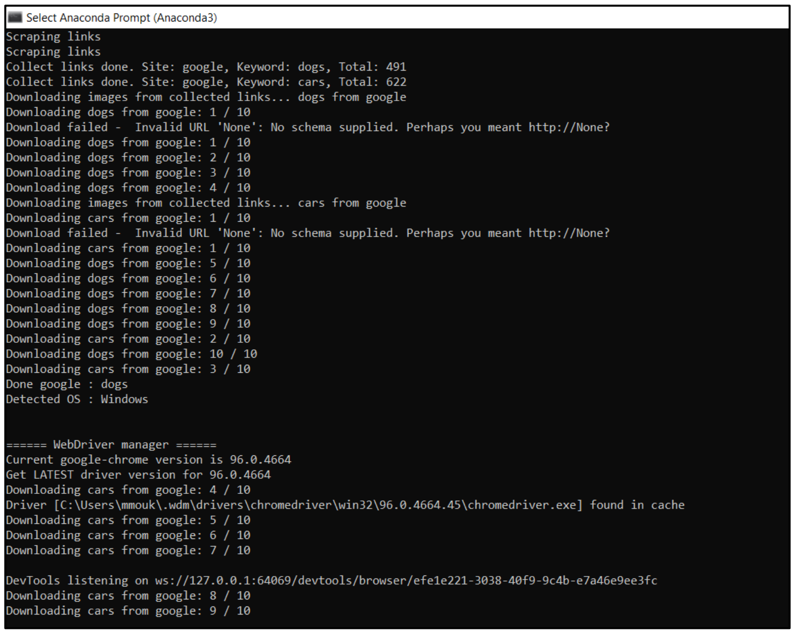


```
Select Anaconda Prompt (Anaconda3)
Downloading whale from google: 9 / 10
Downloading whale from google: 10 / 10
Done google : whale
Task ended. Pool join.
Data imbalance checking...
dir: download/cars, file_count: 11
dir: download/dogs, file_count: 11
dir: download/elephant, file_count: 11
dir: download/garden, file_count: 11
dir: download/sun, file_count: 11
dir: download/whale, file_count: 11
Data imbalance not detected.
End Program
Execution time: 334.96447920799255 seconds
```

Fig. 12: End of downloads and verification

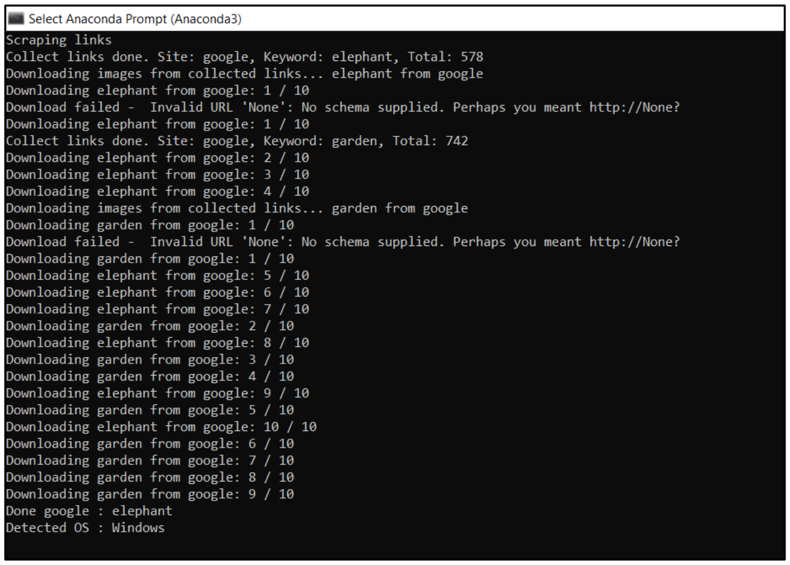
The first run involved setting the threads number to 1 and the limit of the download to 10. This set of downloads was conducted for thumbnails images which can be scraped directly from google images in small resolution instead of their original resolution. The single threaded program executed each download of the 6 provided keywords namely cars, dogs, elephants, garden, sun and nature separately. On the completion of the download on 1 thread, we can observe that the execution time comes out to be 334.9644 seconds.

5.3.2) 2 Threads and Limit 10



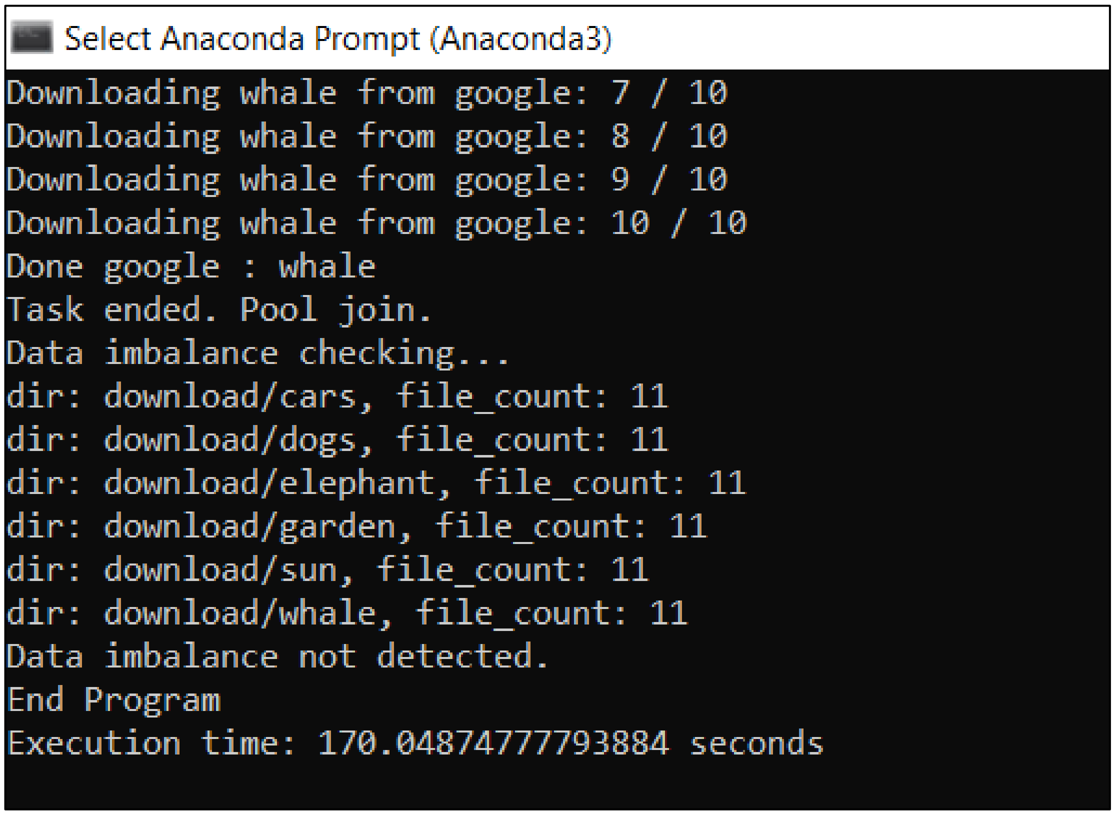
```
Select Anaconda Prompt (Anaconda3)
Scraping links
Scraping links
Collect links done. Site: google, Keyword: dogs, Total: 491
Collect links done. Site: google, Keyword: cars, Total: 622
Downloading images from collected links... dogs from google
Downloading dogs from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading dogs from google: 1 / 10
Downloading dogs from google: 2 / 10
Downloading dogs from google: 3 / 10
Downloading dogs from google: 4 / 10
Downloading images from collected links... cars from google
Downloading cars from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading cars from google: 1 / 10
Downloading dogs from google: 5 / 10
Downloading dogs from google: 6 / 10
Downloading dogs from google: 7 / 10
Downloading dogs from google: 8 / 10
Downloading dogs from google: 9 / 10
Downloading cars from google: 2 / 10
Downloading dogs from google: 10 / 10
Downloading cars from google: 3 / 10
Done google : dogs
Detected OS : Windows


====== WebDriver manager ======
Current google-chrome version is 96.0.4664
Get LATEST driver version for 96.0.4664
Downloading cars from google: 4 / 10
Driver [C:\Users\mmouk\.wdm\drivers\chromedriver\win32\96.0.4664.45\chromedriver.exe] found in cache
Downloading cars from google: 5 / 10
Downloading cars from google: 6 / 10
Downloading cars from google: 7 / 10

DevTools listening on ws://127.0.0.1:64069/devtools/browser/efe1e221-3038-40f9-9c4b-e7a46e9ee3fc
Downloading cars from google: 8 / 10
Downloading cars from google: 9 / 10
```

Fig. 13: Parallel downloads of dogs and cats



```
Select Anaconda Prompt (Anaconda3)
Scraping links
Collect links done. Site: google, Keyword: elephant, Total: 578
Downloading images from collected links... elephant from google
Downloading elephant from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading elephant from google: 1 / 10
Collect links done. Site: google, Keyword: garden, Total: 742
Downloading elephant from google: 2 / 10
Downloading elephant from google: 3 / 10
Downloading elephant from google: 4 / 10
Downloading images from collected links... garden from google
Downloading garden from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading garden from google: 1 / 10
Downloading elephant from google: 5 / 10
Downloading elephant from google: 6 / 10
Downloading elephant from google: 7 / 10
Downloading garden from google: 2 / 10
Downloading elephant from google: 8 / 10
Downloading garden from google: 3 / 10
Downloading garden from google: 4 / 10
Downloading elephant from google: 9 / 10
Downloading garden from google: 5 / 10
Downloading elephant from google: 10 / 10
Downloading garden from google: 6 / 10
Downloading garden from google: 7 / 10
Downloading garden from google: 8 / 10
Downloading garden from google: 9 / 10
Done google : elephant
Detected OS : Windows
```

Fig. 14: Parallel downloads of elephant and garden

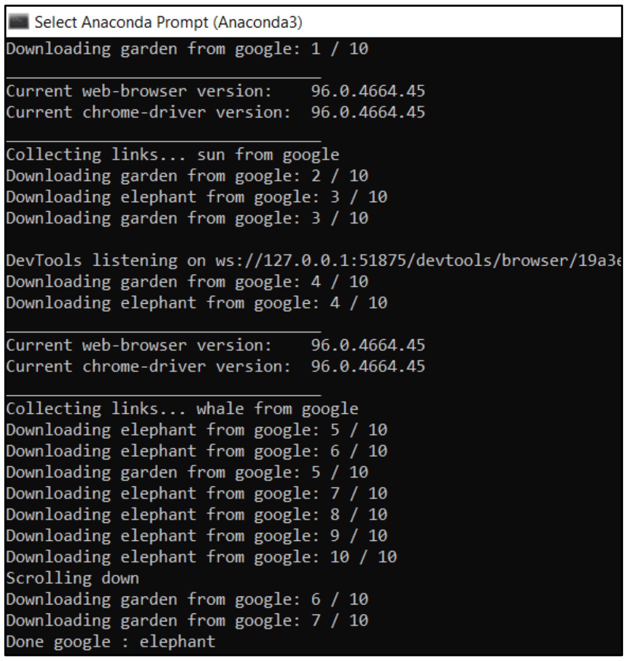


```
Downloading whale from google: 7 / 10
Downloading whale from google: 8 / 10
Downloading whale from google: 9 / 10
Downloading whale from google: 10 / 10
Done google : whale
Task ended. Pool join.
Data imbalance checking...
dir: download/cars, file_count: 11
dir: download/dogs, file_count: 11
dir: download/elephant, file_count: 11
dir: download/garden, file_count: 11
dir: download/sun, file_count: 11
dir: download/whale, file_count: 11
Data imbalance not detected.
End Program
Execution time: 170.04874777793884 seconds
```

Fig. 15: Execution Time with Two Threads

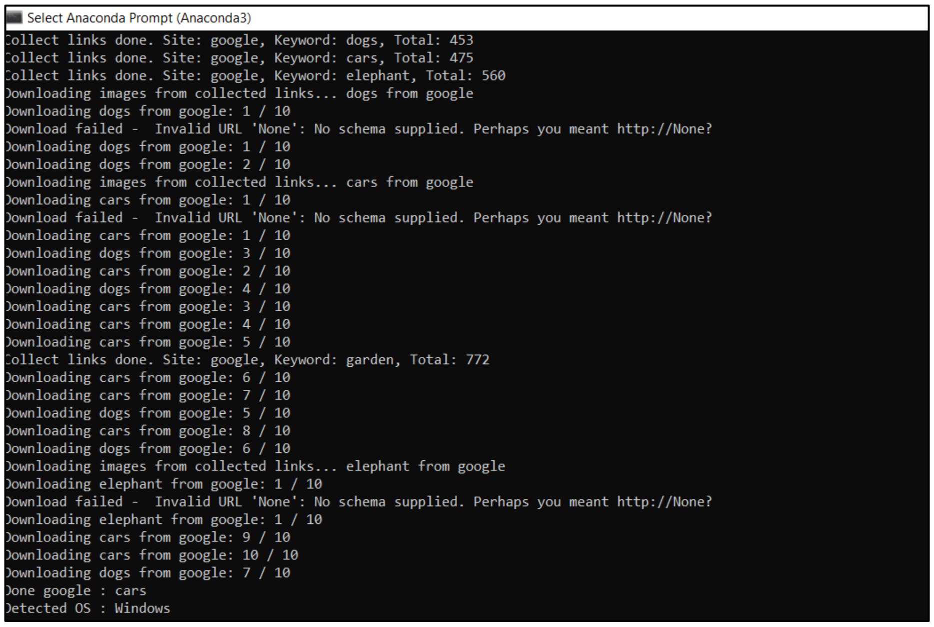
The above execution was conducted for 2 threads and a limit of 10 image downloads was set again for consistency in execution time comparison. It was observed that on setting thread number as two, two different keywords were scraped at the same time and downloaded in parallel as well. There was a sharp reduction in execution time as well. Execution time for two threads came out to be 170.0487 seconds.

5.3.3) 4 Threads and Limit 10



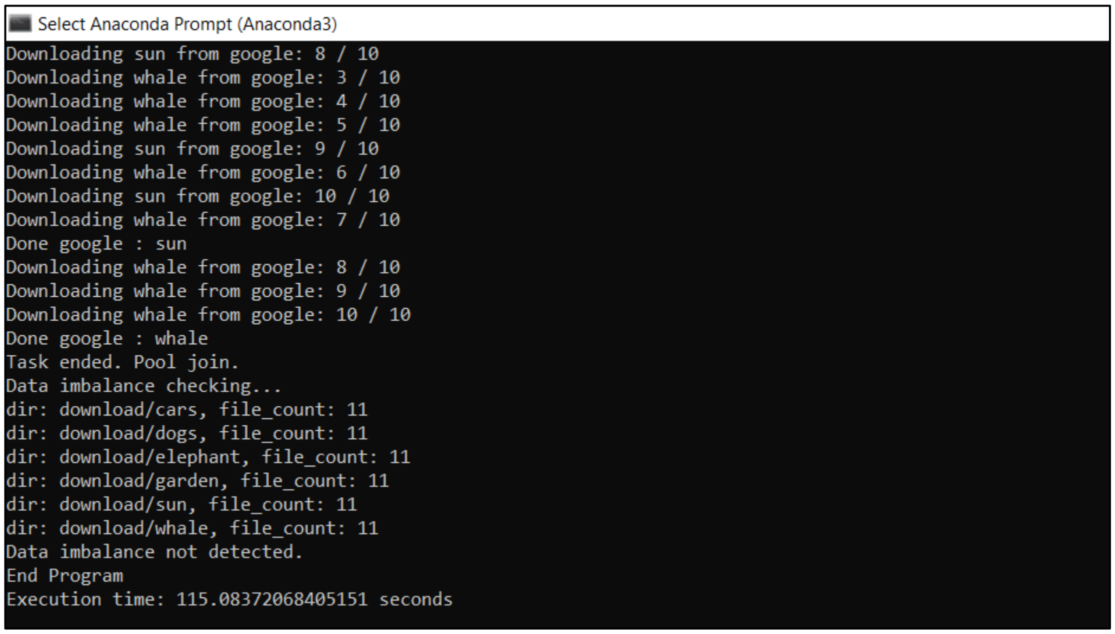
```
Select Anaconda Prompt (Anaconda3)
Downloading garden from google: 1 / 10
_________________________________
Current web-browser version:    96.0.4664.45
Current chrome-driver version:  96.0.4664.45
_________________________________
Collecting links... sun from google
Downloading garden from google: 2 / 10
Downloading elephant from google: 3 / 10
Downloading garden from google: 3 / 10

DevTools listening on ws://127.0.0.1:51875/devtools/browser/19a3e
Downloading garden from google: 4 / 10
Downloading elephant from google: 4 / 10
_________________________________
Current web-browser version:    96.0.4664.45
Current chrome-driver version:  96.0.4664.45
_________________________________
Collecting links... whale from google
Downloading elephant from google: 5 / 10
Downloading elephant from google: 6 / 10
Downloading garden from google: 5 / 10
Downloading elephant from google: 7 / 10
Downloading elephant from google: 8 / 10
Downloading elephant from google: 9 / 10
Downloading elephant from google: 10 / 10
Scrolling down
Downloading garden from google: 6 / 10
Downloading garden from google: 7 / 10
Done google : elephant
```

Fig. 16: Parallel download on 4 threads



```
Select Anaconda Prompt (Anaconda3)
Collect links done. Site: google, Keyword: dogs, Total: 453
Collect links done. Site: google, Keyword: cars, Total: 475
Collect links done. Site: google, Keyword: elephant, Total: 560
Downloading images from collected links... dogs from google
Downloading dogs from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading dogs from google: 1 / 10
Downloading dogs from google: 2 / 10
Downloading images from collected links... cars from google
Downloading cars from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading cars from google: 1 / 10
Downloading dogs from google: 3 / 10
Downloading cars from google: 2 / 10
Downloading dogs from google: 4 / 10
Downloading cars from google: 3 / 10
Downloading cars from google: 4 / 10
Downloading cars from google: 5 / 10
Collect links done. Site: google, Keyword: garden, Total: 772
Downloading cars from google: 6 / 10
Downloading cars from google: 7 / 10
Downloading dogs from google: 5 / 10
Downloading cars from google: 8 / 10
Downloading dogs from google: 6 / 10
Downloading images from collected links... elephant from google
Downloading elephant from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading elephant from google: 1 / 10
Downloading cars from google: 9 / 10
Downloading cars from google: 10 / 10
Downloading dogs from google: 7 / 10
Done google : cars
Detected OS : Windows
```

Fig. 17: Parallel download on 4 threads continued

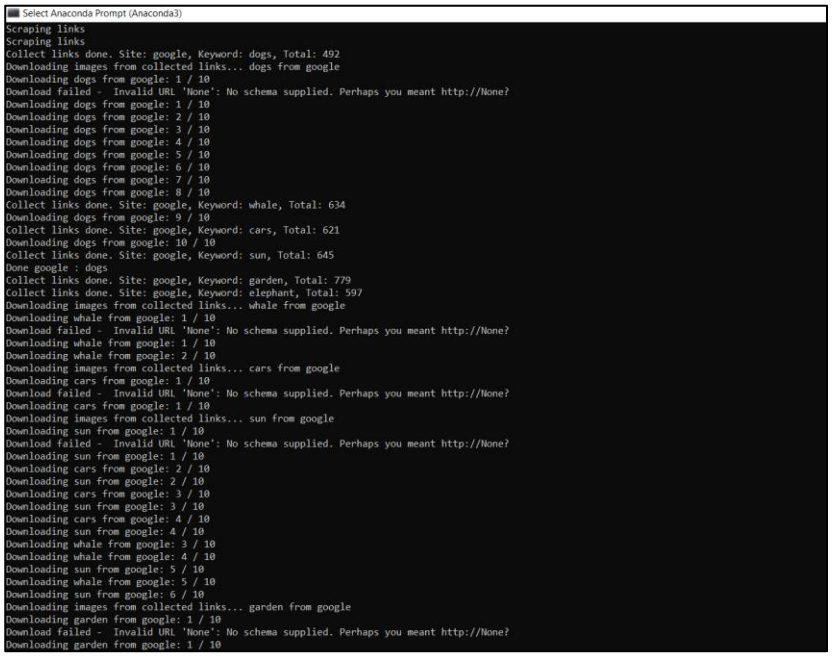


```
Downloading sun from google: 8 / 10
Downloading whale from google: 3 / 10
Downloading whale from google: 4 / 10
Downloading whale from google: 5 / 10
Downloading sun from google: 9 / 10
Downloading whale from google: 6 / 10
Downloading sun from google: 10 / 10
Downloading whale from google: 7 / 10
Done google : sun
Downloading whale from google: 8 / 10
Downloading whale from google: 9 / 10
Downloading whale from google: 10 / 10
Done google : whale
Task ended. Pool join.
Data imbalance checking...
dir: download/cars, file_count: 11
dir: download/dogs, file_count: 11
dir: download/elephant, file_count: 11
dir: download/garden, file_count: 11
dir: download/sun, file_count: 11
dir: download/whale, file_count: 11
Data imbalance not detected.
End Program
Execution time: 115.08372068405151 seconds
```

Fig. 18: Execution Time on 4 Threads

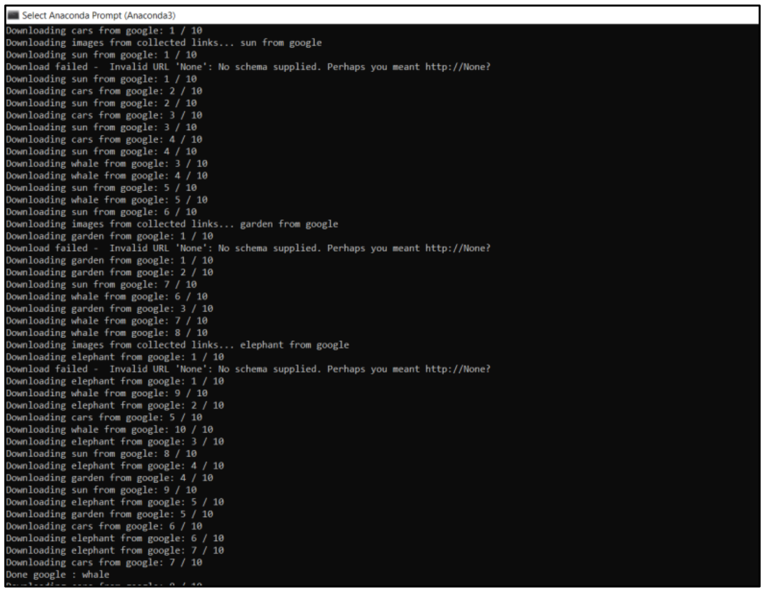
The above execution was conducted for 4 threads and a limit of 10 image downloads was set again for consistency in execution time comparison. It was observed that on setting thread number as four, four different keywords were scraped at the same time and downloaded in parallel as well. More reduction in execution time was observed in comparison to single and two thread execution times. Execution time for four threads came out to be 115.0837 seconds.

5.3.4) 8 Threads and Limit 10



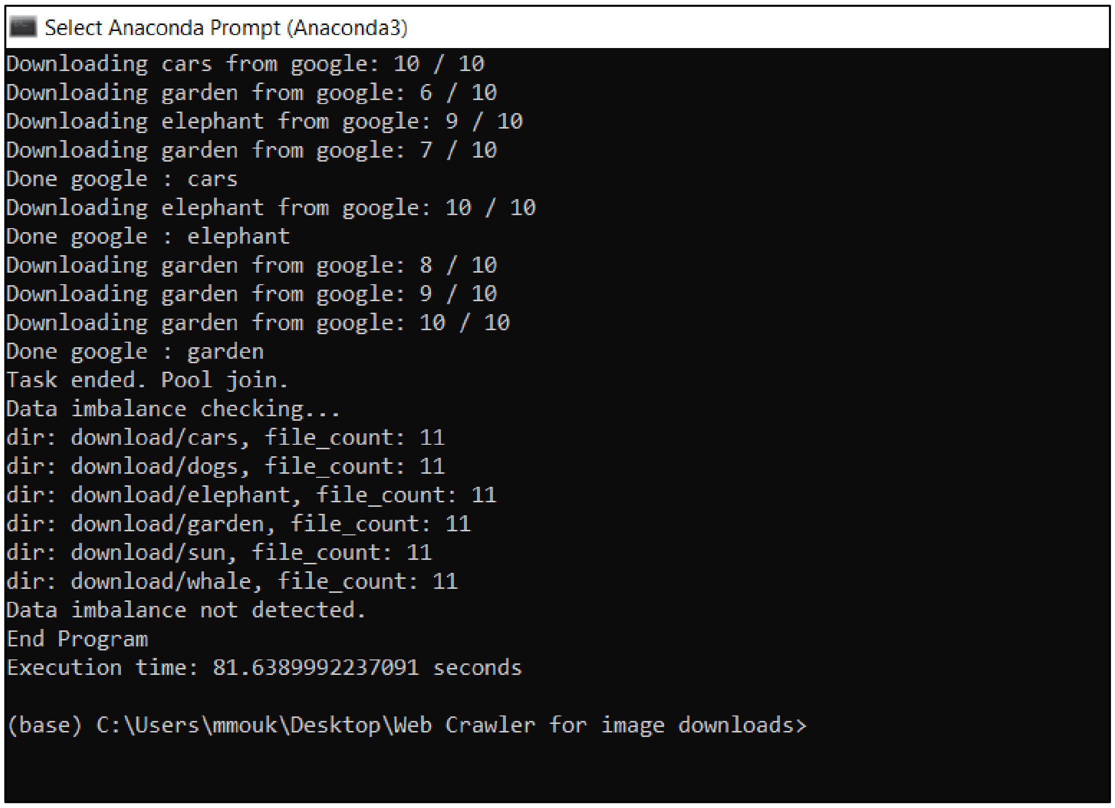
```
Select Anaconda Prompt (Anaconda3)
Scraping links
Scraping links
Collect links done. Site: google, Keyword: dogs, Total: 492
Downloading images from collected links... dogs from google
Downloading dogs from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading dogs from google: 1 / 10
Downloading dogs from google: 2 / 10
Downloading dogs from google: 3 / 10
Downloading dogs from google: 4 / 10
Downloading dogs from google: 5 / 10
Downloading dogs from google: 6 / 10
Downloading dogs from google: 7 / 10
Downloading dogs from google: 8 / 10
Collect links done. Site: google, Keyword: whale, Total: 634
Downloading dogs from google: 9 / 10
Collect links done. Site: google, Keyword: cars, Total: 621
Downloading dogs from google: 10 / 10
Collect links done. Site: google, Keyword: sun, Total: 645
Done google : dogs
Collect links done. Site: google, Keyword: garden, Total: 779
Collect links done. Site: google, Keyword: elephant, Total: 597
Downloading images from collected links... whale from google
Downloading whale from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading whale from google: 1 / 10
Downloading whale from google: 2 / 10
Downloading images from collected links... cars from google
Downloading cars from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading cars from google: 1 / 10
Downloading images from collected links... sun from google
Downloading sun from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading sun from google: 1 / 10
Downloading cars from google: 2 / 10
Downloading sun from google: 2 / 10
Downloading cars from google: 3 / 10
Downloading sun from google: 3 / 10
Downloading cars from google: 4 / 10
Downloading sun from google: 4 / 10
Downloading whale from google: 3 / 10
Downloading whale from google: 4 / 10
Downloading sun from google: 5 / 10
Downloading whale from google: 5 / 10
Downloading sun from google: 6 / 10
Downloading images from collected links... garden from google
Downloading garden from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading garden from google: 1 / 10
```

Fig. 19: Parallel Execution on 8 threads



```
Select Anaconda Prompt (Anaconda3)
Downloading cars from google: 1 / 10
Downloading images from collected links... sun from google
Downloading sun from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading sun from google: 1 / 10
Downloading cars from google: 2 / 10
Downloading sun from google: 2 / 10
Downloading cars from google: 3 / 10
Downloading sun from google: 3 / 10
Downloading cars from google: 4 / 10
Downloading sun from google: 4 / 10
Downloading whale from google: 3 / 10
Downloading whale from google: 4 / 10
Downloading sun from google: 5 / 10
Downloading whale from google: 5 / 10
Downloading sun from google: 6 / 10
Downloading images from collected links... garden from google
Downloading garden from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading garden from google: 1 / 10
Downloading garden from google: 2 / 10
Downloading sun from google: 7 / 10
Downloading whale from google: 6 / 10
Downloading garden from google: 3 / 10
Downloading whale from google: 7 / 10
Downloading whale from google: 8 / 10
Downloading images from collected links... elephant from google
Downloading elephant from google: 1 / 10
Download failed -  Invalid URL 'None': No schema supplied. Perhaps you meant http://None?
Downloading elephant from google: 1 / 10
Downloading whale from google: 9 / 10
Downloading elephant from google: 2 / 10
Downloading cars from google: 5 / 10
Downloading whale from google: 10 / 10
Downloading elephant from google: 3 / 10
Downloading sun from google: 8 / 10
Downloading elephant from google: 4 / 10
Downloading garden from google: 4 / 10
Downloading sun from google: 9 / 10
Downloading elephant from google: 5 / 10
Downloading garden from google: 5 / 10
Downloading cars from google: 6 / 10
Downloading elephant from google: 6 / 10
Downloading elephant from google: 7 / 10
Downloading cars from google: 7 / 10
Done google : whale
```

Fig. 20: Parallel Execution and Download on 8 threads



Select Anaconda Prompt (Anaconda3)

```
Downloading cars from google: 10 / 10
Downloading garden from google: 6 / 10
Downloading elephant from google: 9 / 10
Downloading garden from google: 7 / 10
Done google : cars
Downloading elephant from google: 10 / 10
Done google : elephant
Downloading garden from google: 8 / 10
Downloading garden from google: 9 / 10
Downloading garden from google: 10 / 10
Done google : garden
Task ended. Pool join.
Data imbalance checking...
dir: download/cars, file_count: 11
dir: download/dogs, file_count: 11
dir: download/elephant, file_count: 11
dir: download/garden, file_count: 11
dir: download/sun, file_count: 11
dir: download/whale, file_count: 11
Data imbalance not detected.
End Program
Execution time: 81.6389992237091 seconds

(base) C:\Users\mmouk\Desktop\Web Crawler for image downloads>
```

Fig. 21: Execution Time on 8 Threads

Finally, the last execution conducted was for 8 threads and a limit of 10 image downloads was set again for consistency in execution time comparison. It was observed that on setting thread number as eight, all 6 keywords were scraped at the same time and downloaded in parallel as well. More reduction in execution time was observed in comparison to single and two and four thread execution times. Execution time for eight threads came out to be 81.6389 seconds.

5.3.5) Comparison of Execution Times

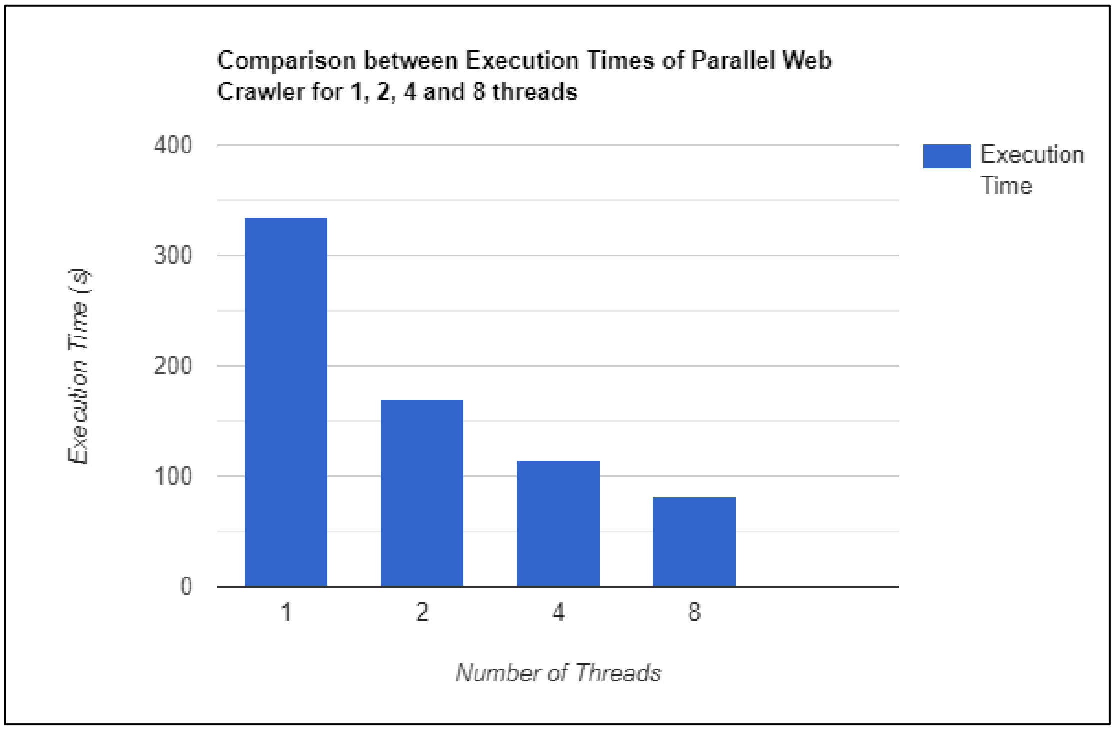


Fig. 22: Comparison of Execution Times between 1,2,4 and 8 Threads

A graphical representation of the reduction in execution times with increasing number of threads.

From the results above, we can see that for the 6 keywords provided, the execution time of the parallel web crawler keeps on decreasing as the number of threads increases. This implies that we have successfully parallelized the web crawler.

The impact of the number of threads on performance will be more relevant when the number of keywords will be extremely large (for example 100). Greater number of threads will be able to execute the program more effectively than lesser number of threads.

5.4. GUI of Web Crawler

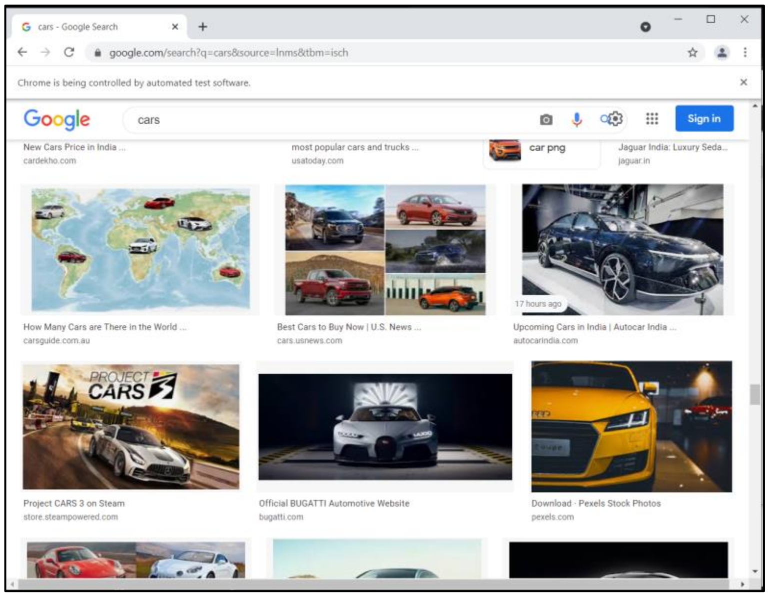

cars - Google Search
google.com/search?q=cars&source=lnms&tbm=isch
Chrome is being controlled by automated test software.
Google
cars
Sign in
New Cars Price in India ...
cardekho.com
most popular cars and trucks ...
usatoday.com
car png
Jaguar India: Luxury Seda...
jaguar.in
How Many Cars are There in the World ...
carsguide.com.au
Best Cars to Buy Now | U.S. News ...
cars.usnews.com
17 hours ago
Upcoming Cars in India | Autocar India ...
autocarindia.com
Project CARS 3 on Steam
store.steampowered.com
Official BUGATTI Automotive Website
bugatti.com
Download · Pexels Stock Photos
pexels.com

Fig. 23: GUI of car images being scraped

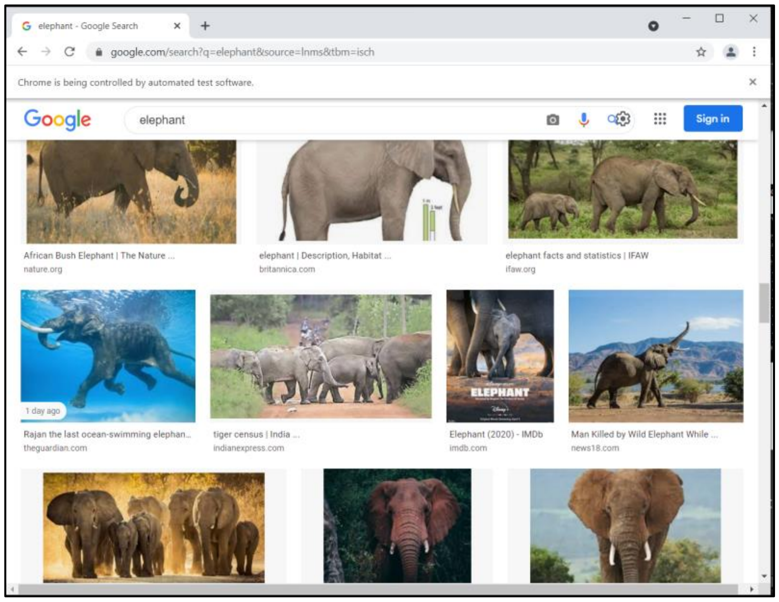

elephant - Google Search
google.com/search?q=elephant&source=lnms&tbm=isch
Chrome is being controlled by automated test software.
Google
elephant
Sign in
African Bush Elephant | The Nature ...
nature.org
elephant | Description, Habitat ...
britannica.com
elephant facts and statistics | IFAW
ifaw.org
1 day ago
Rajan the last ocean-swimming elephan...
theguardian.com
tiger census | India ...
indianexpress.com
Elephant (2020) - IMDb
imdb.com
Man Killed by Wild Elephant While ...
news18.com

Fig. 24: GUI of elephant images being scraped

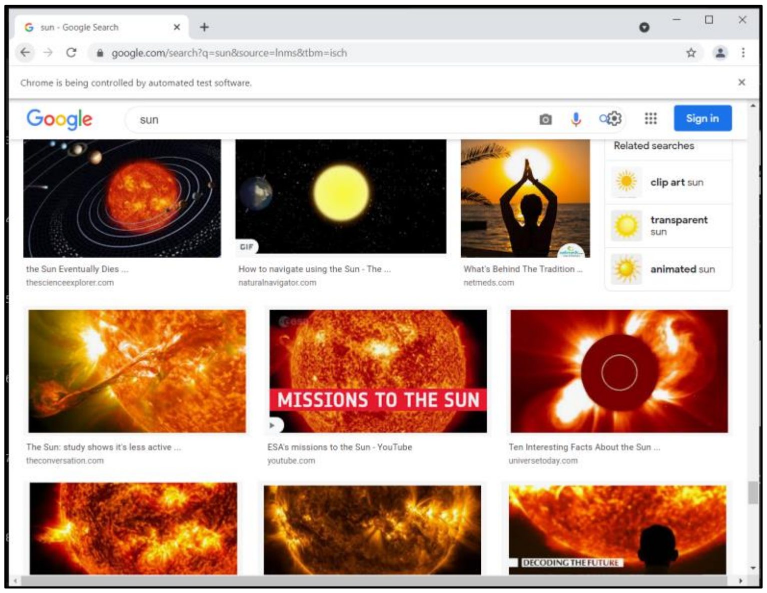


Fig. 25: GUI of sun images being scraped

The above figures show the GUI of the parallel web crawler when it scrapes the thumbnail images on google images based on the given keywords.

5.5. Downloaded Thumbnail Images

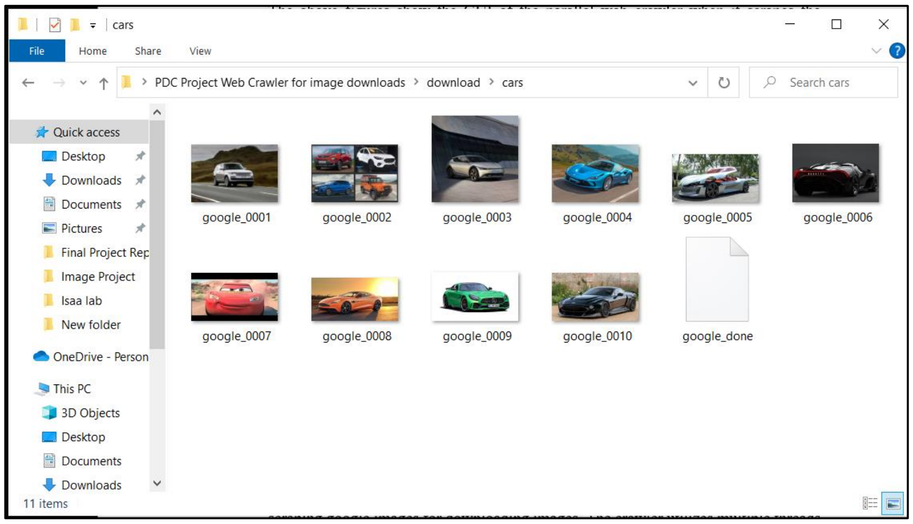


Fig. 26: Downloaded Car Images

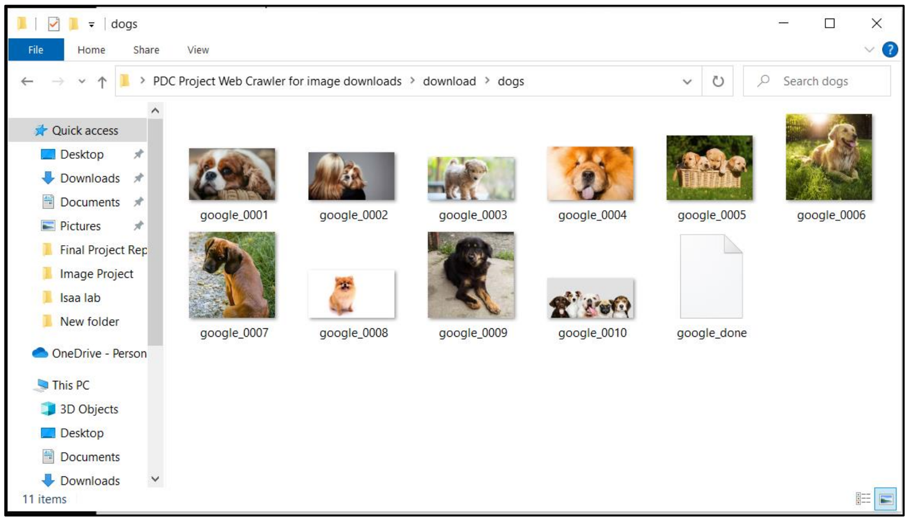


Fig. 27: Downloaded Dog Images

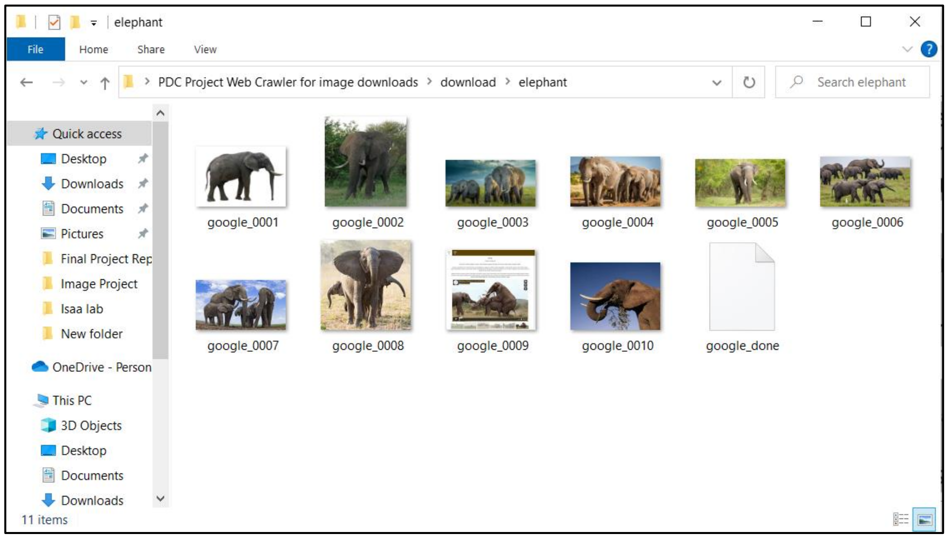


Fig. 28: Downloaded Elephant Images

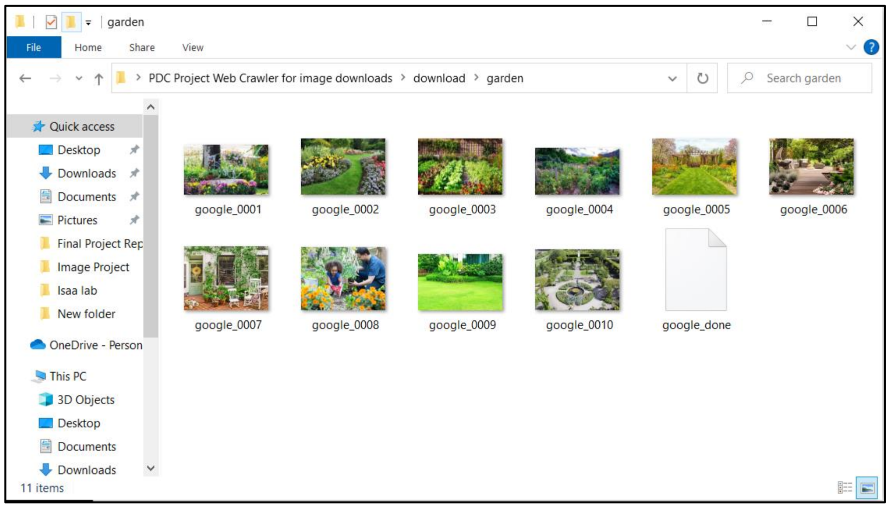


Fig. 29: Downloaded Garden Images

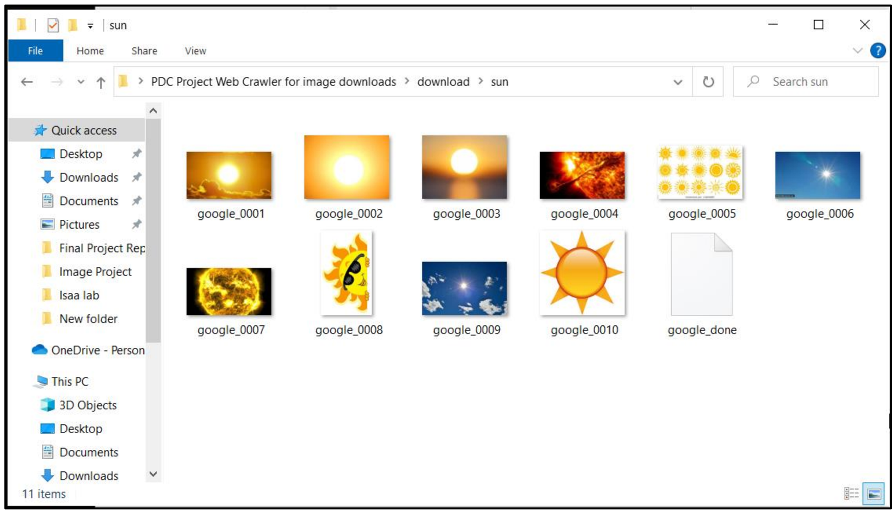


Fig. 30: Downloaded Sun Images

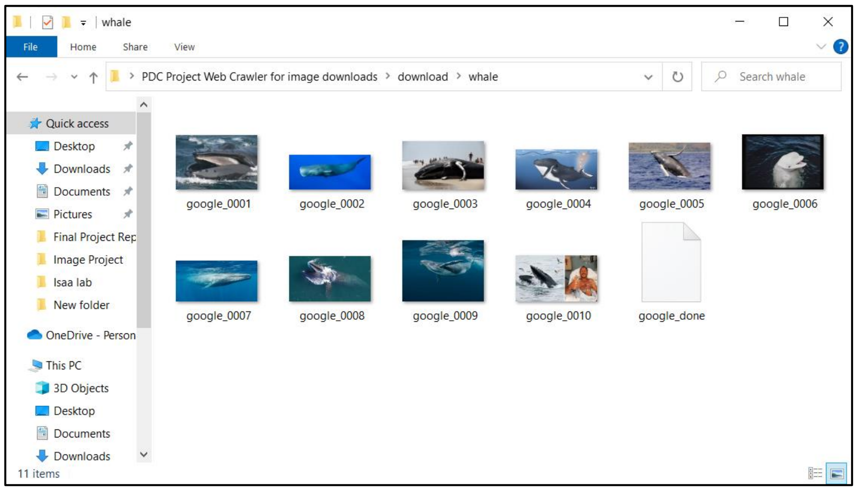


Fig. 31: Downloaded Whale Images

The above figures show that all the images of the 6 provided keywords namely cars, dogs, elephant, garden, sun and whale have been successfully downloaded. Every folder has 11 files where 10 are the image files and 1 is the response file which is created when 10 images have been successfully downloaded.

6. CONCLUSION

We have successfully created a parallel web crawler which automates the process of scraping google images for downloading images. The crawler utilizes multiple threads which can be set by the user during runtime, and it assign tasks to the threads automatically by means of the multiprocessing library in python. The main functionality of the web crawler is to scrape google images for the thumbnail images as well as provide a method to scrape the source of the image so that the full resolution image can be downloaded of the specified keyword. We have also successfully used Selenium and Chrome Driver to automate the process of scraping parallelly. The main parallel processes of our project have also been successfully implemented which are namely the scraper and the downloader both of which execute properly on multiple threads. The application also ensures that no duplicate links which are scraped, are being downloaded multiple times to ensure the uniqueness of the downloaded images. A functionality to verify that all the images have been successfully downloaded is also in place to ensure that the limit of image downloads specified by the user is met. A comparison between the execution times of 1,2 ,4 and 8 threads has been conducted and plotted in the form of a bar graph. The execution time is seen to decrease with an increase in the number of threads thus demonstrating the impact of parallelization on performance. In conclusion, we have successfully managed to achieve the objectives that we set out for while creating the application and it successfully scrapes google images and downloads images using multiple threads as planned.

7. REFERENCES

[1] P. S., “an Image Crawler for Content Based Image Retrieval System,” Int. J. Res. Eng. Technol., vol. 02, no. 11, pp. 33–37, 2013, doi: 10.15623/ijret.2013.0211006.

[2] D. Yadav, A. K. Sharma, and J. P. Gupta, “Parallel crawler architecture and web page change detection,” WSEAS Trans. Comput., vol. 7, no. 7, pp. 929–940, 2008.

[3] S. Khalil and M. Fakir, “RCrawler: An R package for parallel web crawling and scraping,” SoftwareX, vol. 6, pp. 98–106, 2017, doi: 10.1016/j.softx.2017.04.004.

[4] S. Deshmukh and K. Vishwakarma, “A survey on crawlers used in developing search engine,” Proc. - 5th Int. Conf. Intell. Comput. Control Syst. ICICCS 2021, no. Iciccs, pp. 1446–1452, 2021, doi: 10.1109/ICICCS51141.2021.9432368.