Web Crawler Optimization

Norlan Prudente
University of the Pacific
3601 Pacific Ave
Stockton, CA 95211
n_prudente@u.pacific.edu

1 Abstract

Programs now a days have a lot of issues. Some programs are slow, other programs have problems, and some have both. Optimization is one of a few techniques that can help resolve the issues of a program being slow and not working properly. The project, web crawler, was developed by a few students and the program is currently being optimized for better performance. The work that are currently in progress includes solving the problem of DDOS defense mechanism being triggered, implementing GPU parallelization using OpenCL, and cleaning some source code for better performance. The DDOS problem was previously solved by temporarily putting the program to sleep, but that makes the program slow. The sleeping solution will be replaced with ip shuffler using proxies. Parallelization using GPU can make the program faster through task parallelization because tasks will be distributed to all the available cores. Source code cleanup will focus on eliminating unnecessary code and revising some code to be more specific in searches. With the usage of different optimization techniques, speed and efficiency will improve.

2 Keywords

Python, parallel computing, DDOS, object oriented, interpreter, compiler, dynamic semantics: typing and binding, multithreading, multiprocessing, CPU, GPU, and web crawler.

3 Introduction

One of the many goals of a Software Developer is to push the software's capabilities and reach its limit with our current, year 2018, hardware. [1] Speed is crucial in software applications because users lose interest in slow programs. As an example, the load speed of a web page will determine how many users visit a web site and, surprisingly, for every 1-second that a web page get delayed, 5.8% of the users abandon it and try another website. [2] Users abandoned slow websites because there are other websites that can provide similar service or information. Rather than spending time waiting, users can be more productive by going to other websites.

Optimization of software, applications, and programs, benefits Software Developers and those in related fields of study. By studying the current state of software optimization techniques, researchers can start developing new, powerful, and better optimization techniques. As a result, we, Computer Scientists and people in related fields of study, can eliminate one of the many problems software development is facing, which is that software is advancing slower than hardware. Additional benefits of developing improved optimization techniques are improvement on algorithms. Old algorithm can be modified or extended during research projects.

Users, developers, and researchers benefit from software optimization. For users, their time can be used in more productive ways than waiting for software. For developers and researchers, new discoveries and advancement in technology will provide us new ways to create, develop, or improve programs. These are the reasons why optimization is a good start for improvement of software programs.

This research paper will focus on web crawler's optimization through parallel programming and solving the defense mechanism of DDOS being triggered by large a number of requests in an unreasonable time frame. The next part of the paper is structured as follows: section 4 will cover background to help readers understand some of the terminology used; Section 5 will cover literature review; Section 6 will cover implementation of parallel programming, solution to the issue of DDOS, and source code revision; Section 6 will then cover the procedures of data collecting, testing, and comparison with the original software. Section 7 will conclude the whole paper.

4 Background

4.1 Python

Python [7] is a high-level programming language that is free. It uses an interpreter instead of a compiler. An interpreter [7] can execute a line of code without going through the whole program. A compiler [7] needs to go through the whole code, compile it to machine code, and create an executable before running it. Python follows an object-oriented [8] paradigm, which mean programming style will be based on objects, which are called classes. A class will have its own data and methods. Classes and objects are treated the same as real world objects, so no two objects should be the same. An example would be having a car with its make, model, and year as data, while turning the engine on will be one of its methods. Python uses dynamic semantics such as dynamic typing and binding. [9] Dynamic semantics makes variable assignment easier by not making the programmer initialize the type of the variable: integer, float, double, Boolean, etc. By using dynamic typing and binding, programmers can use the same variable name, and if they are done with it, assign a new type without the need to specify it. The purpose of dynamic typic and binding is to make coding faster by focusing more on the logic rather than on each variable. Python has a nice and easy structure for people who are just starting to program because it emphasizes in readability and indentation. Lastly, Python has huge libraries and packages that will assist programmers because difficult algorithms are already implemented to perform heavy calculations such as in statistics and calculus. Dynamic semantics makes programming easier than other computer languages. We can use Python to make parallel computing easier.

4.2 Parallel Computing

Before and after parallel computation was a practice, program uses serial computation. Serial computing is when a program executes a single statement of code at any given time. [10] Meaning, serial computing reads and executes code one after the other, until the program ends. So, what is wrong with that? Nothing, but to gain speed and execute another part of the program that does not need any of the previous line of code, parallel computing is one of the solutions. Parallel computing [10] is mainly used to increase the speed of a program. Rather than waiting for a program to execute previous lines of source code, or statements, parallelizing can eliminate the wait. Parallel computing has certain requirements such as having multiple processing unit cores. It can be as little as two core or as huge as thousands of cores. The beauty of parallel computing is that the utilization of the hardware's power is for the software's usage. Parallelization makes use of all available cores or threads, depending on how it is designed, and that can speed up our software.

4.2.1 Multithreading VS Multiprocessing

Multithreading [11] is not parallel computing, but it is a parallel programming practice. More importantly, multithreading can be done using a uniprocessor. A thread is a simple sequence of instructions that tells what a computer needs to do. Multithreading is the process of switching threads to execute multiple sequences of instructions. The switching is done fast and made to look like it was executed simultaneously.

Multiprocessing [11] is a true parallel computing paradigm. Multiprocessing utilizes the processing unit's cores to executes different section of the program. Making possible for a program to run code in asynchronized order. Multiprocessing can be done in a Central Processor Unit (CPU) and in a Graphics Processing Unit (GPU).

4.2.2 CPU VS GPU

There are differences between using CPU and GPU in parallel programming. [12] First of all, CPU has a smaller number of cores in comparison to a GPU. To clarify, CPU cores can, currently in the year 2018, be between 1 and 8. Of course, this is counting a single CPU and not including clustering. Clustering is connecting multiple computers together, so that they can perform or execute instructions using all of the CPU cores from all of the computers, which are called nodes. Even though CPU may have a lower number of cores, each core is running in high frequency and the amount of cache is larger than a GPU. CPU is also good at predicting branches and performing an out of order execution. Furthermore, when multiple cores are used, CPU multiprocessing can perform task-parallelism by splitting the task to all available cores.

GPU have a larger number of cores, hundreds of them. Each core run from low to medium frequency and have a small amount of cache. As for parallelism, GPU uses multithreading and multiprocessing to executes tens of thousands of threads. Meaning, GPU uses all the available cores and spawn threads on each one of them. GPU can perform data-parallelization better because each core can be treated like a storage and can be accessed concurrently, thus performance is increased.

4.3 Defense Mechanism of DDOS

Distributed Denial of Service, DDOS, [13] is a defense mechanism that a website can have. DDOS can be triggered in many ways, such as when a repetitive task that a human would not likely perform occurs. Another way to trigger a defense mechanism is when a repetitive amount of request/download from a website occurs. Lastly, accessing links that are meant to catch a web crawler is called a honey pot. Honey pots are links that cannot be accessed by regular users and can only be seen if the page source is viewed, which makes it a great trap. There are good and bad reasons for having DDOS. The good part is getting rid of malicious attackers, but the bad part is not being able to automate certain tasks. Things such as automatically checking if a website is working properly through the use of web crawler and one of the biggest companies, google, uses a web crawler to make a website visible when searching through their search engine.

4.4 Web Crawler, Web Scraper, and Data Mining

DDOS and web crawler has been mentioned a lot, but how is this related to optimization? Well, web crawler [15] is the focus of the current project and it will be optimized. Web crawler can trigger the defense mechanism of DDOS because it is making multiple request from a website. Additionally, optimization does not only mean gaining speed in a program, but also making the program work properly and with as few errors as possible. Web crawler has names such as web scraper and data miner. All of those names do the same tasks: crawl over the website's pages and gather data. Web crawler automates repetitive tasks.

5 LITERATURE REVIEW

5.1 Old VS Modern Technology

Technology has changed our lives at home, school, and work. In 1976, Cowan conducted a research study of how family's role was changed by implementing technology at home. The majority of the housework were done by women, but with the implementation of the technology in the household, manual workload got reduced and family roles almost became non-existent. [4] On the other hand, school usage of computers has increased the expectation of how much knowledge a student will have. Before the computers were introduced to schools, expectation for students learning algebra were 3.5%. [5] It is not bad that the expectation is higher because the students have more tools to help them. Even now, 2018, the standards of knowledge we had as a high school graduate is higher than it was 10 or 20 years ago. Therefore, expectation are based on what students are capable of and not just because new tools were introduced. Concerning workplaces, research was conducted in 2001 by Black. Black measured the productivity level using the old Fordist model and compared it with the new high-performance work system. Along with the model and system, they also added union. The difference in number was 4.5% higher productivity for the Fordist model and 13.5% higher productivity level in the high-performance system. [3] So, how are all of these optimizations related to web crawler's optimization? Some, or even majority, of us may not know that our famous search engine, Google, is the biggest web crawler. Google search engine crawl through the internet, so that when users search something, it will give the best match it can find. In relates to the web crawler that is being optimized in this research. Knowing that many people may someday use this web crawler, optimizing it will help those people have a nice software to use and have an extra time in their hands to do other things.

Digging deep into web crawler, data mining is one of the many functionalities of a web crawler. Data mining used to be slow and the memory usage was huge. [15], [16] The reason that data mining was slow is because the old algorithm used are not as developed

```
# iterate over each CIK in the list of CIKs
for CIK in cikList:
    rnum = int(CIK.strip("\n")) % 10
    sleep(rnum)  # for debugging, there is still an issue with long runs of the program. It ends
    # up making so many requests so fast that the DDOS defense mechanisms of the SEC
    # site are triggered. Adding sleep will mitigate this problem but slow it down tremendously

    if int(CIK.strip("\n")) % 100 == 0:  # sleep 2 minutes every 200 CIKS
        sleep(120)
```

Figure 1. Source code to put the program to sleep for 2 minutes.

and as efficient, but with the newer algorithm researchers showed that program can be optimized by having the proper algorithm implemented. The algorithm that were used are called Apriori and AprioriTid which researchers compared with AIS and SETM. In addition, researchers combined what is good in both Apriori and AprioriTid and made a new algorithm called AprioriHybrid. As for the high memory usage, researchers compared BTK and iNTK. BTK used less memory and performed faster in comparison to iNTK. The differences between the BTK and iNTK are the type of trees and algorithms that were used. Both the Apriori and BTK researches helped made the program, that the algorithm was used on, faster. These researches also showed that new and better algorithm can be created.

5.2 Increased Demand for Faster Application Overtime

Increased demand for faster application can be seen as technologies advances. An example would be mobile phones. Newer phones get faster and better with the new hardware being put in them. As a result, people want newer phones and if they use some of the older models, complaints and critics will be heard. So by optimizing programs, the need to buy new devices will not be as often as how the society does in our current year, 2018. Prioritizing users' needs rather than giving business an updated product to sell. This section relates to web crawler's optimization as a motivation to help people get the most out of their devices.

5.3 Parallel Computing from Hardware Dependent to Portability

Some of the parallel programming tools were designed to communicate with one type of processing unit. [17] In Karimi's research paper called "A Performance Comparison of CUDA and OpenCL," that was published in 2010, the researchers talked about different ways to do parallel programing. CUDA is a hardware and a brand dependent because graphics cards, that can do CUDA programming, belongs to Nvidia. As for CPU parallel programming dependent, MPI is the candidate. With the problem of hardware dependent when it comes to parallel programming, OpenCL was made. OpenCL can communicate with many types of processors, not just CPU and GPU, which makes it portable. In fact, OpenCL can communicate and distribute instructions to both CPU and GPU in parallel. This section is related to web crawler's optimization because OpenCL is one of the optimizations that will be used, GPU parallel programming.

6 Optimization techniques

6.1 Solving or Finding Loophole on DDOS

There are a few suggestions on how to prevent triggering the DDOS defense mechanism. [18] ScrapeHero, a company that offers services and product to scrape websites, shared a blog called "How to prevent getting blacklisted while scraping," that is relevant to the paper's focus, web crawler's optimization for not being blocked of service. The first suggestion is to be nice to the website's server. It is achievable by making the crawler slower and that can be done by putting the crawler to sleep every few pages visited. The previous writer of the project used this technique and put the program to sleep, two minutes, every 200 CIKs, see Figure 1.

The comment on the source code shows that when the program run for a long period of time, DDOS get triggered. Further down the line, the source code of putting the program to sleep for two minutes every 200 CIKs is shown. The problem with this solution is that the users suffer from the program sleeping for two minutes doing nothing. Another suggestion is to be careful about honey pot traps. Honey pots are not a concern to this particular project because the website is directly accessed based on human interaction to the website, which makes links, that are not visible to human, not being used. The suggestion that we are interested in is the ip rotation through usage of proxies. By using different ip addresses, we can imitate a behavior of different computer accessing the same website. A code will be written and will act in the following behavior. First, look for a proxy that is working by testing the connection and receiving a 200 response, successful connection, when used on the website of the user's interest. The next step is to make sure to change the ip address every 200 CIKs. From sleeping to changing ip address, sleeping for 2 minutes will be eliminated.

6.2 GPU Parallelization using OpenCL

The web crawler is currently written in serial computation and not parallel computation. The program is a great candidate for task parallelization because assigning different ip address on each core will make the program run faster. The more cores that the program can use, the faster the speed will be. The design will be based on a basic parallel programing algorithm that divide the task to all the available core, see Equation 1 and Equation 2. Both equations are usable on data and task parallelism.

$$problem\ per\ core = \frac{n}{p} \quad for\ n > p$$

Equation 1. n is evenly divisible by p.

Equation 1 will be used when the number of problems, n, is divisible by the number of cores, p, for any problem n > p. For example, if n = 10 and p = 2, then each core will get 5 problem each, cutting half of the workload of a processing core and doubling the speed of the program.

```
tdSoup = soup.find_all("td")
for title in tdSoup:  # dump file name, file date, and link to file into a dictionary
    if "Documents" in title.text in title.text:
        newLinkFlag = True
        link = "https://www.sec.gov" + str(tdSoup[indexCounter]).split('"')[3]
        if ".txt" in link or ".htm" in link:

            if tdSoup[indexCounter - 1].text not in filingsData:
                filingsData[tdSoup[indexCounter - 1].text] = [
                    {"Filing Date": tdSoup[indexCounter + 2].text, "link": link}]
            else:
                filingsData[tdSoup[indexCounter - 1].text].append(
                    {"Filing Date": tdSoup[indexCounter + 2].text, "link": link})
    indexCounter += 1
```

Figure 2. Source code for scraping html td tag.

```
<tr>
  <td nowrap="nowrap">10-K</td>
  <td nowrap="nowrap">
    <a href="/Archives/edgar/data/20/000095012310024631/0000950123-10-
            024631-index.htm" id="documentsbutton"> Documents</a>
  </td>
  <td class="small" >Annual report [Section 13 and 15(d), not S-K Item 405]
    <br />Acc-no: 0000950123-10-024631 (34 Act)  Size: 938 KB
  </td>
  <td>2010-03-15</td>
  <td nowrap="nowrap">
    <a href="/cgi-bin/browse-edgar?action=getcompany&amp;filenum=000-09576
            &amp;owner=exclude&amp;count=100">000-09576</a>
    <br>10681142
  </td>
</tr>
```

Figure 3. Sample html code.

$$problem\ per\ core = \frac{n}{p} + 1.\quad for\ n > p$$

Equation 2. n is not evenly divisible by p.

The difference on Equation 2 is that we add one to Equation 1's result. Equation 2 will only be used based on the of remainder. For example, if n = 10 and p = 8, then the remainder would be 2. The remainder 2 will decides that Equation 2 will be used on the first 2 cores and the rest of the cores will use Equation 1. The first 2 core gets 2 problem and the rest gets 1, 2 * 2 + 6 * 1 = 10.

6.3 Code Clean Up and Refactoring

Refactoring and cleaning up code can optimize a program because bad code can cause a program to slow down. The web crawler was written to scrape all of the "td" tag of an html page. The problem with getting all of the "td" tag is that it will scrape unnecessary html content, see Figure 2 and Figure 3. Figure 2 shows that the web crawler will scrape all the html content with the tag of "td," which is the source code soup.find_all(td). After getting all the content that was asked, it then iterates through all the finding and search for any that contain "Documents," then add the appropriate links to a dictionary [6], a type data structure where item can be accessed using a key. Searching adds up to the total time that the program will run. In Figure 3, all the "td" tag will be scrape and all the program currently need is the "td" tag that has the nowrap="nowrap" in it. A solution will be to add the nowrap="nowrap" when using soup.find_all, see Figure 4.

```
tdSoup = soup.find_all('td', {'nowrap' : 'nowrap'})
for i in range(0, len(tdSoup), 3):
    links.append("https://www.sec.gov" + str(tdSoup[i+1]).split('"')[3])
```

Figure 4. More specific search than just searching for "td" tag.

Figure 4's first source code's line shows a modified version of Figure 2. The modification made it more specific on what the crawler are looking for. The rest of the line shows how we can directly get a link for every third item and store it in a list [6], another modification, because use of dictionary is more expensive in memory usage.

7 TEST PROCEDURES

Test procedures will include data collection and a comparison test with the original program. The data of interest is problem size, execution time, and number of cores that was used. The program will be run 200, 400, 600, 800, and 1000 for the problem size with the same number of cores to get the execution time. Successfully running the program with the maximum problem size, it will be more than 1000, can determine if the DDOS problem has been solved. The DDOS will run with the serial modified code before attempting parallelization. As for the parallelization, Equation 3 shows the speedup formula and Equation 4 shows the efficiency formula. Table 1 shows test data, while Figure 5 shows the graph for the speedup and Figure 6 shows the graph for the effiency.

$$S = \frac{T_{serial}}{T_{parallel}}$$

Equation 3. Speedup formula. [19]

Speedup, S, is equal to quotient of the time of serial, T_{serial}, and time of parallel, $T_{parallel}$. The times that was mentioned are execution time. The equation measures how much speed was gain.

$$E = \frac{S}{p} = \frac{\frac{T_{serial}}{T_{parallel}}}{p} = \frac{T_{serial}}{p \cdot T_{parallel}}$$

Equation 4. Efficiency formula. [19]

Efficiency, E, is equal to the quotient of Speedup and the number of processors, p.

	p	1	2	4	8	16
Half	S	1.0	1.9	3.1	4.8	6.2
	E	1.0	0.95	0.78	0.60	0.39
Original	S	1.0	1.9	3.6	6.5	10.8
	E	1.0	0.95	0.90	0.81	0.68
Double	S	1.0	1.9	3.9	7.5	14.2
	E	1.0	0.95	0.98	0.94	0.89

Table 1. Sample Data. [19]

P, as mentioned above, is the number of cores used. From a single core up to 16 cores were used. The problem sizes that were used were halved, original, and doubled. Using one core will not show difference because parallel computing required 2 or more cores. The aim for speedup is to be as high as possible, while the efficiency's goal is to be as closed to 1 as possible.

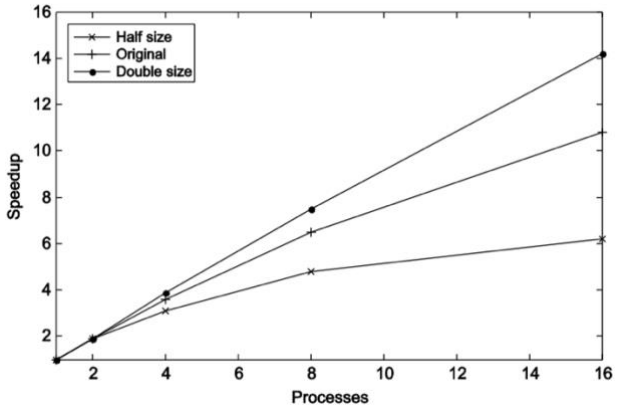

Figure 5. Speedup graph based on Table 1. [19]

Figure 5 shows that the more data the program gets, the higher the speedup is, and it is closer to linear. The explanation for higher speedup is because the more data the program gets, the slower the serial will get and the program will run for a longer amount of time. The ratio between serial and parallel time are increasing in difference.

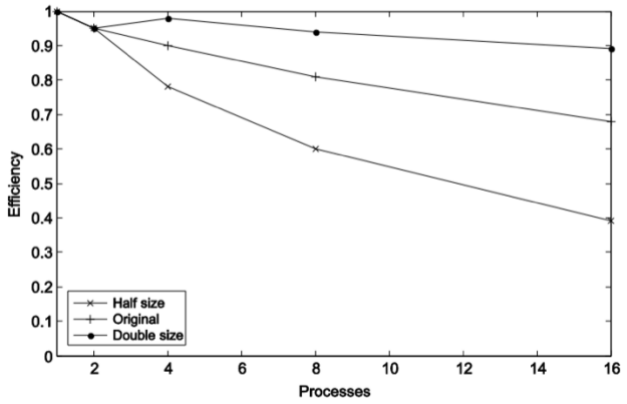

Figure 6. Efficiency graph based on Table 1.

Figure 6 shows that the higher the number of data the program has, the lower the efficiency. The explanation would be the same as the speedup, which is serial time plays a big part on efficiency calculation.

8 CONCLUSION

8.1 Summary

Web crawler might not be as familiar as other technology to a regular person, but it is a big part of our current life. Web crawlers are used by search engines such as google, and data scientists uses web crawler for data mining or scraping websites for certain information. Web crawler, that has been made by another student, has a few problems such as DDOS getting triggered, running too slow, and grabbing unnecessary data while crawling. A few suggestions to solve those problems are stated above. DDOS defense mechanism being triggered was solved by putting the program to sleep for two minutes, but that makes the program slow. As a solution, shuffling and rotating ip address by using proxies can eliminate the slow down. To gain additional speed, parallelizing the program using OpenCL, for portability purpose, will make the program run faster, more than double the execution time of a serial program. Gaining speed is the main focus of the project because speed is crucial in a program for greater productivity.

8.2 Future Works

8.2.1 Python

For Python, this falls more on the developer because of the python issue on multithreading. Python does not perform true multithreading because of the GIL [6] lock mechanism it has. The works that needs to be done is to allow user to manage their own resource locking without the use of in-line C codes. The in-line C code is powerful for programming certain code that Python does not allow, but many users that has been programming with python might not have prior knowledge with C language.

8.2.2 Optimization

For the future of web crawler, more optimization can be done. Different Programmer has different knowledge. Machine learning and AI can be implemented into the web crawler to make it smarter, correct the mistake done on its own and learn from it Another optimization would be to have more freedom such as selecting the type of document to fetch and selecting only a certain date range. With that freedom of given in the previous sentence, the program will gain speed because it does not need to fetch every single document.

9 REFERENCES

[1] Culler, David E., Jaswinder Pal Singh, Anoop Gupta, "Parallel Computer Architecture: A Hardware/Software Approach," *IEEE Concurrency 7*, pp. 83-84, 1999.

[2] K. E. Person. (-06-27T16:15:23.000Z). *Why App Speed Matters: Revenue*. Available: https://fly.io/articles/why-fast-pages-are-important/.

[3] S. E. Black and L. M. Lynch, "How to compete: The impact of workplace practices and information technology on productivity," *Rev. Econ. Stat.*, vol. 83, *(3)*, pp. 434-445, 2001. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0035635211&doi=10.1162%2f00346530152480081&partnerID=40&md5=5ed31f40030737c73ec6b85177d8632b. DOI: 10.1162/00346530152480081.

[4] R. S. Cowan, "The "industrial revolution" in the home," Technology and Culture, vol. 17, (1), pp. 1-23, 1976. Available: http://www.econis.eu/PPNSET?PPN=386438706.

[5] J. Roschelle *et al*, "Changing How and What Children Learn in School with Computer-Based Technologies," *The Future of Children*, vol. 10(2), pp. 76-101, 2001. Available: https://telearn.archives-ouvertes.fr/hal-00190610.

[6] Python Software Foundation Python Language Reference, version python 3.6, Available at https://www.python.org.

[7] Programiz. *Interpreter Vs Compiler : Difference Between Interpreter and Compiler*. Available: https://www.programiz.com/article/difference-compiler-interpreter.

[8] TutorialsPoint. OOAD Object Oriented Paradigm. Available: https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_object_oriented_paradigm.htm.

[9] S. Ferg, "Static vs. dynamic typing of programming languages," 2009. Available: https://pythonconquerstheuniverse.wordpress.com/2009/10/03/static-vs-dynamic-typing-of-programming-languages/.

[10] B. Barney, 'Intro to Parallel Computing.' Presentation, Livermore Computing, 2007. Available: https://www.lrde.epita.fr/~ricou/intro_parallel_comp.pdf.

[11] GeeksforGeeks, "Operating System | Difference between multitasking, multithreading and multiprocessing," 2018. Available: https://www.geeksforgeeks.org/operating-system-difference-multitasking-multithreading-multiprocessing/.

[12] J. E. Stone, D. Gohara and Guochun Shi, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems," *Cise-M*, vol. 12, *(3)*, pp. 66-73, 2010. Available: https://ieeexplore.ieee.org/document/5457293. DOI: 10.1109/MCSE.2010.69.

[13] M. Li, "An approach to reliably identifying signs of DDOS flood attacks based on LRD traffic pattern recognition," *Computers & Security*, vol. 23, *(7)*, pp. 549-558, 2004. Available: https://www.sciencedirect.com/science/article/pii/S0167404804001245. DOI: 10.1016/j.cose.2004.04.005.

[14] A. Heydon and M. Najork, "Mercator: A scalable, extensible Web crawler," *World Wide Web*, vol. 2, *(4)*, pp. 219-229, 1999. . DOI: 1019213109274.

[15] L. Huang *et al*, "A fast algorithm for mining association rules," *J. Comput. Sci. & Technol*, vol. 15, *(6)*, pp. 619-624, 2000. Available: https://search.proquest.com/docview/881243871. DOI: 10.1007/BF02948845.

[16] T. -. Dam *et al*, "An efficient algorithm for mining top-rank-k frequent patterns," *Appl. Intell.*, vol. 45, *(1)*, pp. 96-111, 2016. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84955591432&doi=10.1007%2fs10489-015-0748-9&partnerID=40&md5=369135fb8e3f6803f5ab9458c33433e6. DOI: 10.1007/s10489-015-0748-9.

[17] K. Karimi, N. G. Dickson and F. Hamze, "A Performance Comparison of CUDA and OpenCL," 2010. Available: http://arxiv.org/abs/1005.2581.

[18] ScrapeHero, "How to prevent getting blacklisted while scraping," 2014. Available: https://www.scrapehero.com/how-to-prevent-getting-blacklisted-while-scraping/.

[19] P. Pacheco, *Introduction to Parallel Programming*. Burlington, MA: Morgan Kaufmann, 2017.

Appendix I: Literature Table Summary

Year Publish	First Author	Before	After	Tools | System | Strategy | Algorithm Involved	Performance
1976	Cowan	Family roles are applied and many things that are needed to be done at home are usually fulfiled by woman.	Majority of task are done using technology. Making family roles non-existant.	Incorporate technology in household.	Decrease or elminate family role for household work.
2001	Roschelle	Expectaion from student's learning capabilities are low. only 3.5% of students were expected to learn algebra before completing high school.	Expectation are higher. Must be able to read and understand unfamiliar text and competent in the processes of scientific inquiry and mathematics problem solving, including algebra.	Used of computer in school.	With the ease of calculation by the usage of computer, expectation for knowledge gain increased.
2001	Black	Lower productivity.	Higher productivity.	Fordist model. High-performance work system.	4.5% higher productivity with union, but no high-performance work system. 13.5% higher productivity with union and high-performance work system.
2000	Huang	Mining algorithm that was used before was not as fast	New algorithm creted that out performed the old ones and created a new one by combining two similar algorithm	AIS, SETM, Apriori, AprioriTid, and AprioriHybrid (cmbination of the two Apriori, combining the good quality).	Apriori did the best on everything compared to AIS. AIS always did cosiderably better than SETM. Apriori and AprioriTid did the same on small problem, but the difference showed on larger scale.
2016	Dam	Mining uses a lot of memory and a not as fast.	Decreased of memory and increased in mining speed.	BTK and iNTK	iNTK uses more memory for all data set. BTK are faster in mining for all data set.
2017	McKinney	Working with data sets common to finance, statstic, and related field was not easy in Python.	increased in ease of used and readability for data sets are implemented in Python	pandas	Increased. readability: from array to tables, grouping, and allignment. Automaticaly handled missing data, and combined or joined data sets. Ease for statistical calculation and visualization.
2005	Dalcin	MPI only in C and C++	MPI implemented in Python	MPI	Not as best as C or C++, but with Pyhton extensibility feature, inline C code can be used for critical performance needed. It's also easier to use in Python because a lot are already automaticaly handled.
2010	Karimi	CPU or GPU parallel computing only. Hardware dependent	Portability and can parallel compute in either or both CPU and GPU. OpenCL can also use other type of processors.	OpenCL(portable) and CUDA(GPU only)	OpenCL's data transfer overhead does not change significantly for different problem sizes. CUDA version of the application processes more variables per seconds than the OpenCL version. The OpenCL kernel's performance is between about 13% and 63% slower, and the end- to-end time is between about 16% and 67% slower. CUDA performed better than OpenCL.
2011	Dalcin	Using high performance computing resources is expensive in Python	Relieve the cost by the usage of two MPI and PetSc in Python	C, MPI, PetSc	MPI and PetSc are able to support serious medium and large scale parallel applications.In comparison to pure C codes, MPI for Python can communicate Python array data at nearly full speed over Gigabit Ethernet and around half speed over shared memory channels. PETSc for Python overhead is consistently less than 10%.
2014	Pierro	OpenCl in original C/C++ is hard to do	Python makes it a bit easier, but it has some drawback on serializing object that creates overhead, which affects the performance.	OpenCL in Python and C/C++	No actual test was performed, but it shows the easability of coding in python when using openCL, named pyOpenCL.

Appendix II: Conceptual Plan

Optimizing a web crawler for speed and efficiency is the overall goal of the research project. The equipment required for the research are computer, with enough storage, and an internet connection. Testing will be conducted by running the unmodified program with different amount of CIK, 100, 200, 300, etc. The program will run the same amount of CIKs to get the average execution time. The same test will be conducted to the modified program. A comparison between the two result will show the difference on how much the speed the program has gained. With parallelization, the program will be tested with multiple cores. Data collection will consist of execution time from both the modified and unmodified program. Also, checking if the data collected by the web crawler is correct or not is another concern. Using the speed/efficiency equation for parallelization would be one of the analytical methods that will be used. Another is averaging the execution time. Experiment will vary on the amount of success and failure due to the project is exploratory. The project can take from 2 or more months. Studying the problem for base on previous work – 1 week. Studying where the problem lies – 1-2 weeks. Start writing the code and testing 2-4 weeks. Finish doing reports and any last improvement – 1 week. The project can be done in Pacific because all the equipment needed are already in any Computer Science class. The 3-research hypothesis in the project are: parallelizing the prom will give it speed, changing the IP will solved the issue of DDOS defense mechanism being triggered, and cleaning up source code will speed up the program. The project will have universality because people demands faster and efficient programs. The research can be reproduced by people who have computer science knowledge.