Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

The influence of operating system on the energy consumption of software and algorithms

Johnny Youssef

Course of Study:	Informatik
Examiner:	Prof. Dr. Marco Aiello
Supervisor:	Prof. Dr. Marco Aiello
Commenced:	December 15, 2021
Completed:	May 15, 2022

Abstract

Energy consumption of computers rises with the continuous development of more powerful and complex components. To counter the increase in energy consumption resulting from this continuous developments, several solutions were implemented over the years including manufacturing more efficient hardware by shrinking the size of various components like the transistor or optimizing the software used to consume less energy. Many people nowadays use separate computers for work and entertainment with each one being used mainly to perform one or a couple of specific tasks like writing documents, compiling code or video editing. This thesis investigates if the operating system (OS) influences the energy consumption of software and algorithms running on it. To archive this, a series of tests were conducted. These include algorithms written in different programming languages and different software. In addition, the tests were performed across three different operating systems on two different computers. This not only allows the impact of different operating systems on the efficiency of programs or algorithms to be examined, but also whether this impact is the same with different hardware. The result of the conducted tests showed that some algorithms exhibited an increase in efficiency and performance of up to 50 % by simply changing the operating system.

Contents

List of Figures

List of Tables

Acronyms

CPU Central Processing Unit. 9, 13, 15, 17, 18, 19, 22, 24, 25, 27, 30, 31, 34, 35, 36, 38, 39, 40, 41, 47

DC direct current. 25

DDR Double Data Rate. 18, 25

GPU Graphics Processing Unit. 7, 9, 18, 23, 24, 25, 27, 39, 40, 41, 42, 43

HDD Hard Disk Drives. 19

JIT Just in Time. 47

JVM Java Virtual Machine. 47

kWh kilowatt hour. 18, 29

kWs kilowatt second. 29

NVIDIA smi NVIDIA System Management Interface. 24

OS operating system. 3, 7, 13, 15, 18, 19, 20, 21, 22, 24, 26, 27, 29, 30, 31, 32, 33, 35, 36, 37, 38, 40, 41, 43, 44, 45, 46, 47

SSD Solid State Drive. 19, 25

VM Virtual machines. 21, 22

VS Code Visual Studio Code. 22, 30

1 Introduction

Computers have become an essential part of our lives. They are influencing almost every aspect of it from managing banks and companies to video chatting, work, entertainment, and learning. Due to the constant developing, computers are getting faster and more powerful every year further increasing their usability and our dependency on them. However, this improvement does not occur without a cost and in this case it is the constant rising energy consumption of computers and data centers. According to [25], the world largest data centers use enough energy to power around 80000 US households. The total energy usage of data centers in 2018 accounted for around 1% of the global electricity usage with an expected increase by more than five times by 2030. This huge energy usage is coupled with a large CO2 emission which is estimated to be as big as the commercial airline industry. Creating more efficient hardware is one approach to help reducing the energy consumption of computers. This is mainly done by shrinking transistors size. However, as the size of the transistor currently is just a couple of silicon atoms, this approach almost reached its physical limits. Software are also being optimized to be more efficient further reducing the power consumption. The performance of a program or algorithm is influenced by many factors like the speed of the storage, memory, and central processing unit (CPU) and the amount of background process and services running.
In scope of this bachelor thesis, I want to investigate the influence of the operating system on the energy consumption of software and algorithms and if changing the operating system would alter the efficiency of an algorithm without any changing in its implementation. To archive this, three implementations of Dijkstra and Boyer Moore written in Java, Python, and C++ were tested, as well as an implementation of a linear regression algorithm written in Python. Three additional software were also tested. These were VeraCrypt, an encryption software, Lightworks, a video editing software, and Blender, a famous photos, videos and animations rendering software. These tests were conducted on two different computers, each running two different OSs. A total of three different OSs (Windows 10, Windows 11 and Linux Ubuntu) were used for the tests. In this way, the effects of different OSs on the efficiency of programs and algorithms could be studied.
The remainder of this paper is structured as follows: Section 2 will cover related work to this thesis. Section 3 will discuss the necessary fundamentals needed to understand how the energy consumption of a process is calculated and what factors influence it. Section 4 will cover the hardware setup and all the software necessary for the tests in addition to presenting the conducted tests. Section 5 will present the result of the conducted tests as well as the data collected during the tests. Section 6 will contain a conclusion about this work in addition to a short outlook.

2 Related work

The rapid growth of IT systems led the energy consumption of computers and similar devices to become a real concern since it became one of the factors determining how big or a powerful a system can be. This led to a number of researches on this topic to be conducted. A systematic search of Google Scholar and IEEE Xplore was preformed using predefined criteria and the search term (energy) AND (consumption OR usage) AND (software OR system). Multiple studies were found focusing on this topic and very similar topics. These studies present both hardware and software solutions or purely software solutions such as proposing metrics and a method to measure the energy consumption of a software [16] or introducing tools to help identify "the critical areas in a software code regarding the energy consumption using adapted Spectrum-based Fault Localization (SFL)" [7]. There is also a study that classifies exciting metrics for monitoring the efficiency and performance of data centers. It also introduces new metrics that take into account aspects such as the age and geographic location of a data centers to minimize their energy consumption and maintenance costs [26]. Since the standard way of using a watt meter only deliver information about the energy consumption of the whole system, it does not provide sufficient information regarding the energy usage of a single software which is why a tool like InnoMetrics was developed by [9]. It collects data regarding the energy consumption of a software during the development process by measuring and analyzing the active and passive processes within the system. This provides the developers with more information regarding "the energy consumption of both the development process and the resulting software product"[9]. Another similar tool was developed by [29] which can accurately estimate the energy usage of each OS process running in a Linux server using a function which combines CPU, disk and memory usage of the process. This method showed accuracy over 95% while also having the benefit of considering all the recourse of the system instead of using the CPU usage as a representing of the energy consumption of the whole system.
There are also studies that look at the impact of code obfuscation on mobile device energy consumption, such as [27], which provides information about the balance mobile application developers "must strike between protecting their applications and preserving the battery lives of their users' devices"[27] since both software piracy and battery life are important concerns. There are many ways to approach this problem both on the software and hardware level. Another study propose a hardware-based software to "physically measure the energy usage of android devices and automates the energy consumption testing of applications"[15]. A test harness was developed and used to automatically execute Android application and measure their power usage. [1] on the other hand propose a pure software solution which enable servers to enter a similar state to sleep "while maintaining their application's expected network presence". This solution shows impressive results of 60%-80% energy saving using thirty machines as a test setup. Another software based tool to measure the energy consumption of an application is GreenTracker by [3] which uses the average CPU usage of an application over a set period to estimate its usages. The authors used GreenTracker to compare the average CPU usage of Firefox and Safari to determine which browser is more efficient.

3 Background

3.1 Energy consumption measurement

Calculating the total energy consumption of any computer is very easy since it can be done by using a watt meter in order to get the total energy the system is draining from the power grid. This however becomes complicated when only the energy consumption of one process or program is calculated. Many solutions propose focusing only on the CPU usage and neglecting the rest of the system. While it is true that most software are CPU heavy meaning that the CPU has the most impact on the performance and energy consumption, other components of the system are also needed to be taken into consideration when calculating the energy consumption in order to archive more accurate results. This nonetheless makes the problem of calculating the energy consumption of single process more complicated as each component requires a different measuring approach. Virtualization, i.e., the partitioning of the hardware resources of one or more computers into several different virtual machines, is the basis of cloud computing. It introduces additional challenges to accurately measuring the energy consumption of a process, as several different pieces of hardware from different machines are combined into one virtual machine.
There are different methods to calculate the energy consumption of a process in a system. The first and most easy way is by using a watt meter. This can be done by first measuring the total energy consumption of the system while it is idling and after that starting the process which its consumption needs to be calculated and measure the overall energy consumption again. The energy consumption of the process can then be approximated as the difference between the two measured results. This is however a naive way to do this since the energy consumption can be influenced by any process running in the background while the measurements are performed, which means that the measured results will not be consistent because measurements at different times can lead to different results both when the system is idle and during the execution of the process. This can be improved by conducting many measurements and averaging the results which should produce a better estimate of the energy consumption of that process. This method nevertheless cannot be used to produce very accurate results, because any change in a one or more background process of the operating system will affect the conducted measurement leading to a reduced accuracy of the measured results.
Another more accurate method to calculate the energy consumption is by monitoring the system's resources while the process or software is running and logging the results. The results need then be averaged over the period of the test time and can at that point be utilized to calculate the energy consumption of that process or software by adding up the measured component's energy consumption. The Windows Task Manager provides a detailed information about the resource's utilization of each active process and software. However, at the time of writing this, Microsoft does not provide an option to save this information making it practically unusable meaning that a program with comparable functions with the additional feature of a result save option will be required. This will be further elaborated on in chapter 4. This method can be used to obtain the

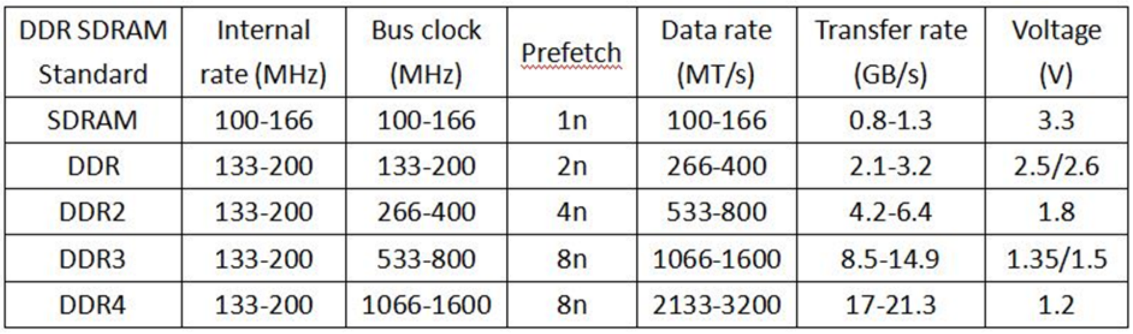
percentage CPU usage, amount of memory used, percentage Graphics Processing Unit (GPU) usage and the read and write speed of a process. These information need to be translated into energy for the consumption to be calculated which requires information regarding the physical components of the system including the peak energy consumption of the CPU and GPU under full load, the type, generation and speed of the memory, read and write speed of the storage device used in the system. In addition to the two methods mentioned earlier, there exist programs which feature the ability to calculate the energy consumption of every process running in a system. An example of such a software is powerTOP [21] for the Linux OS which can “provide an estimate of the total power usage of the system and also individual power usage for each process, device, kernel worker, timer, and interrupt handler”[21] .

3.2 The influence of software and hardware on the consumption

The energy consumption of any software depends on the hardware components of the system in addition to the software environment it is running in. Efficient hardware can reduce the overall energy consumption of the system, which in turn reduces the energy consumption of the individual software. The software environment and specifically the OS can influence the performance of a software depending on the priority and resources provided to it. This can influence the speed of a software which consequently influences the energy consumption. To further elaborate on the importance of more efficient hardware the following section will review the improvement the three most important parts of any computer received over the years.

- **CPU**: The CPU is the main component of any computer meaning that any improvement it receives will have a direct impact on the energy consumption of any process running in the system. This led the CPU manufactures to focus not only on improving the speed of their CPUs but also make them as efficient as possible. AMD, which is one of the biggest CPU manufactures, set a goal for itself in 2014 to deliver CPUs at least 25 times more efficient in 2020. According to its own testing [2] in 2020 they have produced CPUs which are 31.7 times more energy efficient. AMD also gives an example on how much energy these new CPUs can save. It claims that if an enterprise upgrades 50000 AMD laptops from 2014 to 2020 models, then they would use 84% less energy while also finishing any given task 80% faster. This translates to 1.4 million kilowatt hour (kWh) less energy used which is the equivalent to 1 million kilograms of CO2 emissions. This is just one example that illustrates the importance and impact of developing more efficient hardware on energy consumption and the environment.
- **Memory**: The improvement to computer hardware did not stop at the CPU but extends to every component of it. Double Data Rate (DDR) memory is also a very important part of any computer and has been improved dramatically over the year in the form of new DDR generation. Each generation is not only faster but also more efficient. For example, according to [33] DDR3 memory is 40% more efficient in comparison to the older DDR2 memory. Figure 3.1 shows the different memory generation with their specification which shows the improvement in speed and efficiency each generation has over the older ones.
- **Storage**: The last important computer component which needs to be mentioned is the storage which may not seem very important when talking about personal computers but is a huge part of any data center. According to [18] storage energy consumption totals about 25% - 35% of

the total energy of data centers further emphasizing its importance. [31] is a study about the difference between Hard Disk Drives (HDD) and Solid State Drive (SSD) and their impact on the energy consumption of data centers. The study conducted tests in which one to four HDDs and SSDs were connected to a laptop to compare the effect they had on the overall energy consumption. It concluded that while the system is idle, SSDs are more efficient but they use more energy than the HDD as soon as the storage system become active which is to be expected. This is a result of their higher read and write speed. The study also concluded that for all common server workloads of web, file, and mail, SSD storage is one to two orders of magnitude more energy efficient than HDD.



DDR SDRAM Standard	Internal rate (MHz)	Bus clock (MHz)	Prefetch	Data rate (MT/s)	Transfer rate (GB/s)	Voltage (V)
SDRAM	100-166	100-166	1n	100-166	0.8-1.3	3.3
DDR	133-200	133-200	2n	266-400	2.1-3.2	2.5/2.6
DDR2	133-200	266-400	4n	533-800	4.2-6.4	1.8
DDR3	133-200	533-800	8n	1066-1600	8.5-14.9	1.35/1.5
DDR4	133-200	1066-1600	8n	2133-3200	17-21.3	1.2

Figure 3.1: The difference between the different memory generation [33]

Software can affect the energy consumption of a computer or more specifically a process running on it by two different ways. The first way is the efficiency of the code which creates the process or the software and the second way is the OS. This includes aspects like time and space complexity of each algorithm used which can be also influenced by the implementation. Dijkstra, one of the most famous shortest path algorithms, requires a data structure to read and write partial solutions sorted by distance from the start. The implementation of this data structure changes the time complexity drastically. The time complexity for a Dijkstra implementation using an array is $O(V^2)$ while using a min-priority queue instead improves the time complexity to $O(V + ElogV)$ (V is the number of nodes, E is the number of edges, and O the time complexity).
Programming languages play an important role when talking about the speed and efficiency of an algorithm. [13] shows a comparison between six different programming languages regarding speed and resource usage. It concludes that there is a noticeable difference regarding the two investigated aspects with implementations in C and C++ being the fastest while using the least amount of memory making them more efficient in the conducted tests. Similar to the programming languages the OS also influences the energy consumption of any process running in it. This is because more efficient and optimized OSs use fewer overall resources and have less services running in the background which permit any software running in the system not only to use more resources, but also to have a higher priority, because of the lower number of overall active processes. This becomes more important in case of devices that use batteries and have a limited amount of available energy like smartphones. The more background processes an OS has, the less time the CPU and other component will be able to dedicate to other process. There have been already studies regarding the differences between the energy consumption of different OSs. In [19] five different OSs including Windows 10 and Ubuntu 18.04 are tested on the same laptop to eliminate any influence the hardware may have on the consumption. The study concludes that no major difference was presented between

the five different OSs. [8] is a similar study but focuses on the OSs of mobile devices instead. Unlike these studies, which focuses on the overall energy consumption of the system, my study focuses on the differences in efficiency software and algorithms may have while running on different OSs.

4 Test Design

4.1 Test environment

This section will focus on describing the hardware and software used to conduct the tests. Two different systems were used to conduct the tests with the aim of testing if the influence of the OSs on the tests differs when different hardware are used. The fist system is a desktop computer. The information about its components are listed in table 4.1.

Component	specification
CPU	AMD Ryzen 5 2600X six-Core Processor
Motherboard	A320M-A PRO MAX
Memory	16GB (2x 8GB) DDR4 2667MHz
Storage	SK hynix HFM256GDJTNG-8310A 256GB M.2 PCIe NVMe SSD Crucial BX500 480 GB 3D NAND SATA
GPU	NIVIDIA GeForce GTX 1650 SUPER

Table 4.1: Desktop computer components list

Windows 10 and Linux Ubuntu 20.04 are installed separately on this system to eliminate any interference using a Virtual machines (VM) may cause. This allows the same tests to be conducted on the same hardware using different OSs. In addition, the same tests are repeated on a different system to also test the impact of hardware on consumption while using the same OS. For this, a second testing system is used. It is a HP laptop and the information about its components are listed in table 4.2.

Component	specification
CPU	Intel® Core™ i3-8130U CPU two-Core Processor
Motherboard	HP 8487
Memory	8 GB DDR4 2400MHz
Storage	Toshiba KBG30ZMV 256GB PCIe NVMe SSD Intenso 2.5 SSD III
GPU	NIVIDIA GeForce MX 130

Table 4.2: Laptop components list

On the laptop both Windows 11 and Linux Ubuntu 20.04 are installed separately. The reason for running Linux on both devices is to test if different hardware will impact its influence on the energy consumption of the conducted tests.

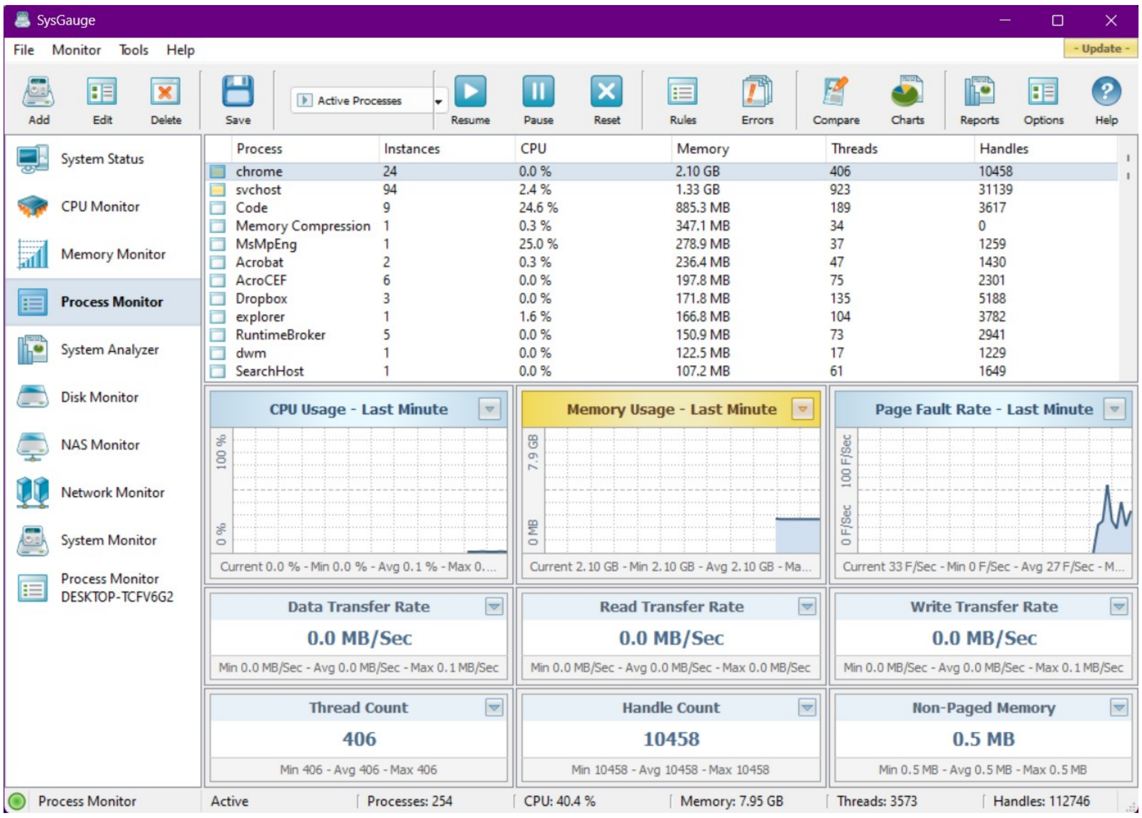
A Linux remote VM provided by the cloud services bwcloud was originally planned to be used as a third testing machine alongside the two above mentioned devices. This transpire later to not be possible because of the lack of information regarding the hardware dedicated for the instance provided to me. Cloud computing providers use the same hardware across different VM instances and do not provide specific information regarding the exact hardware chosen for each instance. This means that although some of the tests can be conducted using this instance, calculating the power consumed during these tests is impossible. This limits the range of machines which can be used in the tests to only standalone physical machines.

4.2 Utilized software

Several programs were required to perform the tests as well as monitoring the behavior of the different components of the systems while the tests were running. Below is a list of the programs used with an explanation regarding the necessity of each one.

- **Visual Studio Code (VS Code)**:VS Code is a light weight and fast code editor created by Microsoft. It runs on Windows, Mac and Linux. VS Code allows for viewing, editing, debugging, and running codes in many different programming languages including but not limited to Java, C, C++, and Python. This powerful editor was installed in every operating system and was used to compile the algorithms selected for the tests, regardless of their programming language. Using the same compiler to run the algorithms across the different OSs helps eliminating any influence different compilers may have on performance and efficiency, leading to more accurate results.
- **SysGauge**: SysGauge [30] is a powerful system performance monitor tool. It works similar to Windows Task Manager in that it provides a list of each process and service running in addition to a detailed information about the resources each process use. This includes the total CPU and Memory usage, the total number of instances running for each process, the number of threads, data transfer rate, read and write transfer rate. One of SysGauge most important features is the ability to export the above mentioned metrics for each process running since the program was started to one of many different file formats including HTML, excel and PDF. This tool was used on both windows 10 and windows 11 to monitor the behavior of each system while the tests are running. The produced results were then used to calculate the average energy consumption of each algorithm and software tested.
- **Anaconda**: “Anaconda Distribution is a Python/R data science distribution and a collection of over 7,500+ open-source packages” [4]. It was used in order to provide VS Code the capability of importing and using different Python library needed for the Python algorithms to run. Anaconda is available on Windows, Mac, and Linux and was used in every OS alongside VS Code.
- **Blender**: Blender is a free and open source 3d design software. It “supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation” [6]. Blender is chosen because it is cross platform program which mean that it can run on each OS used for testing. The versatility of blender allows many different tests to be performed. For this reason, two different tests were performed using it to test its performance in different scenarios.

- **Lightworks**: Lightworks [20] is a popular cross platform video editing software. It offers a free version in addition to multiple premium paid versions. Lightworks was chosen because of the increasing popularity of video editing over the years as social media has grown in popularity, making video editing a desirable career option.
- **VeraCrypt**: VeraCrypt is "a free open source disk encryption software for Windows, Mac OSX and Linux" [32]. It can be used to encrypt portable storage devices as well as hard drives. VeraCrypt features different encryption protocols including AES, Camellia, serpent, twofish and kuznyechick. In today's world where everything is connected to the Internet, privacy and security have become serious concerns for both businesses and consumers, which amplifies the importance of data encryption as it helps protect privacy and increases the security of data even in the event of a data breach.
- **MSI Afterburner**: MSI afterburner [22] is a famous GPU monitor and control tool. It features real time monitor of many different aspect of the GPU including the temperature, percentage usage, memory usage, core clock, memory clock and power consumption.



Process	Instances	CPU	Memory	Threads	Handles
chrome	24	0.0 %	2.10 GB	406	10458
svchost	94	2.4 %	1.33 GB	923	31139
Code	9	24.6 %	885.3 MB	189	3617
Memory Compression	1	0.3 %	347.1 MB	34	0
MsMpEng	1	25.0 %	278.9 MB	37	1259
Acrobat	2	0.3 %	236.4 MB	47	1430
AcroCEF	6	0.0 %	197.8 MB	75	2301
Dropbox	3	0.0 %	171.8 MB	135	5188
explorer	1	1.6 %	166.8 MB	104	3782
RuntimeBroker	5	0.0 %	150.9 MB	73	2941
dwm	1	0.0 %	122.5 MB	17	1229
SearchHost	1	0.0 %	107.2 MB	61	1649

CPU Usage - Last Minute
Current 0.0 % - Min 0.0 % - Avg 0.1 % - Max 0....

Memory Usage - Last Minute
Current 2.10 GB - Min 2.10 GB - Avg 2.10 GB - Ma...

Page Fault Rate - Last Minute
Current 33 F/Sec - Min 0 F/Sec - Avg 27 F/Sec - M...

Data Transfer Rate
0.0 MB/Sec
Min 0.0 MB/Sec - Avg 0.0 MB/Sec - Max 0.1 MB/Sec

Read Transfer Rate
0.0 MB/Sec
Min 0.0 MB/Sec - Avg 0.0 MB/Sec - Max 0.0 MB/Sec

Write Transfer Rate
0.0 MB/Sec
Min 0.0 MB/Sec - Avg 0.0 MB/Sec - Max 0.1 MB/Sec

Thread Count
406
Min 406 - Avg 406 - Max 406

Handle Count
10458
Min 10458 - Avg 10458 - Max 10458

Non-Paged Memory
0.5 MB
Min 0.5 MB - Avg 0.5 MB - Max 0.5 MB

Process Monitor | Active | Processes: 254 | CPU: 40.4 % | Memory: 7.95 GB | Threads: 3573 | Handles: 112746

Figure 4.1: SysGauge monitoring software

4.3 Data collection

In order to obtain accurate measurements of the power consumption of the hardware used in the tests, various software and scripts were used. The following section will describe how Information regarding the CPU, memory, GPU and storage of each OS has been collected.

- **Windows 10 and Windows 11**: As mentioned before SysGauge was used to collect and log the resources usage of each process running in the system. The only problem with SysGauge is lack of support to monitor the GPU which is needed for the blender animation test and the Lightworks video rendering test. For this reason, MSI afterburner was used to monitor and log the behavior of the GPU during the two previously mentioned tests.
- **Linux**: Linux on the other hand supports natively the ability to monitor and log the resource usage of the different running process. For this the top command can be used. It displays a list of all running process in the system with detailed information about each process including the CPU usage, memory usage, the PID of each process, the shared memory and the virtual memory usage. In order to save the result produced by the top command over a period of time a simple shell script was written. This script would simply call the top command once every second and write the output to a text file. Similar to SysGauge, top lacks support for monitoring the GPU which means that another software had to be used. For this, NVIDIA System Management Interface (NVIDIA smi) [23] was chosen as both GPUs used in the tests were from Nvidia. NVIDIA smi is "a command line utility . . . intended to aid in the management and monitoring of NVIDIA GPU devices" [23]. Similar to MSI afterburner it provides information regarding the temperature, percentage usage, memory usage and power consumption. Similar shell script to the one used in the top command was written and used to write the output of this command to a text file. In addition, Gnome System Monitor was used to monitor the disk usage of each algorithm and software while the tests were conducted. Gnome provides the ability to display the total amount of information that each service and software has read and written since its launch, so accurate results can be obtained in this regard.

After conducting the tests and using the above mentioned programs and tools to monitor and save the usage of different parts of each system during the tests, the total power consumption of each test need to be calculated. In order to accomplish this, the maximum power usage of each component need to be known. A simple, naive way to do this is to use the data provided by the manufacturer. The problem with using these data is that for parts like the CPU, the precise maximum energy consumption can differ between two identical CPUs. Reasons for this difference include the performance variation in different parts of the silicon wafers used in the manufacture process. This is also the reason to why each CPU has different overclocking potential. Another reason is the thermal output of the CPU and the type of the cooler used. If the cooling system used is not powerful enough, the CPU will have to reduce its clock speed to reduce the thermal output and thus reducing its power draw.
To produce more accurate results, the maximum energy consumption of each CPU and GPU used for the tests were monitored under full load. Table 4.3 compares the measured result to the data provided by the manufacture. As Table 4.3 shows there is a slight difference between the expected results delivered by the manufacture and the measured results. Almost all measured results are lower than what was expected.

CPU/GPU name	Manufacture data (watt)	Measured results (watt)
AMD Ryzen 5 2600x	95	95.8
Intel i3-8120U	15	15.02
Nvidia GTX 1650 super	100	99.464
Nvidia mx130	30	27.81

Table 4.3: Measured and expected power consumption of the used CPUs and GPUs

The storage and the memory are now the only components used in the tests without their power consumption being measured. These two components cannot be tested with only software but require the use of special hardware to measure their power consumption. The memory used in the two computers used for the tests is manufactured by ADATA. However, the manufacturers do not provide any information about their power consumption. For this reason, measurements published on the internet had to be used as a substitute. According to [5], 8 GB DDR4 memory uses between 2,98 – 3,06 watts depends on its speed. Since the speed of the memory in the laptop is 2400 MHz, the laptop's memory power consumption under full load was estimated at 3,02 watts. For the desktop PC, the maximum power consumption for the 16 GB DDR4 memory was estimated at 6.14 watts. As its memory runs at 2666 MHz, which is faster than the laptop's, it consumes more power.

Two types of storage were used in the tests, M.2 NVME SSD for Windows 10 and Windows 11 and SATA SSD for Linux Ubuntu on the laptop and the desktop PC. For the SATA SSD used for Linux on the desktop PC. according to the information provided by the manufacture in [10], the active maximum power consumption is measured at 4 W. no information regarding the energy consumption were provided by the manufacture of the SATA SSD used for Linux on the laptop. Because both devices feature the same capacity and speed, 4 W will be used as the maximum energy consumption of both devices. As for the M.2 SSD, neither manufacture provided information regarding their power consumption. Information regarding the power consumption of the SSD used for Windows 10 can be found in [12]. according to it, the SSD uses 3 W for active read and 3.5 W for active write. Since the storage was mainly used for writing in the tests, the measurement of 3.5 W was used to represent the maximum power consumption. As for the M.2 SSD used in Windows 11, I could not find any information about its power consumption online. However, the SSD is powered by direct current (DC) and draws 3.3 V and 1.1 A, as can be seen in Figure 4.2. This corresponds to a maximum power consumption of 3.3 W.

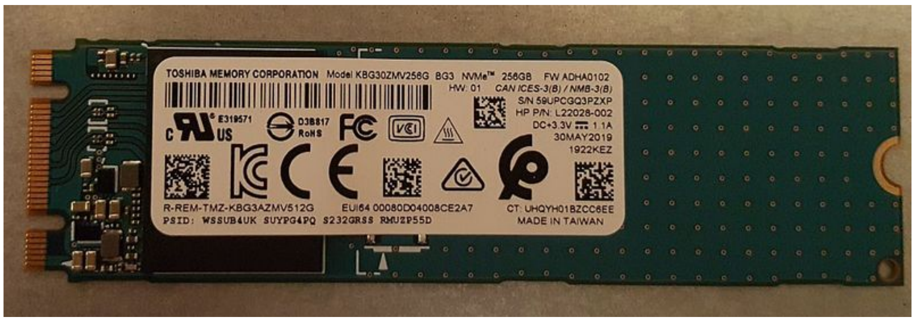

TOSHIBA MEMORY CORPORATION Model KBG30ZMV256G BG3 NVMe™ 256GB FW ADHA0102
HW: 01 CAN ICES-3(B) / NMB-3(B)
S/N 59UPCGQ3PZXP
HP P/N: L22028-002
DC+3.3V 1.1A
30MAY2019
1922KEZ
R-REM-TMZ-KBG3AZMV512G
EUI64 00080D04008CE2A7
CT: UHQYH01BZCC6EE
MADE IN TAIWAN
PSID: WSSUB4UK SUYPG4PQ S232GRSS RMUZP55D

Figure 4.2: The M.2 NVME SSD used for Windows 11[10]

With this the total power consumption of any process over a specific time duration can be calculated using the following equation:

$$E(p) = (E_{CPU}(P) + E_{GPU}(P) + E_{RAM}(P) + E_{Storage}(P)) * \frac{t}{1000} \tag{4.1}$$

E : power consumption
P : process
t : time

4.4 Test parameters and Methodology

After presenting the different software and tools used to run the tests and monitor the system usage during the tests, this section will present the conducted tests and explain which parameters were selected and how each test was performed. It should be first mentioned that the algorithms tests were aimed at studying the effects of different OSs on the efficiency of a given algorithm and not the efficiency and speed of the same algorithm using different programming languages. This is important to mention because the implementations of Dijkstra and Pattern search were tested differently depending on the speed of the programming language. Dijkstra, for example, was tested with different number of iterations up to 50 million. However, the results across the different iterations were almost identical in every aspect except for the time required, which increased linearly as expected. The Dijkstra test with 5 million iterations needed 67,136 seconds and the test with 50 million iterations needed 679 seconds. This behavior was also observed across all tested algorithms. This meant that very time consuming tests could be omitted in favor of faster and more frequently repeated tests because no different in their performance was found. For this reason, the number of iterations for each implementation was chosen so that they all took about 60 seconds to finish.

- **Dijkstra**: Dijkstra is one of the most famous shortest path algorithms. It was created by Edsger W. Dijkstra in 1956. It aims to calculate the shortest path between a start node and every other node of a given graph making it useful in many different applications including navigation systems. The most important aspect of any Dijkstra implementation is the type of priority queue used. Linked list, binary heap, Fibonacci heap and min heap are examples for priority queues which can be used. The implementation used in the tests used a binary heap as its priority queue. For the tests 3 identically implementation written in Java, C++ and Python were used. They all used the same type of priority queue and were given the same graph and start node in order to make the tests identical. The code for the implementation was provided by [14] which is free tutorials, live and online programming course platform. For the tests, a while loop was written so that Dijkstra would be called repentantly. The loop was repeated 1 million times for Python, 5 million times for Java, and 20 million times for C++. The reason for calling Dijkstra millions of times instead of using a large graph and only a single call is that any properly implementation of Dijkstra can perform a search on a graph containing every street in Germany in about 15 seconds or less, apart from the fact that the size of such a graph is several gigabytes. This information are based on a project I worked on a year ago.

- **Pattern search**: Pattern search is another famous problem in computer science. The goal is to search for a specific string pattern in a literature file, data bank or an array or list of words. Many different solutions were presented over the years including Rabin-Karp algorithm, Boyer Moore algorithm, suffix array and finite automate. For the tests Boyer Moore algorithm was used. This algorithm combines two different Techniques: bad character heuristic and good suffix heuristic, making it very fast. Similar to the previous test this test was also implemented three times in same three programming languages (Java, C++ and Python). The code for the implementation was also provided by [14] for the same reasons as for the Dijkstra tests. A similar while loop was used for this test, with 0.1 million iterations for Python, 4 million for Java, and 5 million for C++.

- **Linear regression**: Linear regression is an algorithm that aims to determine the relationship between two sets of data by creating a linear equation. This equation describes the dependency between these data to predict a dependent variable y based on a given independent variable x. This can be done using two or more set of variables. The test was performed using two randomly generated data sets with 150 million elements in each set.

- **VeraCrypt**: VeraCrypt has a benchmark feature which tests the performance of different encryption protocols. This feature was chosen for the energy consumption test. For the setting of the benchmark, the largest possible test file with size of 1 GB was chosen. This test would be running in the memory and would not involve the storage.

- **Blender**: Blender was used to conduct two different tests. The first one was an image rendering test in which a file would be imported and then rendered as an image. The file is one of the many different demo files provided on the Blender official website. The second test was an animation rendering test. This test was also one of the demo files from the official Blender website. The animation has a duration of 7.5 seconds and consists of 200 frames. Unlike the image rendering test, which relies only on the CPU, this test uses both the CPU and GPU to accelerate the rendering process.
  Another test that was planned to be conducted using blender was a realistic 3d animation of water pouring out of tube using physic engine. The animation is 40 seconds long. However, after starting the rendering process and finish rendering the first frame out of a total of 1000 frames, I decided to cancel this test because each device would have taken too long to complete the test. The desktop PC, the faster system used for the tests, took more than 4 minutes to render one frame. At this speed, it would have taken more than two and a half days to finish the rendering.

- **Lightworks**: Lightworks was used to import multiple videos and render them as a single video. This was used to learn how a video editing program behaves and how it allocates its resources while it is running in different OSs. The videos which were used for this test were downloaded from [24] which is a copyright free images, video, and music sharing website. The exported video was rendered at 720P resolution at 60 frames per second, as this was the highest rendering resolution available in the free version of Lightworks.

5 Results

In this chapter the results of the tests mentioned previously will be presented and evaluated.

5.1 Dijkstra in Java

The first performed test was the Java implementation of the Dijkstra's algorithm. The performance of this test varies significantly between the different OSs. Table 5.1 shows the resource and energy consumption of the test in the individual OSs. For both systems, Windows tended to require more memory to complete the test compared to Linux. The most important key figure in this test was the time required to complete the test. The time difference between Windows 10 and 11 compared to Linux was so great that Windows 11 on the laptop was faster than Linux on the desktop PC despite the hardware difference. It is also noticeable that the speed difference between the two Linux tests, which ran on different hardware, was only 16%, while Windows 10 was almost twice as fast as Windows 11. In my opinion, this is directly related to the way the test was performed, as after each Dijkstra iteration a result table was printed out. Looking at the read and write speeds, it is clear that Windows was faster than Linux in this aspect, which could explain Linux's slower result.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	17.63	1962.8	25.6	0	67.136	1.177927	0.000327
Linux desktop	21.07	1648	3.1	0	157.037	3.273643	0,000909
Windows 11	68.12	2729.52	15.2	0	131	1.534450	0.000426
Linux laptop	62.8	662.86	2.56	0	183	1.776915	0.000494

Table 5.1: Dijkstra algorithm in Java

This difference in performance becomes even more apparent when looking at the energy consumption of the tests under the individual OSs. Despite the speed difference, the energy consumption of the Linux test on the laptop consumed about 15.8% more energy, although Windows 10 had 311% higher memory usage, while the desktop Linux test consumed 177.9% more energy than the Windows 10 test. The time needed for the tests to finish was the most impacting factor on the consumption as can be seen clearly in Figure 5.1 which compares the energy consumption of the four tested OSs and illustrates the results in kilowatt second (kWs). It should be noted that using kWh or kWs for the figures produces the same results, since 1 kWh is equal to 3600 kWs.

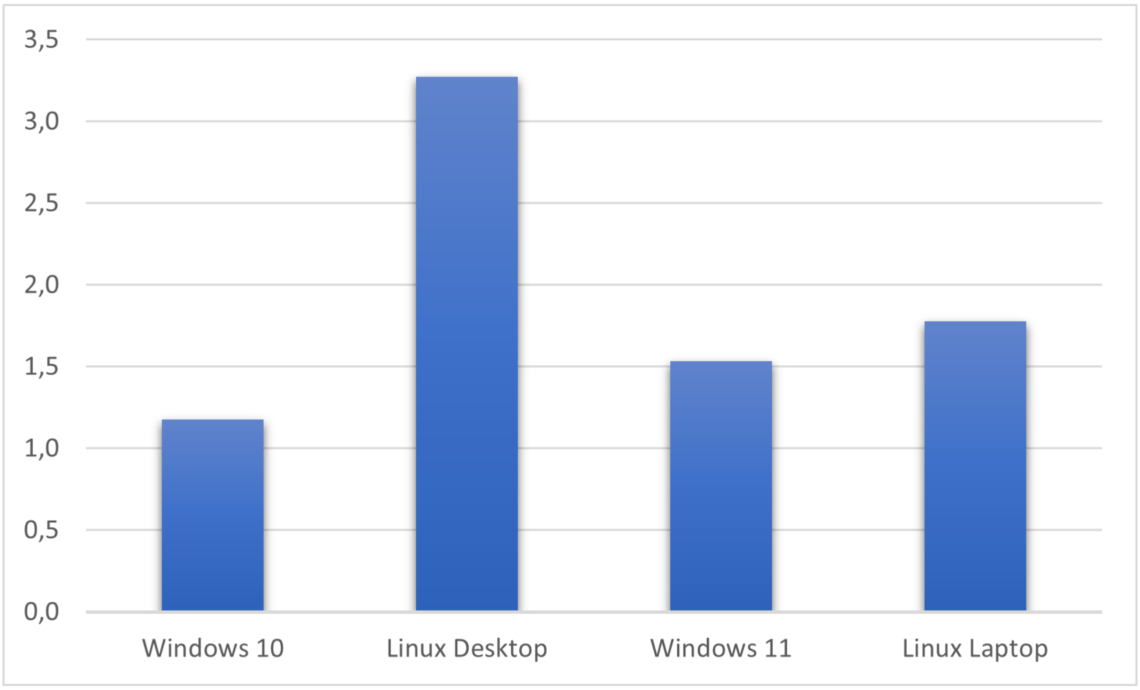


Figure 5.1: Energy consumption of Dijkstra in Java

5.2 Dijkstra in Python

The results of this test were similar to the previous test in that the tests performed using the Linux OS were slower and had higher power consumption. Some interesting findings can be seen in Table 5.2. The first is the higher CPU utilization in the Linux tests especially on the desktop PC. In the case of the laptop, using Linux instead of Windows 11 resulted in a 9.79% increase in CPU utilization and a 138.8% decrease in memory usage. A similar result can be seen in the case of the desktop PC, where the Linux test shows a 96.61% increase in CPU utilization and a 105.8% decrease in memory usage. Similar behavior is observed in the Python implementation of Pattern search, which leads me to conclude that it is related to how individual OSs compile and execute Python programs. Another interesting observation is the lack of information regarding the storage usage in the Linux OS. This is kind of strange, since after each Dijkstra iteration, an output of the calculated results is printed to the console in VS Code. However, this was not registered in the gnome monitor, which is specifically used to monitor the amount of data each process in the system reads and writes.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	19.68	1584.48	7.6	0	82.012	1.598757	0.000444
Linux desktop	38.694	769.9	0	0	122.22	4.566665	0.001269
Windows 11	71.31	1636.72	4.6	0	138	1.582759	0.000440
Linux laptop	78.296	685.36	0	0	143.464	1.724754	0.000479

Table 5.2: Dijkstra algorithm in Python

As for the power consumption of the tests. As can be seen in Figure 5.2, both Windows 11 and Linux on the laptop had very similar results to Windows 10 on the desktop PC. The only anomaly was the Linux test on the desktop PC which consumed 184.4% more power than the other three results. This huge power consumption can be attributed to two reasons. The first is the higher CPU utilization compared to the Windows 10 test which ran on the same hardware. The second reason is the time it took the test to finish, which was 49% higher than the Windows 10 test. Each test was repeated five times and results presented in Table 5.2 are the averaged results. This implies that the high energy consumption of the Linux test on the desktop PC was not a one time anomaly, but a consistent result.

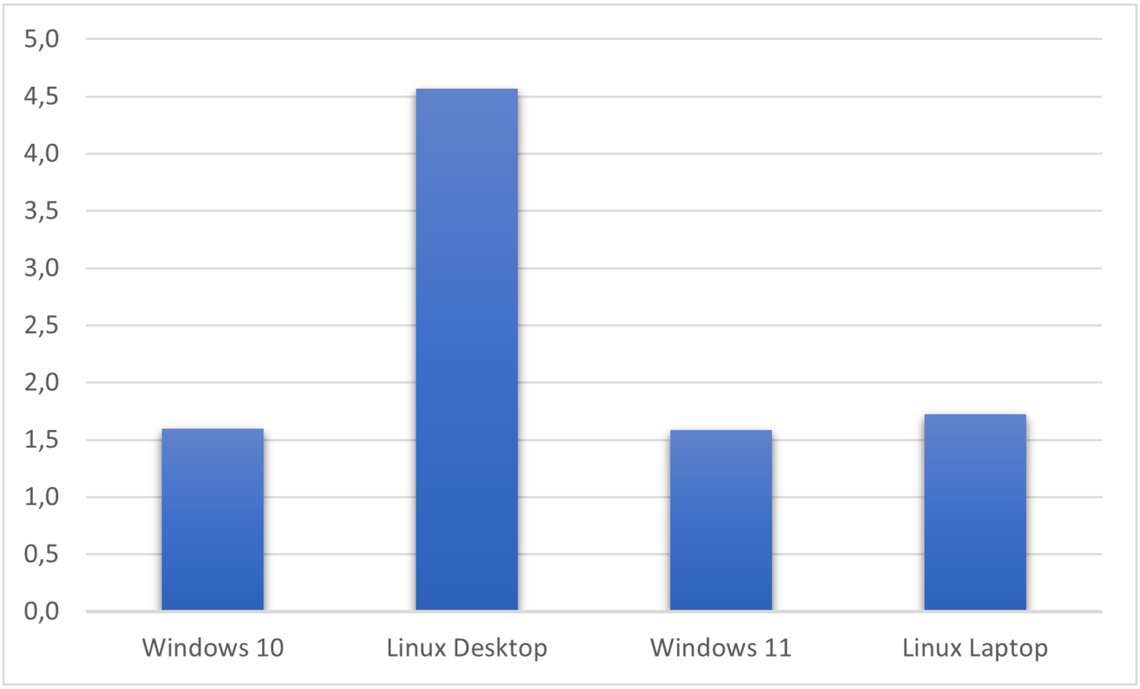


Figure 5.2: Energy consumption of Dijkstra in Python

5.3 Dijkstra in C++

In contrast to the previous tests, Dijkstra was faster and more efficient in C++ under the Linux OS than Windows 10 or Windows 11, regardless of the hardware used. The memory usage of Windows 11 seems to be lower than the other tests. This is attributable to the fact that the laptop has only 8 GB of memory and the test needs a total of 5.99 GB to run. Since Windows 11 normally occupies 3.9 GB of the memory, the memory compression feature of Windows 11 had to be used by the OS for the test to run successfully. This feature forces memory to be freed when the total memory usage exceeds 95%. Figure 5.3 shows the total memory usage of the test in Windows 11. It is clearly visible at which point the memory compression feature was used. This problem did not occur in the Linux test since the Linux OS occupies about 1 GB of memory which leaves enough memory

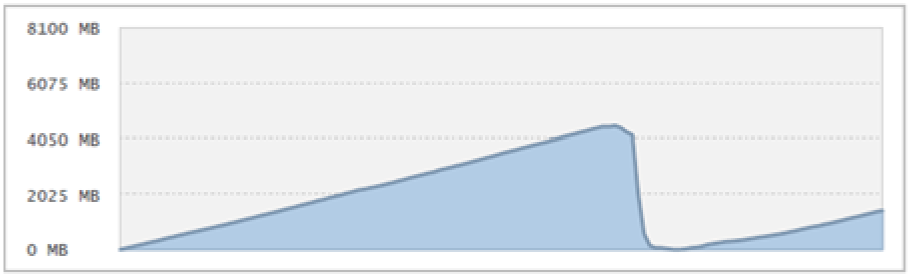


Figure 5.3: Memory usage of the Dijkstra algorithm in C++ under Windows 11

for the test. Although this feature is useful and allows the test to start and complete successfully even though the total required memory is not available, it slowed down the test and resulted in an increase in power consumption.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	14.51	4600.56	94.2	0	65.482	1.052386	0.000292
Linux desktop	19.75	4829	48.65	0	36.931	0.783904	0.000218
Windows 11	56.45	2555.86	46.2	0	153	1.650376	0.000458
Linux laptop	51.87	5154	36	0	50	0.504858	0.000140

Table 5.3: Dijkstra algorithm in C++

Due to the aforementioned memory compression feature, the test performed on Windows 11 was three times slower than the same test performed on the same laptop but using Linux instead. This resulted in the test under Windows 11 consuming about three times more energy than the laptop test under Linux. Windows 10 still took almost twice as much time as the Linux test on the same hardware and consumed 34.25% more energy. The large difference in speed between Linux and Windows 10 or Windows 11 can have several reasons. This includes C++ compilation time under the individual OSs, the cache management or OS services in the background. Whatever the reason may be, the results show a clear difference not only in speed, but also in energy consumption between the different OSs. The pattern search test in C++ should provide further insight into this behavior.

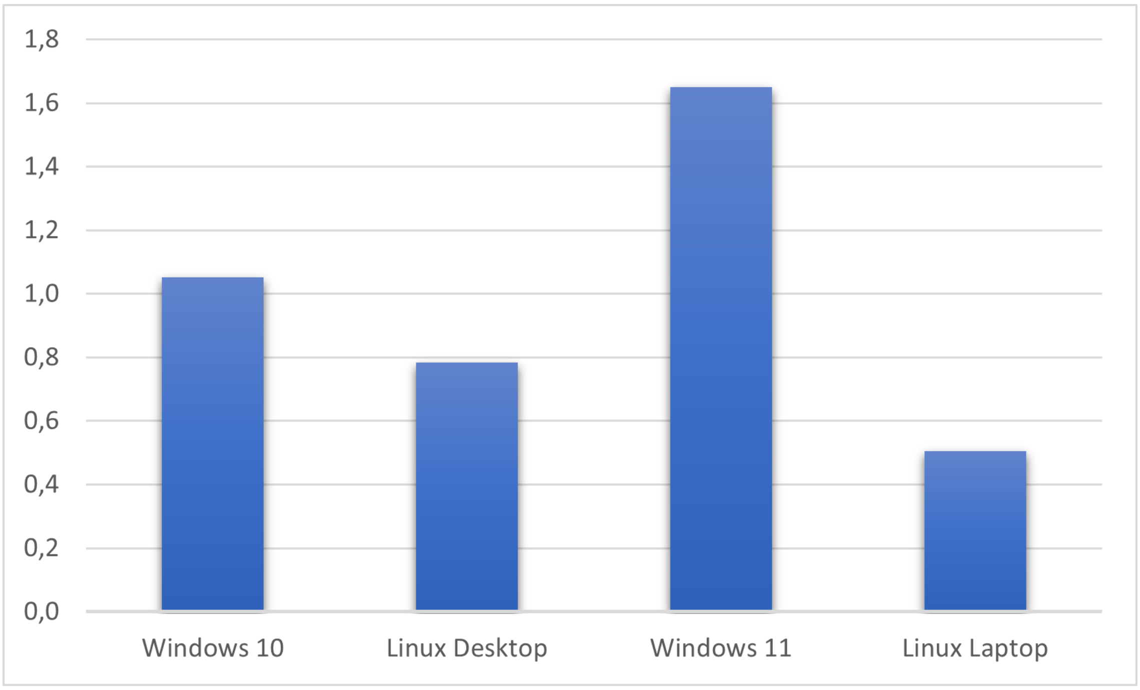


Figure 5.4: Energy consumption of Dijkstra in C++

5.4 Pattern search in Java

As Table 5.4 shows, similar behavior is observed here on the desktop PC as in Dijkstra's Java test, where the test ran slower in the Linux OS and therefore consumed more energy. On the laptop side, the slower performance of the Linux test did not result in noticeably higher energy consumption thanks to the lower average memory usage and slower read and write speeds.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	17.88	2176.21	20.1	0	57.835	1.043959	0.000290
Linux desktop	22.18	1300.8	4.33	0	69.288	1.509640	0.000419
Windows 11	69.94	2732.05	16.2	0	73.9	0.888019	0.000247
Linux laptop	66.26	664	3.37	0	87.683	0.897666	0.000249

Table 5.4: Pattern search algorithm in Java

As Figure 5.5 shows, the laptop's power consumption was almost identical across the two different OSs. However, the Linux test on the desktop PC consumed almost 41.9% more power than Windows 10. Similar results were previously observed in the Dijkstra's tests, where the Linux desktop PC test also had a similar average resource utilization as the Windows 10 test, but took longer to finish, ultimately resulting in higher power consumption. Although the total energy consumption on the laptop was similar, the Linux test was still slower than the Windows 11 test. This means that in a scenario where time is of the essence, which is almost always the case, a faster compile time of an algorithm with similar energy consumption is always considered superior.

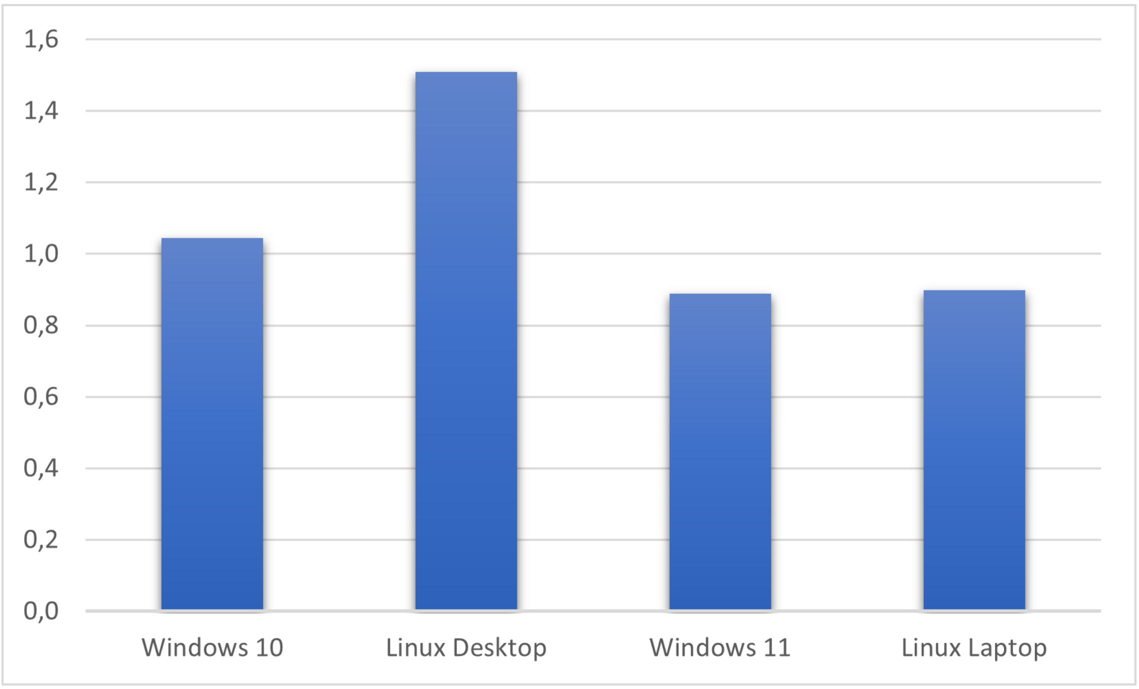


Figure 5.5: Energy consumption of Pattern search in Java

5.5 Pattern search in Python

Unlike the previous Python test, this test does not require a large amount of memory, which directly improved the Windows 11 results. The time required for the tests performed under Linux was not significantly different from that of the Windows tests with the same hardware. Table 5.5 shows that the biggest difference between the tests is in CPU utilization in the Linux desktop PC test, which was 174.4% higher than Windows 10. Combined with the fact that the test also ended 8 seconds later, this resulted in a significantly higher power consumption. A similar behavior was already observed in the Dijkstra Python test, where the Linux test also displayed a higher power consumption.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	11.9	1527.01	1.3	0	51	0.611581	0.000170
Linux desktop	32.69	841.6	0	0	59	1.866759	0.000519
Windows 11	45.91	1350	0.8	0	91	0.676594	0.000188
Linux laptop	74.2	1026.96	0	0	83	0.954856	0.000265

Table 5.5: Pattern search algorithm in Python

Similar to the desktop test, the average CPU utilization in the Linux test on the laptop was higher than in Windows 11, which resulted in higher power consumption. The lower difference in power consumption of 31.16% compared to 181.29% in the case of the desktop PC despite the higher CPU utilization is due to the lower power consumption of the laptop's CPU. This is also the reason for the significant difference in power consumption on the desktop PC between the tests in Linux and Windows 10. Figure 5.6 shows the energy consumption of the different tests side by side. It shows that the Windows 10 and Windows 11 tests produce very similar results despite their different hardware. The Linux test on the laptop had slightly higher consumption than the Windows 10 and 11 tests, while the Linux test on the desktop PC had significantly higher power consumption than the rest of the tests.

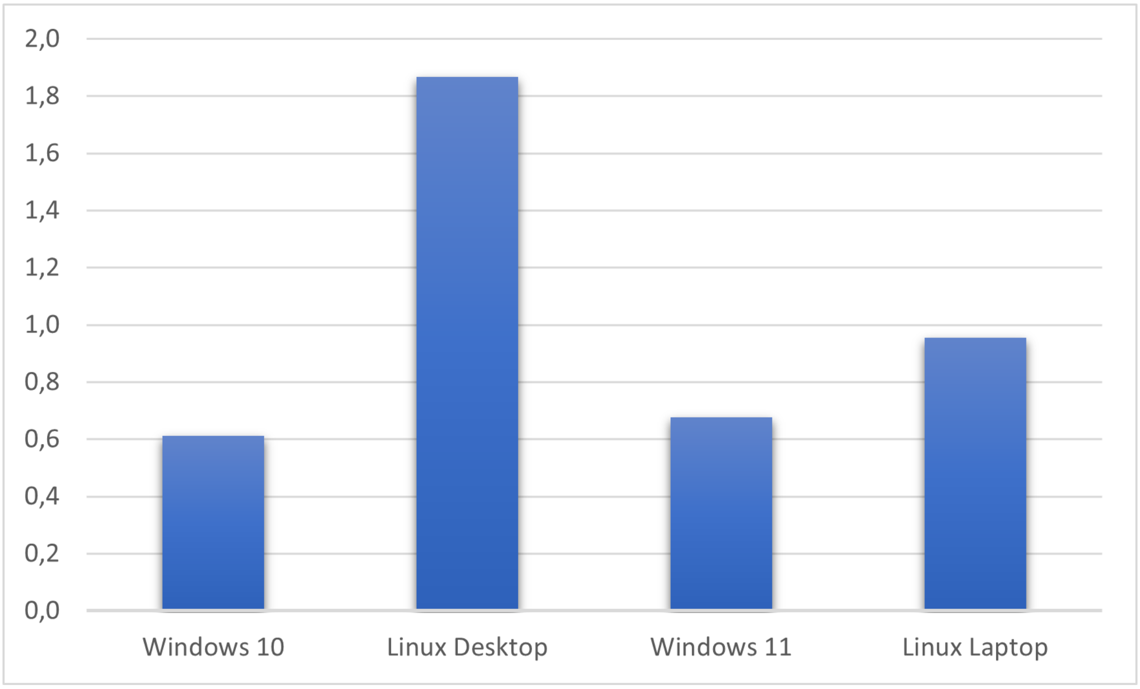


Figure 5.6: Energy consumption of Pattern search in Python

5.6 Pattern search in C++

In contrast to Dijkstra's results in C++, the result of this test shows no noticeable difference between the desktop tests and some time difference between the laptop tests, resulting in a corresponding difference in energy consumption. The Linux tests were faster than the Windows tests which is an observation that was also present in the previous C++ tests. The energy consumption in the desktop tests was higher using the Linux OS. This is most likely related to the higher CPU utilization. The smaller difference in the results of this test compared to Dijkstra's C++ test is most likely a result of the Boyer Moore algorithm not being as computationally intensive and complex as Dijkstra. This meant that the influence of the OSs on the tests was not as clear as in the previous tests. However, some distinguishing features, such as Linux's faster execution time, were still noticeable. In the laptop tests, the time difference resulted in the Windows 11 test having 25.66% higher energy consumption. The difference between the two C++ test results is a direct result of the different complexity of Boyer Moore and Dijkstra. Boyer Moore has an average time complexity of $O(n)$(where n is the length of input text), while Dijkstra, on the other hand, has an average time complexity of $O(ElogV)$(where V is the number of nodes and E is the number of

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	19.82	1421.24	15.2	0	70	1.371886	0.000381
Linux desktop	23.378	1152	3.47	0	68	1.555193	0.000432
Windows 11	71.8	945.93	10.4	0	120.12	1.374795	0.000382
Linux laptop	63.96	1177.04	2.15	0	109.92	1.107652	0.000308

Table 5.6: Pattern search algorithm in C++

edges). Higher complexity means that more calculations are performed during each cycle, which leads to a higher CPU utilization and highlights more clearly the impact of the various OSs on the energy consumption.

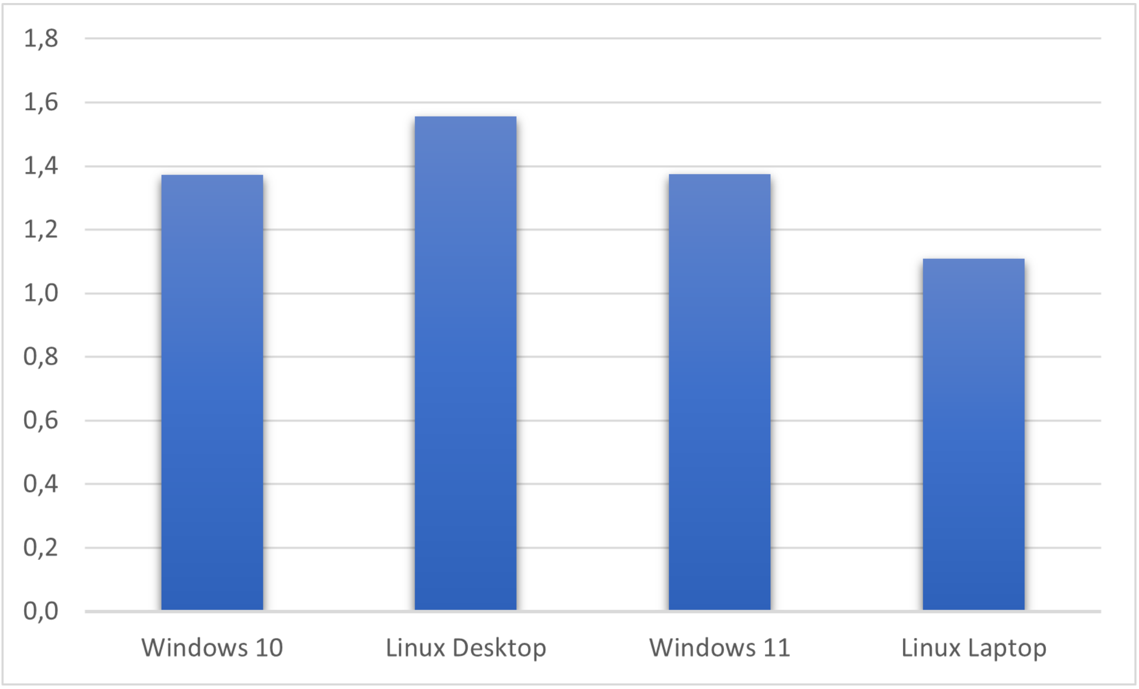


Figure 5.7: Energy consumption of Pattern search in C++

5.7 Linear regression in Python

This test also required a large amount of memory, which meant that the test running in Windows 11 was noticeably slower than the other tests. This test had a peak memory usage of around 7 GB, which was not a problem on the desktop PC since it has 16 GB of memory. Linux on the laptop also ran the test without any problems since the Linux OS occupied a small amount of memory. Windows in general consume much more memory than Linux because of it being a heavier OS with a large number of services running in the background. This caused the laptop's memory usage to reach 95% several times during the test in Windows 11 resulting in the Windows' memory compression feature to be used multiple times.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	11.04	4085.31	0	0	51	0.619407	0.000172
Linux desktop	8.32	3989.76	0	0	53.901	0.512147	0.000142
Windows 11	23.21	2058.05	0	0	104	0.444428	0.000123
Linux laptop	21.88	4756.8	0	0	78.88	0.402750	0.000112

Table 5.7: Linear Regression in Python

Thanks to this feature, the test could be performed and completed with less memory than necessary. However, this had a large impact on the overall performance of the test. The memory compression resulted in slower overall performance, which in turn increased the power consumption. Figure 5.8 shows the memory usage of this test in Windows 11. As can be seen, the memory compression feature of Windows was used twice during the test. The first time was relatively early in the test and the second time was in the middle of the test. The test's memory usage exceeded 4 GB before the memory compression feature kicked in. This means that any program or algorithm with similar memory requirements will perform worst in Windows 11 compared to running in Linux on this laptop. The overall higher energy consumption of the desktop PC tests is caused by the higher energy consumption of its hardware.

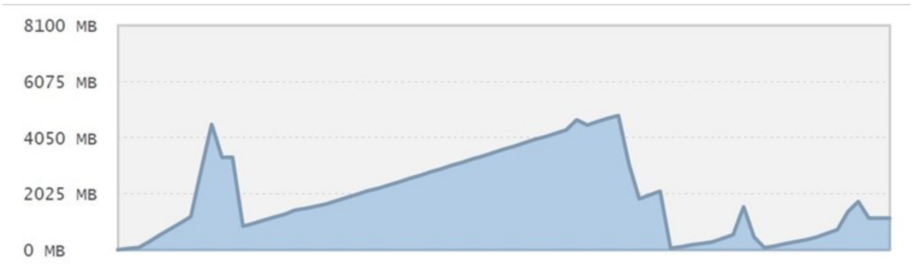


Figure 5.8: Memory usage of Linear Regression in Python in Windows 11

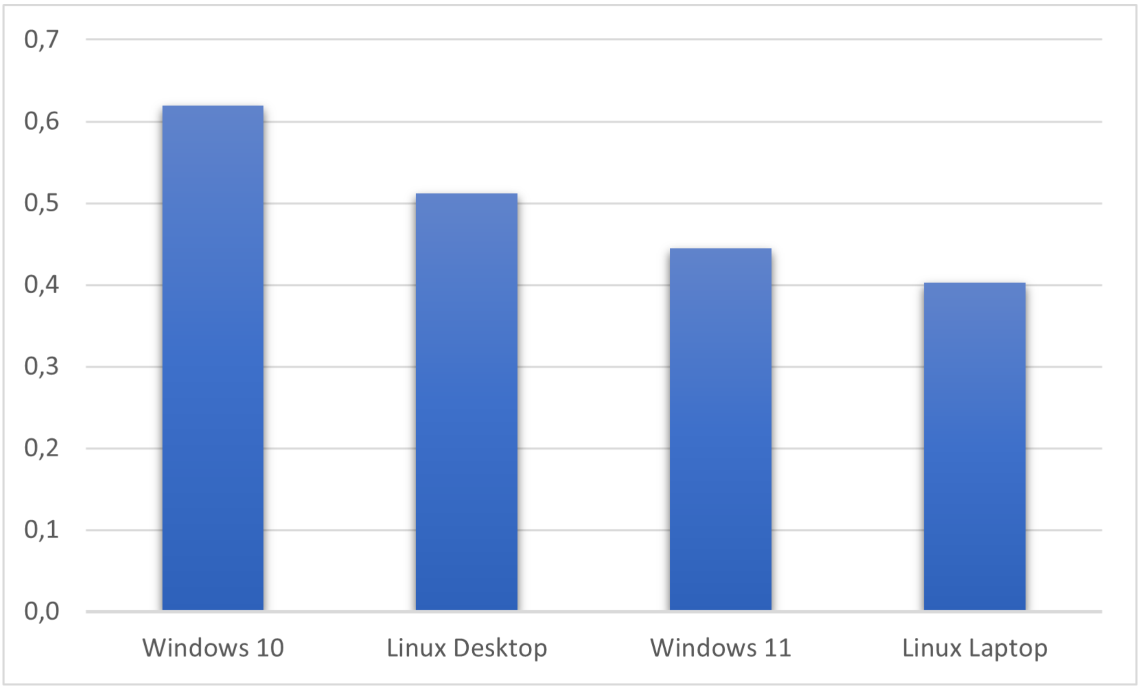


Figure 5.9: Energy consumption of Linear Regression in Python

5.8 VeraCrypt

This test was performed using the benchmark feature with the largest possible buffer size of 1 GB. Since the test used the memory for the buffer, the storage was not used in this test, which can be seen in Table 5.8, as no storage reads or writes were recorded from VeraCrypt during the tests. In addition, Table 5.8 shows that each test used about 1 GB of memory, which is to be expected given the buffer size. the CPU usage was very similar regardless of the operating system, while the time required to complete the test increased by 31.25% on the desktop PC test when Linux was used instead of Windows 10. This directly increased power consumption by 30.42%, as time was the only difference between the tests. Similar results were observed in the laptop tests, where Linux was 28.81% slower than Windows 11, which is consistent with the desktop PC results. Since the storage was not used in this test, which is the only hardware difference between the different OSs in each computer, all components involved in this test were identical. This leaves only the OS as the difference between the various tests, leading to the conclusion that the performance difference shown here is attributed to the influence of the OS. These results mean that using Windows instead of Linux for encrypting or decrypting large files or working with different encryption algorithms will yield faster results and reduce the power consumption required for these operations.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	92.15	1010.77	0	0	32	2.837481	0.000788
Linux desktop	91.55	1056	0	0	42	3.700626	0.001028
Windows 11	90.69	988.66	0	0	110	1.540005	0.000428
Linux laptop	90.52	1109.6	0	0	141	1.976894	0.000549

Table 5.8: VeraCrypt resources usage

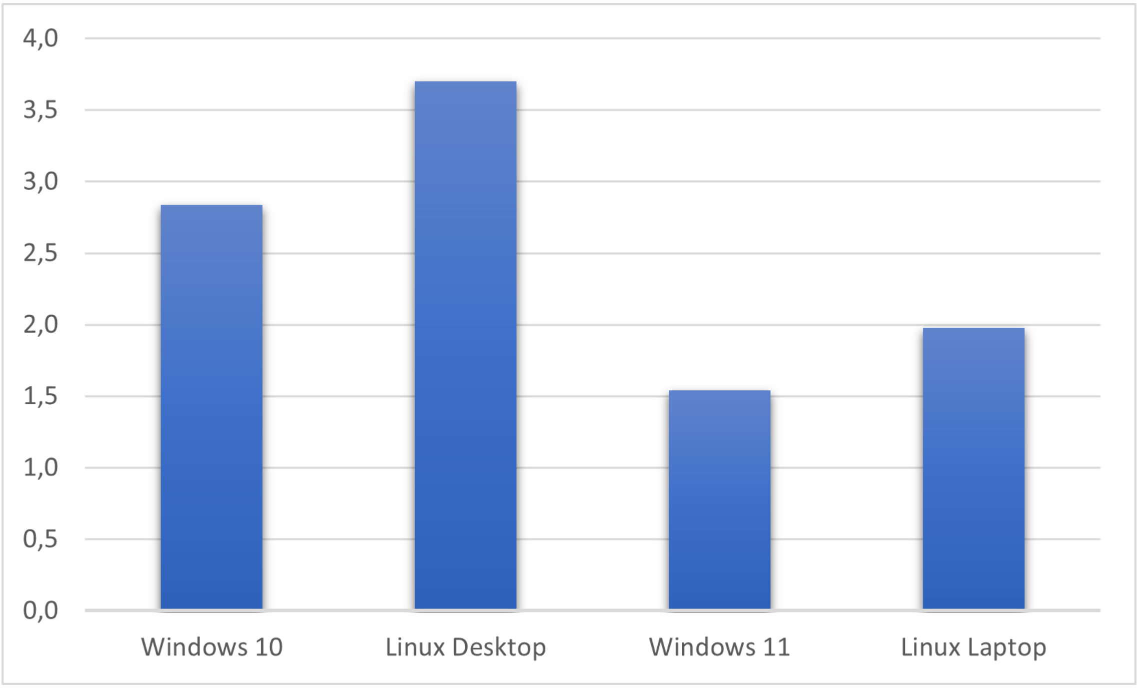


Figure 5.10: Energy consumption of VeraCrypt

5.9 Lightworks

This test and the Blender animation rendering test are the only tests that involve the GPU in the testing process. Windows 10 and Windows 11 had higher average CPU utilization compared to Linux running on the same hardware. Memory usage barely varied across the four tests and the average GPU usage was also very similar. The first noticeable difference was the average write speed, which was almost identical for Windows 11 and Linux on the laptop, while Windows 10 had a 50% higher average than Linux on the desktop PC. Although the 50% difference is quite large, it did not have a significant impact on the overall energy consumption because the energy consumption of the storage devices used is relatively low. The biggest influencing factor, as in the previous tests, was the time needed to finish the tests. This was only noticeable in the desktop PC tests, as the laptop tests concluded within a difference of 2 seconds, so the higher CPU utilization of Windows 11 was the reason for its higher power consumption compared to the Linux test. However, on the desktop PC, the test performed under Linux was 38.38% slower than the one performed under Windows 10. This considerable difference resulted in a significantly higher power consumption, even though Windows 10 had a 20% higher average CPU utilization.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	76.45	1703.65	9.21	25	99	9.781052	0.002717
Linux desktop	56.32	2076.8	6.07	33	137	12.005467	0.003335
Windows 11	97.89	1574.84	2.39	29	342	8.016198	0.002227
Linux laptop	83.67	1717.2	2.41	29	346	7.373741	0.002048

Table 5.9: Lightworks resources usage

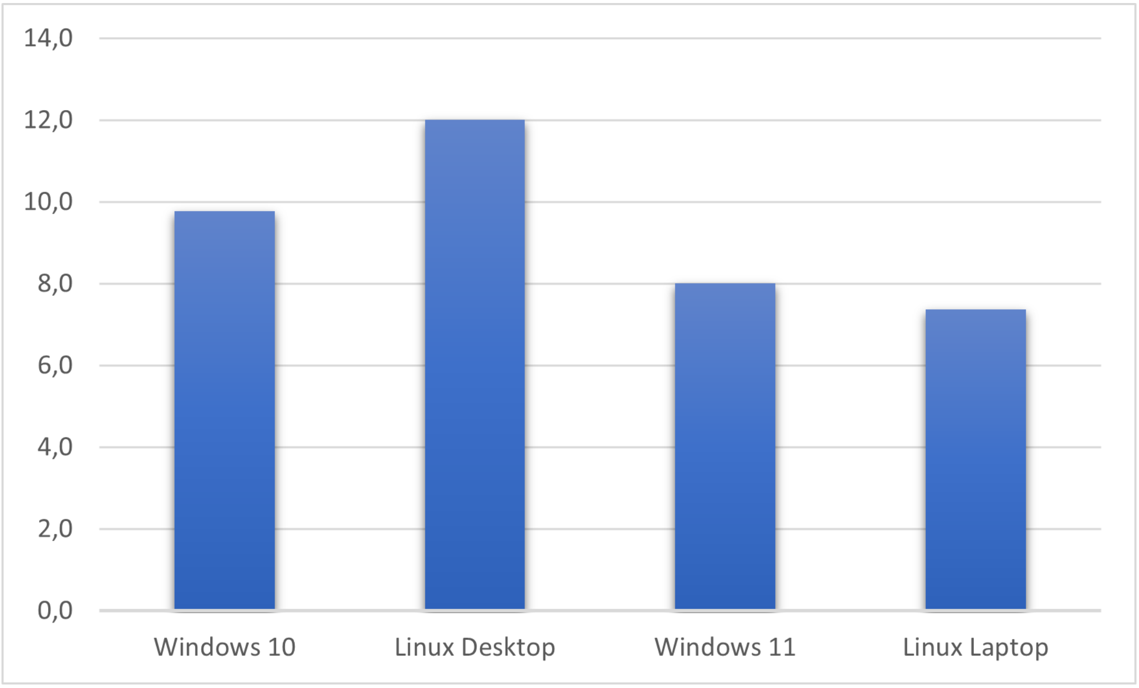


Figure 5.11: Energy consumption of Lightworks

5.10 Blender image

The Blender rendering test focuses on the CPU and neglects the GPU. This can be clearly seen in Table 5.10 with 0% GPU usage in all tests. This was intentionally done because the next test, which is also a Blender rendering test, uses the GPU. The CPU and memory usage did not change significantly across the different OSs, but the time required for rendering did. Windows 10 was 10% slower than Linux while Windows 11 was 20% slower than Linux. This slower performance of Windows 10 and Windows 11 resulted in 13.3% and 25.85% higher power consumption, respectively. [34] tested the performance difference between Linux and Windows in CPU specific rendering using 12 different CPUs. Their results are similar to the result shown in Table 5.10 and Figure 5.12, as Linux was faster than Windows in every test case in the BMW CPU benchmark and the Classroom CPU benchmark which are two benchmarks from the blender official website. Linux was also faster than Windows in 10 of the total 12 test cases of the Chaos Group V-Ray benchmark. Although they only presented the time result and neglected the average resource usage, according to my tests the average CPU and memory usage of Linux and Windows are very similar, regardless of the hardware used. This leaves the time as the only significant factor influencing the power consumption, leading to the conclusion that it is faster and more efficient to use Linux instead of Windows when rendering with Blender and using only the CPU.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	93.84	3299.32	0	0	55	5.014066	0.001393
Linux desktop	90.27	3668.8	0	0	50	4.413663	0.001226
Windows 11	98.19	2918.32	0	0	192	3.046036	0.000846
Linux laptop	92.02	3209.6	0	0	159.98	2.407551	0.000669

Table 5.10: Blender image resources usage

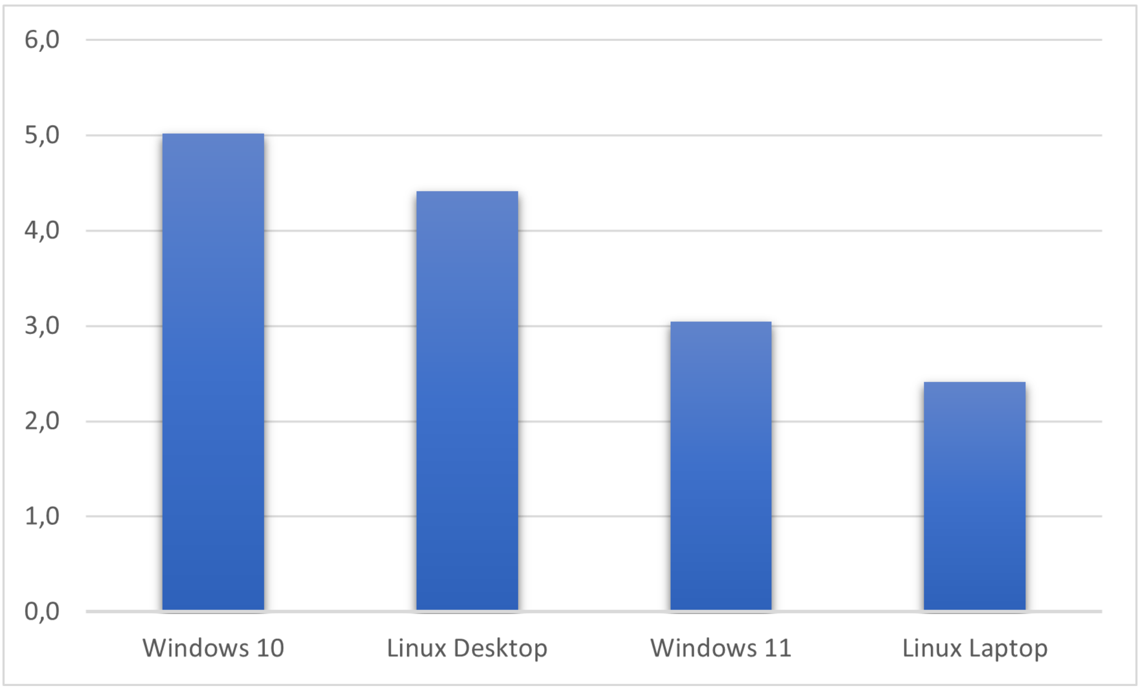


Figure 5.12: Energy consumption of Blender image rendering

5.11 Blender animation

As mentioned previously, this Blender rendering test does not focus on the CPU, but on the GPU. Eevee was chosen as the render engine and CUDA as the Cycles render device for the tests. The results in Table 5.11 and Figure 5.13 show an almost identical performance on the devices regardless of the OS used. In terms of power consumption, the biggest difference between Windows and Linux was about 5.29%. As Table 5.11 shows, the average GPU usage was not very high and it seems that the tests did not fully utilize the GPUs. Figure 5.14 shows a one minute snapshot of the GPUs utilization during the tests. It shows that GPU utilization occurred in bursts, reaching 60-70% on the desktop PC and 100% on the laptop. These spikes in utilization occurred during the creation of the 128 render samples of each frame. Render samples are the “Number of paths to trace for each

pixel in the final render. As more samples are taken, the solution becomes less noisy and more accurate"[28].
Since the GPUs were used for most calculations in this test, the difference between Windows and Linux shrank to almost inconspicuous values. The only difference in the results was related to the different hardware used for testing. The newer, faster and more efficient hardware used in the desktop PC resulted in four times faster rendering time while consuming almost half the power compared to the laptop's older hardware.

Operating system	CPU (%)	RAM (MB)	W/R (MB/s)	GPU (%)	Time (s)	kWs	kWh
Windows 10	26.47	1690.3	4.81	30.48	2972	167.455072	0.046515
Linux desktop	24.17	2568	2.09	27	3355	171.155388	0.047543
Windows 11	39.21	1610.2	1.2	59	12136	278.493269	0.077359
Linux laptop	38.22	2644	0.59	56	11847	264.556191	0.073488

Table 5.11: Blender animation resources usage

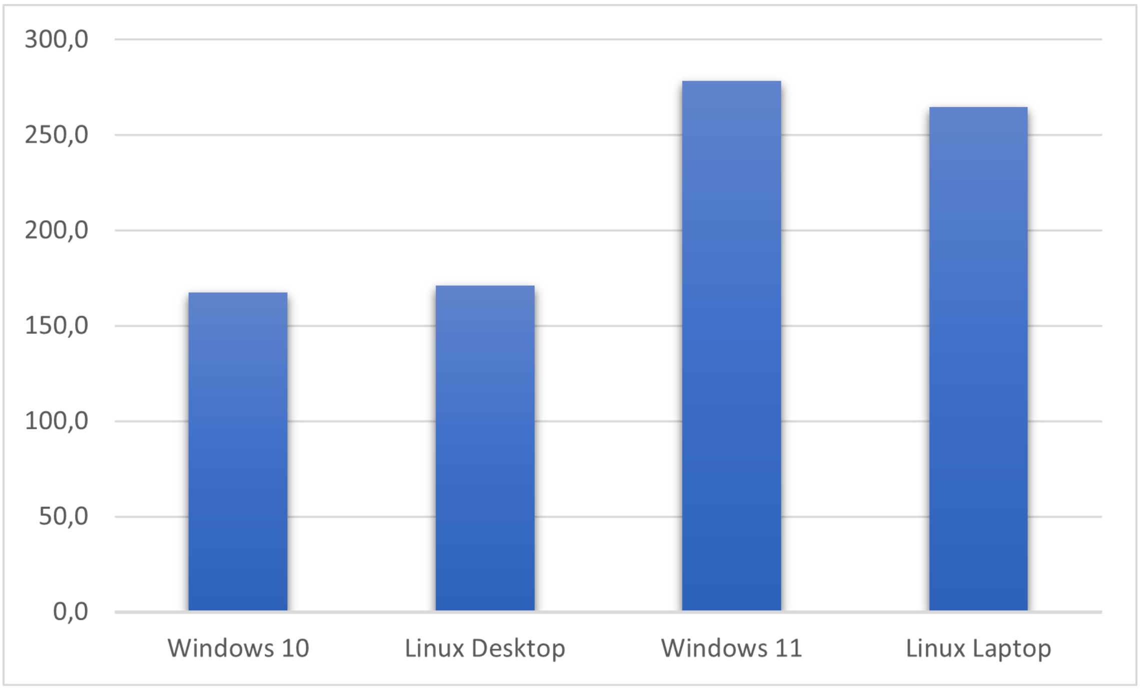


Figure 5.13: Energy consumption of Blender animation rendering

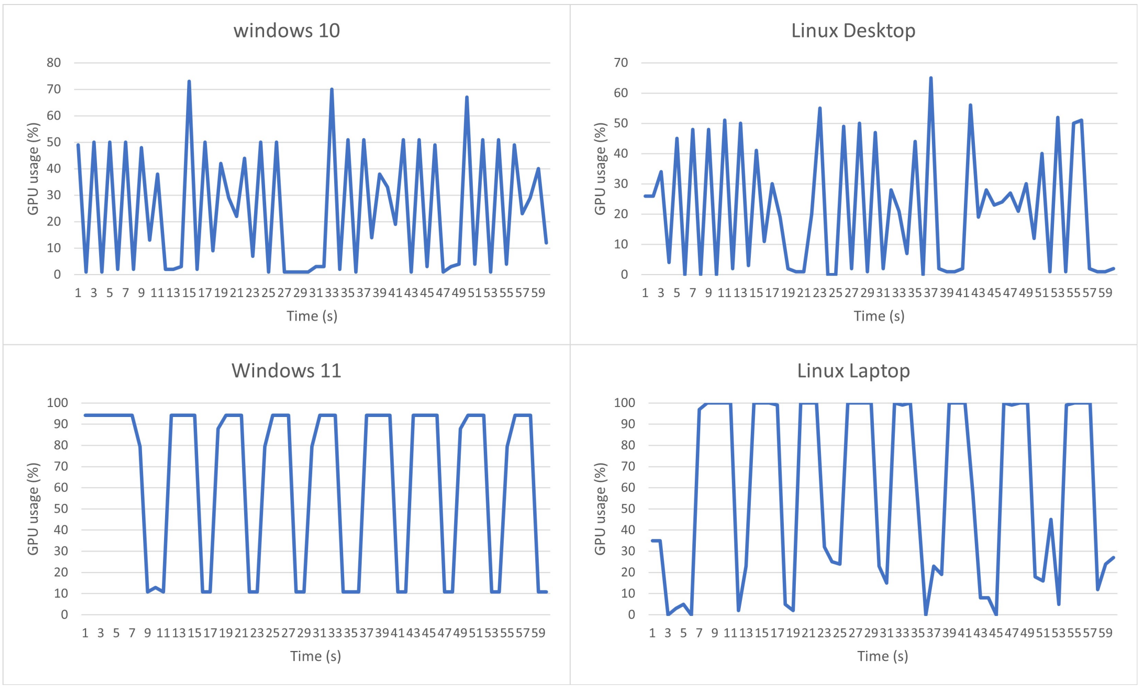


Figure 5.14: One minute snapshot of the GPUs usage

5.12 Combined results

After presenting the results of each conducted test individually, this section combines them to provide an overall picture of the performance of each OS.
Figure 5.15 shows the total energy consumption of all tested algorithm side by side. Dijkstra's Java and Python tests consumed significantly more energy on the Linux desktop PC as previously mentioned in the corresponding sections in this chapter. Except in these two cases, the difference between the various OSs is still noticeable, but not as significant. This can be better seen in Figure 5.16, which shows the combined energy consumption of all algorithm tests performed, sorted by the different OSs. The Linux desktop PC tests consumed 88.18% more energy than the tests performed on the same system using Windows 10. Whereas the tests conducted under Windows 11 consumed only 10.61% more energy than the Linux tests on the same hardware. The difference in energy consumption between Windows 10 and Windows 11 was 9.04%, compared to 90.89% between the two Linux systems.
Figure 5.17 illustrates the energy consumption of the individual software tests performed using the different OSs. Figure 5.17 had to be scaled logarithmically because of the vast difference in energy consumption between the Blender animation test and the rest of the software tests. On the desktop, Windows 10 had lower power consumption than Linux in every software tested, while on the laptop, Windows 11 had lower power consumption than Linux only in the VeraCrypt encryption test. Figure 5.18 illustrates the total energy consumption of the software tests of each OS. As expected, the results are almost identical to those of the Blender animation test, as this test took the longest and consumed the most energy, making its results the dominant among the various software tested.
Figure 5.19 shows the total energy consumption of all conducted tests. This provides an overview of the consumption of the various OSs. In the desktop's case, using Windows 10 instead of Linux

resulted in 6.64% less power consumption. For the laptop, it was found that Linux consumed 5.49% less power than Windows 11. The difference in the laptop's energy consumption is strongly related to the slower performance of Windows 11 in the Blender animation test. On the desktop, the Linux OS's significantly higher power consumption in the algorithm tests, especially in Dijkstra's Java and Python implementations, in addition to its slower performance in the Blender animation test, led to most of the performance differences between the two OSs.

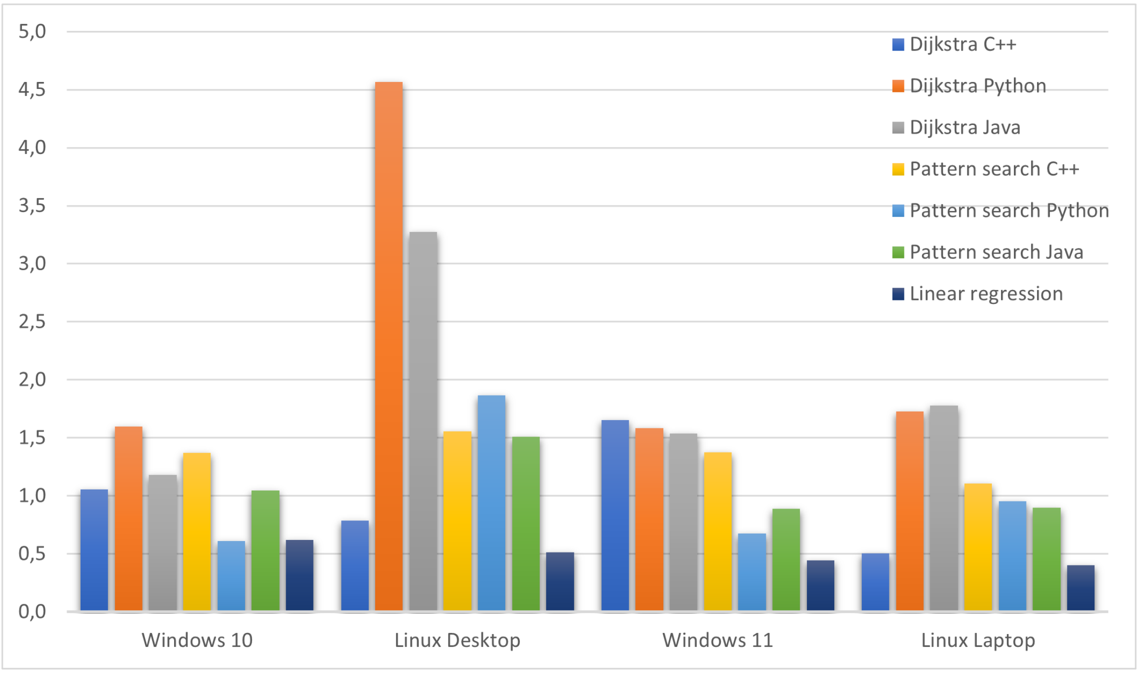


Figure 5.15: The energy consumption of all the algorithms tested

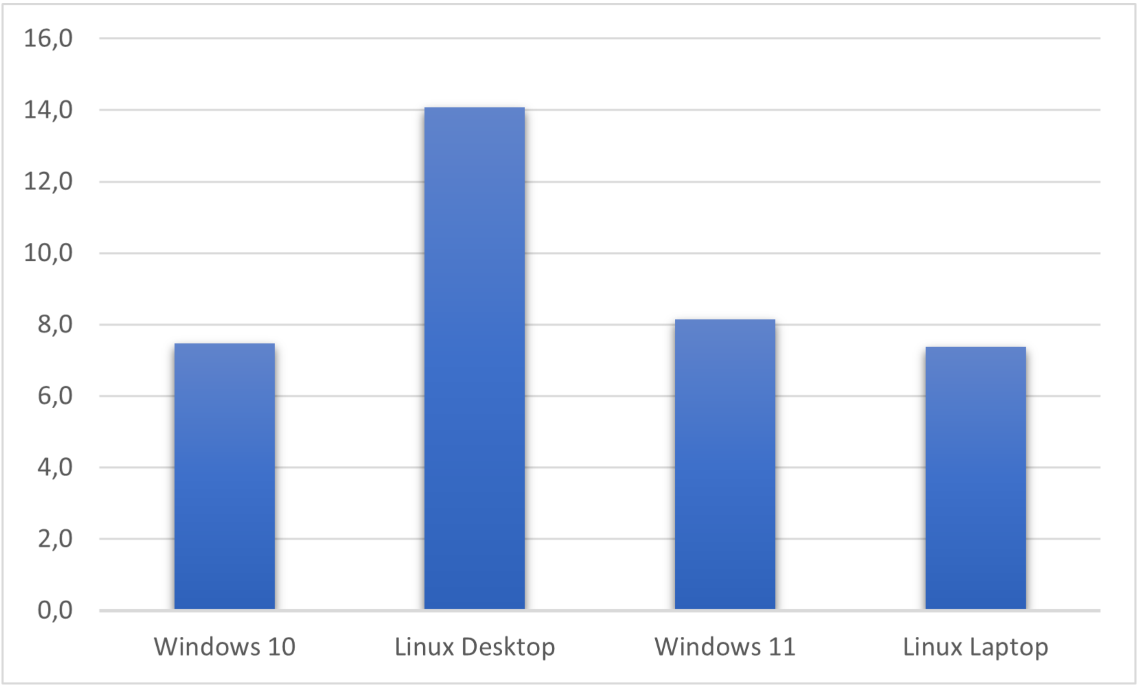


Figure 5.16: The total energy consumption of the algorithms per OS

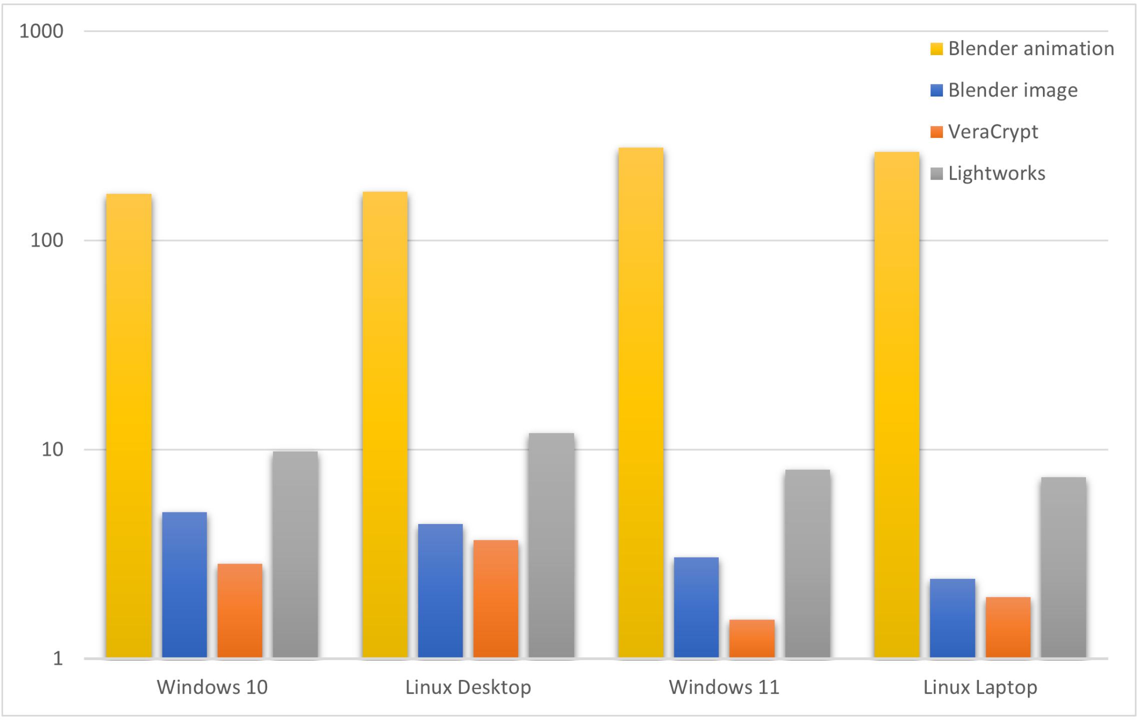


Figure 5.17: The energy consumption of all the software tested

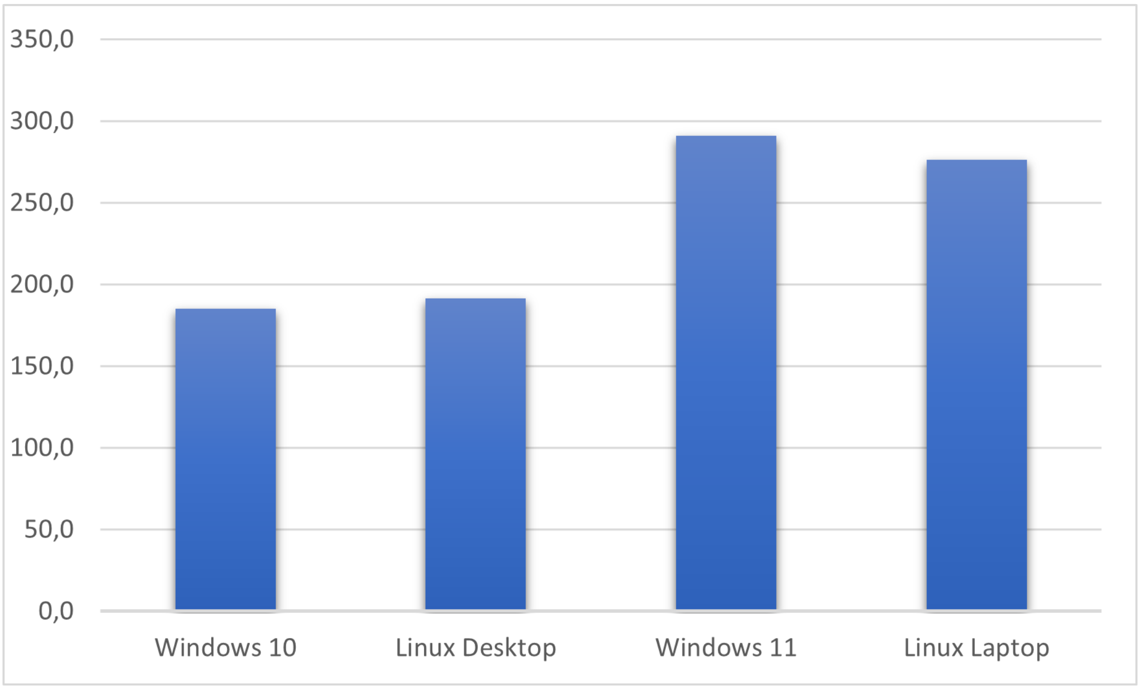


Figure 5.18: The total energy consumption of the software per OS

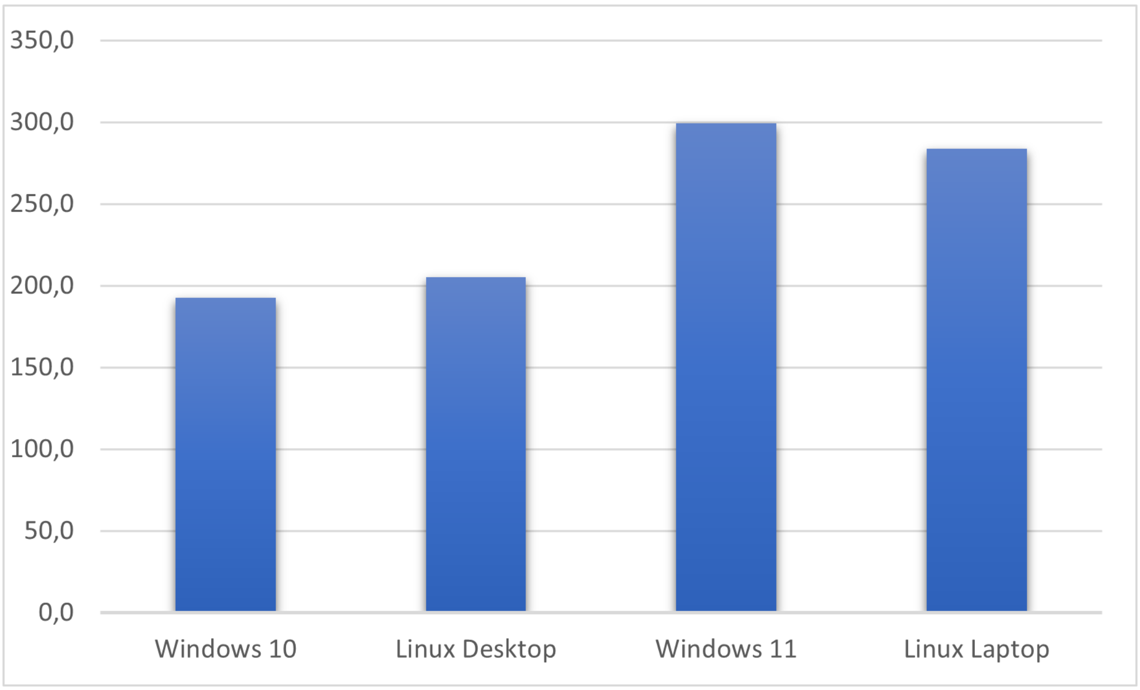


Figure 5.19: The total energy consumption of all the tests per OS

6 Conclusion and Outlook

The differences in performance between C++ and the other two languages tested, Python and Java, is that C++ is a compiled language, while Python and Java are interpreted languages. An interpreted language like java is compiled down to bytecode by the Java compiler. The bytecode is then executed by a Java Virtual Machine (JVM). Modern JVMs use a technique called Just in Time (JIT) compilation [17] to compile the bytecode to native instructions understood by the CPU on the fly at runtime. The CPU can only understand machine code. For interpreted programs, the ultimate goal of an interpreter is to interpret the program code into machine code. However, usually a modern interpreted language does not interpret human code directly because it is too inefficient. The Python interpreter first reads the human code and optimizes it to some intermediate code before interpreting it into machine code. Therefore, another program is always required to execute a Python script, unlike in C++ where the compiled executable of the code can be executed directly.
This explains why programs written in C++ tends to perform faster than similar programs written in Java or Python. This behavior however can be changed if complex libraries are imported and used in the compiled code. As for the performance difference between the OSs, it is most likely related to the different kernel that Windows and Linux use.
Windows uses the micro-kernel which “provides low-level address space management as well as Inter Process Communication (IPC). Furthermore, OS functions like the virtual memory manager, file system, and CPU scheduler are built on top of the microkernel”[11]. Linux uses the monolithic kernel which offers “CPU scheduling, device management, file management, memory management, process management, and other OS services via the system calls”[11]. Unlike the microkernel, in which Every service has its address space to make them secure, user and kernel services in the monolithic kernel run in the same address space. This means that “the execution of the microkernel is slower because communication between the system’s application and hardware is established by message passing”[11]. The execution of the monolithic kernel is faster because “the system call establishes the communication of the system’s application and hardware”[11]. This can explain why the C++ tests performed faster and more efficient under Linux. The reason for the Linux desktop PC test to consume more power than windows 10 is most likely related to the huge number of results which had to be printed on the VS code console (8 million lines in total).
This thesis has shown the important impact of the operating system on the energy consumption of the various software running on it. Ultimately, I hope that this work serves as a small impetus for further complex research not only on energy consumption, but also on the overall performance impact of operating systems on all types of software and hardware, including large data centers.

Bibliography

[1] Y. Agarwal, S. Savage, R. Gupta. “SleepServer: A Software-Only Approach for Reducing the Energy Consumption of PCs within Enterprise Environments”. In: (2010). URL: https://www.cs.princeton.edu/courses/archive/fall10/cos597B/papers/sleepservers-usenix10.pdf (cit. on p. 15).

[2] “AMD 25x20 Energy Efficiency Initiative”. In: (2020). URL: https://www.amd.com/en/technologies/25x20 (cit. on p. 18).

[3] N. Amsel, B. Tomlinson. “Green Tracker: A Tool for Estimating the Energy Consumption of Software”. In: (2010). URL: http://dmrussell.net/CHI2010/docs/p3337.pdf (cit. on p. 15).

[4] *Anaconda documentation*. URL: https://docs.anaconda.com/anaconda/install/index.html (cit. on p. 22).

[5] C. Angelini, I. Wallossek. “Intel Core i7-5960X, -5930K And -5820K CPU Review: Haswell-E Rises”. In: (2014). URL: https://www.tomshardware.com/reviews/intel-core-i7-5960x-haswell-e-cpu,3918-13.html (cit. on p. 25).

[6] Blender. URL: https://www.blender.org/about/ (cit. on p. 22).

[7] T. Carc¸ao. “Measuring and Visualizing Energy Consumption within Software Cod”. In: (2014). URL: https://www4.di.uminho.pt/~jas/Research/Papers/vl-hcc2014-proceedings IEEE.pdf (cit. on p. 15).

[8] A. Carroll, G. Heiser. “An Analysis of Power Consumption in a Smartphone”. In: (2010). URL: https://www.researchgate.net/publication/215697458_An_Analysis_of_Power_Consumption_in_a_Smartphone (cit. on p. 20).

[9] P. Ciancarini, S. Ergasheva, Z. Kholmatova, A. Kruglov, G. Succi, X. Vasquez, E. Zuev. “Analysis of Energy Consumption of Software Development Process Entities”. In: (2020). URL: https://www.researchgate.net/publication/346218191_Analysis_of_Energy_Consumption_of_Software_Development_Process_Entities (cit. on p. 15).

[10] crucial. URL: https://www.ediatlanta.com/images/crucial-bx500-ssd-datasheet.pdf (cit. on p. 25).

[11] *Difference between Microkernel and Monolithic Kernel*. URL: https://www.javatpoint.com/microkernel-vs-monolithic-kernel (cit. on p. 47).

[12] digchip. URL: https://www.digchip.com/datasheets/parts/datasheet/2/202/HFM256GDJTNG-8310A.php (cit. on p. 25).

[13] M. Fourment, M. R. Gillings. “A comparison of common programming languages used in bioinformatics”. In: (2008). URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-82 (cit. on p. 19).

[14] geeksforgeeks. URL: https://www.geeksforgeeks.org/about/?ref=footer (cit. on pp. 26, 27).

[15] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. Campbell, S. Romansky. "GreenMiner: A Hardware Based Mining Software Repositories Software Energy Consumption Framework". In: (2014). URL: https://dl.acm.org/doi/abs/10.1145/2597073.2597097 (cit. on p. 15).

[16] T. Johann, M. Dick, S. Naumann, E. Kern. "How to measure energy-efficiency of software: Metrics and measurement results". In: (2012). URL: https://www.researchgate.net/publication/254040409_How_to_measure_energy-efficiency_of_software_Metrics_and_measurement_results (cit. on p. 15).

[17] "Just in Time Compilation Explained". In: (2020). URL: https://www.freecodecamp.org/news/just-in-time-compilation-explained/ (cit. on p. 47).

[18] J. Kim, D. Rotem. "Energy proportionality for disk storage using replication". In: (2011). URL: https://dl.acm.org/doi/10.1145/1951365.1951378 (cit. on p. 18).

[19] M. Larabel. "A Look At The Windows 10 vs. Linux Power Consumption On A Dell XPS 13 Laptop". In: (2018). URL: https://www.phoronix.com/scan.php?page=news_item&px=Windows-Linux-Power-Dell-XPS (cit. on p. 19).

[20] Lightworks. URL: https://lwks.com/ (cit. on p. 23).

[21] *managing power consumption with powertop*. URL: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/monitoring_and_managing_system_status_and_performance/managing-power-consumption-with-powertop_monitoring-and-managing-system-status-and-performance (cit. on p. 18).

[22] *MSI afterburner*. URL: https://www.msi.com/Landing/afterburner/graphics-cards (cit. on p. 23).

[23] *NVIDIA System Management Interface*. URL: https://developer.nvidia.com/nvidia-system-management-interface#:~:text=The%20NVIDIA%20System%20Management%20Interface,monitoring%20of%20NVIDIA%20GPU%20devices (cit. on p. 24).

[24] Pixabay. URL: https://pixabay.com/videos/ (cit. on p. 27).

[25] S. Ratka, F. Boshell. "The nexus between data centres, efficiency and renewables: a role model for the energy transition". In: (2020). URL: https://energypost.eu/the-nexus-between-data-centres-efficiency-and-renewables-a-role-model-for-the-energy-transition/ (cit. on p. 13).

[26] V. D. Reddy, B. Setz, G. S. V. R. K. Rao, G. R. Gangadharan, M. Aiello. "Metrics for Sustainable Data Centers". In: (2017) (cit. on p. 15).

[27] C. Sahin, M. Wan, P. Tornquist, R. McKenna, Z. Pearson, W. G. J. Halfond, J. Clause. "How does code obfuscation impact energy usage?" In: (2016). URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1762 (cit. on p. 15).

[28] *sampling*. URL: https://docs.blender.org/manual/en/latest/render/cycles/render_settings/sampling.html (cit. on p. 42).

[29] V. K. Singh, K. Dutta, D. VanderMeer. "Estimating the Energy Consumption of Executing Software Processes". In: (2013). URL: https://ieeexplore.ieee.org/abstract/document/6682054 (cit. on p. 15).

[30] SysGauge. URL: https://www.sysgauge.com/ (cit. on p. 22).

[31] E. Tomes, N. Altiparmak. “A Comparative Study of HDD and SSD RAIDs’ Impact on Server Energy Consumption”. In: (2017). URL: https://www.researchgate.net/publication/320025946_A_Comparative_Study_of_HDD_and_SSD_RAIDs'_Impact_on_Server_Energy_Consumption (cit. on p. 19).

[32] VeraCrypt. URL: https://www.veracrypt.fr/en/Home.html (cit. on p. 23).

[33] *What is the difference between SDRAM, DDR1, DDR2, DDR3 and DDR4?* URL: https://www.transcend-info.com/support/faq-296 (cit. on pp. 18, 19).

[34] R. Williams. “Blender And V-Ray CPU Rendering: Linux vs. Windows”. In: (2019). URL: https://techgage.com/article/blender-and-v-ray-linux-vs-windows/ (cit. on p. 40).

All links were last followed on May 9, 2022.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

stuttgart 10.05.2022 [illegible]

place, date, signature