Comparison of Interactivity Performance of Linux CFS and Windows 10 CPU schedulers

Wei-Cong Fan, Chee-Siang Wong
Faculty of Information & Communication Technology
Universiti Tunku Abdul Rahman
Kampar, Perak, Malaysia
wcfan0306@1utar.my , wongcs@utar.edu.my

Wai-Kong Lee, Seong-Oun Hwang
Department of Software and Communications Engineering
Hongik University
Sejong, South Korea
waikong.lee@gmail.com, sohwang@hongik.ac.kr

***Abstract* — One of the main goals of a user-oriented operating system design is to ensure low response time of interactive tasks. Otherwise, high scheduling latencies will become obvious to the user and will in turn affect the user experience. One of the most popular benchmarks that can measure interactive tasks response time is Interbench. However, Interbench was only available in Linux, thus making interactivity performance benchmarking for other operating systems impossible. This research ported the Interbench to Windows operating system so that the interactive performance of Windows and Linux can be evaluated and compared. We studied the effect of various simulated workloads on interactive performance of both operating systems and concluded that the Linux CPU scheduler tends to have lower latencies than Windows 10 in most scenarios, except when a heavy background load is executed concurrently with heavy load interactive task.**

Keywords-component : Linux; Windows; Interactivity; Interbench

I. Introduction

Operating system (OS) is a software that acts as an interface between programs and computer hardware [1]. It is comprised of a fundamental component, known as kernel. At any time, there are more than one task in an OS requesting to execute. Thus, the interrupt handler, process manager, and Central Processing Unit (CPU) scheduler are designed to handle these requests and to ensure fair tasks execution. At the same time, the CPU scheduling algorithm must be efficient so that it could make scheduling decisions at the shortest time possible.

Most CPU scheduler benchmarking programs are focused on measuring the system performance in terms of non-interactive tasks' throughput [2]. These benchmarking programs are not suitable to measure the interactive performance of an OS because their goal is to measure how many instructions/tasks can be executed over time. In contrast, user-oriented systems prioritize on interactive performance, where low response time is crucial to ensure good user experience. Response time is defined as the time from the submission of a request until the response is received [3]. It is the required time for a task to switch from Ready State to Running State. Ready State is when a task requests for access to the CPU to execute instructions. The task will be enqueued to a data structure and will wait it is chosen by the CPU scheduler to execute on the CPU. The task will then be switched to the Running State.

One of the most popular benchmarks that can measure interactive performance is Interbench for Linux [4]. The main goal of this benchmark program is to measure the latencies and jitters that exist in the Linux kernel schedulers under different simulated conditions, called interactivity. In other words, it is to determine the response time of an interactive task. When a great amount of latency exists within the scheduling process, lag will become obvious to users, causing the users to have an unpleasant experience when they are doing some tasks on their computer systems.

However, Interbench is only available for Linux OS. Other major OS such as the Mircosoft Windows and Apple MacOS, which are designed for user-oriented applications, are not supported. Thus, this paper aims to port the Interbench to the Windows OS so that we can compare the interactivity performance among Linux (CFS) and Windows 10. The contribution of this paper is to evaluate the interactive performance of the Windows 10 under various simulated workloads, which to the author's knowledge, has never been done before.

II. Related Work

Previous works have been carried out to determine the interactivity performance and fairness of schedulers included in different Linux kernel versions [5]. O(1) scheduler which was introduced since Linux kernel version 2.6 until version 2.6.22, was compared with its successor, namely the Completely Fair Scheduler (CFS). The work concluded that the CFS is better than O(1) kernel scheduler in terms of efficiency and fairness of CPU bandwidth distribution without sacrificing its interactivity performance.

Besides, much work has been done to figure out the latencies that exist in schedulers included in Linux and FreeBSD kernels with a focus on its application in real-time systems [6]. Involved schedulers are O(1), CFS, 4BSD and ULE; all of them are UNIX-like OS. A new benchmark named PI-ping was developed to compare the four involved schedulers as inaccurate benchmark results were produced under several conditions. According to the end results, CFS was proven to have an advantage in both applications in

978-1-7281-6295-9/20/$31.00 ©2020 IEEE
DOI 10.1109/ICGHIT49656.2020.00014

servers and embedded systems, which requires high throughput or number of tasks completed within a specified time and low latencies respectively.

Both of the mentioned work done are able to give a clear idea regarding Linux kernel schedulers' interactivity performance. However, the comparison of various types of kernel schedulers is limited by the original Interbench as it was originally intended to compare the schedulers in Linux. The same set of tests was not available for comparison and evaluation purposes in other operating systems such as Windows. [7] proposed VNCplay that records a user's interactive session with a system and replays it multiple times under different system configurations. The response time is then measured by comparing the time at which similar screen updates occur in each of the replayed sessions. However, this method is not practical for modern systems usage because the workloads used are too light as it does not utilize any background load. Also, the libraries used are obsolete.

III. Interactivity Test And Results

A. *Interactivity Test*

Through Interbench, multiple different interactive tasks and background loads can be simulated. Each interactive task will be executed concurrently with a background load. The details for the interactive tasks and background loads are shown in TABLE I and TABLE II below respectively.

TABLE I. Interactive Tasks Details

Interactive Task	Description
Audio	Executes at an interval of 50ms and utilizes 5% of CPU.
Video	Utilizes 40% of CPU and attempts to receive from the CPU 60 times per second.
X-window	Utilizes 0% to 100% of CPU.
Gaming	Utilizes as much CPU as possible.

TABLE II. Background Loads details

Background Load	Description
None	Idle system.
Video	Utilizes 40% of CPU and attempts to receive from the CPU 60 times per second. Load generated by video simulation thread.
X-window	Utilizes 0% to 100% of CPU. Load generated by X-window simulation thread.
Burn	Number of threads fully CPU bound that can be configured.

B. *General workflow of Interbench*

In order to reproduce a fixed percentage of CPU utilization in every benchmark run, the system will be benchmarked by Interbench first by invoking a function called `calibrate_loop` to determine the number of meaningless loops that can be executed by the system within one millisecond. Then the value will be written into a file for the subsequent benchmark runs. For the simulation of interactive tasks and background loads, `run_loadchild` and `run_benchchild` are responsible to control the thread such as starting and stopping a thread. Lastly, `timekeeping_thread` is responsible to keep track of the time taken for an interactive task to be scheduled. Figure 1 shows the general flow of the original and ported Interbench. Due to the incompatibility of system-related libraries, the semaphore implementation, threads creation and timing mechanism need to be modified on the Windows port.

Each pair combination of interactive tasks and background loads will be carried out repeatedly for 30 times. Average and maximum scheduling latency will be recorded. In order to deliver a consistent measurement, the interactive tasks and background loads are executed on the same processor with Intel Turbo Boost disabled.

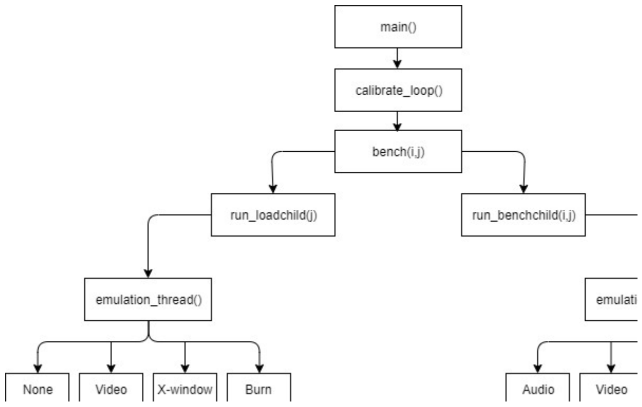

Figure 1. General workflow of Interbench

C. *Benchmark Results*

The average latencies for Audio on Linux and Windows are similar under None and Video background loads, as shown in Figure 2. However, under X-window and Burn background loads, a higher average latency was recorded on Windows compared to Linux. Figure 3 shows that Windows produced higher maximum latency than Linux in Audio interactive tasks under all background loads tested.

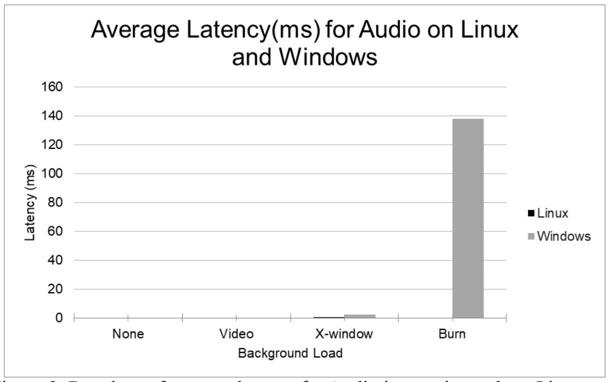

Figure 2. Bar chart of average latency for Audio interactive task on Linux and Windows under different background loads

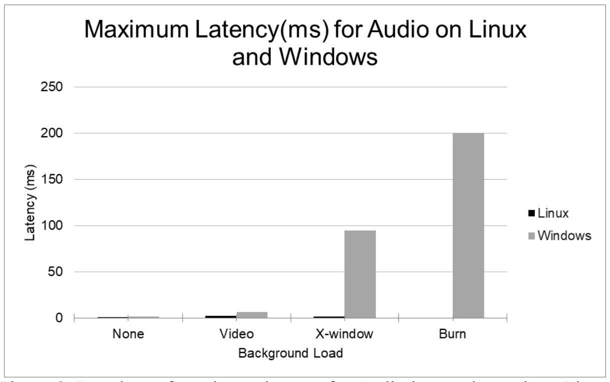

Figure 3. Bar chart of maximum latency for Audio interactive task on Linux and Windows under different background loads

As for Video interactive task shown in Figure 4 and Figure 5, Windows shows higher average latency than Linux in every background load. Both Linux and Windows produce similar maximum latency when the system is idle, but when the systems are under X-window and Burn load, the maximum latency in Windows is higher than in Linux.

In handling X-window interactive task shown in Figure 6 and Figure 7, under None and Video background load representing low and medium load respectively, both operating systems behave similarly by producing insignificant amount of average latency. However, under high amount of background load simulated by Burn, Windows is able to produce a lower average latency compared to Linux. For maximum latency, Windows shows a slightly higher maximum latency than Linux under low and medium background load. But when the interactive task is simulated under high amount of load, a lower maximum latency is produced by Windows.

Based on the results, although Windows performed slightly worse than Linux for X-window in None and Video background loads by producing greater amount of average latency and maximum latency, it can perform much better than Linux for the same interactive task under high amount of background load such as Burn.

Lastly, the average latency and maximum latency for Gaming interactive tasks are represented by Figure 8 and Figure 9 respectively. For average latency, both Linux and Windows produced similar results with nearly zero average latency under None and Video background loads. When X-window background load is simulated, a higher average latency is produced by Windows compared to Linux. However, in conditions with high background load, Windows performed better in terms of interactivity by producing a lower amount of average latency. For maximum latency, an insignificant maximum latency is shown by both operating systems, but Windows produce a slightly higher maximum latency. Just like for average latency for this interactive task under X-window background load, Linux's maximum latency is still lower compared to Windows. Under Burn background load, Windows is still able to maintain its advantage with lower maximum latency.

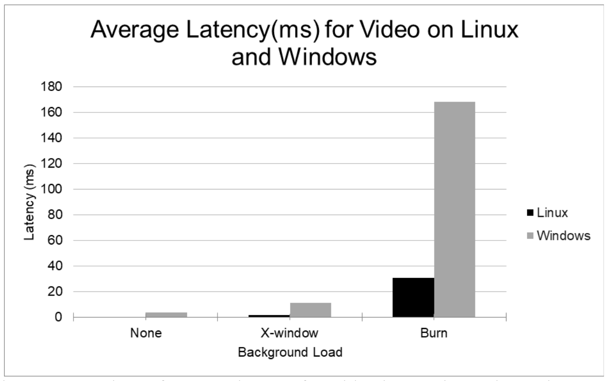

Figure 4. Bar chart of average latency for Video interactive task on Linux and Windows under different background loads

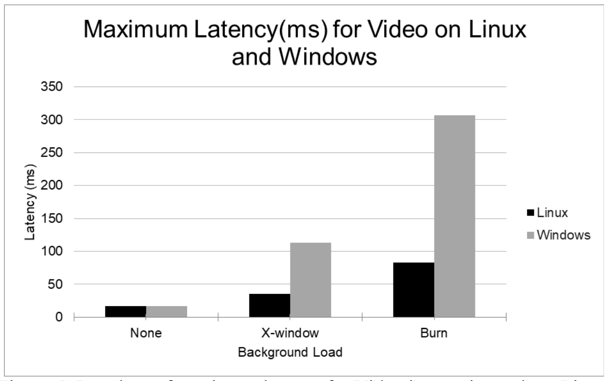

Figure 5. Bar chart of maximum latency for Video interactive task on Linux and Windows under different background loads

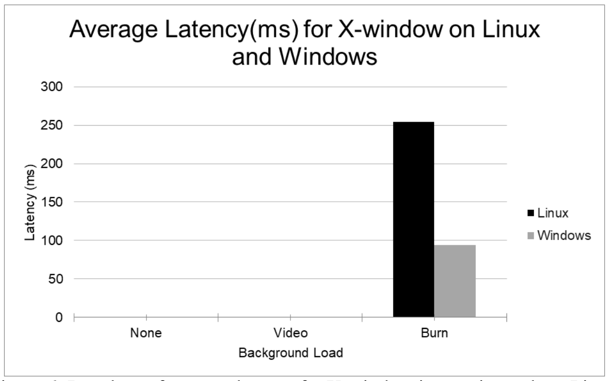

Figure 6. Bar chart of average latency for X-window interactive task on Linux and Windows under different background loads

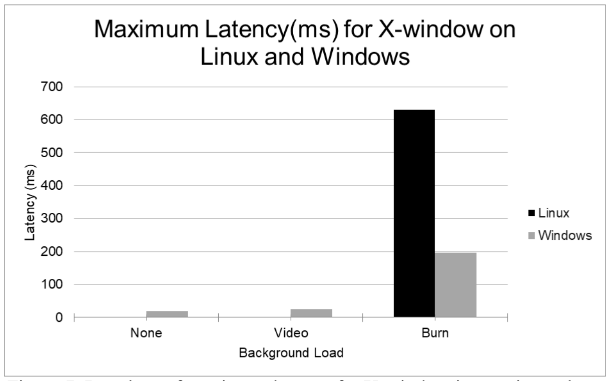

Figure 7. Bar chart of maximum latency for X-window interactive task on Linux and Windows under different background loads

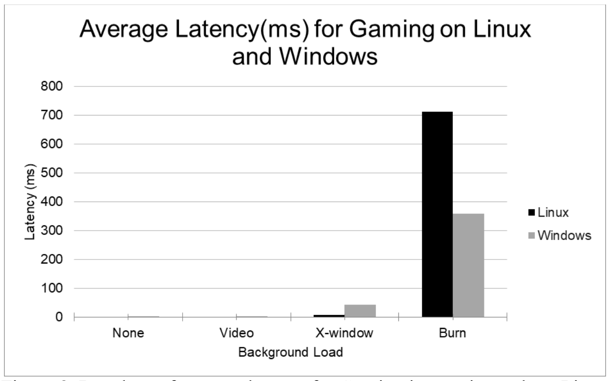

Figure 8. Bar chart of average latency for Gaming interactive task on Linux and Windows under different background loads

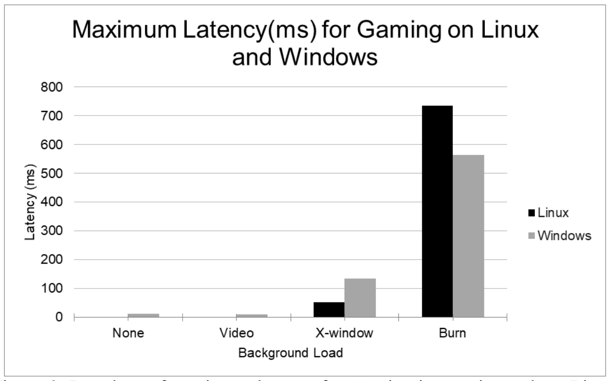

Figure 9. Bar chart of maximum latency for Gaming interactive task on Linux and Windows under different background loads

CONCLUSION

In conclusion, the Interbench is ported to Windows 10 and the benchmark is carried out to compare the interactive performance of Windows 10 and Linux. The Linux kernel scheduler has better interactivity performance in most scenarios, especially in light interactive tasks such as Audio and Video. However, for X-Window and Gaming interactive tasks, the Windows 10 has lower latencies in the presence of Burn, which is a high load background task. For future work, we may want to investigate the reason that causes such performance characteristics for both OS.

REFERENCES

[1] A. Silberschatz, P.B. Galvin, and G. Gagne, *Operating System Concepts*. New York: John Wiley & Sons Inc, 2018, p. 3.

[2] Phoronix Test Suite, https://www.phoronix-test-suite.com/

[3] W. Stallings, *Operating Systems, Internals and Design Principles*. New Jersey: Pearson Prentice Hall. 2018, p. 430.

[4] C. Kolivas, “The homepage of Interbench The Linux interactivity benchmark,” http://www.users.on.net/~ckolivas/interbench/, July 2019

[5] C.S. Wong, I.K.T. Tan, J.W. Lam and R.D. Kumari, “Fairness and interactive performance of O(1) and CFS Linux kernel schedulers,” 2008 International Symposium on Information Technology, Kuala Lumpur, 2008, pp. 1-8.

[6] J. Abaffy and T. Krajcovic, “Latencies in Linux and FreeBSD kernels with different schedulers – O(1), CFS, 4BSD, ULE,” unpublished.

[7] N. Zeldovich and R. Chandra, “Interactive Performance Measurement with VNCplay,” Proceedings of the FREENIX Track: 2005 USENIX Annual Technical Conference, Anaheim, CA, 2005.