Physically Based Benchmarking: Comparing the Performance of Windows and GNU/Linux Using Physically Based Ray Tracing (PBRT)

David Valverde Garro
School of Computer Engineering
Costa Rica Institute of Technology

***Abstract*—The study of operating systems performance often makes use of programs that will stress its capabilities in order to obtain measures representative of real-world scenarios or extreme cases. Said programs can take the form of dedicated benchmarking tools or software meant for unrelated purposes such as database or network intrusion detection systems. The existing literature employs a multitude of methods based on the different dimensions of what constitutes performance as a whole. In this paper, we conduct an empirical comparison between Windows 10, Debian, and three Ubuntu environments using the rendering system described in Physically Based Rendering (PBRT). We also analyze the effect of virtual environments such as virtual machines and Windows Subsystem for Linux (WSL).**

Our experiment results show that Windows 10 achieves significantly slower rendering times in almost all scenarios, while Ubuntu and Debian are tied for the first place. Additionally, the collected data suggests that WSL has a performance that is competitive with its bare-metal counterpart. Virtual machine results place it between Windows and the other GNU/Linux options; however, available hardware limitations make these VM conclusions unreliable.

***Index Terms*—PBRT, OS Performance, Windows, GNU/Linux, Virtual Machine, Windows Subsystem for Linux**

I. Introduction

THE vast number of free and proprietary options for operating systems in the market means that there will always be a need to compare said systems for a similarly extensive list of purposes. Comparisons can take into account UI/UX, price, features, customization, privacy, and of course, performance. Furthermore, measurements of performance can also be categorized into many dimensions such as network, overall latency, and resource management, among many others. This paper focuses on studying a particular type of performance, namely CPU management, using physically based rendering workloads.

Microsoft's Windows 10 is currently one of the most popular operating systems, widely used in home, industrial, and academic environments. On the other hand, GNU/Linux systems tend to dominate the server market and are regarded as great alternatives. Existing research shows that the Linux kernel, responsible for core resource management tasks [1], tends to perform better than Window's kernel when dealing with heavy CPU and IO-bound loads [2] as well as on interactivity performance [3]. Additionally, efforts to compare the performance of different GNU/Linux distributions have shown that no particular distribution is better than the others in all tasks [4], which further serves to reinforce the strength of their common kernel.

With this view on performance in mind, certain interesting questions start to appear. For example, if we suppose that Windows is slower than GNU/Linux, how big is the difference? Is it negligible or does it warrant setting up dual-boot systems more often? It could also be that we are considering using virtualization options like a virtual machine (VM) or even the recent Windows Subsystem for Linux [5]. However, even more questions arise: Will performance be reduced? Should we run this heavy task natively or virtually? Is WSL a better option than a traditional VM? Also, the multiple GNU/Linux distributions and their available flavors, while providing a great deal of flexibility, serve to further increase our doubts. Is one distribution better than the other? Will similar distributions perform similarly?

II. Related Work

Several approaches have been tried to compare existing operating systems' performance. Salah et al. [2] performed a study comparing GNU/Linux and Windows Server using the Snort Network Intrusion Detection System. Said study measured the throughput and packet loss of the software under different types of loads representative of CPU and IO-bound scenarios. Fan et al. [3] approached this problem from the angle of interactivity performance, which relates to the response times of an OS when executing interactive tasks. For this purpose, they ported the GNU/Linux-only Interbench to Windows 10 in order to perform fair comparisons between the two systems. This benchmarking software allows the observation of latency and other values under different interactive tasks and background load combinations.

Comparison of operating systems is not constrained only to Windows vs GNU/Linux, for example, Bellows [4] tested four different distributions, including Ubuntu and Debian, using a Raspberry Pi 2. Their method involved running different benchmarks of the Phoronix Test Suite to measure values under three main categories: memory access, disk access, and

processor time. Similarly, Abdulganiyu [6] compared multiple versions of the Windows OS, including Windows 10. Their study, however, focused on an external hardware method using a function generator and an oscilloscope to observe the latency, jitter, and worst-case response time of an intermediary machine running one of the Windows systems.

Some approaches even extend beyond the OS itself and instead focus on the environment it runs on. One such example is the work of Felter et al. [7] which compares different virtualization options such as KVM virtual machines and Docker containers. Their study also uses Redis and MySQL database software along with SysBench to simulate heavily loaded server scenarios.

While there is a considerable amount of literature on the subject, there is no standard methodology, nor any clear consensus from the community regarding benchmarks, measurements, analysis, and other aspects. Generating a review of this particular area of research is beyond the scope of this study, however, it might prove useful in reaching a much-needed convention.

III. Experiment Proposal

We propose using the physically based ray tracer (PBRT) created by Pharr et al. [8] as a benchmark for comparing five different operating systems. Rendering with PBRT involves numerous floating-point operations to calculate the bouncing of simulated rays of light in a given scene. This system also makes use of multi-threading to accelerate the rendering process. As such, we speculate that this method can generate heavy CPU-bound loads that will stress each kernel's scheduling capabilities. To the best of our knowledge, PBRT has never been used for this particular purpose before.

A. Environments

Windows 10: As mentioned previously, Windows 10 is one of the most popular operating systems. By including an environment running this system we can compare its performance against four other GNU/Linux systems. The Windows 10 environment is also used as the host OS for the virtual machine and WSL environments.

Ubuntu: Ubuntu is a very popular GNU/Linux distribution used in both desktop and server machines. This particular environment runs natively on the hardware in a dual-boot setup and is meant to serve as a point of direct comparison with the Windows system.

Ubuntu (VM): As part of the questions presented in the introduction of this paper, we also wish to compare the performance of GNU/Linux systems running in a virtualized environment. For this purpose, we set up a Hyper-V virtual machine running the same Ubuntu version as the bare-metal environment.

Ubuntu (WSL): We also include Microsoft's Windows Subsystem for Linux to test against the native and VM environments. Ubuntu WSL is technically also a virtual machine [9], however, it is designed in such a way that this is invisible to the user. This also includes numerous optimizations to approximate the performance of Ubuntu running natively on the hardware.

Debian: Finally, we also include another popular GNU/Linux distribution to serve as a point of comparison against both Windows and Ubuntu. In doing so, we intend to study any potential advantages using one distribution over another might bring.

B. Scenes

We chose a set of four scenes to render, of all which were chosen from PBRT-V4's official scene repository [10]. Some of the scenes are offered in different versions, as such, only one of those variations was selected based on initial testing of rendering times on the Windows 10 environment. Two fast and two slow scenes were picked, where a fast scene implies a render time of fewer than 10 minutes and a fast one will take more than 10 minutes. This selection allows the observation of performance during short and long periods of CPU load. The scenes and their original filenames are as follows:

- **Clouds** - clouds.pbrt
- **Orb** - lte-orb-blue-agat-spec.pbrt
- **Killeroo** - killeroo-moving.pbrt
- **Transparent** - frame1266.pbrt

C. Filters

PBRT uses filters to blend samples near a pixel in order to determine its final value [8]. We chose to include a subset of options for this parameter to study the rendering times in a more diverse variety of scenarios. The selected filters include the box and Gaussian options. The box method is the default used by PBRT, meanwhile, the Gaussian filter is said to offer better results than the box counterpart [8].

D. Accelerators

Accelerators are data structures used by PBRT to optimize the process of checking the intersection of a ray with all the objects in a scene [8]. PBRT offers two accelerator options: bounding volume hierarchies (BVH), which construct bounding boxes around primitives and puts them in a hierarchy; and Kd-tree, which recursively divides the scene space and constructs a tree in which primitives are associated with its nodes [8]. We chose these two parameters given that they should have an impact on rendering performance and also because one of these two is always used.

E. Design

The experiment follows a factorial design in which all possible combinations of factors are tested. The filter and accelerator options have been folded into a single 2^k factor to ease the analysis process. Table I show the resulting factors and their corresponding levels. Coding of the parameters factor is as follows: **BX** - box filter, **GS** - Gaussian filter, **BV** - BVH accelerator and **KD** - KD tree accelerator.

TABLE I
FACTORS OF THE EXPERIMENT WITH THEIR CORRESPONDING LEVELS.

	Factors		
	Environment	Scene	Parameters
	Debian	Orb	BX-BV
	Ubuntu	Clouds	BX-KD
Levels	Ubuntu-VM	Killeroo	GS-BV
	Ubuntu-WSL	Transparent	GS-KD
	Windows		

TABLE II
OS AND KERNEL VERSION NUMBERS OF THE ENVIRONMENTS USED IN OUR EXPERIMENT.

	Version	
Environment	OS	Kernel
Debian	11.3	5.10.0-14
Ubuntu	22.04 LTS	5.15.0-27
Ubuntu-VM	22.04 LTS	5.15.0-30
Ubuntu-WSL	20.04 LTS	5.10.102.1
Windows 10 Pro	21H2	-

IV. EXPERIMENT METHODOLOGY

A. Experimental setup

All of the environments were run on the same desktop machine with the following specifications:

- **CPU:** AMD Ryzen-7 3700X (8 core, 16 thread)
- **RAM:** G.Skill Trident-Z 16 GB 3200 MHz
- **HDD:** WD Blue 5200 RPM 2TB
- **Motherboard:** ASUS X470-F
- **Cooling solution:** be quiet! Dark Rock Pro 4

We partitioned the 2TB hard disk to allow an approximate of 256 GB per environment. Windows 10 Pro, Debian and Ubuntu were configured in a dual boot setup. Meanwhile, both the virtual machine and Windows Subsystem for Linux environments ran on top of the Windows 10 partition to avoid additional configuration steps. All of the systems were configured using the minimal installation option when provided. Table II show the major OS and kernel version numbers used by each environment.

Other relevant software and their version numbers include:

- **Hyper-V:** Version 10.0.19041.1
- **WSL:** Version 2
- **PBRT:** Version 4 (Early release)
- **GCC:** Version 12.1
- **CMake:** Version 3.23.1
- **Visual Studio:** 2022 17.2 (for building on Windows)

The Ubuntu virtual machine was configured with 8 virtual processors and 8 GB of startup RAM. The rest of the settings were left on their default values as set by the quick create method.

B. Samples and repetitions

The combination of all possible environments, scenes, filters, and accelerators net 80 unique scenarios. Each of these scenarios was executed 5 times to obtain a total of 400 observations which results in a confidence interval of 95%, precision of 5% and 50% variability [11].

C. Measurements

Time in seconds was the only response variable observed for each sample. This value is specifically taken from the render time reported by PBRT's console output after the command is done executing.

D. Procedure

It is important to note that, since our dual-boot operating systems run on the same hardware one at a time, it was not possible to gather the entirety of the samples in a truly random fashion. Instead, these were gathered in blocks per environment in the following order: Windows 10, Ubuntu-VM, Ubuntu, Debian, and Ubuntu-WSL. This order was chosen randomly without any particular considerations.

The procedure in each environment involved executing the same Python script which computed all scene-parameter combinations and called PBRT to render them 5 times each in completely random order. Once the call to PBRT is done, this same script also captures the console output and parses it for the render time.

E. Statistical analysis

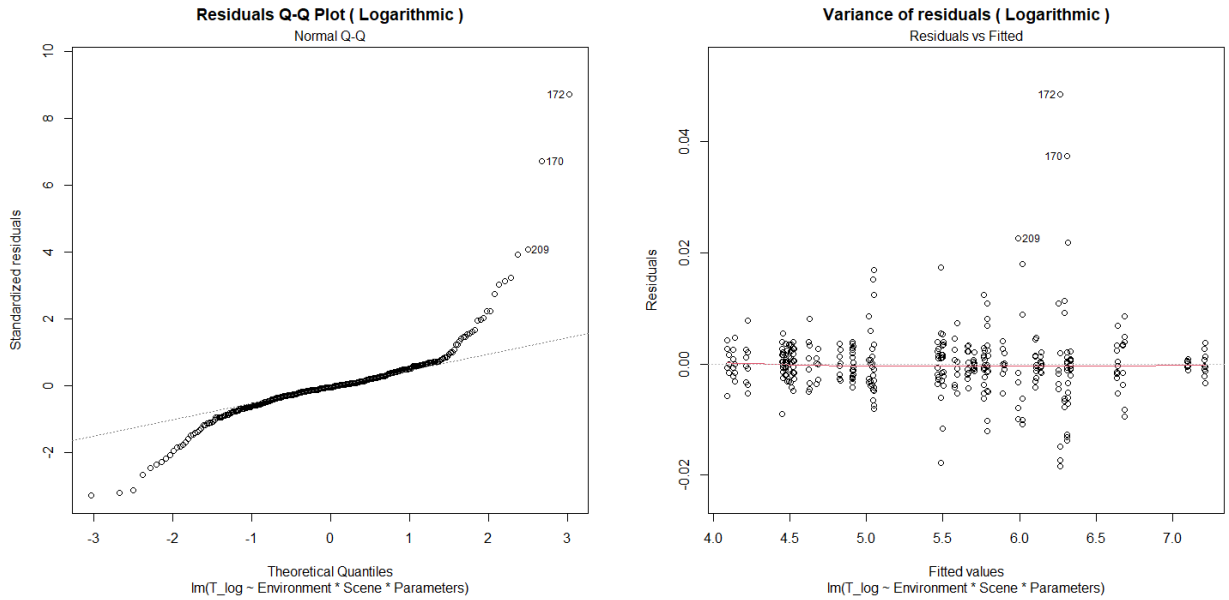
All statistical analysis was done with **R language version 4.1.2** [12] and **RStudio version 2022.02.0 Build 443** [13]. Two-way multivariate analysis of variance (MANOVA) was used to study the effects of the dependent variables and their interactions on the render time response. Pairwise t-tests were also used as a means of post hoc testing to identify particular differences between scenarios (e.g. combinations of environment and scenes, or environment and parameters). Finally, all tests were set to a significance level of 5%.

V. RESULTS

A. ANOVA assumptions

To ensure the validity of the conclusions derived from ANOVA, a small set of visual tests were carried out to ensure that the linear model residuals met the normality and homoscedasticity requirements. These involve the use a Q-Q plot for the former, and a residuals vs fitted values plot for the latter.

A few different transformations were applied to better meet the requirements, starting with a square root transformation followed by a cubic root. These two methods did not yield a satisfactory adjustment and thus are omitted for the sake of brevity. Finally, the logarithmic method resulted in residuals that satisfy both requirements sufficiently. While the Q-Q plot of figure 1a shows some deviation in the tails, ANOVA is known for being particularly robust to violations of normality [14] and as such, the resulting distribution can be considered good enough. As for the homogeneity of variances, the lack of any obvious pattern in the residuals vs fitted plot of figure 1b suggests this requirement is also met.



(a) (b)

Fig. 1. Graphs used in visual evaluation of ANOVA assumptions. A logarithmic transformation had to be applied since the original data does not satisfy the normality and homoscedasticity requirements. Plot **(a)** indicates an acceptable normality, while **(b)** indicates homogeneity of variances.

TABLE III
P-VALUES OBTAINED FROM PAIRWISE COMPARISONS OF ENVIRONMENTS. BOLD LETTERS INDICATE P-VALUES > 0.05.

	Environment			
	Ubuntu	WSL	VM	Debian
Windows 10	< 0.001	< 0.001	0.027	< 0.001
Ubuntu	-	**0.426**	< 0.001	**0.966**

B. Effects on the response variable

ANOVA results show that all the involved factors and their interactions have a significant effect on the response variable. Further analysis via post hoc testing was conducted to identify specific differences between groups. The environment factor will be treated as the main variable under study as it represents the different operating systems of our experiment.

C. Comparison of environments

When comparing the levels of the environment factor the pairwise t-test suggests that there is a statistically significant difference between Windows 10 and all the GNU/Linux environments (table III). Upon visual inspection via boxplots (figure 2), it seems that Windows 10 obtained higher rendering times than its counterparts across the board. Another interesting observation is that there is no significant difference between Ubuntu and Debian (p = 0.966) and Ubuntu and WSL (p = 0.426).

D. Comparison of environment-scene interactions

The p-values in table IV once again show a significant difference between Windows and all the other environments and

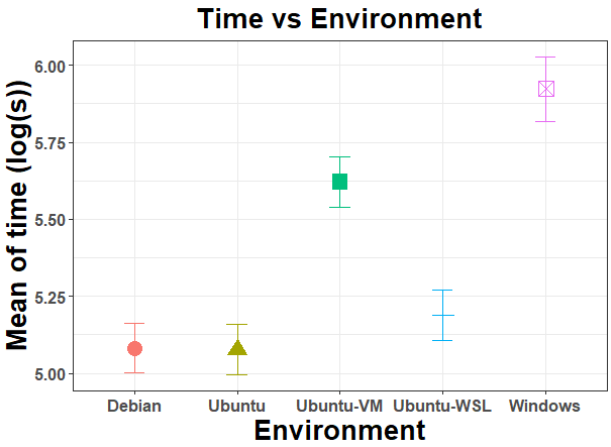


Fig. 2. Mean of log-transformed rendering times for each environment. Error bars indicate the standard error.

TABLE IV
P-VALUES OBTAINED FROM PAIRWISE COMPARISONS OF ENVIRONMENT-SCENE PAIRS. BOLD LETTERS INDICATE P-VALUES > 0.05.

		Environment			
	Scene	Ubuntu	WSL	VM	Debian
Windows 10	Clouds	$< .001$	$< .001$	0.046	$< .001$
	Orb	$< .001$	$< .001$	$< .001$	$< .001$
	Killeroo	$< .001$	$< .001$	$< .001$	$< .001$
	Transparent	$< .001$	$< .001$	$< .001$	$< .001$
Ubuntu	Clouds	-	**0.29**	$< .001$	**0.77**
	Orb	-	0.019	$< .001$	**0.69**
	Killeroo	-	**0.12**	$< .001$	**0.88**
	Transparent	-	**0.067**	$< .001$	**0.6**

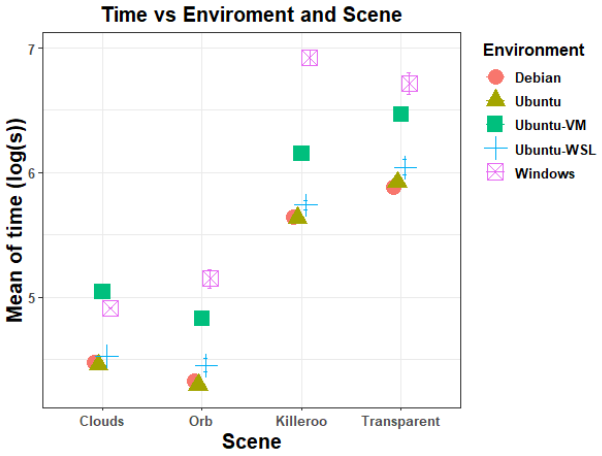


Fig. 3. Mean of log-transformed rendering times for each environment and scene. Error bars indicate the standard error.

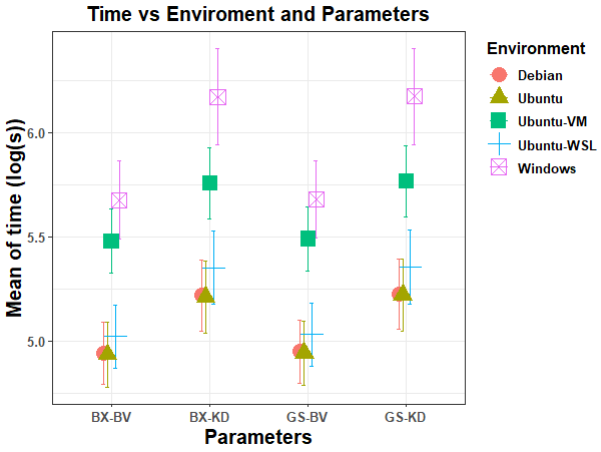


Fig. 4. Mean of log-transformed rendering times for each environment and set of parameters. Error bars indicate the standard error.

TABLE V
P-VALUES OBTAINED FROM PAIRWISE COMPARISONS OF ENVIRONMENT-PARAMETER PAIRS. BOLD LETTERS INDICATE P-VALUES > 0.05.

		Environment			
	Parameters	Ubuntu	WSL	VM	Debian
Windows 10	BX-BV	0.013	0.03	**0.57**	0.013
	BX-KD	0.0012	0.006	**0.19**	0.0012
	GS-BV	0.013	0.03	**0.57**	0.0013
	GS-KD	0.0012	0.006	**0.19**	0.0012
Ubuntu	BX-BV	-	**0.82**	**0.084**	**0.99**
	BX-KD	-	**0.71**	**0.084**	**0.99**
	GS-BV	-	**0.82**	**0.084**	**0.99**
	GS-KD	-	**0.71**	**0.084**	**0.99**

no significant difference between Ubuntu, Debian, and WSL when comparing on a per-scene level. One small exception to this is the orb scene when comparing Ubuntu and WSL, in which there is a significant difference ($p = 0.019$). The boxplot in figure 3 also repeats the pattern of Windows performing worse than the other environments. The only exception to this is the clouds scene, in which the Ubuntu virtual machine takes longer on average to render.

E. Comparison of environment-parameter interactions

When comparing environments across parameter sets, the t-tests (table V) indicate a significant difference between Windows 10 and Ubuntu, Debian, and WSL. However, there doesn't seem to be a difference with regards to Ubuntu-VM. This could serve to explain why both of these environments tend to be closer in performance and why sometimes (e.g. clouds, figure 3) they trade places with each other. The boxplot of figure 4 also serves to further reinforce the ranking of performance that has been observed so far. That is, Ubuntu, Debian, and Ubuntu-WSL tied for first place, followed by Ubuntu-VM and Windows.

VI. DISCUSSION

The results shown in the previous section allowed us to synthesize a series of findings regarding the differences between the tested environments. The following section lists and expands on said findings in order to better explain what could potentially have led to the observed trends in the results.

Windows is slower at rendering in almost all cases: With the exception of one very specific scenario, rendering on Windows 10 Pro took a significantly higher amount of time than its GNU/Linux peers. One possible explanation for this is the lack of an option to perform a minimal installation in Windows 10's official installer. On the other hand Ubuntu and Debian did offer this option, and as such, could have been at an advantage with less software competing for resources. As for Ubuntu WSL, while there is no minimal option, the number of non-essential software is inherently reduced since this environment is primarily meant for technical tasks rather than as a full-blown consumer operating system with all the extras and pluses that entails.

Debian and Ubuntu achieved a similar performance: The purpose of comparing these two common GNU/Linux distributions was to see if there is any advantage in using one system over another. All of the results strongly suggest that this is not the case as there were no significant differences between their rendering times. This is not surprising given that the two distributions share a number of similarities starting with the fact that Ubuntu is a derivative of Debian. Additionally, the variants of these systems used in our experimental setup both use the GNOME desktop environment.

Rendering performance on Windows Subsystem for Linux is comparable to the native option: One of the more surprising findings among our results is that the WSL variant of Ubuntu achieved a similar performance to its native counterpart in almost all cases. We speculate that this might caused by a number different reasons, starting with the

fact that WSL runs as a command-line only system without any graphical desktop environment. This, combined with the reduced "bloatware" aspect, means that its resource usage should be lower than when running Windows or a standard virtual machine. Additionally, while WSL is also a virtual machine [9], it is important to remember that it is engineered, built and optimized for running a specific set of GNU/Linux distributions. This gives it an advantage over the standard Ubuntu-VM environment used in our experiment. Finally, the superior performance of the native Ubuntu and Debian environments imply that the Linux kernel by itself is very good at managing the CPU-bound loads of PBRT. As such, this good management is better exploited by an optimized environment like WSL.

Ubuntu on a virtual machine affords a worse performance than running natively: On the other hand, and more expectedly, the VM environment performed worse in almost all cases than the native Ubuntu option. This could be largely explained by the fact that the VM was configured with less resources than were actually available so as to not starve the host OS and risk performance degradation. This will be further discussed in the limitations section down below.

WSL could be the better option for running GNU/Linux on a Windows host: Given the significant difference between the VM and native environments, and on the other hand, lack of a difference between WSL and Ubuntu, for now it seems that WSL is the better option when performing CPU-intensive tasks. Once again, it is important to remember that the virtual machine was constrained in its resources and as such might be at an unfair disadvantage. Further discussion is provided below.

VII. Conclusions

A. Limitations and Future Work

GPU rendering: Although this paper focused mostly on stressing the CPU via ray tracing operations, it would have also been interesting to study the differences when running workloads on a GPU as an additional factor. One of the main reasons for leaving this out was the fact that we lacked the hardware requirements to enable GPU passthrough on Hyper-V [15]. Additionally, the WSL version of the CUDA toolkit used by PBRT lacks support for OpenGL interoperability at the time of writing this paper [16]. Without these key features, PBRT is unable to render on GPU on the VM and WSL environments.

Compare against lightweight distributions: Certain GNU/Linux distributions like Lubuntu or Mint Xfce come installed with minimal sets of packages and resource-economic graphical desktop environments. Involving these types of systems in a future study might prove useful in verifying how much further the Linux kernel's efficiency can be taken in what could be called a lighter environment.

VM environment was heavily constrained by the hardware: While the hardware on our testing machine tends to be sufficient for a single system, it is not enough to run a full virtual machine comfortably on top of a host OS. Testing on server-grade hardware might allow an improvement in the performance of the VM environment. Additionally, we consider that the many configuration options offered by Hyper-V could also serve to optimize said environment beyond the few basic ones that were tweaked.

Measure CPU and memory usage in a future experiment: While time can be a good indicator of performance, it is not the only dimension that is measurable. CPU and memory usage could also indicate how well an OS manages resources under heavy load. Thankfully, PBRT's console options allow the logging of said values to disk, so tracking these measurements in a future study should be possible.

B. Final statements

We set out to study the impact on rendering performance using PBRT on Windows and a number of different GNU/Linux-based environments as a means to compare the CPU management capabilities of each OS. Our results show that Windows 10 performs significantly worse at physically based rendering than all the other tested systems. Additionally, we found that performance on Windows Subsystem for Linux is competitive with Ubuntu and Debian systems running natively on hardware, while the virtual machine environment performed worse than the other GNU/Linux systems but better than Windows 10. These findings suggest that the Linux kernel is superior at dealing with CPU-intensive workloads like the ones generated by PBRT and that WSL is a fairly good virtualization option when running a Windows host.

References

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating systems concepts*, 9th ed. 2012, ISBN: 978-1-118-06333-0.

[2] K. Salah and A. Kahtani, "Performance evaluation comparison of snort nids under linux and windows server," *Journal of Network and Computer Applications*, vol. 33, pp. 6–15, 1 Jan. 2010, ISSN: 10848045. DOI: 10.1016/j.jnca.2009.07.005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1084804509001040.

[3] W.-C. Fan, C.-S. Wong, W.-K. Lee, and S.-O. Hwang, "Comparison of interactivity performance of linux cfs and windows 10 cpu schedulers," in *2020 International Conference on Green and Human Information Technology (ICGHIT)*, 2020, pp. 31–34. DOI: 10.1109/ICGHIT49656.2020.00014.

[4] J. Bellows, "Comparing linux operating systems for the raspberry pi 2," Winona State University, Apr. 2016, pp. 8–13. [Online]. Available: https://cs.winona.edu/cs-website/current_students/Projects/CSConference/2016conference.pdf#page=10.

[5] Microsoft, *What is windows subsystem for linux*, 2022. [Online]. Available: https://docs.microsoft.com/en-us/windows/wsl/about.

[6] A. Abdulganiyu, "Comparative analysis of real-time operating system (rtos) of some selected os using external signal generator and oscilloscope," *International Journal of Science and Engineering Investigations*, vol. 6, pp. 47–53, 63 Apr. 2017, ISSN: 2251-8843. [Online]. Available: www.IJSEI.com.

[7] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015, pp. 171–172. DOI: 10.1109/ISPASS.2015.7095802.

[8] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: From theory to implementation: Third edition*, 3rd. Elsevier Inc., Sep. 2016, pp. 1–1233, ISBN: 9780128006450.

[9] Microsoft, *Comparing wsl 1 and wsl 2*, Apr. 2022. [Online]. Available: https://docs.microsoft.com/en-us/windows/wsl/compare-versions#wsl-2-architecture.

[10] M. Pharr, *Pbrt-v4 scenes*, 2022. [Online]. Available: https://github.com/mmp/pbrt-v4-scenes.

[11] G. Israel, "Determining sample size," Institute of Food and Agricultural Sciences (IFAS), University of Florida, 1992.

[12] R Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2021. [Online]. Available: https://www.R-project.org/.

[13] RStudio Team, *Rstudio: Integrated development environment for r*, RStudio, PBC, Boston, MA, 2022. [Online]. Available: http://www.rstudio.com/.

[14] P. K. Ito, *7 robustness of anova and manova test procedures*, Jan. 1980. DOI: 10.1016/S0169-7161(80)01009-7.

[15] Microsoft, *Plan for deploying devices using discrete device assignment*, Apr. 2022. [Online]. Available: https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/plan/plan-for-deploying-devices-using-discrete-device-assignment#machine-profile-script.

[16] NVIDIA, *Cuda on wsl*, 2022. [Online]. Available: https://docs.nvidia.com/cuda/wsl-user-guide/index.html#features-not-yet-supported.