Analysis of Parallel Programs in Different Operating Systems

Siim-Morten Ojasalu
Supervisor: Eero Vainikko
Institute of Computer Science
Tartu University, Ülikooli 18, 50090 Tartu, Estonia
siim-morten.ojasalu@ut.ee

***Abstract*—The following paper conducts research on three of the most common operating systems on computers and their performance in parallel computing. The OS's considered are Windows, Linux, and macOS. The goal of this paper is to introduce the topic of parallel computing, bring out the differences between the operating systems like their performance differences in multiple benchmarks and find out why supercomputers use certain operating systems. Similar work seems to be missing and therefore this paper can give an initial overview of the topic. Benchmarking results however are not the only factors that determine whether it is good to use an operating system. It is almost impossible to virtualize macOS on some other hardware than Apple's and therefore it could not even be benchmarked. Windows faces issues as well and therefore performance numbers can not be a deciding factor on its own. During testing it became apparent why most supercomputers use Linux or other Linux-based operating systems - they are reliable, easy to set up, and fast. Each of these operating systems has its flaws but some of them have a lot more than others.**

I. Introduction

SUPER computers are becoming more and more advanced and the computing power increases massively each year. Browsing through all the top supercomputers, the operation systems (OS) tend to stand out. Common operating systems such as Windows or macOS are not used very widely. Even Ubuntu for example is not very common. Most operating systems seem to be custom-made for these computers. As the software plays a big role in utilizing the hardware as efficiently as possible, the operating system is a very important choice. With limited time, the possibility of modifying an OS or creating a new one might be out of reach for certain solutions. Therefore it is important to know the differences between the most popular choices and right now there seems to be a lack of such comparisons.

By statistics [2], Windows is the most used operating system by a large margin on desktop computers (overall Android is the most used). However, it is not used by supercomputers. The same goes for macOS. As it stands in the second place for the most common OS's, it takes after the top spot and is missing from the top supercomputers. As we reach the less common operating systems like Linux and the others, we also them to be more frequent among the most powerful computers. But why that is, still remains quite unclear. For the sake of better understanding, such experiments would greatly advance beginners trying to choose their working platform of choice for parallel computing and high-performance computing.

II. Background

THROUGHOUT the past two decades the performance of consumer processors has increased a lot. So far, Moore's Law is still holding up and the transistor count along with the number of cores is consistently increasing every year. To fully utilize all these cores, the software has to be developed by keeping in mind the parallel processing capabilities. Tasks that do not depend on each other can be run in parallel. Big computing tasks that can be split into smaller independent tasks are completed much quicker if they are then run on multiple cores.

However, the speedup is not quite just a multiplication by the number of cores available and put to work. As demonstrated by Amdahl's law, the speedup of parallel computing eventually hits a ceiling [1]. With a 95% parallel portion in an algorithm, the maximum speedup of the runtime is around 20 times which is achieved with around 4000 processors working in parallel. Adding even more processors or cores will not speed the algorithm up anymore. The number of processors has a much bigger effect in the lower numbers. With just 16 cores, an algorithm with a 95% parallel portion may achieve a speedup of around 10 times over a single processor.

III. Related work

THERE seem to be no similar experiments that compare the parallel computing performance in different operating systems. The topic of parallel computing is still quite evolving and still gaining importance at the common software development level. There exist different comparisons of OS's, even dating back into the previous century like a study on their differences in parallel computing [7].

The non-existence of concrete tests and benchmarks on mainstream platforms can become a problem if the goal is to gain as much performance as possible but not knowing what OS to use. Even for students who are just starting out with software engineering. Maybe certain algorithms run very slow on one set-up but are utilized completely differently on the other just because of the operating system. Therefore such a test may prove to be very useful for advancing performance with minimal cost and effort. Just making the right choice for an operating system could play a huge role.

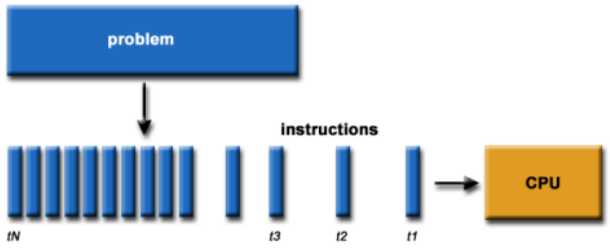

Fig. 1. Serial computing[4]

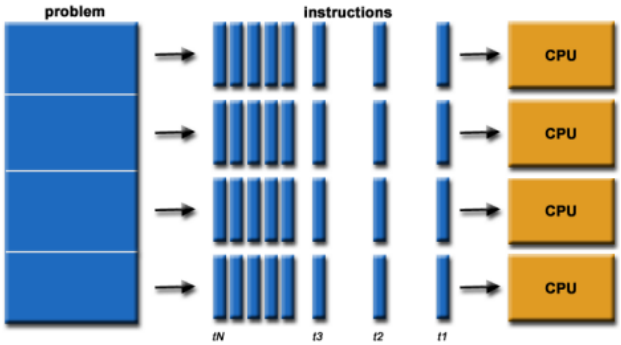

Fig. 2. Parallel computing[4]

IV. Parallel computing

Traditionally programs and software overall have been written in a way that it is separated into a series of tasks that are executed one after the other. In the presentation [4], it is described that before parallel programming there was serial computation which basically involved four parts. The software was meant to run on a single computer with one processing unit, a big problem had to be split into smaller steps that could be executed by the computer, these smaller steps were then executed by the computer and only one of these instructions could be processed at a time. Such a computational model can be seen in figure 1.

That meant that even though some parts of the software might not require another one's completion to be processed, it still needed to do every step independently. However, with parallel computing, a heavy computational task that is split into many smaller instructions could be executed faster by processing certain tasks in parallel or in other words, at the same time. Central processing units have their own control units, which direct the work and instructions needed to be completed to the multiple cores available. B. Barney explains it in a similar manner in his presentation by comparing it to serial computing. Before the software was run on a single processing unit then parallel computing uses multiple processing units. The overall problem is splitting the parts in a way, that they can be computed concurrently. The splits that are obtained from the original problem are then broken down into discrete instructions and are executed on different processing units. Just doing one calculation at a time was not a limit anymore as can be seen in figure 2.

A. Benefits

As brought out in the presentation [4], there are multiple benefits to using parallel computing. It helps save time by completing tasks faster. It also allows computers to tackle more complex and larger problems without taking an unreasonable amount of time to calculate. And lastly as the main benefit, parallel computing removes the inefficiency by executing independent parts concurrently.

However, these are not all the benefits. For example, a processor's clock speed determines how fast it can perform calculations or execute instructions. Nowadays processors have reached the 5 GHz mark and even higher. This means that one core is able to make 5,000,000,000 calculations per second. Although increasing the clock speeds is slowly hitting its limit. That is due to the thermal limits because by increasing the speeds, the processor requires more power. The voltage has to be increased exponentially and therefore the processing unit gets hotter until it reaches the point of thermal throttling without sufficient cooling. Liquid nitrogen is used to cool overclocked processors that have reached clock speeds of 8 GHz and higher however such a cooling mechanism is not sustainable, especially in consumer and everyday computers. Therefore it is not just possible to make processors faster by having them calculate more instructions per second due to thermal limits. However, with parallel computing, many cores can achieve the same result - more instruction executions per second.

B. Necessity

As mentioned before, the clock speeds of processors are slowly hitting the ceiling and one way to increase performance is to add more cores. However, in order to take advantage of this kind of advancement, the software has to be written to utilize all the cores as well as possible. This is where the necessity of parallel computing and the knowledge of it comes in. G. C. Fox et al. bring out in their book [5], that parallelism is the only way to achieve vastly higher speeds in computing than just with a single processor. They also explain that this concept is much cheaper than a very powerful sequential computer. Having multiple cores is like having a mini-supercomputer that would require much fewer resources than the matching serial processing computer. Tests have shown that even the fastest serial computing system does not even come close to supercomputers that utilize many processing cores.

Most processors nowadays have multiple cores and using just one of them would be disregarding the advancements of new processors. As H. Wan et al. describe in their paper [8], the importance of parallel computing should be thought about on the academic level in order to help students realize its importance and overall understand the parallel systems. They also bring out the significance of the curricula and teachers to forward all this information and knowledge to the students. The Computer Science department must update its program in order to ensure everyone gets an education that is in line with modern technology.

V. Operating systems

An operating system is the base software for a computer. It helps all the other programs run and manages their resources. A. Adekotujo et al. explain in their paper [3] that the OS controls the main computer hardware, peripherals, software, and also users. On the software side, the operating system provides services for processes. These include scheduling processor time and managing access to storage. This is especially important in parallel computing systems because the way that the resources are divided between tasks makes one OS better at parallelization than some other system. Every computer, even little consoles use operating systems. There is a big variety of them and they all have certain functionalities and features.

Every operating system has its strengths and weaknesses. For example, macOS is widely considered not to be good just for playing games. It does not support a wide variety of them and falls short in the ones it does due to its hardware restrictions. Windows is most widely used but computer science experts like software engineers might prefer Linux-based systems due to their compatibility and open source.

As this paper covers macOS, Linux-based Ubuntu, and Windows, other operating systems will not be described in detail. A. Adekotujo et al. have brought out different pros and cons in their paper [3]. For example, they bring out that Apple has designed their macOS to only be deployed on Apple computers and with the purpose of them being used by workstations or personal computers and the software that can be used is very limited. Windows, on the other hand, can be run on any modern enough hardware, is very highly compatible with all kinds of software, and also includes a media center for its intended purposes. However, it has the biggest security risks as macOS and Linux have negligible security risks. That may be due to its high popularity. Linux is said to be less user-friendly but the open-source and high compatibility and scalability make it a popular choice for servers or desktops.

Another very important use for operating systems is the cloud. Cloud computing is gaining popularity fast and therefore has to be mentioned briefly. As H. Malallah et al. describe in their paper [6], a cloud is just a category of operating systems. Its purpose is to work in a cloud computing network or virtualizations for example. It can control the different services that are running, even virtual computers or servers. The main goal is to make computing services as accessible as possible. Cloud computing provides tools, storage, and applications for businesses. That means these businesses do not need to buy hardware themselves but can run their services on an operating system provided through the network.

VI. Testing and benchmarks

In order to compare the three different operating systems and their performance in parallel computing, several benchmarks have been chosen from openbenchmarking. For further references, please see https://openbenchmarking.org/. All the tests chosen must run on all of the operating systems and utilize parallel computing in order to ensure that the results would be comparable. Testing the systems with different tasks would yield no decisive outcome. However, due to complications with macOS and the restrictions for its use on hardware levels, multiple testing setups were required to take into use. The main problem with testing macOS is that it requires Apple's own hardware and will not run on just any machine by virtualization. Some tricks can be used to get past this problem but overall it does not allow such behavior.

A. Testing setups

Hyper-V in Windows was used in order to virtualize Windows and Linux systems. For further references see https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows. MacOS can not be virtualized with Hyper-V and therefore it is not compared in this test. The base computer is running Microsoft Windows 10 Pro with the following hardware:

- AMD Ryzen 5 3600 6-core 12-thread processor
- MAG B550M Mortar motherboard
- 32GB of total physical memory

Each of the virtual systems were allocated with:

- 2/4 processor cores
- 4GB of physical memory

Each virtual system had a basic installation of the operating system and only the necessary installations for running the test suites. This includes the main program which is the phoronix-test-suite. More information can be found on https://www.phoronix-test-suite.com/. After acquiring the program, each of the benchmarks could easily be run with it, for example in Ubuntu with the command "**phoronix-test-suite benchmark compress-7zip**". In order to avoid any sort of thermal-throttling issues and therefore causing performance drops, the indicators of the physical system were monitored and none of the temperatures reached critical stages - the processor for example never exceeded 75° Celsius.

For the second setup, a MacBook Pro with a base operating system of macOS was used. With a program called Parallels Desktop it is possible to virtualize Windows, Linux, and macOS itself on an Apple computer and therefore provide another case of benchmark results, just on a different setup:

- MacBook Pro (13-inch, M1, 2020)
- 16GB of total physical memory
- macOS Monterey 12.3

B. Complications

The biggest problems during testing were macOS and the new Apple M1 chip itself. Apple has restricted the use of macOS to only their own hardware. Therefore it is not possible to virtualize macOS on regular PC hardware with Hyper-V for example. With certain system modifications and config changes in VirtualBox, it is possible to launch macOS, however, the level of virtualization in VirtualBox does not allow for a fair comparison between the operating systems and therefore it was not used as a testing setup for this paper.

As for virtualization on macOS itself, the computer used for testing had the new Apple's own M1 chip in it with arm64 architecture. That means that the software is not quite stable

for it yet and many difficulties were experienced. Virtualizing Ubuntu was simple and everything worked great. Virtualizing Windows 11 and macOS Monterey 12.3 itself worked as well but phoronix-test-suite did not work on these environments. For macOS, there was an error in the PHP code on the testing program side and nothing could be done to resolve it. Editing the code and trying to debug did not yield any results and therefore it was not possible to run phoronix-test-suite on virtualized macOS. Similar problems were encountered with the Windows 11 virtualization. Installing phoronix-test-suite worked fine and everything seemed to be correct until benchmarking. Trying to launch a benchmark threw errors that installed tests were not found although they were installed properly. The path in which phoronix-test-suite was looking for the tests was incorrect and no way to change it was found. Therefore the Windows 11 environment on macOS could not be used for benchmarking either.

C. Benchmarking the systems

Due to multiple complications, software incompatibilities, and lack of time to seriously debug PHP code, it was impossible to find a way to virtualize macOS. One possibility would just be to run the tests on the base computer. However, as the core amount and physical memory can not be controlled this way makes the comparison to other setups quite incomparable.

What is more, the fact that two different setups were used, Windows 10 and Monterey 12.3 computers, the hardware underlying these setups is different. Although the core amounts and physical memory were identical, the physical cores themselves differ from one another on the PC and MacBook Pro. Therefore the differences in results can not be considered too significant.

Disregarding all the problems that in the end did not allow much variation in testing, the benchmarks were conducted with phoronix-test-suite. It enables an easy way to run benchmarks, save results and even visualize them. Three different benchmarks were used to run on each virtualization environment with two different core amounts. Each test was run five times. So in total 90 successful benchmarks were conducted. However, many tests had failures and considering all the complications then the time spent on benchmarking was a lot higher than just running 90 successful tests.

Chosen benchmarks include Darktable, Compress-7zip, and John The Ripper. All the tests had to support arm64 architecture and therefore not just anything would suffice. Darktable had problems running on Windows 11 so therefore this benchmark was only run in Ubuntu on both the MacBook and PC. All these tests create heavy processor workloads in order to test the operating system's parallel computing ability. Darktable is a rendering benchmark and the Boat test was used. Compress-7zip is a benchmark that compresses and decompresses files. Therefore it basically includes two different benchmarks in it. Lastly, John The Ripper is a password cracking benchmark. During every run, the environment resources were monitored and the CPU always hit 100% usage so parallel computing worked in every case and the processing power was used as much as possible.

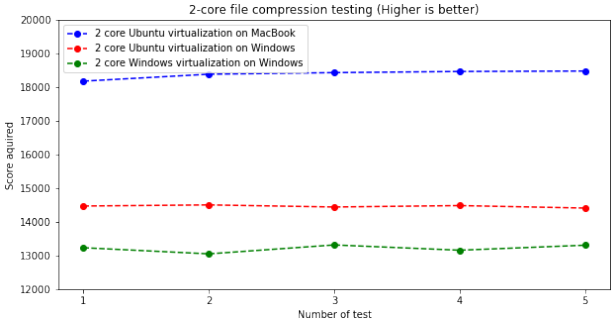

Fig. 3. 2-core compression testing

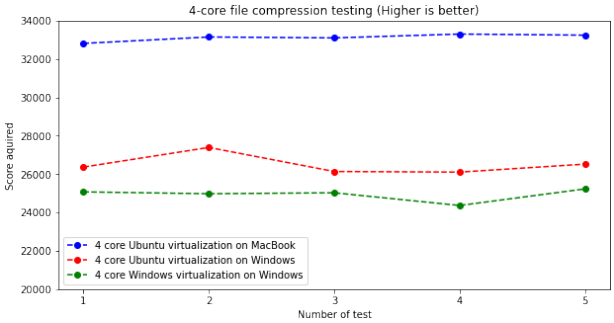

Fig. 4. 4-core compression testing

D. Results

The main goal of this paper is to figure out why supercomputers mostly use Linux and other Linux-based operating systems. From the graphs, it can be seen that the results between virtualized environments and operating systems vary based on the benchmarks. However, the numerical results are not the only ones considered in this paper. The main points are brought out in the following section.

E. Interpreting the results

From the provided graphs it can be seen that the two different Ubuntu systems provide very different results. These differences should mostly be caused by the hardware differences between the two setups. The core amounts allocated to each system were the same and the physical memory was 4GB for every virtual environment. However, the allocated cores

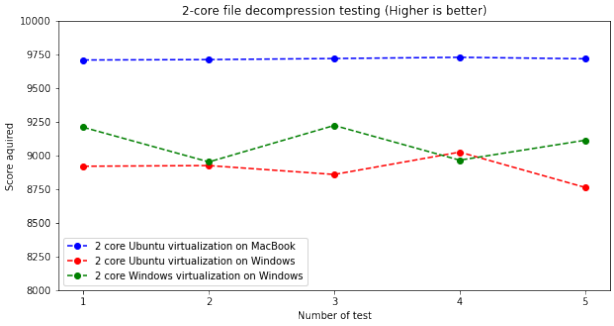

Fig. 5. 2-core decompression testing

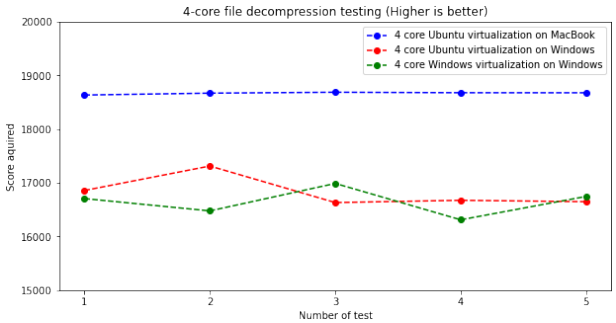

Fig. 6. 4-core decompression testing

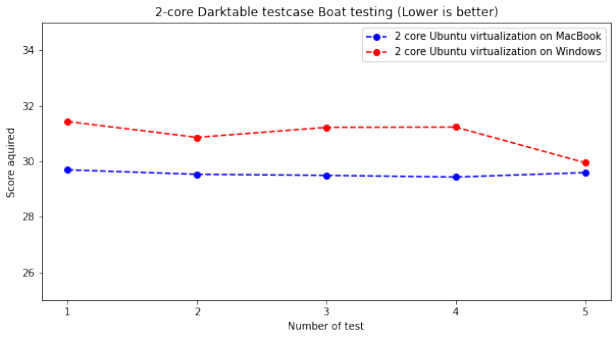

Fig. 7. 2-core Darktable testing

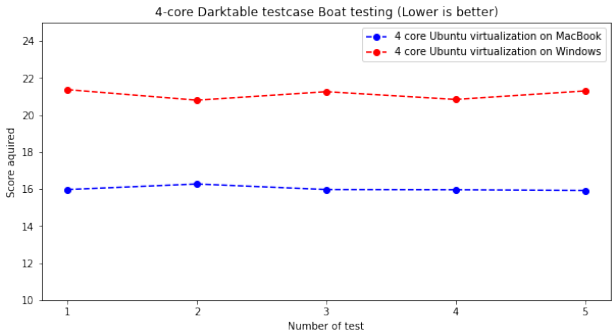

Fig. 8. 4-core Darktable testing

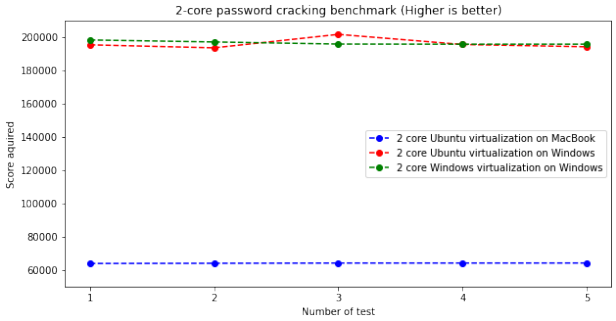

Fig. 9. 2-core password cracking testing

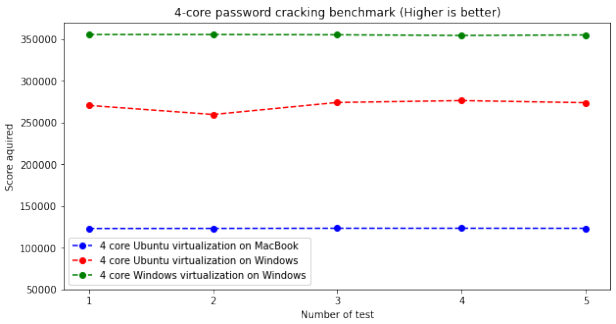

Fig. 10. 4-core password cracking testing

themselves differ and that is why the MacBook virtualization differs more than the Ubuntu and Windows one on the Windows computer.

For file compression and decompression, Ubuntu on the macOS comes ahead of the Windows-based virtualizations by quite the margin. The latter two, Ubuntu and Windows 11 on the Windows computer have similar results but the first one comes ahead by just a small amount. Decompression tests were close for the two but in the file compression tests, Ubuntu was ahead in every test. Moving on to Darktable and image processing, the gap between the macOS-based Ubuntu and Windows-based on changed as the latter won in both the 2-core and 4-core tests. Windows took its only victory in the last test - password cracking. The 2-core benchmarking saw the two Windows-based systems achieve similar results but the outcome changed when two more cores were enabled. After that, Windows 11 got a comfortable lead ahead of Ubuntu. The latter achieved especially awful results being based on macOS. Overall though, Ubuntu came ahead of Windows 11, whether it was being based on macOS or Windows, although the results did change quite a bit from that difference.

However, a performance comparison is not the only variable for choosing an operating system. Other factors include ease of use and compatibility for example. Ubuntu was definitely the easiest operating system to set up for benchmarking. A few errors occurred rarely and they were easily distinguishable and took no time to fix. Windows was usable but ran into much more problems that were hard to track and needed extra software or some missing files to work correctly. And in the end, it did not even work as virtualization on macOS. Whether that was a problem with the phoronix-test-suite software or just the virtual environment is hard to say but nevertheless, it was not as pleasant of an experience as it was with Ubuntu. The biggest revelation that has been discovered while writing this paper is why macOS is not used in supercomputers. It just can not be. Apple does not allow any other hardware to run their operating system so unless the supercomputer is owned, monitored, or for example manufactured by Apple, there is no way to get macOS properly running on it without trying to manipulate system files or changing configurations. And even on Apple's own hardware, it is very hard to get external software working correctly. The architecture (M1 chip) is new and solutions are unstable. With these reasons combined, anyone else other than Apple should not and even can not use

macOS for their supercomputers.

VII. Conclusion

OPERATING systems differ a lot from one another. Each of them has its pros and cons but considering building a supercomputer leaves one obvious choice which is a Linux-based OS like Ubuntu. MacOS has too many restrictions and problems with compatibility that it is not possible to use even if someone wanted to. Unless of course, the supercomputer is manufactured by Apple. Windows has similar issues. Although it works on most modern hardware, it runs into far more problems than Ubuntu and performs worse in most cases. The goal of this paper was to find out why mostly Linux or any custom OS based on it was used in supercomputers. The answer is very simple. It is faster in most benchmarks, is easy to set up and use, and does not have any hardware restrictions making it impossible to use in certain cases. Further testing could be conducted but overall the results should not vary by a lot. Even if macOS performs better in tests, it still can not be used with external hardware and Ubuntu or any other Linux system simply offers more than the other two operating systems. However as the software becomes more stable for both Windows and macOS, maybe new tests could prove these results wrong.

References

[1] Amdahl's law, https://en.wikipedia.org/wiki/Amdahls_law, accessed: 2022-04-03.

[2] Operating system market share worldwide, https://gs.statcounter.com/os-market-share#monthly-202204-202204-bar, accessed: 2022-04-30.

[3] A. Adekotujo, A. Odumabo, A. Adedokun, O. Aiyeniko, A comparative study of operating systems: Case of windows, unix, linux, mac, android and ios, International Journal of Computer Applications 176 (2020) 16–23.

[4] B. Barney, et al., Introduction to parallel computing, Lawrence Livermore National Laboratory 6 (13) (2010) 10.

[5] G. C. Fox, R. D. Williams, P. C. Messina, Parallel computing works!, Elsevier, 2014.

[6] H. Malallah, S. Zeebaree, R. R. Zebari, M. Sadeeq, Z. S. Ageed, I. M. Ibrahim, H. M. Yasin, K. J. Merceedi, A comprehensive study of kernel (issues and concepts) in different operating systems, Asian Journal of Research in Computer Science 8 (3) (2021) 16–31.

[7] G. J. Nutt, A parallel processor operating system comparison, IEEE Transactions on Software Engineering (6) (1977) 467–475.

[8] H. Wan, X. Gao, X. Long, B. Jiang, Introducing parallel computing concepts in computer system related courses, in: 2017 IEEE Frontiers in Education Conference (FIE), IEEE, 2017, pp. 1–7.