UNETHICAL OPTIMIZATION OF COMPUTER SOFTWARE AND POSSIBLE ALTERNATIVES – THE WEB

By

Alexander J. Cossifos

A paper submitted in partial fulfillment of the requirements to complete Honors in the Department of Computer Science.

Examining Committee: Approved By:

Dr. Burkhard Englert Dr. Ron Vetter

Department of Computer Science Faculty Supervisor

Dr. Kevin D. Matthews   
Congdon School

Dr. Shawn Bingham Dr. Toni Pence

Director of the Honors Scholars College Department of Computer Science

University of North Carolina Wilmington

Wilmington, North Carolina

April 2023

Table of Contents

[*ABSTRACT* 3](#_Toc134020323)

[*LIST OF FIGURES* 4](#_Toc134020324)

[*1* *INTRODUCTION* 5](#_Toc134020325)

[**1.1** **A Brief History** 5](#_Toc134020326)

[**1.2** **The Problem** 5](#_Toc134020327)

[**1.3** **The Dilemma of the Modern Web** 7](#_Toc134020328)

[**1.4** **The Question and the Goal** 9](#_Toc134020329)

[*2* *METHODS AND PROCEDURES* 10](#_Toc134020330)

[**2.1** **Platform and Tools** 10](#_Toc134020331)

[**2.2** **Chosen Websites** 19](#_Toc134020332)

[3 RESULTS 23](#_Toc134020333)

[**3.1** **Search Engines** 23](#_Toc134020334)

[**3.1.1** **Home Page Performance Profiles** 23](#_Toc134020335)

[**3.1.2** **Home Page Heap Snapshots** 26](#_Toc134020336)

[**3.1.3** **Landing Page Performance Profiles** 27](#_Toc134020337)

[**3.1.4** **Landing Page Heap Snapshots** 28](#_Toc134020338)

[**3.2** **News Sites** 29](#_Toc134020339)

[**3.2.1** **Home Page** **Performance Profiles** 29](#_Toc134020340)

[**3.2.2** **Home Page** **Heap Snapshots** 30](#_Toc134020341)

[**3.3** **Online Shopping** 31](#_Toc134020342)

[**3.3.1** **Home Page** **Performance Profiles** 31](#_Toc134020343)

[**3.3.2** **Home Page** **Heap Snapshots** 32](#_Toc134020344)

[**3.4** **Social Media** 33](#_Toc134020345)

[**3.4.1** **Home Page** **Performance Profiles** 33](#_Toc134020346)

[**3.4.2** **Home Page** **Heap Snapshots** 34](#_Toc134020347)

[**3.5** **Video Hosting** 35](#_Toc134020348)

[**3.5.1** **Home Page Performance Profiles** 35](#_Toc134020349)

[**3.5.2** **Home Page Heap Snapshots** 36](#_Toc134020350)

[**3.5.3** **Landing Page Performance Profiles** 37](#_Toc134020351)

[**3.5.4** **Landing Page Heap Snapshots** 38](#_Toc134020352)

[*4* *CONCLUSIONS* 39](#_Toc134020353)

[*5* *CRITICAL REFLECTIONS* 40](#_Toc134020354)

[6 Appendix 41](#_Toc134020355)

[BIBLIOGRAPHY 42](#_Toc134020357)

***ABSTRACT***

Software that was written for the first computers had to leave no room for wasted resources if they didn’t a program simply could not run. Since then, computer hardware has increased exponentially in speed, power, and storage so much that you don’t need to make many optimizations for software to run decently enough. But this has led to developers becoming negligent in optimizing their software and taking advantage of that leftover processing power for ads and telemetry. The problem with this, beyond looking at software on an individual level, is that with multiple of these resource hogs (applications) it can bring even the most powerful modern-day computers to a standstill. The modern web is one of the worst cases of this. My research uses Google Chrome’s built-in developer tools to profile multiple websites of various categories to show how much performance and time is being stolen from the end-user.

Keywords: Resource Hogs, Telemetry, Profile

***LIST OF FIGURES***

[Figure 1 - How to Access Google Chrome Developer Tools 11](#_Toc133946555)

[Figure 2 - Google Chrome Developer Tools Perfomance Profiling Initial Screen 12](#_Toc133946556)

[Figure 3 - Google Chrome Developer Tools Peformance Profiling Overview 13](#_Toc133946557)

[Figure 4 - Google Chrome Developer Tools Heap Snapshot Overview 15](#_Toc133946558)

[Figure 5 - Google Chrome Developer Tools Security Tab Overview 17](#_Toc133946559)

[Figure 6 - Excel Spreadsheet Data Overview 18](#_Toc133946560)

[Figure 7 - Search Engine Home Page Performance Profiles (ms) 23](#_Toc133946561)

[Figure 8 - Search Engine Home Page Heap Snapshots (kB) 26](#_Toc133946562)

[Figure 9 - Search Engine Landing Page Search for “google” Performance Profiles (ms) 27](#_Toc133946563)

[Figure 10 - Search Engine Landing Page Search for "google" Heap Snapshots (kB) 28](#_Toc133946564)

[Figure 11 - News Sites Home Page Performance Profiles (ms) 29](#_Toc133946565)

[Figure 12 - News Sites Home Page Heap Snapshots (kB) 30](#_Toc133946566)

[Figure 13 - Online Shopping Home Page Performance Profiles (ms) 31](#_Toc133946567)

[Figure 14 - Online Shopping Home Page Heap Snapshots (kB) 32](#_Toc133946568)

[Figure 15 - Social Media Home Page Performance Profiles (ms) 33](#_Toc133946569)

[Figure 16 - Social Media Home Page Heap Snapshots (kB) 34](#_Toc133946570)

[Figure 17 - Video Hosting Home Page Performance Profiles (ms) 35](#_Toc133946571)

[Figure 18 - Video Hosting Home Page Heap Snapshots (kB) 36](#_Toc133946572)

[Figure 19 - Video Hosting Landing Page Performance Profiles (ms) 37](#_Toc133946573)

[Figure 20 - Video Hosting Landing Page Heap Snapshots (kB) 38](#_Toc133946574)

***1*** ***INTRODUCTION***

**1.1** **A Brief History**

During the advent of computers, the hardware resources were at bare minimum for them to be able to function on their own. This also meant that software developed for those machines by programmers at the time could not waste resources and do only what their stated purpose was, no fancy bells or whistles. As time went on the hardware began having resources such as processing power, memory, and storage to spare leading not only to faster software but with it being more complete with features to help the end user that were not possible before. Fast forward to now and the capabilities of computer hardware have increased exponentially, high speed multi-core processors, copious amount of RAM, instantaneous storage access with SSDs, and GPUs (graphics processing units) with built in hardware accelerators to speed up certain processes that would wreak havoc on the CPU (central processing units) otherwise. These developments in modern hardware were not always realized in modern software. Modern software systems have stagnated with respect to performance and in some cases even regressed.

**1.2** **The Problem**

With all the hardware resources available for software to use there was no need to use the appropriate tools or optimize code when developing software anymore. Today, software is often made with a high-level language, like Python or Java, often inside a development framework. Examples include Unity and Unreal game engines which offer bleeding edge features for quickly making video games with a high level of abstraction from the actual hardware. The point to take away is that it has become a priority to make development easier to save time and money while sacrificing the end user experience by having software not perform to its full potential.

But on top of that comes the other way software developers are stunting their product, telemetry. It is advantageous in the time and effort for software to have telemetry abilities to provide valuable data to developers. There are cases where this can be justified, like seeing what features are utilized most by the user base, and when there is a setting to turn all telemetry off, as well as choosing what kind can stay or go. Most modern software goes beyond that use case and will not let you turn it off, even if there is an option to turn it off, it is for not collecting information which was not valuable to the developers to begin with. Why is this information of value to developers? Although they can use certain information related to the use of the software to make it better it is also because the data they collect can be sold for a lot of money, not just once but to multiple buyers repeatedly. The information that is so valuable to the entities that are willing to pay for it goes beyond relation to the software it was collected from. Those entities are looking for your name, phone number, email address, actual address, photos, what your social media is, your internet history, and even your private conversations. Not only can this be considered high unethical having your data being stolen from you often without your explicit consent, that will never be given through a hundred plus page terms of service no one has the time to read, but also your time, money, and computer hardware’s capabilities are being stolen from you.

If this was just a couple of mainstream pieces of software out of the bunch, then there would not be a major problem. But not only do most, if not all, behave this way, the ability and need to have multiple of these programs going at the same time as well as multiple instances of the same program can kill your computers performance. If you are on mobile, using a laptop or a smartphone, it can also kill its battery life. There are many places where we can see this exemplified, office suites, video games, utility apps that you must pay for but offer a worse experience than a piece of open-source software that does the same thing. The one I want to focus on is the most notorious where it has affected most of the population, at one point or another, which is the modern web.

**1.3** **The Dilemma of the Modern Web**

Websites started out as plain text, being able to display only basic information. Although computer hardware at the time could display much more, the infrastructure of the internet was a bottleneck. The dial-up modems used to connect were not only slow but could only serve one client per household, good luck if someone had to make a phone call. Broadband connections came along to replace dial up which could have multiple users but was still bottlenecked by bandwidth. Overtime images, sound, video, and other types of user interactive data were added to enhance the user experience and what websites could be used for as technologies like MP3 and JPEG made file sizes small enough to be used on these slow connections. As broadband technology improved around the late 2000s and early 2010s saw the advent of video hosting sites like YouTube, made possible by video compression technology like MP4, H.264, VP8 and others. Around the same time Wi-Fi started gaining mass market adoption along with the first smart phones. Twitter, Facebook, Amazon, new websites, and others all become popular around that time too, while not saturating the available bandwidth they benefited from that by being super-fast.

The rest of the 2010s saw how websites could take advantage of the websites with telemetry and adding new features that would take up more bandwidth alongside the original content. The late 2010s saw the introduction of fiber-optic internet connections, using light to transmit data at high speeds and low latency. On the wireless side, for home Wi-Fi continued to integrate adding a super-fast 5 Ghz band and more recently an ultra-fast 6 Ghz band. Also, mobile 4G LTE has replaced 3G as the lowest level of mobile data with great coverage and decent speeds. 5G and its Ultra-Wide variant are the latest revision promising home internet speeds on the go but currently cannot seem to make the mark and God forbid a tree branch gets between your phone and the tower.

This improvement, and it feels like we have hit a roadblock regarding website performance. It has stagnated due to modern technology like trackers and bad practices leading to unoptimized code. These both can cause even a new computer with the best specs to be slow. For perspective, I have a Framework Laptop with a Intel i7 1165G7, a four core eight thread processor, decent integrated graphics from it, 64 GB of RAM, 2 TG of solid state storage, and running Ubuntu Linux; Even with all of this power, speed, resources, and an optimized operating system if I go to a random website the fan will start to sound like I was trying to run the latest triple A game with maxed out graphics settings. Open enough Chrome tabs, play a video, or have a single tab open of a particularly awful site and your computer will start to lag. Even with the data collection occurring on almost any website you visit it should not result in your computer becoming a stuttering mess. There must be alternatives to the common websites we use that will not cost us performance and long-term hearing loss.

**1.4** **The Question and the Goal**

I went to see if anyone had researched this issue or how websites use your computer resources. With the help of my professor who is helping me with this research I found a research paper dating back to the very beginning of the 2010s (Butkiewicz, Madhyastha, & Sekar, 2011). The findings of the research showed that certain kinds of content, like stuff that uses Adobe Flash Player, uses more resources than those that do not. But what made me think that a new study must be done on this, besides the fact that Flash is no longer supported, is that the average size of images for the sites they tested was only 2 KB. Not only did that seem low for the time but there is no way that can be true now, we have surpassed the HD era and now the standard is a resolution four times as big appropriately being called “4K” or Ultra-HD. Website design factors in to this too, over a decade of tweaks to optimize the space that can be used by the website for its own content or ads on both desktop and mobile screens. Websites try to sell you their content, and they will try to put as much of it in your face as possible. It is time for new research to shed light on what these websites of many types do and if there are any alternatives that can benefit the end user.

***2*** ***METHODS AND PROCEDURES***

**2.1** **Platform and Tools**

I conducted research on how different websites use their computing resources, including what it spends its processing time doing as well as what it stores in memory. The computer I used to conduct these experiments was a MacBook Pro Retina 13 inch from early 2015. The processor is a 2.9 GHz Dual-Core Intel Core i5. It has 16 GB of 1867 MHz DDR3 memory and Intel Iris graphics 6100 that is allocated 1536 MB of that memory. The computer is running MacOS Monterey, Versions 12.6.3–12.6.5, and the browser being used to run the tests is Google Chrome, Versions 110.0.5563.64–112.0.5615.137, using its own code profiler and other developer tools. Not updating unfortunately could be avoided as small incremental updates that I could not control often made the OS (Operating System) lag or just broke Chrome. I will say that I did not notice any difference between updates that the OS or Chrome could cause, data collection remains a consistent process throughout. During testing, all background processes are minimized, and any open processes will not use up resources that would affect the test results. To record the result, I used Microsoft Excel with multiple sheets for the various categories and OneDrive open to keep my files with the results synchronized and backed up. All settings on Chrome are default without any changes or added extensions to simulate how most people will use the browser. The computer was connected to the internet via Wi-Fi on the University of North Carolina Wilmington’s (UNCW) network, though a few tests were done with the laptop connected to my Pixel 7 Pro acting as a Verizon 5G hotspot utilizing our campuses newly installed 5G tower.

Graphical user interface, application, Teams

Description automatically generated

Figure 1 - How to Access Google Chrome Developer Tools

To access the developer tools, I used to get the profiling results I right clicked then clicked “inspect” from the dropdown menu or I clicked the Google Chrome menu button, of three dots stacked vertically, hovered on “More Tools” in the first dropdown menu and then clicked “Developer Tools” in the second dropdown menu. Both bring up sub windows that split your screen space between the current website and the developer tools. It starts you on the console tab which will not be used for this research, the three main developer tool tabs used are “Performance”, “Memory”, and “Security” that if missing from the top bar can be seen using the button with the two stacked arrows pointing to the right designated as “More tabs” that shows and allows you to click on those missing tabs.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 2 - Google Chrome Developer Tools Perfomance Profiling Initial Screen

To begin the first profiling test head to the performance tab, once there, in a separate tab, open the Google Chrome setting page and select the “Privacy and security” section. Click on “Clear browsing data”, select the “Advanced” tab, click the “Time range” dropdown menu and select the “All time” option, this is to ensure that no precached data is manipulating our test results. Once cleared, to run the first performance profile either use manual mode, the whole circle in top left corner of the performance tab, or auto-reload mode, the circular arrow right next to it. In manual mode once the button is clicked it will start recording processes happening for your current blank tab, “about:blank”, clear the address bar, then enter the website URL and then press Enter. It will load the website and then display it normally in the left sub window, once the website seems like it has loaded everything in, a good way to tell is if in the bottom left corner there are no URLs popping up in a small black box, click the manual mode or rather record button again to stop the profiling. Auto-reload mode automatically starts profiling a website and stops when it decides that nothing else needs to be loaded in from the website give it time, between ten to sixty seconds depending on the length and the content, to display the profiled data.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 3 - Google Chrome Developer Tools Peformance Profiling Overview

On top you can see a timeline showing what the processing time is used by color, and below it synchronized with the timeline are screenshots of what the tab looked like at a certain point of time during the profiling. Near the bottom is a sub-sub window with “Summary selected showing you what the website spent processing time doing based on what you have range of the time selected using two gray bars or what auto-reload mode selected for you. It is easy to get the range for manual mode to be remarkably close to what auto-reload mode would produce, if you are off by a bit, it will mostly add idle time which is not too important to this research. The Summary tab has hollow color-coded pie chart with the total time range in the middle, including idle time, and to the right is the time in milliseconds are type of process took, the color of the process, and then the name of the process all stacked on top of each other. The types of processes and their colors are “Loading” as blue, “Scripting” as yellow, “Rendering” as purple, “Painting” as green, “System” as gray, and “Idle” as white. “Loading” is the time spent loading and parsing HTML, CSS, and JavaScript. Scripting is the processes enabled by JavaScript. Rendering and painting are displaying HTML and CSS from code to visuals. “System” is calling to use system resources, like your camera or microphone, and built-in system libraries like graphics APIs (application program interfaces) I then copied these results to the appropriate cells in Excel. I saved the performance profile by pressing a button with an arrow pointing down at a thin horizontal line. I can view those results again later by pressing the button next to the download button, the one with the arrow pointed up, and then select the file on my computer.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 4 - Google Chrome Developer Tools Heap Snapshot Overview

The second part of the profiling is the memory using the “Memory” tab. It will show three different options for profiling, for the most part “Heap snapshot” with the checkbox “Include numerical values in capture” under it checked, will be all that is needed. To begin the snapshot, press the blue button at the bottom of the developer tools window “Take snapshot” which will then begin the snapshot taking as low as a few seconds up to a minute based on the content. It will then show the heap snapshot selected in the left side bar with the result shown to the right. The results needed are not in the “Summary” tab in the sub-sub window so we will change it to “Statistics” in the dropdown menu, which gives a familiar looking hollow color-coded but not complete pie chart. To the right of that pie chart is the amount of memory, in kB, taken up by a certain aspect, the aspects color, and then the aspects name all stacked on top of each other. They are “Code” in red, “Strings” in green, “JS arrays” in blue, “Typed arrays” in yellow, “System object” in purple, and the combined total amount of memory at the bottom. Code refers to the instruction being sent to the CPU. Strings refers to text and other objects to be displayed. JS arrays are the array structure JavaScript uses. Typed arrays are raw memory, or rather binary data, so they are super-fast and do not have to be converted to another representation like regular arrays. System objects memory is used to maintain processes that call on built-in system libraries, like those that process audio and video.

For websites that have a continuous amount of content being profiled, like video hosting, two heap snapshots will be recorded one before and one after the video content has loaded in. During the playback of the video an “Allocation instrumentation on timeline” profile will be recorded with the “Record stack traces of allocations (extra performance overhead)” box checked. This will be a manual recording of the memory usage with a timeline showing when new memory is used and what that is. Just like the performance profile timeline you can select a time range to see only what was loaded into memory at that time. This is useful for video playback because it can show how the website decides to load in video, medium sized chunks at regular intervals, small chunks all throughout the run time, or one or a few large chunks. On the left side bar, I can choose to download that heap snapshot or allocation profile. Like the performance profile I can use a load button, this time at the bottom where the button to begin the profiling was, to view that data again later. I take these results and then put them in the appropriate cells in Excel.

Graphical user interface, application

Description automatically generated

Figure 5 - Google Chrome Developer Tools Security Tab Overview

Lastly is the Security tab which does not need to be started manually but the develop tools sub window must be opened before the website is loaded to get the results, it does not matter what tab. Once in the security tab you with start with the “Overview” on the left side bar selected, showing to the right the “Security Overview.” This shows if the HTPPS is valid, if the certificate is valid and trusted, details on the certificate, if the connection is secure, details about how the connection is secure, if resources are served securely and if there is any extra info on that. This info would be put into Excel. The other info needed does not require another sub menu but is listed on the left side bar under the overview as “Main origin”, “Secure Origins”, and if there are any “Unknown / canceled” origins. These are written manually into a separate Word document, one document for each website category, except for when the number of secure origins started exceeding around fifty plus items and could not be effectively done in the time from given. Once that started to happen screenshots were used to capture the origins. These files once created were put into the same folder as the websites profiling data files. The data put into Excel would be the main origin URL, the number of secure origins and the number of unknown or canceled origins.

Graphical user interface, application, table, Excel

Description automatically generated

Figure 6 - Excel Spreadsheet Data Overview

The final pieces of data in Excel are done before the rest, but since it is not the focus, it is pushed to the right behind all the other data. Using the Ookla Speedtest app from the Mac App Store, I recorded the date (YEAR\_MM\_DD), network, download speed (Mbps), upload speed (Mbps), Ping(ms), Jitter(ms), and Packet Loss (%). Most profiling was done using the UNCW (University of North Carolina Wilmington) Wi-Fi network with an average speed of ~120 Mbps, ping of ~17 ms, jitter being zero, and zero percent packet loss. For a few profiles I had to use my phone’s hotspot capabilities using the Verizon 5G network because certain websites were blocked. The speed using my phone as a hotspot ranged from about 50-100 Mbps down, 3-10 Mbps up, 37-53 ms ping, 11-14 ms jitter, and 0% packet loss. The download speed is the most important which is adequate, and while the upload speed is now much lower it does not really matter because all we are uploading to the internet are requests to load websites which are much smaller than what we a receiving. That is not as much as what the websites send to the computer to be stored in RAM.

**2.2** **Chosen Websites**

The websites chosen to be profiled can be put into multiple categories that will theoretically have different usage and levels of usage of the computer hardware. The categories are search engines, news, online shopping, social media, and video hosting. These categories were chosen because they all cover different use cases and potential resource needs. Search engines take your input and computes what it thinks you are looking for, and then returns that to you in some form whether it be single text hyperlinks or multimedia of some kind. Search engines are also a part of every other category to find content particular to that site. News and online shopping are both categories of sites that are trying to sell you something explicitly, so they are going to potentially use system resources to display items it wants to promote, track your preferences for items, and serve better suite items to you. Social media and video hosting both delve into user generated content and for the latter the addition of video playback. The websites will all be profiled at least once on their home page and a second time if they have any landing pages that visually or theoretically would have a big enough content change to warrant a profile. The landing pages profiled will be as similar as possible, for example, all search engines will search “google” when profiling their landing page.

For search engines we have Google, Bing, Yahoo, Yandex, DuckDuckGo, DuckDuckGo HTML, DuckDuckGo LITE, SearX, and Swisscows. Google is the primary search engine that people use and will use many resources due to the company being a data collection powerhouse. Bing, owned by Microsoft, and Yahoo, owned by Verizon, are the two main competitors to Google that are backed by big Corporations. Yandex is a search engine originating from Russia taking on the likes of Google, it is this one will be giving your information to the Kremlin. DuckDuckGo is a pro-privacy and generally “pro-free speech” search engine, that has three different versions to choose from, two of which being non-JavaScript versions. SearX is a search engine that is an aggregator, which means it pulls from a multitude of search engines to combine into one and is hosted across many domains. It is possible to set up your own instance of SearX. Swisscows is a search engine that is focused on education and blocks any “inappropriate” search results, a dedicated Google Scholar.

For news we have NBC, NYTimes, Forbes, CBS, CNN, Fox News, and RealClearPolitics. There is no main website that people go to for news here, besides Fox News for the right side of the aisle, and RealClearPolitics is just an aggregated site of news from the previously mentioned sites and more. There is also Apple News and Google News but those aggregators and not really meant for desktops and function better on phones, so I will not be test those.

Online shopping includes Amazon, Walmart, eBay, Etsy, Alibaba, and Wish. Amazon is the main online shopping site that people use. Walmart was selected because a short-lived competitor named Jet was integrated into Walmart’s site. eBay is the go-to site for buying and selling used goods. Etsy is focused on the sale of handmade goods. Alibaba and Wish can both be described as value product sites where stuff can be bought for cheap but often has fraud listings.

Social media being made up of Twitter, Gab, Rumble, Getter, Reddit, and 4chan. These sites were chosen because of the ability to view them without a profile, one of the reasons Facebook was not included. Twitter, while the biggest, is not as far ahead of its competitors compared to the other categories. Gab, Rumble, and Getter are a group of Alt-Tech sites created due to restrictions made by Twitter and Facebook alike. What Alt-Tech is will be explained more later.

The last main group, and what was the most demanding, is video hosting which has YouTube, Dailymotion, BitChute, Odysee (LBRY), and Rumble. YouTube is the biggest video hosting site and to some is also considered the second largest search engine, it is owned by Google. Dailymotion was a competitor to during the late 2000s and early 2010s but has not gotten more popular since and has been overshadowed by YouTube. BitChute, Odysee, and Rumble can be described as another group of Alt-Tech sites, created in part due to increased restrictions on the content that is allowed on mainstream sites like YouTube.

So, what exactly are these alternative technology sites I previously mentioned? They sites created to be alternatives to user-driven content sites, categories covered being social media, video hosting, and in a sense search engine. The market for these sites was not created overnight but over the years. One thing that created them was the abuse of copyright laws and DMCA. People often used copy right material in their videos whether that be music, clips from tv shows or movies, pictures, etc. And while people did often upload the original content unaltered these chases could have been handled on their own but instead the websites decided to do blanket demonetizations or even removal on that content which hurt the user base. This continues to be an issue today where content that should be protected under fair use is taken down unjustly. The other factor that became a catalyst for their birth was political censorship. It started out with just the “fringes” of the political spectrum, but overtime quickly started covering viewpoints and statements in the mainstream. Because of this the inclusion of Alt-Tech sites may be looked down upon by some but they provide us with an excellent contrast. Some of these Alt-Tech sites use frameworks that are different from the mainstream tech sites, some part due to innovation and some forced innovation due to them not being able to use those established frameworks. One of those frameworks would be LBRY, what Odysee uses to power its website, which is based on Blockchain, uses Web 3.0, and is a network that other sites can build from can have the exact same content.

**3** **RESULTS**

**3.1** **Search Engines**

**3.1.1** **Home Page Performance Profiles**

Figure 7 - Search Engine Home Page Performance Profiles (ms)

The results of the search engine home page performance profile give us an immediately visible contrast into these kinds of websites. The first thing to note is that Google does not seem to chart at all here, but since this is Google’s own browser it is possible that it is already preloaded into Chrome so there is no time spent on anything. Second, when it comes to profiles six through eight, eight which is SearX did score on the measurements, but it is so low compared to larger ones that they do not chart, and six as well as seven, being DuckDuckGo HTML and LITE, while like the later do not chart any scripting because there is nothing to script.

With those important pieces of info out of the way, the results are predictable. All websites took extraordinarily little time to load, with Yahoo taking the longest at just 195 ms. Both Bing and Yahoo take the longest amount of time overall, with Bing being slightly higher with scripting and system CPU time use. Yandex used around one fourth of the CPU time that Bing and Yahoo used, making it a smaller version of the three mainstream search engines. With Yandex being relatively new and not having any dedication towards privacy, it is possible one day it could catch up to the big guys in its CPU time usage. The rest of the search engines, including all DuckDuckGo versions, SearX, and Swisscows are not CPU time light. While you can say that Swisscows is the one spending the most overall CPU time followed by DuckDuckGo JS, at this point where we are getting to a total CPU time of around one second. Using search engines with a sub one second CPU time is ideal in some use cases but does come at the cost of looks and functionality, which of search engines are varied.

Google is the simplest, it is a blank background with the Google logo and the search bar which when something is initially typed autocompletes the query to the best of its ability. Most search engines, except those with no JavaScript, can do this. Bing adds onto this by making their home page a bit flashier and introducing a news section. Yahoo takes this and runs with it; the entire page is filled with adds for news stories. Because of this Yahoo’s number of secure origins are around triple to quadruple the amount Google and Bing had while dwarfing the rest of the search engines which on average had around one to two secure origins, maybe a few more in Yandex’ case. Speaking of which had a remarkably similar home page to Bing if talking about components. DuckDuckGo JS’ is most like Google along with SearX. DuckDuckGo HTML and LITE are even more bare, with a logo, a search bar, and a begin search button, all without auto search query completion as well. Swisscows home page is quite different and is in a way most like Yahoo’s in an effective way though. The search engine has the search bar at the top and then below it is all other information about the search engine and the company including new, updates, and advertisement not for other companies’ products but their own.

**3.1.2** **Home Page Heap Snapshots**

Figure 8 - Search Engine Home Page Heap Snapshots (kB)

The results of the search engine home page heap snapshot profile are a similar story. Yahoo takes up the most, because of all the ads it loaded up. Bing being the second largest is interesting, because while it is smaller it is not as small as one thinks it would be considering it has a fraction of the ads. Google is even more surprising, while again it is a good chunk lower it still is using lots of memory for code, given Google’s advanced algorithm this may not be much of a surprise. Besides Swisscows with a notable bump in the memory it uses for coding, the rest of the search engines use such a small amount of memory it is not worth considering.

**3.1.3** **Landing Page Performance Profiles**

Figure 9 - Search Engine Landing Page Search for “google” Performance Profiles (ms)

The results of the search engine landing page performance profile show comparable results to the home page one but a bit more interesting when we look at the details. Google does a lot of scripting, again because of their advanced algorithm from being in the game for so long. Bing uses CPU time in scripting, rendering, and system quite equally. Yahoo, Yandex, and DuckDuckGo JS all have a similar pattern of CPU time usage. The rest of the search engines are even lower, making performance worries while searching in multiple tabs an afterthought. Swisscows might be using up a decent number of resources for a privacy focused browser because of its filter which blocks “inappropriate” content.

**3.1.4** **Landing Page Heap Snapshots**

Figure 10 - Search Engine Landing Page Search for "google" Heap Snapshots (kB)

The results of the search engine landing page heap snapshot profile show a shocking result where Google takes it away in terms of memory usage and the other mainline sites, while using more than the bare bones search engines, are dwarfed by google. Besides Yahoo using a suspicious amount of memory for system objects the rest of these are incredibly low and should not need to be worried about.

**3.2** **News Sites**

**3.2.1** **Home Page** **Performance Profiles**

Figure 11 - News Sites Home Page Performance Profiles (ms)

On the news sites we see that scripting is the one thing that they do the most. Fox news is the highest in most categories but during testing CNN started playing live footage during profiling. This led to an automatic refresh, which could not be replicated because they mark your IP for no more live TV. The aggregate site is the winner here with the New York Times being the only non-aggregated site that comes close to it.

**3.2.2** **Home Page** **Heap Snapshots**

Figure 12 - News Sites Home Page Heap Snapshots (kB)

There is nothing much to differentiate the news sites when it comes to memory use. The only interesting finding is that CNN uses more memory for code, possibly because of its live TV streaming feature. This caused trouble in the initial testing because as profiling was happening the page reloaded because the live TV stream acted as a free trail. The website no longer automatically played the live TV stream as it marked the IP address as having used up its free trail. Though this heap snapshot was after that happened, this could mean that CNN still loads in all the code to stream it before it checks trail status of the IP address.

**3.3** **Online Shopping**

**3.3.1** **Home Page** **Performance Profiles**

Figure 13 - Online Shopping Home Page Performance Profiles (ms)

Online shopping is unfortunately not as interesting, with all using most of the CPU time for scripting, a bit for system, and little for everything else. Wish uses the most CPU time and with the knowledge that Wish is a site that is sort of “sketchy”, it might explain why.

**3.3.2** **Home Page** **Heap Snapshots**

Figure 14 - Online Shopping Home Page Heap Snapshots (kB)

Again, not interesting, all use a similar amount of memory of a similar configuration. Alibaba joins Wish for this one in using the most memory. Wish’s distribution for its memory usage is different from all the others as the next biggest aspect taking up memory is strings and not typed arrays.

**3.4** **Social Media**

**3.4.1** **Home Page** **Performance Profiles**

Figure 15 - Social Media Home Page Performance Profiles (ms)

Twitter seems to either have been preloaded to Google Chrome not allowing to get numbers for profiling or does just not work well with it. Interestingly Getter, an Alt-Tech site, and Reddit, a Big Tech site both use a large amount of CPU time. The previous version of reddit uses a fraction of the CPU time at the cost of looking “nice” but that can be all a matter of personal preference. Old Reddit and 4chan are similar in CPU time usage but differ in that Old Reddit’s is more spread out while 4chan has it all in scripting.

**3.4.2** **Home Page** **Heap Snapshots**

Figure 16 - Social Media Home Page Heap Snapshots (kB)

Getter dwarfs the rest of the sites through memory for system objects. Twitter and New Reddit are the only ones that use up a noticeable amount of memory while the rest of the sites can be forgotten about if trying to free up memory.

**3.5** **Video Hosting**

**3.5.1** **Home Page Performance Profiles**

Figure 17 - Video Hosting Home Page Performance Profiles (ms)

YouTube uses a great amount of CPU time scripting followed by Odysee and then Dailymotion. BitChute and Rumble are the lowest and do not use much of anything.

**3.5.2** **Home Page Heap Snapshots**

Figure 18 - Video Hosting Home Page Heap Snapshots (kB)

Story is similar for memory, YouTube uses an equal amount of memory for code and typed arrays, Odysee uses a ton of strings and half of that space for code, and Dailymotion puts it all into code. BitChute and Rumble are very resource efficient and do not use a lot of memory.

**3.5.3** **Landing Page Performance Profiles**

Figure 19 - Video Hosting Landing Page Performance Profiles (ms)

Once video playback begins YouTube uses the most CPU time and uses much of that scripting. Odysee and Dailymotion use an amount of CPU time right in the middle between YouTube and then at the lower end BitChute and Rumble. It should be noted though that profiling was done with the auto resolution, which is default for all sites except BitChute which does not have a quality setting. YouTube auto selected to 720p while the rest were around 480p.

**3.5.4** **Landing Page Heap Snapshots**

Figure 20 - Video Hosting Landing Page Heap Snapshots (kB)

YouTube, Odysee, and Dailymotion continue to be the largest resource users for video. The same pattern appears again for the memory as it did for the home pages. Unfortunately the allocation timeline was of no use in comparing these sites because the video was so short, under thirty seconds that all the data the needed to be downloaded was done all at once.

***4*** ***CONCLUSIONS***

After profiling all these websites of various categories and different target audiences the findings are mixed. For search engines, social media, and video hosting there are alternatives to the tried and true that offer lower resource usage and respect that your machine as well as your data are yours. When it comes to news sites though, and especially online shopping, there are few alternatives, and all the main players are quite similar. The only redeeming feature is that the aggregator method seems to work for news as well as it did for search engines. Online shopping, unless an aggregator can be made to access it without an account for purchases, will face no individual new competitors because of the need for a physical presence for shipping the goods. The unethical optimization of computer software for the web may continue for longer than wanted, but alternatives are already making headway and it is even better that you do need to give up much to use them.

***5*** ***CRITICAL REFLECTIONS***

**The experience of completing an Honors Thesis has been a tough and at some points soul crushing journey but gave me much needed experience in separate way. I based this thesis off a final paper I did for one of Dr. Vetter’s classes who is my faulty supervisor for this Honors Thesis. I have gone on a journey throughout my time doing it, not a physical but a mental one. Where I was no longer afraid of work and any daunting tasks that lay ahead. I unlocked my inner workaholic that I thought I lost during the lockdowns. I am thankful that not only Dr. Vetter but also Dr. Matthews and everybody else, Professors and fellow students who helped to guide me along not only my thesis but also my general education. I will be heading into graduate school here at UNCW, I plan to make up for lost time and opportunities but also to continue my research and work that will form my character into something better.**

**6** **Appendix**

**Link to Excel Spreadsheets with all data collected from research:** [**https://uncw4-my.sharepoint.com/:x:/g/personal/ajc7186\_uncw\_edu/Ech03hLe2StPh0SawRrHUEUBehOoWVUqtgz\_lRTWcfyn0w?e=YqTARW**](https://uncw4-my.sharepoint.com/:x:/g/personal/ajc7186_uncw_edu/Ech03hLe2StPh0SawRrHUEUBehOoWVUqtgz_lRTWcfyn0w?e=YqTARW)

Link to all documents and information used to complete this paper not including references: <https://uncw4-my.sharepoint.com/:f:/g/personal/ajc7186_uncw_edu/EpU8JsVhestOgFMWUpskZl8BeZIfEehJ978xaHqQk3t3fQ?e=xGcpT5>

# **BIBLIOGRAPHY**

Butkiewicz, M., Madhyastha, H. V., & Sekar, V. (2011). Understanding Website Complexity: Measurements, Metrics, and Implications. *Internet Measurement Conference* (pp. 313-328). Berlin, Germany: ACM.

Davies, J. (2015, January 31). *Using the Chrome web developer tools, Part 4: The Timeline Tab*. Retrieved from Command Line Fanatic: https://commandlinefanatic.com/cgi-bin/showarticle.cgi?article=art036#:~:text=Loading%20refers%20to%20network%20interaction,the%20CSS%20%22box%20model%22.

Davies, J. (2015, February 27). *Using the Chrome web developer tools, part 5: The CPU Profiler*. Retrieved from Command Line Fanatic: https://commandlinefanatic.com/cgi-bin/showarticle.cgi?article=art037

Davies, J. (2015, March 30). *Using the Chrome web developer tools, part 6: The Memory Profiler*. Retrieved from Command Line Fanatic: https://commandlinefanatic.com/cgi-bin/showarticle.cgi?article=art038

Grintsvayg, A., & Kauffman, J. (2023). *LBRY: A Decentralized Digital Content Marketplace.* Retrieved from LBRY - Content Freedom: lbry.tech/spec